FEC Framework
Internet-Draft
Intended status: Standards Track
Expires: March 4, 2011

S. Galanos
O. Peck
RADVISION
V. Roca
INRIA

August 31, 2010

RTP Payload Format for Reed Solomon FEC
draft-galanos-fecframe-rtp-reedsolomon-02

Abstract

This document defines an RTP payload format for the Forward Error Correction (FEC) that uses Reed-Solomon codes. The format defined by this document enables the protection of source media encapsulated in RTP with one or more repair flows and is based on the FEC framework and the SDP Elements for FEC Framework. The Reed-Solomon codes used in this document belong to the class of Maximum Distance Separable (MDS) codes which means they offer optimal protection against random and bursty packet losses.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 4, 2011.

Copyright Notice

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

This document defines new RTP payload formats for the Forward Error Correction (FEC) that is generated by the Reed-Solomon code.

By nature, interactive Real-time applications are extremely sensitive to delay and require very low latency. As a result, retransmission of lost packets and using other closed-loop schemes are not valid options while the use of Forward Error Correction (FEC) is an effective approach.

A primary requirement from FEC for real time applications is the ability to correctly recover from both random and bursty packet losses. The Reed-Solomon FEC codes used in this document belong to the class of Maximum Distance Separable (MDS) codes that are optimal in terms of erasure recovery capability for both situations.

The format defined by this document enables the protection of media source flow with one or more repair flows without adding additional information to the source packets. Such behavior reduces the delay presented by any FEC scheme and maintains backwards compatibility with non FEC-enabled receivers.

Number of previous drafts were composed to draw different FEC schemes suitable for different applications. The scheme defined in this draft is designed to compensate a burst of packet loss over RTP networks with minimum delay, which is needed in interactive IP-based applications such as video conferencing.

The method described in this document is generic to all media types and provides the sender with the flexibility of deciding if FEC protection is required and if so, how many protecting packets and how many source packets to use in a block according to network conditions. Furthermore it allows applying unequal error protection that provides different level of protection to different packets. For example, it can be combined with Scalable Video Coding to protect only the base layer packets of the video flow. At the receiver, both the FEC and original media are received. If no media packets are lost, the FEC packets can be ignored. In the event of a loss, the FEC packets can be combined with other received media to recover all or part of the missing media packets.

The Reed-Solomon codes used in this document have been specified in [RFC5510] and are compatible with Luigi Rizzo codec (see [Rizzo97]). This document is compliant with the Forward Error Correction (FEC) Framework (described in [I-D.ietf-fecframe-framework]) and SDP Elements for FEC Framework (described in [I-D.ietf-fecframe-sdp-elements] [RFC4566]). This draft completes

[I-D.roca-fecframe-rs] by defining Reed-Solomon usage for RTP transport ([RFC3550]) and specifies the appropriate media types (see [RFC4288] [RFC3555]).

2. Requirements Notation

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

3. Definitions, Notations and Abbreviations

This document uses the following definitions and notations. For further definitions that apply to FEC Framework in general, see [I-D.ietf-fecframe-framework].

3.1. Definitions

This document uses the following terms and definitions. Some of them are FEC scheme specific and are in line with [RFC5052]:

Source symbol: unit of data used during the encoding process.

Encoding symbol: unit of data generated by the encoding process. With systematic codes, source symbols are part of the encoding symbols.

Repair symbol: encoding symbol that is not a source symbol.

Code rate: the k/n ratio, i.e., the ratio between the number of source symbols and the number of encoding symbols. By definition, the code rate is such that: 0 < code rate <= 1. A code rate close to 1 indicates that a small number of repair symbols have been produced during the encoding process.

Systematic code: FEC code in which the source symbols are part of the encoding symbols. The Reed-Solomon codes introduced in this document are systematic.

Source block: a block of k source symbols that are considered together for the encoding.

Packet Erasure Channel: a communication path where packets are either dropped (e.g., by a congested router, or because the number of transmission errors exceeds the correction capabilities of the physical layer codes) or received. When a packet is received, it

is assumed that this packet is not corrupted.

Some of them are FECFRAME framework specific and are in line with [I-D.ietf-fecframe-framework]:

Application Data Unit (ADU): a unit of data coming from (sender) or given to (receiver) the media delivery application. In this document, an ADU MUST use an RTP encapsulation.

(Source) ADU Flow: a flow of ADUs from a media delivery application and to which FEC protection is applied. In this document, there MUST be a single ADU flow per FECFRAME framework instance.

ADU Block: a set of ADUs that are considered together by the FECFRAME instance for the purpose of the FEC scheme.

FEC Framework Configuration Information: the FEC scheme specific information that enables the synchronization of the FECFRAME sender and receiver instances.

FEC Source Packet: an RTP data packet submitted to (sender) or received from (receiver) the transport protocol. In this document, FEC Source Packets and ADU MUST be the same (e.g., for backward compability purposes).

FEC Repair Packet: an RTP repair packet submitted to (sender) or received from (receiver) the transport protocol. It contains a repair symbol along with its Explicit Repair FEC Payload ID.

3.2. Notations

This document uses the following notations: Some of them are FEC scheme specific:

k denotes the number of source symbols in a source block.

max_k denotes the maximum number of source symbols for any source block.

n_r denotes the number of repair symbols generated for a source block.

n denotes the number of encoding symbols generated for a source block. Therefore: n = k + n_r.

max_n denotes the maximum number of encoding symbols generated for any source block.

S denotes the encoding symbol length in units of m-bit elements. When m = 8, then S and E are equal.

GF(q) denotes a finite field (also known as Galois Field) with q elements. We assume that q = 2^^m in this document.

m defines the length of the elements in the finite field, in bits. In this document, m belongs to {2..16}.

q defines the number of elements in the finite field. We have: q = 2^^m in this specification.

CR denotes the "code rate", i.e., the k/n ratio.

a^^b denotes a raised to the power b.

3.3. Abbreviations

This document uses the following abbreviations:

ADU stands for Application Data Unit.

ESI stands for Encoding Symbol ID.

FEC stands for Forward Error Correction code.

FFCI stands for FEC Framework Configuration Information.

RS stands for Reed-Solomon.

MDS stands for Maximum Distance Separable code.

4. Reed Solomon Codes

The detailed operation and theory behind Reed Solomon codes is out of the scope of this document. In general a Reed Solomon code takes a group of k source symbols and generates n - k repair symbols. A receiver needs to receive any k of the n source or repair symbols in order to recover the remaining n-k symbols. As explained in RFC 5510, the Reed-Solomon algorithm operates over multiple elements each taken from a single source symbol. Symbols are composed of S "m-bit elements" where m is the Galois Field exponent GF(2^m). In the usual case of GF(2^8), elements are bytes, and the size S in terms of elements is of course equal to the symbol size in bytes. The symbol

size can be different in different implementations. Any symbol size can be used in the format offered by this document. However, it is recommended in terms of implementation simplicity to use 8-bits elements. For more information on Reed Solomon codes, the reader is referred to [Rizzo97].

5. Source Block Creation

This draft defines the protection of an RTP source flow using one or more FEC repair flows.

A source block for the Reed-Solomon code contains k source symbols. In the scheme presented by this document, each source symbol contains a single Application Data Unit (ADU, as defined in [I-D.ietf-fecframe-framework]), which is in our case an RTP packet. Therefore a source block contains exactly k RTP packets. The Reed-Solomon code generates n_r = n - k repair symbols that are transmitted using n_r = n - k FEC repair packets. Each FEC repair packet contains a single repair block.

To create a source block the steps outlined below should be followed:

1. Determine the largest RTP packet size (in bytes) of the source block. During this computation, both the RTP header and payload are considered

2. For each ADU of this source block, create a byte array (of size 2 + this largest RTP packet size), as follows:

   A. In the first two bytes, place the unsigned network-ordered 16-bit representation of the RTP packet size in bytes (including RTP header size and payload size)

   B. Append the entire RTP packet including its RTP header

   C. Add zero padding so that the byte array is the size of the largest packet protected by this source block plus two (to consider the initial two bytes). Therefore, the largest packet does not contain padding.

3. Append all the byte arrays one after the other in the following way:

   A. The packets are in an increasing order of the sequence number as it appears in the RTP packet header taking wraparound into account

Figure 1 demonstrates how a source block is created from 4 packets (P1, P2, P3, P4) with different sizes. The largest packet protected in this source block has a size of 5 (L = 5) and therefore P1 and P3 are both padded with zeros to this size. The source block contains the RTP packet size before each packet. (Note that this example is not a binary representation of the source block. The Packet size spans over two bytes as stated above)

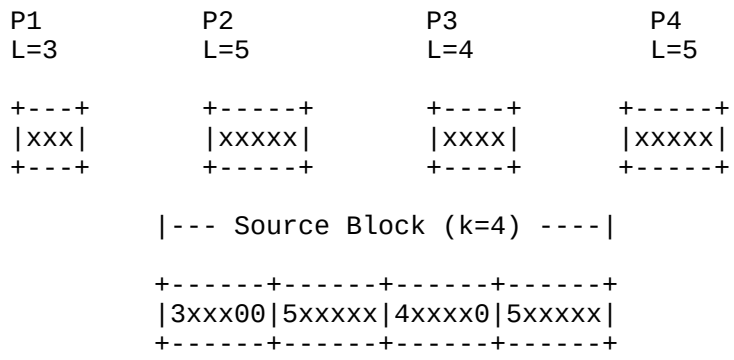

Figure 1: Structure of a Source Block

The FEC Reed-Solomon Scheme gets a source block created from k packets and generates n-k FEC repair packets that protect the entire source block. These packets are then transmitted in the repair flow. Note that source packets padding is done only for FEC packet calculation and the original payloads are transmitted without extra padding.

6. Packet Formats

This section defines the formats of the source and repair packets

6.1. FEC Source Packets

The FEC Framework requires that FEC source packets contain information identifying the source block and the position within the source block occupied by the packet. However, in order to maintain backwards compatibility, the scheme defined by this document enables the receiver to get this information without appending additional information to the source packet. Specifically this information is obtained using the combination of sequence number found in the RTP header and information provided in the FEC header of each FEC repair packet. Such behavior enables both non-FEC-capable and FEC-capable receivers to receive and interpret the same source packets sent in a

multicast session.

6.2. FEC Repair Packets

The FEC repair packets contain information that enables the receiver to reconstruct the source block in the remote end. This is done by using the RTP header of the FEC repair packets as well as another dedicated header that is placed within the RTP payload. This header, referred to as the FEC header, complies with [I-D.ietf-fecframe-framework] (section 6.4.1), as shown in Figure 2.

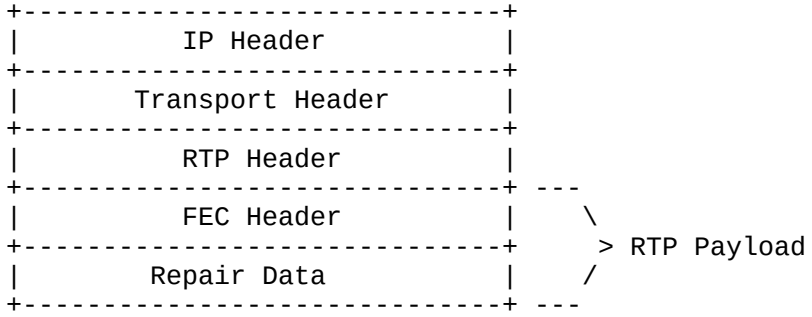

Figure 2: Format of repair packets

6.2.1. RTP header format

The RTP header is formatted according to [RFC3550] with some further clarifications listed below:

- Marker (M) Bit: This bit is not used for this payload type, and is set to 0.

- Payload Type: The (dynamic) payload type for the repair packets is determined through out-of-band means. Note that this document registers new payload formats for the repair packets (Refer to Section 5 for details). According to [RFC3550], an RTP receiver that cannot recognize a payload type must discard it. This provides for backward compatibility. The FEC mechanisms can then be used in a multicast group with mixed FEC-capable and non-FEC-capable receivers. If a non-FEC-capable receiver receives a repair packet, it will not recognize the payload type, and hence, will discard the repair packet. In case more than one repair flow is used, different Payload Types will be used to distinguish between the different flows.

- Sequence Number (SN): The sequence number maintains the standard definition. It is one higher than the sequence number in the previously transmitted repair packet. The initial value of the sequence number is random (unpredictable) [RFC3550].

- Timestamp (TS): The timestamp is set to a time corresponding to the repair packet's transmission time. Note that the timestamp value has no use in the actual FEC protection process and is usually useful for jitter calculations. FEC packets that are the result of the same FEC encoding operation will use the same value as their Timestamp.

- Synchronization Source (SSRC): The SSRC value is randomly assigned as suggested by [RFC3550].

6.2.2. FEC header format

The FEC header includes information that enables the receiver to reconstruct the source block and to identify the FEC repair packets associated with each source block, in their correct order.

The format of the FEC header is shown in figure 3.

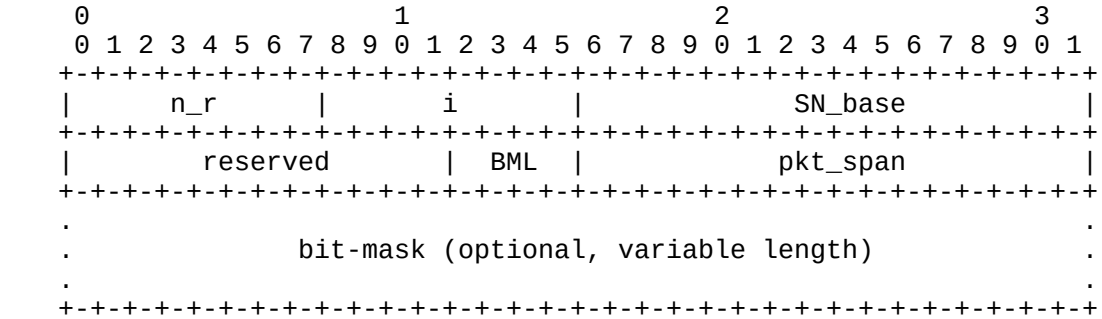

Figure 3: FEC Header Format

The FEC header consists of the following general fields:

- n_r (8-bit field): the number of FEC repair packets used to protect this source block.

- i (8-bit field): the 0-based index in the sequence of n_r FEC repair packets. This index is equal to ESI - k, where ESI is the Encoding Symbol ID of the associated repair symbol.

- SN_base (16-bit field): the lowest RTP sequence number (taking wraparound into account) of the FEC source packets in the

associated source block. This SN_base also identifies the source block. In order to avoid any risk of confusion, two consecutive source blocks MUST use different SN_base values, which is easily verified by the sender (this situation might happen with Reed-Solomon over GF(2^16)).

- o pkt_span (16-bit field): the number of consecutive FEC Source Packets considered. A subset of these FEC Source Packets may be missing, as indicated by the 0 entries of the optional bit-mask.

- o Reserved (12-bit field): reserved for future use. This field MUST be set to zero in this specification.

- o Bit Mask Length, BML (4-bit field): when the pkt_span source packets of the source block don't have consecutive RTP sequence numbers, a bit-mask MUST be used to indicate which packets are protected by this FEC packet. This field indicates the length of the bit-mask in units of 32-bit words, as the following table shows. In any case, only the first pkt_span bits of this bit-field are meaningful, the remaining bits (if any) MUST be set to 0.

- o bit-mask (Optional field, length multiple of 32 bits): When BML is set to a value different than 0000, a bit-mask field is added, whose length in term of number of 32-bit words is indicated by the BML field. The bit-mask indicates which source packets have been considered in the source block ("1" bit valude entry in the bit-field) and which source packets have been ignored ("0" bit valude entry in the bit-field) (usually this happens when a source packet has been erased (lost) before reaching the FECFRAME encoder). The first packet in the bit-mask (corresponding to bit position 0 of the first 32-bit word) corresponds to the source packet whose RTP sequence number is specified in field SN_base.

BML value	bit-mask length (in bits)	bit-mask length (in words)
0	0 (no mask)	0
1	32 bits	1 x 32-bit word
10	64 bits	2 x 32-bit word
11	96 bits	3 x 32-bit word
100	128 bits	4 x 32-bit word
101	160 bits	5 x 32-bit word
110	192 bits	6 x 32-bit word
111	224 bits	7 x 32-bit word
1000	256 bits	8 x 32-bit word
...	...	...
1111	480 bits	15 x 32-bit word

6.2.3. Repair Data Format

The repair data follows the FEC header in the FEC repair packet. It includes the result of the Reed-Solomon encoding over the source block. Note that the first two bytes of the repair data contain the result of the Reed-Solomon encoding over the packet sizes in the source block and that the size of the repair data equals the size of the largest packet protected by this source block plus 2. Therefore, the size of an FEC repair packet (FEC header and data) is larger than the longest source packet. This should be taken under consideration when deciding on the Maximum Transmission Unit (MTU) size used for the source packets.

7. Payload Format Parameters

According to the FEC framework, when RTP is used as a transport for repair packet flows, the scheme must define an RTP Payload Format for the repair data. This section provides the media subtype registration for the Reed-Solomon FEC. The parameters that are required to configure the FEC encoding and decoding operations are also defined in this section.

7.1. Media Type Registration

This registration is done using the template defined in [RFC4288] and following the guidance provided in [RFC3555].

7.1.1. Registration of audio/reed-solomon-fec

Type name: audio

Subtype name: reed-solomon-fec

Required parameters:

- max_n: The upper limit for the sum of source and repair packets that belong to the same FEC block. max_n is a positive integer. The application can change both k and n-k. max_n is the upper limit for n. The value of max_n must be equal to or lower than the codec limitation (2^m).

- repair-window: The time that spans the source packets and the corresponding repair packets. The size of the repair window is specified in microseconds.

- The repair-window impacts the maximum number of source packets in a FEC block at the sender side, and defines the time which the receiver should wait for the repair packets. The repair-window value may be negotiated between the sender and receiver. the details of such negotiation are out-of-scope for this document.

- element-size: a non-negative integer indicating the length of each encoding elements in bits. This value equals to the "m" parameter in the GF (represented by 2^m).

Optional parameters: None.

Encoding considerations: This media type is framed and binary, see section 4.8 in [RFC4288]

Security considerations: Please see security consideration in [I-D.ietf-fecframe-framework]

Interoperability considerations: None.

Published specification: TBD

Applications that use this media type: Multimedia applications that want to improve resiliency against packet loss by sending redundant data in addition to the source media.

Additional information: None.

Magic number(s): none defined

File extension(s): none defined

Macintosh file type code(s): none defined

Person & email address to contact for further information: Sarit Galanos, sarit@radvision.com

Intended usage: COMMON

Restrictions on usage: This media type depends on RTP framing, and hence is only defined for transfer via RTP [RFC3550]. Transport within other framing protocols is not defined at this time.

7.1.2. Registration of video/reed-solomon-fec

Type name: video

Subtype name: reed-solomon-fec

Required parameters:

- max_n: The upper limit for the sum of source and repair packets that belong to the same FEC block. max_n is a positive integer. The application can change both k and n-k. max_n is the upper limit for n. The value of max_n must be equal to or lower than the codec limitation (2^m).

- repair-window: The time that spans the source packets and the corresponding repair packets. The size of the repair window is specified in microseconds.

- The repair-window impacts the maximum number of source packets in a FEC block at the sender side, and defines the time which the receiver should wait for the repair packets. The repair-window value may be negotiated between the sender and receiver. the details of such negotiation are out-of-scope for this document.

- element-size: a non-negative integer indicating the length of each encoding elements in bits. This value equals to the "m" parameter in the GF (represented by 2^m).

Optional parameters: None.

Encoding considerations: This media type is framed and binary, see section 4.8 in [RFC4288]

Security considerations: Please see security consideration in [I-D.ietf-fecframe-framework]

Interoperability considerations: None.

Published specification: TBD

Applications that use this media type: Multimedia applications that want to improve resiliency against packet loss by sending redundant data in addition to the source media.

Additional information: None.

Magic number(s): none defined

File extension(s): none defined

Macintosh file type code(s): none defined

Person & email address to contact for further information: Sarit Galanos, sarit@radvision.com

Intended usage: COMMON

Restrictions on usage: This media type depends on RTP framing, and hence is only defined for transfer via RTP [RFC3550]. Transport within other framing protocols is not defined at this time.

7.1.3. Registration of text/reed-solomon-fec

Type name: text

Subtype name: reed-solomon-fec

Required parameters:

- max_n: The upper limit for the sum of source and repair packets that belong to the same FEC block. max_n is a positive integer. The application can change both k and n-k. max_n is the upper limit for n. The value of max_n must be equal to or lower than the codec limitation (2^m).

- repair-window: The time that spans the source packets and the corresponding repair packets. The size of the repair window is specified in microseconds.

- The repair-window impacts the maximum number of source packets in a FEC block at the sender side, and defines the time which the receiver should wait for the repair packets. The repair-window value may be negotiated between the sender and receiver. the details of such negotiation are out-of-scope for this document.

- element-size: a non-negative integer indicating the length of each encoding elements in bits. This value equals to the "m" parameter in the GF (represented by 2^m).

Optional parameters: None.

Encoding considerations: This media type is framed and binary, see section 4.8 in [RFC4288]

Security considerations: Please see security consideration in [I-D.ietf-fecframe-framework]

Interoperability considerations: None.

Published specification: TBD

Applications that use this media type: Multimedia applications that want to improve resiliency against packet loss by sending redundant data in addition to the source media.

Additional information: None.

Magic number(s): none defined

File extension(s): none defined

Macintosh file type code(s): none defined

Person & email address to contact for further information: Sarit Galanos, sarit@radvision.com

Intended usage: COMMON

Restrictions on usage: This media type depends on RTP framing, and hence is only defined for transfer via RTP [RFC3550]. Transport within other framing protocols is not defined at this time.

7.1.4. Registration of application/reed-solomon-fec

Type name: application

Subtype name: reed-solomon-fec

Required parameters:

- max_n: The upper limit for the sum of source and repair packets that belong to the same FEC block. max_n is a positive integer. The application can change both k and n-k. max_n is the upper

limit for n. The value of max_n must be equal to or lower than the codec limitation (2^m).

- repair-window: The time that spans the source packets and the corresponding repair packets. The size of the repair window is specified in microseconds.

- The repair-window impacts the maximum number of source packets in a FEC block at the sender side, and defines the time which the receiver should wait for the repair packets. The repair-window value may be negotiated between the sender and receiver. the details of such negotiation are out-of-scope for this document.

- element-size: a non-negative integer indicating the length of each encoding elements in bits. This value equals to the "m" parameter in the GF (represented by 2^m).

Optional parameters: None.

Encoding considerations: This media type is framed and binary, see section 4.8 in [RFC4288]

Security considerations: Please see security consideration in [I-D.ietf-fecframe-framework]

Interoperability considerations: None.

Published specification: TBD

Applications that use this media type: Multimedia applications that want to improve resiliency against packet loss by sending redundant data in addition to the source media.

Additional information: None.

Magic number(s): none defined

File extension(s): none defined

Macintosh file type code(s): none defined

Person & email address to contact for further information: Sarit Galanos, sarit@radvision.com

Intended usage: COMMON

Restrictions on usage: This media type depends on RTP framing, and hence is only defined for transfer via RTP [RFC3550]. Transport

within other framing protocols is not defined at this time.

7.2. Mapping of SDP Parameters

For a proper operation details of the FEC scheme have to be communicated between the sender and the receiver. Specifically, the receiver has to know the relationship between the source and the repair flows, how the sender applied protection to the source flow and how the repair flows can be used to recover the lost data. One way to provide this information is to use the Session Description Protocol (SDP) [RFC4566].

The mapping of the media type specification for "reed-solomon-fec" and their parameters in SDP is as follows:

- The media type (e.g., "application") goes into the "m=" line as the media name.
- The media subtype ("reed-solomon-fec") goes into the "a=rtpmap" line as the encoding name.
- The remaining required payload-format-specific parameters ("max_n", "repair-window") go into the "a=fmtp" line by copying them directly from the media type string as a semicolon-separated list of parameter=value pairs.

See section 9 for SDP examples.

8. Protection and Recovery Procedures

This section provides a complete specification of the protection and recovery procedures.

8.1. Overview

The FEC repair packets allow end-systems to recover from a loss of media packets. The following sections specify the steps involved in generating the FEC repair packets and reconstructing the missing source packets from the FEC repair packets.

8.2. FEC Repair Packet Construction

The RTP header of a FEC repair packet is formed based on the guidelines given in Section 6.2.1. The FEC header is formed based on the guidelines given in Figure 3. Before Reed-Solomon encoding, two bytes are prepended to each ADU (RTP source packet in our case) that contain this ADU length in bytes, stored in network-order. FEC

encoding can then take place and the n_r = n - k repair symbols are created. Each repair symbol is then appended to its FEC header.

8.3. Source Packet Reconstruction

Recovery requires two distinct operations. The first operation determines which packets (source and repair) must be considered in order to recover the missing packets of a given block. Once this is done, the second step is the reconstruction of the missing data.

8.3.1. Associating the Source and Repair Packets

Association of the FEC source packets and FEC repair packets is done using a combination of the source packet sequence number and the information found in the RTP header and the FEC header of the FEC repair packets. The first step is to accumulate some of the n_r = n - k repair packets that were generated in the protection operation. For that the application has to follow the steps listed below:

- For each received packet, retrieve the payload type parameter from the RTP header to identify the packet as a repair packet of the reed-solomon scheme. In case multiple repair flows are used, different payload types will be used to distinguish between the different repair flows.

- If a FEC repair packet is received, retrieve the sequence number (SN) from the RTP header and the n_r and i parameters from the FEC header. With these parameters, identify the collection of FEC repair packets generated for the source block. For example, if n_r = 4, i = 2 and SN = 1003, the receiver deduces that 4 FEC repair packets with sequence numbers 1001, 1002, 1003 and 1004 have been generated for this source block.

- Still in case of a FEC repair packet, retrieve the BML, pkt_span and optional bit-mask fields. If BML equals 0, then k = pkt_span and the source packets have sequence numbers SN_base up to SN_base + k. If BML is greater than 0, the first pkt_span bits of the bit-mask must be analyzed. k is then equal to the number of bits equal to 1 in this bit-mask. The sequence numbers of the source packets that are acutally part of the source block are equal to SN_base plus the offset of the bits equal to 1 in this bit-mask.

8.3.2. Recovering the source packet

In order to recover the lost source packets, the application has to rebuild the source block according to the guidelines given in Section 5 and append the repair data to it in the correct order. Zero padding will replace the lost packets in the constructed source

block. The size of each source block data packet in bytes will be equal to the size of the repair data found in the repair packets. The repair data size is the size of the RTP payload in the repair packet without the FEC header information (see figure 2). The application will then append the repair data taken from each repair packet. This new block is provided to the Reed-Solomon code.

Reconstruction of lost packets (source or repair packets) is possible only if at least any k packets were received (source or repair).

The Reed-Solomon code will reconstruct the lost data into the provided source block overriding the zero padded blocks. The application can then recover the lost packets as follows:

- The first two bytes specify the RTP packet size.
- According to the RTP packet size the application can retrieve the RTP packet (RTP header and payload).
- Any extra padding bytes if exist are ignored.

9. SDP Examples

The following example demonstrates source flow with flow ID of 0 that is protected by a single repair flow R1.

```
v=0
o=sarit 1122334455 1122334466 IN IP4 fec.example.com
s= Reed Solomon FEC Example
t=0 0
a=group:FEC S1 R1
m=video 30000 RTP/AVP 100
c=IN IP4 233.252.0.1/127
a=rtpmap:100 MP2T/90000
a=fec-source-flow: id=0
a=mid:S1
m=application 30000 RTP/AVP 110
c=IN IP4 233.252.0.2/127
a=rtpmap:110 reed-solomon-fec /90000
a=fmtp:110 max_n:16; repair-window:200000; symbol-size:8
a=mid:R1
```

Figure 4

10. Implementation Considerations

Using Reed-Solomon FEC protection over RTP may be useful for efficiently overcoming newtork packet losses in interactive communications where latency constraints apply. Protection may be applied for small encoding blocks, and therefore latency caused by waiting for the FEC repair packets is minimized.

This document allows the application to set the FEC strength dynamically according to the experienced and measured loss rate, for optimizing bandwidth utilization while recovering from network errors.

When FEC protection is used due to network congestion conditions, it is important that the application will reduce the bandwidth used for FEC protection from the bandwidth used by the source flow, in order not to overload the already congested network with the additional FEC repair packets.

In order to minimize bandwidth overhead for repair packets, algorithm for applying FEC on source packets should be designed carefully. Using source packets with similar lengths (when possible) can minimize the bandwidth overhead of the FEC repair packets.

In order to maximize the FEC strength, when a ratio of k/n is chosen, the larger the source blocks size (n) is, the stronger the FEC protection is. Of course, on the other hand the larger the source block size is, the larger the latency is (caused by waiting for the FEC repair packet). The application should choose carefuly the FEC block size in order to maximize the FEC strength while keeping an acceptable latency at the receiver waiting for the FEC repair packets.

11. Offer/Answer considerations

None.

12. Security Considerations

12.1. Problem Statement

A content delivery system is potentially subject to many attacks. Some of them target the network (e.g., to compromise the routing infrastructure, by compromising the congestion control component), others target the Content Delivery Protocol (CDP) (e.g., to compromise its normal behavior), and finally some attacks target the

content itself. Since this document focuses on various FEC schemes, this section only discusses the additional threats that their use within the FECFRAME framework can create to an arbitrary CDP.

More specifically, these attacks may have several goals:

- o those that are meant to give access to a confidential content (e.g., in case of a non-free content),
- o those that try to corrupt the ADU Flows being transmitted (e.g., to prevent a receiver from using it),
- o and those that try to compromise the receiver's behavior (e.g., by making the decoding of an object computationally expensive).

These attacks can be launched either against the data flow itself (e.g., by sending forged FEC Source/Repair Packets) or against the FEC parameters that are sent either in-band (e.g., in the Repair FEC Payload ID) or out-of-band (e.g., in a session description).

12.2. Attacks Against the Data Flow

First of all, let us consider the attacks against the data flow.

12.2.1. Access to Confidential Contents

Access control to the ADU Flow being transmitted is typically provided by means of encryption. This encryption can be done within the content provider itself, by the application (for instance by using the Secure Real-time Transport Protocol (SRTP) [RFC3711]), or at the Network Layer, on a packet per packet basis when IPSec/ESP is used [RFC4303]. If confidentiality is a concern, it is RECOMMENDED that one of these solutions be used. Even if we mention these attacks here, they are not related nor facilitated by the use of FEC.

12.2.2. Content Corruption

Protection against corruptions (e.g., after sending forged FEC Source/Repair Packets) is achieved by means of a content integrity verification/sender authentication scheme. This service is usually provided at the packet level. In this case, after removing all forged packets, the ADU Flow may be sometimes recovered. Several techniques can provide this source authentication/content integrity service:

- o at the application level, the Secure Real-time Transport Protocol (SRTP) [RFC3711] provides several solutions to authenticate the source and check the integrity of RTP and RTCP messages, among

other services. For instance, associated to the Timed Efficient Stream Loss-Tolerant Authentication (TESLA) [RFC4383], SRTP is an attractive solution that is robust to losses, provides a true authentication/integrity service, and does not create any prohibitive processing load or transmission overhead. Yet, checking a packet requires a small delay (a second or more) after its reception with TESLA. Other building blocks can be used within SRTP to provide authentication/content integrity services.

- o at the Network Layer, IPSec/ESP offers (among other services) an integrity verification mechanism that can be used to provide authentication/content integrity services.

Techniques relying on public key cryptography (e.g., digital signatures) require that public keys be securely associated to the entities. This can be achieved by a Public Key Infrastructure (PKI), or by a PGP Web of Trust, or by pre-distributing the public keys of each group member.

Techniques relying on symmetric key cryptography require that a secret key be shared by all group members. This can be achieved by means of a group key management protocol, or simply by pre-distributing the secret key (but this manual solution has many limitations).

It is up to the developer and deployer, who know the security requirements and features of the target application area, to define which solution is the most appropriate. Nonetheless it is RECOMMENDED that at least one of these techniques be used.

12.3. Attacks Against the FEC Parameters

Let us now consider attacks against the FEC parameters included in the FFCI that are usually sent out-of-band (e.g., in a session description). Attacks on these FEC parameters can prevent the decoding of the associated object. For instance modifying the m field (when applicable) will lead a receiver to consider a different code. Modifying the E parameter will lead a receiver to consider bad Repair Symbols for a received FEC Repair Packet.

It is therefore RECOMMENDED that security measures be taken to guarantee the FFCI integrity. When the FFCI is sent out-of-band in a session description, this latter SHOULD be protected, for instance by digitally signing it.

Attacks are also possible against some FEC parameters included in the Explicit Repair FEC Payload ID. For instance modifying the SN_base of a FEC Repair Packet will lead a receiver to assign this packet to

a wrong block.

It is therefore RECOMMENDED that security measures be taken to guarantee the Explicit Repair FEC Payload ID integrity. To that purpose, one of the packet-level source authentication/content integrity techniques of Section 12.2.2 can be used.

13. IANA Considerations

New media subtypes are subject to IANA registration. For the registration of the payload formats and their parameters introduced in this document, refer to Section 7.

14. Acknowledgments

Some parts of this document are borrowed from the following documents: [RFC5109], [I-D.ietf-fecframe-1d2d-parity-scheme], [I-D.roca-fecframe-rs], [I-D.ietf-avt-reedsolomon]. The author would like to thank the editors of these documents. The authors would also like to thank the members of the FP7 3DPresence project consortium for their contribution to this draft writing.

15. References

15.1. Normative References

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

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[RFC4288] Freed, N. and J. Klensin, "Media Type Specifications and Registration Procedures", BCP 13, RFC 4288, December 2005.

[RFC3555] Casner, S. and P. Hoschka, "MIME Type Registration of RTP Payload Formats", RFC 3555, July 2003.

[RFC4756] Li, A., "Forward Error Correction Grouping Semantics in Session Description Protocol", RFC 4756, November 2006.

[RFC5510] Lacan, J., Roca, V., Peltotalo, J., and S. Peltotalo, "Reed-Solomon Forward Error Correction (FEC) Schemes", RFC 5510, April 2009.

[I-D.ietf-fecframe-framework] Watson, M., "Forward Error Correction (FEC) Framework", draft-ietf-fecframe-framework-09 (work in progress), July 2010.

15.2. Informative References

[I-D.ietf-fecframe-1d2d-parity-scheme] Begen, A., "RTP Payload Format for Non-Interleaved and Interleaved Parity FEC", draft-ietf-fecframe-1d2d-parity-scheme-01 (work in progress), May 2009.

[RFC5109] Li, A., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, December 2007.

[I-D.ietf-avt-reedsolomon] Rosenberg, J. and H. Schulzrinne, "An RTP Payload Format for Reed Solomon Codes", May 1999.

[Rizzo97] Rizzo, L., "Effective Erasure Codes for Reliable Computer Communication Protocols", ACM SIGCOMM Computer Communication Review Vol.27, No.2, pp.24-36, April 1997.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

[RFC4383] Baugher, M. and E. Carrara, "The Use of Timed Efficient Stream Loss-Tolerant Authentication (TESLA) in the Secure Real- time Transport Protocol (SRTP)", RFC 4383, February 2006.

[I-D.ietf-fecframe-sdp-elements] Begen, A., "Session Description Protocol (SDP) Elements for FEC Framework", draft-ietf-fecframe-sdp-elements-08 (work in progress), August 2010.

[I-D.roca-fecframe-rs] Roca, V., Cunche, M., Lacan, J., Bouabdallah, A., and K. Matsuzono, "Reed-Solomon Forward Error Correction (FEC)

Schemes for FECFRAME", draft-roca-fecframe-rs-03 (work in progress), July 2010.

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

Authors' Addresses

Sarit Galanos
RADVISION
24 Raul Wallenberg St.
Tel Aviv 69719
Israel

Email: sarit@radvision.com

Orly Peck
RADVISION
24 Raul Wallenberg St.
Tel Aviv 69719
Israel

Email: orlyp@radvision.com

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

Email: vincent.roca@inria.fr

FecFrame
Internet-Draft
Intended status: Experimental
Expires: April 25, 2011

V. Roca
INRIA
M. Cunche
NICTA
J. Lacan
ISAE/LAAS-CNRS

October 22, 2010

Simple LDPC-Staircase Forward Error Correction (FEC) Scheme for FECFRAME
draft-roca-fecframe-ldpc-01

Abstract

This document describes a fully-specified simple FEC scheme for LDPC-staircase codes that can be used to protect media streams along the lines defined by the FECFRAME framework. These codes have many interesting properties: they are systematic codes, they perform close to ideal codes in many use-cases and they also feature very high encoding and decoding throughputs. LDPC-Staircase codes are therefore a good solution to protect a single high bitrate source flow, or to protect globally several mid-rate flows within a single FECFRAME instance. They are also a good solution whenever the processing load of a software encoder or decoder must be kept to a minimum.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 25, 2011.

Copyright Notice

Table of Contents

1. Introduction

The use of Forward Error Correction (FEC) codes is a classic solution to improve the reliability of unicast, multicast and broadcast Content Delivery Protocols (CDP) and applications [RFC3453]. The [FECFRAME-FRAMEWORK] document describes a generic framework to use FEC schemes with media delivery applications, and for instance with real-time streaming media applications based on the RTP real-time protocol. Similarly the [RFC5052] document describes a generic framework to use FEC schemes with with objects (e.g., files) delivery applications based on the ALC [RFC5775] and NORM [RFC5740] reliable multicast transport protocols.

More specifically, the [RFC5053] (Raptor) and [RFC5170] (LDPC-Staircase and LDPC-Triangle) FEC schemes introduce erasure codes based on sparse parity check matrices for object delivery protocols like ALC and NORM. Similarly, the [RFC5510] document introduces Reed-Solomon codes based on Vandermonde matrices for the same object delivery protocols. All these codes are systematic codes, meaning that the k source symbols are part of the n encoding symbols. Additionally, the Reed-Solomon FEC codes belong to the class of Maximum Distance Separable (MDS) codes that are optimal in terms of erasure recovery capabilities. It means that a receiver can recover the k source symbols from any set of exactly k encoding symbols out of n. This is not the case with either Raptor or LDPC-Staircase codes, and these codes require a certain number of encoding symbols in excess to k. However, this number is small in practice when an appropriate decoding scheme is used at the receiver [SPSC08]. Another key difference is the high encoding/decoding complexity of Reed-Solomon codecs compared to Raptor or LDPC-Staircase codes. A difference of one or more orders of magnitude or more in terms of encoding/decoding speed exists between the Reed-Solomon and LDPC-Staircase software codecs [SPSC08][CunchePHD10]. Finally, Raptor and LDPC-Staircase codes are large block FEC codes, in the sense of [RFC3453], since they can efficiently deal with a large number of source symbols.

The present document focuses on LDPC-Staircase codes, that belong to the well-known class of "Low Density Parity Check" codes. Because of their key features, these codes are a good solution to protect a single high bitrate source flow as in [LCN10], or to protect globally several mid-rate source flows within a single FECFRAME instance. They are also a good solution whenever processing requirements at a software encoder or decoder must be kept to a minimum, independently of the ADU flow(s) bitrate.

This documents inherits from [RFC5170] the specifications of the core LDPC-Staircase codes. Therefore this document specifies only the

information specific to the FECFRAME context and refers to [RFC5170] for the core specifications of the codes. To that purpose, the present document introduces:

- the Fully-Specified FEC Scheme with FEC Encoding ID XXX that specifies a simple way of using LDPC-Staircase codes in order to protect arbitrary ADU flows.

Finally, a publicly available reference implementation of these codes is available and distributed under a GNU/LGPL (Lesser General Public License) [LDPC-codec].

2. Terminology

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 RFC 2119 [RFC2119].

3. Definitions Notations and Abbreviations

3.1. Definitions

This document uses the following terms and definitions. Some of them are FEC scheme specific and are in line with [RFC5052]:

Source symbol: unit of data used during the encoding process. In this specification, there is always one source symbol per ADU.

Encoding symbol: unit of data generated by the encoding process. With systematic codes, source symbols are part of the encoding symbols.

Repair symbol: encoding symbol that is not a source symbol.

Code rate: the k/n ratio, i.e., the ratio between the number of source symbols and the number of encoding symbols. By definition, the code rate is such that: 0 < code rate <= 1. A code rate close to 1 indicates that a small number of repair symbols have been produced during the encoding process.

Systematic code: FEC code in which the source symbols are part of the encoding symbols. The Reed-Solomon codes introduced in this document are systematic.

Source block: a block of k source symbols that are considered together for the encoding.

Packet Erasure Channel: a communication path where packets are either dropped (e.g., by a congested router, or because the number of transmission errors exceeds the correction capabilities of the physical layer codes) or received. When a packet is received, it is assumed that this packet is not corrupted.

Some of them are FECFRAME framework specific and are in line with

[FECFRAME-FRAMEWORK]:

Application Data Unit (ADU): a unit of data coming from (sender) or given to (receiver) the media delivery application. Depending on the use-case, an ADU can use an RTP encapsulation. In this specification, there is always one source symbol per ADU.

(Source) ADU Flow: a flow of ADUs from a media delivery application and to which FEC protection is applied. Depending on the use-case, several ADU flows can be protected together by the FECFRAME framework.

ADU Block: a set of ADUs that are considered together by the FECFRAME instance for the purpose of the FEC scheme. Along with the F[], L[], and Pad[] fields, they form the set of source symbols over which FEC encoding will be performed.

ADU Information (ADUI): a unit of data constituted by the ADU and the associated Flow ID, Length and Padding fields (Section 4.3). This is the unit of data that is used as source symbol.

FEC Framework Configuration Information: the FEC scheme specific information that enables the synchronization of the FECFRAME sender and receiver instances.

FEC Source Packet: a data packet submitted to (sender) or received from (receiver) the transport protocol. It contains an ADU along with its optional Explicit Source FEC Payload ID.

FEC Repair Packet: a repair packet submitted to (sender) or received from (receiver) the transport protocol. It contains a repair symbol along with its Repair FEC Payload ID.

The above terminology is illustrated in Figure 1 (sender’s point of view):

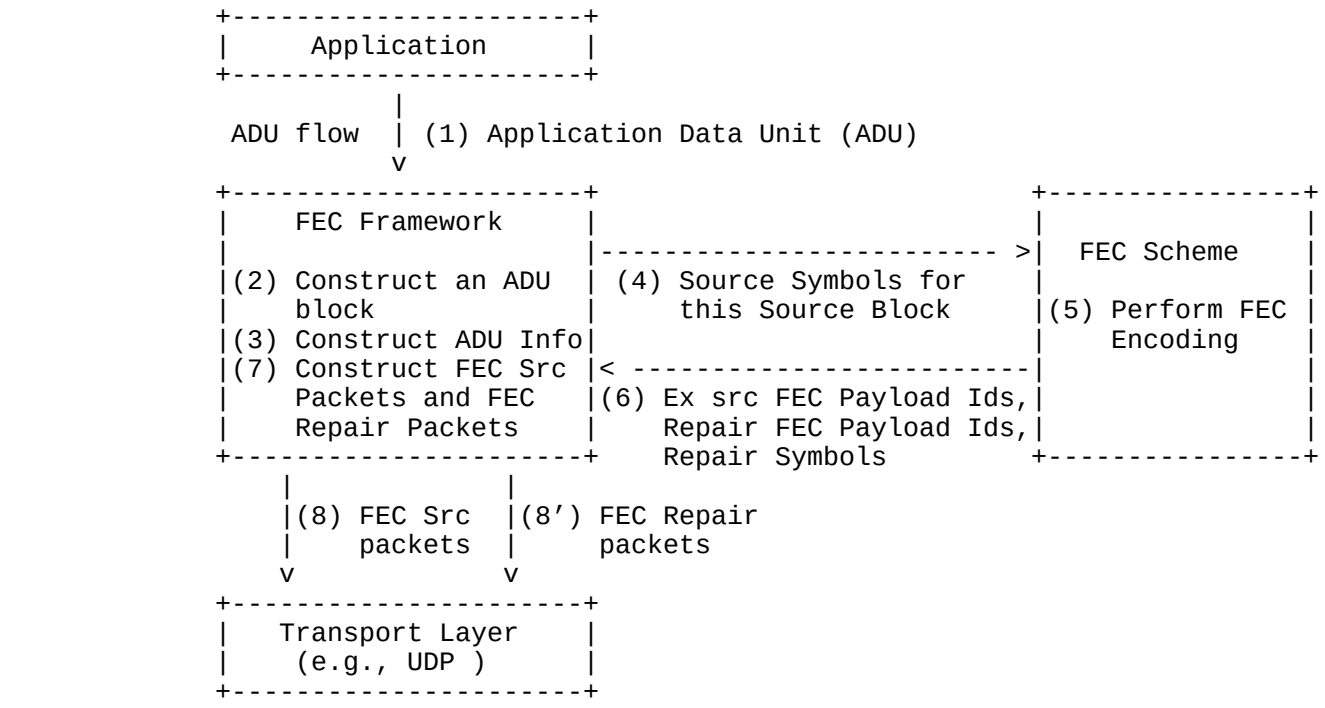

Figure 1: Terminology used in this document (sender).

3.2. Notations

This document uses the following notations: Some of them are FEC scheme specific:

k	denotes the number of source symbols in a source block.
max_k	denotes the maximum number of source symbols for any source block.
n	denotes the number of encoding symbols generated for a source block.
E	denotes the encoding symbol length in bytes.
CR	denotes the "code rate", i.e., the k/n ratio.
N1	denotes the target number of "1s" per column in the left side of the parity check matrix.
N1m3	denotes the value N1 - 3.
a^^b	denotes a raised to the power b.

Some of them are FECFRAME framework specific:

B denotes the number of ADUs per ADU block.
max_B denotes the maximum number of ADUs for any ADU block.

3.3. Abbreviations

This document uses the following abbreviations:
ADU stands for Application Data Unit.
ESI stands for Encoding Symbol ID.
FEC stands for Forward Error (or Erasure) Correction code.
FFCI stands for FEC Framework Configuration Information.
LDPC stands for Low Density Parity Check.
RS stands for Reed-Solomon.
MDS stands for Maximum Distance Separable code.

4. Common Procedures Related to the ADU Block and Source Block Creation

This section introduces the procedures that are used during the ADU block and the related source block creation, for the FEC scheme considered.

4.1. Restrictions

This specification has the following restrictions:
- there MUST be exactly one source symbol per ADUI, and therefore per ADU;
- there MUST be exactly one repair symbol per FEC Repair Packet;
- there MUST be exactly one source block per ADU block;
- the use of the LDPC-Staircase scheme is such that there MUST be exactly one encoding symbol per group, i.e., G MUST be equal to 1 [RFC5170];

4.2. ADU Block Creation

Several aspects must be considered, that impact the ADU block creation:
- the maximum source block size (max_k parameter);
- the potential real-time constraints, that impact the maximum ADU block size, since the larger the block size, the larger the decoding delay;

We now detail each of these aspects.

The maximum source block length in symbols, max_k, depends on several parameters: the code rate (CR), the Encoding Symbol ID (ESI) field length in the Explicit Source/Repair FEC Payload ID (16 bits), as well as possible internal codec limitations. More specifically, max_k cannot be larger than the following values, derived from the ESI field size limitation, for a given code rate:

max1_k = 2^^(16 - ceil(Log2(1/CR)))

Some common max1_k values are:

- CR == 1 (no repair symbol): max1_k = 2^^16 = 65536 symbols
- 1/2 <= CR < 1: max1_k = 2^^15 = 32,768 symbols
- 1/4 <= CR < 1/2: max1_k = 2^^14 = 16,384 symbols

Additionally, a codec MAY impose other limitations on the maximum block size, for instance, because of a limited working memory size. This decision MUST be clarified at implementation time, when the target use-case is known. This results in a max2_k limitation.

Then, max_k is given by:

max_k = min(max1_k, max2_k)

Note that this calculation is only required at the encoder (sender), since the actual k parameter (k <= max_k) is communicated to the decoder (receiver) through the Explicit Source/Repair FEC Payload ID.

The source ADU flows usually have real-time constraints. It means that the maximum number of ADUs of an ADU block must not exceed a certain threshold since it directly impacts the decoding delay. It is the role of the developer, who knows the flow real-time features, to define an appropriate upper bound to the ADU block size, max_rt.

If we take into account these constraints, we find: max_B = min(max_k, max_rt). Then max_B gives an upper bound to the number of ADUs that can constitute an ADU block.

4.3. Source Block Creation

In its most general form the FECFRAME framework and the LDPC-Staircase FEC scheme are meant to protect a set of independent flows. Since the flows have no relationship to one another, the ADU size of each flow can potentially vary significantly. Even in the special case of a single flow, the ADU sizes can largely vary (e.g., the various frames of a "Group of Pictures (GOP) of an H.264 flow). This diversity must be addressed since the RS FEC scheme requires a constant encoding symbol size (E parameter) per source block. Since this specification requires that there is only one source symbol per ADU, E must be large enough to contain all the ADUs of an ADU block along with their prepended 3 bytes (see below).

In situations where E is determined per source block (default, specified by the FCCI/FSSI with S = 0, Section 5.1.1.2), E is equal to the size of the largest ADU of this source block plus three (for the prepended 3 bytes, see below). In this case, upon receiving the first FEC Repair Packet for this source block, since this packet MUST contain a single repair symbol (Section 5.1.3), a receiver determines the E parameter used for this source block.

In situations where E is fixed (specified by the FCCI/FSSI with S = 1, Section 5.1.1.2), then E must be greater or equal to the size of the largest ADU of this source block plus three (for the prepended 3 bytes, see below). If this is not the case, an error is returned. How to handle this error is use-case specific (e.g., a larger E parameter may be communicated to the receivers in an updated FFCI message, using an appropriate mechanism) and is not considered by this specification.

The ADU block is always encoded as a single source block. There are a total of B <= max_B ADUs in this ADU block. For the ADU i, with 0 <= i <= B-1, 3 bytes are prepended (Figure 2):

- The first byte, FID[i] (Flow ID), contains the integer identifier associated to the source ADU flow to which this ADU belongs to. It is assumed that a single byte is sufficient, or said differently, that no more than 256 flows will be protected by a single instance of the FECFRAME framework.
- The following two bytes, L[i] (Length), contain the length of this ADU, in network byte order (i.e., big endian). This length is for the ADU itself and does not include the FID[i], L[i], or Pad[i] fields.

Then zero padding is added to ADU i (if needed) in field Pad[i], for alignment purposes up to a size of exactly E bytes. The data unit resulting from the ADU i and the F[i], L[i] and Pad[i] fields, is called ADU Information (or ADUI). Each ADUI contributes to exactly one source symbol to the source block.

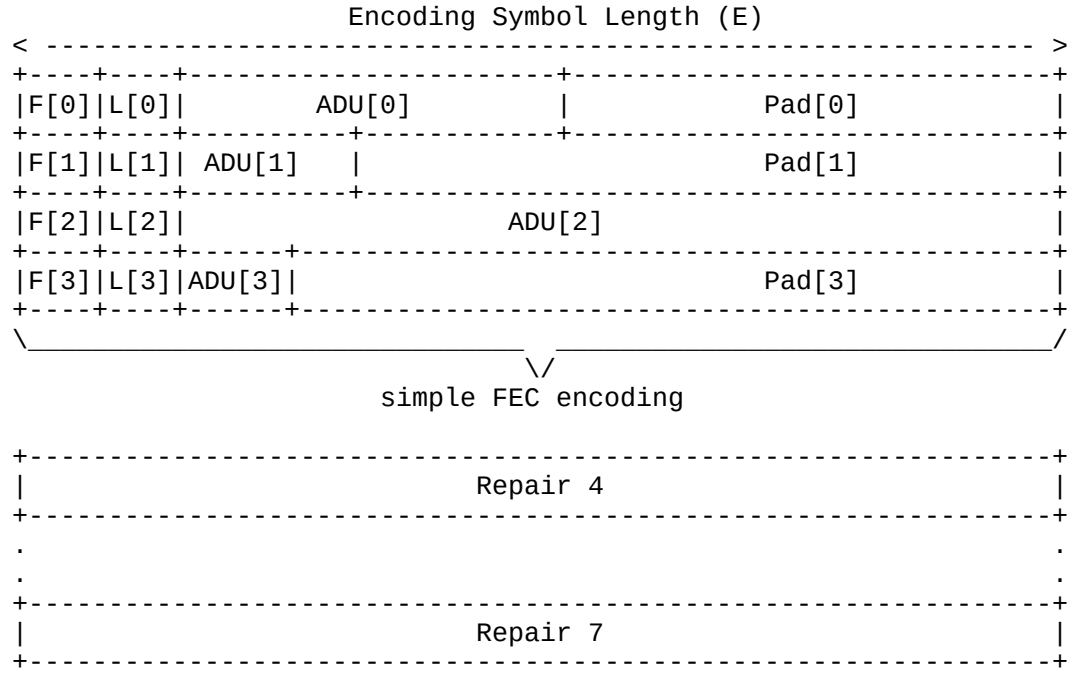

Figure 2: Source block creation, for code rate 1/2 (equal number of source and repair symbols, 4 in this example), and S = 0.

Note that neither the initial 3 bytes nor the optional padding are sent over the network. However, they are considered during FEC encoding. It means that a receiver who lost a certain FEC source packet (e.g., the UDP datagram containing this FEC source packet) will be able to recover the ADUI if FEC decoding succeeds. Thanks to the initial 3 bytes, this receiver will get rid of the padding (if any) and identify the corresponding ADU flow.

5. LDPC-Staircase FEC Scheme for Arbitrary ADU Flows

5.1. Formats and Codes

5.1.1. FEC Framework Configuration Information

The FEC Framework Configuration Information (or FFCI) includes information that MUST be communicated between the sender and receiver(s). More specifically, it enables the synchronization of the FECFRAME sender and receiver instances. It includes both mandatory elements and scheme-specific elements, as detailed below.

5.1.1.1. Mandatory Information

FEC Encoding ID: the value assigned to this fully-specified FEC scheme MUST be XXX, as assigned by IANA (Section 7).
When SDP is used to communicate the FFCI, this FEC Encoding ID is carried in the 'encoding-id' parameter.

5.1.1.2. FEC Scheme-Specific Information

The FEC Scheme Specific Information (FSSI) includes elements that are specific to the present FEC scheme. More precisely:
PRNG seed (seed): a non-negative 32 bit integer used as the seed of the Pseudo Random Number Generator, as defined in [RFC5170].
Encoding symbol length (E): a non-negative integer that indicates either the length of each encoding symbol in bytes (strict mode, i.e., if S = 1), or the maximum length of any encoding symbol (i.e., if S = 0).
Strict (S) flag: when set to 1 this flag indicates that the E parameter is valid for the whole session, unless otherwise notified. When set to 0 this flag indicates that the E parameter is only the maximum length of each encoding symbol, for the whole session, unless otherwise notified.
N1 minus 3 (n1m3): an integer between 0 (default) and 7, inclusive. The number of "1s" per column in the left side of the parity check matrix, N1, is then equal to N1m3 + 3, as specified in [RFC5170].
These elements are required both by the sender (LDPC-Staircase encoder) and the receiver(s) (LDPC-Staircase decoder).

When SDP is used to communicate the FFCI, this FEC scheme-specific information is carried in the 'fssi' parameter in textual representation as specified in [SDP_ELEMENTS]. For instance:

```
fssi = seed:1234,E:1400,S:0,n1m3:0
```

If another mechanism requires the FSSI to be carried as an opaque octet string (for instance after a Base64 encoding), the encoding format consists of the following 7 octets:
- PRNG seed (seed): 32 bit field.
- Encoding symbol length (E): 16 bit field.
- Strict (S) flag: 1 bit field.
- N1m3 parameter (n1m3): 7 bit field.

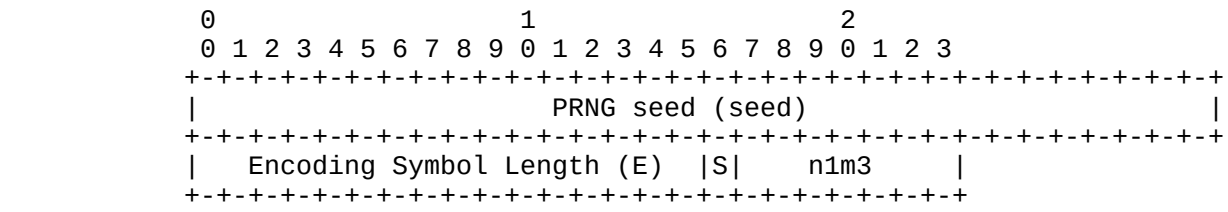

Figure 3: FSSI encoding format.

5.1.2. Explicit Source FEC Payload ID

A FEC source packet MUST contain an Explicit Source FEC Payload ID that is appended to the end of the packet as illustrated in Figure 4.

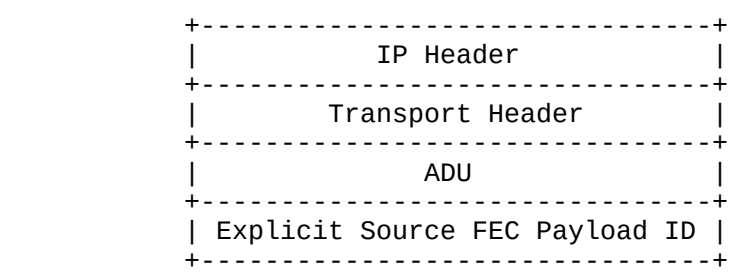

Figure 4: Structure of a FEC Source Packet with the Explicit Source FEC Payload ID.

More precisely, the Explicit Source FEC Payload ID is composed of the following fields (Figure 5):

Source Block Number (SBN) (16 bit field): this field identifies the source block to which this FEC source packet belongs.

Encoding Symbol ID (ESI) (16 bit field): this field identifies the source symbol contained in this FEC source packet. This value is such that 0 <= ESI <= k - 1 for source symbols.

Source Block Length (k) (16 bit field): this field provides the number of source symbols for this source block, i.e., the k parameter.

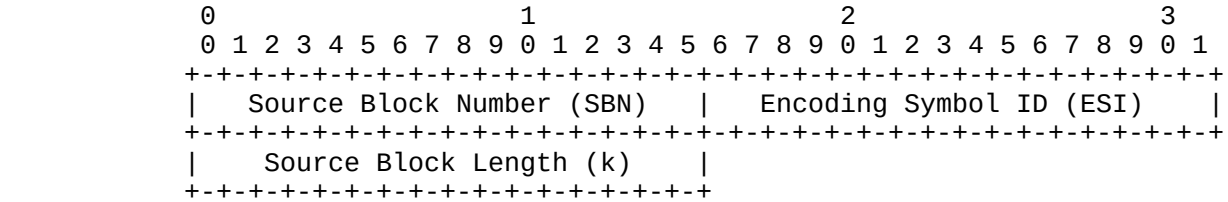

Figure 5: Source FEC Payload ID encoding format.

5.1.3. Repair FEC Payload ID

A FEC repair packet MUST contain a Repair FEC Payload ID that is prepended to the repair symbol(s) as illustrated in Figure 6. There MUST be a single repair symbol per FEC repair packet.

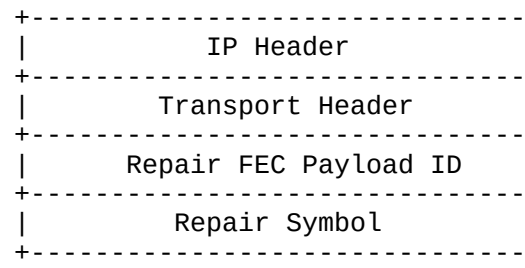

Figure 6: Structure of a FEC Repair Packet with the Repair FEC Payload ID.

More precisely, the Repair FEC Payload ID is composed of the following fields: (Figure 7):

Source Block Number (SBN) (16 bit field): this field identifies the source block to which the FEC repair packet belongs.

Encoding Symbol ID (ESI) (16 bit field) this field identifies the repair symbol contained in this FEC repair packet. This value is such that k <= ESI <= n - 1 for repair symbols.

Source Block Length (k) (16 bit field): this field provides the number of source symbols for this source block, i.e., the k parameter.

Number of Encoding Symbols (n) (16 bit field): this field provides the number of encoding symbols for this source block, i.e., the n parameter.

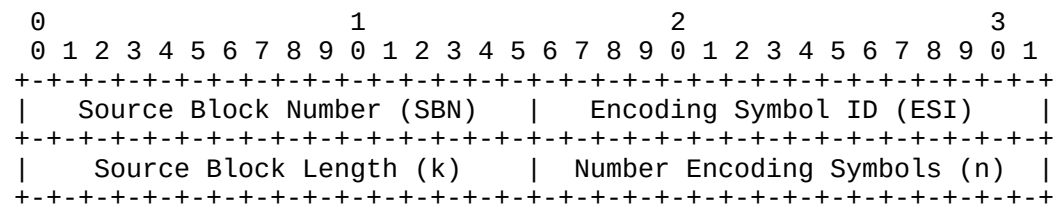

Figure 7: Repair FEC Payload ID encoding format.

5.2. Procedures

The following procedures apply:

- The source block creation procedures are specified in Section 4.3.
- The SBN value is incremented for each new source block, starting at 0 for the first block of the ADU flow. Wrapping to zero will happen for long sessions, after value 2^^16 - 1.
- The ESI of encoding symbols is managed sequentially, starting at 0 for the first symbol. The first k values (0 <= ESI <= k - 1) identify source symbols, whereas the last n-k values (k <= ESI <= n - 1) identify repair symbols.
- The FEC repair packet creation procedures are specified in Section 5.1.3.

5.3. FEC Code Specification

The present document inherits from [RFC5170] the specification of the core LDPC-Staircase codes for a packet erasure transmission channel.

Because of the requirement to have exactly one encoding symbol per group, i.e., because G MUST be equal to 1 (Section 4.1), several parts of [RFC5170] are useless. In particular, this is the case of Section 5.6. "Identifying the G Symbols of an Encoding Symbol Group".

6. Security Considerations

6.1. Problem Statement

A content delivery system is potentially subject to many attacks. Some of them target the network (e.g., to compromise the routing infrastructure, by compromising the congestion control component), others target the Content Delivery Protocol (CDP) (e.g., to compromise its normal behavior), and finally some attacks target the content itself. Since this document focuses on various FEC schemes, this section only discusses the additional threats that their use within the FECFRAME framework can create to an arbitrary CDP.

More specifically, these attacks may have several goals:
- those that are meant to give access to a confidential content (e.g., in case of a non-free content),
- those that try to corrupt the ADU Flows being transmitted (e.g., to prevent a receiver from using it),
- and those that try to compromise the receiver’s behavior (e.g., by making the decoding of an object computationally expensive).

These attacks can be launched either against the data flow itself (e.g., by sending forged FEC Source/Repair Packets) or against the FEC parameters that are sent either in-band (e.g., in the Repair FEC Payload ID) or out-of-band (e.g., in a session description).

6.2. Attacks Against the Data Flow

First of all, let us consider the attacks against the data flow.

6.2.1. Access to Confidential Contents

Access control to the ADU Flow being transmitted is typically provided by means of encryption. This encryption can be done within the content provider itself, by the application (for instance by using the Secure Real-time Transport Protocol (SRTP) [RFC3711]), or at the Network Layer, on a packet per packet basis when IPSec/ESP is used [RFC4303]. If confidentiality is a concern, it is RECOMMENDED that one of these solutions be used. Even if we mention these attacks here, they are not related nor facilitated by the use of FEC.

6.2.2. Content Corruption

Protection against corruptions (e.g., after sending forged FEC Source/Repair Packets) is achieved by means of a content integrity verification/sender authentication scheme. This service is usually provided at the packet level. In this case, after removing all forged packets, the ADU Flow may be sometimes recovered. Several techniques can provide this source authentication/content integrity service:

- at the application level, the Secure Real-time Transport Protocol (SRTP) [RFC3711] provides several solutions to authenticate the source and check the integrity of RTP and RTCP messages, among other services. For instance, associated to the Timed Efficient Stream Loss-Tolerant Authentication (TESLA) [RFC4383], SRTP is an attractive solution that is robust to losses, provides a true authentication/integrity service, and does not create any prohibitive processing load or transmission overhead. Yet, checking a packet requires a small delay (a second or more) after its reception with TESLA. Other building blocks can be used within SRTP to provide authentication/content integrity services.
- at the Network Layer, IPSec/ESP offers (among other services) an integrity verification mechanism that can be used to provide authentication/content integrity services.

It is up to the developer and the person in charge of deployment, who know the security requirements and features of the target application area, to define which solution is the most appropriate. Nonetheless it is RECOMMENDED that at least one of these techniques be used.

6.3. Attacks Against the FEC Parameters

Let us now consider attacks against the FEC parameters included in the FFCI that are usually sent out-of-band (e.g., in a session

description). Attacks on these FEC parameters can prevent the decoding of the associated object. For instance modifying the PRNG seed or N1m3 fields will lead a receiver to consider a different parity check matrix, i.e., a different code. Modifying the E parameter will lead a receiver to consider bad Repair Symbols for a received FEC Repair Packet.

It is therefore RECOMMENDED that security measures be taken to guarantee the FFCI integrity. When the FFCI is sent out-of-band in a session description, this latter SHOULD be protected, for instance by digitally signing it.

Attacks are also possible against some FEC parameters included in the Explicit Source FEC Payload ID and Repair FEC Payload ID. For instance modifying the Source Block Number of a FEC Source of Repair Packet will lead a receiver to assign this packet to a wrong block.

It is therefore RECOMMENDED that security measures be taken to guarantee the Explicit Source FEC Payload ID and Repair FEC Payload ID integrity. To that purpose, one of the packet-level source authentication/content integrity techniques of Section 6.2.2 can be used.

7. IANA Considerations

The FEC Encoding ID value is subject to IANA registration.

TBD

8. Acknowledgments

TBD

9. References

9.1. Normative References

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

[RFC5170] Roca, V., Neumann, C., and D. Furodet, "Low Density Parity Check (LDPC) Forward Error Correction", RFC 5170, June 2008.

[FECFRAME-FRAMEWORK]

Watson, M., "Forward Error Correction (FEC) Framework", draft-ietf-fecframe-framework-10 (Work in Progress), September 2010.

[SDP_ELEMENTS]
Begen, A., "SDP Elements for FEC Framework", draft-ietf-fecframe-sdp-elements-10 (Work in Progress), October 2010.

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.

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

[RFC5510] Lacan, J., Roca, V., Peltotalo, J., and S. Peltotalo, "Reed-Solomon Forward Error Correction (FEC) Schemes", RFC 5510, April 2009.

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

[RFC5740] Adamson, B., Bormann, C., Handley, M., and J. Macker, "NACK-Oriented Reliable Multicast (NORM) Transport Protocol", RFC 5740, November 2009.

[RFC5775] Luby, M., Watson, M., and L. Vicisano, "Asynchronous Layered Coding (ALC) Protocol Instantiation", RFC 5775, April 2010.

[SPSC08] Cunche, M. and V. Roca, "Optimizing the Error Recovery Capabilities of LDPC-staircase Codes Featuring a Gaussian Elimination Decoding Scheme", 10th IEEE International Workshop on Signal Processing for Space Communications (SPSC’08), October 2008.

[CunchePHD10]
Cunche, M., "High performances AL-FEC codes for the erasure channel : variation around LDPC codes", PhD dissertation (in French) (http://tel.archives-ouvertes.fr/tel-00451336/en/), June 2010.

[LCN10] Matsuzono, K., Detchart, J., Cunche, M., Roca, V., and H.

Asaeda, "Performance Analysis of a High-Performance Real-Time Application with Several AL-FEC Schemes", 35th Annual IEEE Conference on Local Computer Networks 2010 (LCN 2010), October 2010.

[LDPC-codec] Cunche, M., Roca, V., Neumann, C., and J. Laboure, "LDPC-Staircase/LDPC-Triangle Codec Reference Implementation", INRIA Rhone-Alpes and STMicroelectronics, <http://planete-bcast.inrialpes.fr/>.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

[RFC4383] Baugher, M. and E. Carrara, "The Use of Timed Efficient Stream Loss-Tolerant Authentication (TESLA) in the Secure Real-time Transport Protocol (SRTP)", RFC 4383, February 2006.

Authors' Addresses

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

Email: vincent.roca@inria.fr
URI: http://planete.inrialpes.fr/people/roca/

Mathieu Cunche
NICTA
Australia

Email: mathieu.cunche@nicta.com.au
URI: http://mathieu.cunche.free.fr/

Jerome Lacan
ISAE/LAAS-CNRS
1, place Emile Blouin
Toulouse 31056
France

Email: jerome.lacan@isae.fr
URI: http://dmi.ensica.fr/auteur.php3?id_auteur=5

FecFrame
Internet-Draft
Intended status: Experimental
Expires: April 25, 2011

V. Roca
INRIA
M. Cunche
NICTA
J. Lacan
A. Bouabdallah
ISAE/LAAS-CNRS
K. Matsuzono
Keio University

October 22, 2010

Simple Reed-Solomon Forward Error Correction (FEC) Scheme for FECFRAME
draft-roca-fecframe-simple-rs-01

Abstract

This document describes a fully-specified simple FEC scheme for Reed-Solomon codes over GF(2^^m), with 2 <= m <= 16, that can be used to protect arbitrary media streams along the lines defined by the FECFRAME framework. Reed-Solomon codes belong to the class of Maximum Distance Separable (MDS) codes which means they offer optimal protection against packet erasures. They are also systematic codes, which means that the source symbols are part of the encoding symbols. The price to pay is a limit on the maximum source block size, on the maximum number of encoding symbols, and a computational complexity higher than that of LDPC codes for instance.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 25, 2011.

Copyright Notice

Table of Contents

1. Introduction

The use of Forward Error Correction (FEC) codes is a classic solution to improve the reliability of unicast, multicast and broadcast Content Delivery Protocols (CDP) and applications. The [FECFRAME-FRAMEWORK] document describes a generic framework to use FEC schemes with media delivery applications, and for instance with real-time streaming media applications based on the RTP real-time protocol. Similarly the [RFC5052] document describes a generic framework to use FEC schemes with with objects (e.g., files) delivery applications based on the ALC [RFC5775] and NORM [RFC5740] reliable multicast transport protocols.

More specifically, the [RFC5053] and [RFC5170] FEC schemes introduce erasure codes based on sparse parity check matrices for object delivery protocols like ALC and NORM. These codes are efficient in terms of processing but not optimal in terms of erasure recovery capabilities when dealing with "small" objects.

The Reed-Solomon FEC codes described in this document belong to the class of Maximum Distance Separable (MDS) codes that are optimal in terms of erasure recovery capability. It means that a receiver can recover the k source symbols from any set of exactly k encoding symbols. These codes are also systematic codes, which means that the k source symbols are part of the encoding symbols. However they are limited in terms of maximum source block size and number of encoding symbols. Since the real-time constraints of media delivery applications usually limit the maximum source block size, this is not considered to be a major issue in the context of the FEC Framework for many (but not necessarily all) use-cases. Additionally, if the encoding/decoding complexity is higher with Reed-Solomon codes than it is with [RFC5053] or [RFC5170] codes, it remains reasonable for most use-cases, even in case of a software codec.

Many applications dealing with reliable content transmission or content storage already rely on packet-based Reed-Solomon erasure recovery codes. In particular, many of them use the Reed-Solomon codec of Luigi Rizzo [RS-codec] [Rizzo97]. The goal of the present document is to specify a simple Reed-Solomon scheme that is compatible with this codec.

More specifically, the [RFC5510] document introduced such Reed-Solomon codes and several associated FEC schemes that are compatible with the [RFC5052] framework. The present document inherits from [RFC5510] the specification of the core Reed-Solomon codes based on Vandermonde matrices, and specifies a simple FEC scheme that is compatible with the FECFRAME framework [FECFRAME-FRAMEWORK]. Therefore this document specifies only the information specific to

the FECFRAME context and refers to [RFC5510] for the core specifications of the codes. To that purpose, the present document introduces:

- the Fully-Specified FEC Scheme with FEC Encoding ID XXX that specifies a simple way of using of Reed-Solomon codes over GF(2^^m), with 2 <= m <= 16, with a simple FEC encoding for arbitrary packet flows;

For instance, with this scheme, a set of Application Data Units (or ADUs) coming from one or several (resp. one) media delivery applications (e.g., a set of RTP packets), are grouped in an ADU block and FEC encoded as a whole. With Reed-Solomon codes over GF(2^^8), there is a strict limit over the number of ADUs that can be protected together, since the number of encoded symbols, n, must be inferior or equal to 255. This constraint is relaxed when using a higher finite field size (m > 8), at the price of an increased computational complexity.

2. Terminology

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 RFC 2119 [RFC2119].

3. Definitions Notations and Abbreviations

3.1. Definitions

This document uses the following terms and definitions. Some of them are FEC scheme specific and are in line with [RFC5052]:

Source symbol: unit of data used during the encoding process. In this specification, there is always one source symbol per ADU.

Encoding symbol: unit of data generated by the encoding process. With systematic codes, source symbols are part of the encoding symbols.

Repair symbol: encoding symbol that is not a source symbol.

Code rate: the k/n ratio, i.e., the ratio between the number of source symbols and the number of encoding symbols. By definition, the code rate is such that: 0 < code rate <= 1. A code rate close to 1 indicates that a small number of repair symbols have been produced during the encoding process.

Systematic code: FEC code in which the source symbols are part of the encoding symbols. The Reed-Solomon codes introduced in this document are systematic.

Source block: a block of k source symbols that are considered together for the encoding.

Packet Erasure Channel: a communication path where packets are either dropped (e.g., by a congested router, or because the number of transmission errors exceeds the correction capabilities of the physical layer codes) or received. When a packet is received, it is assumed that this packet is not corrupted.

Some of them are FECFRAME framework specific and are in line with [FECFRAME-FRAMEWORK]:

Application Data Unit (ADU): a unit of data coming from (sender) or given to (receiver) the media delivery application. Depending on the use-case, an ADU can use an RTP encapsulation. In this specification, there is always one source symbol per ADU.

(Source) ADU Flow: a flow of ADUs from a media delivery application and to which FEC protection is applied. Depending on the use-case, several ADU flows can be protected together by the FECFRAME framework.

ADU Block: a set of ADUs that are considered together by the FECFRAME instance for the purpose of the FEC scheme. Along with the F[], L[], and Pad[] fields, they form the set of source symbols over which FEC encoding will be performed.

ADU Information (ADUI): a unit of data constituted by the ADU and the associated Flow ID, Length and Padding fields (Section 4.3). This is the unit of data that is used as source symbol.

FEC Framework Configuration Information: the FEC scheme specific information that enables the synchronization of the FECFRAME sender and receiver instances.

FEC Source Packet: a data packet submitted to (sender) or received from (receiver) the transport protocol. It contains an ADU along with its optional Explicit Source FEC Payload ID.

FEC Repair Packet: a repair packet submitted to (sender) or received from (receiver) the transport protocol. It contains a repair symbol along with its Repair FEC Payload ID.

The above terminology is illustrated in Figure 1 (sender’s point of view):

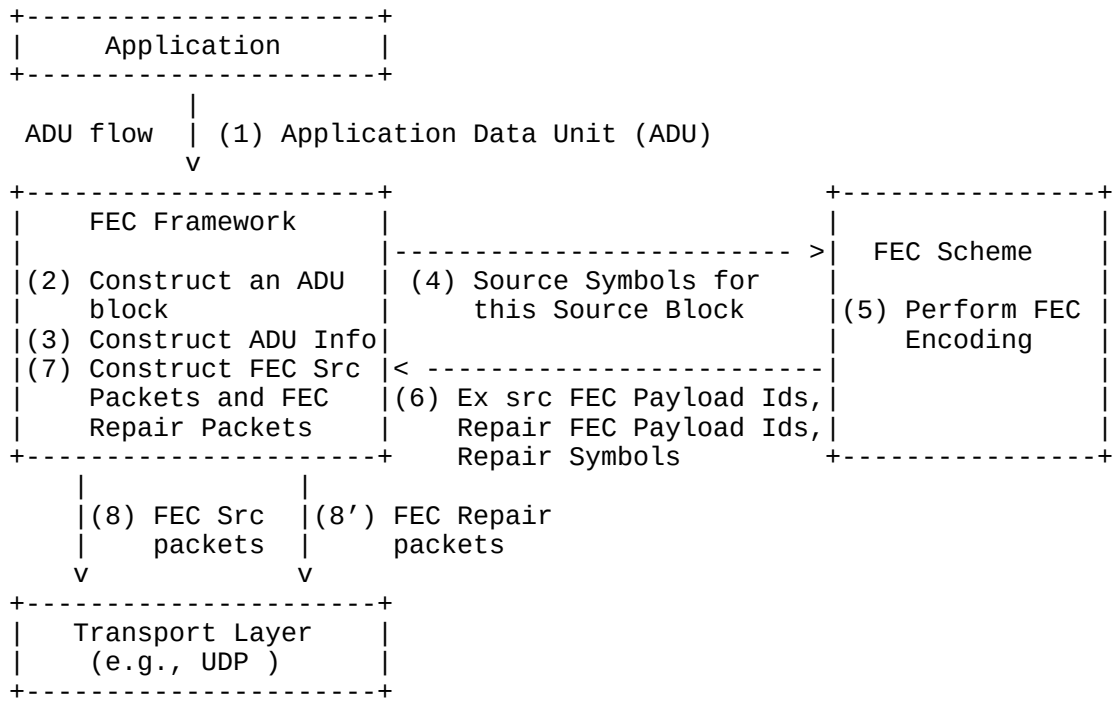

Figure 1: Terminology used in this document (sender).

3.2. Notations

This document uses the following notations: Some of them are FEC scheme specific:

k denotes the number of source symbols in a source block.

max_k denotes the maximum number of source symbols for any source block.

n denotes the number of encoding symbols generated for a source block.

E denotes the encoding symbol length in bytes.

GF(q) denotes a finite field (also known as Galois Field) with q elements. We assume that q = 2^^m in this document.

m defines the length of the elements in the finite field, in bits. In this document, m is such that 2 <= m <= 16.

q defines the number of elements in the finite field. We have: q = 2^^m in this specification.

CR denotes the "code rate", i.e., the k/n ratio.

a^^b denotes a raised to the power b.

Some of them are FECFRAME framework specific:
B denotes the number of ADUs per ADU block.
max_B denotes the maximum number of ADUs for any ADU block.

3.3. Abbreviations

This document uses the following abbreviations:
ADU stands for Application Data Unit.
ESI stands for Encoding Symbol ID.
FEC stands for Forward Error (or Erasure) Correction code.
FFCI stands for FEC Framework Configuration Information.
RS stands for Reed-Solomon.
MDS stands for Maximum Distance Separable code.

4. Common Procedures Related to the ADU Block and Source Block Creation

This section introduces the procedures that are used during the ADU block and the related source block creation, for the FEC scheme considered.

4.1. Restrictions

This specification has the following restrictions:
- there MUST be exactly one source symbol per ADUI, and therefore per ADU;
- there MUST be exactly one repair symbol per FEC Repair Packet;
- there MUST be exactly one source block per ADU block;

4.2. ADU Block Creation

Several aspects must be considered, that impact the ADU block creation:
- the maximum source block size (k parameter) and number of encoding symbols (n parameter), that are constrained by the finite field size (m parameter);
- the potential real-time constraints, that impact the maximum ADU block size, since the larger the block size, the larger the decoding delay;

We now detail each of these aspects.

The finite field size parameter, m, defines the number of non zero elements in this field which is equal to: q - 1 = 2^^m - 1. This q - 1 value is also the theoretical maximum number of encoding symbols that can be produced for a source block. For instance, when m = 8 (default) there is a maximum of 2^^8 - 1 = 255 encoding symbols. So:

k < n <= 255. Given the target FEC code rate (e.g., provided by the end-user or upper application when starting the FECFRAME framework, and taking into account the (known or estimated) packet loss rate), the sender calculates:

```
max_k = floor((2^^m - 1) * CR)
```

This max_k value leaves enough room for the sender to produce the desired number of repair symbols. Since there is one source symbol per ADU, max_k is also an upper bound to the maximum number of ADUs per ADU block.

The source ADU flows usually have real-time constraints. It means that the maximum number of ADUs of an ADU block must not exceed a certain threshold since it directly impacts the decoding delay. It is the role of the developer, who knows the flow real-time features, to define an appropriate upper bound to the ADU block size, max_rt.

If we take into account these constraints, we find: max_B = min(max_k, max_rt). Then max_B gives an upper bound to the number of ADUs that can constitute an ADU block.

4.3. Source Block Creation

In its most general form the FECFRAME framework and the RS FEC scheme are meant to protect a set of independent flows. Since the flows have no relationship to one another, the ADU size of each flow can potentially vary significantly. Even in the special case of a single flow, the ADU sizes can largely vary (e.g., the various frames of a "Group of Pictures (GOP) of an H.264 flow). This diversity must be addressed since the RS FEC scheme requires a constant encoding symbol size (E parameter) per source block. Since this specification requires that there is only one source symbol per ADU, E must be large enough to contain all the ADUs of an ADU block along with their prepended 3 bytes (see below).

In situations where E is determined per source block (default, specified by the FCCI/FSSI with S = 0, Section 5.1.1.2), E is equal to the size of the largest ADU of this source block plus three (for the prepended 3 bytes, see below). In this case, upon receiving the first FEC Repair Packet for this source block, since this packet MUST contain a single repair symbol (Section 5.1.3), a receiver determines the E parameter used for this source block.

In situations where E is fixed (specified by the FCCI/FSSI with S = 1, Section 5.1.1.2), then E must be greater or equal to the size of the largest ADU of this source block plus three (for the prepended 3 bytes, see below). If this is not the case, an error is returned. How to handle this error is use-case specific (e.g., a larger E parameter may be communicated to the receivers in an updated FFCI

message, using an appropriate mechanism) and is not considered by this specification.

The ADU block is always encoded as a single source block. There are a total of B <= max_B ADUs in this ADU block. For the ADU i, with 0 <= i <= B-1, 3 bytes are prepended (Figure 2):

- The first byte, FID[i] (Flow ID), contains the integer identifier associated to the source ADU flow to which this ADU belongs to. It is assumed that a single byte is sufficient, or said differently, that no more than 256 flows will be protected by a single instance of the FECFRAME framework.
- The following two bytes, L[i] (Length), contain the length of this ADU, in network byte order (i.e., big endian). This length is for the ADU itself and does not include the FID[i], L[i], or Pad[i] fields.

Then zero padding is added to ADU i (if needed) in field Pad[i], for alignment purposes up to a size of exactly E bytes. The data unit resulting from the ADU i and the F[i], L[i] and Pad[i] fields, is called ADU Information (or ADUI). Each ADUI contributes to exactly one source symbol to the source block.

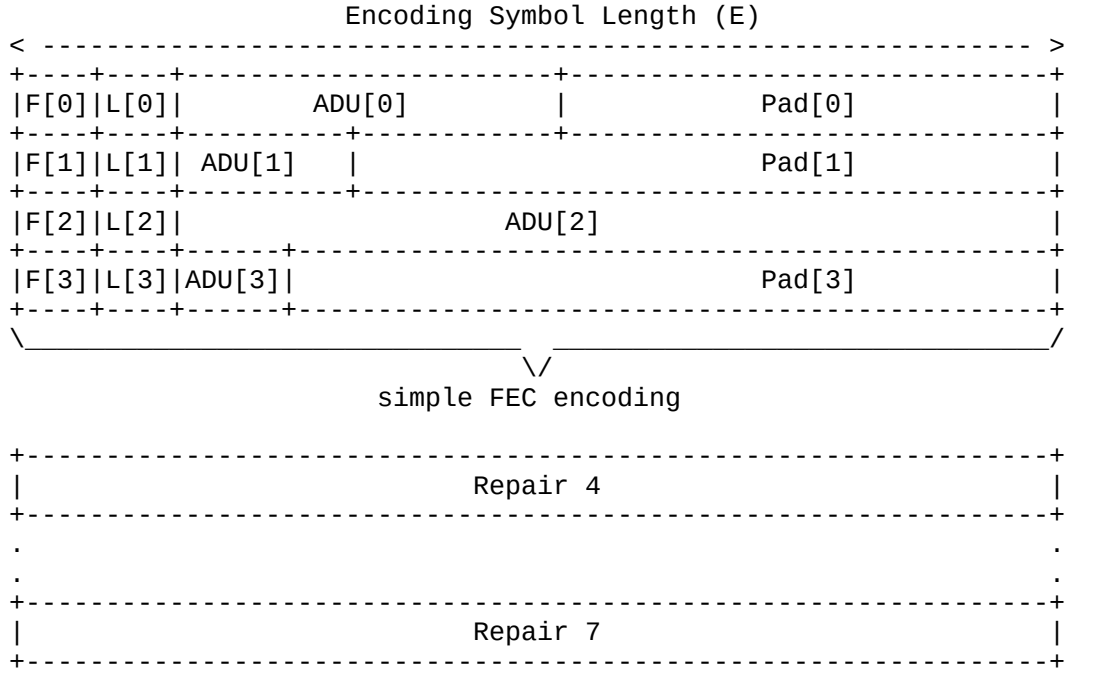

Figure 2: Source block creation, for code rate 1/2 (equal number of source and repair symbols, 4 in this example), and S = 0.

Note that neither the initial 3 bytes nor the optional padding are sent over the network. However, they are considered during FEC encoding. It means that a receiver who lost a certain FEC source packet (e.g., the UDP datagram containing this FEC source packet) will be able to recover the ADUI if FEC decoding succeeds. Thanks to the initial 3 bytes, this receiver will get rid of the padding (if any) and identify the corresponding ADU flow.

5. Simple Reed-Solomon FEC Scheme over GF(2^^m) for Arbitrary ADU Flows

This Fully-Specified FEC Scheme specifies the use of Reed-Solomon codes over GF(2^^m), with 2 <= m <= 16, with a simple FEC encoding for arbitrary packet flows.

5.1. Formats and Codes

5.1.1. FEC Framework Configuration Information

The FEC Framework Configuration Information (or FFCI) includes information that MUST be communicated between the sender and receiver(s). More specifically, it enables the synchronization of the FECFRAME sender and receiver instances. It includes both mandatory elements and scheme-specific elements, as detailed below.

5.1.1.1. Mandatory Information

FEC Encoding ID: the value assigned to this fully-specified FEC scheme MUST be XXX, as assigned by IANA (Section 7).
When SDP is used to communicate the FFCI, this FEC Encoding ID is carried in the 'encoding-id' parameter.

5.1.1.2. FEC Scheme-Specific Information

The FEC Scheme Specific Information (FSSI) includes elements that are specific to the present FEC scheme. More precisely:
Encoding symbol length (E): a non-negative integer that indicates either the length of each encoding symbol in bytes (strict mode, i.e., if S = 1), or the maximum length of any encoding symbol (i.e., if S = 0).
Strict (S) flag: when set to 1 this flag indicates that the E parameter is valid for the whole session, unless otherwise notified. When set to 0 this flag indicates that the E parameter is only the maximum length of each encoding symbol, for the whole session, unless otherwise notified.

m parameter (m): an integer that defines the length of the elements in the finite field, in bits. We have: 2 <= m <= 16.
These elements are required both by the sender (RS encoder) and the receiver(s) (RS decoder).

When SDP is used to communicate the FFCI, this FEC scheme-specific information is carried in the 'fssi' parameter in textual representation as specified in [SDP_ELEMENTS]. For instance:

fssi = E:1400,S:0,m:8

If another mechanism requires the FSSI to be carried as an opaque octet string (for instance after a Base64 encoding), the encoding format consists of the following 3 octets:
o Encoding symbol length (E): 16 bit field.
o Strict (S) flag: 1 bit field.
o m parameter (m): 7 bit field.

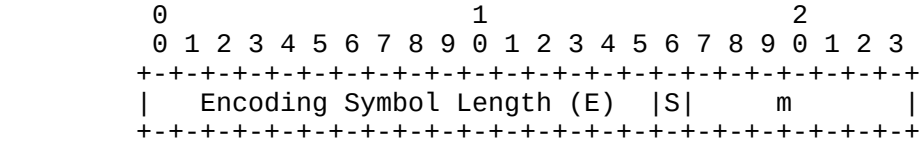

Figure 3: FSSI encoding format.

5.1.2. Explicit Source FEC Payload ID

A FEC source packet MUST contain an Explicit Source FEC Payload ID that is appended to the end of the packet as illustrated in Figure 4.

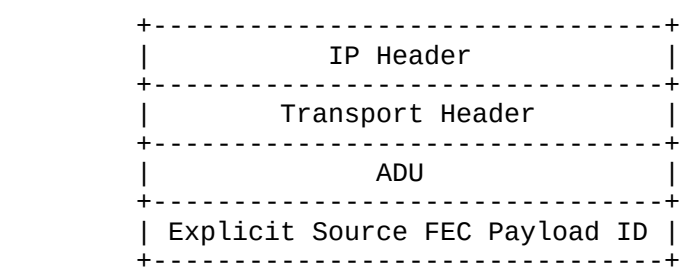

Figure 4: Structure of a FEC Source Packet with the Explicit Source FEC Payload ID.

More precisely, the Explicit Source FEC Payload ID is composed of the Source Block Number, the Encoding Symbol ID, and the Source Block Length. The length of the first two fields depends on the m parameter (transmitted separately in the FFCI, Section 5.1.1.2):

Source Block Number (SBN) (32-m bit field): this field identifies the source block to which this FEC source packet belongs.
Encoding Symbol ID (ESI) (m bit field): this field identifies the source symbol contained in this FEC source packet. This value is such that 0 <= ESI <= k - 1 for source symbols.
Source Block Length (k) (16 bit field): this field provides the number of source symbols for this source block, i.e., the k parameter. If 16 bits are too much when m <= 8, it is needed when 8 < m <= 16. Therefore we provide a single common format regardless of m.

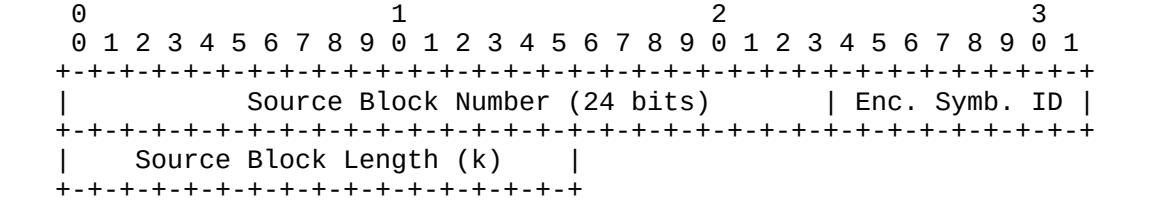

Figure 5: Source FEC Payload ID encoding format for m = 8 (default).

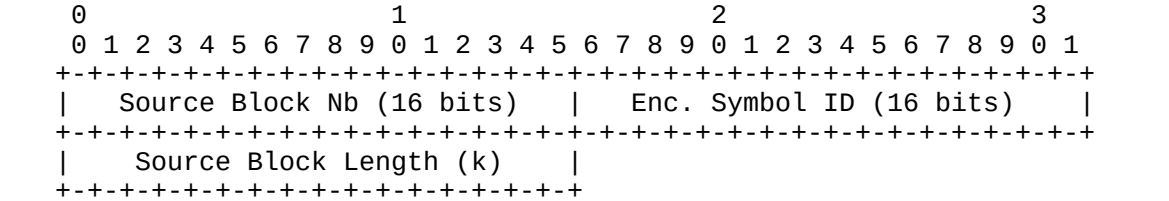

Figure 6: Source FEC Payload ID encoding format for m = 16.

The format of the Source FEC Payload ID for m = 8 and m = 16 are illustrated in Figure 5 and Figure 6 respectively.

5.1.3. Repair FEC Payload ID

A FEC repair packet MUST contain a Repair FEC Payload ID that is prepended to the repair symbol(s) as illustrated in Figure 7. There MUST be a single repair symbol per FEC repair packet.

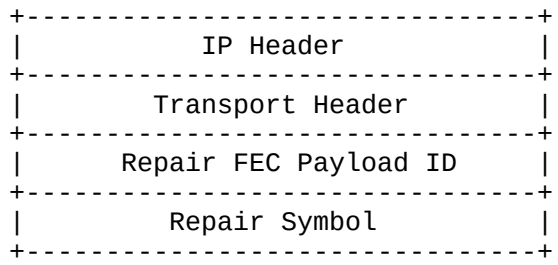

Figure 7: Structure of a FEC Repair Packet with the Repair FEC Payload ID.

More precisely, the Repair FEC Payload ID is composed of the Source Block Number, the Encoding Symbol ID, and the Source Block Length. The length of the first two fields depends on the m parameter (transmitted separately in the FFCI, Section 5.1.1.2):

Source Block Number (SBN) (32-m bit field): this field identifies the source block to which the FEC repair packet belongs.

Encoding Symbol ID (ESI) (m bit field) this field identifies the repair symbol contained in this FEC repair packet. This value is such that k <= ESI <= n - 1 for repair symbols.

Source Block Length (k) (16 bit field): this field provides the number of source symbols for this source block, i.e., the k parameter. If 16 bits are too much when m <= 8, it is needed when 8 < m <= 16. Therefore we provide a single common format regardless of m.

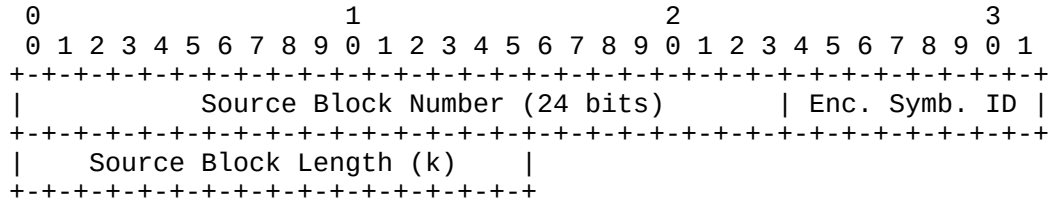

Figure 8: Repair FEC Payload ID encoding format for m = 8 (default).

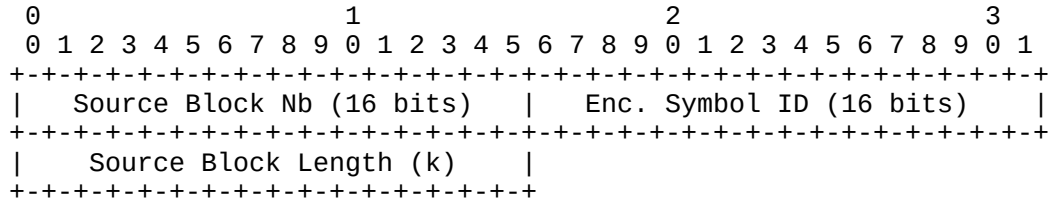

Figure 9: Repair FEC Payload ID encoding format for m = 16.

The format of the Repair FEC Payload ID for m = 8 and m = 16 are illustrated in Figure 8 and Figure 9 respectively.

5.2. Procedures

The following procedures apply:

- The source block creation procedures are specified in Section 4.3.
- The SBN value is incremented for each new source block, starting at 0 for the first block of the ADU flow. Wrapping to zero will happen for long sessions, after value 2^^(32-m) - 1.
- The ESI of encoding symbols is managed sequentially, starting at 0 for the first symbol. The first k values (0 <= ESI <= k - 1) identify source symbols, whereas the last n-k values (k <= ESI <= n - 1) identify repair symbols.
- The FEC repair packet creation procedures are specified in Section 5.1.3.

5.3. FEC Code Specification

The present document inherits from [RFC5510] the specification of the core Reed-Solomon codes based on Vandermonde matrices for a packet transmission channel.

6. Security Considerations

6.1. Problem Statement

A content delivery system is potentially subject to many attacks. Some of them target the network (e.g., to compromise the routing infrastructure, by compromising the congestion control component), others target the Content Delivery Protocol (CDP) (e.g., to compromise its normal behavior), and finally some attacks target the content itself. Since this document focuses on various FEC schemes, this section only discusses the additional threats that their use within the FECFRAME framework can create to an arbitrary CDP.

More specifically, these attacks may have several goals:

- those that are meant to give access to a confidential content (e.g., in case of a non-free content),
- those that try to corrupt the ADU Flows being transmitted (e.g., to prevent a receiver from using it),
- and those that try to compromise the receiver’s behavior (e.g., by making the decoding of an object computationally expensive).

These attacks can be launched either against the data flow itself (e.g., by sending forged FEC Source/Repair Packets) or against the FEC parameters that are sent either in-band (e.g., in the Repair FEC Payload ID) or out-of-band (e.g., in a session description).

6.2. Attacks Against the Data Flow

First of all, let us consider the attacks against the data flow.

6.2.1. Access to Confidential Contents

Access control to the ADU Flow being transmitted is typically provided by means of encryption. This encryption can be done within the content provider itself, by the application (for instance by using the Secure Real-time Transport Protocol (SRTP) [RFC3711]), or at the Network Layer, on a packet per packet basis when IPSec/ESP is used [RFC4303]. If confidentiality is a concern, it is RECOMMENDED that one of these solutions be used. Even if we mention these attacks here, they are not related nor facilitated by the use of FEC.

6.2.2. Content Corruption

Protection against corruptions (e.g., after sending forged FEC Source/Repair Packets) is achieved by means of a content integrity verification/sender authentication scheme. This service is usually provided at the packet level. In this case, after removing all forged packets, the ADU Flow may be sometimes recovered. Several techniques can provide this source authentication/content integrity service:

- at the application level, the Secure Real-time Transport Protocol (SRTP) [RFC3711] provides several solutions to authenticate the source and check the integrity of RTP and RTCP messages, among other services. For instance, associated to the Timed Efficient Stream Loss-Tolerant Authentication (TESLA) [RFC4383], SRTP is an attractive solution that is robust to losses, provides a true authentication/integrity service, and does not create any prohibitive processing load or transmission overhead. Yet, checking a packet requires a small delay (a second or more) after its reception with TESLA. Other building blocks can be used within SRTP to provide authentication/content integrity services.
- at the Network Layer, IPSec/ESP offers (among other services) an integrity verification mechanism that can be used to provide authentication/content integrity services.

It is up to the developer and the person in charge of deployment, who know the security requirements and features of the target application area, to define which solution is the most appropriate. Nonetheless it is RECOMMENDED that at least one of these techniques be used.

6.3. Attacks Against the FEC Parameters

Let us now consider attacks against the FEC parameters included in the FFCI that are usually sent out-of-band (e.g., in a session

description). Attacks on these FEC parameters can prevent the decoding of the associated object. For instance modifying the m field (when applicable) will lead a receiver to consider a different code. Modifying the E parameter will lead a receiver to consider bad Repair Symbols for a received FEC Repair Packet.

It is therefore RECOMMENDED that security measures be taken to guarantee the FFCI integrity. When the FFCI is sent out-of-band in a session description, this latter SHOULD be protected, for instance by digitally signing it.

Attacks are also possible against some FEC parameters included in the Explicit Source FEC Payload ID and Repair FEC Payload ID. For instance modifying the Source Block Number of a FEC Source of Repair Packet will lead a receiver to assign this packet to a wrong block.

It is therefore RECOMMENDED that security measures be taken to guarantee the Explicit Source FEC Payload ID and Repair FEC Payload ID integrity. To that purpose, one of the packet-level source authentication/content integrity techniques of Section 6.2.2 can be used.

7. IANA Considerations

The FEC Encoding ID value is subject to IANA registration.

TBD

8. Acknowledgments

The authors want to thank Hitoshi Asaeda for his valuable comments.

9. References

9.1. Normative References

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

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

[RFC5510] Lacan, J., Roca, V., Peltotalo, J., and S. Peltotalo, "Reed-Solomon Forward Error Correction (FEC) Schemes", RFC 5510, April 2009.

[FECFRAME-FRAMEWORK]
Watson, M., "Forward Error Correction (FEC) Framework", draft-ietf-fecframe-framework-10 (Work in Progress), September 2010.

[SDP_ELEMENTS]
Begen, A., "SDP Elements for FEC Framework", draft-ietf-fecframe-sdp-elements-10 (Work in Progress), October 2010.

9.2. Informative References

[RS-codec]
Rizzo, L., "Reed-Solomon FEC codec (revised version of July 2nd, 1998), available at http://info.iet.unipi.it/~luigi/vdm98/vdm980702.tgz and mirrored at http://planete-bcast.inrialpes.fr/", July 1998.

[Rizzo97] Rizzo, L., "Effective Erasure Codes for Reliable Computer Communication Protocols", ACM SIGCOMM Computer Communication Review Vol.27, No.2, pp.24-36, April 1997.

[RFC5170] Roca, V., Neumann, C., and D. Furodet, "Low Density Parity Check (LDPC) Forward Error Correction", RFC 5170, June 2008.

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

[RFC5775] Luby, M., Watson, M., and L. Vicisano, "Asynchronous Layered Coding (ALC) Protocol Instantiation", RFC 5775, April 2010.

[RFC5740] Adamson, B., Bormann, C., Handley, M., and J. Macker, "Negative-acknowledgment (NACK)-Oriented Reliable Multicast (NORM) Protocol", RFC 5740, November 2009.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

[RFC4383] Baugher, M. and E. Carrara, "The Use of Timed Efficient Stream Loss-Tolerant Authentication (TESLA) in the Secure

Real-time Transport Protocol (SRTP)", RFC 4383,
February 2006.

Authors' Addresses

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

Email: vincent.roca@inria.fr
URI: http://planete.inrialpes.fr/people/roca/

Mathieu Cunche
NICTA
Australia

Email: mathieu.cunche@nicta.com.au
URI: http://mathieu.cunche.free.fr/

Jerome Lacan
ISAE/LAAS-CNRS
1, place Emile Blouin
Toulouse 31056
France

Email: jerome.lacan@isae.fr
URI: http://dmi.ensica.fr/auteur.php3?id_auteur=5

Amine Bouabdallah
ISAE/LAAS-CNRS
1, place Emile Blouin
Toulouse 31056
France

Email: Amine.Bouabdallah@isae.fr
URI: http://dmi.ensica.fr/

Kazuhisa Matsuzono
Keio University
Graduate School of Media and Governance
5322 Endo
Fujisawa, Kanagawa 252-8520
Japan

Email: kazuhisa@sfc.wide.ad.jp