Forward Error Correction (FEC) Framework version 2

Vincent Roca, Ali Begen

TSVWG
Internet-Draft
Obsoletes: 6363 (if approved)
Intended status: Standards Track
Expires: April 9, 2017

V. Roca
INRIA
A. Begen
Networked Media

October 6, 2016

Forward Error Correction (FEC) Framework version 2
draft-roca-tsvwg-fecframev2-02

Abstract

This document describes a framework for using Forward Error Correction (FEC) codes with applications in public and private IP networks to provide protection against packet loss. The framework supports applying FEC to arbitrary packet flows over unreliable transport and is primarily intended for real-time, or streaming, media. This framework can be used to define Content Delivery Protocols that provide FEC for streaming media delivery or other packet flows. Content Delivery Protocols defined using this framework can support any FEC scheme (and associated FEC codes) that is compliant with various requirements defined in this document. Thus, Content Delivery Protocols can be defined that are not specific to a particular FEC scheme, and FEC schemes can be defined that are not specific to a particular Content Delivery Protocol. The first version of FECFRAME defined in RFC 6363 was restricted to block FEC codes. The FECFRAME version 2 defined in this document adds the possibility to use Convolutional FEC Codes in addition to Block FEC Codes. It obsoltes RFC 6363.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Many applications have a requirement to transport a continuous stream of packetized data from a source (sender) to one or more destinations (receivers) over networks that do not provide guaranteed packet delivery. Primary examples are real-time, or streaming, media applications such as broadcast, multicast, or on-demand forms of audio, video, or multimedia.

Forward Error Correction (FEC) is a well-known technique for improving the reliability of packet transmission over networks that do not provide guaranteed packet delivery, especially in multicast and broadcast applications. The FEC Building Block, defined in [RFC5052], provides a framework for the definition of Content Delivery Protocols (CDPs) for object delivery (including, primarily, file delivery) that make use of separately defined FEC schemes. Any CDP defined according to the requirements of the FEC Building Block can then easily be used with any FEC scheme that is also defined according to the requirements of the FEC Building Block. However [RFC5052] is restricted to block FEC codes, which means that the input flow(s) MUST be segmented into a sequence of blocks: FEC encoding (at a sender/coding node) must be performed on a per-block basis, and decoding (at a receiver/decoding node) MUST be performed independently on a per-block basis. This approach has a major impact on coding and decoding delays when used with block FEC codes (e.g., [RFC6681], [RFC6816] or [RFC6865]) since encoding requires that all the source symbols be known at the encoder. In case of continuous input flow(s), even if source symbols can be sent immediately, repair symbols are naturally delayed by the block creation time, that directly depends on the block size (i.e., the number of source symbols in this block, k). This block creation time is also the minimum decoding latency any receiver will experience in case of erasures, since no repair symbol for the current block can be received before. A good value for the block size is necessarily a good balance between the minimum decoding latency at the receivers (which must be in line with the most stringent real-time requirement of the flow(s)) and the desired robustness against long erasure bursts (which depends on the block size).

On the opposite, a convolutional code associated to a sliding encoding window (of fixed size) or a sliding elastic encoding window (of variable size) removes this minimum decoding delay, since repair symbols can be generated and sent on-the-fly, at any time, from the source symbols present in the current coding window. Using a sliding encoding window mode is therefore highly beneficial to real-time flows, one of the primary targets of FECFRAME. [FECFRAMEv2-Motivations] discusses more in detail the motivations behind this document.

Note that the term "Forward Erasure Correction" is sometimes used, erasures being a type of error in which data is lost and this loss can be detected, rather than being received in corrupted form. The focus of this document is strictly on erasures, and the term "Forward Error Correction" is more widely used.

This document defines a framework for the definition of CDPs that provide for FEC protection for arbitrary packet flows over unreliable transports such as UDP, using either block FEC codes as in [RFC6363] (i.e., the original FECFRAME, also called FECFRAME version 1 in this document), or convolutional FEC codes that is specific to FECFRAME version 2 described in this document. As such, when used with block FEC codes, this document complements the FEC Building Block of [RFC5052], by providing for the case of arbitrary packet flows over unreliable transport, the same kind of framework as that document provides for object delivery. This document does not define a complete CDP; rather, it defines only those aspects that are expected to be common to all CDPs based on this framework.

This framework does not define how the flows to be protected are determined, nor does it define how the details of the protected flows and the FEC streams that protect them are communicated from sender to receiver. It is expected that any complete CDP specification that makes use of this framework will address these signaling requirements. However, this document does specify the information that is required by the FEC Framework at the sender and receiver, e.g., details of the flows to be FEC protected, the flow(s) that will carry the FEC protection data, and an opaque container for FEC-Scheme-Specific Information.

FEC schemes designed for use with this framework must fulfill a number of requirements defined in this document. These requirements are different from those defined in [RFC5052] for FEC schemes for object delivery. However, there is a great deal of commonality, and FEC schemes defined for object delivery may be easily adapted for use with the framework defined in this document.

Since RTP [RFC3550] is (often) used over UDP, this framework can be applied to RTP flows as well. FEC repair packets may be sent directly over UDP or RTP. The latter approach has the advantage that RTP instrumentation, based on the RTP Control Protocol (RTCP), can be used for the repair flow. Additionally, the post-repair RTCP extended reports [RFC5725] may be used to obtain information about the loss rate after FEC recovery.

The use of RTP for repair flows is defined for each FEC scheme by defining an RTP payload format for that particular FEC scheme (possibly in the same document).

Editor’s notes:

- o FECFRAME does not define any header/trailer (but FEC Schemes do) and there is no "version" field that could be used to signal this is FECFRAME version 2 and not version 1. Therefore the notion of "version" is purely abstract and could be removed altogether without affecting FECFRAME interoperability at all. Indeed, a receiver that only supports FECFRAME "version 1" FEC Schemes will not join a session for which the SDP file indicates an unsupported (e.g., Convolutional) FEC Scheme, since this receiver will be able to process neither the FEC Source Packets nor FEC Repair Packets. This is exactly the same behavior when a receiver wants to join a FECFRAME version 1 session with an unsupported Block FEC Scheme. From this point of view, FECFRAME version 2 extends the applicability of FECFRAME to new types of FEC codes in a fully backward compatible way. However, supporting these new FEC codes does impact the FECFRAME software: implementation is seriously impacted due to different working modes, the notion of sliding encoding/decoding window being added to that of source block. From this point of view, adding the notion of version to FECFRAME makes sense to easily identify some of the capabilities of a FECFRAME implementation. The current document uses the notion of version for the sake of clarity only.

- o Writing an I-D equivalent to [RFC5052] and focused on convolutional FEC codes remains to be done.

2. Definitions and Abbreviations

Application Data Unit (ADU): The unit of source data provided as payload to the transport layer.

ADU Flow: A sequence of ADUs associated with a transport-layer flow identifier (such as the standard 5-tuple {source IP address, source port, destination IP address, destination port, transport protocol}).

AL-FEC: Application-layer Forward Error Correction.

Application Protocol: Control protocol used to establish and control the source flow being protected, e.g., the Real-Time Streaming Protocol (RTSP).

Content Delivery Protocol (CDP): A complete application protocol specification that, through the use of the framework defined in this document, is able to make use of FEC schemes to provide FEC capabilities.

FEC Code: An algorithm for encoding data such that the encoded data flow is resilient to data loss. Note that, in general, FEC codes may also be used to make a data flow resilient to corruption, but that is not considered in this document.

Block FEC Code: FEC Code that operate in a block manner, i.e., for which the input flow MUST be segmented into a sequence of blocks, FEC encoding and decoding being performed independently on a per-block basis.

Convolutional FEC Code: FEC Code that can generate repair symbols on-the-fly, at any time, from the source symbols present in the current encoding window.

FEC Framework: A protocol framework for the definition of Content Delivery Protocols using FEC, such as the framework defined in this document.

FEC Framework Configuration Information: Information that controls the operation of the FEC Framework.

FEC Payload ID: Information that identifies the contents of a packet with respect to the FEC scheme.

FEC Repair Packet: At a sender (respectively, at a receiver), a payload submitted to (respectively, received from) the transport protocol containing one or more repair symbols along with a Repair FEC Payload ID and possibly an RTP header.

FEC Scheme: A specification that defines the additional protocol aspects required to use a particular FEC code with the FEC Framework.

FEC Source Packet: At a sender (respectively, at a receiver), a payload submitted to (respectively, received from) the transport protocol containing an ADU along with an optional Explicit Source FEC Payload ID.

Protection Amount: The relative increase in data sent due to the use of FEC.

Repair Flow: The packet flow carrying FEC data.

Repair FEC Payload ID: A FEC Payload ID specifically for use with repair packets.

Source Flow: The packet flow to which FEC protection is to be applied. A source flow consists of ADUs.

Source FEC Payload ID: A FEC Payload ID specifically for use with source packets.

Source Protocol: A protocol used for the source flow being protected, e.g., RTP.

Transport Protocol: The protocol used for the transport of the source and repair flows, e.g., UDP and the Datagram Congestion Control Protocol (DCCP).

Encoding Window: Set of Source Symbols available at the sender/coding node that are used to generate a repair symbol, with a Convolutional FEC Code.

Decoding Window: Set of received or decoded source and repair symbols available at a receiver that are used to decode erased source symbols, with a Convolutional FEC Code.

The following definitions are aligned with [RFC5052]. Unless otherwise mentioned, they apply both to Block and Convolutional FEC Codes:

Code Rate: 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.

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

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.

Repair Symbol: Encoding symbol that is not a source symbol.

Source Block: Group of ADUs that are to be FEC protected as a single block. This notion is restricted to Block FEC Codes.

Source Symbol: Unit of data used during the encoding process.

Systematic Code: FEC code in which the source symbols are part of the encoding symbols.

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. Architecture Overview

The FEC Framework is described in terms of an additional layer between the transport layer (e.g., UDP or DCCP) and protocols running over this transport layer. As such, the data path interface between the FEC Framework and both underlying and overlying layers can be thought of as being the same as the standard interface to the transport layer; i.e., the data exchanged consists of datagram payloads each associated with a single ADU flow identified by the standard 5-tuple {source IP address, source port, destination IP address, destination port, transport protocol}. In the case that RTP is used for the repair flows, the source and repair data can be multiplexed using RTP onto a single UDP flow and needs to be consequently demultiplexed at the receiver. There are various ways in which this multiplexing can be done (for example, as described in [RFC4588]).

It is important to understand that the main purpose of the FEC Framework architecture is to allocate functional responsibilities to separately documented components in such a way that specific instances of the components can be combined in different ways to describe different protocols.

The FEC Framework makes use of a FEC scheme, in a similar sense to that defined in [RFC5052] in case of Block FEC Codes, and uses the terminology of that document. The FEC scheme defines the FEC encoding and decoding, and it defines the protocol fields and procedures used to identify packet payload data in the context of the FEC scheme. The interface between the FEC Framework and a FEC scheme, which is described in this document, is a logical one that exists for specification purposes only. At an encoder, the FEC Framework passes ADUs to the FEC scheme for FEC encoding. The FEC scheme returns repair symbols with their associated Repair FEC Payload IDs and, in some cases, Source FEC Payload IDs, depending on the FEC scheme. At a decoder, the FEC Framework passes transport packet payloads (source and repair) to the FEC scheme, and the FEC scheme returns additional recovered source packet payloads.

This document defines certain FEC Framework Configuration Information that MUST be available to both sender and receiver(s). For example, this information includes the specification of the ADU flows that are to be FEC protected, specification of the ADU flow(s) that will carry the FEC protection (repair) data, and the relationship(s) between these source and repair flows (i.e., which source flow(s) are protected by repair flow(s)). The FEC Framework Configuration Information also includes information fields that are specific to the FEC scheme. This information is analogous to the FEC Object Transmission Information defined in [RFC5052].

The FEC Framework does not define how the FEC Framework Configuration Information for the stream is communicated from sender to receiver. This has to be defined by any CDP specification, as described in the following sections.

In this architecture, we assume that the interface to the transport layer supports the concepts of data units (referred to here as Application Data Units (ADUs)) to be transported and identification of ADU flows on which those data units are transported. Since this is an interface internal to the architecture, we do not specify this interface explicitly. We do require that ADU flows that are distinct from the transport layer point of view (for example, distinct UDP flows as identified by the UDP source/destination addresses/ports) are also distinct on the interface between the transport layer and the FEC Framework.

As noted above, RTP flows are a specific example of ADU flows that might be protected by the FEC Framework. From the FEC Framework point of view, RTP source flows are ADU flows like any other, with the RTP header included within the ADU.

Depending on the FEC scheme, RTP can also be used as a transport for repair packet flows. In this case, a FEC scheme has to define an RTP payload format for the repair data.

The architecture outlined above is illustrated in Figure 1. In this architecture, two (optional) RTP instances are shown, for the source and repair data, respectively. This is because the use of RTP for the source data is separate from, and independent of, the use of RTP for the repair data. The appearance of two RTP instances is more natural when one considers that in many FEC codes, the repair payload contains repair data calculated across the RTP headers of the source packets. Thus, a repair packet carried over RTP starts with an RTP header of its own, which is followed (after the Repair Payload ID) by repair data containing bytes that protect the source RTP headers (as well as repair data for the source RTP payloads).

```
     +--------------------------------------------+
     |                 Application                |
     +--------------------------------------------+
                           |
                           |
                           |
   + - - - - - - - - - - - - - - - - - - - - - - - -+
   | +--------------------------------------------+ |
     |            Application Layer               |
   | +--------------------------------------------+ |
                           |                |
   | + -- -- -- -- -- -- -- -- -- -- --+     |      |
     |           RTP (Optional)          |     |
   | |                                  |     |- Configuration/
     +- -- -- -- -- -- -- -- -- -- -- -+     |  Coordination
   |                    |                     |      |
                        | ADU flows          |
   |                    |                    v       |
     +--------------------------------------------+     +------------+
   | |      FEC Framework (This document)         |<--->| FEC Scheme |
     +--------------------------------------------+     +------------+
   |                |               |                |
            Source |         Repair |
   |                |               |                |
     +-- -- -- -- --|-- --+ -- -- -- -- -- + -- --+
   | | RTP Layer    |     | RTP Processing |      | |
     | (Optional)   |     +-- -- -- |- -- -+      |
   | |        +-- -- -- -- -- -- -- |--+          | |
     |        |  RTP (De)multiplexing  |          |
   | +-- -- -- --- -- -- -- -- -- -- -- -- -- -- -+ |
                           |
   | +--------------------------------------------+ |
     |          Transport Layer (e.g., UDP)       |
   | +--------------------------------------------+ |
                           |
   | +--------------------------------------------+ |
     |                      IP                    |
   | +--------------------------------------------+ |

   | Content Delivery Protocol                       |
   + - - - - - - - - - - - - - - - - -  - - - - - - - +
```

Figure 1: FEC Framework Architecture

The content of the transport payload for repair packets is fully defined by the FEC scheme. For a specific FEC scheme, a means MAY be defined for repair data to be carried over RTP, in which case, the repair packet payload format starts with the RTP header. This corresponds to defining an RTP payload format for the specific FEC scheme.

The use of RTP for repair packets is independent of the protocols used for source packets: if RTP is used for source packets, repair packets may or may not use RTP and vice versa (although it is unlikely that there are useful scenarios where non-RTP source flows are protected by RTP repair flows). FEC schemes are expected to recover entire transport payloads for recovered source packets in all cases. For example, if RTP is used for source flows, the FEC scheme is expected to recover the entire UDP payload, including the RTP header.

4. Procedural Overview

4.1. General

The mechanism defined in this document does not place any restrictions on the ADUs that can be protected together, except that the ADU be carried over a supported transport protocol (see Section 7). The data can be from multiple source flows that are protected jointly. For instance, with a Block FEC Code, the FEC Framework handles the source flows as a sequence of source blocks each consisting of a set of ADUs, possibly from multiple source flows that are to be protected together. For example, each source block can be constructed from those ADUs related to a particular segment in time of the flow.

At the sender, with a Block FEC Code, the FEC Framework passes the payloads for a given block to the FEC scheme for FEC encoding. With a Convolutional FEC Code, the FEC Framework passes the payloads currently present in the Encoding Window to the FEC scheme for FEC encoding. Then the FEC scheme performs the FEC encoding operation and returns the following information:

- o Optionally, FEC Payload IDs for each of the source payloads (encoded according to a FEC-Scheme-Specific format).
- o One or more FEC repair packet payloads.
- o FEC Payload IDs for each of the repair packet payloads (encoded according to a FEC-Scheme-Specific format).

The FEC Framework then performs two operations. First, it appends the Source FEC Payload IDs, if provided, to each of the ADUs, and sends the resulting packets, known as "FEC source packets", to the receiver. Second, it places the provided FEC repair packet payloads and corresponding Repair FEC Payload IDs appropriately to construct FEC repair packets and send them to the receiver.

This document does not define how the sender determines which ADUs are included in which source blocks (in case of a Block FEC Code) or in the Encoding Window (in case of a Convolutional FEC Code), or the sending order and timing of FEC source and repair packets. A specific CDP MAY define this mapping, or it MAY be left as implementation dependent at the sender. However, a CDP specification MUST define how a receiver determines a minimum length of time that it needs to wait to receive FEC repair packets for any given source block. FEC schemes MAY define limitations on this mapping (such as maximum size of source blocks with a Block FEC Code), but they SHOULD NOT attempt to define specific mappings. The sequence of operations at the sender is described in more detail in Section 4.2.

At the receiver, original ADUs are recovered by the FEC Framework directly from any FEC source packets received simply by removing the Source FEC Payload ID, if present. The receiver also passes the contents of the received ADUs, plus their FEC Payload IDs, to the FEC scheme for possible decoding.

If any ADUs have been lost, then the FEC scheme can perform FEC decoding to recover the missing ADUs (assuming sufficient FEC source and repair packets related to that source block have been received).

Note that the receiver might need to buffer received source packets to allow time for the FEC repair packets to arrive and FEC decoding to be performed before some or all of the received or recovered packets are passed to the application. If such a buffer is not provided, then the application has to be able to deal with the severe re-ordering of packets that can occur. However, such buffering is CDP- and/or implementation-specific and is not specified here. The receiver operation is described in more detail in Section 4.3.

With a Block FEC Code, the FEC source packets MUST contain information that identifies the source block and the position within the source block (in terms specific to the FEC scheme) occupied by the ADU. Similarly, with a Convolutional FEC Code, the FEC source packet MUST contain information to identify the position within the source flow (in terms specific to the FEC scheme) occupied by the ADU. In both cases this information is known as the Source FEC Payload ID. The FEC scheme is responsible for defining and interpreting this information. This information MAY be encoded into a specific field within the FEC source packet format defined in this specification, called the Explicit Source FEC Payload ID field. The exact contents and format of the Explicit Source FEC Payload ID field are defined by the FEC schemes. Alternatively, the FEC scheme MAY define how the Source FEC Payload ID is derived from other fields within the source packets. This document defines the way that the Explicit Source FEC Payload ID field is appended to source packets to form FEC source packets.

With a Block FEC Code, the FEC repair packets MUST contain information that identifies the source block and the relationship between the contained repair payloads and the original source block. Similarly, with a Convolutional FEC Code, the FEC repair packets MUST contain information that identifies the relationship between the contained repair payloads and the original source symbols used during encoding. In both cases this is known as the Repair FEC Payload ID. This information MUST be encoded into a specific field, the Repair FEC Payload ID field, the contents and format of which are defined by the FEC schemes.

The FEC scheme MAY use different FEC Payload ID field formats for source and repair packets.

4.2. Sender Operation with Block FEC Codes

It is assumed that the sender has constructed or received original data packets for the session. These could be carrying any type of data. The following operations, illustrated in Figure 2 for the case of UDP repair flows and in Figure 3 for the case of RTP repair flows, describe a possible way to generate compliant source and repair flows:

1. ADUs are provided by the application.

2. A source block is constructed as specified in Section 5.2.

3. The source block is passed to the FEC scheme for FEC encoding. The Source FEC Payload ID information of each source packet is determined by the FEC scheme. If required by the FEC scheme, the Source FEC Payload ID is encoded into the Explicit Source FEC Payload ID field.

4. The FEC scheme performs FEC encoding, generating repair packet payloads from a source block and a Repair FEC Payload ID field for each repair payload.

5. The Explicit Source FEC Payload IDs (if used), Repair FEC Payload IDs, and repair packet payloads are provided back from the FEC scheme to the FEC Framework.

6. The FEC Framework constructs FEC source packets according to Section 5.3, and FEC repair packets according to Section 5.4,

using the FEC Payload IDs and repair packet payloads provided by the FEC scheme.

7. The FEC source and repair packets are sent using normal transport-layer procedures. The port(s) and multicast group(s) to be used for FEC repair packets are defined in the FEC Framework Configuration Information. The FEC source packets are sent using the same ADU flow identification information as would have been used for the original source packets if the FEC Framework were not present (for example, in the UDP case, the UDP source and destination addresses and ports on the IP datagram carrying the source packet will be the same whether or not the FEC Framework is applied).

```
+----------------------+
|     Application      |
+----------------------+
           |
           |(1) ADUs
           |
           v
+----------------------+                           +----------------+
|    FEC Framework     |                           |                |
|                      |-------------------------->|   FEC Scheme   |
|(2) Construct source  |(3) Source Block           |                |
|    blocks            |                           |(4) FEC Encoding|
|(6) Construct FEC     |<--------------------------|                |
|    source and repair |                           |                |
|    packets           |(5) Explicit Source FEC    |                |
+----------------------+    Payload IDs            +----------------+
           |                Repair FEC Payload IDs
           |                Repair symbols
           |
           |(7) FEC source and repair packets
           v
+----------------------+
|   Transport Layer    |
|     (e.g., UDP)      |
+----------------------+
```

Figure 2: Sender Operation with Block FEC Codes

```
   +----------------------+
   |     Application      |
   +----------------------+
              |
              |(1) ADUs
              |
              v
   +----------------------+                           +----------------+
   |    FEC Framework     |                           |                |
   |                      |-------------------------->|   FEC Scheme   |
   |(2) Construct source  |(3) Source Block           |                |
   |    blocks            |                           |(4) FEC Encoding|
   |(6) Construct FEC     |<--------------------------|                |
   |    source packets and|                           |                |
   |    repair payloads   |(5) Explicit Source FEC    |                |
   +----------------------+    Payload IDs            +----------------+
       |             |         Repair FEC Payload IDs
       |             |         Repair symbols
       |             |
       |(7) Source   |(7’) Repair payloads
       |    packets  |
       |             |
       |      + -- -- -- -- -+
       |      |     RTP      |
       |      +-- -- -- -- --+
       v             v
   +----------------------+
   |   Transport Layer    |
   |     (e.g., UDP)      |
   +----------------------+
```

Figure 3: Sender Operation with RTP Repair Flows with Block FEC Codes

4.3. Receiver Operation with Block FEC Codes

The following describes a possible receiver algorithm, illustrated in Figures 4 and 5 for the case of UDP and RTP repair flows, respectively, when receiving a FEC source or repair packet:

1. FEC source packets and FEC repair packets are received and passed to the FEC Framework. The type of packet (source or repair) and the source flow to which it belongs (in the case of source packets) are indicated by the ADU flow information, which identifies the flow at the transport layer.

   In the special case that RTP is used for repair packets, and source and repair packets are multiplexed onto the same UDP flow, then RTP demultiplexing is required to demultiplex source and repair flows. However, RTP processing is applied only to the repair packets at this stage; source packets continue to be handled as UDP payloads (i.e., including their RTP headers).

2. The FEC Framework extracts the Explicit Source FEC Payload ID field (if present) from the source packets and the Repair FEC Payload ID from the repair packets.

3. The Explicit Source FEC Payload IDs (if present), Repair FEC Payload IDs, and FEC source and repair payloads are passed to the FEC scheme.

4. The FEC scheme uses the received FEC Payload IDs (and derived FEC Source Payload IDs in the case that the Explicit Source FEC Payload ID field is not used) to group source and repair packets into source blocks. If at least one source packet is missing from a source block, and at least one repair packet has been received for the same source block, then FEC decoding can be performed in order to recover missing source payloads. The FEC scheme determines whether source packets have been lost and whether enough data for decoding of any or all of the missing source payloads in the source block has been received.

5. The FEC scheme returns the ADUs to the FEC Framework in the form of source blocks containing received and decoded ADUs and indications of any ADUs that were missing and could not be decoded.

6. The FEC Framework passes the received and recovered ADUs to the application.

The description above defines functionality responsibilities but does not imply a specific set of timing relationships. Source packets that are correctly received and those that are reconstructed MAY be delivered to the application out of order and in a different order from the order of arrival at the receiver. Alternatively, buffering and packet re-ordering MAY be applied to re-order received and reconstructed source packets into the order they were placed into the source block, if that is necessary according to the application.

```
+----------------------+
|     Application      |
+----------------------+
           ^
           |
           |(6) ADUs
           |
+----------------------+                           +----------------+
|    FEC Framework     |                           |                |
|                      |<--------------------------|   FEC Scheme   |
|(2)Extract FEC Payload|(5) ADUs                   |                |
|   IDs and pass IDs & |                           |(4) FEC Decoding|
|   payloads to FEC    |-------------------------->|                |
|   scheme             |(3) Explicit Source FEC    |                |
+----------------------+    Payload IDs            +----------------+
           ^                Repair FEC Payload IDs
           |                Source payloads
           |                Repair payloads
           |
           |(1) FEC source and repair packets
           |
+----------------------+
|   Transport Layer    |
|     (e.g., UDP)      |
+----------------------+
```

Figure 4: Receiver Operation with Block FEC Codes or Convolutional FEC Codes

```
   +----------------------+
   |     Application      |
   +----------------------+
              ^
              |
              |(6) ADUs
              |
   +----------------------+                           +----------------+
   |    FEC Framework     |                           |                |
   |                      |<--------------------------|   FEC Scheme   |
   |(2)Extract FEC Payload|(5) ADUs                   |                |
   |   IDs and pass IDs & |                           |(4) FEC Decoding|
   |   payloads to FEC    |-------------------------->|                |
   |   scheme             |(3) Explicit Source FEC    |                |
   +----------------------+    Payload IDs            +----------------+
       ^             ^         Repair FEC Payload IDs
       |             |         Source payloads
       |             |         Repair payloads
       |             |
       |Source       |Repair payloads
       |packets      |
       |             |
   +-- |- -- -- -- -- -- -+
   |RTP| | RTP Processing |
   |   | +-- -- -- --|-- -+
   | +-- -- -- -- -- |--+ |
   | | RTP Demux       | |
   +-- -- -- -- -- -- -- -+
              ^
              |(1) FEC source and repair packets
              |
   +----------------------+
   |   Transport Layer    |
   |     (e.g., UDP)      |
   +----------------------+
```

Figure 5: Receiver Operation with RTP Repair Flows with Block FEC Codes or Convolutional FEC Codes

Note that the above procedure might result in a situation in which not all ADUs are recovered.

4.4. Sender Operation with Convolutional FEC Codes

Let us now consider FECFRAME version 2 using a Convolutional FEC Code. The following operations, illustrated in Figure 6 for the case of UDP repair flows and in Figure 7 for the case of RTP repair flows, describe a possible way to generate compliant source and repair flows:

1. A new ADU is provided by the application.

2. The FEC Framework communicates this ADU to the FEC scheme.

3. The (sliding) encoding window is updated by the FEC scheme. The ADU to source symbols mapping as well as the encoding window management details are the responsibility of the FEC scheme. However Appendix A provide some hints on the way it might be performed.

4. The Source FEC Payload ID information of the source packet is determined by the FEC scheme. If required by the FEC scheme, the Source FEC Payload ID is encoded into the Explicit Source FEC Payload ID field and returned to the FEC Framework.

5. The FEC Framework constructs the FEC source packet according to Section 5.3, using the Explicit Source FEC Payload ID provided by the FEC scheme if applicable.

6. The FEC source packet is sent using normal transport-layer procedures. This packet is sent using the same ADU flow identification information as would have been used for the original source packet if the FEC Framework were not present (for example, in the UDP case, the UDP source and destination addresses and ports on the IP datagram carrying the source packet will be the same whether or not the FEC Framework is applied).

7. When the FEC Framework needs to send one or several FEC repair packets (e.g., according to the target Code Rate), it asks the FEC scheme to create one or several repair packet payloads from the current sliding encoding window along with their Repair FEC Payload ID.

8. The Repair FEC Payload IDs and repair packet payloads are provided back from the FEC scheme to the FEC Framework.

9. The FEC Framework constructs FEC repair packets according to Section 5.4, using the FEC Payload IDs and repair packet payloads provided by the FEC scheme.

10. The FEC repair packets are sent using normal transport-layer procedures. The port(s) and multicast group(s) to be used for FEC repair packets are defined in the FEC Framework Configuration Information.

```
+----------------------+
|     Application      |
+----------------------+
           |
           | (1) New Application Data Unit (ADU)
           v
+---------------------+                           +----------------+
| FEC Framework ver.2 |                           |   FEC Scheme   |
|                     |-------------------------->|                |
|                     | (2) New ADU               |(3) Update of   |
|                     |                           |    encoding    |
|                     |<--------------------------|    window      |
|(5) Construct FEC    | (4) Explicit Source       |                |
|    source packet    |     FEC Payload ID(s)     |(7) FEC         |
|                     |<--------------------------|    encoding    |
|(9) Construct FEC    | (8) Repair FEC Payload ID |                |
|    repair packet(s) |     + Repair symbol(s)    |                |
+---------------------+                           +----------------+
           |
           | (6)  FEC source packet
           |
           | (10) FEC repair packets
           v
+----------------------+
|   Transport Layer    |
|     (e.g., UDP)      |
+----------------------+
```

Figure 6: Sender Operation with Convolutional FEC Codes

```
   +----------------------+
   |     Application      |
   +----------------------+
              |
              | (1) New Application Data Unit (ADU)
              v
   +---------------------+                           +----------------+
   | FEC Framework ver.2 |                           |   FEC Scheme   |
   |                     |-------------------------->|                |
   |                     | (2) New ADU               |(3) Update of   |
   |                     |                           |    encoding    |
   |                     |<--------------------------|    window      |
   |(5) Construct FEC    | (4) Explicit Source       |                |
   |    source packet    |     FEC Payload ID(s)     |(7) FEC         |
   |                     |<--------------------------|    encoding    |
   |(9) Construct FEC    | (8) Repair FEC Payload ID |                |
   |    repair packet(s) |     + Repair symbol(s)    |                |
   +---------------------+                           +----------------+
       |             |
       |(6) Source   |(10) Repair payloads
       |    packets  |
       |             |
       |      + -- -- -- -- -+
       |      |     RTP      |
       |      +-- -- -- -- --+
       v             v
   +----------------------+
   |   Transport Layer    |
   |     (e.g., UDP)      |
   +----------------------+
```

Figure 7: Sender Operation with RTP Repair Flows with Convolutional FEC Codes

4.5. Receiver Operation with Convolutional FEC Codes

The following describes a possible receiver algorithm in the case of Convolutional FEC Code. Figures 4 and 5 for the case of UDP and RTP repair flows, respectively, when receiving a FEC source or repair packet also apply here. The only difference lies in step (4):

4. The FEC scheme uses the received FEC Payload IDs (and derived FEC Source Payload IDs in the case that the Explicit Source FEC Payload ID field is not used) to insert source and repair packets into the decoding window in the right way. If at least one source packet is missing and at least one repair packet has been received and the rank of the associated linear system permits it, then FEC decoding can be performed in order to recover missing source

payloads. The FEC scheme determines whether source packets have been lost and whether enough data for decoding of any or all of the missing source payloads in the decoding window has been received.

Not shown in these Figures is the management of the decoding window at a receiver. For instance this decoding window is composed of a set of linear equations (we are using a linear FEC code) associated to each FEC repair packet received, and whose variables are the available (i.e., received or decoded) or unknown source symbols associated to ADUs. The decoding window is under the control of the FEC scheme and management details MUST be specified by the FEC scheme.

5. Protocol Specification

5.1. General

This section specifies the protocol elements for the FEC Framework. Three components of the protocol are defined in this document and are described in the following sections:

1. With a Block FEC Code, construction of a source block from ADUs. The FEC code will be applied to this source block to produce the repair payloads.

2. A format for packets containing source data.

3. A format for packets containing repair data.

The operation of the FEC Framework is governed by certain FEC Framework Configuration Information, which is defined in this section. A complete protocol specification that uses this framework MUST specify the means to determine and communicate this information between sender and receiver.

Note that the FEC Framework does not specify the management of the encoding window. This is left to the FEC scheme associated to a Convolutional FEC Code. This is motivated by the links that exist between the encoding window management features and the FEC scheme signaling features. For instance, an encoding window that is composed of a non sequential set of ADUs may require an appropriate signaling to inform a FEC Framework receiver of the identity of each ADU composing the encoding window. On the opposite, an encoding window always composed of a sequential set of ADUs simplifies signaling. For instance, providing the identity of the first ADU (or first source symbol of this ADU) and the number of ADUs (or source symbols) used to generate a FEC repair packet is sufficient to identify all the ADUs (or source symbols) present in the encoding window. Appendix A gives an example of encoding window management (non normative text).

Similarly the FEC Framework does not specify the management of the decoding window which is also left to the FEC scheme associated to a Convolutional FEC Code.

Note that the FEC Framework does not specify the ADU to source symbol mapping, neither for Block FEC Codes nor for Convolutional FEC Codes.

5.2. Structure of the Source Block with Block FEC Codes

The FEC Framework and FEC scheme exchange ADUs in the form of source blocks. A source block is generated by the FEC Framework from an ordered sequence of ADUs. The allocation of ADUs to blocks is dependent on the application. Note that some ADUs may not be included in any block. Each source block provided to the FEC scheme consists of an ordered sequence of ADUs where the following information is provided for each ADU:

- A description of the source flow with which the ADU is associated.
- The ADU itself.
- The length of the ADU.

5.3. Packet Format for FEC Source Packets

The packet format for FEC source packets MUST be used to transport the payload of an original source packet. As depicted in Figure 8, it consists of the original packet, optionally followed by the Explicit Source FEC Payload ID field. The FEC scheme determines whether the Explicit Source FEC Payload ID field is required. This determination is specific to each ADU flow.

```
+------------------------------------+
|              IP Header             |
+------------------------------------+
|           Transport Header         |
+------------------------------------+
|         Application Data Unit      |
+------------------------------------+
|    Explicit Source FEC Payload ID  |
+------------------------------------+
```

Figure 8: Structure of the FEC Packet Format for FEC Source Packets

The FEC source packets MUST be sent using the same ADU flow as would have been used for the original source packets if the FEC Framework were not present. The transport payload of the FEC source packet MUST consist of the ADU followed by the Explicit Source FEC Payload ID field, if required.

The Explicit Source FEC Payload ID field contains information required to associate the source packet with a source block (in case of Block FEC Code) or to the source flow (in case of Convolutional FEC code) and for the operation of the FEC algorithm, and is defined by the FEC scheme. The format of the Source FEC Payload ID field is defined by the FEC scheme. In the case that the FEC scheme or CDP defines a means to derive the Source FEC Payload ID from other information in the packet (for example, a sequence number used by the application protocol), then the Source FEC Payload ID field is not included in the packet. In this case, the original source packet and FEC source packet are identical.

In applications where avoidance of IP packet fragmentation is a goal, CDPs SHOULD consider the Explicit Source FEC Payload ID size when determining the size of ADUs that will be delivered using the FEC Framework. This is because the addition of the Explicit Source FEC Payload ID increases the packet length.

The Explicit Source FEC Payload ID is placed at the end of the packet, so that in the case that Robust Header Compression (ROHC) [RFC3095] or other header compression mechanisms are used, and in the case that a ROHC profile is defined for the protocol carried within the transport payload (for example, RTP), then ROHC will still be applied for the FEC source packets. Applications that are used with this framework need to consider that FEC schemes can add this Explicit Source FEC Payload ID and thereby increase the packet size.

In many applications, support for FEC is added to a pre-existing protocol, and in this case, use of the Explicit Source FEC Payload ID can break backward compatibility, since source packets are modified.

5.3.1. Generic Explicit Source FEC Payload ID

In order to apply FEC protection using multiple FEC schemes to a single source flow, all schemes have to use the same Explicit Source FEC Payload ID format. In order to enable this, it is RECOMMENDED that FEC schemes support the Generic Explicit Source FEC Payload ID format described below.

The Generic Explicit Source FEC Payload ID has a length of two octets and consists of an unsigned packet sequence number in network-byte order. The allocation of sequence numbers to packets is independent of any FEC scheme and of the source block construction or encoding window management, except that the use of this sequence number places a constraint on source block construction or encoding window management. Source packets within a given source block or encoding window MUST have consecutive sequence numbers (where consecutive includes wrap-around from the maximum value that can be represented in two octets (65535) to 0). Sequence numbers SHOULD NOT be reused until all values in the sequence number space have been used.

Note that if the original packets of the source flow are already carrying a packet sequence number that is at least two bytes long, there is no need to add the generic Explicit Source FEC Payload ID and modify the packets.

5.4. Packet Format for FEC Repair Packets

The packet format for FEC repair packets is shown in Figure 9. The transport payload consists of a Repair FEC Payload ID field followed by repair data generated in the FEC encoding process.

```
+------------------------------------+
|              IP Header             |
+------------------------------------+
|          Transport Header          |
+------------------------------------+
|        Repair FEC Payload ID       |
+------------------------------------+
|           Repair Symbols           |
+------------------------------------+
```

Figure 9: Packet Format for FEC Repair Packets

The Repair FEC Payload ID field contains information required for the operation of the FEC algorithm at the receiver. This information is defined by the FEC scheme. The format of the Repair FEC Payload ID field is defined by the FEC scheme.

5.4.1. Packet Format for FEC Repair Packets over RTP

For FEC schemes that specify the use of RTP for repair packets, the packet format for repair packets includes an RTP header as shown in Figure 10.

```
+----------------------------------+
|             IP Header            |
+----------------------------------+
|      Transport Header (UDP)      |
+----------------------------------+
|             RTP Header           |
+----------------------------------+
|       Repair FEC Payload ID      |
+----------------------------------+
|          Repair Symbols          |
+----------------------------------+
```

Figure 10: Packet Format for FEC Repair Packets over RTP

5.5. FEC Framework Configuration Information

The FEC Framework Configuration Information is information that the FEC Framework needs in order to apply FEC protection to the ADU flows. A complete CDP specification that uses the framework specified here MUST include details of how this information is derived and communicated between sender and receiver.

The FEC Framework Configuration Information includes identification of the set of source flows. For example, in the case of UDP, each source flow is uniquely identified by a tuple {source IP address, source UDP port, destination IP address, destination UDP port}. In some applications, some of these fields can contain wildcards, so that the flow is identified by a subset of the fields. In particular, in many applications the limited tuple {destination IP address, destination UDP port} is sufficient.

A single instance of the FEC Framework provides FEC protection for packets of the specified set of source flows, by means of one or more packet flows consisting of repair packets. The FEC Framework Configuration Information includes, for each instance of the FEC Framework:

1. Identification of the repair flows.

2. For each source flow protected by the repair flow(s):

   A. Definition of the source flow.

   B. An integer identifier for this flow definition (i.e., tuple). This identifier MUST be unique among all source flows that are protected by the same FEC repair flow. Integer identifiers can be allocated starting from zero and increasing by one for each flow. However, any random (but still unique) allocation is also possible. A source flow identifier need not be carried in source packets, since source packets are directly associated with a flow by virtue of their packet headers.

3. The FEC Encoding ID, identifying the FEC scheme.

4. The length of the Explicit Source FEC Payload ID (in octets).

5. Zero or more FEC-Scheme-Specific Information (FSSI) elements, each consisting of a name and a value where the valid element names and value ranges are defined by the FEC scheme.

Multiple instances of the FEC Framework, with separate and independent FEC Framework Configuration Information, can be present at a sender or receiver. A single instance of the FEC Framework protects packets of the source flows identified in (2) above; i.e., all packets sent on those flows MUST be FEC source packets as defined in Section 5.3. A single source flow can be protected by multiple instances of the FEC Framework.

The integer flow identifier identified in (2B) above is a shorthand to identify source flows between the FEC Framework and the FEC scheme. The reason for defining this as an integer, and including it in the FEC Framework Configuration Information, is so that the FEC scheme at the sender and receiver can use it to identify the source flow with which a recovered packet is associated. The integer flow identifier can therefore take the place of the complete flow description (e.g., UDP 4-tuple).

Whether and how this flow identifier is used is defined by the FEC scheme. Since repair packets can provide protection for multiple source flows, repair packets either would not carry the identifier at all or can carry multiple identifiers. However, in any case, the flow identifier associated with a particular source packet can be recovered from the repair packets as part of a FEC decoding operation.

A single FEC repair flow provides repair packets for a single instance of the FEC Framework. Other packets MUST NOT be sent within this flow; i.e., all packets in the FEC repair flow MUST be FEC repair packets as defined in Section 5.4 and MUST relate to the same FEC Framework instance.

In the case that RTP is used for repair packets, the identification of the repair packet flow can also include the RTP payload type to be used for repair packets.

FSSI includes the information that is specific to the FEC scheme used by the CDP. FSSI is used to communicate the information that cannot be adequately represented otherwise and is essential for proper FEC encoding and decoding operations. The motivation behind separating the FSSI required only by the sender (which is carried in a Sender-Side FEC-Scheme-Specific Information (SS-FSSI) container) from the rest of the FSSI is to provide the receiver or the third-party entities a means of controlling the FEC operations at the sender. Any FSSI other than the one solely required by the sender MUST be communicated via the FSSI container.

The variable-length SS-FSSI and FSSI containers transmit the information in textual representation and contain zero or more distinct elements, whose descriptions are provided by the fully specified FEC schemes.

For the CDPs that choose the Session Description Protocol (SDP) [RFC4566] for their multimedia sessions, the ABNF [RFC5234] syntax for the SS-FSSI and FSSI containers is provided in Section 4.5 of [RFC6364].

5.6. FEC Scheme Requirements

In order to be used with this framework, a FEC scheme MUST be capable of processing data either arranged into blocks of ADUs (source blocks) in case of a Block FEC Code, or arranged as a continuous flow of ADUs in case of a Convolutional FEC Code.

A specification for a new FEC scheme MUST include the following:

1. The FEC Encoding ID value that uniquely identifies the FEC scheme. This value MUST be registered with IANA, as described in Section 12.

2. The type, semantics, and encoding format of the Repair FEC Payload ID.

3. The name, type, semantics, and text value encoding rules for zero or more FEC-Scheme-Specific Information elements.

4. A full specification of the FEC code.

   This specification MUST precisely define the valid FEC-Scheme-Specific Information values, the valid FEC Payload ID values, and the valid packet payload sizes (where packet payload refers to the space within a packet dedicated to carrying encoding symbols).

Furthermore, given valid values of the FEC-Scheme-Specific Information, a valid Repair FEC Payload ID value, a valid packet payload size and in case of a Block FEC Code a source block as defined in Section 5.2, the specification MUST uniquely define the values of the encoding symbols to be included in the repair packet payload of a packet with the given Repair FEC Payload ID value.

A common and simple way to specify the FEC code to the required level of detail is to provide a precise specification of an encoding algorithm that -- given valid values of the FEC-Scheme-Specific Information, a valid Repair FEC Payload ID value, a valid packet payload size, and in case of a Block FEC Code a source block as input -- produces the exact value of the encoding symbols as output.

5. A description of practical encoding and decoding algorithms.

   This description need not be to the same level of detail as for the encoding above; however, it has to be sufficient to demonstrate that encoding and decoding of the code are both possible and practical.

FEC scheme specifications MAY additionally define the following:

Type, semantics, and encoding format of an Explicit Source FEC Payload ID.

Whenever a FEC scheme specification defines an 'encoding format' for an element, this has to be defined in terms of a sequence of bytes that can be embedded within a protocol. The length of the encoding format either MUST be fixed or it MUST be possible to derive the length from examining the encoded bytes themselves. For example, the initial bytes can include some kind of length indication.

FEC scheme specifications SHOULD use the terminology defined in this document and SHOULD follow the following format:

1. Introduction <Describe the use cases addressed by this FEC scheme>

2. Formats and Codes

   2.1. Source FEC Payload ID(s) <Either define the type and format of the Explicit Source FEC Payload ID or define how Source FEC Payload ID information is derived from source packets>

2.2. Repair FEC Payload ID <Define the type and format of the Repair FEC Payload ID>

2.3. FEC Framework Configuration Information <Define the names, types, and text value encoding formats of the FEC-Scheme-Specific Information elements>

3. Procedures <Describe any procedures that are specific to this FEC scheme, in particular derivation and interpretation of the fields in the FEC Payload IDs and FEC-Scheme-Specific Information>

4. FEC Code Specification <Provide a complete specification of the FEC Code>

Specifications can include additional sections including examples.

Each FEC scheme MUST be specified independently of all other FEC schemes, for example, in a separate specification or a completely independent section of a larger specification (except, of course, a specification of one FEC scheme can include portions of another by reference). Where an RTP payload format is defined for repair data for a specific FEC scheme, the RTP payload format and the FEC scheme can be specified within the same document.

6. Feedback

Many applications require some kind of feedback on transport performance, e.g., how much data arrived at the receiver, at what rate, and when? When FEC is added to such applications, feedback mechanisms may also need to be enhanced to report on the performance of the FEC, e.g., how much lost data was recovered by the FEC?

When used to provide instrumentation for engineering purposes, it is important to remember that FEC is generally applied to relatively small sets of data (in the sense that each block or symbols of an encoding window is transmitted over a relatively small period of time). Thus, feedback information that is averaged over longer periods of time will likely not provide sufficient information for engineering purposes. More detailed feedback over shorter time scales might be preferred. For example, for applications using RTP transport, see [RFC5725].

Applications that use feedback for congestion control purposes MUST calculate such feedback on the basis of packets received before FEC recovery is applied. If this requirement conflicts with other uses of the feedback information, then the application MUST be enhanced to support information calculated both pre- and post-FEC recovery. This is to ensure that congestion control mechanisms operate correctly based on congestion indications received from the network, rather than on post-FEC recovery information that would give an inaccurate picture of congestion conditions.

New applications that require such feedback SHOULD use RTP/RTCP [RFC3550].

7. Transport Protocols

This framework is intended to be used to define CDPs that operate over transport protocols providing an unreliable datagram service, including in particular the User Datagram Protocol (UDP) and the Datagram Congestion Control Protocol (DCCP).

8. Congestion Control

This section starts with some informative background on the motivation of the normative requirements for congestion control, which are spelled out in Section 8.2.

8.1. Motivation

- The enforcement of congestion control principles has gained a lot of momentum in the IETF over recent years. While the need for congestion control over the open Internet is unquestioned, and the goal of TCP friendliness is generally agreed upon for most (but not all) applications, the problem of congestion detection and measurement in heterogeneous networks can hardly be considered solved. Most congestion control algorithms detect and measure congestion by taking (primarily or exclusively) the packet loss rate into account. This appears to be inappropriate in environments where a large percentage of the packet losses are the result of link-layer errors and independent of the network load.

- The authors of this document are primarily interested in applications where the application reliability requirements and end-to-end reliability of the network differ, such that it warrants higher-layer protection of the packet stream, e.g., due to the presence of unreliable links in the end-to-end path and where real-time, scalability, or other constraints prohibit the use of higher-layer (transport or application) feedback. A typical example for such applications is multicast and broadcast streaming or multimedia transmission over heterogeneous networks. In other cases, application reliability requirements can be so high that the required end-to-end reliability will be difficult to achieve. Furthermore, the end-to-end network reliability is not necessarily known in advance.

- This FEC Framework is not defined as, nor is it intended to be, a quality-of-service (QoS) enhancement tool to combat losses resulting from highly congested networks. It should not be used for such purposes.

- In order to prevent such misuse, one approach is to leave standardization to bodies most concerned with the problem described above. However, the IETF defines base standards used by several bodies, including the Digital Video Broadcasting (DVB) Project, the Third Generation Partnership Project (3GPP), and 3GPP2, all of which appear to share the environment and the problem described.

- Another approach is to write a clear applicability statement. For example, one could restrict the use of this framework to networks with certain loss characteristics (e.g., wireless links). However, there can be applications where the use of FEC is justified to combat congestion-induced packet losses -- particularly in lightly loaded networks, where congestion is the result of relatively rare random peaks in instantaneous traffic load -- thereby intentionally violating congestion control principles. One possible example for such an application could be a no-matter-what, brute-force FEC protection of traffic generated as an emergency signal.

- A third approach is to require, at a minimum, that the use of this framework with any given application, in any given environment, does not cause congestion issues that the application alone would not itself cause; i.e., the use of this framework must not make things worse.

- Taking the above considerations into account, Section 8.2 specifies a small set of constraints for FEC; these constraints are mandatory for all senders compliant with this FEC Framework. Further restrictions can be imposed by certain CDPs.

8.2. Normative Requirements

- The bandwidth of FEC repair data MUST NOT exceed the bandwidth of the original source data being protected (without the possible addition of an Explicit Source FEC Payload ID). This disallows the (static or dynamic) use of excessively strong FEC to combat high packet loss rates, which can otherwise be chosen by naively implemented dynamic FEC-strength selection mechanisms. We acknowledge that there are a few exotic applications, e.g., IP traffic from space-based senders, or senders in certain hardened military devices, that could warrant a higher FEC strength.

However, in this specification, we give preference to the overall stability and network friendliness of average applications.

- Whenever the source data rate is adapted due to the operation of congestion control mechanisms, the FEC repair data rate MUST be similarly adapted.

9. Implementation Status

Editor's notes:

- RFC Editor, please remove this section motivated by RFC 7942 before publishing the RFC. Thanks.

An implementation of FECFRAME version 2 exists:

- Organisation: Inria
- Description: This is an implementation of FECFRAME version 2, using the (convolutional) RLC FEC Scheme (see "RLC FEC Scheme I-D" when available). It is based on:
  * an implementation of FECFRAME, made by Inria and commercialized by Expway (http://www.expway.com/), for which interoperability tests have been conducted;
  * a proprietary implementation of RLC Convolutional FEC Codes.
- Maturity: the FECFRAME version 1 maturity is "production", the FECFRAME version 2 extension maturity is "under progress".
- Coverage: the software implements a subset of [RFC6363], as specialized by the 3GPP eMBMS standard [MBMSTS]. This software also covers the additional features of FECFRAME version 2 for convolutional codes, relying on the RLC FEC Scheme.
- Lincensing: proprietary.
- Implementation experience: maximum.
- Information update date: October 2016.
- Contact: vincent.roca@inria.fr

10. Security Considerations

First of all, it must be clear that the application of FEC protection to a stream does not provide any kind of security. On the contrary, the FEC Framework itself could be subject to attacks or could pose new security risks. The goals of this section are to state the problem, discuss the risks, and identify solutions when feasible. It also defines a mandatory-to-implement (but not mandatory-to-use) security scheme.

10.1. Problem Statement

A content delivery system is potentially subject to many attacks. Attacks can target the content, the CDP, or the network itself, with completely different consequences, particularly in terms of the number of impacted nodes.

Attacks can have several goals:

- They can try to give access to confidential content (e.g., in the case of non-free content).

- They can try to corrupt the source flows (e.g., to prevent a receiver from using them), which is a form of denial-of-service (DoS) attack.

- They can try to compromise the receiver's behavior (e.g., by making the decoding of an object computationally expensive), which is another form of DoS attack.

- They can try to compromise the network's behavior (e.g., by causing congestion within the network), which potentially impacts a large number of nodes.

These attacks can be launched either against the source and/or repair flows (e.g., by sending fake FEC source and/or repair packets) or against the FEC parameters that are sent either in-band (e.g., in the Repair FEC Payload ID or in the Explicit Source FEC Payload ID) or out-of-band (e.g., in the FEC Framework Configuration Information).

Several dimensions to the problem need to be considered. The first one is the way the FEC Framework is used. The FEC Framework can be used end-to-end, i.e., it can be included in the final end-device where the upper application runs, or the FEC Framework can be used in middleboxes, for instance, to globally protect several source flows exchanged between two or more distant sites.

A second dimension is the threat model. When the FEC Framework operates in the end-device, this device (e.g., a personal computer) might be subject to attacks. Here, the attacker is either the end-user (who might want to access confidential content) or somebody else. In all cases, the attacker has access to the end-device but does not necessarily fully control this end-device (a secure domain can exist). Similarly, when the FEC Framework operates in a middlebox, this middlebox can be subject to attacks or the attacker can gain access to it. The threats can also concern the end-to-end transport (e.g., through the Internet). Here, examples of threats include the transmission of fake FEC source or repair packets; the replay of valid packets; the drop, delay, or misordering of packets; and, of course, traffic eavesdropping.

The third dimension consists in the desired security services. Among them, the content integrity and sender authentication services are probably the most important features. We can also mention DoS mitigation, anti-replay protection, or content confidentiality.

Finally, the fourth dimension consists in the security tools available. This is the case of the various Digital Rights Management (DRM) systems, defined outside of the context of the IETF, that can be proprietary solutions. Otherwise, the Secure Real-Time Transport Protocol (SRTP) [RFC3711] and IPsec/Encapsulating Security Payload (IPsec/ESP) [RFC4303] are two tools that can turn out to be useful in the context of the FEC Framework. Note that using SRTP requires that the application generate RTP source flows and, when applied below the FEC Framework, that both the FEC source and repair packets be regular RTP packets. Therefore, SRTP is not considered to be a universal solution applicable in all use cases.

In the following sections, we further discuss security aspects related to the use of the FEC Framework.

10.2. Attacks against the Data Flows

10.2.1. Access to Confidential Content

Access control to the source flow being transmitted is typically provided by means of encryption. This encryption can be done by the content provider itself, or within the application (for instance, by using SRTP [RFC3711]), or at the network layer on a 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 neither related to nor facilitated by the use of FEC.

Note that when encryption is applied, this encryption MUST be applied either on the source data before the FEC protection or, if done after the FEC protection, on both the FEC source packets and repair packets (and an encryption at least as cryptographically secure as the encryption applied on the FEC source packets MUST be used for the FEC repair packets). Otherwise, if encryption were to be performed only on the FEC source packets after FEC encoding, a non-authorized receiver could be able to recover the source data after decoding the FEC repair packets, provided that a sufficient number of such packets were available.

The following considerations apply when choosing where to apply encryption (and more generally where to apply security services beyond encryption). Once decryption has taken place, the source data is in plaintext. The full path between the output of the deciphering module and the final destination (e.g., the TV display in the case of a video) MUST be secured, in order to prevent any unauthorized access to the source data.

When the FEC Framework endpoint is the end-system (i.e., where the upper application runs) and if the threat model includes the possibility that an attacker has access to this end-system, then the end-system architecture is very important. More precisely, in order to prevent an attacker from getting hold of the plaintext, all processing, once deciphering has taken place, MUST occur in a protected environment. If encryption is applied after FEC protection at the sending side (i.e., below the FEC Framework), it means that FEC decoding MUST take place in the protected environment. With certain use cases, this MAY be complicated or even impossible. In such cases, applying encryption before FEC protection is preferred.

When the FEC Framework endpoint is a middlebox, the recovered source flow, after FEC decoding, SHOULD NOT be sent in plaintext to the final destination(s) if the threat model includes the possibility that an attacker eavesdrops on the traffic. In that case, it is preferable to apply encryption before FEC protection.

In some cases, encryption could be applied both before and after the FEC protection. The considerations described above still apply in such cases.

10.2.2. Content Corruption

Protection against corruptions (e.g., against forged FEC source/ repair packets) is achieved by means of a content integrity verification/source authentication scheme. This service is usually provided at the packet level. In this case, after removing all the forged packets, the source flow might sometimes be recovered.

Several techniques can provide this content integrity/source authentication service:

- o At the application layer, SRTP [RFC3711] provides several solutions to check the integrity and authenticate the source of RTP and RTCP messages, among other services. For instance, when associated with 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, with TESLA, checking a packet requires a small delay (a second or more) after its reception. Whether or not this extra delay, both in terms of startup delay at the client and end-to-end delay, is appropriate depends on the target use case. In some situations, this might degrade the user experience. In other situations, this will not be an issue. Other building blocks can be used within SRTP to provide content integrity/authentication services.

- o At the network layer, IPsec/ESP [RFC4303] 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.

Note that when integrity protection is applied, it is RECOMMENDED that it take place on both FEC source and repair packets. The motivation is to keep corrupted packets from being considered during decoding, as such packets would often lead to a decoding failure or result in a corrupted decoded source flow.

10.3. Attacks against the FEC Parameters

Attacks on these FEC parameters can prevent the decoding of the associated object. For instance, modifying the finite field size of a Reed-Solomon FEC scheme (when applicable) will lead a receiver to consider a different FEC code.

Therefore, it is RECOMMENDED that security measures be taken to guarantee the integrity of the FEC Framework Configuration Information. Since the FEC Framework does not define how the FEC Framework Configuration Information is communicated from sender to receiver, we cannot provide further recommendations on how to guarantee its integrity. However, any complete CDP specification MUST give recommendations on how to achieve it. When the FEC

Framework Configuration Information is sent out-of-band, e.g., in a session description, it 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, with a Block FEC Code, modifying the Source Block Number of a FEC source or repair packet will lead a receiver to assign this packet to a wrong block.

Therefore, it is RECOMMENDED that security measures be taken to guarantee the integrity of the Explicit Source FEC Payload ID and Repair FEC Payload ID. To that purpose, one of the packet-level source authentication/content integrity techniques described in Section 10.2.2 can be used.

10.4. When Several Source Flows Are to Be Protected Together

When several source flows, with different security requirements, need to be FEC protected jointly, within a single FEC Framework instance, then each flow MAY be processed appropriately, before the protection. For instance, source flows that require access control MAY be encrypted before they are FEC protected.

There are also situations where the only insecure domain is the one over which the FEC Framework operates. In that case, this situation MAY be addressed at the network layer, using IPsec/ESP (see Section 10.5), even if only a subset of the source flows has strict security requirements.

Since the use of the FEC Framework should not add any additional threat, it is RECOMMENDED that the FEC Framework aggregate flow be in line with the maximum security requirements of the individual source flows. For instance, if denial-of-service (DoS) protection is required, an integrity protection SHOULD be provided below the FEC Framework, using, for instance, IPsec/ESP.

Generally speaking, whenever feasible, it is RECOMMENDED that FEC protecting flows with totally different security requirements be avoided. Otherwise, significant processing overhead would be added to protect source flows that do not need it.

10.5. Baseline Secure FEC Framework Operation

The FEC Framework has been defined in such a way to be independent from the application that generates source flows. Some applications might use purely unidirectional flows, while other applications might also use unicast feedback from the receivers. For instance, this is the case when considering RTP/RTCP-based source flows.

This section describes a baseline mode of secure FEC Framework operation based on the application of the IPsec protocol, which is one possible solution to solve or mitigate the security threats introduced by the use of the FEC Framework.

Two related documents are of interest. First, Section 5.1 of [RFC5775] defines a baseline secure Asynchronous Layered Coding (ALC) operation for sender-to-group transmissions, assuming the presence of a single sender and a source-specific multicast (SSM) or SSM-like operation. The proposed solution, based on IPsec/ESP, can be used to provide a baseline FEC Framework secure operation, for the downstream source flow.

Second, Section 7.1 of [RFC5740] defines a baseline secure NACK-Oriented Reliable Multicast (NORM) operation, for sender-to-group transmissions as well as unicast feedback from receivers. Here, it is also assumed there is a single sender. The proposed solution is also based on IPsec/ESP. However, the difference with respect to [RFC5775] relies on the management of IPsec Security Associations (SAs) and corresponding Security Policy Database (SPD) entries, since NORM requires a second set of SAs and SPD entries to be defined to protect unicast feedback from receivers.

Note that the IPsec/ESP requirement profiles outlined in [RFC5775] and [RFC5740] are commonly available on many potential hosts. They can form the basis of a secure mode of operation. Configuration and operation of IPsec typically require privileged user authorization. Automated key management implementations are typically configured with the privileges necessary to allow the needed system IPsec configuration.

11. Operations and Management Considerations

The question of operating and managing the FEC Framework and the associated FEC scheme(s) is of high practical importance. The goals of this section are to discuss aspects and recommendations related to specific deployments and solutions.

In particular, this section discusses the questions of interoperability across vendors/use cases and whether defining mandatory-to-implement (but not mandatory-to-use) solutions is beneficial.

11.1. What Are the Key Aspects to Consider?

Several aspects need to be considered, since they will directly impact the way the FEC Framework and the associated FEC schemes can be operated and managed.

This section lists them as follows:

1. A Single Small Generic Component within a Larger (and Often Legacy) Solution: The FEC Framework is one component within a larger solution that includes one or several upper-layer applications (that generate one or several ADU flows) and an underlying protocol stack. A key design principle is that the FEC Framework should be able to work without making any assumption with respect to either the upper-layer application(s) or the underlying protocol stack, even if there are special cases where assumptions are made.

2. One-to-One with Feedback vs. One-to-Many with Feedback vs. One-to-Many without Feedback Scenarios: The FEC Framework can be used in use cases that completely differ from one another. Some use cases are one-way (e.g., in broadcast networks), with either a one-to-one, one-to-many, or many-to-many transmission model, and the receiver(s) cannot send any feedback to the sender(s). Other use cases follow a bidirectional one-to-one, one-to-many, or many-to-many scenario, and the receiver(s) can send feedback to the sender(s).

3. Non-FEC Framework Capable Receivers: With the one-to-many and many-to-many use cases, the receiver population might have different capabilities with respect to the FEC Framework itself and the supported FEC schemes. Some receivers might not be capable of decoding the repair packets belonging to a particular FEC scheme, while some other receivers might not support the FEC Framework at all.

4. Internet vs. Non-Internet Networks: The FEC Framework can be useful in many use cases that use a transport network that is not the public Internet (e.g., with IPTV or Mobile TV). In such networks, the operational and management considerations can be achieved through an open or proprietary solution, which is specified outside of the IETF.

5. Congestion Control Considerations: See Section 8 for a discussion on whether or not congestion control is needed, and its relationships with the FEC Framework.

6. Within End-Systems vs. within Middleboxes: The FEC Framework can be used within end-systems, very close to the upper-layer application, or within dedicated middleboxes (for instance, when it is desired to protect one or several flows while they cross a lossy channel between two or more remote sites).

7. Protecting a Single Flow vs. Several Flows Globally: The FEC Framework can be used to protect a single flow or several flows globally.

11.2. Operational and Management Recommendations

Overall, from the discussion in Section 11.1, it is clear that the CDPs and FEC schemes compatible with the FEC Framework differ widely in their capabilities, application, and deployment scenarios such that a common operation and management method or protocol that works well for all of them would be too complex to define. Thus, as a design choice, the FEC Framework does not dictate the use of any particular technology or protocol for transporting FEC data, managing the hosts, signaling the configuration information, or encoding the configuration information. This provides flexibility and is one of the main goals of the FEC Framework. However, this section gives some RECOMMENDED guidelines.

1. A Single Small Generic Component within a Larger (and Often Legacy) Solution: It is anticipated that the FEC Framework will often be used to protect one or several RTP streams. Therefore, implementations SHOULD make feedback information accessible via RTCP to enable users to take advantage of the tools using (or used by) RTCP to operate and manage the FEC Framework instance along with the associated FEC schemes.

2. One-to-One with Feedback vs. One-to-Many with Feedback vs. One-to-Many without Feedback Scenarios: With use cases that are one-way, the FEC Framework sender does not have any way to gather feedback from receivers. With use cases that are bidirectional, the FEC Framework sender can collect detailed feedback (e.g., in the case of a one-to-one scenario) or at least occasional feedback (e.g., in the case of a multicast, one-to-many scenario). All these applications have naturally different operational and management aspects. They also have different requirements or features, if any, for collecting feedback, processing it, and acting on it. The data structures for carrying the feedback also vary.

   Implementers SHOULD make feedback available using either an in-band or out-of-band asynchronous reporting mechanism. When an out-of-band solution is preferred, a standardized reporting mechanism, such as Syslog [RFC5424] or Simple Network Management Protocol (SNMP) notifications [RFC3411], is RECOMMENDED. When required, a mapping mechanism between the Syslog and SNMP reporting mechanisms could be used, as described in [RFC5675] and [RFC5676].

3. Non-FEC Framework Capable Receivers: Section 5.3 gives recommendations on how to provide backward compatibility in the presence of receivers that cannot support the FEC scheme being used or the FEC Framework itself: basically, the use of Explicit Source FEC Payload ID is banned. Additionally, a non-FEC Framework capable receiver MUST also have a means not to receive the repair packets that it will not be able to decode in the first place or a means to identify and discard them appropriately upon receiving them. This SHOULD be achieved by sending repair packets on a different transport-layer flow. In the case of RTP transport, and if both source and repair packets will be sent on the same transport-layer flow, this SHOULD be achieved by using an RTP framing for FEC repair packets with a different payload type. It is the responsibility of the sender to select the appropriate mechanism when needed.

4. Within End-Systems vs. within Middleboxes: When the FEC Framework is used within middleboxes, it is RECOMMENDED that the paths between the hosts where the sending applications run and the middlebox that performs FEC encoding be as reliable as possible, i.e., not be prone to packet loss, packet reordering, or varying delays in delivering packets.

   Similarly, when the FEC Framework is used within middleboxes, it is RECOMMENDED that the paths be as reliable as possible between the middleboxes that perform FEC decoding and the end-systems where the receiving applications operate.

5. Management of Communication Issues before Reaching the Sending FECFRAME Instance: Let us consider situations where the FEC Framework is used within middleboxes. At the sending side, the general reliability recommendation for the path between the sending applications and the middlebox is important, but it may not guarantee that a loss, reordering, or long delivery delay cannot happen, for whatever reason. If such a rare event happens, this event SHOULD NOT compromise the operation of the FECFRAME instances, at either the sending side or the receiving side. This is particularly important with FEC schemes that do not modify the ADU for backward-compatibility purposes (i.e., do not use any Explicit Source FEC Payload ID) and rely on, for instance, the RTP sequence number field to identify FEC source

packets within their source block (Block FEC Code) or source flow (Convolutional FEC Code). In this case, packet loss or packet reordering leads to a gap in the RTP sequence number space seen by the FECFRAME instance. Similarly, varying delay in delivering packets over this path can lead to significant timing issues. With FEC schemes for a Block FEC Code that indicate in the Repair FEC Payload ID, for each source block, the base RTP sequence number and number of consecutive RTP packets that belong to this source block, a missing ADU or an ADU delivered out of order could cause the FECFRAME sender to switch to a new source block. However, some FEC schemes and/or receivers may not necessarily handle such varying source block sizes. In this case, one could consider duplicating the last ADU received before the loss, or inserting zeroed ADU(s), depending on the nature of the ADU flow. Implementers SHOULD consider the consequences of such alternative approaches, based on their use cases.

6. Protecting a Single Flow vs. Several Flows Globally: In the general case, the various ADU flows that are globally protected can have different features, and in particular different real-time requirements (in the case of real-time flows). The process of globally protecting these flows SHOULD take into account the requirements of each individual flow. In particular, it would be counterproductive to add repair traffic to a real-time flow for which the FEC decoding delay at a receiver makes decoded ADUs for this flow useless because they do not satisfy the associated real-time constraints. From a practical point of view, this means that the source block creation process (Block FEC Code) or encoding window management process (Convolutional FEC Code) at the sending FEC Framework instance SHOULD consider the most stringent real-time requirements of the ADU flows being globally protected.

7. ADU Flow Bundle Definition and Flow Delivery: By design, a repair flow might enable a receiver to recover the ADU flow(s) that it protects even if none of the associated FEC source packets are received. Therefore, when defining the bundle of ADU flows that are globally protected and when defining which receiver receives which flow, the sender SHOULD make sure that the ADU flow(s) and repair flow(s) of that bundle will only be received by receivers that are authorized to receive all the ADU flows of that bundle. See Section 10.4 for additional recommendations for situations where strict access control for ADU flows is needed.

   Additionally, when multiple ADU flows are globally protected, a receiver that wants to benefit from FECFRAME loss protection SHOULD receive all the ADU flows of the bundle. Otherwise, the missing FEC source packets would be considered lost, which might significantly reduce the efficiency of the FEC scheme.

12. IANA Considerations

FEC schemes for use with this framework are identified in protocols using FEC Encoding IDs. Values of FEC Encoding IDs are subject to IANA registration. For this purpose, this document reuses the registry called the "FEC Framework (FECFRAME) FEC Encoding IDs".

The values that can be assigned within the "FEC Framework (FECFRAME) FEC Encoding IDs" registry are numeric indexes in the range (0, 255). Values of 0 and 255 are reserved. Assignment requests are granted on an IETF Review basis as defined in [RFC5226]. Section 5.6 defines explicit requirements that documents defining new FEC Encoding IDs should meet.

13. Acknowledgments

This document is based on [RFC6363] that was initiated and mostly written by Mark Watson. Great thanks are due to you, Mark.

This document is based in part on [FEC-SF], and so thanks are due to the additional authors of that document: Mike Luby, Magnus Westerlund, and Stephan Wenger. That document was in turn based on the FEC Streaming Protocol defined by 3GPP in [MBMSTS], and thus, thanks are also due to the participants in 3GPP SA Working Group 4. Further thanks are due to the members of the FECFRAME Working Group for their comments and reviews.

14. References

14.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3411] Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, DOI 10.17487/RFC3411, December 2002, <http://www.rfc-editor.org/info/rfc3411>.

[RFC5052] Watson, M., Luby, M., and L. Vicisano, "Forward Error Correction (FEC) Building Block", RFC 5052, DOI 10.17487/RFC5052, August 2007, <http://www.rfc-editor.org/info/rfc5052>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC5424] Gerhards, R., "The Syslog Protocol", RFC 5424, DOI 10.17487/RFC5424, March 2009, <http://www.rfc-editor.org/info/rfc5424>.

[RFC6363] Watson, M., Begen, A., and V. Roca, "Forward Error Correction (FEC) Framework", RFC 6363, DOI 10.17487/RFC6363, October 2011, <http://www.rfc-editor.org/info/rfc6363>.

14.2. Informative References

[FEC-SF] Watson, M., Luby, M., Westerlund, M., and S. Wenger, "Forward Error Correction (FEC) Streaming Framework", Work in Progress, July 2005.

[FECFRAMEv2-Motivations] IRTF NetWork Coding Research Group (NWCRG), "FECFRAMEv2: Adding Sliding Encoding Window Capabilities to the FEC Framework: Problem Position", May 2016, <https://tools.ietf.org/html/draft-roca-nwcrg-fecframev2-problem-position-02>.

[MBMSTS] 3GPP, "Multimedia Broadcast/Multicast Service (MBMS); Protocols and codecs", 3GPP TS 26.346, March 2009, <http://ftp.3gpp.org/specs/html-info/26346.htm>.

[RFC3095] Bormann, C., Burmeister, C., Degermark, M., Fukushima, H., Hannu, H., Jonsson, L-E., Hakenberg, R., Koren, T., Le, K., Liu, Z., Martensson, A., Miyazaki, A., Svanbro, K., Wiebke, T., Yoshimura, T., and H. Zheng, "RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed", RFC 3095, DOI 10.17487/RFC3095, July 2001, <http://www.rfc-editor.org/info/rfc3095>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <http://www.rfc-editor.org/info/rfc3550>.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <http://www.rfc-editor.org/info/rfc3711>.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <http://www.rfc-editor.org/info/rfc4303>.

[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, DOI 10.17487/RFC4383, February 2006, <http://www.rfc-editor.org/info/rfc4383>.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <http://www.rfc-editor.org/info/rfc4566>.

[RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, DOI 10.17487/RFC4588, July 2006, <http://www.rfc-editor.org/info/rfc4588>.

[RFC5675] Marinov, V. and J. Schoenwaelder, "Mapping Simple Network Management Protocol (SNMP) Notifications to SYSLOG Messages", RFC 5675, DOI 10.17487/RFC5675, October 2009, <http://www.rfc-editor.org/info/rfc5675>.

[RFC5676] Schoenwaelder, J., Clemm, A., and A. Karmakar, "Definitions of Managed Objects for Mapping SYSLOG Messages to Simple Network Management Protocol (SNMP) Notifications", RFC 5676, DOI 10.17487/RFC5676, October 2009, <http://www.rfc-editor.org/info/rfc5676>.

[RFC5725] Begen, A., Hsu, D., and M. Lague, "Post-Repair Loss RLE Report Block Type for RTP Control Protocol (RTCP) Extended Reports (XRs)", RFC 5725, DOI 10.17487/RFC5725, February 2010, <http://www.rfc-editor.org/info/rfc5725>.

[RFC5740] Adamson, B., Bormann, C., Handley, M., and J. Macker, "NACK-Oriented Reliable Multicast (NORM) Transport Protocol", RFC 5740, DOI 10.17487/RFC5740, November 2009, <http://www.rfc-editor.org/info/rfc5740>.

[RFC5775] Luby, M., Watson, M., and L. Vicisano, "Asynchronous Layered Coding (ALC) Protocol Instantiation", RFC 5775, DOI 10.17487/RFC5775, April 2010, <http://www.rfc-editor.org/info/rfc5775>.

[RFC6364] Begen, A., "Session Description Protocol Elements for FEC Framework", RFC 6364, October 2011.

[RFC6681] Watson, M., Stockhammer, T., and M. Luby, "Raptor Forward Error Correction (FEC) Schemes for FECFRAME", RFC 6681, DOI 10.17487/RFC6681, August 2012, <http://www.rfc-editor.org/info/rfc6681>.

[RFC6816] Roca, V., Cunche, M., and J. Lacan, "Simple Low-Density Parity Check (LDPC) Staircase Forward Error Correction (FEC) Scheme for FECFRAME", RFC 6816, DOI 10.17487/RFC6816, December 2012, <http://www.rfc-editor.org/info/rfc6816>.

[RFC6865] Roca, V., Cunche, M., Lacan, J., Bouabdallah, A., and K. Matsuzono, "Simple Reed-Solomon Forward Error Correction (FEC) Scheme for FECFRAME", RFC 6865, DOI 10.17487/RFC6865, February 2013, <http://www.rfc-editor.org/info/rfc6865>.

Appendix A. Possible management within a FEC Scheme of the Encoding Window with Convolutional FEC Codes (non Normative)

The FEC Framework does not specify the management of the encoding window, which is left to the FEC scheme associated to a Convolutional FEC Code. This section is therefore non normative. On the opposite, the FEC scheme associated to a Convolutional FEC Code:

- o MUST define an encoding window that slides over the set of ADUs and its management;
- o MUST define the relationships between ADUs and the associated source symbols (as with Block FEC Codes).

Source symbols are added to the sliding encoding window each time a new ADU arrives, where the following information is provided for this ADU by the FEC Framework:

- o A description of the source flow with which the ADU is associated;
- o The ADU itself;
- o The length of the ADU.

Source symbols and the corresponding ADUs are removed from the sliding encoding window, for instance:

- o after a certain delay, for situations where the sliding encoding window is managed on a time basis. The motivation is that an old ADU of a real-time flow becomes useless after a certain delay. The ADU retention delay in the sliding encoding window is therefore initialized according to the real-time features of incoming flow(s). Note that because of possible inter-dependancies between source symbols (and potentially between ADUs), this strategy may prove to be sub-optimum if applied in a very strict way;
- o once the sliding encoding window has reached its maximum size (there is usually an upper limit to the sliding encoding window size), the oldest symbol is removed each time a new symbol is added;
- o when the sliding encoding window is of fixed size or has reached its maximum size, the oldest symbol is removed each time a new symbol is added.

Limitations MAY exist that impact the encoding window management. For instance:

- at the FEC Framework level: the source flows can have real-time constraints that limit the number of ADUs in the encoding window;

- at the FEC scheme level: there may be theoretical or practical limitations (e.g., because of computational complexity aspect or field size limits in the signaling headers) that limit the number of ADUs in the encoding window.

The most stringent limitation defines the maximum encoding window size, either in terms of number of source symbols or number of ADUs, whichever applies.

Appendix B. Changes with Respect to RFC 6363

This annex summarizes the changes made with respect to [RFC6363]:

- Section 1 includes a discussion on the limits of block FEC codes in the context of real-time flows (the main target of FECFRAME) as well as motivates for the addition of convolutional FEC codes.

- definitions specific to Convolutional FEC Codes are added to Section 2 and clarifications are made when definitions are restricted to Block FEC Codes.

- Section 4.1 is updated to distinguish the procedures related to Block FEC Codes from those related to Convolutional FEC Codes.

- Section 4.4 is added to specify the sender operations when Convolutional FEC Codes are used. This section is structured in a similar way as that of Block FEC Codes. It also clarify operations that have to be specified in the associated FEC Scheme rather than the FECFRAME version 2 document.

- Section 4.5 is added to specify the receiver operations when Convolutional FEC Codes are used. This section is structured in a similar way as that of Block FEC Codes. It also clarify operations that have to be specified in the associated FEC Scheme rather than the FECFRAME version 2 document.

- Section 5.1 clarifies and motivates operations (i.e., the management of the encoding window and decoding window, and the ADU to source mapping) that are the responsibility of the FEC Scheme.

- minor modifications in Section 5.6 to distinguish the case of Block versus Convolutional specific information.

- Section 9 is added to give insights on the implementation status of FECFRAME version 2. It will be removed from the final document as well as this item.

- minor modifications in Section 10.3 to distinguish the case of Block versus Convolutional specific information.

- minor modifications in Section 11.2 to distinguish the case of Block versus Convolutional specific information.

- clarification in Section 12 that the registry created for the needs of FECFRAME version 1 has to be reused for FECFRAME version 2.

- added non normative appendix Appendix A for clarification and illustration purposes.

- authors list update (at his demand Mark Watson has been removed).

Authors' Addresses

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

EMail: vincent.roca@inria.fr

Ali Begen
Networked Media
Konya
Turkey

EMail: ali.begen@networked.media