RTCWEB
Internet-Draft
Intended status: Informational
Expires: August 2, 2013

G. Mandyam
Qualcomm Innovation Center
M. Luby
Qualcomm Technologies Inc.
T. Stockhammer
Nomor Research
C. Foisy
Qualcomm Technologies Inc.

January 29, 2013

Forward Error Correction for WebRTC using FEC FRAME
draft-mandyam-rtcweb-fecframe-00

Abstract

WebRTC provides a solution for peer-to-peer streaming between web applications by leveraging a Real-Time Protocol (RTP) stream between two clients. This RTP stream is expected to be sent over an UDP (Universal Datagram Protocol) connection, which by definition has no built-in reliability. Recently the FEC FRAME Working Group of the IETF has come up with a framework and technical recommendations for applying forward error correction (FEC) to unreliable streams. This framework can be applied to WebRTC with minimal changes to the specification.

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 August 2, 2013.

Copyright Notice

Table of Contents

1. Introduction

Forward Error Correction (FEC) is a well-known technique for improving the reliability of packet transmission over networks that do not provide guaranteed packet delivery by adding repair packets to the original packets. The IETF has defined a "building block" approach to the specification of FEC to streaming protocols, based on RFC 5052 [RFC5052] and RFC 6363 [RFC6363]. Borrowing from the terminology of RFC 5052 [RFC5052], the FEC can be applied to any payload of a CDP (content delivery protocol). Since WebRTC defines RTP as its target CDP for media streaming, it stands to reason that the IETF building block framework is readily applicable to WebRTC.

1.1. Block Code Introduction

Detailed understanding of all the different types of FEC is beyond the scope of this document. Note that FEC in its most fundamental description involves the encoding of a message derived from an alphabet into a representation that is also derived from the same alphabet. Moreover, FEC encoding results in redundancy, i.e. the addition of information to the message that can help in retrieving the original message in the presence of loss of parts of the transmission. If the message is composed of k individual entries from the alphabet and the FEC encoding results in a new collection of entries (also referred to as a codeword) of length n, then the FEC code is said to be of rate k/n or is sometimes said to be a (k,n) code.

FEC’s that operate on individual messages without dependency on other messages in a given sequence are sometimes referred to as block codes. As another way of visualizing this, assume that a message to be sent over a communications channel is derived from a grouping of symbols derived from an alphabet (e.g. a binary alphabet can be represented by the set {0,1}), and can be represented as a collection of words {m_0, m_1, ..., m_(k-1)}. Then the resultant codeword generated by applying the FEC can be represented as a collection of words derived from the same alphabet {c_0, c_1, ..., c_(n-1)}. If all n of the words of this codeword are sent over a communications link and at most n-k of the words of the codeword are lost, then ideally an FEC code can completely recover the original message.

The block code can be represented in terms of a generator matrix G (of binary entries) acting upon a message vector m.

```
G = |g_(0,0)   g_(0,1)   ... g_(0,n-1)  |, m =|m_0 m_1 ... m_(k-1)|
    |g_(1,0)   g_(1,1)   ... g_(1,n-1)  |
    |                ...                |
    |g_(k-1,0) g_(k-1,1) ... g_(k-1,n-1)|
```

Generator Matrix and Message Vector

Figure 1

A codeword c is generated by a matrix-vector multiplication of G with m. Some of the words of the codeword c, each word sent in a separate packet together with a word identifier, may be lost and not arrive at the receiver. The receiver can determined which words are received in packets from the word identifiers included in the packets. Based on the word identifiers, the FEC decoder can recover the original message.

If the block code in question is systematic, then a subset of the generated codeword is the original message itself. Such codes allow for a clean separation of the codeword into source symbols and redundancy symbols. In this case, then generator matrix can be represented as

```
G = |g_(0,0)   g_(0,1)   ... g_(0,n-k-1)  1 0 0 ... 0|
    |g_(1,0)   g_(1,1)   ... g_(1,n-k-1)  0 1 0 ... 0|
    |                ...                    ...    |
    |g_(k-1,0) g_(k-1,1) ... g_(k-1,n-k-1)0 0 0 ... 1|
```

Generator Matrix for Systematic Code

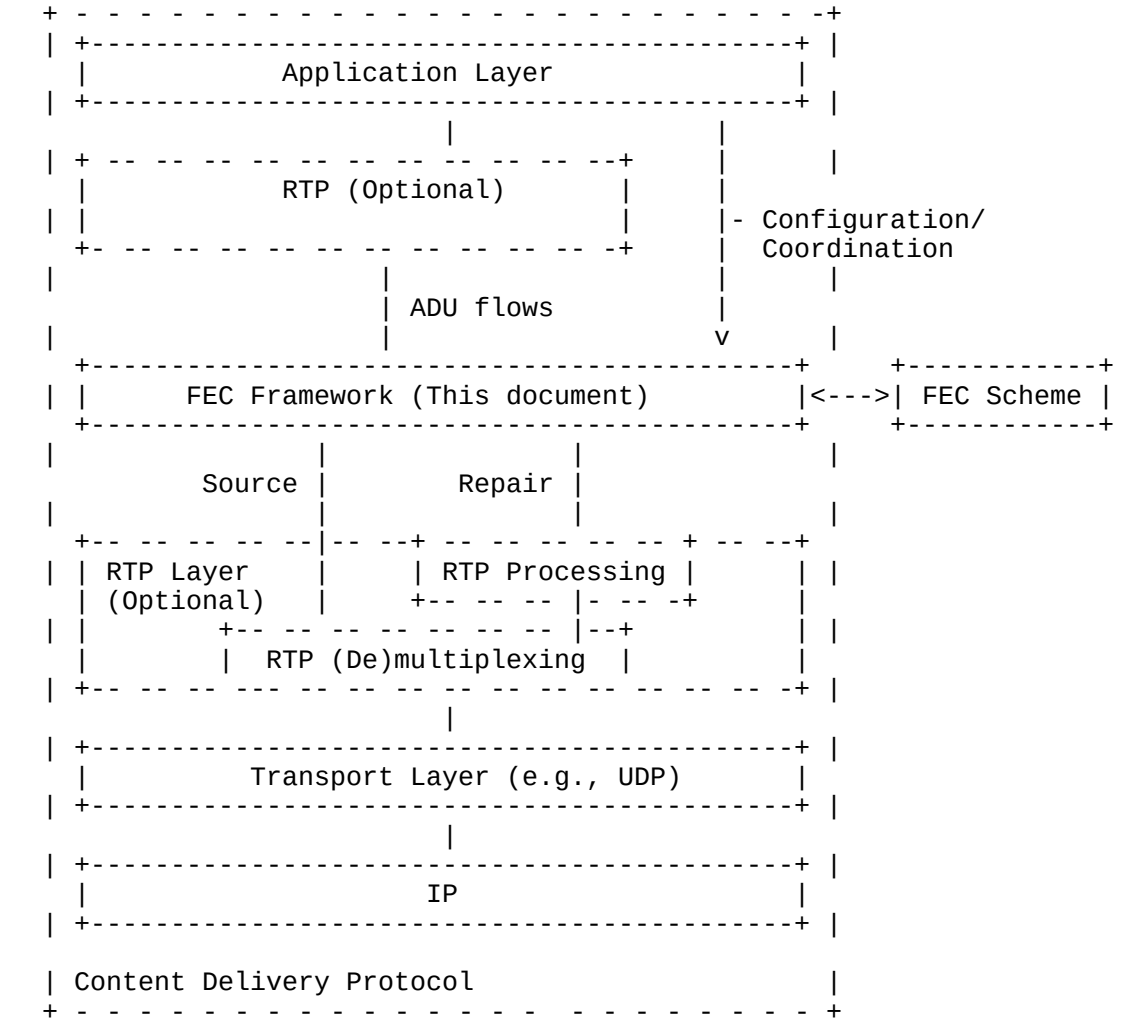
Figure 2

A more detailed overview of block codes, their derivation, and theoretical analysis can be found in textbooks such as [Wicker].

2. FEC FRAME Overview

The building block approach of RFC 6363 [RFC6363] allows for a straightforward application of a preferred block code to streaming protocols such as RTP. The chosen block code must satisfy the requirements of RFC 5052 [RFC5052]. A valid FEC encoding scheme will have an IANA-assigned FEC Encoding ID. Since the building block approach to applying block codes to streaming protocols has been standardized by the IETF, it is not necessary to discuss the

specified approach in this document. However, a simple mapping between the Framework Architecture of RFC 6363 [RFC6363] is reproduced here to provide context.



FEC Framework Architecture

Figure 3

With respect to the above figure, the application layer would essentially be a logical collection of the user agent along with the

WebRTC-enabled web application, and the application data unit (ADU) flows could be the RTP packets provided by the user agent implementation of WebRTC. Note that it is also allowed in the specification to multiplex additional RTP-encapsulated redundancy packets onto UDP, which is useful for providing additional resiliency for multicast traffic. After application of block encoding, the encoded blocks can be identified by an FEC Source ID which is appended to the block of data. It is not required if there are other means to indicate to the receiver a unique identifier for the encoded data block.

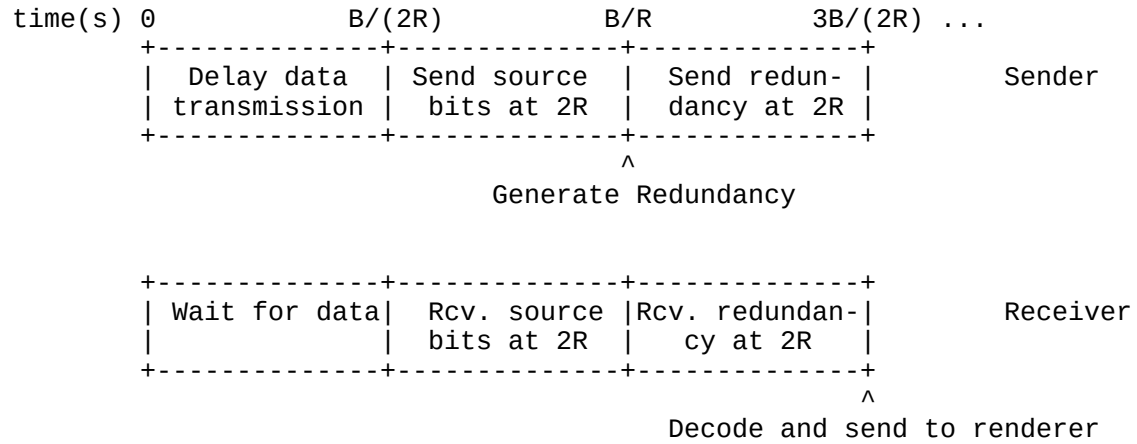
It should be noted that the use of FEC does not come without its own costs. For instance, the selection of the block size (ADU size) has a direct impact on latency as the transmission of the stream must be delayed while the payload is formed for FEC encoding. This is important due to the fact that many block coding schemes tend to be more effective with larger block sizes. Moreover, encoding and decoding latency are applicable (although advancement in processor speed has made this less of an issue). Finally, the amount of redundancy affects the overall throughput. The larger the amount of redundancy, the less likelihood that random losses will result in the receiver being unable to decode data. However, greater redundancy (i.e. a smaller k/n ratio) has a direct impact on the amount of application data that can be sent over a fixed time interval.

3. Latency Mitigation

While FEC encoding provides resiliency in the face of packet loss, it also introduces latency. Assume a system where the source rate for a stream is R bits/second, and the number of bits to be encoded using a k/n code is B. In order to encode B bits to produce the necessary redundancy, which is in the amount given by B(n-k)/k, it is necessary for the sender to buffer a block of B information bits. Therefore, one would assume that the sender would have to delay transmission by at least the time it takes to generate one block of information bits from the source, i.e. B/R. However, note that the in most block codes, the first part of the codeword sent is the actual source information bits. Therefore it is conceivable that transmission from the sender to receiver could commence while the source bits are being buffered at the sender-side encoder.

Assume that the rate of the code is k/n = 1/2, i.e. the number of redundancy bits is equal to the number of source bits when encoding a block of data. The sender, as one latency mitigation strategy, could delay transmission of the encoded data (source and redundancy bits) by half the duration of a source block, B/(2R), and then send at twice the source data rate (2R). After the entire block of

information bits B has been buffered at the sender, then the sender can transmit the remaining code bits at 2R. This results in an overall latency of B(n-k)/(2k), or half of the time it takes for the source to generate B bits of data.



Latency Mitigation Strategy Timeline for Rate 1/2 Block Code

Figure 4

4. Session Description Protocol Impacts

The implications for SDP at the time of the writing of this document are not fully known as there is still debate as to the semantics of SDP for WebRTC. A proposal for SDP usage in WebRTC was described in [I-D.nandakumar-rtcweb-sdp]. In addition, the SDP elements required for FEC are described in RFC 6364 [RFC6364]. Leveraging the example of Section 5.1 in [I-D.nandakumar-rtcweb-sdp], a 2-way video and audio session offer/answer exchange can be depicted for two sample endpoints (Alice and Bob) with the video stream being protected by FEC:

Alice->Bob: Offer(Audio:G.711,AMR-WB Video:H.264 FEC-encoding:Reed-Solomon,LDPC Staircase)

Bob->Alice: Answer(Audio:G.711,AMR-WB Video:H.264 FEC-encoding:Reed-Solomon)

Alice->Bob: Two-way AMR-WB Audio, H.264 Video, Reed-Solomon FEC-Encoding

SDP Contents	Notes
v=0	Protocol Version
o=alice 20518 0 IN IP4 0.0.0.0	Session Origin
s=FEC for WebRTC	Session Name
t=0 0	Time session is active
a=ice-ufrag:074c6550	Session Level ICE param
a=ice-pwd:a28a397a4c3f31747d1ee3474af08a068	Session Level ICE param
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	Session Level DTLS Fingerprint for SRTP
a=group:FEC-FR S1 R1	FEC group source/repair
m=audio 54609 RTP/SAVPF 0 109 98	
c= IN IP4 24.23.204.141	Connection Data
a=rtpmap:0 PCMU/8000	G.711 8kbps
a=rtpmap:109 AMR-WB/16000/2	2-chan AMR-WB 16 kbps
a=fmtp:99 interleaving=30	
a=maxptime:100	
a=sendrecv	Can send/rcv audio
a=mid:S0	
a=rtcp-mux	Can mux RTP/RTCP
b=AS:256	Bandwidth
b=RS:0	RTCP Bandwdith
b=RR:0	RTCP Bandwidth
a=candidate:0 1 UDP 2113667327 192.168.1.4 54609 typ host	Host ICE Candidate for audio
a=candidate:1 1 UDP 694302207 24.23.204.141 54609 typ srflx raddr 192.168.1.4 rport 54609	Server Reflexive ICE Candidate for the above host candidate
a=rtcp-fb:109 nack	NACK RTCP feedback
m=video 62537 RTP/SAVPF 99 120	
c= IN IP4 24.23.204.141	Connection data-source
a=rtpmap:99 H264/90000	
a=fmtp:99 profile-level-id=4d0028;packetization-mode=1	
a=fec-source-flow: id=0	Source flow for FEC
a=mid:S1	
a=sendrecv	Can send/rcv video
a=rtcp-mux	Can mux RTP/RTCP
m=application 30000 UDP/FEC	
c= IN IP4 24.23.204.141	Connection data-repair
a=fec-repair-flow: encoding-id=2;	Reed-Solomon code

fssi=E:1400,S:0,m:8	support
a=fec-repair-flow: encoding-id=3; fssi=seed:1234,E:1400,S:0,n1m3:0	LDPC Staircase support
a=repair-window:200ms	Time duration of source and repair blocks
a=mid:R1	
a=candidate:0 1 UDP 2113667327 192.168.1.4 62537 typ host	Host ICE Candidate for video
a=candidate:1 1 UDP 1694302207 24.23.204.141 62537 typ srflx raddr 192.168.1.4 rport 62537	Server Reflexive ICE Candidate for the above host candidate

SDP Offer (Alice to Bob)

Figure 5

SDP Contents	Notes
v=0	Protocol Version
o=bob 16833 0 IN IP4 0.0.0.0	Session Origin
s=FEC for WebRTC	Session Name
t=0 0	Time session is active
a=ice-ufrag:c300d85b	Session Level ICE param
a=ice-pwd:de4e99bd291c325921d5d47efbabd9a2	Session Level ICE param
a=fingerprint:sha-1 99:41:49:83:4a:97:0e:1f:ef:6d:f7:c9:c7:70:9d:1f:66:79:a8:07	Session Level DTLS Fingerprint for SRTP
a=group:FEC-FR S1 R1	FEC group source/repair
m=audio 49203 RTP/SAVPF 109	
c= IN IP4 98.248.92.77	Connection Data
a=rtpmap:109 AMR-WB/16000/2	2-chan AMR-WB 16 kbps
a=fmtp:99 interleaving=30	
a=maxptime:100	
a=sendrecv	Can send/rcv audio
a=mid:S0	
a=rtcp-mux	Can mux RTP/RTCP
b=AS:256	Bandwidth
b=RS:0	RTCP Bandwdith
b=RR:0	RTCP Bandwidth

a=candidate:0 1 UDP 2113667327 192.168.1.7 49203 typ host	Host ICE Candidate for audio
a=candidate:1 1 UDP 694302207 98.248.92.77 49203 typ srflx raddr 192.168.1.7 rport 49203	Server Reflexive ICE Candidate for the above host candidate
a=rtcp-fb:109 nack	NACK RTCP feedback
m=video 63130 RTP/SAVPF 99 c= IN IP4 98.248.92.771	Connection data-source
a=rtpmap:99 H264/90000 a=fmtp:99 profile-level-id=4d0028;packetization-mode=1	
a=fec-source-flow: id=0	Source flow for FEC
a=mid:S1	
a=sendrecv	Can send/rcv video
a=rtcp-mux	Can mux RTP/RTCP
m=application 30000 UDP/FEC c= IN IP4 98.248.92.771	Connection data-repair
a=fec-repair-flow: encoding-id=2; fssi=E:1400,S:0,m:8	Reed-Solomon code support
a=repair-window:200ms	Time duration of source and repair blocks
a=mid:R1	
a=candidate:0 1 UDP 2113667327 192.168.1.7 63130 typ host	Host ICE Candidate for video
a=candidate:1 1 UDP 1694302207 98.248.92.77 63130 typ srflx raddr 192.168.1.7 rport 63130	Server Reflexive ICE Candidate for the above host candidate

SDP Answer (Bob to Alice)

Figure 6

5. Discussion

The use of FEC in WebRTC should not require a significant standards change, as the FEC Framework approved by the IETF already specifies the use of FEC for streaming protocols. There certainly exists tradeoffs between the benefits of FEC at smaller block sizes, and the latency incurred due to larger block sizes. However, these tradeoffs should be considered by WebRTC implementers and not as part of the standardization effort for WebRTC. An item that can be considered is whether a specific FEC scheme should be designated as mandatory-to-

implement, so as to provide a level of interoperability among WebRTC clients.

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

Security considerations are TBD.

8. References

8.1. Normative References

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

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

[RFC6363] Watson, M., Begen, A., and V. Roca, "Forward Error Correction (FEC) Framework", RFC 6363, October 2011.

[RFC6364] Begen, A., "Session Description Protocol Elements for the Forward Error Correction (FEC) Framework", RFC 6364, October 2011.

8.2. Informative References

[I-D.nandakumar-rtcweb-sdp] Nandakumar, S. and C. Jennings, "SDP for the WebRTC", draft-nandakumar-rtcweb-sdp-00 (work in progress), October 2012.

[I-D.narten-iana-considerations-rfc2434bis] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", draft-narten-iana-considerations-rfc2434bis-09 (work in progress), March 2008.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC

Text on Security Considerations", BCP 72, RFC 3552, July 2003.

[Wicker] Wicker, S., "Error Control Systems", Upper Saddle River, NJ: Prentice-Hall Inc., 1995.

Appendix A. Additional Stuff

This becomes an Appendix.

Authors' Addresses

Giridhar Mandyam
Qualcomm Innovation Center
5775 Morehouse Drive
San Diego, California 92121
USA

Phone: +1 858 651 7200
Email: mandyam@quicinc.com

Mike Luby
Qualcomm Technologies Inc.
2030 Addison Street
Berkeley, California 94704
USA

Phone: +1 510 725 3502
Email: luby@qti.qualcomm.com

Thomas Stockhammer
Nomor Research
Brecherspitzstrasse 8
Munich 81541
Germany

Phone: +49 8997898002
Email: stockhammer@nomor.de

Christian Foisy
Qualcomm Technologies Inc.

Phone: +1 450 510 3202
Email: cfoisy@qti.qualcomm.com