              Support for Multiple Clock Rates in an RTP Session

Abstract

   This document clarifies the RTP specification regarding the use of
   different clock rates in an RTP session.  It also provides guidance
   on how legacy RTP implementations that use multiple clock rates can
   interoperate with RTP implementations that use the algorithm
   described in this document.  It updates RFC 3550.

Status of This Memo

   This is an Internet Standards Track document.

   This document is a product of the Internet Engineering Task Force
   (IETF).  It represents the consensus of the IETF community.  It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG).  Further information on
   Internet Standards is available in Section 2 of RFC 5741.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   http://www.rfc-editor.org/info/rfc7160.

Table of Contents

1.  Introduction

   The clock rate is a parameter of the payload format as identified in
   RTP and RTCP (RTP Control Protocol) by the payload type value.  It is
   often defined as being the same as the sampling rate but that is not
   always the case (see, for example, the G722 and MPA audio codecs
   [RFC3551]).

   An RTP sender can switch between different payloads during the
   lifetime of an RTP session and because clock rates are defined by
   payload format, it is possible that the clock rate will also vary
   during an RTP session.  Schulzrinne, et al. [RFC3550] lists using
   multiple clock rates as one of the reasons to not use different
   payloads on the same Synchronization Source (SSRC).  Unfortunately,
   this advice has not always been followed and some RTP implementations
   change the payload in the same SSRC, even if the different payloads
   use different clock rates.

This creates three problems:

o  The method used to calculate the RTP timestamp field in an RTP
   packet is underspecified.

o  When the same SSRC is used for different clock rates, it is
   difficult to know what clock rate was used for the RTP timestamp
   field in an RTCP Sender Report (SR) packet.

o  When the same SSRC is used for different clock rates, it is
   difficult to know what clock rate was used for the interarrival
   jitter field in an RTCP Receiver Report (RR) packet.

Table 1 contains a non-exhaustive list of fields in RTCP packets that
uses a clock rate as a unit:

| Field name          | RTCP packet type  | Reference  |
|---------------------|-------------------|------------|
| RTP timestamp       | SR                | [RFC3550]  |
| Interarrival jitter | RR                | [RFC3550]  |
| min_jitter          | XR Summary Block  | [RFC3611]  |
| max_jitter          | XR Summary Block  | [RFC3611]  |
| mean_jitter         | XR Summary Block  | [RFC3611]  |
| dev_jitter          | XR Summary Block  | [RFC3611]  |
| Interarrival jitter | IJ                | [RFC5450]  |
| RTP timestamp       | SMPTETC           | [RFC5484]  |
| Jitter              | RSI Jitter Block  | [RFC5760]  |
| Median jitter       | RSI Stats Block   | [RFC5760]  |

Table 1

Section 3 and its subsections try to list all of the algorithms known
to be used in existing RTP implementations at the time of writing.
These sections are not normative.

Section 4 and its subsections recommend a unique algorithm that
modifies RFC 3550.  These sections are normative.

2.  Terminology

   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].

   In addition, this document uses the following terms:

   Clock rate     The multiplier used to convert from a wallclock value
                  in seconds to an equivalent RTP timestamp value
                  (without the fixed random offset).  Note that RFC 3550
                  uses various terms like "clock frequency", "media
                  clock rate", "timestamp unit", "timestamp frequency",
                  and "RTP timestamp clock rate" as synonymous to clock
                  rate.

   RTP Sender     A logical network element that sends RTP packets,
                  sends RTCP SR packets, and receives RTCP reception
                  report blocks.

   RTP Receiver   A logical network element that receives RTP packets,
                  receives RTCP SR packets, and sends RTCP reception
                  report blocks.

3.  Legacy RTP

   The following sections describe the various ways in which legacy RTP
   implementations behave when multiple clock rates are used.  "Legacy
   RTP" refers to RFC 3550 without the modifications introduced by this
   document.

3.1.  Different SSRC

   One way of managing multiple clock rates is to use a different SSRC
   for each different clock rate, as in this case there is no ambiguity
   on the clock rate used by fields in the RTCP packets.  This method
   also seems to be the original intent of RTP as can be deduced from
   points 2 and 3 of Section 5.2 of RFC 3550.

   On the other hand, changing the SSRC can be a problem for some
   implementations designed to work only with unicast IP addresses,
   where having multiple SSRCs is considered a corner case.  Lip
   synchronization can also be a problem in the interval between the
   beginning of the new stream and the first RTCP SR packet.

3.2.  Same SSRC

   The simplest way to manage multiple clock rates is to use the same
   SSRC for all of the payload types regardless of the clock rates.

   Unfortunately, there is no clear definition on how the RTP timestamp
   should be calculated in this case.  The following subsections present
   the algorithms currently in use.

3.2.1.  Monotonic Timestamps

   This method of calculating the RTP timestamp ensures that the value
   increases monotonically.  The formula used by this method is as
   follows:

   timestamp = previous_timestamp
               + (current_capture_time - previous_capture_time)
               * current_clock_rate

   The problem with this method is that the jitter calculation on the
   receiving side gives an invalid result during the transition between
   two clock rates, as shown in Table 2 (Appendix A).  The capture and
   arrival time are measured in seconds, starting at the beginning of
   the capture of the first packet; clock rate is measured in Hz; the
   RTP timestamp does not include the random offset; and the transit,
   jitter, and average jitter use the clock rate as a unit.

   Calculating the correct transit time on the receiving side can be
   done by using the following formulas:

   1.  current_capture_time = (current_timestamp - previous_timestamp) /
       current_clock_rate + previous_capture_time

   2.  transit = current_clock_rate * (arrival_time -
       current_capture_time)

   3.  previous_capture_time = current_capture_time

   The main problem with this method, in addition to the fact that the
   jitter calculation described in RFC 3550 cannot be used, is that it
   is dependent on the previous RTP packets, which can be reordered or
   lost in the network.

3.2.2.  Non-monotonic Timestamps

   An alternate way of generating the RTP timestamps is to use the
   following formula:

                    timestamp = capture_time * clock_rate

   With this formula, the jitter calculation is correct but the RTP
   timestamp values are no longer increasing monotonically as shown in
   Table 3 (Appendix A).  RFC 3550 states that "[t]he sampling instant
   MUST be derived from a clock that increments monotonically . . .",
   but it does not say that the RTP timestamp must increment
   monotonically.

   The advantage with this method is that it works with the jitter
   calculation described in RFC 3550, as long as the correct clock rates
   are used.  It seems that this is what most implementations are using
   (based on a survey done at SIPit26 and on a survey of open source
   implementations, see Appendix C).

4.  Recommendations

   The following subsections describe behavioral recommendations for RTP
   senders (with and without RTCP) and RTP receivers.

4.1.  RTP Sender (with RTCP)

   An RTP Sender with RTCP turned on MUST use a different SSRC for each
   different clock rate.  An RTCP BYE MUST be sent and a new SSRC MUST
   be used if the clock rate switches back to a value already seen in
   the RTP stream.

   To accelerate lip synchronization, the next compound RTCP packet sent
   by the RTP sender MUST contain multiple SR packets, the first one
   containing the mapping for the current clock rate and the subsequent
   SR packet(s) containing the mapping for the other clock rates seen
   during the last period.

   The RTP extension defined by Perkins & Schierl [RFC6051] MAY be used
   to accelerate the synchronization.

4.2.  RTP Sender (without RTCP)

   An RTP Sender with RTCP turned off (i.e., having set the RTP Sender
   and RTP Receiver bandwidth modifiers [RFC3556] to 0) SHOULD use a
   different SSRC for each different clock rate but MAY use different
   clock rates on the same SSRC as long as the RTP timestamp is
   calculated as explained below:

Each time the clock rate changes, the start_offset and capture_start
values are calculated with the following formulas:

start_offset += (capture_time - capture_start) * previous_clock_rate
capture_start = capture_time

For the first RTP packet, the values are initialized with the
following values:

                   start_offset = random_initial_offset
                   capture_start = capture_time

After eventually updating these values, the RTP timestamp is
calculated with the following formula:

        timestamp = (capture_time - capture_start) * clock_rate
                    + start_offset

Note that in all the formulas, capture_start is the first instant
that the new timestamp rate is used.  The output of the above method
is exemplified in Table 4 (Appendix A).

4.3.  RTP Receiver

An RTP Receiver MUST calculate the jitter using the following
formula:

        D(i,j) = (arrival_time_j * clock_rate_i - timestamp_j)
                 - (arrival_time_i * clock_rate_i - timestamp_i)

An RTP Receiver MUST be able to handle a compound RTCP packet with
multiple SR packets.

5.  Security Considerations

When the algorithm described in Section 4.1 is used, the security
considerations described in RFC 3550 apply.

The algorithm described in Section 4.2 is new and so its security
properties were not considered in RFC 3550.  Although the RTP
timestamp is initialized with a random value like before, the
timestamp value depends on the current and previous clock rates; this
may or may not introduce a security vulnerability in the protocol.

6.  Acknowledgements

   Thanks to Colin Perkins, Ali C. Begen, Harald Alvestrand, Qin Wu,
   Jonathan Lennox, Barry Leiba, David Harrington, Stephen Farrell,
   Spencer Dawkins, Wassim Haddad, and Magnus Westerlund for comments,
   suggestions, and questions that helped to improve this document.

   Thanks to Bo Burman, who provided the values in Table 4 of
   Appendix A.

   Thanks to Robert Sparks and the attendees of SIPit 26 for the survey
   on multiple clock rates interoperability.

7.  References

7.1.  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.

7.2.  Informative References

   [AVT-VAR-RATE]
              Wenger, S. and C. Perkins, "RTP Timestamp Frequency for
              Variable Rate Audio Codecs", Work in Progress, October
              2004.

   [RFC3551]  Schulzrinne, H. and S. Casner, "RTP Profile for Audio and
              Video Conferences with Minimal Control", STD 65, RFC 3551,
              July 2003.

   [RFC3556]  Casner, S., "Session Description Protocol (SDP) Bandwidth
              Modifiers for RTP Control Protocol (RTCP) Bandwidth", RFC
              3556, July 2003.

   [RFC3611]  Friedman, T., Caceres, R., and A. Clark, "RTP Control
              Protocol Extended Reports (RTCP XR)", RFC 3611, November
              2003.

   [RFC5450]  Singer, D. and H. Desineni, "Transmission Time Offsets in
              RTP Streams", RFC 5450, March 2009.

   [RFC5484]  Singer, D., "Associating Time-Codes with RTP Streams", RFC
              5484, March 2009.

   [RFC5760]  Ott, J., Chesterfield, J., and E. Schooler, "RTP Control
              Protocol (RTCP) Extensions for Single-Source Multicast
              Sessions with Unicast Feedback", RFC 5760, February 2010.

   [RFC6051]  Perkins, C. and T. Schierl, "Rapid Synchronisation of RTP
              Flows", RFC 6051, November 2010.

Appendix A.  Example Values

   The following tables illustrate the timestamp and jitter values
   produced when the various methods discussed in the text are used.

   The values shown are purely exemplary, illustrative, and non-
   normative.

```
+-------+-------+-----------+---------+---------+--------+----------+
| Capt. | Clock | RTP       | Arrival | Transit | Jitter | Average  |
| time  | rate  | timestamp | time    |         |        | jitter   |
+-------+-------+-----------+---------+---------+--------+----------+
| 0     | 8000  | 0         | 0.1     | 800     |        |          |
|       |       |           |         |         |        |          |
| 0.02  | 8000  | 160       | 0.12    | 800     | 0      | 0        |
|       |       |           |         |         |        |          |
| 0.04  | 8000  | 320       | 0.14    | 800     | 0      | 0        |
|       |       |           |         |         |        |          |
| 0.06  | 8000  | 480       | 0.16    | 800     | 0      | 0        |
|       |       |           |         |         |        |          |
| 0.08  | 16000 | 800       | 0.18    | 2080    | 480    | 30       |
|       |       |           |         |         |        |          |
| 0.1   | 16000 | 1120      | 0.2     | 2080    | 0      | 28       |
|       |       |           |         |         |        |          |
| 0.12  | 16000 | 1440      | 0.22    | 2080    | 0      | 26       |
|       |       |           |         |         |        |          |
| 0.14  | 8000  | 1600      | 0.24    | 320     | 720    | 70       |
|       |       |           |         |         |        |          |
| 0.16  | 8000  | 1760      | 0.26    | 320     | 0      | 65       |
+-------+-------+-----------+---------+---------+--------+----------+
```

                      Table 2: Monotonic Timestamps

| Capt. time | Clock rate | RTP timestamp | Arrival time | Transit | Jitter | Average jitter |
|------------|------------|---------------|--------------|---------|--------|----------------|
| 0          | 8000       | 0             | 0.1          | 800     |        |                |
| 0.02       | 8000       | 160           | 0.12         | 800     | 0      | 0              |
| 0.04       | 8000       | 320           | 0.14         | 800     | 0      | 0              |
| 0.06       | 8000       | 480           | 0.16         | 800     | 0      | 0              |
| 0.08       | 16000      | 1280          | 0.18         | 1600    | 0      | 0              |
| 0.1        | 16000      | 1600          | 0.2          | 1600    | 0      | 0              |
| 0.12       | 16000      | 1920          | 0.22         | 1600    | 0      | 0              |
| 0.14       | 8000       | 1120          | 0.24         | 800     | 0      | 0              |
| 0.16       | 8000       | 1280          | 0.26         | 800     | 0      | 0              |

Table 3: Non-monotonic Timestamps

| Capt. time | Clock rate | RTP timestamp | Arrival time | Transit | Jitter | Average jitter |
|--------|-------|-----------|---------|---------|--------|---------|
| 0      | 8000  | 0         | 0.1     | 800     |        |         |
| 0.02   | 8000  | 160       | 0.12    | 800     | 0      | 0       |
| 0.04   | 8000  | 320       | 0.14    | 800     | 0      | 0       |
| 0.06   | 8000  | 480       | 0.16    | 800     | 0      | 0       |
| 0.08   | 16000 | 640       | 0.18    | 1600    | 0      | 0       |
| 0.1    | 16000 | 960       | 0.2     | 1600    | 0      | 0       |
| 0.12   | 16000 | 1280      | 0.22    | 1600    | 0      | 0       |
| 0.14   | 8000  | 1600      | 0.24    | 320     | 0      | 0       |
| 0.16   | 8000  | 1760      | 0.26    | 320     | 0      | 0       |

Table 4: Recommended Method for RTP Sender (without RTCP)

Appendix B.  Using a Fixed Clock Rate

   An alternate way of fixing the issue with using multiple clock rates
   was proposed by Wenger and Perkins [AVT-VAR-RATE].  This document
   proposed to define a unified clock rate, but the proposal was
   rejected at IETF 61.

Appendix C.  Behavior of Legacy Implementations

C.1.  libccrtp 2.0.2

   This library uses the formula described in Section 3.2.2.

   Note that this library uses gettimeofday(2) which is not guaranteed
   to increment monotonically (e.g., when the clock is adjusted by NTP).

C.2.  libmediastreamer0 2.6.0

   This library (which uses the oRTP library) uses the formula described
   in Section 3.2.2.

   Note that in some environments this library uses gettimeofday(2),
   which is not guaranteed to increment monotonically.

C.3.  libpjmedia 1.0

   This library uses the formula described in Section 3.2.2.

C.4.  Android RTP Stack 4.0.3

   This library changes the SSRC each time the format changes, as
   described in Section 3.1.

Authors' Addresses

   Marc Petit-Huguenin
   Impedance Mismatch

   EMail: petithug@acm.org


   Glen Zorn (editor)
   Network Zen
   227/358 Thanon Sanphawut
   Bang Na, Bangkok  10260
   Thailand

   Phone: +66 (0) 8-1000-4155
   EMail: glenzorn@gmail.com