Network Working Group
Internet-Draft
Intended status: Experimental
Expires: September 10, 2015

S. Holmer
M. Flodman
E. Sprang
Google

March 9, 2015

RTP Extensions for Transport-wide Congestion Control
draft-holmer-rmcat-transport-wide-cc-extensions-00

Abstract

This document proposes an RTP header extension and an RTCP message for use in congestion control algorithms for RTP-based media flows. It adds transport-wide packet sequence numbers and corresponding feedback message so that congestion control can be performed on a transport level at the send-side, while keeping the receiver dumb.

Requirements Language

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 RFC 2119 [RFC2119].

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 10, 2015.

Copyright Notice

Table of Contents

1. Introduction

This document proposes RTP header extension containing a transport-wide packet sequence number and an RTCP feedback message feeding back the arrival times and sequence numbers of the packets received on a connection.

Some of the benefits that these extensions bring are:

- The congestion control algorithms are easier to maintain and improve as there is less synchronization between sender and receiver versions needed. It should be possible to implement [I-D.alvestrand-rmcat-congestion], [I-D.zhu-rmcat-nada] and [I-D.johansson-rmcat-scream-cc] with the proposed protocol.

- More flexibility in what algorithms are used, as long as they are having most of their logic on the send-side. For instance different behavior can be used depending on if the rate produced is application limited or not.

2. Transport-wide Sequence Number

2.1. Semantics

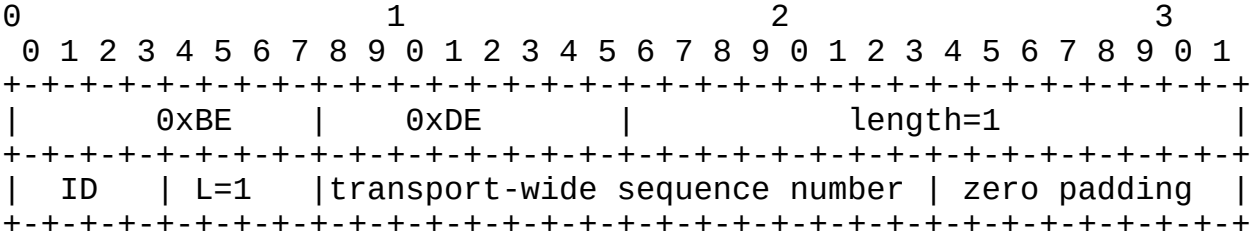
This RTP header extension is added on the transport layer, and uses the same counter for all packets which are sent over the same connection (for instance as defined by bundle).

The benefit with a transport-wide sequence numbers is two-fold:

- It is a better fit for congestion control as the congestion controller doesn’t operate on media streams, but on packet flows.
- It allows for earlier packet loss detection (and recovery) since a loss in stream A can be detected when a packet from stream B is received, thus we don’t have to wait until the next packet of stream A is received.

2.2. RTP header extension format

This document describes a message using the application specific payload type. This is suitable for experimentation; upon standardization, a specific type can be assigned for the purpose.



An RTP header extension with a 16 bits sequence number attached to all packets sent. This sequence number is incremented by 1 for each packet being sent over the same socket.

2.3. Signaling of use of this extension

When signalled in SDP, the standard mechanism for RTP header extensions [RFC5285] is used:

a=extmap:3 http://www.webrtc.org/experiments/rtp-hdrext/transport-sequence-number

3. Transport-wide RTCP Feedback Message

To allow the most freedom possible to the sender, information about each packet delivered is needed. The simplest way of accomplishing that is to have the receiver send back a message containing an arrival timestamp and a packet identifier for each packet received. This way, the receiver is dumb and simply records arrival timestamps (A) of packets. The sender keeps a map of in-flight packets, and upon feedback arrival it looks up the on-wire timestamp (S) of the corresponding packet. From these two timestamps the sender can compute metrics such as:

- Link propagation time delta: d(i) = A(i) - S(i) - (A(i-1) - S(i-1))
- Estimated queueing delay: q(i) = A(i) - S(i) - min{j=i-1..i-w}(A(j) - S(j))

Since the sender gets feedback about each packet sent, it will be set to better assess the cost of sending bursts of packets compared to aiming at sending at a constant rate decided by the receiver.

Two down-sides with this approach are:

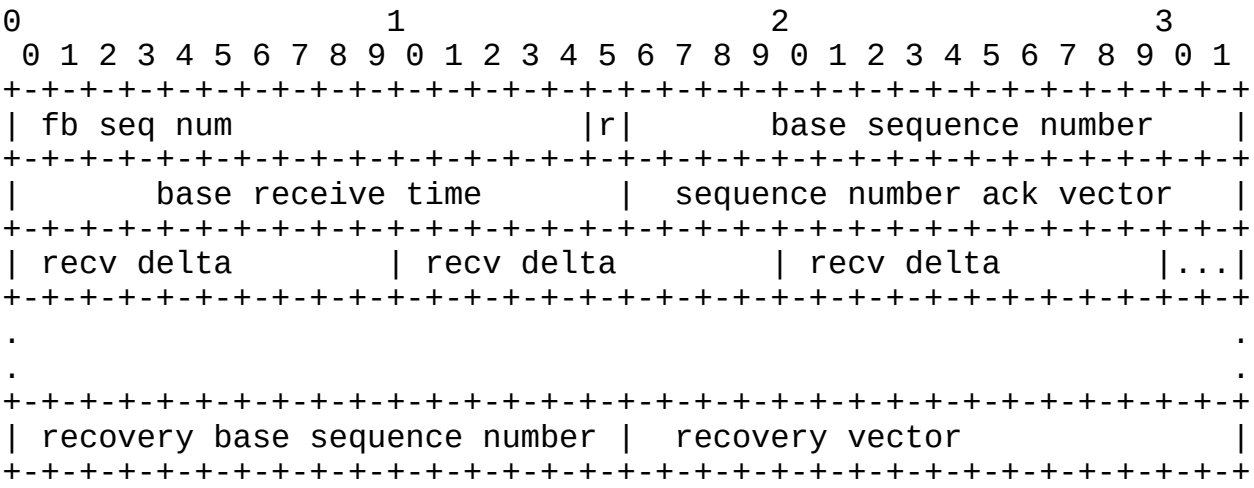
- It isn’t possible to differentiate between lost feedback on the downlink and lost packets on the uplink.
- Increased feedback rate on the reverse direction.

Lost feedback messages shouldn’t be a big problem as we could simply ignore losses which coincide with lost feedback messages from a congestion control perspective.

It is recommended that a feedback message is sent for every frame received, but in cases of low uplink bandwidth it is acceptable to send them less frequently, e.g., for instance once per RTT.

3.1. Message format

The message is an RTCP message with payload type 206. RFC 3550 [RFC3550] defines the range, RFC 4585 [RFC3550] defines the specific PT value 206 and the FMT value 15.



fb sequence number: Incremented by one for each feedback message sent. This can be used to figure out if feedback messages have been lost, so that the sender can avoid interpreting lost feedback messages on the downlink as lost media packets on the uplink.

r: Set if the recovery base sequence number and recovery vector are included.

base transport sequence number: The lowest received (not recovered) sequence number of this feedback message.

base receive time: Receive time of the base packet in multiples of 0.1 milliseconds, able to represent up to (2^16 - 1) * 0.1 = 6553.5 milliseconds. This allows probing of up to 96 Mbps with 1200 bytes packets.

sequence number ack vector: A bit vector where a 1 at position i represents that base sequence number + i + 1 has been received, and that a recv delta will be included in the feedback message. Recovered packets are not acked here, but will instead be acked using the recovery base sequence number and the recovery vector.

recv delta: A signed receive delta in multiples of 0.1 milliseconds relative to the base receive time, able to represent up to (2^9 - 1) * 0.1 = +/-51.1 milliseconds between packets. A feedback message contains the same number of recv deltas as there are 1s in the sequence number ack vector.

recovery base sequence number: The lowest recovered sequence number of this feedback message. It is optional and can be

omitted if no sequence numbers were recovered. If it is included the r bit of the second byte should be set.

recovery vector: A bit vector where a 1 at position i represents that sequence number recovery base + i + 1 has been recovered and therefore the sender can stop sending it.

The length of a feedback message can be derived by counting the number of acked packets and acked feedback packets. Therefore several feedback messages can be stacked to ack more than 17 packets with a single RTCP.

4. Overhead discussion

The overhead of this scheme will be higher than what the overhead is for a regular audio/video call over RTP. To get an understanding of this overhead, let’s consider the following example:

A 2 Mbps, 30 fps, (208 pps) video is sent in one direction and audio only is sent in the other direction. Average packet size of the video stream is 1200 bytes. A feedback message is sent over RTCP sent for every frame received.

The average feedback delay will be ~16.7 ms, compared to having logic at the receiver and immediately sending an RTCP when an event is detected.

The average protocol overhead is:

- 30 packets per second and (5*4 + 3*4) * 30 * 8 = 7680 bits per second.

- Transport-wide sequence number overhead: 4 * 8 * 208 = 6656 bps.

For extremely asymmetric connections the feedback frequency could be reduced.

5. IANA considerations

Upon publication of this document as an RFC (if it is decided to publish it), IANA is requested to register the string "goog-remb" in its registry of "rtcp-fb" values in the SDP attribute registry group.

6. Security Considerations

If the RTCP packet is not protected, it is possible to inject fake RTCP packets that can increase or decrease bandwidth. This is not different from security considerations for any other RTCP message.

7. Acknowledgements

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.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, July 2008.

8.2. Informative References

[I-D.alvestrand-rmcat-congestion] Holmer, S., Cicco, L., Mascolo, S., and H. Alvestrand, "A Google Congestion Control Algorithm for Real-Time Communication", draft-alvestrand-rmcat-congestion-02 (work in progress), February 2014.

[I-D.johansson-rmcat-scream-cc] Johansson, I. and Z. Sarker, "Self-Clocked Rate Adaptation for Multimedia", draft-johansson-rmcat-scream-cc-05 (work in progress), March 2015.

[I-D.zhu-rmcat-nada] Zhu, X., Pan, R., Ramalho, M., Cruz, S., Ganzhorn, C., Jones, P., and S. D'Aronco, "NADA: A Unified Congestion Control Scheme for Real-Time Media", draft-zhu-rmcat-nada-04 (work in progress), September 2014.

Appendix A. Change log

A.1. First version

Authors' Addresses

Stefan Holmer
Google
Kungsbron 2
Stockholm 11122
Sweden

Email: holmer@google.com

Magnus Flodman
Google
Kungsbron 2
Stockholm 11122
Sweden

Email: mflodman@google.com

Erik Sprang
Google
Kungsbron 2
Stockholm 11122
Sweden

Email: sprang@google.com