Forward Error Correction (FEC) Framework Extension to Sliding Window Codes (RFC 8680)

Vincent Roca, Ali Begen

Stream: Internet Engineering Task Force (IETF)
RFC: 8680
Updates: 6363
Category: Standards Track
Published: January 2020
ISSN: 2070-1721
Authors: V. Roca, *INRIA*; A. Begen, *Networked Media*

RFC 8680
Forward Error Correction (FEC) Framework Extension to Sliding Window Codes

Abstract

RFC 6363 describes a framework for using Forward Error Correction (FEC) codes 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. However, FECFRAME as per RFC 6363 is restricted to block FEC codes. This document updates RFC 6363 to support FEC codes based on a sliding encoding window, in addition to block FEC codes, in a backward-compatible way. During multicast/broadcast real-time content delivery, the use of sliding window codes significantly improves robustness in harsh environments, with less repair traffic and lower FEC-related added latency.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8680.

Copyright Notice

Table of Contents

1. Introduction

Many applications need to transport a continuous stream of packetized data from a source (sender) to one or more destinations (receivers) over networks that do not provide guaranteed packet delivery. In particular, packets may be lost, which is strictly the focus of this document: we assume that transmitted packets are either lost (e.g., because of a congested router, a poor signal-to-noise ratio in a wireless network, or because the number of bit errors exceeds the correction capabilities of the physical-layer error-correcting code) or were received by the transport protocol without any corruption (i.e., the bit errors, if any, have been fixed by the physical-layer error-correcting code and therefore are hidden to the upper layers).

For these use cases, Forward Error Correction (FEC) applied within the transport or application layer is an efficient technique to improve packet transmission robustness in the presence of packet losses (or "erasures") without going through packet retransmissions that create a delay often incompatible with real-time constraints. The FEC Building Block defined in [RFC5052] provides a framework for the definition of Content Delivery Protocols (CDPs) that make use of separately defined FEC schemes. Any CDP defined according to the requirements of the FEC Building Block can then easily be used with any FEC scheme that is also defined according to the requirements of the FEC Building Block.

Then, FECFRAME [RFC6363] provides a framework to define Content Delivery Protocols (CDPs) that provide FEC protection for arbitrary packet flows over an unreliable datagram service transport, such as UDP. It is primarily intended for real-time or streaming media applications that are using broadcast, multicast, or on-demand delivery. A subset of FECFRAME is currently part of the 3GPP Evolved Multimedia Broadcast/Multicast Service (eMBMS) standard [MBMSTS].

However, [RFC6363] only considers block FEC schemes defined in accordance with the FEC Building Block [RFC5052] (e.g., [RFC6681], [RFC6816], or [RFC6865]). These codes require the input flow(s) to be segmented into a sequence of blocks. Then, FEC encoding (at a sender or an encoding middlebox) and decoding (at a receiver or a decoding middlebox) are both performed on a per-block basis. For instance, if the current block encompasses the 100's to 119's source symbols (i.e., a block of size 20 symbols) of an input flow, encoding (and decoding) will be performed on this block independently of other blocks. This approach has major impacts on FEC encoding and decoding delays. The data packets of continuous media flow(s) may be passed to the transport layer immediately, without delay. But the block creation time, which depends on the number of source symbols in this block, impacts both the FEC encoding delay (since encoding requires that all source symbols be known) and, mechanically, the packet loss recovery delay at a receiver (since no repair symbol for the current block can be generated and therefore received before that time). Therefore, a good value for the block size is necessarily a balance between the maximum FEC decoding latency at the receivers (which must be in line with the most stringent real-time requirement of the protected flow(s), hence an incentive to reduce the block size) and the desired robustness against long loss bursts (which increases with the block size, hence an incentive to increase this size).

This document updates [RFC6363] in order to also support FEC codes based on a sliding encoding window (a.k.a., convolutional codes) [RFC8406]. This encoding window, either fixed or variable size, slides over the set of source symbols. FEC encoding is launched whenever needed from the set of source symbols present in the sliding encoding window at that time. This approach significantly reduces FEC-related latency, since repair symbols can be generated and passed to the transport layer on the fly at any time and can be regularly received by receivers to quickly recover packet losses. Using sliding window FEC codes is therefore highly beneficial to real-time flows, one of the primary targets of FECFRAME. [RFC8681] provides an example of such a FEC scheme for FECFRAME, which is built upon the simple sliding window Random Linear Code (RLC).

This document is fully backward compatible with [RFC6363]. Indeed:

- This FECFRAME update does not prevent or compromise in any way the support of block FEC codes. Both types of codes can nicely coexist, just like different block FEC schemes can coexist.
- Each sliding window FEC scheme is associated with a specific FEC Encoding ID subject to IANA registration, just like block FEC schemes.
- Any receiver -- for instance, a legacy receiver that only supports block FEC schemes -- can easily identify the FEC scheme used in a FECFRAME session. Indeed, the FEC Encoding ID that identifies the FEC scheme is carried in FEC Framework Configuration Information (see Section 5.5 of [RFC6363]). For instance, when the Session Description Protocol (SDP) is used to carry the FEC Framework Configuration Information, the FEC Encoding ID can be communicated in the "encoding-id=" parameter of a "fec-repair-flow" attribute [RFC6364]. This mechanism is the basic approach for a FECFRAME receiver to determine whether or not it supports the FEC scheme used in a given FECFRAME session.

This document leverages on [RFC6363] and reuses its structure. It proposes new sections specific to sliding window FEC codes whenever required. The only exception is Section 3, which provides a quick summary of FECFRAME in order to facilitate the understanding of this document to readers not familiar with the concepts and terminology.

2. Terminology

2.1. Definitions and Abbreviations

The following list of definitions and abbreviations is copied from [RFC6363], adding only the Block FEC Code, Sliding Window FEC Code, and Encoding/Decoding Window definitions (tagged with "ADDED"):

Application Data Unit (ADU):
: The unit of source data provided as a payload to the transport layer. For instance, it can be a payload containing the result of the RTP packetization of a compressed video frame.

ADU Flow:

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

AL-FEC:
Application-Layer Forward Error Correction.

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

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

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

Block FEC Code: (ADDED)
A FEC code that operates on blocks, i.e., for which the input flow **MUST** be segmented into a sequence of blocks, with FEC encoding and decoding being performed independently on a per-block basis.

Sliding Window FEC Code: (ADDED)
A FEC code that can generate repair symbols on the fly, at any time, from the set of source symbols present in the sliding encoding window at that time. These codes are also known as convolutional codes.

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

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

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

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

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

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

Repair Flow:
: The packet flow carrying FEC data.

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

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

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

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

Transport Protocol:
: The protocol used for the transport of the source and repair flows. This protocol needs to provide an unreliable datagram service, as UDP does ([RFC6363], Section 7).

Encoding Window: (ADDED)
: Set of source symbols available at the sender/coding node that are used (with a Sliding Window FEC code) to generate a repair symbol.

Decoding Window: (ADDED)
: Set of received or decoded source and repair symbols available at a receiver that are used (with a Sliding Window FEC code) to decode lost source symbols.

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 lost (e.g., in our case, by a congested router, or because the number of transmission errors exceeds the correction capabilities of the physical-layer code) or received. When a packet is received, it is assumed that this packet is not corrupted (i.e., in our case, the bit errors, if any, are fixed by the physical-layer code and are therefore hidden to the upper layers).

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

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

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

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

2.2. Requirements Language

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Summary of Architecture Overview

The architecture of Section 3 of [RFC6363] equally applies to this FECFRAME extension and is not repeated here. However, this section includes a quick summary to facilitate the understanding of this document to readers not familiar with the concepts and terminology.

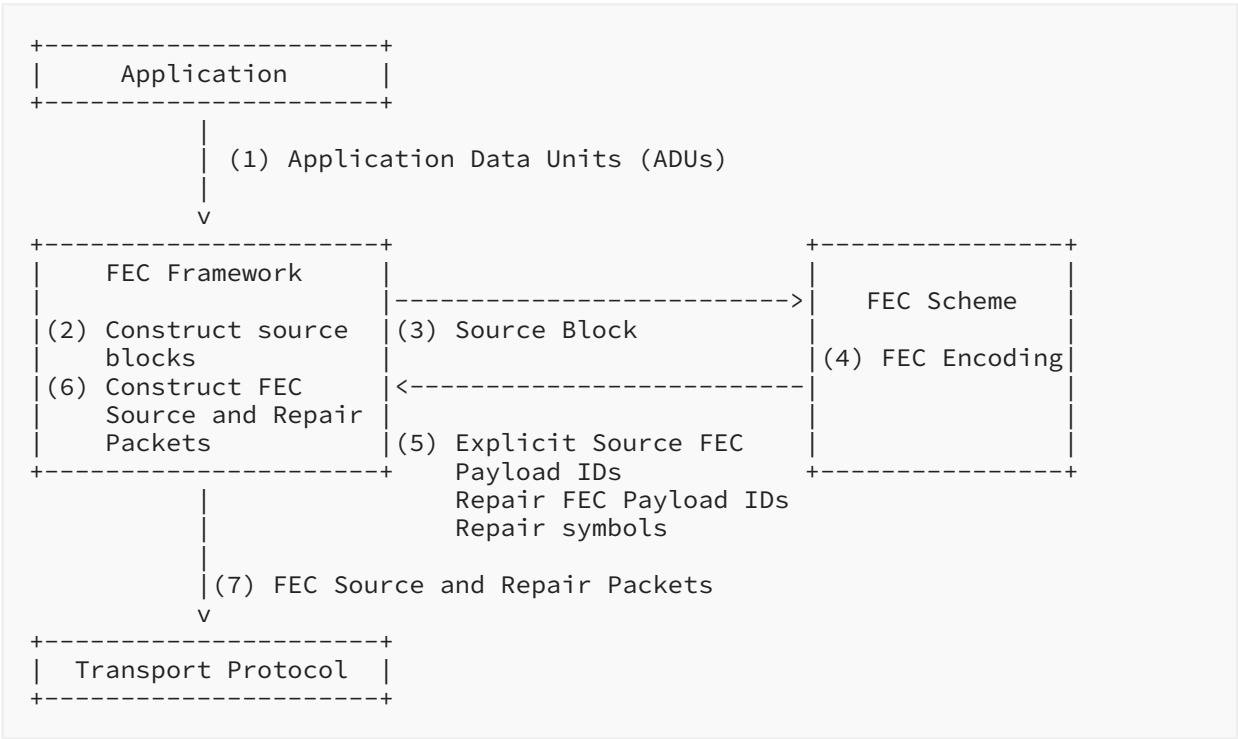

Figure 1: FECFRAME Architecture at a Sender

The FECFRAME architecture is illustrated in Figure 1 for a block FEC scheme from the sender's point of view. It shows an application generating an ADU flow (other flows from other applications may coexist). These ADUs of variable size must be somehow mapped to source symbols of a fixed size (this fixed size is a requirement of all FEC schemes, which comes from the way mathematical operations are applied to the symbols' content). This is the goal of an ADU-to-symbols mapping process that is FEC scheme specific (see below). Once the source block is built, taking into account both the FEC scheme constraints (e.g., in terms of maximum source block size) and the application's flow constraints (e.g., in terms of real-time constraints), the associated source symbols are handed to the FEC scheme in order to produce an appropriate number of repair symbols. FEC Source Packets (containing ADUs) and FEC Repair Packets (containing one or more repair symbols each) are then generated and sent using an appropriate transport protocol (more precisely, Section 7 of [RFC6363] requires a transport protocol providing an unreliable datagram service, such as UDP). In practice, FEC Source Packets may be passed to the transport layer as soon as available without having to wait for FEC encoding to take place. In that case, a copy of the associated source symbols needs to be kept within FECFRAME for future FEC encoding purposes.

At a receiver (not shown), FECFRAME processing operates in a similar way, taking as input the incoming FEC Source and Repair Packets received. In case of FEC Source Packet losses, the FEC decoding of the associated block may recover all (in case of successful decoding) or a subset that is potentially empty (if decoding fails) of the missing source symbols. After source-symbol-to-ADU mapping, when lost ADUs are recovered, they are then assigned to their respective flow (see below). ADUs are returned to the application(s), either in their initial transmission order (in which case all ADUs received after a lost ADU will be delayed until FEC decoding has taken place) or not (in which case each ADU is returned as soon as it is received or recovered), depending on the application requirements.

FECFRAME features two subtle mechanisms whose details are FEC scheme dependent:

- ADUs-to-source-symbols mapping: in order to manage variable size ADUs, FECFRAME and FEC schemes can use small, fixed-size symbols and create a mapping between ADUs and symbols. The mapping details are FEC scheme dependent and must be defined in the associated document. For instance, with certain FEC schemes, to each ADU, this mechanism prepends a length field (plus a flow identifier; see below) and pads the result to a multiple of the symbol size. A small ADU may be mapped to a single source symbol, while a large one may be mapped to multiple symbols.
- Assignment of decoded ADUs to flows in multi-flow configurations: when multiple flows are multiplexed over the same FECFRAME instance, a problem is to assign a decoded ADU to the right flow (UDP port numbers and IP addresses traditionally used to map incoming ADUs to flows are not recovered during FEC decoding). The mapping details are FEC scheme dependent and must be defined in the associated document. For instance, with certain FEC schemes, to make it possible, at the FECFRAME sending instance, each ADU is prepended with a flow identifier (1 byte) during the ADU-to-source-symbols mapping (see above). The flow identifiers are also shared between all FECFRAME instances as part of the FEC Framework Configuration Information. The ADU Information (ADUI), which includes the flow identifier, length, application payload, and padding, is then FEC protected. Therefore, a

decoded ADUI contains enough information to assign the ADU to the right flow. Note that a FEC scheme may also be restricted to the particular case of a single flow over a FECFRAME instance; that would make the above mechanism pointless.

A few aspects are not covered by FECFRAME, namely:

- Section 8 of [RFC6363] does not detail any congestion control mechanisms and only provides high-level normative requirements.
- The possibility of having feedback from receiver(s) is considered out of scope, although such a mechanism may exist within the application (e.g., through RTP Control Protocol (RTCP) messages).
- Flow adaptation at a FECFRAME sender (e.g., how to set the FEC code rate based on transmission conditions) is not detailed, but it needs to comply with the congestion control normative requirements (see above).

4. Procedural Overview

4.1. General

The general considerations of Section 4.1 of [RFC6363] that are specific to block FEC codes are not repeated here.

With a Sliding Window FEC code, the FEC Source Packet **MUST** contain information to identify the position occupied by the ADU within the source flow in terms specific to the FEC scheme. This information is known as the Source FEC Payload ID, and the FEC scheme is responsible for defining and interpreting it.

With a Sliding Window FEC code, the FEC Repair Packets **MUST** contain information that identifies the relationship between the contained repair payloads and the original source symbols used during encoding. This information is known as the Repair FEC Payload ID, and the FEC scheme is responsible for defining and interpreting it.

The sender operation ([RFC6363], Section 4.2) and receiver operation ([RFC6363], Section 4.3) are both specific to block FEC codes and are therefore omitted below. The following two sections detail similar operations for Sliding Window FEC codes.

4.2. Sender Operation with Sliding Window FEC Codes

With a Sliding Window FEC scheme, the following operations, illustrated in Figure 2 for the generic case (non-RTP repair flows) and in Figure 3 for the case of RTP repair flows, describe a possible way to generate compliant source and repair flows:

1. A new ADU is provided by the application.
2. The FEC Framework communicates this ADU to the FEC scheme.
3. The sliding encoding window is updated by the FEC scheme. The ADU-to-source-symbol mapping as well as the encoding window management details are both the responsibility of

the FEC scheme and **MUST** be detailed there. Appendix A provides non-normative hints about what FEC scheme designers need to consider.

4. The Source FEC Payload ID information of the source packet is determined by the FEC scheme. If required by the FEC scheme, the Source FEC Payload ID is encoded into the Explicit Source FEC Payload ID field and returned to the FEC Framework.
5. The FEC Framework constructs the FEC Source Packet according to Figure 6 in [RFC6363], using the Explicit Source FEC Payload ID provided by the FEC scheme if applicable.
6. The FEC Source Packet is sent using normal transport-layer procedures. This packet is sent using the same ADU flow identification information as would have been used for the original source packet if the FEC Framework were not present (e.g., the source and destination addresses and UDP port numbers on the IP datagram carrying the source packet will be the same whether or not the FEC Framework is applied).
7. When the FEC Framework needs to send one or several FEC Repair Packets (e.g., according to the target code rate), it asks the FEC scheme to create one or several repair packet payloads from the current sliding encoding window along with their Repair FEC Payload ID.
8. The Repair FEC Payload IDs and repair packet payloads are provided back by the FEC scheme to the FEC Framework.
9. The FEC Framework constructs FEC Repair Packets according to Figure 7 in [RFC6363], using the FEC Payload IDs and repair packet payloads provided by the FEC scheme.
10. The FEC Repair Packets are sent using normal transport-layer procedures. The port(s) and multicast group(s) to be used for FEC Repair Packets are defined in the FEC Framework Configuration Information.

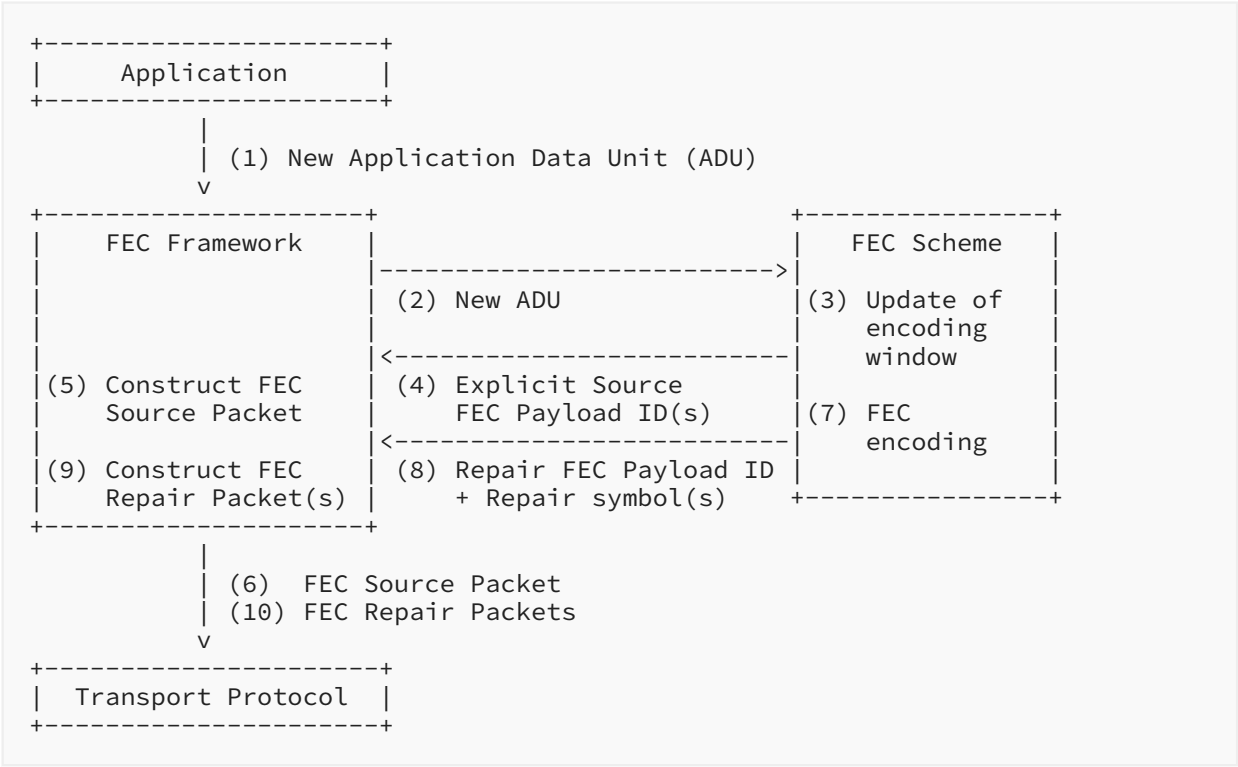

Figure 2: Sender Operation with Sliding Window FEC Codes

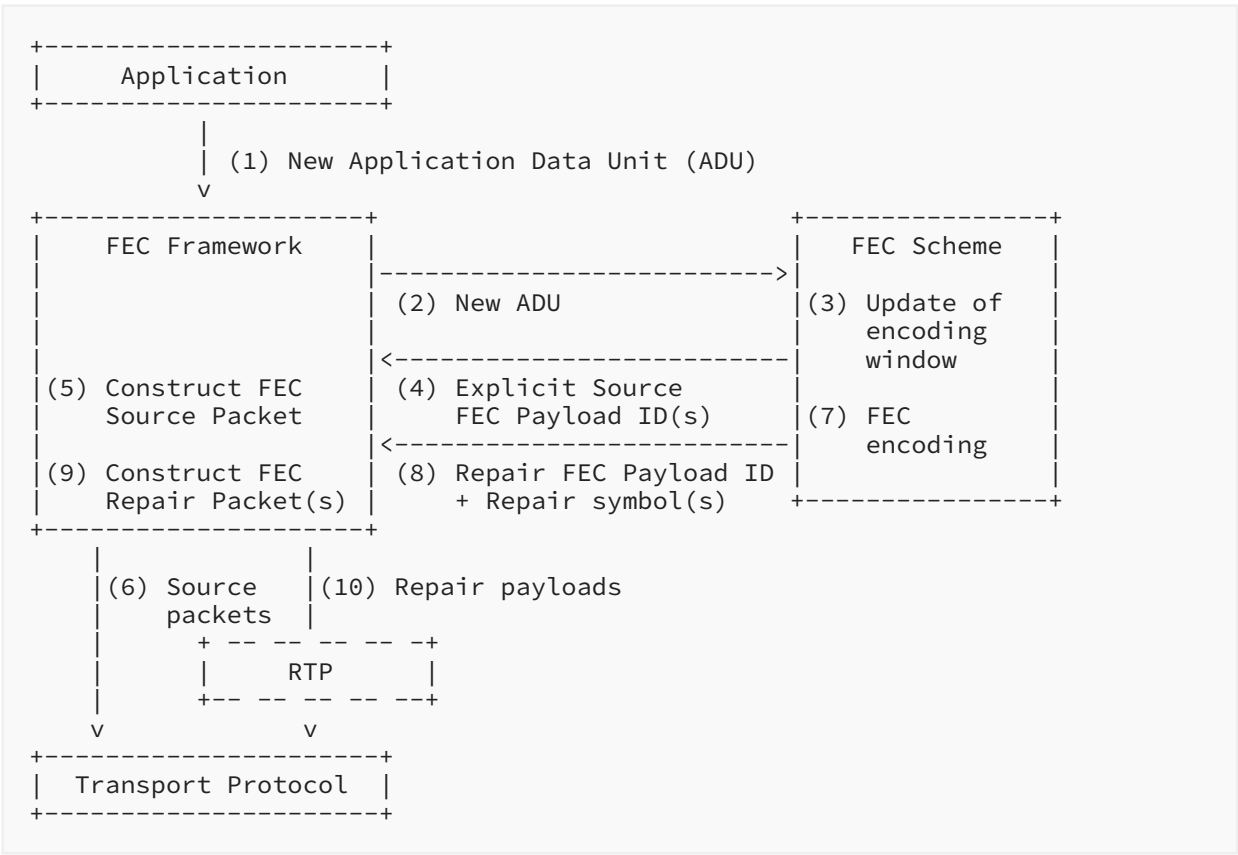

Figure 3: Sender Operation with Sliding Window FEC Codes and RTP Repair Flows

4.3. Receiver Operation with Sliding Window FEC Codes

With a Sliding Window FEC scheme, the following operations are illustrated in Figure 4 for the generic case (non-RTP repair flows) and in Figure 5 for the case of RTP repair flows. The only differences with respect to block FEC codes lie in steps (4) and (5). Therefore, this section does not repeat the other steps of Section 4.3 of [RFC6363] ("Receiver Operation"). The new steps (4) and (5) are:

1. The FEC scheme uses the received FEC Payload IDs (and derived FEC Source Payload IDs when the Explicit Source FEC Payload ID field is not used) to insert source and repair packets into the decoding window in the right way. If at least one source packet is missing and at least one repair packet has been received, then FEC decoding is attempted to recover the missing source payloads. The FEC scheme determines whether source packets have been lost and whether enough repair packets have been received to decode any or all of the missing source payloads.
2. The FEC scheme returns the received and decoded ADUs to the FEC Framework, along with indications of any ADUs that were missing and could not be decoded.

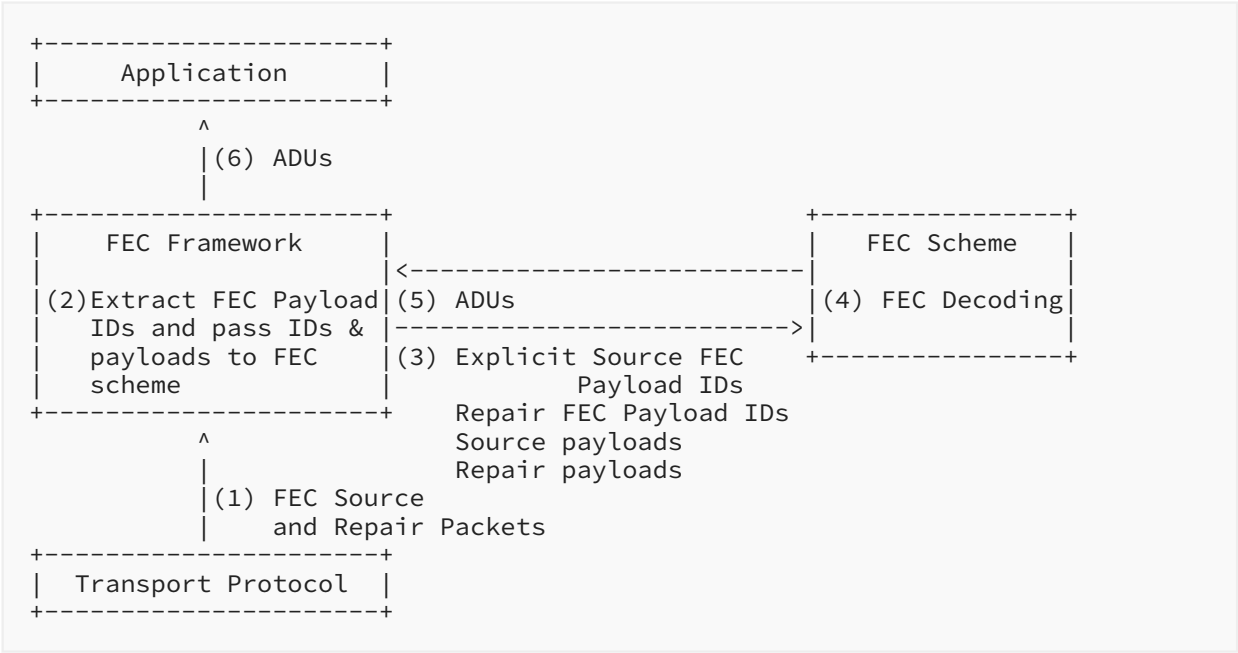

Figure 4: Receiver Operation with Sliding Window FEC Codes

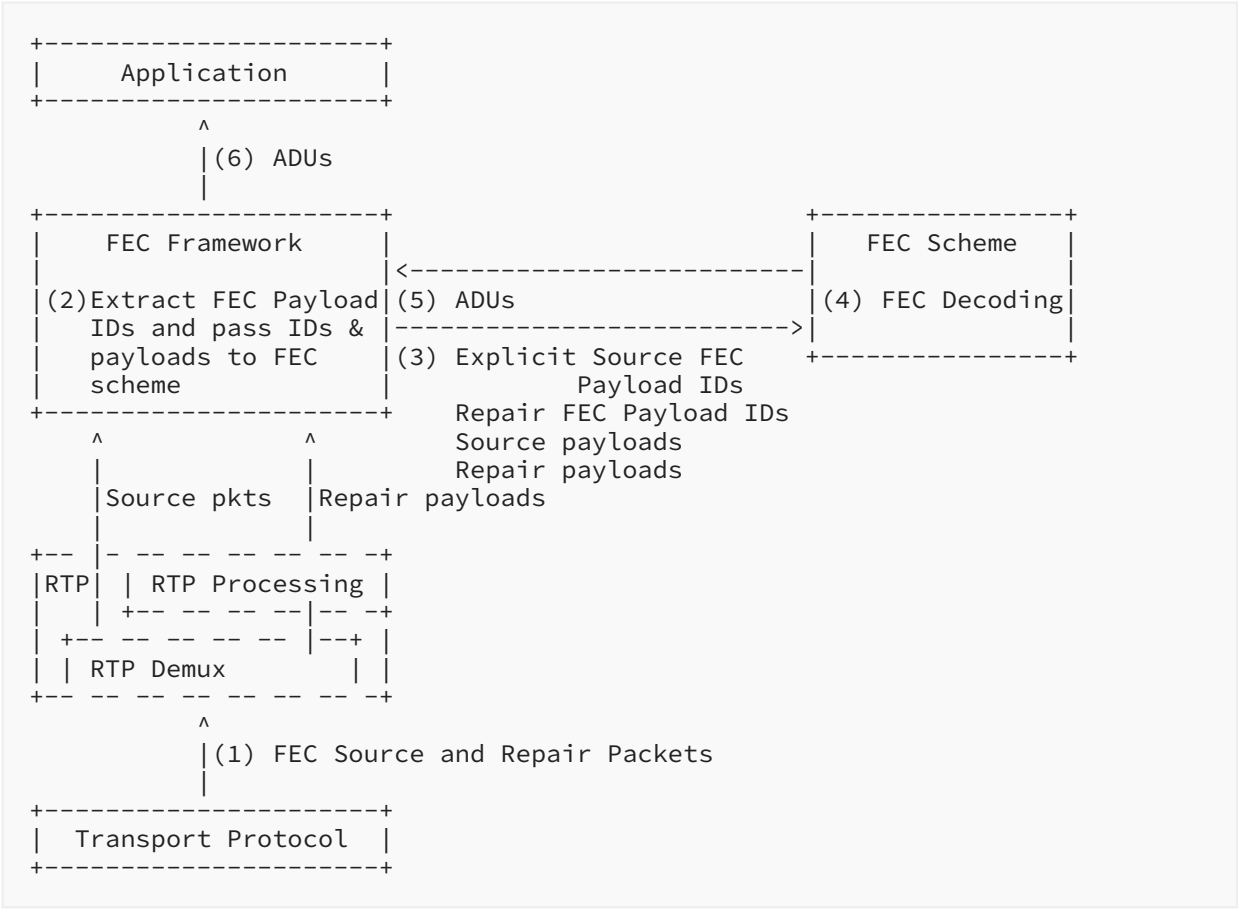

Figure 5: Receiver Operation with Sliding Window FEC Codes and RTP Repair Flows

5. Protocol Specification

5.1. General

This section discusses the protocol elements for the FEC Framework specific to Sliding Window FEC schemes. The global formats of source data packets (i.e., [RFC6363], Figure 6) and repair data packets (i.e., [RFC6363], Figures 7 and 8) remain the same with Sliding Window FEC codes. They are not repeated here.

5.2. FEC Framework Configuration Information

The FEC Framework Configuration Information considerations of Section 5.5 of [RFC6363] equally apply to this FECFRAME extension and are not repeated here.

5.3. FEC Scheme Requirements

The FEC scheme requirements of Section 5.6 of [RFC6363] mostly apply to this FECFRAME extension and are not repeated here. An exception, though, is the "full specification of the FEC code", item (4), which is specific to block FEC codes. In case of a Sliding Window FEC scheme, then the following item (4-bis) applies:

4-bis.
A full specification of the Sliding Window FEC code.

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

Furthermore, given valid values of the FEC-Scheme-Specific Information, a valid Repair FEC Payload ID value, a valid packet payload size, and a valid encoding window (i.e., a set of source symbols), the specification **MUST** uniquely define the values of the encoding symbol (or symbols) to be included in the repair packet payload with the given Repair FEC Payload ID value.

Additionally, the FEC scheme associated with a Sliding Window FEC code:

- **MUST** define the relationships between ADUs and the associated source symbols (mapping).
- **MUST** define the management of the encoding window that slides over the set of ADUs. Appendix A provides non-normative hints about what FEC scheme designers need to consider.
- **MUST** define the management of the decoding window. This usually consists of managing a system of linear equations (for a linear FEC code).

6. Feedback

The discussion in Section 6 of [RFC6363] equally applies to this FECFRAME extension and is not repeated here.

7. Transport Protocols

The discussion in Section 7 of [RFC6363] equally applies to this FECFRAME extension and is not repeated here.

8. Congestion Control

The discussion in Section 8 of [RFC6363] equally applies to this FECFRAME extension and is not repeated here.

9. Security Considerations

This FECFRAME extension does not add any new security considerations. All the considerations of Section 9 of [RFC6363] apply to this document as well. However, for the sake of completeness, the following goal can be added to the list provided in Section 9.1 of [RFC6363] ("Problem Statement"):

- Attacks can try to corrupt source flows in order to modify the receiver application's behavior (as opposed to just denying service).

10. Operations and Management Considerations

This FECFRAME extension does not add any new Operations and Management Considerations. All the considerations of Section 10 of [RFC6363] apply to this document as well.

11. IANA Considerations

This document has no IANA actions.

A FEC scheme for use with this FEC Framework is identified via its FEC Encoding ID. It is subject to IANA registration in the "FEC Framework (FECFRAME) FEC Encoding IDs" registry. All the rules of Section 11 of [RFC6363] apply and are not repeated here.

12. References

12.1. Normative References

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

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

12.2. Informative References

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

[RFC5052]

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

[RFC6364] Begen, A., "Session Description Protocol Elements for the Forward Error Correction (FEC) Framework", RFC 6364, DOI 10.17487/RFC6364, October 2011, <https://www.rfc-editor.org/info/rfc6364>.

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

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

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

[RFC8406] Adamson, B., Adjih, C., Bilbao, J., Firoiu, V., Fitzek, F., Ghanem, S., Lochin, E., Masucci, A., Montpetit, M-J., Pedersen, M., Peralta, G., Roca, V., Ed., Saxena, P., and S. Sivakumar, "Taxonomy of Coding Techniques for Efficient Network Communications", RFC 8406, DOI 10.17487/RFC8406, June 2018, <https://www.rfc-editor.org/info/rfc8406>.

[RFC8681] Roca, V. and B. Teibi, "Sliding Window Random Linear Code (RLC) Forward Erasure Correction (FEC) Schemes for FECFRAME", RFC 8681, DOI 10.17487/RFC8681, January 2020, <https://www.rfc-editor.org/info/rfc8681>.

Appendix A. About Sliding Encoding Window Management (Informational)

The FEC Framework does not specify the management of the sliding encoding window, which is the responsibility of the FEC scheme. This annex only provides a few informational hints.

Source symbols are added to the sliding encoding window each time a new ADU is available at the sender after the ADU-to-source-symbol mapping specific to the FEC scheme has been done.

Source symbols are removed from the sliding encoding window. For instance:

- After a certain delay, when an "old" ADU of a real-time flow times out. The source symbol retention delay in the sliding encoding window should therefore be initialized according to the real-time features of incoming flow(s) when applicable.
- Once the sliding encoding window has reached its maximum size (there is usually an upper limit to the sliding encoding window size). In that case, the oldest symbol is removed each time a new source symbol is added.

Several considerations can impact the management of this sliding encoding window:

- At the source flows level: real-time constraints can limit the total time during which source symbols can remain in the encoding window.
- At the FEC code level: theoretical or practical limitations (e.g., because of computational complexity) can limit the number of source symbols in the encoding window.
- At the FEC scheme level: signaling and window management are intrinsically related. For instance, an encoding window composed of a nonsequential set of source symbols requires appropriate signaling to inform a receiver of the composition of the encoding window, and the associated transmission overhead can limit the maximum encoding window size. On the contrary, an encoding window always composed of a sequential set of source symbols simplifies signaling: providing the identity of the first source symbol plus its number is sufficient, which creates a fixed and relatively small transmission overhead.

Acknowledgments

The authors would like to thank Christer Holmberg, David Black, Gorry Fairhurst, Emmanuel Lochin, Spencer Dawkins, Ben Campbell, Benjamin Kaduk, Eric Rescorla, Adam Roach, and Greg Skinner for their valuable feedback on this document. This document being an extension of [RFC6363], the authors would also like to thank Mark Watson as the main author of that RFC.

Authors' Addresses

Vincent Roca
INRIA
Univ. Grenoble Alpes
France
Email: vincent.roca@inria.fr

Ali Begen
Networked Media
Konya/
Turkey
Email: ali.begen@networked.media