

Network Working Group
Uberti
Internet-Draft
Google
Intended status: Standards Track
2019
Expires: January 4, 2020

J.

Jul 3,

WebRTC Forward Error Correction Requirements
draft-ietf-rtcweb-fec-09

Abstract

This document provides information and requirements for how Forward Error Correction (FEC) should be used by WebRTC implementations.

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 <https://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 January 4, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Uberti
1]

Expires January 4, 2020

[Page

Table of Contents

[1](#). Introduction
[2](#)
[2](#). Terminology
[2](#)
[3](#). Types of FEC
[2](#)
 [3.1](#). Separate FEC Stream
[3](#)
 [3.2](#). Redundant Encoding
[3](#)
 [3.3](#). Codec-Specific In-band FEC
[3](#)
[4](#). FEC for Audio Content
[4](#)
 [4.1](#). Recommended Mechanism
[4](#)
 [4.2](#). Negotiating Support
[5](#)
[5](#). FEC for Video Content
[5](#)
 [5.1](#). Recommended Mechanism
[5](#)
 [5.2](#). Negotiating Support
[6](#)
[6](#). FEC for Application Content
[6](#)
[7](#). Implementation Requirements
[7](#)
[8](#). Adaptive Use of FEC
[7](#)
[9](#). Security Considerations
[8](#)
[10](#). IANA Considerations
[8](#)
[11](#). Acknowledgements
[8](#)
[12](#). References
[8](#)
 [12.1](#). Normative References
[8](#)
 [12.2](#). Informative References
[9](#)
[Appendix A](#). Change log
[11](#)
Author's Address
[13](#)

[1](#). **Introduction**

In situations where packet loss is high, or perfect media quality is

essential, Forward Error Correction (FEC) can be used to proactively recover from packet losses. This specification provides guidance on which FEC mechanisms to use, and how to use them, for WebRTC implementations.

2. Terminology

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. Types of FEC

FEC describes the sending of redundant information in an outgoing packet stream so that information can still be recovered even in the face of packet loss. There are multiple ways this can be

accomplished for RTP media streams [[RFC3550](#)]; this section enumerates the various mechanisms available and describes their tradeoffs.

3.1. Separate FEC Stream

This approach, as described in [[RFC5956](#)], [Section 4.3](#), sends FEC packets as an independent RTP stream with its own synchronization source (SSRC, [[RFC3550](#)]) and payload type, multiplexed with the primary encoding. While this approach can protect multiple packets of the primary encoding with a single FEC packet, each FEC packet will have its own IP+UDP+RTP+FEC header, and this overhead can be excessive in some cases, e.g., when protecting each primary packet with a FEC packet.

This approach allows for recovery of entire RTP packets, including the full RTP header.

3.2. Redundant Encoding

This approach, as described in [[RFC2198](#)], allows for redundant data to be piggybacked on an existing primary encoding, all in a single packet. This redundant data may be an exact copy of a previous payload, or for codecs that support variable-bitrate encodings, possibly a smaller, lower-quality representation. In certain cases, the redundant data could include encodings of multiple prior audio frames.

Since there is only a single set of packet headers, this approach allows for a very efficient representation of primary + redundant data. However, this savings is only realized when the data all fits into a single packet (i.e. the size is less than a MTU). As a result, this approach is generally not useful for video content.

As described in [[RFC2198](#)], [Section 4](#), this approach cannot recover certain parts of the RTP header, including the marker bit, CSRC information, and header extensions.

3.3. Codec-Specific In-band FEC

Some audio codecs, notably Opus [[RFC6716](#)] and AMR [[RFC4867](#)], support their own in-band FEC mechanism, where redundant data is included in the codec payload. This is similar to the redundant encoding mechanism described above, but as it adds no additional framing, it can be slightly more efficient.

For Opus, audio frames deemed important are re-encoded at a lower bitrate and appended to the next payload, allowing partial recovery of a lost packet. This scheme is fairly efficient; experiments

Uberti
3]

Expires January 4, 2020

[Page

performed indicate that when Opus FEC is used, the overhead imposed is only about 20-30%, depending on the amount of protection needed. Note that this mechanism can only carry redundancy information for the immediately preceding audio frame; as such the decoder cannot fully recover multiple consecutive lost packets, which can be a problem on wireless networks. See [\[RFC6716\]](#), [Section 2.1.7](#), and this

Opus mailing list post [\[OpusFEC\]](#) for more details.

For AMR/AMR-WB, packets can contain copies or lower-quality encodings of multiple prior audio frames. See [\[RFC4867\]](#), [Section 3.7.1](#) for details on this mechanism.

In-band FEC mechanisms cannot recover any of the RTP header.

4. FEC for Audio Content

The following section provides guidance on how to best use FEC for transmitting audio data. As indicated in [Section 8](#) below, FEC should

only be activated if network conditions warrant it, or upon explicit application request.

4.1. Recommended Mechanism

When using variable-bitrate codecs without an internal FEC, redundant encoding (as described in [Section 3.2](#)) with lower-fidelity version(s) of the previous packet(s) is RECOMMENDED. This provides reasonable protection of the payload with only moderate bitrate increase, as the redundant encodings can be significantly smaller than the primary encoding.

When using the Opus codec, use of the built-in Opus FEC mechanism is RECOMMENDED. This provides reasonable protection of the audio stream against individual losses, with minimal overhead. Note that, as indicated above, the built-in Opus FEC only provides single-frame redundancy; if multi-packet protection is needed, the aforementioned redundant encoding with reduced-bitrate Opus encodings SHOULD be used instead.

When using the AMR/AMR-WB codecs, use of their built-in FEC mechanism is RECOMMENDED. This provides slightly more efficient protection of the audio stream than redundant encoding.

When using constant-bitrate codecs, e.g., PCMU [\[RFC5391\]](#), redundant

encoding MAY be used, but this will result in a potentially significant bitrate increase, and suddenly increasing bitrate to deal with losses from congestion may actually make things worse.

Because of the lower packet rate of audio encodings, usually a single packet per frame, use of a separate FEC stream comes with a higher overhead than other mechanisms, and therefore is NOT RECOMMENDED.

As mentioned above, the recommended mechanisms do not allow recovery of parts of the RTP header that may be important in certain audio applications, e.g., CSRCs and RTP header extensions like those specified in [\[RFC6464\]](#) and [\[RFC6465\]](#). Implementations SHOULD account

for this and attempt to approximate this information, using an approach similar to those described in [\[RFC2198\], Section 4](#), and [\[RFC6464\], Section 5](#).

4.2. Negotiating Support

Support for redundant encoding of a given RTP stream SHOULD be indicated by including audio/red [\[RFC2198\]](#) as an additional supported media type for the associated m= section in the SDP offer [\[RFC3264\]](#). Answerers can reject the use of redundant encoding by not including the audio/red media type in the corresponding m= section in the SDP answer.

Support for codec-specific FEC mechanisms are typically indicated via "a=fmtp" parameters.

For Opus, a receiver MUST indicate that it is prepared to use incoming FEC data with the "useinbandfec=1" parameter, as specified in [\[RFC7587\]](#). This parameter is declarative and can be negotiated separately for either media direction.

For AMR/AMR-WB, support for redundant encoding, and the maximum supported depth, are controlled by the 'max-red' parameter, as specified in [\[RFC4867\], Section 8.1](#). Receivers MUST include this parameter, and set it to an appropriate value, as specified in [\[TS.26114\]](#), Table 6.3.

5. FEC for Video Content

The following section provides guidance on how to best use FEC for transmitting video data. As indicated in [Section 8](#) below, FEC should only be activated if network conditions warrant it, or upon explicit application request.

5.1. Recommended Mechanism

Video frames, due to their size, often require multiple RTP packets. As discussed above, a separate FEC stream can protect multiple packets with a single FEC packet. In addition, the "flexfec" FEC

mechanism described in [[I-D.ietf-payload-flexible-fec-scheme](#)] is also

Uberti
5]

Expires January 4, 2020

[Page

capable of protecting multiple RTP streams via a single FEC stream, including all the streams that are part of a BUNDLE [[I-D.ietf-mmusic-sdp-bundle-negotiation](#)] group. As a result, for video content, use of a separate FEC stream with the flexfec RTP payload format is RECOMMENDED.

To process the incoming FEC stream, the receiver can demultiplex it by SSRC, and then correlate it with the appropriate primary stream(s) via the CSRC(s) present in the RTP header of flexfec repair packets, or the SSRC field present in the FEC header of flexfec retransmission packets.

5.2. Negotiating Support

Support for a SSRC-multiplexed flexfec stream to protect a given RTP stream SHOULD be indicated by including one of the formats described in [[I-D.ietf-payload-flexible-fec-scheme](#)], Section 5.1.2, as an additional supported media type for the associated m= section in the SDP offer [[RFC3264](#)]. As mentioned above, when BUNDLE is used, only a single flexfec repair stream will be created for each BUNDLE group, even if flexfec is negotiated for each primary stream.

Answerers can reject the use of SSRC-multiplexed FEC, by not including the offered FEC formats in the corresponding m= section in the SDP answer.

Use of FEC-only m-lines, and grouping using the SDP group mechanism as described in [[RFC5956](#)], [Section 4.1](#) is not currently defined for WebRTC, and SHOULD NOT be offered.

Answerers SHOULD reject any FEC-only m-lines, unless they specifically know how to handle such a thing in a WebRTC context (perhaps defined by a future version of the WebRTC specifications).

6. FEC for Application Content

WebRTC also supports the ability to send generic application data, and provides transport-level retransmission mechanisms to support full and partial (e.g. timed) reliability. See [[I-D.ietf-rtcweb-data-channel](#)] for details.

Because the application can control exactly what data to send, it has the ability to monitor packet statistics and perform its own application-level FEC, if necessary.

As a result, this document makes no recommendations regarding FEC for the underlying data transport.

Uberti
6]

Expires January 4, 2020

[Page

7. Implementation Requirements

To support the functionality recommended above, implementations MUST be able to receive and make use of the relevant FEC formats for their supported audio codecs, and MUST indicate this support, as described in [Section 4](#). Use of these formats when sending, as mentioned above, is RECOMMENDED.

The general FEC mechanism described in [\[I-D.ietf-payload-flexible-fec-scheme\]](#) SHOULD also be supported, as mentioned in [Section 5](#).

Implementations MAY support additional FEC mechanisms if desired, e.g., [\[RFC5109\]](#).

8. Adaptive Use of FEC

Because use of FEC always causes redundant data to be transmitted, and the total amount of data must remain within any bandwidth limits indicated by congestion control and the receiver, this will lead to less bandwidth available for the primary encoding, even when the redundant data is not being used. This is in contrast to methods like RTX [\[RFC4588\]](#) or flexfec's retransmission mode ([\[I-D.ietf-payload-flexible-fec-scheme\]](#), Section 1.1.7), which only transmit redundant data when necessary, at the cost of an extra roundtrip.

Given this, WebRTC implementations SHOULD consider using RTX or flexfec retransmissions instead of FEC when RTT is low, and SHOULD only transmit the amount of FEC needed to protect against the observed packet loss (which can be determined, e.g., by monitoring transmit packet loss data from RTCP Receiver Reports [\[RFC3550\]](#)), unless the application indicates it is willing to pay a quality penalty to proactively avoid losses.

Note that when probing bandwidth, i.e., speculatively sending extra data to determine if additional link capacity exists, FEC SHOULD be used in all cases. Given that extra data is going to be sent regardless, it makes sense to have that data protect the primary payload; in addition, FEC can be applied in a way that increases bandwidth only modestly, which is necessary when probing.

When using FEC with layered codecs, e.g., [\[RFC6386\]](#), where only base layer frames are critical to the decoding of future frames, implementations SHOULD only apply FEC to these base layer frames.

Finally, it should be noted that although applying redundancy is often useful in protecting a stream against packet loss, if the loss

Uberti
7]

Expires January 4, 2020

[Page

is caused by network congestion, the additional bandwidth used by the redundant data may actually make the situation worse, and can lead to significant degradation of the network.

9. Security Considerations

In the WebRTC context, FEC is specifically concerned with recovering data from lost packets; any corrupted packets will be discarded by the SRTP [[RFC3711](#)] decryption process. Therefore, as described in [[RFC3711](#)], [Section 10](#), the default processing when using FEC with SRTP is to perform FEC followed by SRTP at the sender, and SRTP followed by FEC at the receiver. This ordering is used for all the SRTP Protection Profiles used in DTLS-SRTP [[RFC5763](#)], which are enumerated in [[RFC5764](#)], [Section 4.1.2](#).

Additional security considerations for each individual FEC mechanism are enumerated in their respective documents.

10. IANA Considerations

This document requires no actions from IANA.

11. Acknowledgements

Several people provided significant input into this document, including Bernard Aboba, Jonathan Lennox, Giri Mandyam, Varun Singh, Tim Terriberry, Magnus Westerlund, and Mo Zanaty.

12. References

12.1. Normative References

- [I-D.ietf-payload-flexible-fec-scheme]
Zanaty, M., Singh, V., Begen, A., and G. Mandyam, "RTP Payload Format for Flexible Forward Error Correction (FEC)", [draft-ietf-payload-flexible-fec-scheme-20](#) (work in progress), May 2019.
- [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>>.
- [RFC2198] Perkins, C., Kouvelas, I., Hodson, O., Hardman, V., Handley, M., Bolot, J., Vega-Garcia, A., and S. Fosse-Parisis, "RTP Payload for Redundant Audio Data", [RFC 2198](#), DOI 10.17487/RFC2198, September 1997, <<https://www.rfc-editor.org/info/rfc2198>>.

Uberti
8]

Expires January 4, 2020

[Page

- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", [RFC 3264](#), DOI 10.17487/RFC3264, June 2002, <<https://www.rfc-editor.org/info/rfc3264>>.
- [RFC4867] Sjoberg, J., Westerlund, M., Lakaniemi, A., and Q. Xie, "RTP Payload Format and File Storage Format for the Adaptive Multi-Rate (AMR) and Adaptive Multi-Rate (AMR-WB) Audio Codecs", [RFC 4867](#), DOI 10.17487/RFC4867, April 2007, <<https://www.rfc-editor.org/info/rfc4867>>.
- Wideband
- [RFC5956] Begen, A., "Forward Error Correction Grouping Semantics in the Session Description Protocol", [RFC 5956](#), DOI 10.17487/RFC5956, September 2010, <<https://www.rfc-editor.org/info/rfc5956>>.
- [RFC7587] Spittka, J., Vos, K., and JM. Valin, "RTP Payload Format for the Opus Speech and Audio Codec", [RFC 7587](#), DOI 10.17487/RFC7587, June 2015, <<https://www.rfc-editor.org/info/rfc7587>>.
- [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>>.
- [TS.26114] 3GPP, "IP Multimedia Subsystem (IMS); Multimedia telephony; Media handling and interaction", 3GPP TS 26.114 15.0.0, September 2017.

12.2. Informative References

- [I-D.ietf-mmusic-sdp-bundle-negotiation] Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", [draft-ietf-mmusic-sdp-bundle-negotiation-54](#) (work in progress), December 2018.
- [I-D.ietf-rtcweb-data-channel] Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channels", [draft-ietf-rtcweb-data-channel-13](#) (work in progress), January 2015.
- [OpusFEC] Terriberry, T., "Opus FEC", January 2013.

Uberti
9]

Expires January 4, 2020

[Page

- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, [RFC 3550](#), DOI 10.17487/RFC3550, July 2003, <<https://www.rfc-editor.org/info/rfc3550>>.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", [RFC 3711](#), DOI 10.17487/RFC3711, March 2004, <<https://www.rfc-editor.org/info/rfc3711>>.
- [RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", [RFC 4588](#), DOI 10.17487/RFC4588, July 2006, <<https://www.rfc-editor.org/info/rfc4588>>.
- [RFC5109] Li, A., Ed., "RTP Payload Format for Generic Forward Error Correction", [RFC 5109](#), DOI 10.17487/RFC5109, December 2007, <<https://www.rfc-editor.org/info/rfc5109>>.
- [RFC5391] Sollaud, A., "RTP Payload Format for ITU-T Recommendation G.711.1", [RFC 5391](#), DOI 10.17487/RFC5391, November 2008, <<https://www.rfc-editor.org/info/rfc5391>>.
- [RFC5763] Fischl, J., Tschofenig, H., and E. Rescorla, "Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS)", [RFC 5763](#), DOI 10.17487/RFC5763, May 2010, <<https://www.rfc-editor.org/info/rfc5763>>.
- [RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", [RFC 5764](#), DOI 10.17487/RFC5764, May 2010, <<https://www.rfc-editor.org/info/rfc5764>>.
- [RFC6386] Bankoski, J., Koleszar, J., Quillio, L., Salonen, J., Wilkins, P., and Y. Xu, "VP8 Data Format and Decoding Guide", [RFC 6386](#), DOI 10.17487/RFC6386, November 2011, <<https://www.rfc-editor.org/info/rfc6386>>.
- [RFC6464] Lennox, J., Ed., Ivov, E., and E. Marocco, "A Real-time Transport Protocol (RTP) Header Extension for Client-to-Mixer Audio Level Indication", [RFC 6464](#), DOI 10.17487/RFC6464, December 2011, <<https://www.rfc-editor.org/info/rfc6464>>.

Uberti
10]

Expires January 4, 2020

[Page

[RFC6465] Iovov, E., Ed., Marocco, E., Ed., and J. Lennox, "A Real-time Transport Protocol (RTP) Header Extension for Mixer-to-Client Audio Level Indication", [RFC 6465](#), DOI 10.17487/RFC6465, December 2011, <<https://www.rfc-editor.org/info/rfc6465>>.

[RFC6716] Valin, JM., Vos, K., and T. Terriberry, "Definition of the Opus Audio Codec", [RFC 6716](#), DOI 10.17487/RFC6716, September 2012, <<https://www.rfc-editor.org/info/rfc6716>>.

[Appendix A](#). Change log

Changes in draft -09:

- o Editorial changes from IETF LC.
- o Added new reference for Opus FEC.

Changes in draft -08:

- o Switch to [RFC 8174](#) boilerplate.

Changes in draft -07:

- o Clarify how bandwidth management interacts with FEC.
- o Make 3GPP reference normative.

Changes in draft -06:

- o Discuss how multiple streams can be protected by a single FlexFEC stream.
- o Discuss FEC for bandwidth probing.
- o Add note about recovery of RTP headers and header extensions.
- o Add note about FEC/SRTP ordering.
- o Clarify flexfec demux text, and mention retransmits.
- o Clarify text regarding offers/answers.
- o Make [RFC2198](#) support SHOULD strength.
- o Clean up references.

Changes in draft -05:

Uberti
11]

Expires January 4, 2020

[Page

- o No changes.

Changes in draft -04:

- o Discussion of layered codecs.
- o Discussion of RTX.
- o Clarified implementation requirements.
- o FlexFEC MUST -> SHOULD.
- o Clarified AMR max-red handling.
- o Updated references.

Changes in draft -03:

- o Added overhead stats for Opus.
- o Expanded discussion of multi-packet FEC for Opus.
- o Added discussion of AMR/AMR-WB.
- o Removed discussion of ssrc-group.
- o Referenced the data channel doc.
- o Referenced the RTP/RTCP RFC.
- o Several small edits based on feedback from Magnus.

Changes in draft -02:

- o Expanded discussion of FEC-only m-lines, and how they should be handled in offers and answers.

Changes in draft -01:

- o Tweaked abstract/intro text that was ambiguously normative.
- o Removed text on FEC for Opus in CELT mode.
- o Changed [RFC 2198](#) recommendation for PCMU to be MAY instead of NOT RECOMMENDED, based on list feedback.
- o Explicitly called out application data as something not addressed in this document.

- o Updated flexible-fec reference.

Changes in draft -00:

- o Initial version, from sidebar conversation at IETF 90.

Author's Address

Justin Uberti
Google
747 6th St S
Kirkland, WA 98033
USA

Email: justin@uberti.name

