A Proposed RTP Payload Format for VP8

Henrik Lundin[1], Patrik Westin[1], Michael Glover[2], Justin Uberti[2], Frank Galligan[2]

[1] Global IP Solutions AB
[2] Google, Inc.

1 Introduction

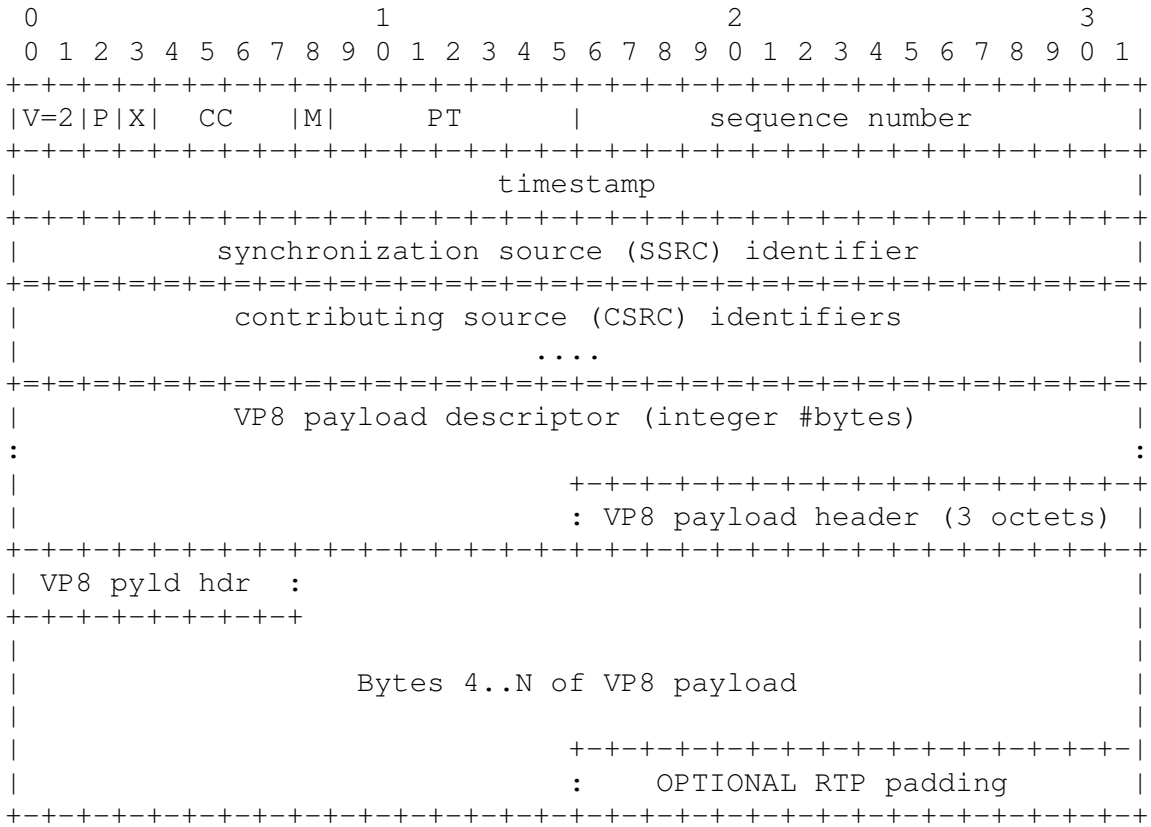
This contribution provides a preview draft of the RTP payload format for the VP8 video codec. The proposal has not yet been submitted to the IETF.

An encoded VP8 frame can be divided into two or more partitions, as described in [1]. The first partition (*prediction* or *mode*) contains prediction mode parameters and motion vectors for all macroblocks. The remaining partitions all contain the DCT/WHT coefficients for the residuals. The first partition is decodable without the remaining residual partitions. The subsequent partitions may be useful even if some part of the frame is lost.

The outline of this paper is similar to that of an RFC draft. We will first go through the format specification in Section 2. After that we discuss two ways of using the VP8 format together with existing RFCs to provide improved loss robustness. In Section 3, we illustrate how VP8 can be combined with uneven level FEC protection. In Section 4, a method to acknowledge receipt of reference frames using RTCP techniques is described. Both these examples serve as motivation for two of the fields included in the payload format: the "1st partition size" and "PictureID" fields.

2 Payload Format

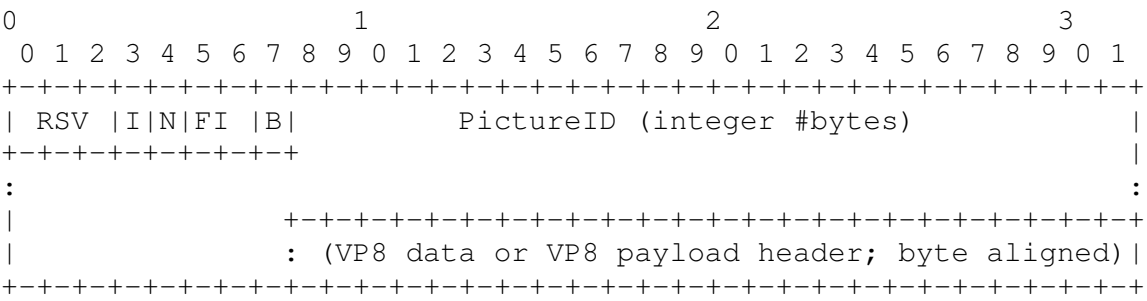
The general RTP payload format for VP8 is depicted below.



The VP8 payload descriptor and VP8 payload header will be described in the sequel.

2.1 VP8 Payload Descriptor

The first bytes after the RTP header are the VP8 payload descriptor, with the following structure.



RSV: 3 bits
Bits reserved for future use. MUST be equal to zero and MUST be ignored by the receiver.

I: 1 bit
PictureID present. When set to one, a PictureID is provided after the first byte of the payload descriptor. When set to zero, the PictureID is omitted, and the one-byte payload descriptor is immediately followed by the VP8 payload.

N: 1 bit
Non-reference frame. When set to one, the frame can be discarded without affecting any other future or past frames.

FI: 2 bits
Fragmentation information field. This field contains information about the fragmentation of VP8 payloads carried in the RTP packet. The four different values are listed below.

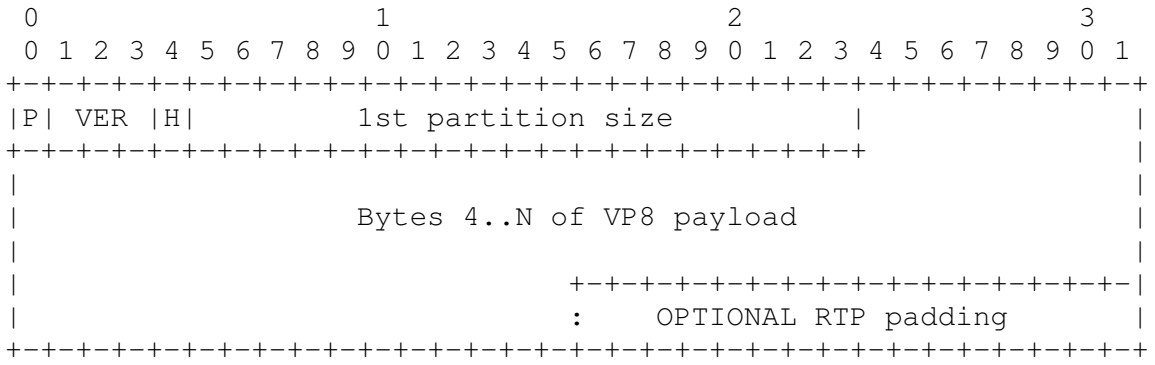
FI	Fragmentation status
00	The RTP packet contains no fragmented VP8 partitions. The payload is one or several complete partitions.
01	The RTP packet contains the first part of a fragmented partition. The fragment must be placed in its own RTP packet.
10	The RTP packet contains a fragment that is neither the first nor the last part of a fragmented partition. The fragment must be placed in its own RTP packet.
11	The RTP packet contains the last part of a fragmented partition. The fragment must be placed in its own RTP packet.

B: 1 bit
Beginning VP8 frame. When set to 1 this signals that a new VP8 frame starts in this RTP packet.

PictureID: Multiple of 8 bits
This is a running index of the frames. The field is present only if the I bit is equal to one. The most significant bit of each byte is an extension flag. The 7 following bits carry (parts of) the PictureID. If the extension flag is one, the PictureID continues in the next byte. If the extension flag is zero, the 7 remaining bits are the last (and least significant) bits in the PictureID. The sender may choose any number of bytes for the PictureID. The PictureID SHALL start on a random number, and SHALL wrap after reaching the maximum ID as chosen by the application.

2.2 VP8 Payload Header

The first three bytes of an encoded VP8 frame are uncompressed, and co-serve as payload header in this RTP format. Note that the header is present only in packets which have the B bit equal to one in the payload descriptor. Subsequent packets for the same frame do not carry the payload header.



P: 1 bit
Inverse key frame flag. When set to 0 the current frame is a key frame. When set to 1 the current frame is an interframe. Defined in [1].

VER: 3 bits
A version number as defined in [1].

H: 1 bit
Show frame bit as defined in [1].

1st partition size: 19 bits
A field containing the size of the first data partition in bytes, as defined in [1].

2.3 Aggregated and Fragmented Payloads

An encoded VP8 frame can be divided into two or more partitions, as described in Section 1. The fragmentation information described in Section 2.1 MUST be used to signal if any fragmentation is applied.

Aggregation of encoded partitions is done without explicit signaling. Partitions MUST be aggregated in decoding order. An aggregation MUST have exactly one payload descriptor. Aggregated partitions MUST represent parts of one and the same video frame. Consequently, an aggregated packet will have one or no payload header, depending on whether the aggregate contains the first partition of a frame or not, respectively. Note that the length of the first partition can always be obtained from the first partition size parameter in the VP8 payload header. Fragments of encoded partitions MUST NOT be aggregated.

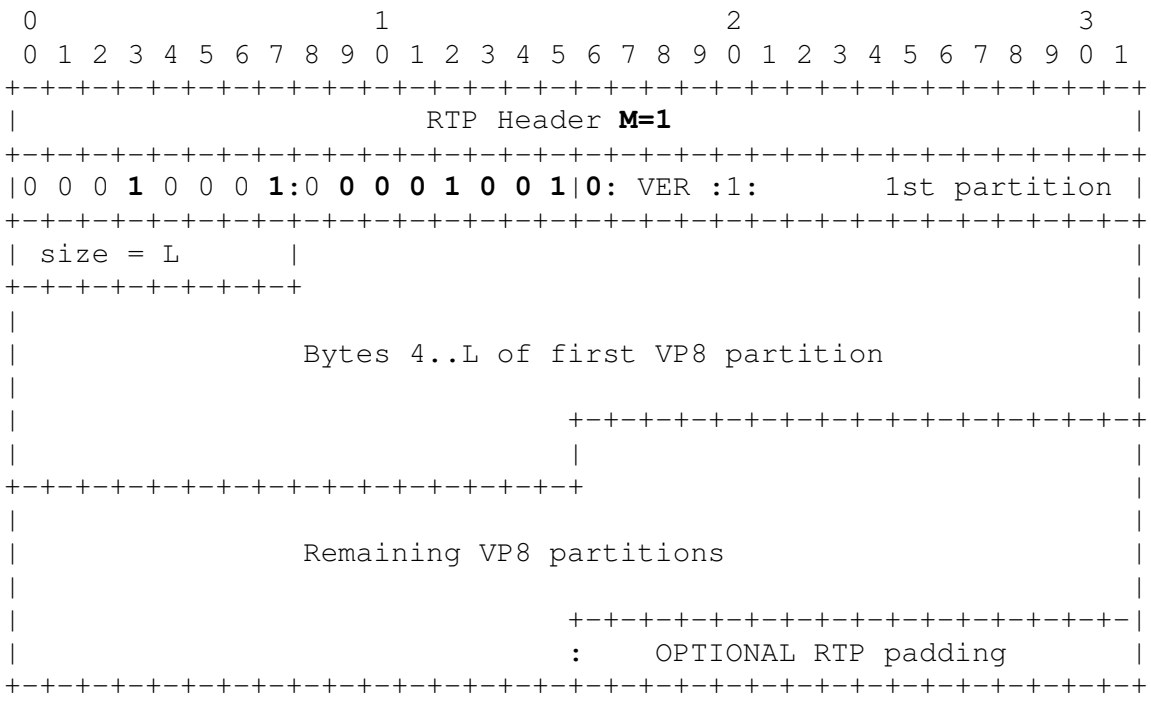
See Section 2.4 for examples.

2.4 Examples of VP8 RTP Steam

A few examples of how the VP8 RTP payload can be used are included below. In each example, the distinguishing features are marked with bold typeface.

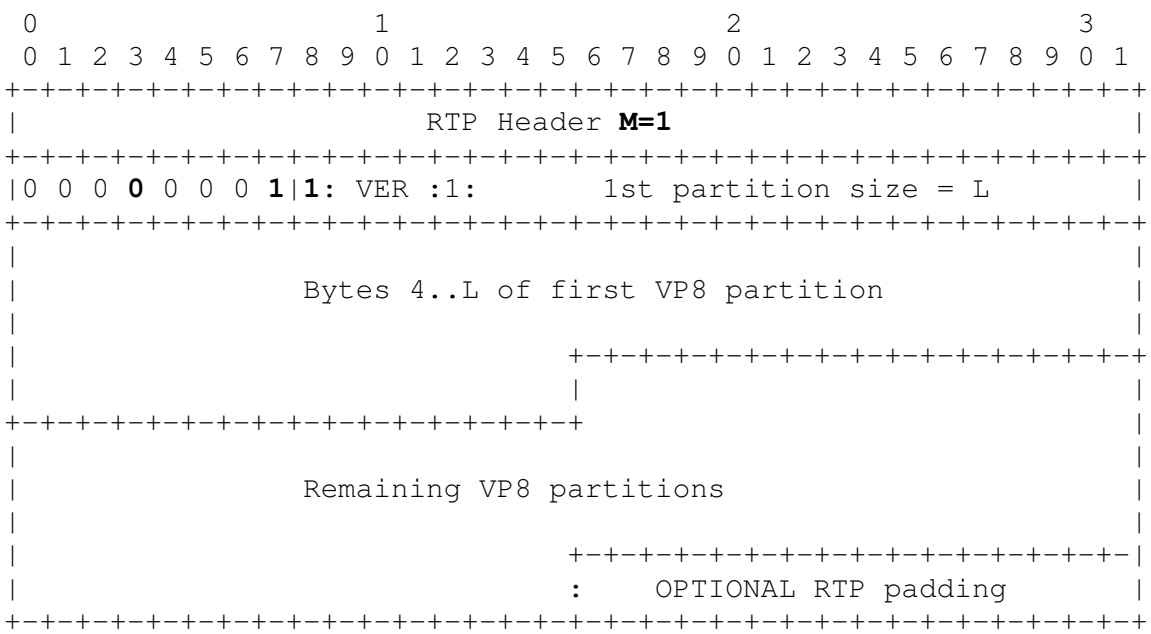
2.4.1 VP8 key frame in a single RTP packet

Marker bit = 1. I = 1. B = 1. PictureID = 17 = 0001001 binary. P = 0.



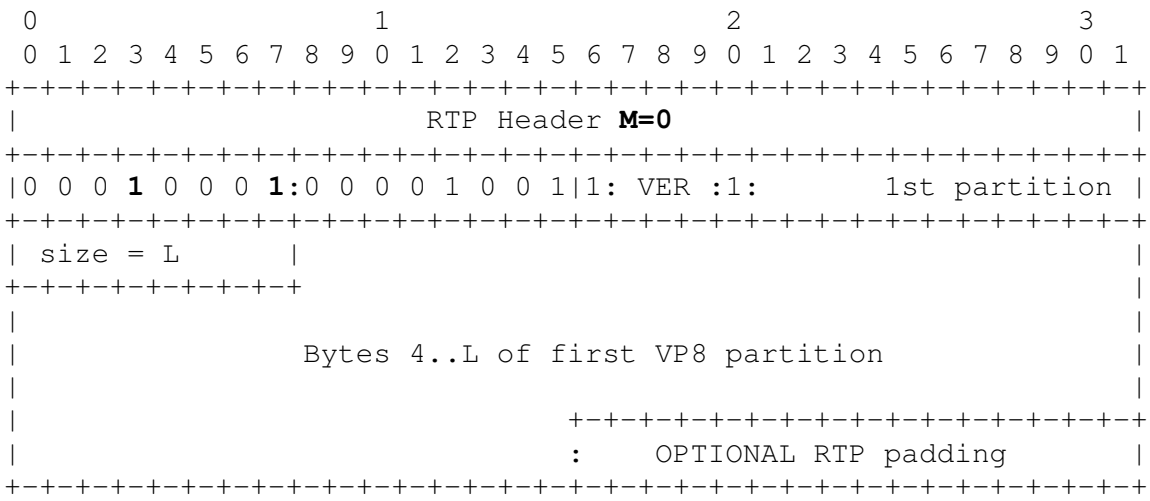
2.4.2 VP8 interframe in a single RTP packet; no PictureID

Marker bit = 1. I = 0. B = 1. P = 1.

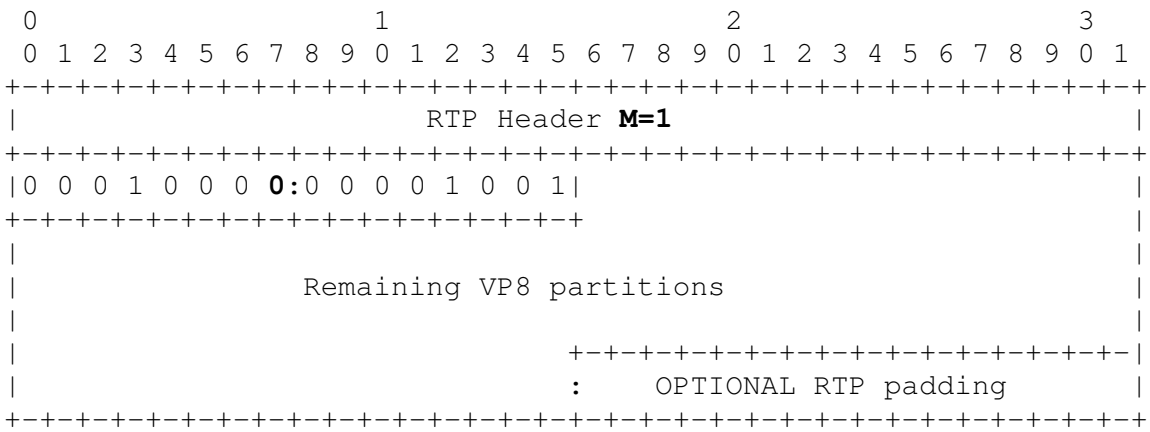


2.4.3 VP8 partitions in separate RTP packets

First RTP packet; marker bit = 0. I = 1. B = 1. PictureID = 17.

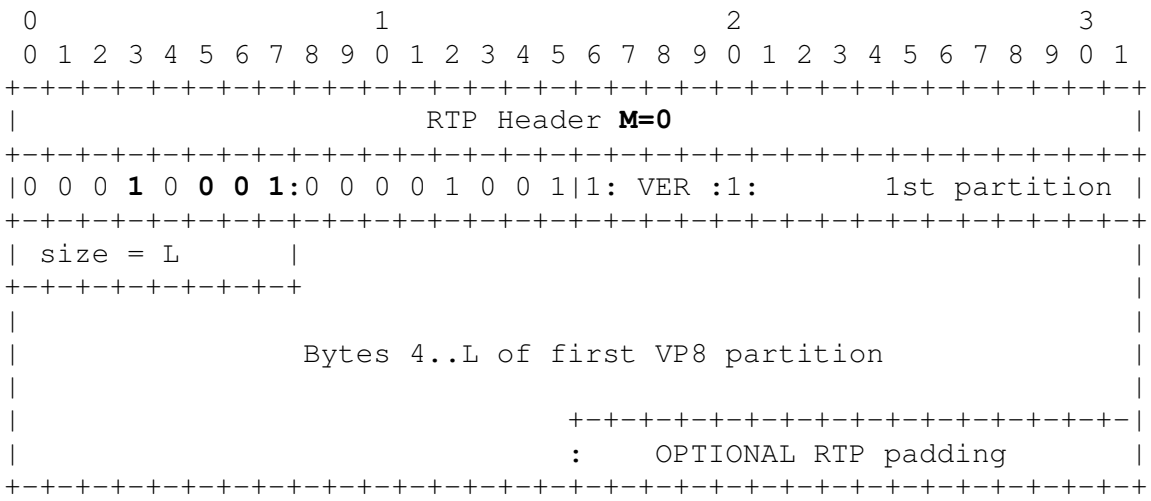


Second RTP packet; marker bit = 1. B = 0.

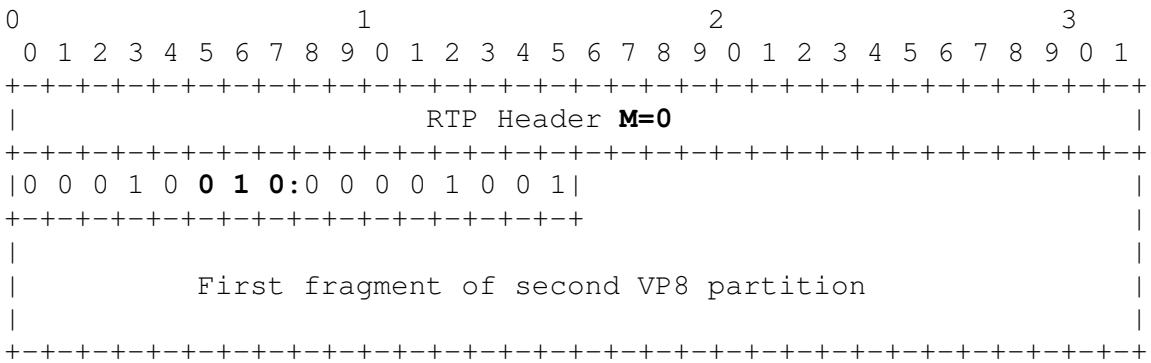


2.4.4 VP8 frame fragmented across RTP packets

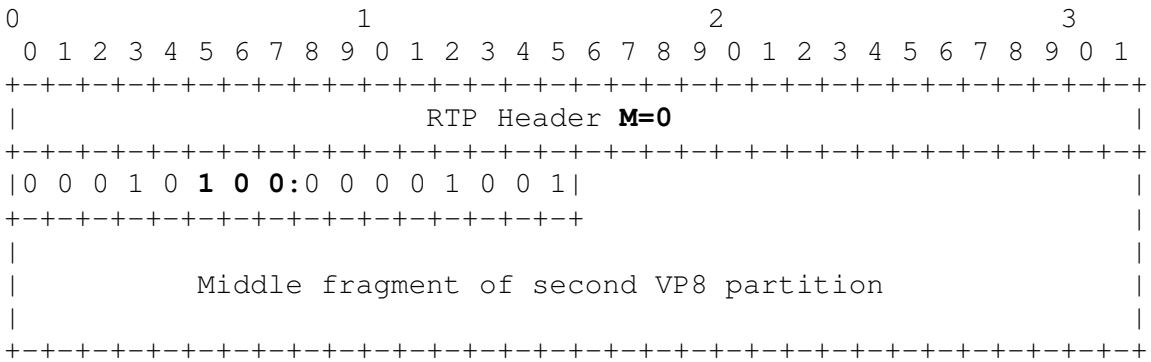
First RTP packet; marker bit = 0. I = 1. FI = 00. B = 1.



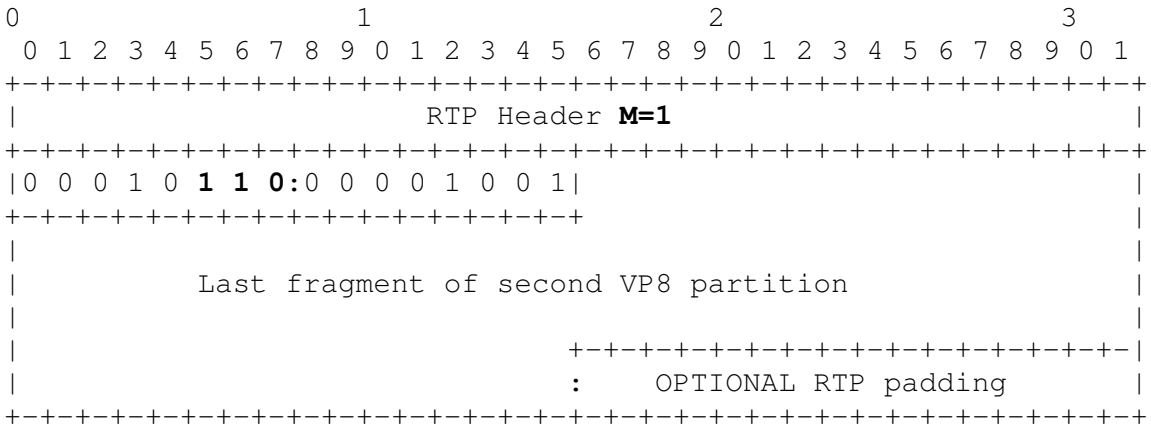
Second RTP packet; marker bit = 0. FI = 01. B = 0.



Third RTP packet; marker bit = 0. FI = 10. B = 0.

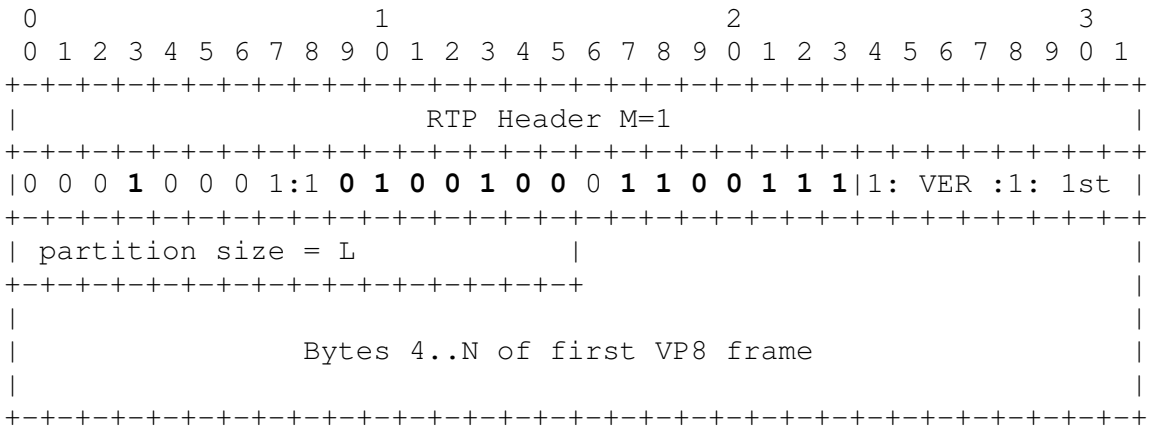


Last RTP packet; marker bit = 1. FI = 11. B = 0.



2.4.5 VP8 frame with long PictureID

PictureID = 4711 = 01001001100111 binary (first 7 bits: 0100100, last 7 bits: 1100111).



3 Using VP8 with Uneven Level FEC Protection

RFC 5109 [2] specifies a payload format for generic forward error correction (FEC) for RTP packets. (An errata with critical changes was also published.) One salient feature of RFC 5109 is that it provides "uneven level protection", ULP, which enables FEC protection of *parts* of an RTP packet. Specifically, the first part of an RTP packet can be given a stronger protection than the remaining part. The special case is where only the first part of the RTP packet is protected. The length of the protected part (actually the length of each protection level) is chosen and changed dynamically during a session.

The concept of ULP FEC fits well with the VP8 video format. The first partition of an encoded VP8 frame consists of context variables and prediction parameters (mode and vectors), while the subsequent partitions contain encoded residual information. For a decoder, the residual information is not useful without the first partition. However, a decoder could successfully use the information in the first partition to provide good packet loss concealment, even if the subsequent partitions are lost. The conclusion is that the first partition deserves a higher protection factor than the remaining data.

By including the "first partition size" parameter in the VP8 payload header (Section 2.2), the application, or even a media aware network element, can apply the ULP FEC to the VP8 payloads, since it can readily identify and obtain the length of the first partition to which (a stronger) protection should be granted.

RFC 5109 suggests two methods for multiplexing the media data and the FEC data: using the RED payload type (RFC 2198, [3]) and using separate sessions. The example below is based on the RED method, although the above RTP format for VP8 does not preclude any method.

Since we are only interested in protecting the first partition of the VP8 payload, it is in some cases possible to obtain increased robustness for that part even without FEC signaling. If the first partition is isolated in a separate RTP packet, this packet can simply be sent twice (exact replica of RTP header and payload).

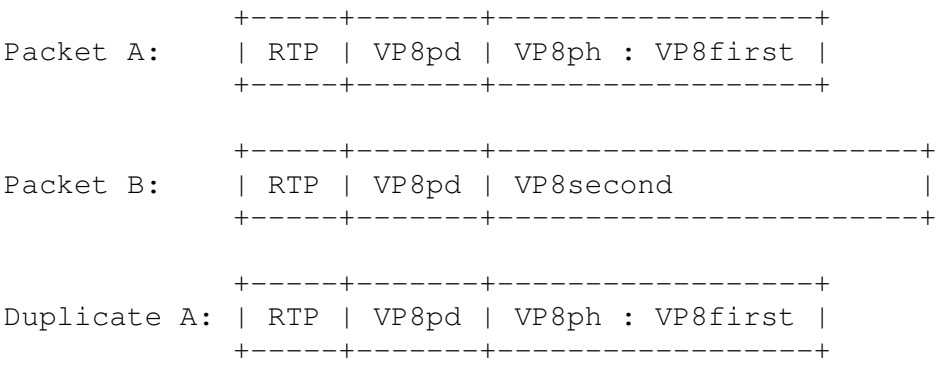
3.1 Examples

The following terminology is used in the examples below:

- RTP: RTP header, 12 octets. (Special rules apply if the header is extended; see [2].)
- RED: Redundancy payload header, 1 or 4 octets.
- VP8pd: VP8 payload descriptor, 1 or more octets (see Section 2.1).
- VP8ph: VP8 payload descriptor, 3 octets (see Section 2.2).
- VP8first: First partition of a VP8 frame, length given in VP8ph.
- VP8second: Second (or later) partition of a VP8 frame.
- FEC: FEC header for FEC packets, 10 octets (see [2]).
- ULP: FEC level header for FEC packets, 2 or 4 octets (see [2]).

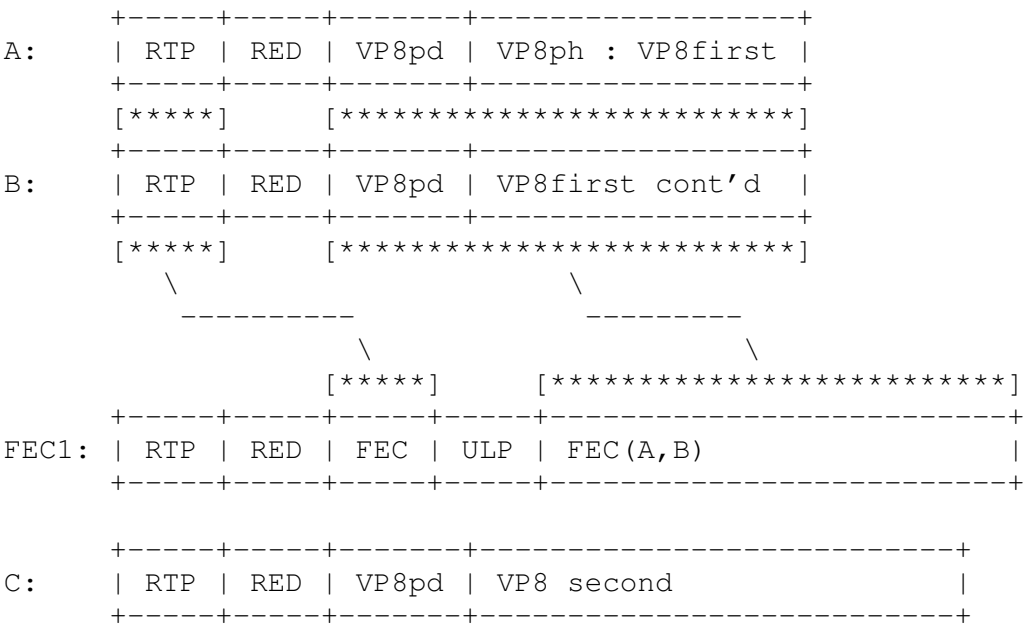
3.1.1 VP8 first partition isolated

The first partition of the VP8 frame is sent in its own RTP packet, which can easily be duplicated for increased robustness. This case provides a 50% protection factor (ratio of number of FEC packet to the sum of FEC and protected packets; sending two duplicates provides a 67% protection, and so on).



3.1.2 VP8 first partition split

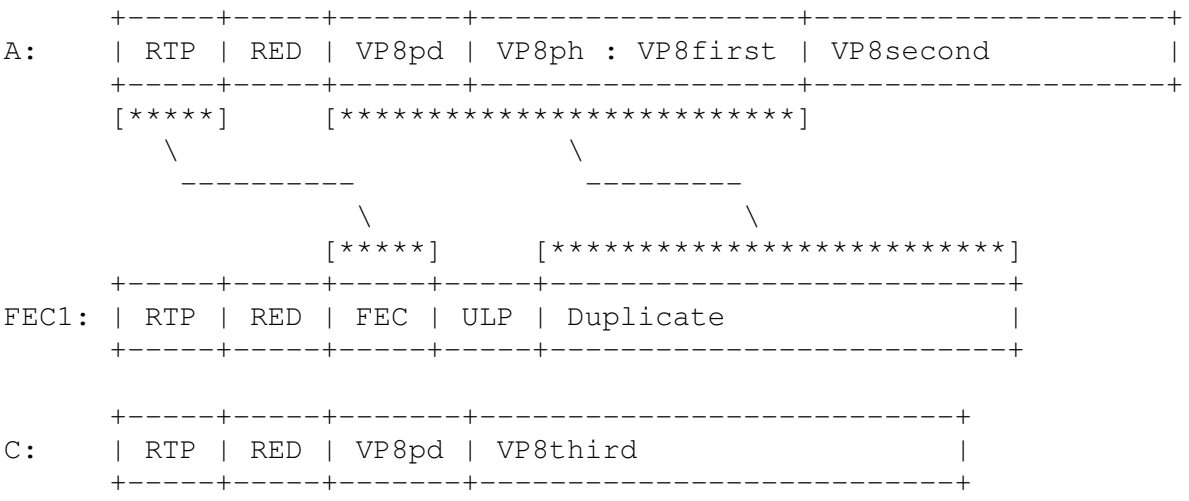
The first partition of the VP8 frame can be split across two (or more) RTP packets. It must be done if the first partition is larger than the MTU, but can also be done to facilitate protection factors lower than 50%. The FEC header is calculated from the RTP headers of packets A and B, while the FEC payload FEC(A,B) is constructed from the VP8 payload descriptor, payload header and first partition, as indicated in the figure below.



The split of the first partition is preferably done such that the payloads used to calculate FEC(A,B) are close to equal size.

3.1.3 VP8 partitions aggregated

In the case when the first partition is sent in the same packet as one or more subsequent partitions, the level protection can be applied to facilitate a bit-conservative protection for only the first partition.



4 Using VP8 with RPSI and SLI Feedback

The VP8 payload descriptor defined in Section 2.1 above contains an optional PictureID parameter. This parameter is included mainly to enable use of reference picture selection index (RPSI) and slice loss indication (SLI), both defined in RFC 4585 [4].

4.1 RPSI

The reference picture selection index is a payload-specific feedback message defined within the RTCP-based feedback format. The RPSI message is generated by a receiver and can be used in two ways. Either it can signal a preferred reference picture when a loss has been detected by the decoder – preferably then a reference that the decoder knows is perfect – or, it can be used as positive feedback

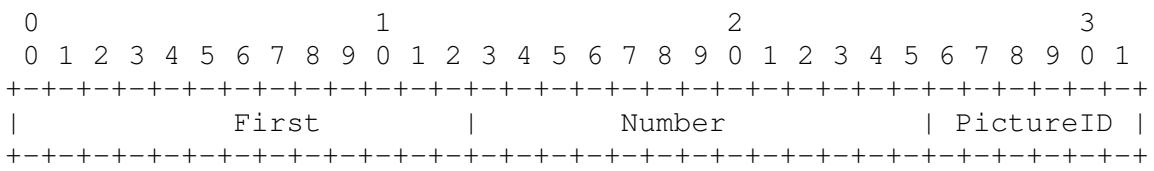
information to acknowledge correct decoding of certain reference pictures. We think that the positive feedback method is better suited for VP8.

The use of RPSI for VP8 is preferably combined with a special update pattern of the codec's two special reference frames – the *golden frame* and the *altref frame* – in which they are updated in an alternating leapfrog fashion.

When a receiver has received and correctly decoded a golden or altref frame, and that frame had a PictureID in the payload descriptor, the receiver can acknowledge this simply by sending an RPSI message back to the sender. The message body (i.e., the "native RPSI bit string" in RFC 4585) is simply the PictureID of the received frame.

4.2 SLI

The slice loss indication is another payload-specific feedback message defined within the RTCP-based feedback format. The SLI message is generated by the receiver when a loss or corruption is detected in a frame. The format of the SLI message is as follows [4]:



Here, **First** is the macroblock address (in scan order) of the first lost block and **Number** is the number of lost blocks. **PictureID** is the six least significant bits of the codec-specific picture identifier in which the loss or corruption has occurred. For VP8, this codec-specific identifier is naturally the PictureID of the current frame, as read from the payload descriptor. If the payload descriptor of the current frame does not have a PictureID, the receiver MAY send the last received PictureID+1 in the SLI message. The receiver MAY set the First parameter to 0, and the Number parameter to the total number of macroblocks per frame, even though only parts of the frame is corrupted.

When the sender receives an SLI message, it can make use of the knowledge from the latest received RPSI message. Knowing that the last golden or altref frame was successfully received, it can encode the next frame with reference to that established reference.

4.3 Example

The use of RSPI and SLI is best illustrated in an example. In this example, the encoder may not update the altref frame until the last sent golden frame has been acknowledged with an RPSI message. If an update is not received within some time, a new golden frame update is sent instead. Once the new golden frame is established and acknowledge, the same rule applies when updating the altref frame.

Event	**Sender**	**Receiver**	**Established reference**
1000	Send golden frame PictureID = 0	Receive and decode golden frame	
1001	Receive RPSI(0)	Send RPSI(0)	golden
…	(sending regular frames)		
1100	Send altref frame PictureID = 100	Altref corrupted or lost	golden
1101	Receive SLI(100)	Send SLI(100)	golden

Event	Sender	Receiver	Established reference
1102	Send frame with reference to golden	Receive and decode frame (decoder state restored)	golden
…	(sending regular frames)		
1200	Send altref frame PictureID = 200	Receive and decode altref frame	golden
1201	Receive RPSI(200)	Send RPSI(200)	altref
…	(sending regular frames)		
1300	Send golden frame PictureID = 300	Receive and decode golden frame	altref
1301	RPSI lost	Send RPSI(300)	altref
1400	Send golden frame PictureID = 400	Receive and decode golden frame	altref
1401	Receive RPSI(400)	Send RPSI(400)	golden

Note that the scheme is robust to loss of the feedback messages. If the RPSI is lost, the sender will try to update the golden (or altref) again after a while, without releasing the established reference. Also, if an SLI is lost, the receiver can keep sending SLI messages at any interval, as long as the picture is corrupted.

5 References

[1] WebM Project, “VP8 Data Format and Decoding Guide”, Google, Inc., July 2010.

[2] A. Li (ed.), “RTP Payload Format for Generic Forward Error Correction”, RFC 5109, December 2007.

[3] C. Perkins, I. Kouvelas, O. Hodson, V. Hardman, M. Handley, J.C. Bolot, A. Vega-Garcia, S. Fosse-Parisis, ”RTP Payload for Redundant Audio Data”, RFC 2198, September 1997.

[4] J. Ott, S. Wenger, N. Sato, C. Burmeister, J. Rey, “Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)”, RFC 4585, July 2006.