FEC Framework
Internet-Draft
Intended status: Standards Track
Expires: September 5, 2011

M. Watson
Netflix, Inc.
A. Begen
Cisco
V. Roca
INRIA

March 4, 2011

Forward Error Correction (FEC) Framework
draft-ietf-fecframe-framework-14

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) which is compliant with various requirements defined in this document. Thus, Content Delivery Protocols can be defined which are not specific to a particular FEC scheme, and FEC schemes can be defined which are not specific to a particular Content Delivery Protocol.

Status of this Memo

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

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

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

This Internet-Draft will expire on September 5, 2011.

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 which do not provide guaranteed packet delivery. Primary examples are real-time, or streaming, media applications such as broadcast, multicast or on-demand audio, video or multimedia.

Forward Error Correction (FEC) is a well-known technique for improving reliability of packet transmission over networks which do not provide guaranteed packet delivery, especially in multicast and broadcast applications. The FEC Building Block defined in [RFC5052] provides a framework for definition of Content Delivery Protocols (CDPs) for object delivery (including, primarily, file delivery) which 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 which is also defined according to the requirements of the FEC Building Block.

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 which provide for FEC protection for arbitrary packet flows over unreliable transports such as UDP. As such, 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, but rather 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 how the details of the protected flows and the FEC streams which protect them are communicated from sender to receiver. It is expected that any complete CDP specification which makes use of this framework will address these signaling requirements. However, this document does specify the information which 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 the RTP protocol 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 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).

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., RTSP.

Content Delivery Protocol (CDP): A complete application protocol specification which, 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.

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

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

FEC Payload ID: Information which 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 which 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: An 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: An 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 transport of the source and repair flows, e.g., UDP and DCCP.

The following definitions are aligned with [RFC5052]:

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 which are to be FEC protected as a single block.

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 an FEC scheme, in a similar sense to that defined in [RFC5052] and uses the terminology of that document. The FEC scheme defines the FEC encoding and decoding, and 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 an FEC scheme, which is described in this document, is a logical one, which 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 which MUST be available to both sender and receiver(s). For example, this information includes the specification of the ADU flows which are to be FEC protected, specification of the ADU flow(s) which 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 each repair flow(s)). The FEC Framework Configuration Information also includes information fields which 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 which 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 which 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 an FEC scheme has to define an RTP payload format for the repair data.

The architecture outlined above is illustrated in the 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 which protect the source RTP headers (as well as repair data for the source RTP payloads).

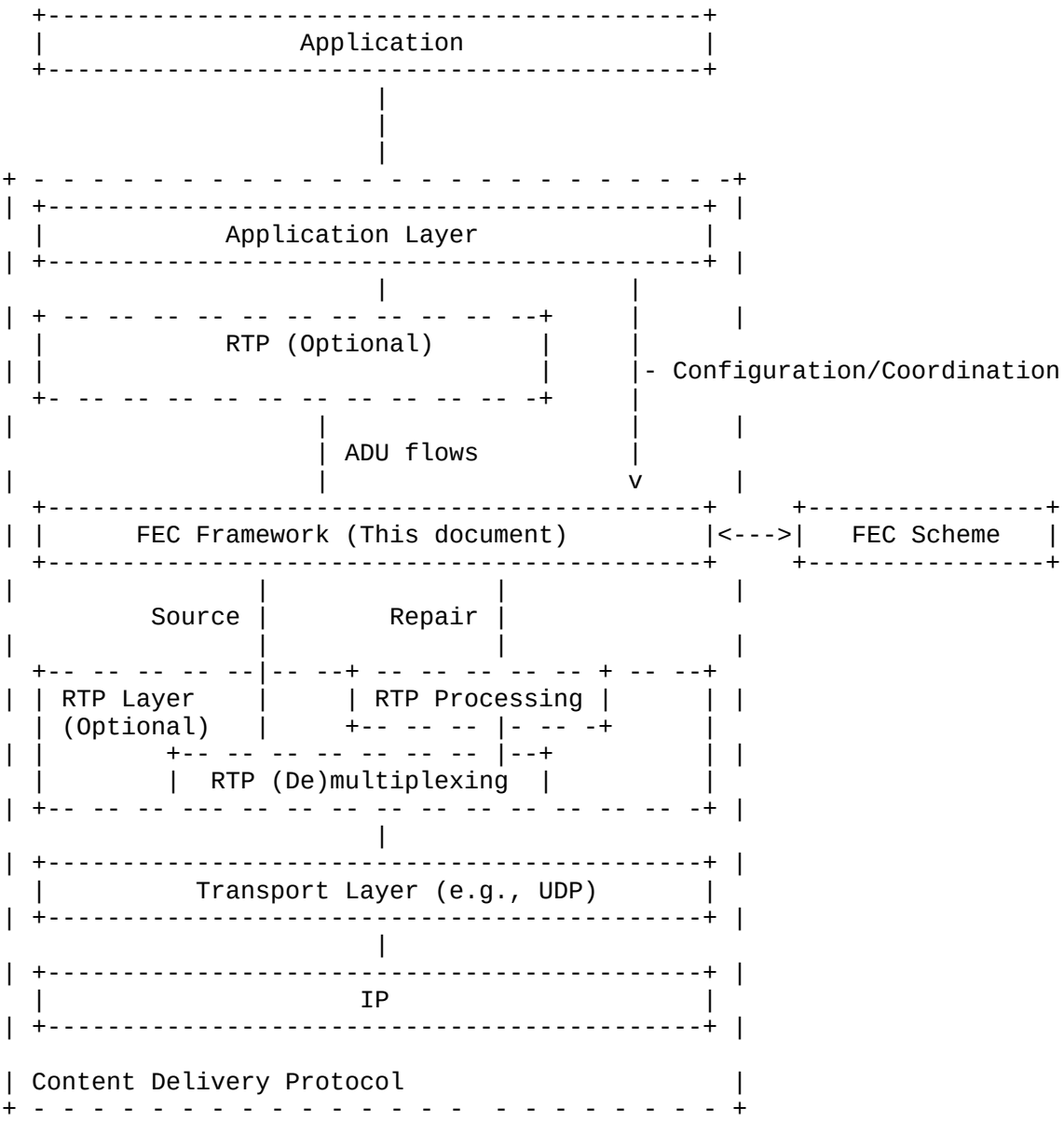

Figure 1: FEC Framework architecture

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 which can be protected together, except that the ADU is carried over a supported transport protocol (See Section 7). The data can be from multiple source flows that are protected jointly. 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 which 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, the FEC Framework passes the payloads for a given block to the FEC scheme for FEC encoding. The FEC scheme performs the FEC encoding operation and returns the following information:

- Optionally, FEC Payload IDs for each of the source payloads (encoded according to an FEC-Scheme-Specific format).
- One or more FEC repair packet payloads.
- FEC Payload IDs for each of the repair packet payloads (encoded according to an 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, and 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 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, but 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 related to a given source block 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.

The FEC source packets MUST contain information which identifies the source block and the position within the source block (in terms specific to the FEC scheme) occupied by the ADU. 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.

The FEC repair packets MUST contain information which identifies the source block and the relationship between the contained repair payloads and the original source block. 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

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 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).

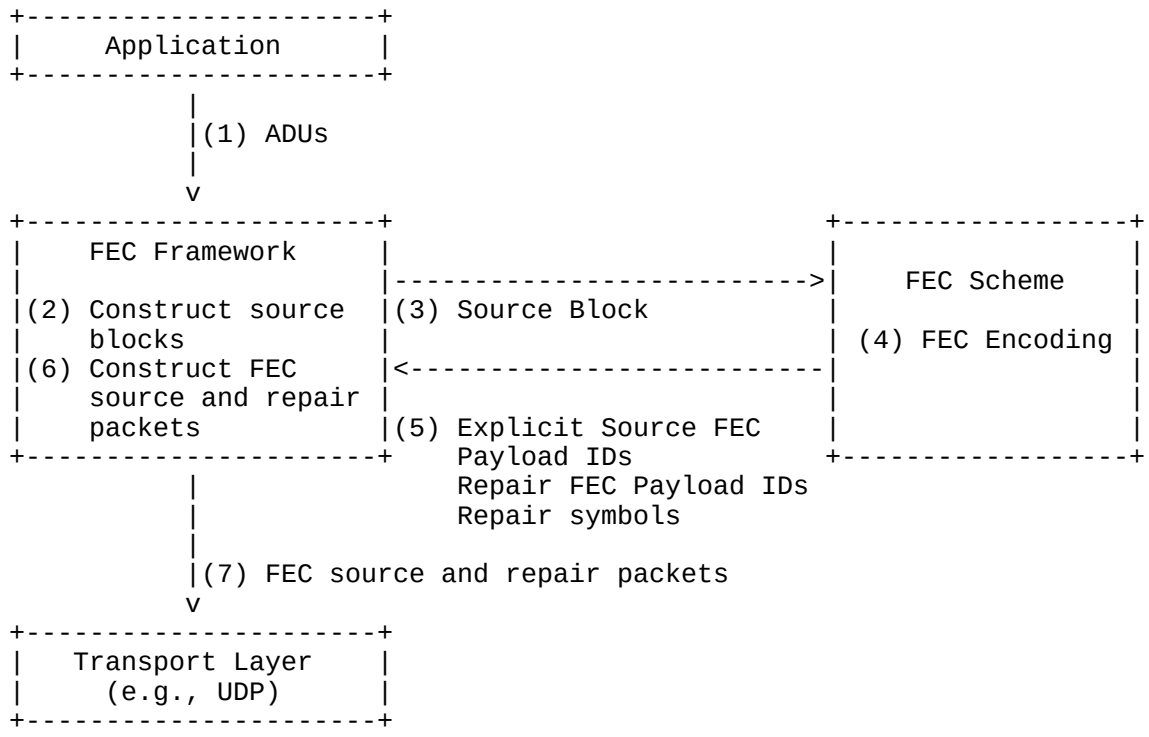

Figure 2: Sender operation

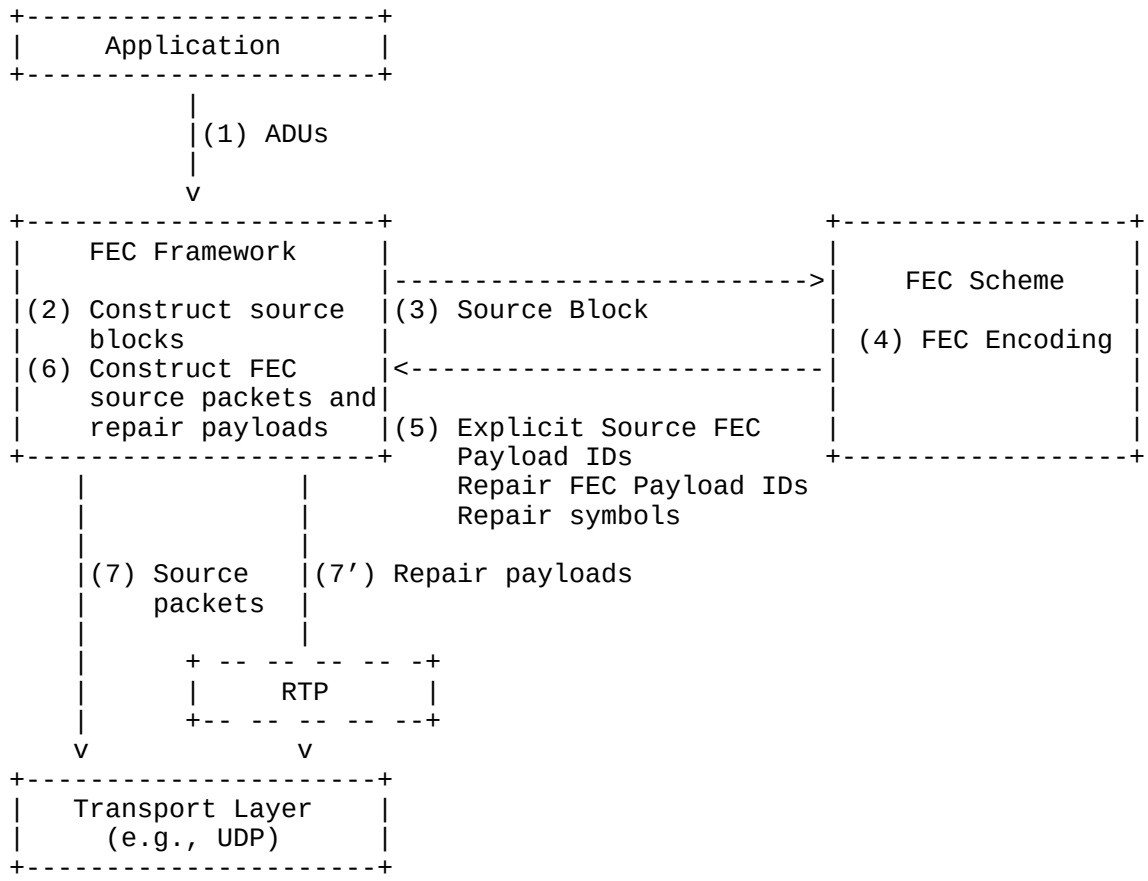

Figure 3: Sender operation with RTP repair flows

4.3. Receiver Operation

The following describes a possible receiver algorithm, illustrated in Figure 4 and Figure 5 for the case of RTP repair flows, when receiving an 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) is 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, 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 which 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 which are correctly received and those which 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.

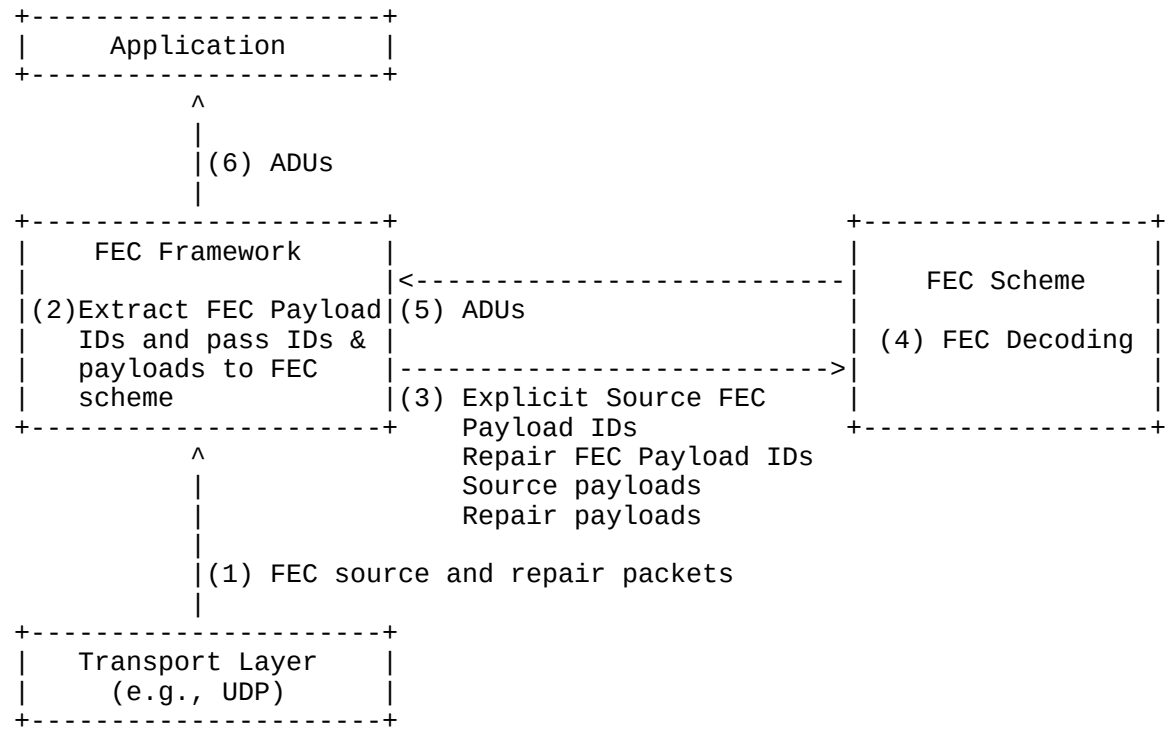

Figure 4: Receiver operation

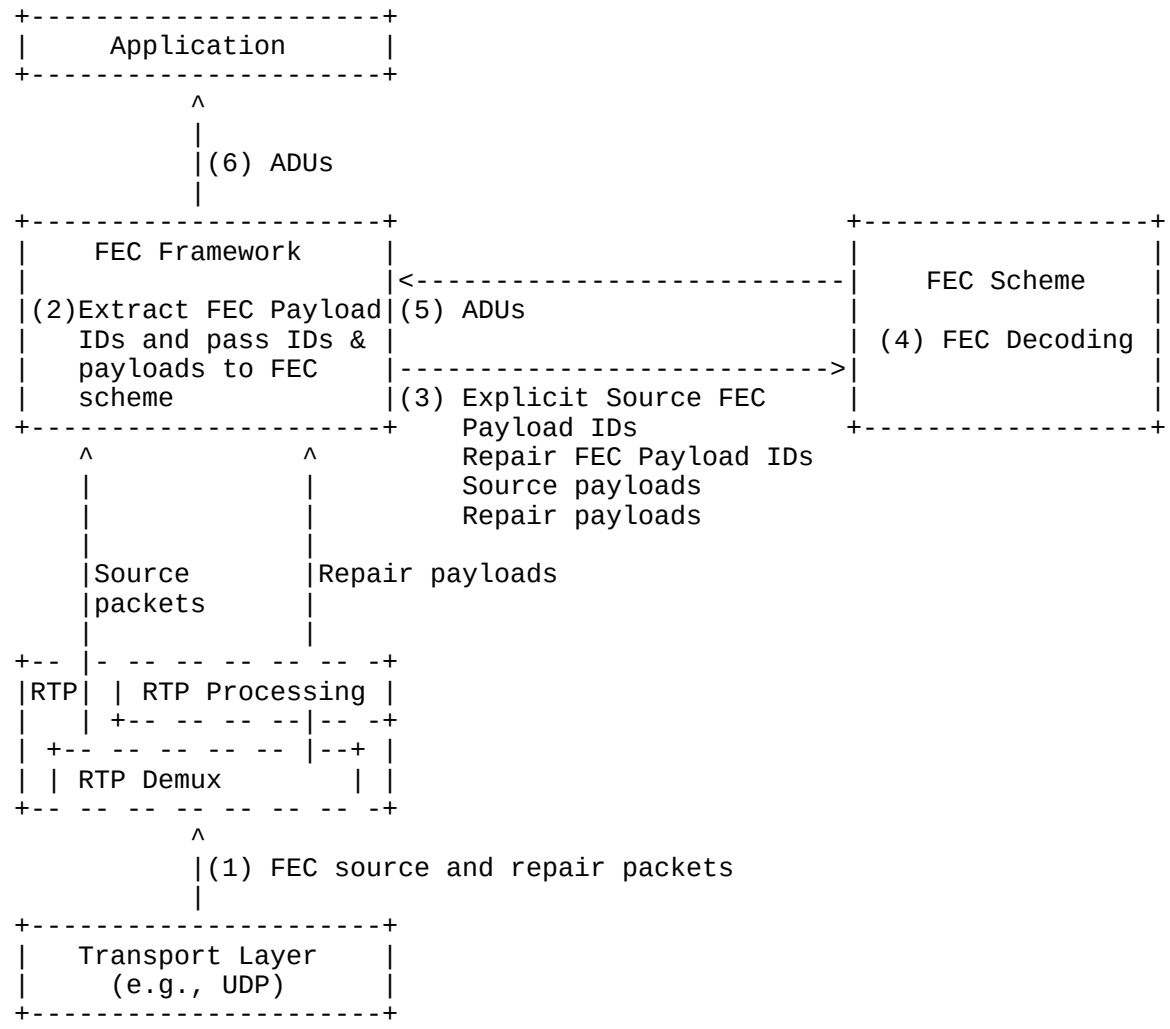

Figure 5: Receiver operation with RTP repair flows

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

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. 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.

5.2. Structure of the Source Block

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 with.

- 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 6, 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.

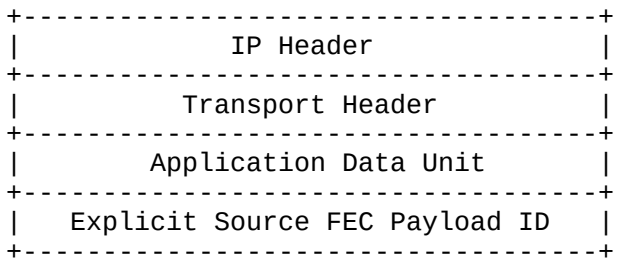

Figure 6: Structure of the FEC packet format for FEC source packets

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 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 backwards 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, except that the use of this sequence number places a constraint on source block construction. Source packets within a given source block MUST have consecutive sequence numbers (where consecutive includes wrap-around from the maximum value which 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 7. The transport payload consists of a Repair FEC Payload ID field followed by repair data generated in the FEC encoding process.

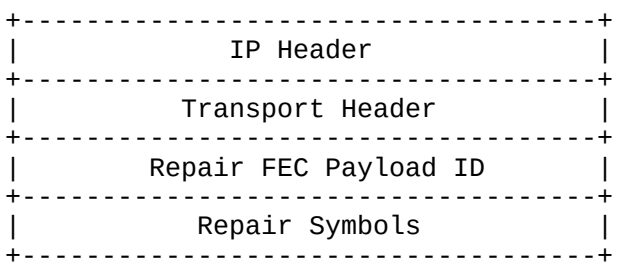

Figure 7: Packet format for repair packets

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 which specify the use of RTP for repair packets, the packet format for repair packets includes an RTP header as shown in Figure 8.

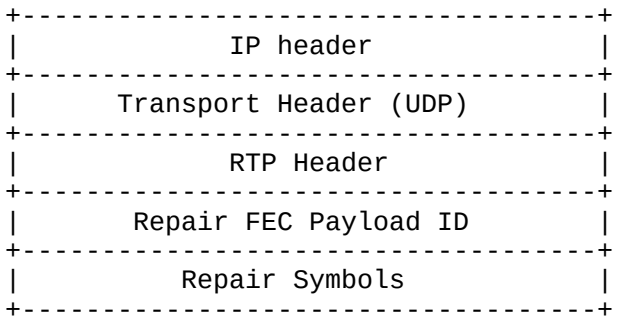

Figure 8: Packet format for repair packets

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 amongst 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 would either 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 an 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 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] as their session description protocol, the ABNF [RFC5234] syntax for the SS-FSSI and FSSI containers is provided in Section 4.5 of [I-D.ietf-fecframe-sdp-elements].

5.6. FEC Scheme Requirements

In order to be used with this framework, an FEC scheme MUST be capable of processing data arranged into blocks of ADUs (source blocks).

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 11.

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 a source block as defined in Section 5.2, valid values of the FEC-Scheme-Specific Information, a valid Repair FEC Payload ID value and a valid packet payload size, 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 which, given a source block, valid values of the FEC-Scheme-Specific Information, a valid Repair FEC Payload ID value and a valid packet payload size 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 is both possible and practical.

FEC scheme specifications MAY additionally define the following:

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

Whenever an FEC scheme specification defines an ’encoding format’ for an element, this has to be defined in terms of a sequence of bytes which can be embedded within a protocol. The length of the encoding format MUST either 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 which 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 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 can 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 blocks of data (in the sense that each block 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 which used 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 both information calculated 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 which would give an inaccurate picture of congestion conditions.

New applications which 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 the recent years. While the need for congestion control over the open Internet is unquestioned, and the goal of TCP friendliness is generally agreed for most (but not all) applications, the subject of congestion detection and measurement in heterogeneous networks can hardly be considered as 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, nor intended, as a QoS enhancement tool to combat losses resulting from highly congested networks. It should not be used for such purposes.

- In order to prevent such mis-use, 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 DVB, 3GPP, 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 a traffic generated as an emergency signal.

- o 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 which the application alone would not itself cause, i.e., the use of this framework must not make things worse.

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

8.2. Normative Requirements

- o 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, which could warrant a higher FEC strength. However, in this specification we give preference to the overall stability and network friendliness of average applications.

- o 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. 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 opposite, 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.

9.1. Problem Statement

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

Attacks can have several goals:

- They can try to give access to a confidential content (e.g., in case of a 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 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 not necessarily to the full control of the 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 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 out of the context of the IETF and that can be proprietary solutions. Otherwise SRTP and IPsec/ESP 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 generates RTP source flows and, when applied below the FEC Framework, that both the FEC source and repair packets to be regular RTP packets. Therefore SRTP is not considered as 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.

9.2. Attacks Against the Data Flows

9.2.1. 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 the Secure Real-time Transport Protocol (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 is 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 either be applied on the source data before the FEC protection, or if done after the FEC protection, then both the FEC source packets and repair packets MUST be encrypted (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 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 to get hold of the plaintext, all processings, 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 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 that case 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 the traffic. In that case also it is preferred 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.

9.2.2. 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:

- 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, associated to the Timed Efficient Stream Loss-Tolerant Authentication (TESLA) [RFC4383], SRTP is an attractive solution that is robust to losses, provides a true authentication/integrity service, and does not create any prohibitive processing load or transmission overhead. Yet, checking a packet requires a small delay (a second or more) after its reception with TESLA. Whether this extra delay, both in terms of startup delay at the client and end-to-end delay, is appropriate or not 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.

- 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 is used.

Note that when integrity protection is applied, it is RECOMMENDED that it takes place on both FEC source and repair packets. The motivation is to avoid corrupted packets to be considered during decoding, which would often lead to a decoding failure or result in a corrupted decoded source flow.

9.3. 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.

It is therefore RECOMMENDED that security measures are taken to guarantee the FEC Framework Configuration Information integrity. 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, modifying the Source Block Number of an FEC source or repair packet will lead a receiver to assign this packet to a wrong block.

It is therefore RECOMMENDED that security measures are taken to guarantee the Explicit Source FEC Payload ID and Repair FEC Payload ID integrity. To that purpose, one of the packet-level source authentication/content integrity techniques of Section 9.2.2 can be used.

9.4. 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 9.5), even if only a subset of the source flows have strict security requirements.

Since the use of 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 to avoid FEC protecting flows with totally different security requirements. Otherwise, an important processing would be added to protect the source flows that do not need it.

9.5. Baseline Secure FEC Framework Operation

This section describes a baseline mode of secure FEC Framework operation based on the application of the IPsec security 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 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 NORM operation, for sender-to-group transmissions as well as unicast feedbacks 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 the first document relies on the management of IPsec Security Associations (SA) and corresponding Security Policy Database (SPD) entries, since NORM requires a second set of SA and SPD to be defined to protect unicast feedbacks from receivers.

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 feedbacks from the receivers. For instance, this is the case when considering RTP/RTCP based source flows. Depending on the specific situation, it is RECOMMENDED that the baseline secure FEC Framework operation inherits from [RFC5775] in case of purely unidirectional sender-to-group flows, or [RFC5740] in case both sender-to-group and unicast feedbacks flows are needed.

Note that the IPsec/ESP requirements profiles outlined in [RFC5775] and [RFC5740] are commonly available on many potential hosts. They can form the basis of a secure mode of operation. One potential limitation, however, is the need for privileged user authorization. However, automated key management implementations are typically configured with the privileges necessary to affect system IPsec configuration.

10. 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 the general requirements, aspects related to a specific deployment and solutions whenever possible.

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.

10.1. 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:

- A Single Small Generic Component within a Larger (and Often Legacy) Solution: The FEC Framework is one component within a larger solution which includes both 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.

- 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).

- 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 be supporting the FEC Framework at all.

- 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.

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

- 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.

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

10.2. Operational and Management Recommendations

Overall, from the discussion of Section 10.1, it is clear that the CDPs and FEC schemes compatible with the FEC Framework widely differ 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 recommendations.

- 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, there are use cases that may take advantage of RTCP to collect feedback information, and one can take advantage of the tools using (or used by) RTCP to operate and manage the FEC Framework instance along with the associated FEC schemes.

- 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 case of a

one-to-one scenario) or at least occasional feedback (e.g., in case of a multicast, one-to-many scenario). All these applications have naturally different operational and management aspects. If any, they also have different requirements or features for collecting feedback, processing it and acting on it. The data structures for carrying the feedback also vary.

- Non-FEC Framework Capable Receivers: Section 5.3 gives recommendations on how to provide backward compatibility in 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 can be achieved by sending repair packets on a different transport-layer flow. In case of an RTP source flow, this can also be achieved by using an RTP framing for FEC repair packets with a different payload type. Both source and repair packets can then be sent on the same transport-layer flow. It is the responsibility of the sender to select the appropriate mechanism when needed.

- 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., are not prone to packet loss, packet reordering, or varying delays in delivering packets.

  Similarly, it is RECOMMENDED that the paths between the middleboxes that perform FEC decoding and the end-systems where the receiving applications operate, in situations where this is a different host, be as reliable as possible.

  The recommendation for the sending side 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 for instance on the RTP sequence number field to identify FEC source packets within their source block. 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 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, another solution SHOULD be considered that is use-case specific and whose consequences need to be carefully considered (e.g., by duplicating the last ADU received before the loss, or by inserting a zero'ed ADU, depending on the ADU flow nature).

- o 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 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 counter-productive 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 at the sending FEC Framework instance, SHOULD consider the most stringent real-time requirements of the ADU flows being globally protected.

- o 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 repair flow(s) SHOULD only be received by receivers that are authorized to receive all the associated ADU flows. See Section 9.4 for additional recommendations for situations where a strict access control to ADU flows is needed.

  Additionally when multiple ADU flows are globally protected, a receiver who 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 as lost which might significantly reduce the efficiency of the FEC scheme.

11. 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 purposes, this document creates a new registry called FEC Framework (FECFRAME) FEC Encoding IDs.

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

12. Acknowledgments

This document is based in part on [I-D.watson-tsvwg-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.

13. References

13.1. Normative references

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

[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, July 2001.

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

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

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

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

13.2. Informative references

[I-D.watson-tsvwg-fec-sf]
Watson, M., "Forward Error Correction (FEC) Streaming Framework", draft-watson-tsvwg-fec-sf-00 (work in progress), July 2005.

[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, February 2010.

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

[RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.

[I-D.ietf-fecframe-sdp-elements]
Begen, A., "Session Description Protocol Elements for FEC

Framework", draft-ietf-fecframe-sdp-elements-11 (work in progress), October 2010.

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

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

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

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

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

[MBMSTS] 3GPP, "Multimedia Broadcast/Multicast Service (MBMS); Protocols and codecs", 3GPP TS 26.346, April 2005.

Authors' Addresses

Mark Watson
Netflix, Inc.
100 Winchester Circle
Los Gatos, CA 95032
USA

Email: watsonm@netflix.com

Ali Begen
Cisco
181 Bay Street
Toronto, ON M5J 2T3
Canada

Email: abegen@cisco.com

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

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

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

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

October 22, 2010

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

Abstract

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

Status of this Memo

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

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

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

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

Copyright Notice

Table of Contents

1. Introduction

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

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

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

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

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

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

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

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. Definitions Notations and Abbreviations

3.1. Definitions

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

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

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

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

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

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

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

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

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

[FECFRAME-FRAMEWORK]:

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

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

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

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

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

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

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

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

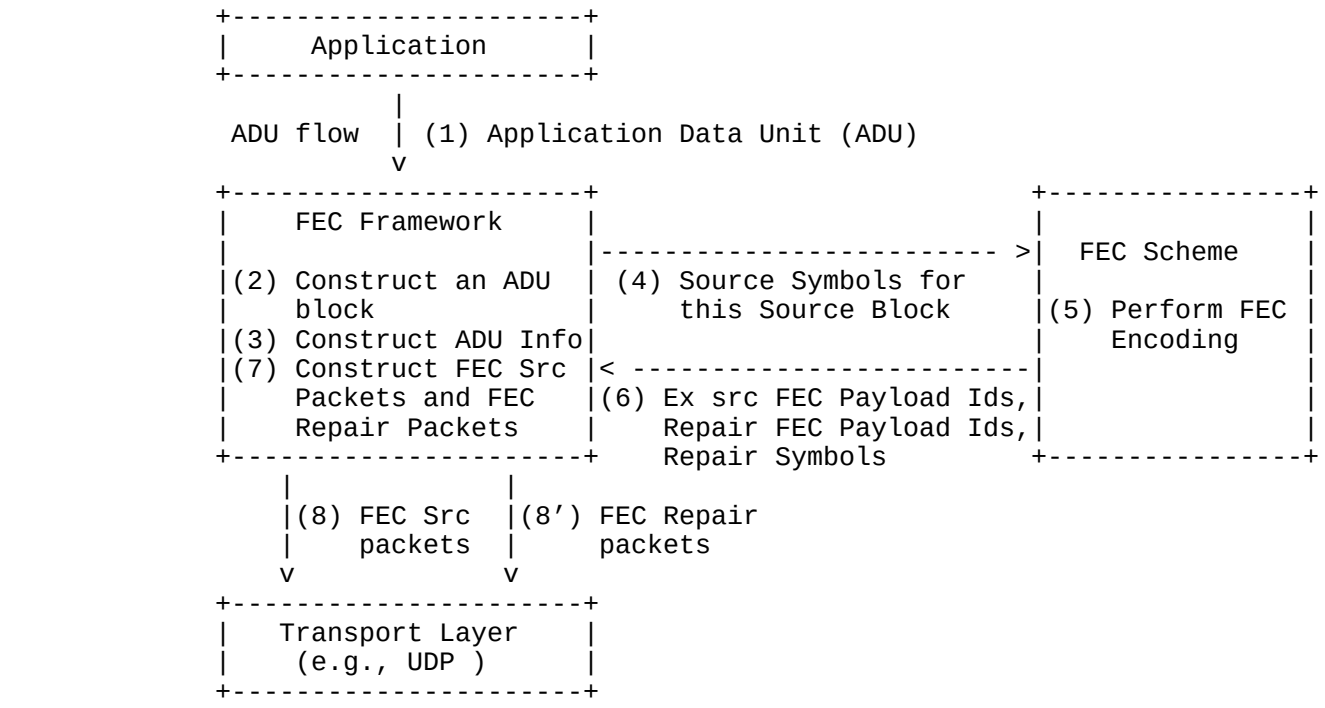

Figure 1: Terminology used in this document (sender).

3.2. Notations

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

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

Some of them are FECFRAME framework specific:

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

3.3. Abbreviations

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

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

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

4.1. Restrictions

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

4.2. ADU Block Creation

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

We now detail each of these aspects.

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

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

Some common max1_k values are:

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

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

Then, max_k is given by:

max_k = min(max1_k, max2_k)

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

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

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

4.3. Source Block Creation

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

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

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

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

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

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

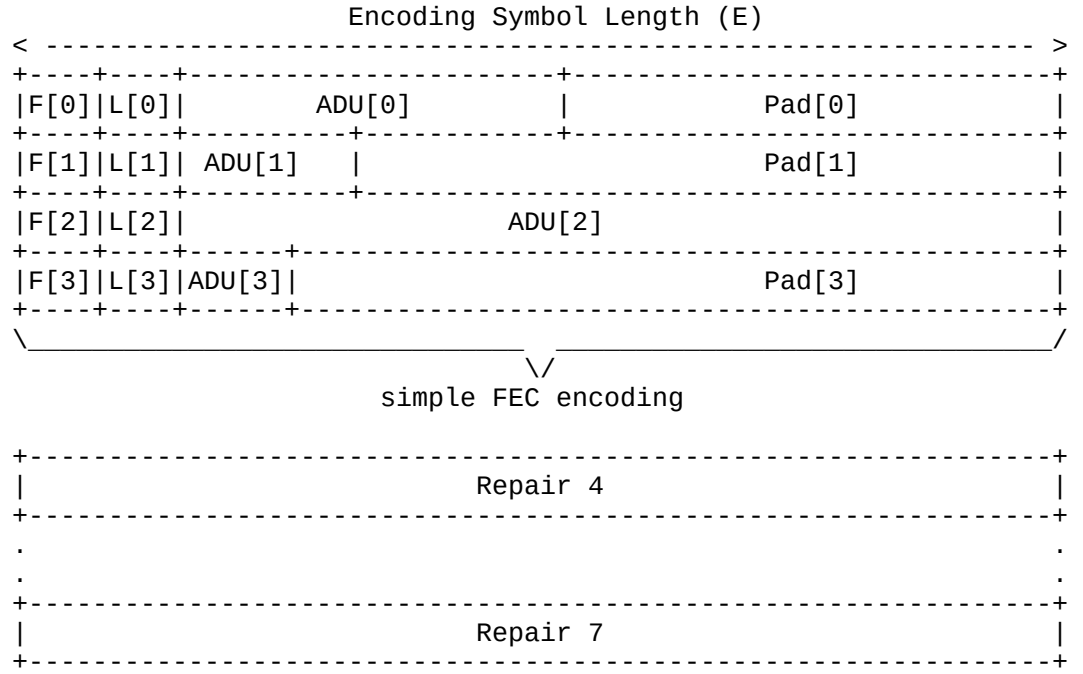

Figure 2: Source block creation, for code rate 1/2 (equal number of source and repair symbols, 4 in this example), and S = 0.

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

5. LDPC-Staircase FEC Scheme for Arbitrary ADU Flows

5.1. Formats and Codes

5.1.1. FEC Framework Configuration Information

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

5.1.1.1. Mandatory Information

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

5.1.1.2. FEC Scheme-Specific Information

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

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

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

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

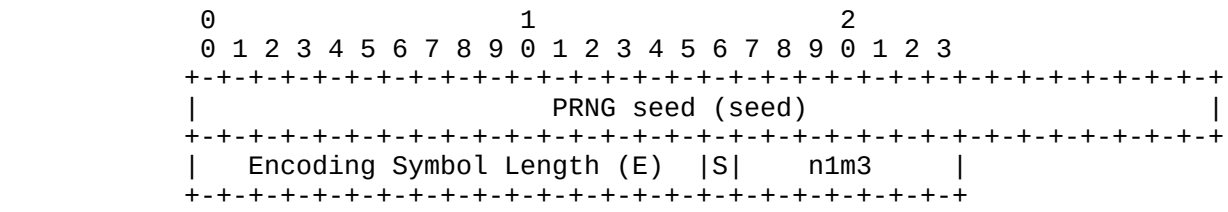

Figure 3: FSSI encoding format.

5.1.2. Explicit Source FEC Payload ID

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

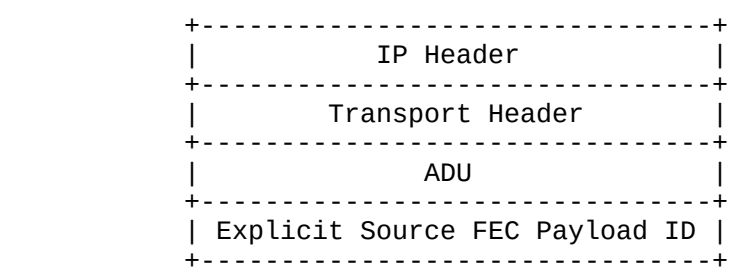

Figure 4: Structure of a FEC Source Packet with the Explicit Source FEC Payload ID.

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

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

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

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

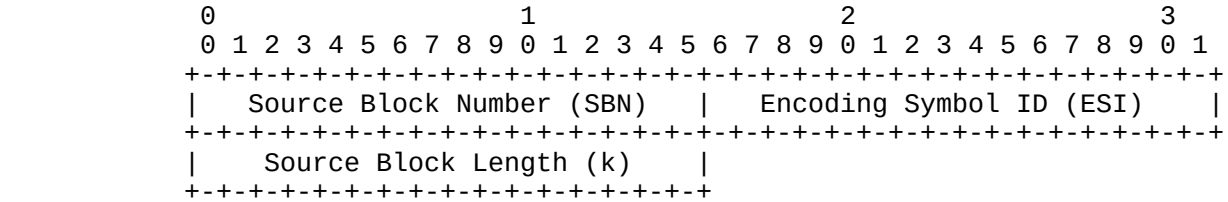

Figure 5: Source FEC Payload ID encoding format.

5.1.3. Repair FEC Payload ID

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

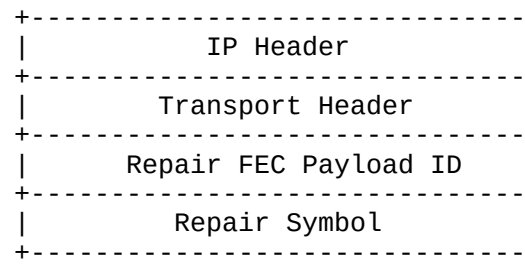

Figure 6: Structure of a FEC Repair Packet with the Repair FEC Payload ID.

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

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

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

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

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

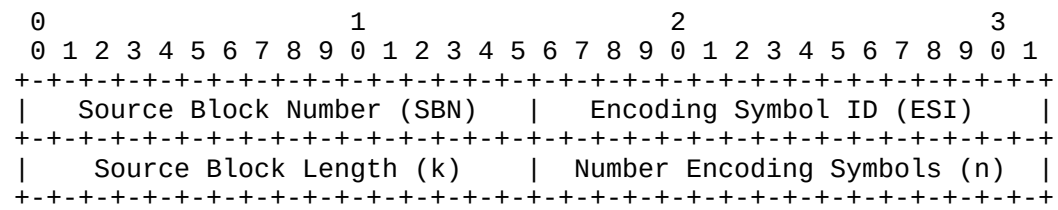

Figure 7: Repair FEC Payload ID encoding format.

5.2. Procedures

The following procedures apply:

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

5.3. FEC Code Specification

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

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

6. Security Considerations

6.1. Problem Statement

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

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

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

6.2. Attacks Against the Data Flow

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

6.2.1. Access to Confidential Contents

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

6.2.2. Content Corruption

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

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

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

6.3. Attacks Against the FEC Parameters

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

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

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

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

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

7. IANA Considerations

The FEC Encoding ID value is subject to IANA registration.

TBD

8. Acknowledgments

TBD

9. References

9.1. Normative References

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

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

[FECFRAME-FRAMEWORK]

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

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

9.2. Informative References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Authors' Addresses

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

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

Mathieu Cunche
NICTA
Australia

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

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

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

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

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

October 22, 2010

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

Abstract

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

Status of this Memo

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

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

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

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

Copyright Notice

Table of Contents

1. Introduction

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

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

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

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

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

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

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

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

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. Definitions Notations and Abbreviations

3.1. Definitions

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

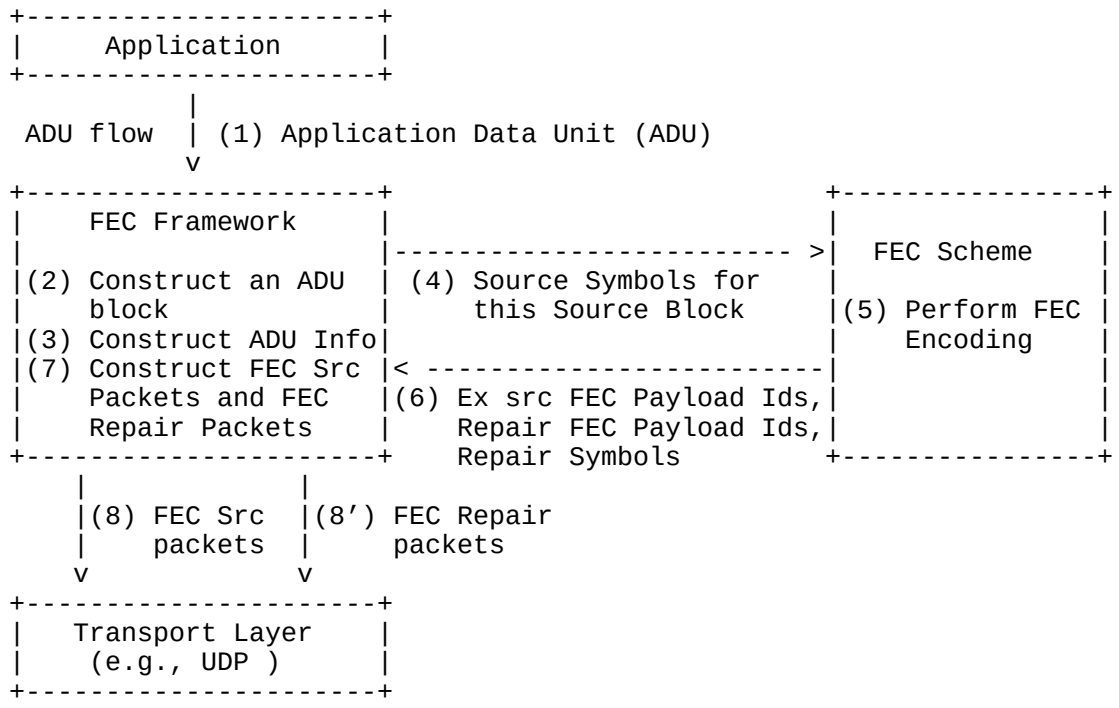

Figure 1: Terminology used in this document (sender).

3.2. Notations

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

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

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

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

E denotes the encoding symbol length in bytes.

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

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

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

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

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

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

3.3. Abbreviations

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

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

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

4.1. Restrictions

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

4.2. ADU Block Creation

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

We now detail each of these aspects.

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

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

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

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

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

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

4.3. Source Block Creation

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

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

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

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

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

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

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

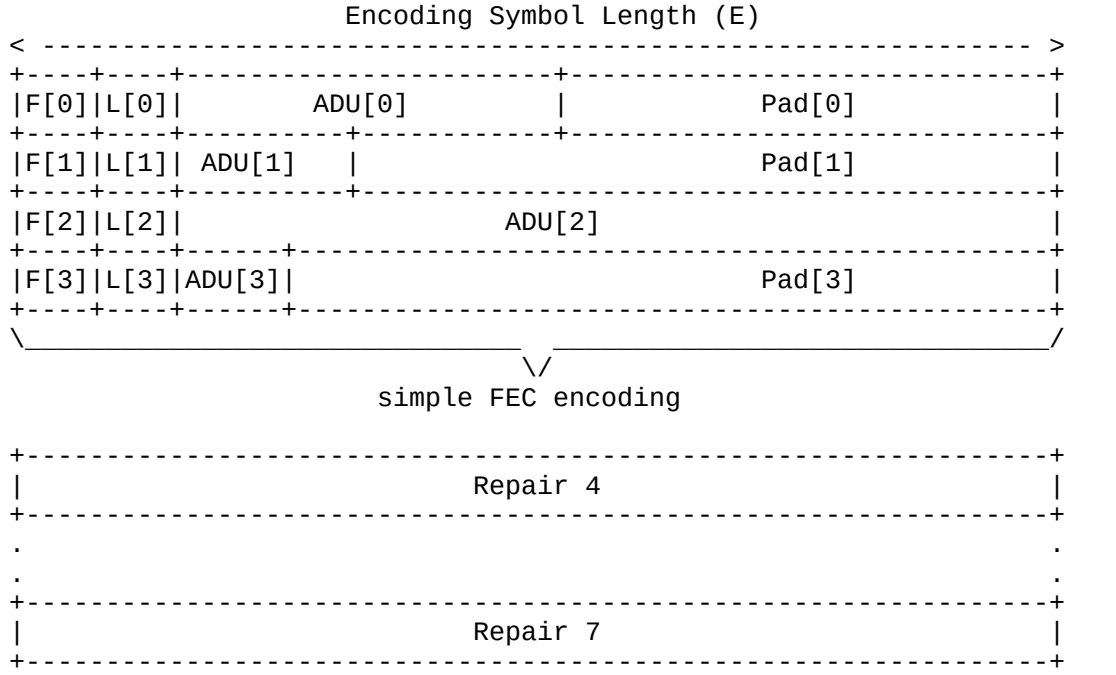

Figure 2: Source block creation, for code rate 1/2 (equal number of source and repair symbols, 4 in this example), and S = 0.

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

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

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

5.1. Formats and Codes

5.1.1. FEC Framework Configuration Information

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

5.1.1.1. Mandatory Information

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

5.1.1.2. FEC Scheme-Specific Information

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

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

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

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

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

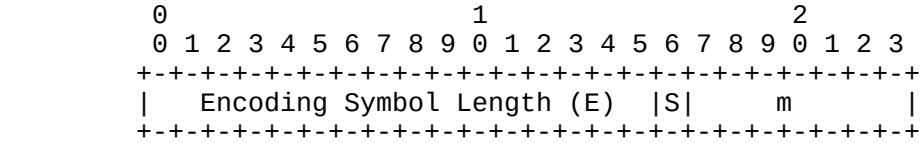

Figure 3: FSSI encoding format.

5.1.2. Explicit Source FEC Payload ID

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

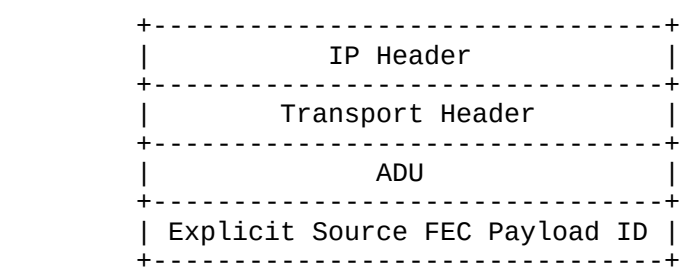

Figure 4: Structure of a FEC Source Packet with the Explicit Source FEC Payload ID.

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

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

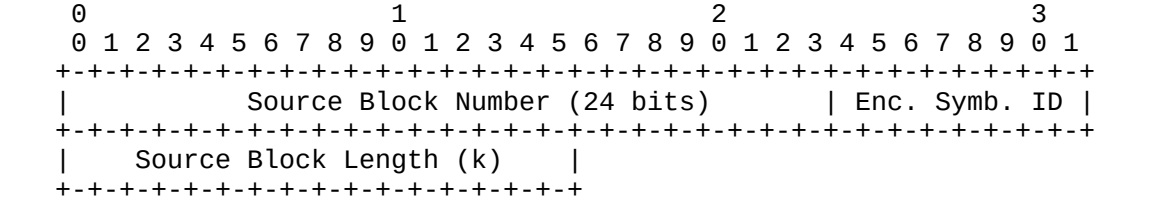

Figure 5: Source FEC Payload ID encoding format for m = 8 (default).

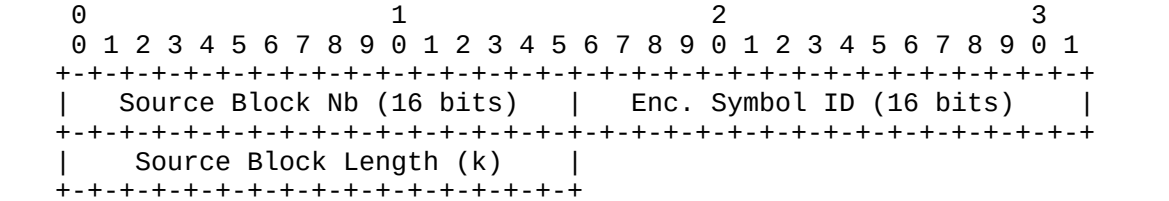

Figure 6: Source FEC Payload ID encoding format for m = 16.

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

5.1.3. Repair FEC Payload ID

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

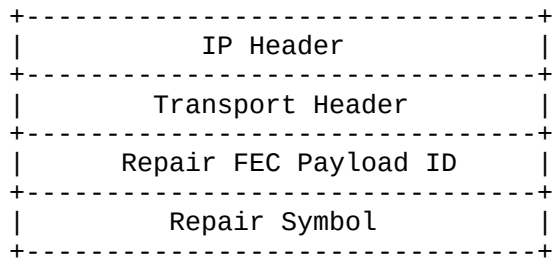

Figure 7: Structure of a FEC Repair Packet with the Repair FEC Payload ID.

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

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

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

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

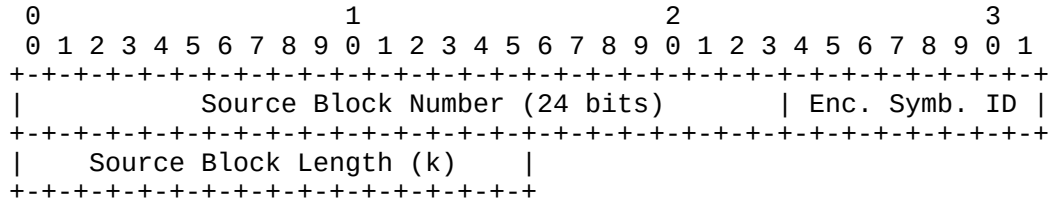

Figure 8: Repair FEC Payload ID encoding format for m = 8 (default).

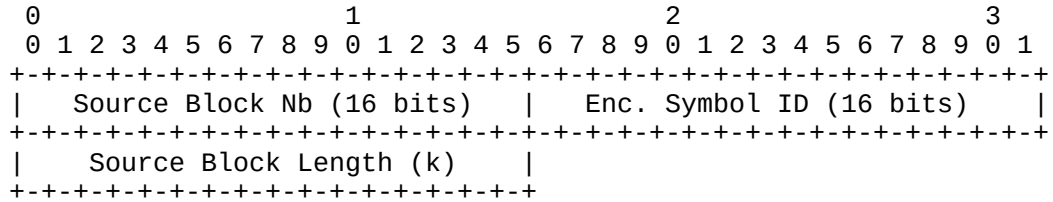

Figure 9: Repair FEC Payload ID encoding format for m = 16.

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

5.2. Procedures

The following procedures apply:

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

5.3. FEC Code Specification

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

6. Security Considerations

6.1. Problem Statement

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

More specifically, these attacks may have several goals:

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

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

6.2. Attacks Against the Data Flow

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

6.2.1. Access to Confidential Contents

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

6.2.2. Content Corruption

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

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

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

6.3. Attacks Against the FEC Parameters

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

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

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

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

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

7. IANA Considerations

The FEC Encoding ID value is subject to IANA registration.

TBD

8. Acknowledgments

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

9. References

9.1. Normative References

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

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

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

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

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

9.2. Informative References

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

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

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

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

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

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

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

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

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

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

Authors' Addresses

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

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

Mathieu Cunche
NICTA
Australia

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

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

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

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

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

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

Email: kazuhisa@sfc.wide.ad.jp