Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 30, 2015

J. Uberti
Google

October 27, 2014

WebRTC Forward Error Correction Requirements
draft-uberti-rtcweb-fec-00

Abstract

This document makes recommendations 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 April 30, 2015.

Copyright Notice

Copyright (c) 2014 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.

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 3
4.1. Recommended Mechanism 3
4.2. Negotiating Support 4
5. FEC for Video Content 4
5.1. Recommended Mechanism 4
5.2. Negotiating Support 5
6. Implementation Requirements 5
7. Adaptive Use of FEC 5
8. Security Considerations 5
9. IANA Considerations 5
10. Acknowledgements 5
11. References 6
11.1. Normative References 6
11.2. Informative References 6
Appendix A. Change log 6
Author's Address 6

1. Introduction

In situations where packet loss is high, or media quality must be perfect, Forward Error Correction (FEC) can be used to proactively recover from packet losses. This document describes what FEC mechanisms should be used by 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 descibed in [RFC2198], allows for redundant data to be piggybacked on an existing primary encoding 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. Since there is only a single set of packet headers, this allows for a very efficient representation of primary + redundant data. However, this savings is only realized when the two encodings both fit into a single packet (i.e. less than a MTU). This approach is also only applicable to audio content.

3.3. Codec-Specific In-band FEC

Some audio codecs, notably Opus [RFC6716], support their own in-band FEC mechanism, where FEC data is included in the codec payload. In the case of Opus specifically, packets deemed as important are re-encoded at a lower bitrate and added to the subsequent packet, allowing partial recovery of a lost packet. See [RFC6716], Section 2.1.7 for details.

4. FEC for Audio Content

The following section provides guidance on how to best use FEC for transmitting audio data. As indicated in Section 7 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 in its default (hybrid) mode, use of the built-in Opus FEC mechanism is RECOMMENDED. This provides reasonable protection of the audio stream against typical losses, with moderate overhead. [TODO: add stats] Note though that this mechanism only protects the SILK layer of the Opus codec; the CELT portion is not protected. This is not an issue when Opus is running in hybrid mode, as the lower frequencies will still be able to be recovered, with minimal quality impact.

When using Opus in CELT mode, or other variable-bitrate codecs, use of [RFC2198] redundant encoding with a lower-fidelity version of the previous packet is RECOMMENDED. When using Opus specifically, the lower-fidelity version can simply be a truncated version of the previous Opus packet. [TODO: decide exact truncated size] This provides reasonable protection of the payload with minimal overhead.

When using constant-bitrate codecs, e.g. PCMU, use of [RFC2198] redundant encoding is NOT RECOMMENDED, as this will result in a potentially significant bitrate increase. Furthermore, suddenly increasing the bitrate to deal with packet losses 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 specifically, this is controlled by the "useinbandfec=1" parameter, as specified in [I-D.ietf-payload-rtp-opus]. These parameters are declarative and can be negotiated separately for either media direction.

5. FEC for Video Content

The following section provides guidance on how to best use FEC for transmitting video data. As indicated in Section 7 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.singh-payload-rtp-1d2d-parity-scheme] is RECOMMENDED. The receiver can demultiplex the incoming FEC stream by SSRC and correlate it with the primary stream via the ssrc-group mechanism.

Note that this only allows the FEC stream to protect a single primary stream. Support for protecting multiple primary streams with a

single FEC stream is complicated by WebRTC's 1-m-line-per-stream policy and requires further study.

5.2. Negotiating Support

To offer support for a separate FEC stream, the offerer MUST offer one of the formats described in [I-D.singh-payload-rtp-1d2d-parity-scheme], Section 5.1, as well as a ssrc-group with "FEC-FR" semantics as described in [RFC5956], Section 4.3.

Answerers can reject the use of FEC by not including FEC payloads in the answer.

6. Implementation Requirements

To support the functionality recommended above, implementations MUST support the redundant encoding mechanism described in [RFC2198] and the FEC mechanism described in [RFC5956] and [I-D.singh-payload-rtp-1d2d-parity-scheme].

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

7. 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 FEC data when network conditions indicate that this is advisable (e.g. by monitoring transmit packet loss data from RTCP Receiver Reports), or the application indicates it is willing to pay a quality penalty to proactively avoid losses.

8. Security Considerations

TODO

9. IANA Considerations

This document requires no actions from IANA.

10. Acknowledgements

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

11. References

11.1. Normative References

[I-D.singh-payload-rtp-1d2d-parity-scheme]
Singh, V., Begen, A., and M. Zanaty, "RTP Payload Format for Non-Interleaved and Interleaved Parity Forward Error Correction (FEC)", draft-singh-payload-rtp-1d2d-parity-scheme-00 (work in progress), October 2014.

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

[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, September 1997.

[RFC5956] Begen, A., "Forward Error Correction Grouping Semantics in the Session Description Protocol", RFC 5956, September 2010.

11.2. Informative References

[I-D.ietf-payload-rtp-opus]
Spittka, J., Vos, K., and J. Valin, "RTP Payload Format for Opus Speech and Audio Codec", draft-ietf-payload-rtp-opus-03 (work in progress), July 2014.

[RFC5109] Li, A., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, December 2007.

[RFC6716] Valin, JM., Vos, K., and T. Terriberry, "Definition of the Opus Audio Codec", RFC 6716, September 2012.

Appendix A. Change log

Changes in draft -00:

o Initial version, from sidebar conversation at IETF 90.

Author's Address

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

Email: justin@uberti.name