

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 21, 2016

J. Uberti
Google
March 20, 2016

WebRTC Forward Error Correction Requirements
draft-ietf-rtcweb-fec-03

Abstract

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

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 21, 2016.

Copyright Notice

Copyright (c) 2016 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 (<http://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.

Internet-Draft

WebRTC FEC

March 2016

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 [4](#)

[5.](#) FEC for Video Content [5](#)

[5.1.](#) Recommended Mechanism [5](#)

[5.2.](#) Negotiating Support [5](#)

[6.](#) FEC for Application Content [6](#)

[7.](#) Implementation Requirements [6](#)

[8.](#) Adaptive Use of FEC [6](#)

[9.](#) Security Considerations [6](#)

[10.](#) IANA Considerations [7](#)

[11.](#) Acknowledgements [7](#)

[12.](#) References [7](#)

[12.1.](#) Normative References [7](#)

[12.2.](#) Informative References [7](#)

[Appendix A.](#) Change log [8](#)

Author's Address [9](#)

[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 client implementations.

[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 [[RFC2119](#)].

[3.](#) Types of FEC

By its name, 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 in which this can be accomplished; this section enumerates the various mechanisms and describes their tradeoffs.

[3.1.](#) Separate FEC Stream

This approach, as described in [\[RFC5956\]](#), [Section 4.3](#), sends FEC packets as an independent SSRC-multiplexed stream, with its own SSRC and payload type. While by far the most flexible, each FEC packet will have its own IP+UDP+RTP+FEC header, leading to additional overhead of the FEC stream.

[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 packet, or for codecs that support variable-bitrate encodings, possibly a smaller, lower-quality representation. In certain cases, the redundant data could include multiple prior packets.

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.

[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.

For Opus, packets deemed as important are re-encoded at a lower bitrate and added to the subsequent packet, allowing partial recovery of a lost packet. This scheme is fairly efficient; experiments performed indicate that when Opus FEC is used, the overhead imposed is about 20-30%, depending on the amount of protection needed. Note that this mechanism can only carry redundancy information for the immediately preceding packet; as such the decoder cannot fully

recover multiple consecutive lost packets. See [\[RFC6716\]](#), [Section 2.1.7](#) for complete details.

For AMR/AMR-WB, packets can contain copies or lower-quality encodings of multiple prior audio frames. This mechanism is similar to the [\[RFC2198\]](#) mechanism described above, but as it adds no additional framing, it can be slightly more efficient. See [\[RFC4867\]](#), [Section 3.7.1](#) for details on this mechanism.

[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 the Opus codec, use of the built-in Opus FEC mechanism is RECOMMENDED. This provides reasonable protection of the audio stream against typical losses, with modest overhead. Note that as indicated above the built-in Opus FEC only provides single-frame redundancy; if multi-packet protection is needed, the built-in FEC should be combined with [\[RFC2198\]](#) redundancy to protect the N-2th, N-3rd, etc. packets.

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 [\[RFC2198\]](#).

When using variable-bitrate codecs without an internal FEC, [\[RFC2198\]](#) redundant encoding with lower-fidelity version(s) of previous packet(s) is RECOMMENDED. This provides reasonable protection of the payload with moderate overhead.

When using constant-bitrate codecs, e.g. PCMU, use of [\[RFC2198\]](#) redundant encoding MAY be used, but note that this will result in a potentially significant bitrate increase, and that 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.

[4.2.](#) Negotiating Support

Support for redundant encoding can be indicated by offering "red" as a supported payload type in the offer. Answerers can reject the use of redundant encoding by not including "red" as a supported payload type in the answer.

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

For Opus, support for FEC at the received side is controlled by the "useinbandfec=1" parameter, as specified in

[[I-D.ietf-payload-rtp-opus](#)]. 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](#). [TODO: figure out any additional recommendations are needed.]

[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

For video content, use of a separate FEC stream with the RTP payload format described in [[I-D.ietf-payload-flexible-fec-scheme](#)] is RECOMMENDED. The receiver can demultiplex the incoming FEC stream by SSRC and correlate it with the primary stream via the SSRC field present in the FEC header.

Support for protecting multiple primary streams with a single FEC stream is complicated by WebRTC's 1-m-line-per-stream policy, which does not allow for a m-line dedicated specifically to FEC.

5.2. Negotiating Support

To offer support for a SSRC-multiplexed FEC stream that is associated with a given primary stream, the offerer **MUST** offer the formats supported for the primary stream, as well as one of the formats described in [[I-D.ietf-payload-flexible-fec-scheme](#)], Section 5.1.

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 can reject the use of SSRC-multiplexed FEC, by not including FEC formats in the answer.

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). This ensures that implementations will not malfunction when said future version of WebRTC enables offers of FEC-only m-lines.

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.

7. Implementation Requirements

To support the functionality recommended above, implementations MUST support the relevant mechanisms for their supported audio codecs, as described in [Section 4](#), and the general FEC mechanism described in [[I-D.ietf-payload-flexible-fec-scheme](#)].

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

[8](#). Adaptive Use of FEC

Since use of FEC causes redundant data to be transmitted, this will lead to less bandwidth available for the primary encoding, when in a bandwidth-constrained environment. Given this, WebRTC implementations 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](#)]), or the application indicates it is willing to pay a quality penalty to proactively avoid losses.

[9](#). Security Considerations

This document makes recommendations regarding the use of FEC. Generally, it should be noted that although applying redundancy is often useful in protecting a stream against packet loss, if the loss 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.

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 Jonathan Lennox, Giri Mandyam, Varun Singh, Tim Terriberry,

and Mo Zanaty.

12. References

12.1. Normative References

- [I-D.ietf-payload-flexible-fec-scheme]
Singh, V., Begen, A., Zanaty, M., and G. Mandyam, "RTP Payload Format for Flexible Forward Error Correction (FEC)", [draft-ietf-payload-flexible-fec-scheme-01](#) (work in progress), October 2015.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [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, <<http://www.rfc-editor.org/info/rfc2198>>.
- [RFC5956] Begen, A., "Forward Error Correction Grouping Semantics in the Session Description Protocol", [RFC 5956](#), DOI 10.17487/RFC5956, September 2010, <<http://www.rfc-editor.org/info/rfc5956>>.

12.2. Informative References

- [I-D.ietf-payload-rtp-opus]
Spittka, J., Vos, K., and J. Valin, "RTP Payload Format for the Opus Speech and Audio Codec", [draft-ietf-payload-rtp-opus-11](#) (work in progress), April 2015.
- [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.

Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, [RFC 3550](#), DOI 10.17487/RFC3550, July 2003, <<http://www.rfc-editor.org/info/rfc3550>>.

[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 Wideband (AMR-WB) Audio Codecs", [RFC 4867](#), DOI 10.17487/RFC4867, April 2007, <<http://www.rfc-editor.org/info/rfc4867>>.

[RFC5109] Li, A., Ed., "RTP Payload Format for Generic Forward Error Correction", [RFC 5109](#), DOI 10.17487/RFC5109, December 2007, <<http://www.rfc-editor.org/info/rfc5109>>.

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

[Appendix A](#). Change log

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

Uberti

Expires September 21, 2016

[Page 9]