avtcore WG
Internet-Draft
Intended status: Informational
Expires: 13 April 2026

E. Sprꬽng
Google

10 October 2025

RTCP Feedback Message and Request Mechanism for Frame-level Acknowledgement
draft-sprang-avtcore-frame-acknowledgement-01

Abstract

This document describes a mechanism for signaling which video frames have been received and decoded by a remote peer. It comprises an RTCP feedback message and an RTP header extension used to request said feedback.

One of the main use cases for this data is to implement various forms of Long Term Reference (LTR) reference structures.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://github.com/sprangerik/frame-acknowledgement/blob/main/draft-sprang-avtcore-frame-acknowledgement.md. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-sprang-avtcore-frame-acknowledgement/.

Discussion of this document takes place on the avtcore WG Working Group mailing list (mailto:avt@ietf.org), which is archived at https://datatracker.ietf.org/wg/avtcore. Subscribe at https://www.ietf.org/mailman/listinfo/avt/.

Source for this draft and an issue tracker can be found at https://github.com/sprangerik/frame-acknowledgement.

Status of This Memo

This Internet-Draft will expire on 13 April 2026.

Copyright Notice

Table of Contents

1. Introduction

The most common way for realtime video to be transmitted is to encode a pretty much fixed scalability structure, such as those in the W3C [SVC] Scalability mode list.

In such a scenario, the video encoder produces frames "blindly" without real knowledge of what state the remote receiver is in. Using recovery mechanisms such as retransmission, forward error correction and fast-forwarding past skippable frames the receiver is assumed to be able to decode the video. In some cases those methods may not be enough, requiring keyframe requests to be sent as a last resort.

On the other hand, if the encoder is able to reason about which frames have been received and decoded it can be more proactive. One way is to store frames that are known to be received so that they can be later used as guaranteed good references in the case of e.g. large loss events, avoiding the need for potentially large retransmissions etc. Collectively this is often referred to as "Long Term Reference" structures or LTR for short, although the exact structure may vary.

In order to achieve this the sender must be able to reason about the state of the receiver, necessitating the need for feedback signals. In this document a new RTCP message called "Frame Acknowledgement" is introduced as a codec agnostic feedback message for this purpose. Further, an RTP header extension is introduced that allows the sender to actively request feedback on decoding of the associated frame. This allows the sender to both request quick feedback on frames that are important for latency, and enables resilience against loss of feedback packets.

Note that it is allowed to report a frame as decoded even if the decode process is not complete - as long as the receiver guarantees that it will attempt to decode the frame. The rationale for this is that we want to reduce the feedback delay as much as possible. Should the decoding of a frame that has been acknowledged fail, then the receiver MUST request a keyframe to recover, even if the failed decoding belongs to a droppable layer.

2. Conventions and Definitions

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.

For the purposes of this document, a "frame" is defined as any decodable unit of bitstream data that results in the update to the codec state (e.g. reference buffers, entropy tables, etc) that can be used as a reference for any subsequent decodable unit of bitstream data. Typically that will be a full frame, but also include cases such as a "no show" frame intended for later reference or even a part of a frame such as a tile or a slice if it is independently decodable and makes updates to encoder state that other tiles/frames can later reference.

3. Applicability

Frame Acknowledgement can be used for video streams in most topologies. It is also designed to be codec agnostic.

In terms of [RFC7667], Point-to-Point is the most straightforward target as it is easiest to reason about a single receiver. A Media Translator or other systems that include a decoder are similarly easy - from the perspective of the sender the middle box is the receiver.

If a Transport Translator is used for Point-to-Multi-Point, then the middlebox must make sure to make valid translations. See section on Frame ID considerations (Section 8) below.

4. Existing Feedback Formats

This section provides an overview, for informational purposes, of some existing feedback formats that could be seen as alternatives.

NACK, defined in [RFC4585], provides only requests for packets the receiver is interested in having retransmitted. Absence of feedback is a poor signal for acknowledgement, especially since said feedback can be lost.

[RFC8888] and [TWCC] provide per-packet acknowledgement and so are more useful. A mapping from packet(s) to frame needs to happen but that is not a big problem. However, even if a frame is confirmed to be received there is no guarantee that it gets decoded.

Reference Picture Selection Indication (RPSI) is another existing message, but it puts the logic of requesting a particular reference frame in the receiver - significantly complicating the system especially in Point-to-Multi-Point systems. It is further codec specific, and several modern codecs lack a specification - including AV1 and H.266.

Loss Notification [LNTF] was a proposed RTCP message intended to solve most of these problems, but it lacks resilience against loss of feedback and also cannot handle out-of-order acknowledgements. The latter makes for instance single-SSRC simulcast structures (e.g. SxTx modes in [SVC]) impossible.

5. Requirements

The messages in this proposal are intended to fulfill the following requirements:

1. Codec agnostic The protocol should be general enough to work across all current and future codecs.

2. Payload Invariant The protocol should not depend on data within the encoded bitstream payload. That includes codec specific frame identifiers, feedback requests and feedback messages.

3. Uses Frame Identifiers Explicit marking of frames, rather than using an indirection via packets.

4. Order Invariant The format should not make assumptions about the required decode order of frames.

5. Send-side Controlled The sender explicitly indicates when and for which frames feedback should be sent.

6. Loss Resilient The sender should be able to detect and recover from lost feedback messages.

7. Low Delay The latency should be small, with the sender being able to tune delay vs rate tradeoff.

8. Low Overhead The network overhead in terms of both packet rate and bitrate should be minimized.

6. Frame Acknowledgment Extension

The Frame Acknowledgement extension is an RTP header extension used both to identify frames and request feedback about the remote state. It SHOULD appear on the last packet of a video frame, and MUST NOT appear more than once on a single frame.

6.1. Frame Identifier

In order to request and receive information about decoded frames, we must be able to identify them. The frame acknowledgement header extension may contain a Frame ID field for this purpose. The Frame ID is an 8-bit unsigned integer field, that wraps around to 0 on overflow.

6.2. Frame Acknowledgment Request

In order to get feedback on the state of the remote decoder, the sender actively requests such feedback using the same frame acknowledgement header extension that is also used for frame identification. The feedback request comprises a Start Frame ID and Length field. Specifying the range explicitly has several advantages, including enabling reliable delivery of the feedback since the sender can effectively make retransmission requests of the feedback.

If a new Frame Acknowledgement Request is sent with an incremented Feedback Start, all status values prior to that Frame ID are considered as acknowledged and can be culled by the receiver. A sender MUST NOT request feedback prior to either the last acknowledged Frame ID or the start of the stream.

6.3. Frame Acknowledgment Request Data Layout

This section describes the data layout for the Frame Acknowledgment RTP Header Extension. The extension data starts with the FFR/ Reserved byte.

For a One-Byte Header (as defined in [RFC5285], Section 4.2), the ID field identifies the extension, and the len field is a 4-bit value N that indicates the number of data bytes following the ID and len fields, _minus one_. Thus, the total number of bytes for the extension data is N+1.

For a Two-Byte Header (as defined in [RFC5285], Section 4.3), the ID field identifies the extension, and the 8-bit length field indicates the total number of bytes for the extension data (i.e., the FFR/ Reserved byte plus any optional fields).

ascii-art Extension Data: 0 0 1 2 3 4 5 6 7 +-+-+-+-+-+-+-+-+ |FFR| Reserved | (First byte of extension data) +-+-+-+-+-+-+-+-+ | Frame ID | (OPTIONAL, see FFR) +-+-+-+-+-+-+-+-+ | Feedb. Start | (OPTIONAL, see FFR) +-+-+-+-+-+-+-+-+ | Feedb. Length | (OPTIONAL, see FFR) +-+-+-+-+-+-+-+-+

6.3.1. FFR: FrameID / Feedback Request (2 bits)

This field is located in the first byte of the extension data. It indicates the presence and meaning of the subsequent optional fields. The total number of bytes for the extension data (and thus the value of N for the one-byte header’s len field, or the length field for two-byte headers) depends on the FFR value:

* *00: Frame ID only.* The FFR/Reserved byte is followed by a one-byte Frame ID field. Total extension data bytes = 1 (FFR/Reserved + Frame ID). For one-byte header: len = 0. For two-byte header: length = 1. No feedback is explicitly requested by this header.

* *01: Frame ID + implicit feedback request.* The FFR/Reserved byte is followed by a one-byte Frame ID field. Total extension data bytes = 1 (FFR/Reserved + Frame ID). For one-byte header: len = 0. For two-byte header: length = 1. Feedback is requested for the frame identified by Frame ID (i.e., Feedback Start = Frame ID, Feedback Length = 1).

* *10: Frame ID + independent feedback request.* The FFR/Reserved byte is followed by three bytes: a one-byte Frame ID, a one-byte Feedback Start, and a one-byte Feedback Length field. Total extension data bytes = 3 (FFR/Reserved + Frame ID + Feedback Start + Feedback Length). For one-byte header: len = 2. For two-byte header: length = 3. This implies both a frame marking with Frame ID and an independent feedback request for the specified range.

* *11: Reserved for future use.*

The remaining 6 bits of the FFR/Reserved byte are reserved and SHOULD be set to 0.

6.3.2. Frame ID (8 bits)

Present if FFR is 00, 01, or 10. An unsigned integer that uniquely identifies a frame. It MUST be incremented by one for each new frame (in sending order) that needs to be identified. It wraps around to 0 on overflow.

6.3.3. Feedback Start (8 bits)

Present if FFR is 10 or 11. An unsigned integer that corresponds to the first Frame ID (inclusive) the sender is requesting feedback for. It wraps around to 0 on overflow.

6.3.4. Feedback Length (8 bits)

Present if FFR is 10 or 11. An unsigned integer that indicates the number of consecutive frames the sender is requesting feedback for, starting from Feedback Start. A value of 0 means no frames are being requested. A value of 1 means only the frame identified by Feedback Start is requested. The range is Feedback Start to Feedback Start + Feedback Length - 1, inclusive, with wrap-around logic applied to Frame IDs.

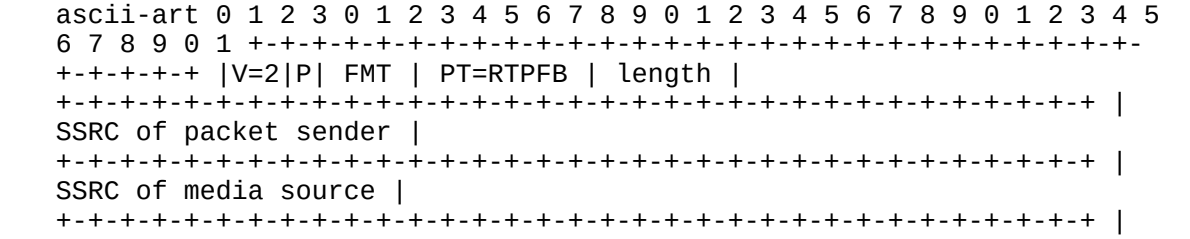
Note that since the Frame ID, Feedback Start, and Feedback Length are 8-bit fields that wrap, care must be taken when calculating ranges. For example, a request with Feedback Start = 254 and Feedback Length = 3 indicates the sender is requesting feedback for frames with Frame IDs 254, 255, and 0.

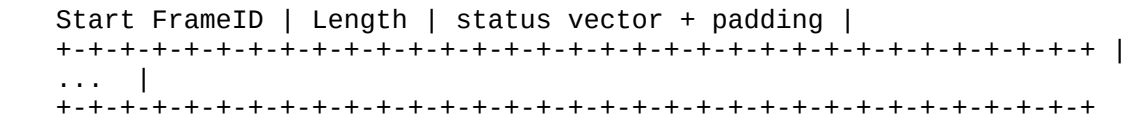
If a sender is not interested in feedback for frames prior to and including a given Frame ID, it can effectively signal this by sending a request (FFR=01, 10, or 10) where the Feedback Start (or Frame ID for FFR=01) is more recent. This implicitly acknowledges prior frames up to the new Feedback Start. Alternatively, a Feedback Length of 0 can be used with FFR=10 if no specific frames need feedback but an acknowledgment point needs to be set.

7. Frame Acknowledgment Feedback RTCP Message

The Frame Acknowledgement Feedback message is an RTCP message ([RFC4585]) containing a vector of status symbols, corresponding to the state for the frames requested in a Frame Acknowledgement Extension.

This message is identified by PT = RTPFB (205) and FMT = TBD (to be assigned by IANA, suggested value 12).





7.1. Start Frame ID (8 bits)

The first Frame ID (inclusive) for which feedback is provided in this message. This corresponds to a Frame ID previously sent in a Frame Acknowledgment Request extension.

7.2. Length (8 bits)

An unsigned integer denoting how many consecutive frames, starting from Start Frame ID, this message contains feedback for. The last Frame ID included in the feedback is (Start Frame ID + Length - 1), with wrap-around logic applied to Frame IDs. A Length of 0 indicates no feedback information is present, though this SHOULD NOT be sent.

7.3. status vector (variable length)

A bit vector of the size indicated by the Length field. Each bit corresponds to a Frame ID, starting from Start Frame ID and incrementing by one for each subsequent bit. * A value of *0* indicates the frame has not been received or has not been decoded (or is not expected to be decoded). * A value of *1* indicates the frame has been received and has been or will be decoded.

The status vector MUST be padded with 0 to 23 zero bits to align to the next 32-bit boundary if its length is not a multiple of 32 bits. This padding is not included in the Length field but is included in the RTCP packet’s length field.

8. Frame ID considerations

As stated above, the sender MUST increment the Frame ID by one for each new frame with the Frame Acknowledgement header extension present, in sending order. More than that, it must make sure that no wrap-around ambiguity can occur. Since feedback is only really necessary for frames which the codec stores in a reference buffer pending future use, the number of outstanding frames is in practice limited by the number of available reference buffers. E.g. for AV1, the upper limit will be 8. Although the optimal behavior will be application dependent, it is often advisable to spread reference buffer usage out across an RTT and to cull earlier buffer usage once later frames have been acknowledged.

8.1. Point-to-Multi-Point

When considering a multi-way application with an SFU/SFM-type relay in the middle, the middlebox may need to do translations/rewriting of Frame IDs such that the outgoing FrameIDs from a middlebox to a receiver still fulfill the requirement that the FrameIDs are incremented by one for each new frame that is marked for feedback. This must be true even if independent video streams for different senders are multiplexed onto the same SSRC. Further the middlebox should typically not acknowledge a frame to a sender unless all active receivers have acknowledged that frame.

9. Security Considerations

The messages in this proposal may expose a small amount of data, namely the number of frames that have been sent, and potentially in an indirect way which frames the sender sees as important for recovery.

This data should however not pose any significant privacy or security risks.

10. IANA Considerations

The RTP header extension needs to have a URI identifier assigned by IANA. See [IANAEXT].

The RTCP message uses PT = 205 (RTPFB, Generic RTP Feedback). As of writing, the next available FMT value is 12. A dedicated ID needs to be assigned by IANA. See [IANARTCP].

11. References

11.1. Normative References

[DD] AOM, "Dependency Descriptor RTP Header Extension", n.d., <https://aomediacodec.github.io/av1-rtp-spec/#dependency-descriptor-rtp-header-extension>.

[IANAEXT] IANA, "RTP Compact Header Extensions", n.d., <https://www.iana.org/assignments/rtp-parameters/rtp-parameters.xhtml#rtp-parameters-10>.

[IANARTCP] IANA, "FMT Values for RTPFB Payload Types", n.d., <https://www.iana.org/assignments/rtp-parameters/rtp-parameters.xhtml#rtp-parameters-4>.

[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/rfc/rfc2119>.

[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/rfc/rfc8174>.

11.2. Informative References

[LNTF] "RTCP feedback Message for Loss Notification", n.d., <https://www.ietf.org/archive/id/draft-majali-avtcore-lntf-feedback-message-00.html>.

[RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, DOI 10.17487/RFC4585, July 2006, <https://www.rfc-editor.org/rfc/rfc4585>.

[RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, DOI 10.17487/RFC5285, July 2008, <https://www.rfc-editor.org/rfc/rfc5285>.

[RFC7667] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 7667, DOI 10.17487/RFC7667, November 2015, <https://www.rfc-editor.org/rfc/rfc7667>.

[RFC8888] Sarker, Z., Perkins, C., Singh, V., and M. Ramalho, "RTP Control Protocol (RTCP) Feedback for Congestion Control", RFC 8888, DOI 10.17487/RFC8888, January 2021, <https://www.rfc-editor.org/rfc/rfc8888>.

[SVC] W3C, "Scalable Video Coding (SVC) Extension for WebRTC", n.d., <https://www.w3.org/TR/webrtc-svc>.

[TWCC] "RTP Extensions for Transport-wide Congestion Control", n.d., <https://datatracker.ietf.org/doc/html/draft-holmer-rmcat-transport-wide-cc-extensions-01>.

[VFTI] "Video Frame Tracking Id", n.d., <http://www.webrtc.org/experiments/rtp-hdrext/video-frame-tracking-id>.

Acknowledgments

Author’s Address

Erik Språng
Google
Email: sprang@google.com