                                      Colin Perkins
                                            USC/ISI
                                 Jonathan Rosenberg
                                         dynamicsoft
                                 Henning Schulzrinne
                                 Columbia University

                     RTP Testing Strategies

                  draft-ietf-avt-rtptest-06.txt

                     Status of this memo

                           Abstract

   This memo describes a possible testing strategy for RTP
   implementations. It is intended as an aid to implementors
   and does not specify a standard of any kind.

Contents

1 Introduction

This memo describes a possible testing strategy for RTP [1] implementations.
The tests are intended to help demonstrate interoperability of multiple
implementations, and to illustrate common implementation errors. They are
not intended to be an exhaustive set of tests and passing these tests does
not necessarily imply conformance to the complete RTP specification.

This memo is for information only and does not specify a standard
of any kind.

2 End systems

The architecture for testing RTP end systems is shown in figure 1.

```
                        +-----------------+
              +--------+ Test instrument +-----+
              |        +-----------------+      |
              |                                 |
     +-------+--------+              +-------+--------+
     |   First RTP   |              |   Second RTP   |
     | implementation |              | implementation |
     +---------------+              +----------------+
```
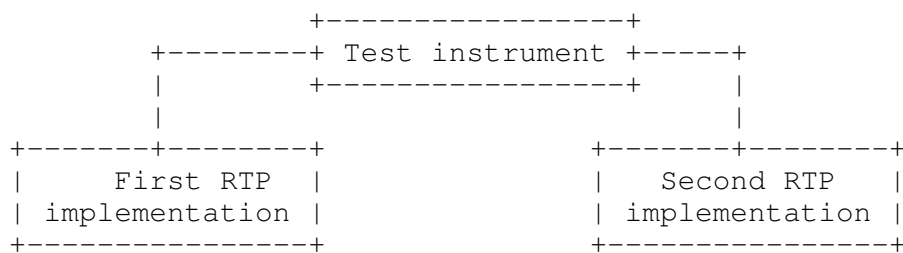
Figure 1: Testing architecture

Both RTP implementations send packets to the test instrument, which
forwards packets from one implementation to the other. Unless otherwise
specified, packets are forwarded with no additional delay and without
loss. The test instrument is required to delay or discard packets
in some of the tests. The test instrument is invisible to the RTP
implementations – it merely simulates poor network conditions.

The test instrument is also capable of logging packet contents for
inspection of their correctness.

A typical test setup might comprise three machines on a single Ethernet
segment. Two of these machines run the RTP implementations, the
third runs the test instrument. The test instrument is an application
level packet forwarder. Both RTP implementations are instructed to
send unicast RTP packets to the test instrument, which forwards packets
between them.

2.1 Media transport

The aim of these tests is to show that basic media flows can be
exchanged between the two RTP implementations. The initial test is
for the first RTP implementation to transmit and the second to receive.
If this succeeds, the process is reversed, with the second implementation
sending and the first receiving.

The receiving application should be able to handle the following
edge cases, in addition to normal operation:

  o Verify reception of packets which contain padding.

  o Verify reception of packets which have the marker bit set

  o Verify correct operation during sequence number wrap-around.

  o Verify correct operation during timestamp wrap-around.

  o Verify that the implementation correctly differentiates packets
     according to the payload type field.

  o Verify that the implementation ignores packets with unsupported
     payload types

  o Verify that the implementation can playout packets containing
     a CSRC list and non-zero CC field (see section 4).

The sending application should be verified to correctly handle the
following edge cases:

  o If padding is used, verify that the padding length indicator
     (last octet of the packet) is correctly set and that the length
     of the data section of the packet corresponds to that of this
     particular payload plus the padding.

  o Verify correct handling of the M bit, as defined by the profile.

  o Verify that the SSRC is chosen randomly.

  o Verify that the initial value of the sequence number is randomly
     selected.

  o Verify that the sequence number increments by one for each packet
     sent.

  o Verify correct operation during sequence number wrap-around.

  o Verify that the initial value of the timestamp is randomly selected.

   o Verify correct increment of timestamp (dependent on the payload
      format).

   o Verify correct operation during timestamp wrap-around.

   o Verify correct choice of payload type according to the chosen
      payload format, profile and any session level control protocol.

2.2 RTP Header Extension

An RTP implementation which does not use an extended header should
be able to process packets containing an extension header by ignoring
the extension.

If an implementation makes use of the header extension, it should
be verified that the profile specific field and the length field
of the extension are set correctly, and that the length of the packet
is consistent.

2.3 Basic RTCP

An RTP implementation is required to send RTCP control packets in
addition to data packets. The architecture for testing basic RTCP
functions is that shown in figure 1.

2.3.1 Sender and receiver reports

The first test requires both implementations to be run, but neither
sends data. It should be verified that RTCP packets are generated
by each implementation, and that those packets are correctly received
by the other implementation. It should also be verified that:

   o all RTCP packets sent are compound packets

   o all RTCP compound packets start with an empty RR packet

   o all RTCP compound packets contain an SDES CNAME packet

The first implementation should then be made to transmit data packets.
It should be verified that that implementation now generates SR packets
in place of RR packets, and that the second application now generates
RR packets containing a single report block. It should be verified
that these SR and RR packets are correctly received. The following
features of the SR packets should also be verified:

o   that the length field is consistent with both the length of
    the packet and the RC field

o   that the SSRC in SR packets is consistent with that in the RTP
    data packets

o   that the NTP timestamp in the SR packets is sensible (matches
    the wall clock time on the sending machine)

o   that the RTP timestamp in the SR packets is consistent with
    that in the RTP data packets

o   that the packet and octet count fields in the SR packets are
    consistent with the number of RTP data packets transmitted

In addition, the following features of the RR packets should also
be verified:

o   that the SSRC in the report block is consistent with that in
    the data packets being received

o   that the fraction lost is zero

o   that the cumulative number of packets lost is zero

o   that the extended highest sequence number received is consistent
    with the data packets being received (provided the round trip
    time between test instrument and receiver is smaller than the
    packet inter-arrival time, this can be directly checked by the
    test instrument).

o   that the interarrival jitter is small (a precise value cannot
    be given, since it depends on the test instrument and network
    conditions, but very little jitter should be present in this
    scenario).

o   that the last sender report timestamp is consistent with that
    in the SR packets (i.e. each RR passing through the test instrument
    should contain the middle 32 bits from the 64 bit NTP timestamp
    of the last SR packet which passed through the test instrument
    in the opposite direction).

o   that the delay since last SR field is sensible (an estimate
    may be made by timing the passage of an SR and corresponding
    RR through the test instrument, this should closely agree with
    the DLSR field)

It should also be verified that the timestamps, packet count and
octet count correctly wrap-around after the appropriate interval.

The next test is to show behaviour in the presence of packet loss.
The first implementation is made to transmit data packets, which
are received by the second implementation. This time, however, the
test instrument is made to randomly drop a small fraction (1% is
suggested) of the data packets. The second implementation should
be able to receive the data packets and process them in a normal
manner (with, of course, some quality degradation). The RR packets
should show a loss fraction corresponding to the drop rate of the
test instrument and should show an increasing cumulative number of
packets lost.

The loss rate in the test instrument is then returned to zero and
it is made to delay each packet by some random amount (the exact
amount depends on the media type, but a small fraction of the average
interarrival time is reasonable). The effect of this should be to
increase the reported interarrival jitter in the RR packets.

If these tests succeed, the process should be repeated with the second
implementation transmitting and the first receiving.

2.3.2 RTCP source description packets

Both implementations should be run, but neither is required to transmit
data packets. The RTCP packets should be observed and it should
be verified that each compound packet contains an SDES packet, that
that packet contains a CNAME item and that the CNAME is chosen according
to the rules in the RTP specification and profile (in many cases
the CNAME should be of the form 'example@10.0.0.1' but this may be
overridden by a profile definition).

If an application supports additional SDES items then it should be
verified that they are sent in addition to the CNAME with some SDES
packets (the exact rate at which these additional items are included
is dependent on the application and profile).

It should be verified that an implementation can correctly receive NAME,
EMAIL, PHONE, LOC, NOTE, TOOL and PRIV items, even if it does not send
them. This is because it may reasonably be expected to interwork with
other implementations which support those items. Receiving and ignoring
such packets is valid behaviour.

It should be verified that an implementation correctly sets the length
fields in the SDES items it sends, and that the source count and

packet length fields are correct. It should be verified that SDES
fields are not zero terminated.

It should be verified that an implementation correctly receives SDES
items which do not terminate in a zero byte.

2.3.3 RTCP BYE packets

Both implementations should be run, but neither is required to transmit
data packets. The first implementation is then made to exit and
it should be verified that an RTCP BYE packet is sent. It should
be verified that the second implementation reacts to this BYE packet
and notes that the first implementation has left the session.

If the test succeeds, the implementations should be restarted and
the process repeated with the second implementation leaving the session.

It should be verified that implementations handle BYE packets containing
the optional reason for leaving text (ignoring the text is acceptable).

2.3.4 Application defined RTCP packets

Tests for the correct response to application defined packets are
difficult to specify, since the response is clearly implementation
dependent. It should be verified that an implementation ignores APP
packets where the 4 octet name field is unrecognised. Implementations
which use APP packets should verify that they behave as expected.

2.4 RTCP transmission interval

The basic architecture for performing tests of the RTCP transmission
interval is shown in figure 2.

The test instrument is connected to the same LAN as the RTP implementation
being tested. It is assumed that the test instrument is preconfigured with
the addresses and ports used by the RTP implementation, and is also aware
of the RTCP bandwidth and sender/receiver fractions. The tests can be
conducted using either multicast or unicast.

The test instrument must be capable of sending arbitrarily crafted RTP and
RTCP packets to the RTP implementation. The test instrument should also be
capable of receiving packets sent by the RTP implementation, parsing them,
and computing metrics based on those packets.

```
                    +--------------+
                    |     test     |
                    |  instrument  |
                    +-----+--------+
                          |
          ------+----------+------------- LAN
              |
        +-------+--------+
        |       RTP      |
        | implementation |
        +----------------+
```
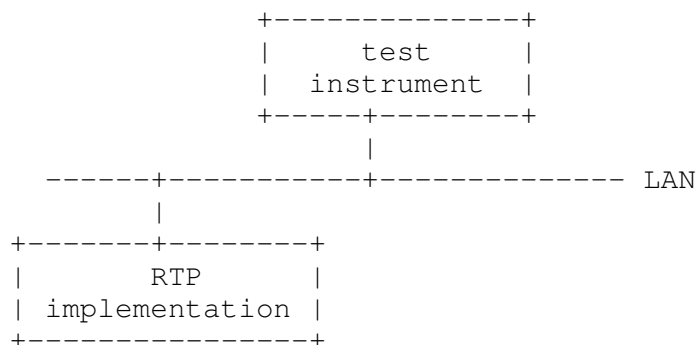
Figure 2: Testing architecture for RTCP

It is furthermore assumed that a number of basic controls over the
RTP implementation exist. These controls are:

o the ability to force the implementation to send or not send
   RTP packets at any desired point in time

o the ability to force the application to terminate its involvement
   in the RTP session, and for this termination to be known immediately
   to the test instrument

o the ability to set the session bandwidth and RTCP sender and
   receiver fractions

The second of these is required only for the test of BYE reconsideration,
and is the only aspect of these tests not easily implementable by
pure automation. It will generally require manual intervention to
terminate the session from the RTP implementation and to convey this
to the test instrument through some non-RTP means.

2.4.1 Basic Behavior

The first test is to verify basic correctness of the implementation
of the RTCP transmission rules. This basic behavior consists of:

o periodic transmission of RTCP packets

o randomization of the interval for RTCP packet transmission

o correct implementation of the randomization interval computations,
   with unconditional reconsideration

The RTP implementation acts as a receiver, and never sends any RTP
data packets. The implementation is configured with a large session
bandwidth, say 1 Mbit/s. This will cause the implementation to use
the minimal interval of 5s rather than the small interval based on
the session bandwidth and membership size. The implementation will
generate RTCP packets at this minimal interval, on average. The
test instrument generates no packets, but receives the RTCP packets
generated by the implementation. When an RTCP packet is received,
the time is noted by the test instrument. The difference in time
between each pair of subsequent packets (called the interval) is
computed. These intervals are stored, so that statistics based on
these intervals can be computed. It is recommended that this observation
process operate for at least 20 minutes.

An implementation passes this test if the intervals have the following
properties:

 o the minimum interval is never less than 2 seconds or more than
    2.5 seconds;

 o the maximum interval is never more than 7 seconds or less than
    5.5 seconds;

 o the average interval is between 4.5 and 5.5 seconds;

 o the number of intervals between x and x+500ms is less than the
    number of intervals between x+500ms and x+1s, for any x.

In particular, an implementation fails if the packets are sent with
a constant interval.

2.4.2 Step join backoff

The main purpose of the reconsideration algorithm is to avoid a flood
of packets that might occur when a large number of users simultaneously
join an RTP session. Reconsideration therefore exhibits a backoff
behavior in sending of RTCP packets when group sizes increase. This
aspect of the algorithm can be tested in the following manner.

The implementation begins operation. The test instrument waits for
the arrival of the first RTCP packet. When it arrives, the test
instrument notes the time and then immediately sends 100 RTCP RR
packets to the implementation, each with a different SSRC and SDES
CNAME. The test instrument should ensure that each RTCP packet is
of the same length. The instrument should then wait until the next

RTCP packet is received from the implementation, and the time of
such reception is noted.

Without reconsideration, the next RTCP packet will arrive within a
short period of time. With reconsideration, transmission of this
packet will be delayed. The earliest it can arrive depends on the
RTCP session bandwidth, receiver fraction, and average RTCP packet
size. The RTP implementation should be using the exponential averaging
algorithm defined in the specification to compute the average RTCP
packet size. Since this is dominated by the received packets (the
implementation has only sent one itself), the average will be roughly
equal to the length of the RTCP packets sent by the test instrument.
Therefore, the minimum amount of time between the first and second
RTCP packets from the implementation is:

$$T > 101 * S / (B * Fr * (e-1.5) * 2)$$

Where S is the size of the RTCP packets sent by the test instrument,
B is the RTCP bandwidth (normally five percent of the session bandwidth),
Fr is the fraction of RTCP bandwidth allocated to receivers (normally
75 percent), and e is the natural exponent. Without reconsideration,
this minimum interval Te would be much smaller:

$$Te > MAX([S / (B * Fr * (e-1.5) * 2)] , [2.5 / (e-1.5)])$$

B should be chosen sufficiently small so that T is around 60 seconds.
Reasonable choices for these parameters are B = 950 bits per second,
and S = 1024 bits. An implementation passes this test if the interval
between packets is not less than T above, and not more than 3 times
T.

Note: in all tests the value chosen for B, the RTCP bandwidth, is
calculated including the lower layer UDP/IP headers. In a typical
IPv4 based implementation, these comprise 28 octets per packet. A
common mistake is to forget that these are included when choosing
the size of packets to transmit.

The test should be repeated for the case when the RTP implementation
is a sender. This is accomplished by having the implementation send
RTP packets at least once a second. In this case, the interval between
the first and second RTCP packets should be no less than:

$$T > S / (B * Fs * (e-1.5) * 2)$$

Where Fs is the fraction of RTCP bandwidth allocated to senders,
usually 25%. Note that this value of T is significantly smaller
than the interval for receivers.

2.4.3 Steady State Behavior

In addition to the basic behavior in section 2.4.1, an implementation
should correctly implement a number of other, slightly more advanced
features:

 o   scale the RTCP interval with the group size;

 o   correctly divide bandwidth between senders and receivers;

 o   correctly compute the RTCP interval when the user is a sender

The implementation begins operation as a receiver. The test instrument
waits for the first RTCP packet from the implementation. When it
arrives, the test instrument notes the time, and immediately sends
50 RTCP RR packets and 50 RTCP SR packets to the implementation,
each with a different SSRC and SDES CNAME. The test instrument then
sends 50 RTP packets, using the 50 SSRC from the RTCP SR packets.
The test instrument should ensure that each RTCP packet is of the
same length. The instrument should then wait until the next RTCP
packet is received from the implementation, and the time of such
reception is noted. The difference between the reception of the
RTCP packet and the reception of the previous is computed and stored.
In addition, after every RTCP packet reception, the 100 RTCP and
50 RTP packets are retransmitted by the test instrument. This ensures
that the sender and member status of the 100 users does not time
out. The test instrument should collect the interval measurements
figures for at least 100 RTCP packets.

With 50 senders, the implementation should not try to divide the
RTCP bandwidth between senders and receivers, but rather group all
users together and divide the RTCP bandwidth equally. The test is
deemed successful if the average RTCP interval is within 5% of:

$$T = 101 * S/B$$

Where S is the size of the RTCP packets sent by the test instrument,
and B is the RTCP bandwidth. B should be chosen sufficiently small
so that the value of T is on the order of tens of seconds or more.
Reasonable values are S=1024 bits and B=3.4 kb/s.

The previous test is repeated. However, the test instrument sends
10 RTP packets instead of 50, and 10 RTCP SR and 90 RTCP RR instead
of 50 of each. In addition, the implementation is made to send at
least one RTP packet between transmission of every one of its own
RTCP packets.

In this case, the average RTCP interval should be within 5% of:

$$T = 11 * S / (B * Fs)$$

Where S is the size of the RTCP packets sent by the test instrument,
B is the RTCP bandwidth, and Fs is the fraction of RTCP banwidth
allocated for senders (normally 25%). The values for B and S should
be chosen small enough so that T is on the order of tens of seconds.
Reasonable choices are S=1024 bits and B=1.5 kb/s.

2.4.4 Reverse Reconsideration

The reverse reconsideration algorithm is effectively the opposite
of the normal reconsideration algorithm. It causes the RTCP interval
to be reduced more rapidly in response to decreases in the group
membership. This is advantageous in that it keeps the RTCP information
as fresh as possible, and helps avoids some premature timeout problems.

In the first test, the implementation joins the session as a receiver.
As soon as the implementation sends its first RTCP packet, the test
instrument sends 100 RTCP RR packets, each of the same length S,
and a different SDES CNAME and SSRC in each. It then waits for the
implementation to send another RTCP packet. Once it does, the test
instrument sends 100 BYE packets, each one containing a different
SSRC, but matching an SSRC from one of the initial RTCP packets.
Each BYE should also be the same size as the RTCP packets sent by
the test instrument. This is easily accomplished by using a BYE
reason to pad out the length. The time of the next RTCP packet from
the implementation is then noted. The delay T between this (the
third RTCP packet) and the previous should be no more than:

$$T < 3 * S / (B * Fr * (e-1.5) * 2)$$

Where S is the size of the RTCP and BYE packets sent by the test
instrument, B is the RTCP bandwidth, Fr is the fraction of the RTCP
bandwidth allocated to receivers, and e is the natural exponent.
B should be chosen such that T is on the order of tens of seconds.
A reasonable choice is S=1024 bits and B=168 bits per second.

This test demonstrates basic correctness of implementation. An
implementation without reverse reconsideration will not send its
next RTCP packet for nearly 100 times as long as the above amount.

In the second test, the implementation joins the session as a receiver. As
soon as it sends its first RTCP packet, the test instrument sends 100 RTCP
RR packets, each of the same length S, followed by 100 BYE packets, also of
length S. Each RTCP packet carries a different SDES CNAME and SSRC, and is
matched with precisely one BYE packet with the same SSRC. This will cause
the implementation to see a rapid increase and then rapid drop in group
membership.

The test is deemed succesful if the next RTCP packet shows up T seconds
after the first, and T is within:

$$2.5 / (e-1.5) < T < 7.5 / (e-1.5)$$

This tests correctness of the maintenance of the pmembers variable.
An incorrect implementation might try to execute reverse reconsideration
every time a BYE is received, as opposed to only when the group membership
drops below pmembers. If an implementation did this, it would end
up sending an RTCP packet immediately after receiving the stream
of BYE's. For this test to work, B must be chosen to be a large
value, around 1Mb/s.

2.4.5 BYE Reconsideration

The BYE reconsideration algorithm works in much the same fashion
as regular reconsideration, except applied to BYE packets. When a
user leaves the group, instead of sending a BYE immediately, it may
delay transmission of its BYE packet if others are sending BYE's.

The test for correctness of this algorithm is as follows. The RTP
implementation joins the group as a receiver. The test instrument
waits for the first RTCP packet. When the test instrument receives
this packet, the test instrument immediately sends 100 RTCP RR packets,
each of the same length S, and each containing a different SSRC and
SDES CNAME. Once the test instrument receives the next RTCP packet
from the implementation, the RTP implementation is made to leave
the RTP session, and this information is conveyed to the test instrument
through some non-RTP means. The test instrument then sends 100 BYE
packets, each with a different SSRC, and each matching an SSRC from
a previously transmitted RTCP packet. Each of these BYE packets
is also of size S. Immediately following the BYE packets, the test

instrument sends 100 RTCP RR packets, using the same SSRC/CNAMEs
as the original 100 RTCP packets.

The test is deemed successful if the implementation either never
sends a BYE, or if it does, the BYE is received by the test instrument
not earlier than T seconds, and not later than 3 * T seconds, after
the implementation left the session, where T is:

$$T = 100 * S / (2 * (e-1.5) * B)$$

S is the size of the RTCP and BYE packets, e is the natural exponent,
B is the RTCP bandwidth, and Fr is the RTCP bandwidth fraction for
receivers. S and B should be chosen so that T is on the order of
50 seconds. A reasonable choice is S=1024 bits and B=1.1 kb/s.

The transmission of the RTCP packets is meant to verify that the
implementation is ignoring non-BYE RTCP packets once it decides to
leave the group.

2.4.6 Timing out members

Active RTP participants are supposed to send periodic RTCP packets.
When a participant leaves the session, they may send a BYE, however
this is not required. Furthermore, BYE reconsideration may cause
a BYE to never be sent. As a result, participants must time out
other participants who have not sent an RTCP packet in a long time.
According to the specification, participants who have not sent an
RTCP packet in the last 5 intervals are timed out. This test verifies
that these timeouts are being performed correctly.

The RTP implementation joins a session as a receiver. The test instrument
waits for the first RTCP packet from the implementation. Once it arrives,
the test instrument sends 100 RTCP RR packets, each with a different SDES
and SSRC, and notes the time. This will cause the implementation to
believe that there are now 101 group participants, causing it to increase
its RTCP interval. The test instrument continues to monitor the RTCP
packets from the implementation. As each RTCP packet is received, the time
of its reception is noted, and the interval between RTCP packets is stored.
The 100 participants spoofed by the test instrument should eventually time
out at the RTP implementation. This should cause the RTCP interval to be
reduced to its minimum.

The test is deemed successful if the interval between RTCP packets
after the first is no less than:

$$Ti > 101 * S / (2 * (e-1.5) * B * Fr)$$

and this minimum interval is sustained no later than Td seconds after
the transmission of the 100 RR's, where Td is:

$$Td = 7 * 101 * S / (B * Fr)$$

and the interval between RTCP packets after this point is no less
than:

$$Tf > 2.5 / (e-1.5)$$

For this test to work, B and S must be chosen so Ti is on the order
of minutes. Recommended values are S = 1024 bits and B = 1.9 kbps.

2.4.7 Rapid SR's

The minimum interval for RTCP packets can be reduced for large session
bandwdiths. The reduction applies to senders only. The recommended
algorithm for computing this minimum interval is 360 divided by the
RTP session bandwidth, in kbps. For bandwidths larger than 72 kbps,
this interval is less than 5 seconds.

This test verifies the ability of an implementation to use a lower
RTCP minimum interval when it is a sender in a high bandwidth session.
The test can only be run on implementations that support this reduction,
since it is optional.

The RTP implementation is configured to join the session as a sender.
The session is configured to use 360 kbps. If the recommended algorithm
for computing the reduced minimum interval is used, the result is
a 1 second interval. If the RTP implementation uses a different
algorithm, the session bandwidth should be set in such a way to cause
the reduced minimum interval to be 1 second.

Once joining the session, the RTP implementation should begin to
send both RTP and RTCP packets. The interval between RTCP packets
is measured and stored until 100 intervals have been collected.

The test is deemed successful if the smallest interval is no less
than 1/2 a second, and the largest interval is no more than 1.5 seconds.
The average should be close to 1 second.

3 RTP translators

RTP translators should be tested in the same manner as end systems,
with the addition of the tests described in this section.

The architecture for testing RTP translators is shown in figure 3.

```
                          +----------------+
              +--------+  RTP Translator +-----+
              |        +----------------+      |
              |                                |
     +-------+--------+           +-------+--------+
     |    First RTP  |           |    Second RTP   |
     | implementation |           | implementation |
     +---------------+           +----------------+
```
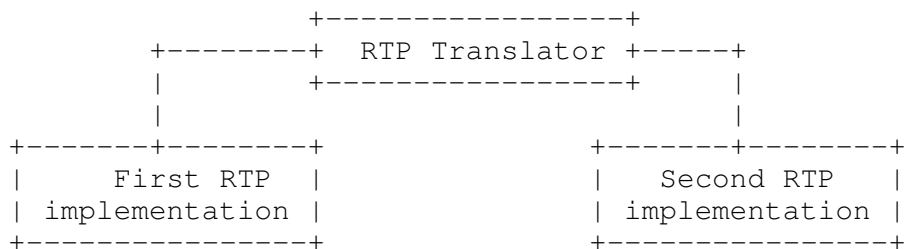
           Figure 3: Testing architecture for translators

The first RTP implementation is instructed to send data to the translator,
which forwards the packets to the other RTP implementation, after
translating then as desired. It should be verified that the second
implementation can playout the translated packets.

It should be verified that the packets received by the second implementation
have the same SSRC as those sent by the first implementation. The
CC should be zero and CSRC fields should not be present in the translated
packets. The other RTP header fields may be rewritten by the translator,
depending on the translation being performed, for example

 o the payload type should change if the translator changes the
    encoding of the data

 o the timestamp may change if, for example, the encoding, packetisation
    interval or framerate is changed

 o the sequence number may change if the translator merges or splits
    packets

 o padding may be added or removed, in particular if the translator
    is adding or removing encryption

 o the marker bit may be rewritten

If the translator modifies the contents of the data packets it should
be verified that the corresponding change is made to the RTCP packets,
and that the receivers can correctly process the modified RTCP packets.
In particular

o   the SSRC is unchanged by the translator

o   if the translator changes the data encoding it should also change
    the octet count field in the SR packets

o   if the translator combines multiple data packets into one it
    should also change the packet count field in SR packets

o   if the translator changes the sampling frequency of the data
    packets it should also change the RTP timestamp field in the
    SR packets

o   if the translator combines multiple data packets into one it
    should also change the packet loss and extended highest sequence
    number fields of RR packets flowing back from the receiver (it
    is legal for the translator to strip the report blocks and send
    empty SR/RR packets, but this should only be done if the transformation
    of the data is such that the reception reports cannot sensibly
    be translated)

o   the translator should forward SDES CNAME packets

o   the translator may forward other SDES packets

o   the translator should forward BYE packets unchanged

o   the translator should forward APP packets unchanged

When the translator exits it should be verified to send a BYE packet
to each receiver containing the SSRC of the other receiver. The
receivers should be verified to correctly process this BYE packet
(this is different to the BYE test in section 2.3.3 since multiple
SSRCs may be included in each BYE if the translator also sends its
own RTCP information).

4 RTP mixers

RTP mixers should be tested in the same manner as end systems, with
the addition of the tests described in this section.

The architecture for testing RTP mixers is shown in figure 4.

The first and second RTP implementations are instructed to send data
packets to the RTP mixer. The mixer combines those packets and sends
them to the third RTP implementation. The mixer should also process
RTCP packets from the other implementations, and should generate its
own RTCP reports.

```
         +----------------+
         |   Second RTP   |
         | implementation |
         +-------+--------+
                 |
                 |        +-----------+
             +-------+ RTP Mixer +-----+
             |        +-----------+     |
             |                          |
 +-------+--------+         +-------+--------+
 |    First RTP   |         |    Third RTP   |
 | implementation |         | implementation |
 +----------------+         +----------------+
```
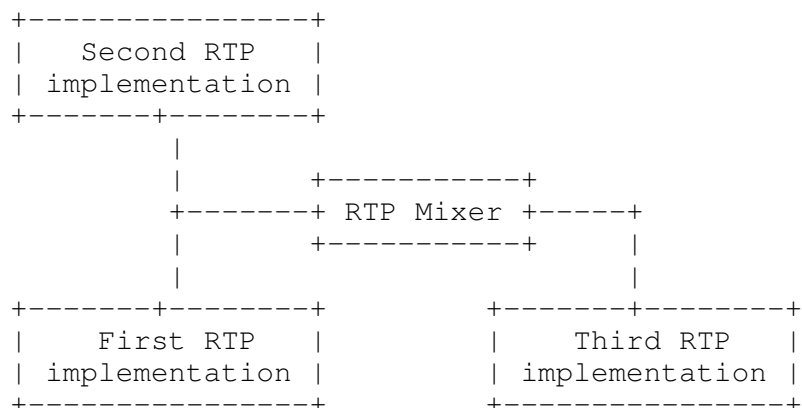
Figure 4: Testing architecture for mixers

It should be verified that the third RTP implementation can playout
the mixed packets. It should also be verified that

 o the CC field in the RTP packets received by the third implementation
    is set to 2

 o the RTP packets received by the third implementation contain
    2 CSRCs corresponding to the first and second RTP implementations

 o the RTP packets received by the third implementation contain
    an SSRC corresponding to that of the mixer

It should next be verified that the mixer generates SR and RR packets
for each cloud. The mixer should generate RR packets in the direction
of the first and second implementations, and SR packets in the direction
of the third implementation.

It should be verified that the SR packets sent to the third implementation
do not reference the first or second implementations, and vice versa.

It should be verified that SDES CNAME information is forwarded across
the mixer. Other SDES fields may optionally be forwarded.

Finally, one of the implementations should be quit, and it should
be verified that the other implementations see the BYE packet. This
implementation should then be restarted and the mixer should be quit.
It should be verified that the implementations see both the mixer
and the implementations on the other side of the mixer quit (illustrating
response to BYE packets containing multiple sources).

5 SSRC collision detection

RTP has provision for the resolution of SSRC collisions. These collisions
occur when two different session participants choose the same SSRC.
In this case, both participants are supposed to send a BYE, leave
the session, and rejoin with a different SSRC, but the same CNAME.
The purpose of this test is to verify that this function is present
in the implementation.

The test is straightforward. The RTP implementation is made to join
the multicast group as a receiver. A test instrument waits for the
first RTCP packet. Once it arrives, the test instrument notes the
CNAME and SSRC from the RTCP packet. The test instrument then generates
an RTCP receiver report. This receiver report contains an SDES chunk
with an SSRC matching that of the RTP implementation, but with a
different CNAME. At this point, the implementation should send a
BYE RTCP packet (containing an SDES chunk with the old SSRC and CNAME),
and then rejoin, causing it to send a receiver report containing
an SDES chunk, but with a new SSRC and the same CNAME.

The test is deemed succesful if the RTP implementation sends the
RTCP BYE and RTCP RR as described above within one minute of receiving
the colliding RR from the test instrument.

6 SSRC Randomization

According to the RTP specification, SSRC's are supposed to be chosen
randomly and uniformly over a 32 bit space. This randomization is
beneficial for several reasons:

 o It reduces the probability of collisions in large groups.

 o It simplifies the process of group sampling [3] which depends
    on the uniform distribution of SSRC's across the 32 bit space.

Unfortunately, verifying that a random number has 32 bits of uniform
randomness requires a large number of samples. The procedure below
gives only a rough validation to the randomness used for generating
the SSRC.

The test runs as follows. The RTP implementation joins the group
as a receiver. The test instrument waits for the first RTCP packet.
It notes the SSRC in this RTCP packet. The test is repeated 2500
times, resulting in a collection of 2500 SSRC.

The are then placed into 25 bins. An SSRC with value X is placed
into bin FLOOR(X/(2**32 / 25)). The idea is to break the 32 bit
space into 25 regions, and compute the number of SSRC in each region.
Ideally, there should be 40 SSRC in each bin. Of course, the actual
number in each bin is a random variable whose expectation is 40.
With 2500 SSRC, the coefficient of variation of the number of SSRC
in a bin is 0.1, which means the number should be between 36 and
44. The test is thus deemed successful if each bin has no less than
30 and no more than 50 SSRC.

Running this test may require substantial amounts of time, particularly
if there is no automated way to have the implementation join the
session. In such a case, the test can be run fewer times. With
26 tests, half of the SSRC should be less than 2**31, and the other
half higher. The cofficient of variation in this case is 0.2, so
the test is successful if there are more than 8 SSRC less than 2**31,
and less than 26.

In general, if the SSRC is collected N times, and there are B bins,
the coefficient of variation of the number of SSRC in each bin is
given by:

coeff = SQRT((B-1)/N)

7 Security Considerations

Implementations of RTP are subject to the security considerations
mentioned in the RTP specification [1] and any applicable RTP profile
(e.g. [2]). There are no additional security considerations implied
by the testing strategies described in this memo.

8 Authors' Address

Colin Perkins
USC Information Sciences Institute
3811 North Fairfax Drive
Suite 200
Arlington, VA 22203
Email: csp@isi.edu

Jonathan Rosenberg
dynamicsoft
72 Eagle Rock Ave.
First Floor
East Hanover, NJ 07936
Email: jdrosen@dynamicsoft.com

Henning Schulzrinne
Columbia University
M/S 0401
1214 Amsterdam Ave.
New York, NY 10027-7003
Email: schulzrinne@cs.columbia.edu

9 References

[1] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson, ``RTP:
A Transport Protocol to Real-Time Applications'', Work in progress,
Internet Engineering Task Force, March 2001.

[2] H. Schulzrinee and S. Casner, ``RTP Profile for Audio and Video
Conferences with Minimal Control'', Work in progress, Internet Engineering
Task Force, March 2001.