Audio/Video Transport Core Maintenance
Internet-Draft
Intended status: Informational
Expires: 10 August 2025

H. Alvestrand
Google

6 February 2025

Absolute Capture Timestamp RTP header extension
draft-ietf-avtcore-abs-capture-time-00

Abstract

This document describes an RTP header extension that can be used to carry information about the capture time of a video frame / audio sample, and include information that allows the capture time to be estimated by the receiver when the frame may have passed over multiple hops before reaching the receiver.

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://ietf-wg-avtcore.github.io/id-abs-capture-timestamp/draft-ietf-avtcore-abs-capture-timestamp.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-avtcore-abs-capture-time/.

Discussion of this document takes place on the Audio/Video Transport Core Maintenance Working Group mailing list (mailto:avt@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/avt/. Subscribe at https://www.ietf.org/mailman/listinfo/avt/.

Source for this draft and an issue tracker can be found at https://github.com/ietf-wg-avtcore/id-abs-capture-timestamp.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

When dealing with separate media streams originating from a single source, it is often desirable to present these in such a fashion that media generated at the same time is presented at the same time; the most well known form of this (between an audio stream and a video stream) is called "lip-sync".

In a simple setup with one source system, a single network hop and one destination system, this is usually done by lining up RTP timestamps. However, when multiple hops and multiple systems are involved, this task becomes more difficult; in particular, when one desires to synchronize media from multiple sources with independent clocks, where the media may have traveled over multiple network hops between the source and destination.

This memo describes one mechanism for providing information to make such synchronization possible.

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.

3. Applicability

The Absolute Capture Time extension is used to stamp RTP packets with a NTP timestamp showing when the first audio or video frame in a packet was originally captured.

Together with a round-trip-time estimate at the last hop, this extension allows a receiver to estimate the offset between its own clock and the capturer’s clock, thus providing a way to translate the capture timestamp into a time in its own clock.

One usage of this functionality is to provide a way to accomplish audio-to-video synchronization when RTCP-terminating intermediate systems (e.g. mixers) are involved.

Another usage is to provide statistics on sender-to-recipient delay in applications with multiple RTP hops without requiring clocks to be synchronized.

In the terminology of [RFC7667], the applicable scenarios are those where RTP is terminated at an intermediate system: Selective Forwarding Middlebox (3.7) and Point to Multipoint using RTCP-terminating MCU (3.9)

Other scenarios such as the Media-Switching Mixer (3.6.2) may be applicable if RTP header rewriting is applied, so that per-hop information is isolated to the hop it is destined for

Multicast scenarios are out of scope.

4. Absolute Capture Time

Name: "Absolute Capture Time"; "RTP Header Extension for Absolute Capture Time"

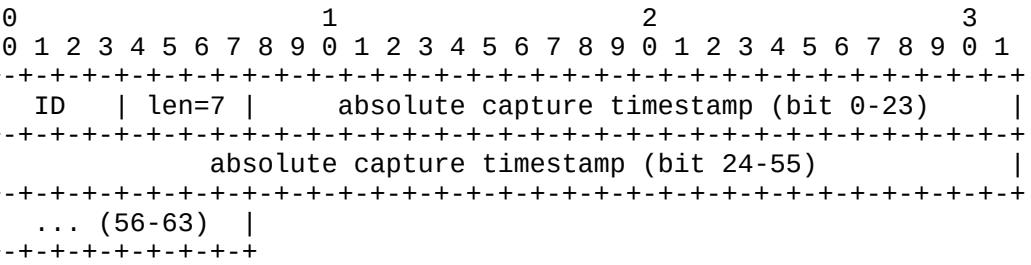
Formal name: http://www.webrtc.org/experiments/rtp-hdrext/abs-capture-time (http://www.webrtc.org/experiments/rtp-hdrext/abs-capture-time)

Status: This extension is defined here to allow for experimentation. Experience with the experiment has shown that it is useful; this draft therefore presents it to the IETF for consideration of whether to standardize it or leave it as a proprietary extension.

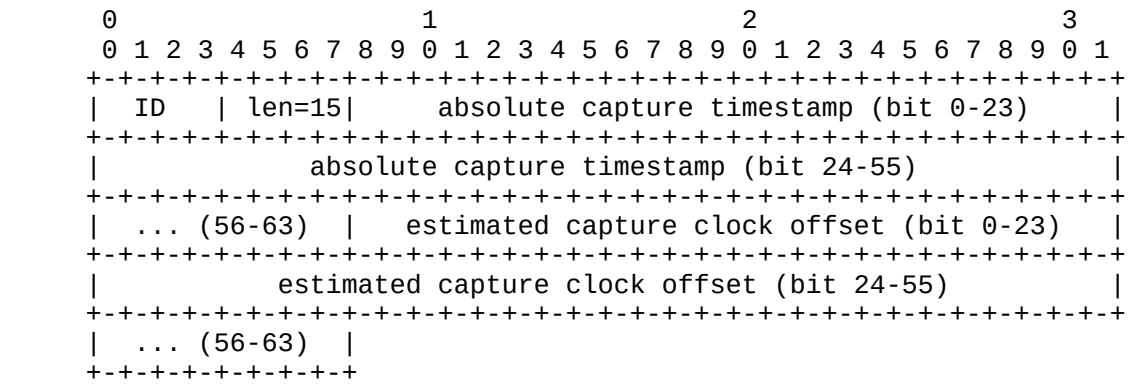
4.1. RTP header extension format

4.1.1. Data layout overview

Data layout of the shortened version of abs-capture-time with a 1-byte header + 8 bytes of data:



Data layout of the extended version of abs-capture-time with a 1-byte header + 16 bytes of data:



4.1.2. Data layout details

4.1.2.1. Absolute capture timestamp

Absolute capture timestamp is the NTP timestamp of when the first frame in a packet was originally captured. This timestamp MUST be based on the same clock as the clock used to generate NTP timestamps for RTCP sender reports on the capture system.

It’s not always possible to do an NTP clock readout at the exact moment of when a media frame is captured. A capture system MAY postpone the readout until a more convenient time. A capture system SHOULD have known delays (e.g. from hardware buffers) subtracted from the readout to make the final timestamp as close to the actual capture time as possible.

This field is encoded as a 64-bit unsigned fixed-point number with the high 32 bits for the timestamp in seconds and low 32 bits for the fractional part. This is also known as the UQ32.32 format and is what the RTP specification defines as the canonical format to represent NTP timestamps.

4.1.2.2. Estimated capture clock offset

Estimated capture clock offset is the sender’s estimate of the offset between its own NTP clock and the capture system’s NTP clock. The sender is here defined as the system that owns the NTP clock used to generate the NTP timestamps for the RTCP sender reports on this stream. The sender system is typically either the capture system or a mixer.

This field is encoded as a 64-bit two's complement *signed* fixed-point number with the high 32 bits for the seconds and low 32 bits for the fractional part. It's intended to make it easy for a receiver, that knows how to estimate the sender system's NTP clock, to also estimate the capture system's NTP clock:

Capture NTP Clock = Sender NTP Clock + Capture Clock Offset

If the estimated capture clock offset is not present (short format), it means that the sending system does not have enough data to compute a clock offset.

4.2. Details of operation

4.2.1. Capture system

The capture system generates the timestamp as close as possible to the true capture time. This may involve subtracting known delays in the capture pipeline from the time at which the system clock is read.

The capture time SHOULD be from the same clock as used to generate the NTP timestamp in RTP Sender Reports (SR) ([RFC3550] section 6.4.1), and indicate this by setting the "estimated capture clock offset" to zero; if this is not possible, the "estimated capture clock offset" MUST indicate the offset between the clock used for the capture timestamp and the clock used for RTP Sender Reports.

When the media is not captured in real time, such as when sending stored media, the sender can choose any reasonable start time; in that case, the offset between the chosen start time and the NTP clock of the sender system MUST be encoded into the "offset" value of the extension.

The "capture" timestamp should then advance according to the natural progression of the media; if this deviates from the NTP clock of the sender system, such as on a reader stall, this change SHOULD be reflected in the "offset" value.

4.2.2. Intermediate systems

An intermediate system MAY compute the outgoing capture clock offset as follows:

* Start with the "estimated capture clock offset" from the incoming packet

* Add the estimated offset between the sender's NTP clock and the intermediate's NTP clock (see Section 4.3)

This should give a reasonable estimate of the offset between the capture system's clock and the NTP timestamps sent in SR blocks by the intermediate system.

An intermediate system (e.g. mixer) MAY adjust these timestamps as needed. It MAY also choose to rewrite the timestamps completely, using its own NTP clock as reference clock, if it wants to present itself as a capture system for A/V-sync purposes.

4.2.3. End systems

A receiver can use the same algorithm as intermediate systems in order to compute the approximate time in the receiver's NTP clock at which the packet was generated. This should be more comparable between source systems with different clocks than just using the raw timestamp.

A receiver MUST treat the first CSRC in the CSRC list of a received packet as if it belongs to the capture system. If the CSRC list is empty, then the receiver MUST treat the SSRC as if it belongs to the capture system. Mixers SHOULD put the most prominent CSRC as the first CSRC in a packet's CSRC list.

4.3. Estimating the NTP clock offset

The NTP clock offset can be calculated from an SR packet in the following way:

* Take the NTP timestamp from the SR packet
* Subtract the arrival time of the SR packet, in the receiver's NTP clock
* Add half the estimated RTT between the sender and the receiver

The resulting number should be a reasonable approximation of the offset between the two clocks, with positive numbers indicating that the sender's clock is running ahead, and negative numbers indicate that the sender's clock is running behind.

Note that this method is sensitive to a number of issues:

* Clock drift means that you have to continuously monitor and update the offset
* RTT variance will cause variation in offset; a smoothed value should be used

* Asymmetric delays, if present, will cause biased estimates

This document is not normative about how the NTP clock offset is estimated.

4.4. Timestamp interpolation

A sender SHOULD save bandwidth by not sending abs-capture-time with every RTP packet. It SHOULD still send them at regular intervals (e.g. every second) to help mitigate the impact of clock drift and packet loss. Mixers SHOULD always send abs-capture-time with the first RTP packet after changing capture system.

A receiver SHOULD memorize the capture system (i.e. CSRC/SSRC), capture timestamp, and RTP timestamp of the most recently received abs-capture-time packet on each received stream. It can then use that information, in combination with RTP timestamps of packets without abs-capture-time, to extrapolate missing capture timestamps.

Timestamp interpolation works fine as long as there’s reasonably low NTP/RTP clock drift. This is not always true. Senders that detect "jumps" between its NTP and RTP clock mappings SHOULD send abs-capture-time with the first RTP packet after such a thing happening.

4.5. Year 2036 considerations

[RFC5905] section 6 describes how the 32-bit unsigned seconds field should be compared to a system clock, explaining how comparison of times works correctly when the 64-bit unsigned seconds field wraps in 2036 (the beginning of NTP era 1) provided proper two’s complement arithmetic is used for subtractions.

For the purposes of this memo, it is sufficient to note that a timestamp represents the closest timestamp in a relevant era.

5. Security Considerations

This extension carries information that may allow an attacker to identify different media streams on a connection. However, this information is already carried in the RTP SSRC, which is not encrypted, so it is unlikely that much additional information is exposed.

6. IANA Considerations

If the WG decides that this extension should be registered as a standardized extension, IANA is requested to perform the appropriate registration.

If the WG decides that this is a private extension, the URL http://www.webrtc.org/experiments/rtp-hdrext/abs-capture-time is used to identify the extension.

7. References

7.1. Normative References

[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>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <https://www.rfc-editor.org/rfc/rfc3550>.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <https://www.rfc-editor.org/rfc/rfc5905>.

[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>.

7.2. Informative References

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

Acknowledgments

Chen Xing, for writing the original version of this specification.

Bernard Aboba and Jonathan Lennox, for helpful comments.

Author’s Address

Harald Alvestrand
Google
Email: hta@google.com