

AVTCORE
Internet-Draft
Updates: 3550, 4585 (if approved)
Intended status: Standards Track
Expires: January 04, 2015

J. Lennox
Vidyo
M. Westerlund
Ericsson
Q. Wu
Huawei
C. Perkins
University of Glasgow
July 03, 2014

Sending Multiple Media Streams in a Single RTP Session
draft-ietf-avtccore-rtp-multi-stream-05

Abstract

This memo expands and clarifies the behaviour of Real-time Transport Protocol (RTP) endpoints that use multiple synchronization sources (SSRCs). This occurs, for example, when an endpoint sends multiple media streams in a single RTP session. This memo updates RFC 3550 with regards to handling multiple SSRCs per endpoint in RTP sessions, with a particular focus on RTCP behaviour. It also updates RFC 4585 to update and clarify the calculation of the timeout of SSRCs and the inclusion of feedback messages.

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 January 04, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. Use Cases For Multi-Stream Endpoints	3
3.1. Endpoints with Multiple Capture Devices	3
3.2. Multiple Media Types in a Single RTP Session	3
3.3. Multiple Stream Mixers	4
3.4. Multiple SSRCs for a Single Media Source	4
4. Use of RTP by endpoints that send multiple media streams . .	5
5. Use of RTCP by Endpoints that send multiple media streams . .	5
5.1. RTCP Reporting Requirement	5
5.2. Initial Reporting Interval	5
5.3. Aggregation of Reports into Compound RTCP Packets	6
5.3.1. Maintaining AVG_RTCP_SIZE	7
5.3.2. Scheduling RTCP with Multiple Reporting SSRCs	8
5.4. Use of RTP/AVPF Feedback	10
5.4.1. Choice of SSRC for Feedback Packets	10
5.4.2. Scheduling an RTCP Feedback Packet	11
6. RTCP Considerations for Streams with Disparate Rates	12
6.1. Timing out SSRCs	13
6.1.1. AVPF T_rr_interval Behaviour	13
6.1.2. Avoiding Premature Timeout	14
6.1.3. RTP/AVP and RTP/AVPF Interoperability	15
6.1.4. Specified Behaviour	16
6.2. Tuning RTCP transmissions	16
6.2.1. RTP/AVP and RTP/SAVP	16
6.2.2. RTP/AVPF and RTP/SAVPF	18
7. Security Considerations	19
8. IANA Considerations	19
9. Open Issues	19
10. References	20
10.1. Normative References	20
10.2. Informative References	20
Authors' Addresses	22

1. Introduction

At the time the Real-Time Transport Protocol (RTP) [RFC3550] was originally designed, and for quite some time after, endpoints in RTP sessions typically only transmitted a single media stream, and thus used a single synchronization source (SSRC) per RTP session, where separate RTP sessions were typically used for each distinct media type. Recently, however, a number of scenarios have emerged in which endpoints wish to send multiple RTP media streams, distinguished by distinct RTP synchronization source (SSRC) identifiers, in a single RTP session. These are outlined in Section 3. Although the initial design of RTP did consider such scenarios, the specification was not consistently written with such use cases in mind. The specifications are thus somewhat unclear.

This memo updates [RFC3550] to clarify behaviour in use cases where endpoints use multiple SSRCs. It also updates [RFC4585] in regards to the timeout of inactive SSRCs to resolve problematic behaviour as well as clarifying the inclusion of feedback messages.

2. Terminology

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 RFC 2119 [RFC2119] and indicate requirement levels for compliant implementations.

3. Use Cases For Multi-Stream Endpoints

This section discusses several use cases that have motivated the development of endpoints that send RTP data using multiple SSRCs in a single RTP session.

3.1. Endpoints with Multiple Capture Devices

The most straightforward motivation for an endpoint to send multiple simultaneous RTP streams in a session is the scenario where an endpoint has multiple capture devices, and thus media sources, of the same media type and characteristics. For example, telepresence endpoints, of the type described by the CLUE Telepresence Framework [I-D.ietf-clue-framework], often have multiple cameras or microphones covering various areas of a room, and hence send several RTP streams.

3.2. Multiple Media Types in a Single RTP Session

Recent work has updated RTP [I-D.ietf-avtcore-multi-media-rtp-session] and SDP [I-D.ietf-mmusic-sdp-bundle-negotiation] to remove the historical assumption in RTP that media sources of different media types would

always be sent on different RTP sessions. In this work, a single endpoint's audio and video RTP media streams (for example) are instead sent in a single RTP session to reduce the number of transport layer flows used.

3.3. Multiple Stream Mixers

There are several RTP topologies which can involve a central device that itself generates multiple RTP media streams in a session. An example is a mixer providing centralized compositing for a multi-capture scenario like that described in Section 3.1. In this case, the centralized node is behaving much like a multi-capturer endpoint, generating several similar and related sources.

A more complex example is the selective forwarding middlebox, described in Section 3.7 of [I-D.ietf-avtcore-rtp-topologies-update]. This is a middlebox that receives media streams from several endpoints, and then selectively forwards modified versions of some RTP streams toward the other endpoints to which it is connected. For each connected endpoint, a separate media source appears in the session for every other source connected to the middlebox, "projected" from the original streams, but at any given time many of them can appear to be inactive (and thus are receivers, not senders, in RTP). This sort of device is closer to being an RTP mixer than an RTP translator, in that it terminates RTCP reporting about the mixed streams, and it can re-write SSRCs, timestamps, and sequence numbers, as well as the contents of the RTP payloads, and can turn sources on and off at will without appearing to be generating packet loss. Each projected stream will typically preserve its original RTCP source description (SDS) information.

3.4. Multiple SSRCs for a Single Media Source

There are also several cases where a single media source results in the usage of multiple SSRCs within the same RTP session. Transport robustness tools like RTP Retransmission [RFC4588] result in multiple SSRCs, one with source data, and another with the repair data. Scalable encoders and their RTP payload formats, like H.264's extension for Scalable Video Coding (SVC) [RFC6190] can be transmitted in a configuration where the scalable layers are distributed over multiple SSRCs within the same session, to enable RTP packet stream level (SSRC) selection and routing in conferencing middleboxes.

4. Use of RTP by endpoints that send multiple media streams

Every RTP endpoint will have an allocated share of the available session bandwidth, as determined by signalling and congestion control. The endpoint **MUST** keep its total media sending rate within this share. However, endpoints that send multiple media streams do not necessarily need to subdivide their share of the available bandwidth independently or uniformly to each media stream and its SSRCs. In particular, an endpoint can vary the allocation to different streams depending on their needs, and can dynamically change the bandwidth allocated to different SSRCs (for example, by using a variable rate codec), provided the total sending rate does not exceed its allocated share. This includes enabling or disabling media streams and their redundancy streams as more or less bandwidth becomes available.

5. Use of RTCP by Endpoints that send multiple media streams

The RTP Control Protocol (RTCP) is defined in Section 6 of [RFC3550]. The description of the protocol is phrased in terms of the behaviour of "participants" in an RTP session, under the assumption that each endpoint is a participant with a single SSRC. However, for correct operation in cases where endpoints can send multiple media streams, the specification needs to be interpreted with each SSRC counting as a participant in the session, so that an endpoint that has multiple SSRCs counts as multiple participants. The following describes several concrete cases where this applies.

5.1. RTCP Reporting Requirement

An RTP endpoint that has multiple SSRCs **MUST** treat each SSRC as a separate participant in the RTP session, sending RTCP reports for each of its SSRCs in every RTCP reporting interval. If the mechanism in [I-D.ietf-avtcore-rtp-multi-stream-optimisation] is not used, then each SSRC will send RTCP reports for all other SSRCs, including those co-located at the same endpoint.

If the endpoint has some SSRCs that are sending data and some that are only receivers, then they will receive different shares of the RTCP bandwidth and calculate different base RTCP reporting intervals. Otherwise, all SSRCs at an endpoint will calculate the same base RTCP reporting interval. The actual reporting intervals for each SSRC are randomised in the usual way, but reports can be aggregated as described in Section 5.3.

5.2. Initial Reporting Interval

When a participant joins a unicast session, the following text from Section 6.2 of [RFC3550] applies: "For unicast sessions... the delay before sending the initial compound RTCP packet MAY be zero." This also applies to the individual SSRCs of an endpoint that has multiple SSRCs, and such endpoints MAY send an initial RTCP packet for each of their SSRCs immediately upon joining a unicast session.

Caution has to be exercised, however, when an endpoint (or middlebox) with a large number of SSRCs joins a unicast session, since immediate transmission of many RTCP reports can create a significant burst of traffic, leading to transient congestion and packet loss due to queue overflows. Implementers are advised to consider sending immediate RTCP packets for only a small number of SSRCs (e.g., the one or two SSRCs they consider most important), with the initial RTCP packets for their other SSRCs being sent after the calculated initial RTCP reporting interval, to avoid self congestion.

(tbd: is this recommendation sufficiently strong?)

5.3. Aggregation of Reports into Compound RTCP Packets

As outlined in Section 5.1, an endpoint with multiple SSRCs has to treat each SSRC as a separate participant when it comes to sending RTCP reports. This will lead to each SSRC sending a compound RTCP packet in each reporting interval. Since these packets are coming from the same endpoint, it might reasonably be expected that they can be aggregated to reduce overheads. Indeed, Section 6.1 of [RFC3550] allows RTP translators and mixers to aggregate packets in similar circumstances:

"It is RECOMMENDED that translators and mixers combine individual RTCP packets from the multiple sources they are forwarding into one compound packet whenever feasible in order to amortize the packet overhead (see Section 7). An example RTCP compound packet as might be produced by a mixer is shown in Fig. 1. If the overall length of a compound packet would exceed the MTU of the network path, it SHOULD be segmented into multiple shorter compound packets to be transmitted in separate packets of the underlying protocol. This does not impair the RTCP bandwidth estimation because each compound packet represents at least one distinct participant. Note that each of the compound packets MUST begin with an SR or RR packet."

This allows RTP translators and mixers to generate compound RTCP packets that contain multiple SR or RR packets from different SSRCs, as well as any of the other packet types. There are no restrictions on the order in which the RTCP packets can occur within the compound packet, except the regular rule that the compound RTCP packet starts

with an SR or RR packet. Due to this rule, correctly implemented RTP endpoints will be able to handle compound RTCP packets that contain RTCP packets relating to multiple SSRCs.

Accordingly, endpoints that use multiple SSRCs MAY aggregate the RTCP packets sent by their different SSRCs into compound RTCP packets, provided they maintain the average RTCP packet size as described in Section 5.3.1, and schedule packet transmission and aggregation as described in Section 5.3.2.

5.3.1. Maintaining AVG_RTCP_SIZE

The RTCP scheduling algorithm in [RFC3550] works on a per-SSRC basis. Each SSRC sends a single compound RTCP packet in each RTCP reporting interval. When an endpoint uses multiple SSRCs, it is desirable to aggregate the compound RTCP packets sent by its SSRCs, reducing the overhead by forming a larger compound RTCP packet. This aggregation can be done as described in Section 5.3.2, provided the average RTCP packet size calculation is updated as follows.

Participants in an RTP session update their estimate of the average RTCP packet size (`avg_rtcp_size`) each time they send or receive an RTCP packet (see Section 6.3.3 of [RFC3550]). When a compound RTCP packet that contains RTCP packets from several SSRCs is sent or received, the `avg_rtcp_size` estimate for each SSRC that is reported upon is updated using `div_packet_size` rather than the actual packet size:

$$\text{avg_rtcp_size} = (1/16) * \text{div_packet_size} + (15/16) * \text{avg_rtcp_size}$$

where `div_packet_size` is `packet_size` divided by the number of SSRCs reporting in that compound packet. The number of SSRCs reporting in a compound packet is determined by counting the number of different SSRCs that are the source of Sender Report (SR) or Receiver Report (RR) RTCP packets within the compound RTCP packet. Non-compound RTCP packets (i.e., RTCP packets that do not contain an SR or RR packet [RFC5506]) are considered report on a single SSRC.

An SSRC doesn't follow the above rule, and instead uses the full RTCP compound packet size to calculate `avg_rtcp_size`, will derive an RTCP reporting interval that is overly large by a factor that is proportional to the number of SSRCs aggregated into compound RTCP packets and the size of set of SSRCs being aggregated relative to the total number of participants. This increased RTCP reporting interval can cause premature timeouts if it is more than five times the interval chosen by the SSRCs that understand compound RTCP that aggregate reports from many SSRCs. A 1500 octet MTU can fit six

typical size reports into a compound RTCP packet, so this is a real concern if endpoints aggregate RTCP reports from multiple SSRCs. If compatibility with non-updated endpoints is a concern, the number of reports from different SSRCs aggregated into a single compound RTCP packet SHOULD be limited.

5.3.2. Scheduling RTCP with Multiple Reporting SSRCs

When implementing RTCP packet scheduling for cases where multiple reporting SSRCs are aggregating their RTCP packets in the same compound packet there are a number of challenges. First of all, we have the goal of not changing the general properties of the RTCP packet transmissions, which include the general inter-packet distribution, and the behaviour for dealing with flash joins as well as other dynamic events.

The below specified mechanism deals with:

- o That one can't have a-priori knowledge about which RTCP packets are to be sent, or their size, prior to generating the packets. In which case, the time from generation to transmission ought to be as short as possible to minimize the information that becomes stale.
- o That one has an MTU limit, that one ought to avoid exceeding, as that requires lower-layer fragmentation (e.g., IP fragmentation) which impacts the packets' probability of reaching the receiver(s).

Schedule all the endpoint's local SSRCs individually for transmission using the regular calculation of T_n for the profile being used. Each time a SSRC's T_n timer expires, do the regular reconsideration. If the reconsideration indicates that an RTCP packet is to be sent:

1. Consider if an additional SSRC can be added. That consideration is done by picking the SSRC which has the T_n value closest in time to now (T_c).
2. Calculate how much space for RTCP packets would be needed to add that SSRC.
3. If the considered SSRC's RTCP Packets fit within the lower layer datagram's Maximum Transmission Unit, taking the necessary protocol headers into account and the consumed space by prior SSRCs, then add that SSRC's RTCP packets to the compound packet and go again to Step 1.

4. If the considered SSRC's RTCP Packets will not fit within the compound packet, then transmit the generated compound packet.
5. Update the RTCP Parameters for each SSRC that has been included in the sent RTCP packet. The Tp value for each SSRC MUST be updated as follows:

For the first SSRC: As this SSRC was the one that was reconsidered the tp value is set to the tc as defined in RTP [RFC3550].

For any additional SSRC: The tp value SHALL be set to the transmission time this SSRC would have had it not been aggregated and given the current existing session context. This value is derived by taking this SSRC's Tn value and performing reconsideration and updating tn until $tp + T \leq tn$. Then set tp to this tn value.

6. For the sent SSRCs calculate new tn values based on the updated parameters and reschedule the timers.

Reverse reconsideration needs to be performed as specified in RTP [RFC3550]. It is important to note that under the above algorithm when performing reconsideration, the value of tp can actually be larger than tc. However, that still has the desired effect of proportionally pulling the tp value towards tc (as well as tn) as the group size shrinks in direct proportion the reduced group size.

The above algorithm has been shown in simulations to maintain the inter-RTCP-packet transmission distribution for the SSRCs and consume the same amount of bandwidth as non-aggregated packets in RTP sessions with static sets of participants. With this algorithm the actual transmission interval for any SSRC triggering an RTCP compound packet transmission is following the regular transmission rules. It also handles the cases where the number of SSRCs that can be included in an aggregated packet varies. An SSRC that previously was aggregated and fails to fit in a packet still has its own transmission scheduled according to normal rules. Thus, it will trigger a transmission in due time, or the SSRC will be included in another aggregate.

The algorithm's behaviour under SSRC group size changes is under investigation. However, it is expected to be well behaved based on the following analyses.

RTP sessions where the number of SSRC are growing: When the group size is growing, the Td values grow in proportion to the number of new SSRCs in the group. The reconsideration when the timer for

the t_n expires, that SSRC will reconsider the transmission and with a certain probability reschedule the t_n timer. This part of the reconsideration algorithm is only impacted by the above algorithm by having t_p values that are in the future instead of set to the time of the actual last transmission at the time of updating t_p . Thus the scheduling causes in worst case a plateau effect for that SSRC. That effect depends on how far into the future t_p can advance.

RTP sessions where the number of SSRC are shrinking: When the group shrinks, reverse reconsideration moves the t_p and t_n values towards t_c proportionally to the number of SSRCs that leave the session compared to the total number of participants when they left. Thus the also group size reductions need to be handled.

In general the potential issue that might exist depends on how far into the future the t_p value can drift compared to the actual packet transmissions that occur. That drift can only occur for an SSRC that never is the trigger for RTCP packet transmission and always gets aggregated and where the calculated packet transmission interval randomly occurs so that $t_n - t_p$ for this SSRC is on average larger than the ones that gets transmitted.

5.4. Use of RTP/AVPF Feedback

This section discusses the transmission of RTP/AVPF feedback packets when the transmitting endpoint has multiple SSRCs.

5.4.1. Choice of SSRC for Feedback Packets

When an RTP/AVPF endpoint has multiple SSRCs, it can choose what SSRC to use as the source for the RTCP feedback packets it sends. Several factors can affect that choice:

- o RTCP feedback packets relating to a particular media type SHOULD be sent by an SSRC that receives that media type. For example, when audio and video are multiplexed onto a single RTP session, endpoints will use their audio SSRC to send feedback on the audio received from other participants.
- o RTCP feedback packets and RTCP codec control messages that are notifications or indications regarding RTP data processed by an endpoint MUST be sent from the SSRC used by that RTP data. This includes notifications that relate to a previously received request or command.
- o If separate SSRCs are used to send and receive media, then the corresponding SSRC SHOULD be used for feedback, since they have

differing RTCP bandwidth fractions. This can also effect the consideration if the SSRC can be used in immediate mode or not.

- o Some RTCP feedback packet types requires consistency in the SSRC used. For example, if one sets a TMMBR limitation, the same SSRC needs to be used to remove the limitation.

When an RTCP feedback packet is sent as part of a compound RTCP packet that aggregates reports from multiple SSRCs, there is no requirement that the compound packet contains an SR or RR packet generated by the sender of the RTCP feedback packet. For reduced-size RTCP packets, aggregation of RTCP feedback packets from multiple sources is not limited further than Section 4.2.2 of [RFC5506].

5.4.2. Scheduling an RTCP Feedback Packet

When an SSRC has a need to transmit a feedback packet in early mode it follows the scheduling rules defined in Section 3.5 in RTP/AVPF [RFC4585]. When following these rules the following clarifications need to be taken into account:

- o That a session is considered to be point-to-point or multiparty not based on the number of SSRCs, but the number of endpoints directly seen in the RTP session by the endpoint. tbd: Clarify what is considered to "see" an endpoint?
- o Note that when checking if there is already a scheduled compound RTCP packet containing feedback messages (Step 2 in Section 3.5.2), that check is done considering all local SSRCs.

TBD: The above does not allow an SSRC that is unable to send either an early or regular RTCP packet with the feedback message within the `T_max_fb_delay` to trigger another SSRC to send an early packet to which it could piggyback. Nor does it allow feedback to piggyback on even regular RTCP packet transmissions that occur within `T_max_fb_delay`. A question is if either of these behaviours ought to be allowed. The latter appears simple and straight forward. Instead of discarding a FB message in step 4a: alternative 2, one could place such messages in a cache with a discard time equal to `T_max_fb_delay`, and in case any of the SSRCs schedule an RTCP packet for transmission within that time, it includes this message. The former case can have more widespread impact on the application, and possibly also on the RTCP bandwidth consumption as it allows for more massive bursts of RTCP packets. Still, on a time scale of a regular reporting interval, it ought to have no effect on the RTCP bandwidth as the extra feedback messages increase the `avg_rtcp_size`.

6. RTCP Considerations for Streams with Disparate Rates

It is possible for a single RTP session to carry streams of greatly differing bandwidth. There are two scenarios where this can occur. The first is when a single RTP session carries multiple flows of the same media type, but with very different quality; for example a video switching multi-point conference unit might send a full rate high-definition video stream of the active speaker but only thumbnails for the other participants, all sent in a single RTP session. The second scenario occurs when audio and video flows are sent in a single RTP session, as discussed in [I-D.ietf-avtcore-multi-media-rtp-session].

An RTP session has a single set of parameters that configure the session bandwidth, the RTCP sender and receiver fractions (e.g., via the SDP "b=RR:" and "b=RS:" lines), and the parameters of the RTP/AVPF profile [RFC4585] (e.g., trr-int) if that profile (or its secure extension, RTP/SAVPF [RFC5124]) is used. As a consequence, the RTCP reporting interval will be the same for every SSRC in an RTP session. This uniform RTCP reporting interval can result in RTCP reports being sent more often than is considered desirable for a particular media type. For example, if an audio flow is multiplexed with a high quality video flow where the session bandwidth is configured to match the video bandwidth, this can result in the RTCP packets having a greater bandwidth allocation than the audio data rate. If the reduced minimum RTCP interval described in Section 6.2 of [RFC3550] is used in the session, which might be appropriate for video where rapid feedback is wanted, the audio sources could be expected to send RTCP packets more often than they send audio data packets. This is most likely undesirable, and while the mismatch can be reduced through careful tuning of the RTCP parameters, particularly trr_int in RTP/AVPF sessions, it is inherent in the design of the RTCP timing rules, and affects all RTP sessions containing flows with mismatched bandwidth.

Having multiple media types in one RTP session also results in more SSRCs being present in this RTP session. This increasing the amount of cross reporting between the SSRCs. From an RTCP perspective, two RTP sessions with half the number of SSRCs in each will be slightly more efficient. If someone needs either the higher efficiency due to the lesser number of SSRCs or the fact that one can't tailor RTCP usage per media type, they need to use independent RTP sessions.

When it comes to configuring RTCP the need for regular periodic reporting needs to be weighted against any feedback or control messages being sent. Applications using RTP/AVPF or RTP/SAVPF are RECOMMENDED to consider setting the trr-int parameter to a value suitable for the application's needs, thus potentially reducing the need for regular reporting and thus releasing more bandwidth for use for feedback or control.

Another aspect of an RTP session with multiple media types is that the RTCP packets, RTCP Feedback Messages, or RTCP XR metrics used might not be applicable to all media types. Instead, all RTP/RTCP endpoints need to correlate the media type of the SSRC being referenced in a message or packet and only use those that apply to that particular SSRC and its media type. Signalling solutions might have shortcomings when it comes to indicating that a particular set of RTCP reports or feedback messages only apply to a particular media type within an RTP session.

6.1. Timing out SSRCs

This section discusses issues around timing out SSRCs. After the discussion, clarified and mandated behaviour for SSRC timeout is specified.

6.1.1. AVPF T_rr_interval Behaviour

The RTP/AVPF profile includes a mechanism for suppressing regular RTCP reporting from being sent unnecessarily frequently if sufficient RTCP bandwidth is configured. This mechanism is defined in Section 3.5.3 of [RFC4585], and can be summarized as follows: if less than a randomized T_rr_interval value has passed since the last regular report, and no feedback messages need to be sent, then the RTCP regular report is suppressed. The randomization is done linearly in the interval 0.5 to 1.5 times T_rr_interval. The randomized T_rr_interval is recalculated after every transmitted regular packet, i.e. when t_rr_last was updated. The benefit of the suppression mechanism is that it avoids wasting bandwidth when there is nothing requiring frequent RTCP transmissions, but still allows utilization of the configured bandwidth when feedback is needed.

Unfortunately this suppression mechanism has some behaviour that is less than ideal. First of all, the randomized T_rr_interval is distributed over a larger range than the actual transmission interval for RTCP would be if T_rr_interval and Td had the same value. The reconsideration mechanism and its compensation factor result in the actual RTCP transmission intervals for a Td having a distribution that is exponentially growing more likely with higher values, and is bounded to the interval $[0.5/1.21828, 1.5/1.21828]*Td$, i.e. with a

Td value of 5 s [2.052, 6.156]. In comparison, the suppression acts in an interval that is 0.5 to 1.5 times the T_rr_interval, i.e. for T_rr_interval = 5 s this is [2.5, 7.5].

The effect of the above is that the time period between two RTCP packets when using T_rr_interval suppression can become very long compared to the average input values. The longest time interval between one transmitted regular RTCP compound packet and the next when T_rr_interval suppression is being used are: First the maximum T_rr_interval, i.e. $1.5 \cdot T_rr_interval$. Assuming that the last suppressed packet would have been sent at $1.5 \cdot T_rr_interval$, the maximum interval until a packet can be sent under the regular scheduling is $1.5/1.21828 \cdot Td$. Thus, the maximum time in total is $1.5 \cdot T_rr_interval + 1.5/1.21828 \cdot Td$.

If Td and T_rr_interval have the same value, i.e. the minimal interval desired (T_rr_interval) and the actual average interval specified by the RTCP scheduling algorithm (Td) are the same, one might expect that RTCP packets would be sent according to the regular mechanism. Instead, this algorithm results in the RTCP packets being sent anywhere from $0.5 \cdot Td$ to $\sim 2.731 \cdot Td$. The probability distribution over that time is also non-trivial in its shape, somewhat similar to a saw tooth.

Thus, we recommend that the AVPF regular transmission mechanism is revised in the future. This issue also has further implications as discussed in the next section.

6.1.2. Avoiding Premature Timeout

In RTP/AVP [RFC3550] the timeout behaviour is simple and is 5 times Td, where Td is calculated with a Tmin value of 5 seconds. In other words, if the RTCP bandwidth allowed for an RTCP interval more frequent than every 5 seconds on average, then timeout happened after $5 \cdot Td = 25$ seconds of no activity from the SSRC (RTP or RTCP), otherwise it was 5 average reporting intervals.

RTP/AVPF [RFC4585] introduced two different behaviours depending on the value of T_rr_interval. When T_rr_interval was 0, it defaulted to the same Td calculation in RTP/AVP [RFC3550]. However, when T_rr_interval is non-zero the Tmin value become T_rr_interval in that calculation, most likely to enable speed up the detection of timed out SSRCs. However, using a non-zero T_rr_interval has two consequences for RTP behaviour.

First, the number of actually sent RTCP packets for an SSRC that currently is not an active RTP sender can become very low due to the issue discussed above in Section 6.1.1. As the RTCP packet interval

can be as long as $2.73 \cdot T_d$, then during a $5 \cdot T_d$ time period an endpoint may in fact transmit only a single RTCP packet. The long intervals result in fewer RTCP packets, to a point where a one or two packet losses in RTCP result in timing out an SSRC.

Second, the change also increased RTP/AVPF's brittleness to both packet loss and configuration errors. In many cases, when one desires to use RTP/AVPF for its feedback, one will ensure that RTCP is configured for more frequent transmissions on average than every 5 seconds. Thus, many more RTP and RTCP packets can be transmitted during the time interval. Lets consider an implementation that would follow the RTP/AVP or RTP/AVPF with $T_{rr_interval} = 0$ rules for timeout, also when $T_{rr_interval}$ is not zero. In such a case when the configured value of the $T_{rr_interval}$ is significantly smaller than 5 seconds, e.g. less than 1 second, then a difference between using 0.1 seconds and 0.6 seconds has no significant impact on when an SSRC will be timed out. However, such a configuration difference between two endpoints following RFC 4585 will result in that the endpoint configured with $T_{rr_interval} = 0.1$ will frequently timeout SSRCs currently not sending RTP, from the endpoint configured with 0.6, as that is six times the T_d value used by the endpoint configured with $T_{rr_interval}=0.1$, assuming sufficient bandwidth. For this reason such a change is implemented below in Section 6.1.4.

6.1.3. RTP/AVP and RTP/AVPF Interoperability

If endpoints implementing the RTP/AVP and RTP/AVPF profiles (or their secure variants) are combined in a single RTP session, and the RTP/AVPF endpoints use a non-zero $T_{rr_interval}$ that is significantly lower than 5 seconds, then there is a risk that the RTP/AVPF endpoints will prematurely timeout the RTP/AVP SSRCs due to their different RTCP timeout intervals. Conversely, if the RTP/AVPF endpoints use a $T_{rr_interval}$ that is significant larger than 5 seconds, there is a risk that the RTP/AVP endpoints will timeout the RTP/AVPF SSRCs.

If such mixed RTP profiles are used, (though this is NOT RECOMMENDED), and the AVPF endpoint is not updated to follow this specification, then the RTP/AVPF session SHOULD use a non-zero $T_{rr_interval}$ that is 4 seconds.

It might appear strange to use a $T_{rr_interval}$ of 4 seconds. It might be intuitive that this value ought to be 5 seconds, as then both the RTP/AVP and RTP/AVPF would use the same timeout period. However, considering regular RTCP transmission and their packet intervals for RTP/AVPF its mean value will (with non-zero $T_{rr_interval}$) be larger than $T_{rr_interval}$ due to the scheduling algorithm's behaviour as discussed in Section 6.1.1. Thus, to enable

an equal amount of regular RTCP transmissions in each directions between RTP/AVP and RTP/AVPF endpoints, taking the altered timeout intervals into account, the optimal value is around four (4), where almost four transmissions will on average occur in each direction between the different profile types given an otherwise good configuration of parameters in regards to `T_rr_interval`. If the RTCP bandwidth parameters are selected so that `Td` based on bandwidth is close to 4, i.e. close to `T_rr_interval` the risk increases that RTP/AVPF SSRCs will be timed out by RTP/AVP endpoints, as the RTP/AVPF SSRC might only manage two transmissions in the timeout period.

6.1.4. Specified Behaviour

The above considerations result in the following clarification and RTP/AVPF specification change.

All SSRCs used in an RTP session MUST use the same timeout behaviour to avoid premature timeouts. This will depend on the RTP profile and its configuration. The RTP specification provides several options that can influence the values used when calculating the time interval. To avoid interoperability issues when using this specification, this document makes several clarifications to the calculations.

For RTP/AVP, RTP/SAVP, RTP/AVPF, and RTP/SAVPF, the timeout interval SHALL be calculated using a multiplier of 5, i.e. the timeout interval becomes $5 \cdot T_d$. The `T_d` calculation SHALL be done using a `T_min` value of 5 seconds, not the reduced minimal interval even if used to calculate RTCP packet transmission intervals. This changes the behaviour for the RTP/AVPF or RTP/SAVPF profiles when `T_rr_interval` $\neq 0$, a behaviour defined in Section 3.5.4 of RFC 4585, i.e. `T_min` in the `T_d` calculation is the `T_rr_interval`.

6.2. Tuning RTCP transmissions

This sub-section discusses what tuning can be done to reduce the downsides of the shared RTCP packet intervals. First, it is considered what possibilities exist for the RTP/AVP [RFC3551] profile, then what additional tools are provided by RTP/AVPF [RFC4585].

6.2.1. RTP/AVP and RTP/SAVP

When using the RTP/AVP or RTP/SAVP profiles the tuning one can do is very limited. The controls one has are limited to the RTCP bandwidth values and whether the minimum RTCP interval is scaled according to the bandwidth. As the scheduling algorithm includes both random factors and reconsideration, one can't simply calculate the expected

average transmission interval using the formula for T_d . But it does indicate the important factors affecting the transmission interval, namely the RTCP bandwidth available for the role (Active Sender or Participant), the average RTCP packet size, and the number of SSRCs classified in the relevant role. Note that if the ratio of senders to total number of session participants is larger than the ratio of RTCP bandwidth for senders in relation to the total RTCP bandwidth, then senders and receivers are treated together.

Let's start with some basic observations:

- a. Unless the scaled minimum RTCP interval is used, then T_d prior to randomization and reconsideration can never be less than 5 seconds (assuming default T_{min} of 5 seconds).
- b. If the scaled minimum RTCP interval is used, T_d can become as low as 360 divided by RTP Session bandwidth in kilobits. In SDP the RTP session bandwidth is signalled using $b=AS$. An RTP Session bandwidth of 72 kbps results in T_{min} being 5 seconds. An RTP session bandwidth of 360 kbps of course gives a T_{min} of 1 second, and to achieve a T_{min} equal to once every frame for a 25 Hz video stream requires an RTP session bandwidth of 9 Mbps! (The use of the RTP/AVPF or RTP/SAVPF profile allows a smaller T_{min} , and hence more frequent RTCP reports, as discussed below).
- c. Let's calculate the number (n) of SSRCs in the RTP session that 5% of the session bandwidth can support to yield a T_d value equal to T_{min} with minimal scaling. For this calculation we have to make two assumptions. The first is that we will consider most or all SSRC being senders, resulting in everyone sharing the available bandwidth. Secondly we will select an average RTCP packet size. This packet will consist of an SR, containing $(n-1)$ report blocks up to 31 report blocks, and an SDES item with at least a CNAME (17 bytes in size) in it. Such a basic packet will be 800 bytes for $n \geq 32$. With these parameters, and as the bandwidth goes up the time interval is proportionally decreased (due to minimal scaling), thus all the example bandwidths 72 kbps, 360 kbps and 9 Mbps all support 9 SSRCs.
- d. The actual transmission interval for a T_d value is $[0.5 \cdot T_d / 1.21828, 1.5 \cdot T_d / 1.21828]$, which means that for $T_d = 5$ seconds, the interval is actually $[2.052, 6.156]$ and the distribution is not uniform, but rather exponentially-increasing. The probability for sending at time X , given it is within the interval, is probability of picking X in the interval times the probability to randomly picking a number that is $\leq X$ within the interval with an uniform probability distribution. This results in that the majority of the probability mass is above the T_d value.

To conclude, with RTP/AVP and RTP/SAVP the key limitation for small unicast sessions is going to be the T_{min} value. Thus the RTP session bandwidth configured in RTCP has to be sufficiently high to reach the reporting goals the application has following the rules for the scaled minimal RTCP interval.

6.2.2. RTP/AVPF and RTP/SAVPF

When using RTP/AVPF or RTP/SAVPF we get a quite powerful additional tool, the setting of the $T_{rr_interval}$ which has several effects on the RTCP reporting. First of all as T_{min} is set to 0 after the initial transmission, the regular reporting interval is instead determined by the regular bandwidth based calculation and the $T_{rr_interval}$. This has the effect that we are no longer restricted by the minimal interval or even the scaling rule for the minimal rule. Instead the RTCP bandwidth and the $T_{rr_interval}$ are the governing factors.

Now it also becomes important to separate between the application's need for regular reports and RTCP feedback packet types. In both regular RTCP mode, as in Early RTCP Mode, the usage of the $T_{rr_interval}$ prevents regular RTCP packets, i.e. packets without any Feedback packets, to be sent more often than $T_{rr_interval}$. This value is applied to prevent any regular RTCP packet to be sent less than $T_{rr_interval}$ times a uniformly distributed random value from the interval [0.5,1.5] after the previous regular packet packet. The random value recalculated after each regular RTCP packet transmission.

So applications that have a use for feedback packets for some media streams, for example video streams, but don't want frequent regular reporting for audio, could configure the $T_{rr_interval}$ to a value so that the regular reporting for both audio and video is at a level that is considered acceptable for the audio. They could then use feedback packets, which will include RTCP SR/RR packets, unless reduced-size RTCP feedback packets [RFC5506] are used, and can include other report information in addition to the feedback packet that needs to be sent. That way the available RTCP bandwidth can be focused for the use which provides the most utility for the application.

Using $T_{rr_interval}$ still requires one to determine suitable values for the RTCP bandwidth value, in fact it might make it even more important, as this is more likely to affect the RTCP behaviour and performance than when using RTP/AVP, as there are fewer limitations affecting the RTCP transmission.

When using $T_{rr_interval}$, i.e. having it be non zero, there are configurations that have to be avoided. If the resulting T_d value is smaller but close to $T_{rr_interval}$ then the interval in which the actual regular RTCP packet transmission falls into becomes very large, from 0.5 times $T_{rr_interval}$ up to 2.73 times the $T_{rr_interval}$. Therefore for configuration where one intends to have T_d smaller than $T_{rr_interval}$, then T_d is RECOMMENDED to be targeted at values less than 1/4th of $T_{rr_interval}$ which results in that the range becomes $[0.5 * T_{rr_interval}, 1.81 * T_{rr_interval}]$.

With RTP/AVPF, using a $T_{rr_interval}$ of 0 or with another low value significantly lower than T_d still has utility, and different behaviour compared to RTP/AVP. This avoids the T_{min} limitations of RTP/AVP, thus allowing more frequent regular RTCP reporting. In fact this will result that the RTCP traffic becomes as high as the configured values.

(tbd: a future version of this memo will include examples of how to choose RTCP parameters for common scenarios)

There exists no method within the specification for using different regular RTCP reporting intervals depending on the media type or individual media stream.

7. Security Considerations

In the secure RTP protocol (SRTP) [RFC3711], the cryptographic context of a compound SRTCP packet is the SSRC of the sender of the first RTCP (sub-)packet. This could matter in some cases, especially for keying mechanisms such as Mikey [RFC3830] which allow use of per-SSRC keying.

Other than that, the standard security considerations of RTP apply; sending multiple media streams from a single endpoint does not appear to have different security consequences than sending the same number of streams.

8. IANA Considerations

No IANA actions needed.

9. Open Issues

At this stage this document contains a number of open issues. The below list tries to summarize the issues:

1. Do we need to provide a recommendation for unicast session joiners with many sources to not use 0 initial minimal interval from bit-rate burst perspective?
2. RTCP parameters for common scenarios in Section 6.2?
3. Is scheduling algorithm working well with dynamic changes?
4. Are the scheduling algorithm changes impacting previous implementations in such a way that the report aggregation has to be agreed on, and thus needs to be considered as an optimization?
5. An open question is if any improvements or clarifications ought to be allowed regarding FB message scheduling in multi-SSRC endpoints.

10. References

10.1. 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.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.
- [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, July 2006.
- [RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, February 2008.
- [RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.

10.2. Informative References

[I-D.ietf-avtcore-multi-media-rtp-session]

Westerlund, M., Perkins, C., and J. Lennox, "Sending Multiple Types of Media in a Single RTP Session", draft-ietf-avtcore-multi-media-rtp-session-05 (work in progress), February 2014.

[I-D.ietf-avtcore-rtp-multi-stream-optimisation]

Lennox, J., Westerlund, M., Wu, Q., and C. Perkins, "Sending Multiple Media Streams in a Single RTP Session: Grouping RTCP Reception Statistics and Other Feedback ", draft-ietf-avtcore-rtp-multi-stream-optimisation-00 (work in progress), July 2013.

[I-D.ietf-avtcore-rtp-topologies-update]

Westerlund, M. and S. Wenger, "RTP Topologies", draft-ietf-avtcore-rtp-topologies-update-02 (work in progress), May 2014.

[I-D.ietf-clue-framework]

Duckworth, M., Pepperell, A., and S. Wenger, "Framework for Telepresence Multi-Streams", draft-ietf-clue-framework-16 (work in progress), June 2014.

[I-D.ietf-mmusic-sdp-bundle-negotiation]

Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", draft-ietf-mmusic-sdp-bundle-negotiation-07 (work in progress), April 2014.

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

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

[RFC3830] Arkko, J., Carrara, E., Lindholm, F., Naslund, M., and K. Norrman, "MIKEY: Multimedia Internet KEYing", RFC 3830, August 2004.

[RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.

[RFC6190] Wenger, S., Wang, Y.-K., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, May 2011.

Authors' Addresses

Jonathan Lennox
Vidyo, Inc.
433 Hackensack Avenue
Seventh Floor
Hackensack, NJ 07601
USA

Email: jonathan@vidyo.com

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

Phone: +46 10 714 82 87
Email: magnus.westerlund@ericsson.com

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: sunseawq@huawei.com

Colin Perkins
University of Glasgow
School of Computing Science
Glasgow G12 8QQ
United Kingdom

Email: csp@csp Perkins.org

AVTCORE WG
Internet-Draft
Intended status: Standards Track
Expires: January 02, 2015

J. Lennox
Vidyo
M. Westerlund
Ericsson
Q. Wu
Huawei
C. Perkins
University of Glasgow
July 01, 2014

Sending Multiple Media Streams in a Single RTP Session: Grouping RTCP
Reception Statistics and Other Feedback
draft-ietf-avtccore-rtp-multi-stream-optimisation-03

Abstract

RTP allows multiple media streams to be sent in a single session, but requires each Synchronisation Source (SSRC) to send RTCP reception quality reports for every other SSRC visible in the session. This causes the number of RTCP reception reports to grow with the number of SSRCs, rather than the number of endpoints. In many cases most of these RTCP reception reports are unnecessary, since all SSRCs of an endpoint are co-located and see the same reception quality. This memo defines a Reporting Group extension to RTCP to reduce the reporting overhead in such scenarios.

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 January 02, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. RTCP Reporting Groups	3
3.1. Semantics and Behaviour of RTCP Reporting Groups	3
3.2. Identifying Members of an RTCP Reporting Group	5
3.2.1. Definition and Use of the RTCP RGRP SDES Item	5
3.2.2. Definition and Use of the RTCP RGRS Packet	6
3.3. Interactions with the RTP/AVPF Feedback Profile	8
3.4. Interactions with RTCP Extended Report (XR) Packets	9
3.5. Middlebox Considerations	9
3.6. SDP Signalling for Reporting Groups	10
4. Properties of RTCP Reporting Groups	10
4.1. Bandwidth Benefits of RTCP Reporting Groups	11
4.2. Compatibility of RTCP Reporting Groups	11
5. Security Considerations	12
6. IANA Considerations	14
7. References	15
7.1. Normative References	15
7.2. Informative References	15
Authors' Addresses	16

1. Introduction

The Real-time Transport Protocol (RTP) [RFC3550] is a protocol for group communication, supporting multiparty multimedia sessions. A single RTP session can support multiple participants sending at once, and can also support participants sending multiple simultaneous media streams. Examples of the latter might include a participant with multiple cameras who chooses to send multiple views of a scene, or a participant that sends audio and video flows multiplexed in a single RTP session. Rules for handling RTP sessions containing multiple media streams are described in [RFC3550] with some clarifications in [I-D.ietf-avtcore-rtp-multi-stream].

An RTP endpoint will have one or more synchronisation sources (SSRCs) that send media streams. It will have at least one SSRC for each

media stream it sends, and might use multiple SSRCs when using media scalability features [RFC6190], forward error correction, RTP retransmission [RFC4588], or similar mechanisms. An endpoint that is not sending any media streams, will have at least one SSRC to use for reporting and any feedback messages. Each SSRC has to send RTCP sender reports corresponding to the RTP packets it sends, and receiver reports for traffic it receives. That is, every SSRC will send RTCP packets to report on every other SSRC. This rule is simple, but can be quite inefficient for endpoints that send large numbers of media streams in a single RTP session. Consider a session comprising ten participants, each sending three media streams with their own SSRC. There will be 30 SSRCs in such an RTP session, and 30 RTCP reception reports will be sent per reporting interval as each SSRC reports on all the others. However, the three SSRCs comprising each participant will almost certainly see identical reception quality, since they are co-located. If there was a way to indicate that several SSRCs are co-located, and see the same reception quality, then two-thirds of those RTCP reports could be suppressed. This would allow the remaining RTCP reports to be sent more often, while keeping within the same RTCP bandwidth fraction.

This memo defines such an RTCP extension, RTCP Reporting Groups. This extension is used to indicate the SSRCs that originate from the same endpoint, and therefore have identical reception quality, hence allowing the endpoints to suppress unnecessary RTCP reception quality reports.

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 [RFC2119].

3. RTCP Reporting Groups

An RTCP Reporting Group is a set of synchronization sources (SSRCs) that are co-located at a single endpoint (which could be an end host or a middlebox) in an RTP session. Since they are co-located, every SSRC in the RTCP reporting group will have an identical view of the network conditions, and see the same lost packets, jitter, etc. This allows a single representative to send RTCP reception quality reports on behalf of the rest of the reporting group, reducing the number of RTCP packets that need to be sent without loss of information.

3.1. Semantics and Behaviour of RTCP Reporting Groups

A group of co-located SSRCs that see identical network conditions can form an RTCP reporting group. If reporting groups are in use, an RTP

endpoint with multiple SSRCs MAY put those SSRCs into a reporting group if their view of the network is identical; i.e., if they report on traffic received at the same interface of an RTP endpoint. SSRCs with different views of the network MUST NOT be put into the same reporting group.

An endpoint that has combined its SSRCs into an RTCP reporting group will choose one (or a subset) of those SSRCs as a "reporting source" for that RTCP reporting group. A reporting source will send RTCP SR/RR reception quality reports on behalf of the other members of the RTCP reporting group. A reporting source MUST suppress the RTCP SR/RR reports that relate to other members of the reporting group, and only report on remote SSRCs. The other members (non reporting sources) of the RTCP reporting group will suppress their RTCP reception quality reports, and instead send an RTCP RGRS packet (see Section 3.2.2) to indicate that they are part of an RTCP reporting group and give the SSRCs of the reporting sources.

If there are large numbers of remote SSRCs in the RTP session, then the reception quality reports generated by the reporting source might grow too large to fit into a single compound RTCP packet, forcing the reporting source to use a round-robin policy to determine what remote SSRCs it includes in each compound RTCP packet, and so reducing the frequency of reports on each SSRC. To avoid this, in sessions with large numbers of remote SSRCs, an RTCP reporting group MAY use more than one reporting source. If several SSRCs are acting as reporting sources for an RTCP reporting group, then each reporting source MUST have non-overlapping sets of remote SSRCs it reports on.

An endpoint SHOULD NOT create an RTCP reporting group that comprises only a single local SSRC (i.e., an RTCP reporting group where the reporting source is the only member of the group), unless it is anticipated that the group might have additional SSRCs added to it in the future.

If a reporting source leaves the RTP session (i.e., if it sends a RTCP BYE packet, or leaves the session without sending BYE under the rules of [RFC3550] section 6.3.7), the remaining members of the RTCP reporting group MUST either (a) have another reporting source, if existing, report on the remote SSRCs the leaving SSRC reported on, (b) choose a new reporting source, or (c) disband the RTCP reporting group and begin sending reception quality reports following [RFC3550] and [I-D.ietf-avtcore-rtp-multi-stream].

The RTCP timing rules assign different bandwidth fractions to senders and receivers. This lets senders transmit RTCP reception quality reports more often than receivers. If a reporting source in an RTCP reporting group is a receiver, but one or more non-reporting SSRCs in

the RTCP reporting group are senders, then the endpoint MAY treat the reporting source as a sender for the purpose of RTCP bandwidth allocation, increasing its RTCP bandwidth allocation, provided it also treats one of the senders as if it were a receiver and makes the corresponding reduction in RTCP bandwidth for that SSRC.

3.2. Identifying Members of an RTCP Reporting Group

When RTCP Reporting Groups are in use, the other SSRCs in the RTP session need to be able to identify which SSRCs are members of an RTCP reporting group. Two RTCP extensions are defined to support this: the RTCP RGRP SDES item is used by the reporting source(s) to identify an RTCP reporting group, and the RTCP RGRS packet is used by other members of an RTCP reporting group to identify the reporting source(s).

3.2.1. Definition and Use of the RTCP RGRP SDES Item

A new RTCP SDES item is defined to identify an RTCP reporting group. The motivation for giving a reporting group an identify is to ensure that the RTCP reporting group and its member SSRCs can be correctly associated when there are multiple reporting sources, and to ensure that a reporting SSRC can be associated with the correct reporting group if an SSRC collision occurs.

The RTCP Source Description (SDES) RGRP item is defined. The RTCP SDES RGRP item MUST be sent by the reporting sources in a reporting group, and MUST NOT be sent by other members of the reporting group or by SSRCs that are not members of any RTCP reporting group. Specifically, every reporting source in an RTCP reporting group MUST include an RTCP SDES packet containing an RGRP item in every compound RTCP packet in which it sends an RR or SR packet (i.e., in every RTCP packet it sends, unless Reduced-Size RTCP [RFC5506] is in use).

Syntactically, the format of the RTCP SDES RGRP item is identical to that of the RTCP SDES CNAME item [RFC7022], except that the SDES item type field MUST have value RGRP=(TBA) instead of CNAME=1. The value of the RTCP SDES RGRP item MUST be chosen with the same concerns about global uniqueness and the same privacy considerations as the RTCP SDES CNAME item. The value of the RTCP SDES RGRP item MUST be stable throughout the lifetime of the reporting group, even if the some or all of the reporting sources change their SSRC due to collisions, or if the set of reporting sources changes.

Note to RFC Editor: please replace (TBA) in the above paragraph with the RTCP SDES item type number assigned to the RGRP item, then delete this note.

An RTP mixer or translator that forwards RTCP SR or RR packets from members of a reporting group MUST forward the corresponding RTCP SDES RGRP items as well, even if it otherwise strips SDES items other than the CNAME item.

3.2.2. Definition and Use of the RTCP RGRS Packet

A new RTCP packet type is defined to allow the members of an RTCP reporting group to identify the reporting sources for that group. This allows participants in an RTP session to distinguish an SSRC that is sending empty RTCP reception reports because it is a member of an RTCP reporting group, from an SSRC that is sending empty RTCP reception reports because it is not receiving any traffic. It also explicitly identifies the reporting sources, allowing other members of the RTP session to know which SSRCs are acting as the reporting sources for an RTCP reporting group, and allowing them to detect if RTCP packets from any of the reporting sources are being lost.

The format of the RTCP RGRS packet is defined below. It comprises the fixed RTCP header that indicates the packet type and length, the SSRC of the packet sender, and a list of reporting sources for the RTCP reporting group of which the packet sender is a member.

```

      0                   1                   2                   3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|V=2|P|          SC   | PT=RGRS (TBA) |          length          |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|          SSRC of packet sender          |
+===+===+===+===+===+===+===+===+===+===+===+===+===+===+===+===+
:          List of SSRCs for the Reporting Source(s)          :
:          ...          :
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The fields in the RTCP RGRS packet have the following definition:

version (V): This field identifies the RTP version. The current RTP version is 2.

padding (P): If set, the padding bit indicates that the RTCP packet contains additional padding octets at the end that are not part of the control information but are included in the length field. See [RFC3550].

Source Count (SC): Indicates the number of reporting source SSRCs that are included in this RTCP packet. As the RTCP RGRS packet MUST NOT be sent by reporting sources, all the SSRCs in the

list of reporting sources will be different from the SSRC of the packet sender. Every RTCP RGRS packet MUST contain at least one reporting source SSRC.

Payload type (PT): The RTCP packet type number that identifies the packet as being an RTCP RGRS packet. The RGRS RTCP packet has the value [TBA].

Note to RFC Editor: please replace [TBA] here, and in the packet format diagram above, with the RTCP packet type that IANA assigns to the RTCP RGRS packet.

Length: The length of this packet in 32-bit words minus one, including the header and any padding. This is in line with the definition of the length field used in RTCP sender and receiver reports [RFC3550]. Since all RTCP RGRS packets include at least one reporting source SSRC, the length will always be 2 or greater.

SSRC of packet sender: The SSRC of the sender of this packet.

List of SSRCs for the Reporting Source(s): A variable length size (as indicated by SC header field) of the 32 bit SSRC values of the reporting sources for the RTCP Reporting Group of which the packet sender is a member.

Every source that belongs to an RTCP reporting group but is not a reporting source MUST include an RTCP RGRS packet in every compound RTCP packet in which it sends an RR or SR packet (i.e., in every RTCP packet it sends, unless Reduced-Size RTCP [RFC5506] is in use). Each RTCP RGRS packet MUST contain the SSRC identifier of at least one reporting source. If there are more reporting sources in an RTCP reporting group than can fit into an RTCP RGRS packet, the members of that reporting group MUST send the SSRCs of the reporting sources in a round-robin fashion in consecutive RTCP RGRS packets, such that all the SSRCs of the reporting sources are included over the course of several RTCP reporting intervals.

An RTP mixer or translator that forwards RTCP SR or RR packets from members of a reporting group MUST also forward the corresponding RGRS RTCP packets. If the RTP mixer or translator rewrites SSRC values of the packets it forwards, it MUST make the corresponding changes to the RTCP RGRS packets.

3.3. Interactions with the RTP/AVPF Feedback Profile

Use of the RTP/AVPF Feedback Profile [RFC4585] allows SSRCs to send rapid RTCP feedback requests and codec control messages. If use of the RTP/AVPF profile has been negotiated in an RTP session, members of an RTCP reporting group can send rapid RTCP feedback and codec control messages following [RFC4585] and [RFC5104], as updated by Section 5.4 of [I-D.ietf-avtcore-rtp-multi-stream], and by the following considerations.

The members of an RTCP reporting group will all see identical network conditions. Accordingly, one might therefore think that it doesn't matter which SSRC in the reporting group sends the RTP/AVPF feedback or codec control messages. There might be, however, cases where the sender of the feedback/codec control message has semantic importance, or when only a subset of the members of an RTCP reporting group might want to send RTP/AVPF feedback or a codec control message in response to a particular event. For example, an RTP video sender might choose to treat packet loss feedback received from SSRCs known to be audio receivers with less urgency than feedback that it receives from video receivers when deciding what packets to retransmit, and a multimedia receiver using reporting groups might want to choose the outgoing SSRC for feedback packets to reflect this.

Each member of an RTCP reporting group SHOULD therefore send RTP/AVPF feedback/codec control messages independently of the other members of the reporting group, to respect the semantic meaning of the message sender. The suppression rules of [RFC4585] will ensure that only a single copy of each feedback packet is (typically) generated, even if several members of a reporting group send the same feedback. When an endpoint knows that several members of its RTCP reporting group will be sending identical feedback, and that the sender of the feedback is not semantically important, then that endpoint MAY choose to send all its feedback from the reporting source and deterministically suppress feedback packets generated by the other sources in the reporting group.

It is important to note that the RTP/AVPF timing rules operate on a per-SSRC basis. Using a single reporting source to send all feedback for a reporting group will hence limit the amount of feedback that can be sent to that which can be sent by one SSRC. If this limit is a problem, then the reporting group can allow each of its members to send its own feedback, using its own SSRC.

If the RTP/AVPF feedback messages or codec control requests are sent as compound RTCP packets, then those compound RTCP packets MUST include either an RTCP RGRS packet or an RTCP SDES RGRP item, depending on whether they are sent by the reporting source or a non-

reporting source in the RTCP reporting group respectively. The contents of non-compound RTCP feedback or codec control messages are not affected by the use of RTCP reporting groups.

3.4. Interactions with RTCP Extended Report (XR) Packets

When using RTCP Extended Reports (XR) [RFC3611] with RTCP reporting groups, it is RECOMMENDED that the reporting source is used to send the RTCP XR packets. If multiple reporting sources are in use, the reporting source that sends the SR/RR packets that relate to a particular remote SSRC SHOULD send the RTCP XR reports about that SSRC. This is motivated as one commonly combine the RTCP XR metrics with the regular report block to more fully understand the situation. Receiving these blocks in different compound packets reduces their value as the measuring intervals are not synchronized in those cases.

Some RTCP XR report blocks are specific to particular types of media, and might be relevant to only some members of a reporting group. For example, it would make no sense for an SSRC that is receiving video to send a VoIP metric RTCP XR report block. Such media specific RTCP XR report blocks MUST be sent by the SSRC to which they are relevant, and MUST NOT be included in the common report sent by the reporting source. This might mean that some SSRCs send RTCP XR packets in compound RTCP packets that contain an empty RTCP SR/RR packet, and that the time period covered by the RTCP XR packet is different to that covered by the RTCP SR/RR packet. If it is important that the RTCP XR packet and RTCP SR/RR packet cover the same time period, then that source SHOULD be removed from the RTCP reporting group, and send standard RTCP packets instead.

3.5. Middlebox Considerations

Many different types of middlebox are used with RTP. RTCP reporting groups are potentially relevant to those types of RTP middlebox that have their own SSRCs and generate RTCP reports for the traffic they receive. RTP middleboxes that do not have their own SSRC, and that don't send RTCP reports on the traffic they receive, cannot use the RTCP reporting groups extension, since they generate no RTCP reports to group.

An RTP middlebox that has several SSRCs of its own can use the RTCP reporting groups extension to group the RTCP reports it generates. This can occur, for example, if a middlebox is acting as an RTP mixer for both audio and video flows that are multiplexed onto a single RTP session, where the middlebox has one SSRC for the audio mixer and one for the video mixer part, and when the middlebox wants to avoid cross reporting between audio and video.

A middlebox cannot use the RTCP reporting groups extension to group RTCP packets from the SSRCs that it is forwarding. It can, however, group the RTCP packets from the SSRCs it is forwarding into compound RTCP packets following the rules in Section 6.1 of [RFC3550] and Section 5.3 of [I-D.ietf-avtcore-rtp-multi-stream]. If the middlebox is using RTCP reporting groups for its own SSRCs, it MAY include RTCP packets from the SSRCs that it is forwarding as part of the compound RTCP packets its reporting source generates.

A middlebox that forwards RTCP SR or RR packets sent by members of a reporting group MUST forward the corresponding RTCP SDES RGRP items, as described in Section 3.2.1. A middlebox that forwards RTCP SR or RR packets sent by member of a reporting group MUST also forward the corresponding RTCP RGRS packets, as described in Section 3.2.2. Failure to forward these packets can cause compatibility problems, as described in Section 4.2.

If a middlebox rewrites SSRC values in the RTP and RTCP packets that it is forwarding, then it MUST make the corresponding changes in RTCP SDES packets containing RGRP items and in RTCP RGRS packets, to allow them to be associated with the rewritten SSRCs.

3.6. SDP Signalling for Reporting Groups

This document defines the "a=rtcp-rgrp" Session Description Protocol (SDP) [RFC4566] attribute to indicate if the session participant is capable of supporting RTCP Reporting Groups for applications that use SDP for configuration of RTP sessions. A participant that proposes the use of RTCP Reporting Groups SHALL itself support the reception of RTCP Reporting Groups.

An offering client that wishes to use RTCP Reporting Groups MUST include the attribute "a=rtcp-rgrp" in the SDP offer. If "a=rtcp-rgrp" is present in the offer SDP, the answerer that supports RTCP Reporting Groups and wishes to use it SHALL include the "a=rtcp-rgrp" attribute in the answer.

In declarative usage of SDP, such as the Real Time Streaming Protocol (RTSP) [RFC2326] and the Session Announcement Protocol (SAP) [RFC2974], the presence of the attribute indicates that the session participant MAY use RTCP Reporting Groups in its RTCP transmissions.

4. Properties of RTCP Reporting Groups

This section provides additional information on what the resulting properties are with the design specified in Section 3. The content of this section is non-normative.

4.1. Bandwidth Benefits of RTCP Reporting Groups

To understand the benefits of RTCP reporting groups, consider a scenario in which the two endpoints in a session each have a hundred sources, of which eight each are sending within any given reporting interval.

For ease of analysis, we can make the simplifying approximation that the duration of the RTCP reporting interval is equal to the total size of the RTCP packets sent during an RTCP interval, divided by the RTCP bandwidth. (This will be approximately true in scenarios where the bandwidth is not so high that the minimum RTCP interval is reached.) For further simplification, we can assume RTCP senders are following the recommendations regarding Compound RTCP Packets in [I-D.ietf-avtcore-rtp-multi-stream]; thus, the per-packet transport-layer overhead will be small relative to the RTCP data. Thus, only the actual RTCP data itself need be considered.

In a report interval in this scenario, there will, as a baseline, be 200 SDES packets, 184 RR packets, and 16 SR packets. This amounts to approximately 6.5 kB of RTCP per report interval, assuming 16-byte CNAMEs and no other SDES information.

Using the original [RFC3550] everyone-reports-on-every-sender feedback rules, each of the 184 receivers will send 16 report blocks, and each of the 16 senders will send 15. This amounts to approximately 76 kB of report block traffic per interval; 92% of RTCP traffic consists of report blocks.

If reporting groups are used, however, there is only 0.4 kB of reports per interval, with no loss of useful information. Additionally, there will be (assuming 16-byte RGRPs, and a single reporting source per reporting group) an additional 2.4 kB per cycle of RGRP SDES items and RGRS packets. Put another way, the unmodified [RFC3550] reporting interval is approximately 8.9 times longer than if reporting groups are in use.

4.2. Compatibility of RTCP Reporting Groups

The RTCP traffic generated by receivers using RTCP Reporting Groups might appear, to observers unaware of these semantics, to be generated by receivers who are experiencing a network disconnection, as the non-reporting sources appear not to be receiving a given sender at all.

This could be a potentially critical problem for such a sender using RTCP for congestion control, as such a sender might think that it is sending so much traffic that it is causing complete congestion collapse.

However, such an interpretation of the session statistics would require a fairly sophisticated RTCP analysis. Any receiver of RTCP statistics which is just interested in information about itself needs to be prepared that any given reception report might not contain information about a specific media source, because reception reports in large conferences can be round-robin.

Thus, it is unclear to what extent such backward compatibility issues would actually cause trouble in practice.

5. Security Considerations

The security considerations of [RFC3550] and [I-D.ietf-avtcore-rtp-multi-stream] apply. If the RTP/AVPF profile is in use, then the security considerations of [RFC4585] (and [RFC5104], if used) also apply. If RTCP XR is used, the security consideration of [RFC3611] and any XR report blocks used also apply.

The RTCP SDES RGRP item is vulnerable to malicious modifications unless integrity protection is used. A modification of this item's length field could cause the parsing of the RTCP packet in which it is contained to fail. Depending on the implementation, parsing of the full compound RTCP packet can also fail causing the whole packet to be discarded. A modification to the value of this SDES item would make the receiver of the report think that the sender of the report was a member of a different RTCP reporting group. This will potentially create an inconsistency, when the RGRS reports the source as being in the same reporting group as another source with another reporting group identifier. What impact on a receiver implementation such inconsistencies would have are difficult to fully predict. One case is when congestion control or other adaptation mechanisms are used, an inconsistent report can result in a media sender to reduce its bit-rate. However, a direct modification of the receiver report or a feedback message itself would be a more efficient attack, and equally costly to perform.

The new RGRS RTCP Packet type is very simple. The common RTCP packet type header shares the security risks with previous RTCP packet types. Errors or modification of the length field can cause the full compound packet to fail header validation (see Appendix A.2 in [RFC3550]) resulting in the whole compound RTCP packet being discarded. Modification of the SC or P fields would cause inconsistency when processing the RTCP packet, likely resulting in

being classified as invalid. A modification of the PT field would cause the packet being interpreted under some other packet type's rules. In such case the result might be more or less predictable but packet type specific. Modification of the SSRC of packet sender would attribute this packet to another sender. Resulting in a receiver believing the reporting group applies also for this SSRC, if it exists. If it doesn't exist, unless also corresponding modifications are done on a SR/RR packet and a SDES packet the RTCP packet SHOULD be discarded. If consistent changes are done, that could be part of a resource exhaustion attack on a receiver implementation. Modification of the "List of SSRCS for the Reporting Source(s)" would change the SSRC the receiver expect to report on behalf of this SSRC. If that SSRC exist, that could potentially change the report group used for this SSRC. A change to another reporting group belonging to another endpoint is likely detectable as there would be a mismatch between the SSRC of the packet sender's endpoint information, transport addresses, SDES CNAME etc and the corresponding information from the reporting group indicated.

In general the reporting group is providing limited impacts attacks. The most significant result from an deliberate attack would be to cause the information to be discarded or be inconsistent, including discard of all RTCP packets that are modified. This causes a lack of information at any receiver entity, possibly disregarding the endpoints participation in the session.

To protect against this type of attacks from external non trusted entities, integrity and source authentication SHOULD be applied. This can be done, for example, by using SRTP [RFC3711] with appropriate key-management, other options exist as discussed in RTP Security Options [RFC7201].

The Report Group Identifier has a potential privacy impacting properties. If this would be generated by an implementation in such a way that is long term stable or predictable, it could be used for tracking a particular end-point. Therefore it is RECOMMENDED that it be generated as a short-term persistent RGRP, following the rules for short-term persistent CNAMEs in [RFC7022]. The rest of the information revealed, i.e. the SSRCS, the size of reporting group and the number of reporting sources in a reporting group is of less sensitive nature, considering that the SSRCS and the communication would anyway be revealed without this extension. By encrypting the report group extensions the SSRC values would preserved confidential, but can still be revealed if SRTP [RFC3711] is used. The size of the reporting groups and number of reporting sources are likely determinable from analysis of the packet pattern and sizes. However, this information appears to have limited value.

6. IANA Considerations

(Note to the RFC-Editor: in the following, please replace "TBA" with the IANA-assigned value, and "XXXX" with the number of this document, then delete this note)

The IANA is requested to register one new RTCP SDES items in the "RTCP SDES Item Types" registry, as follows:

Value	Abbrev	Name	Reference
TBA	RGRP	Reporting Group Identifier	[RFCXXXX]

The definition of the RTCP SDES RGRP item is given in Section 3.2.1 of this memo.

The IANA is also requested to register one new RTCP packet type in the RTCP Control Packet Types (PT) Registry as follows:

Value	Abbrev	Name	Reference
TBA	RGRS	Reporting Group Reporting Sources	[RFCXXXX]

The definition of the RTCP RGRS packet type is given in Section 3.2.2 of this memo.

The IANA is also requested to register one new SDP attribute:

SDP Attribute ("att-field"):

Attribute name:	rtcp-rgrp
Long form:	RTCP Reporting Groups
Type of name:	att-field
Type of attribute:	Media or session level
Subject to charset:	No
Purpose:	Negotiate or configure the use of the RTCP Reporting Group Extension.
Reference:	[RFCXXXX]
Values:	See [RFCXXXX]

The definition of the "a=rtcp-rgrp" SDES attribute is given in Section 3.6 of this memo.

7. References

7.1. Normative References

- [I-D.ietf-avtcore-rtp-multi-stream]
Lennox, J., Westerlund, M., Wu, W., and C. Perkins,
"Sending Multiple Media Streams in a Single RTP Session",
draft-ietf-avtcore-rtp-multi-stream-04 (work in progress),
May 2014.
- [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.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session
Description Protocol", RFC 4566, July 2006.
- [RFC7022] Begen, A., Perkins, C., Wing, D., and E. Rescorla,
"Guidelines for Choosing RTP Control Protocol (RTCP)
Canonical Names (CNAMEs)", RFC 7022, September 2013.

7.2. Informative References

- [RFC2326] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time
Streaming Protocol (RTSP)", RFC 2326, April 1998.
- [RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session
Announcement Protocol", RFC 2974, October 2000.
- [RFC3611] Friedman, T., Caceres, R., and A. Clark, "RTP Control
Protocol Extended Reports (RTCP XR)", RFC 3611, November
2003.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K.
Norrman, "The Secure Real-time Transport Protocol (SRTP)",
RFC 3711, March 2004.
- [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, July
2006.
- [RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R.
Hakenberg, "RTP Retransmission Payload Format", RFC 4588,
July 2006.

- [RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, February 2008.
- [RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.
- [RFC6190] Wenger, S., Wang, Y.-K., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, May 2011.
- [RFC7201] Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", RFC 7201, April 2014.

Authors' Addresses

Jonathan Lennox
Vidyo, Inc.
433 Hackensack Avenue
Seventh Floor
Hackensack, NJ 07601
US

Email: jonathan@vidyo.com

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

Phone: +46 10 714 82 87
Email: magnus.westerlund@ericsson.com

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: sunseawq@huawei.com

Colin Perkins
University of Glasgow
School of Computing Science
Glasgow G12 8QQ
United Kingdom

Email: csp@csp Perkins.org

Network Working Group
Internet-Draft
Obsoletes: 5117 (if approved)
Intended status: Informational
Expires: November 28, 2014

M. Westerlund
Ericsson
S. Wenger
Vidyo
May 27, 2014

RTP Topologies
draft-ietf-avtcore-rtp-topologies-update-02

Abstract

This document discusses point to point and multi-endpoint topologies used in Real-time Transport Protocol (RTP)-based environments. In particular, centralized topologies commonly employed in the video conferencing industry are mapped to the RTP terminology.

This document is updated with additional topologies and is intended to replace RFC 5117.

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 November 28, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Definitions	3
2.1. Glossary	3
3. Topologies	4
3.1. Point to Point	4
3.2. Point to Point via Middlebox	5
3.2.1. Translators	5
3.2.2. Back to Back RTP sessions	9
3.3. Point to Multipoint Using Multicast	10
3.3.1. Any Source Multicast (ASM)	10
3.3.2. Source Specific Multicast (SSM)	11
3.3.3. SSM with Local Unicast Resources	13
3.4. Point to Multipoint Using Mesh	15
3.5. Point to Multipoint Using the RFC 3550 Translator	18
3.5.1. Relay - Transport Translator	18
3.5.2. Media Translator	19
3.6. Point to Multipoint Using the RFC 3550 Mixer Model	20
3.6.1. Media Mixing	22
3.6.2. Media Switching	25
3.7. Selective Forwarding Middlebox	27
3.8. Point to Multipoint Using Video Switching MCUs	30
3.9. Point to Multipoint Using RTCP-Terminating MCU	32
3.10. Split Component Endpoint	33
3.11. Non-Symmetric Mixer/Translators	34
3.12. Combining Topologies	35
4. Comparing Topologies	35
4.1. Topology Properties	36
4.1.1. All to All Media Transmission	36
4.1.2. Transport or Media Interoperability	37
4.1.3. Per Domain Bit-Rate Adaptation	37
4.1.4. Aggregation of Media	37
4.1.5. View of All Session Participants	38
4.1.6. Loop Detection	38
4.2. Comparison of Topologies	39
5. Security Considerations	39
6. IANA Considerations	41
7. Acknowledgements	41
8. References	41
8.1. Normative References	41
8.2. Informative References	42
Authors' Addresses	43

1. Introduction

Real-time Transport Protocol (RTP) [RFC3550] topologies describe methods for interconnecting RTP entities and their processing behavior of RTP and RTCP. This document tries to address past and existing confusion, especially with respect to terms not defined in RTP but in common use in the conversational communication industry, such as the Multipoint Control Unit or MCU.

When the Audio-Visual Profile with Feedback (AVPF) [RFC4585] was developed the main emphasis lay in the efficient support of point to point and small multipoint scenarios without centralized multipoint control. In practice, however, most multipoint conferences operate utilizing centralized units referred to as MCUs. MCUs may implement Mixer or Translator functionality (in RTP [RFC3550] terminology), and signalling support. They may also contain additional application layer functionality. This document focuses on the media transport aspects of the MCU that can be realized using RTP, as discussed below. Further considered are the properties of Mixers and Translators, and how some types of deployed MCUs deviate from these properties.

This document also codifies new multipoint architectures that have recently been introduced and which were not anticipated in RFC 5117. These architectures use scalable video coding and simulcasting, and their associated centralized units are referred to as Selective Forwarding Units (SFU). This codification provides a common information basis for future discussion and specification work.

The document's attempt to clarify and explain sections of the Real-time Transport Protocol (RTP) spec [RFC3550] is informal. It is not intended to update or change what is normatively specified within RFC 3550.

2. Definitions

2.1. Glossary

ASM: Any Source Multicast

AVPF: The Extended RTP Profile for RTCP-based Feedback

CSRC: Contributing Source

Link: The data transport to the next IP hop

Middlebox: A device that is on the Path that media travel between two Endpoints

MCU: Multipoint Control Unit

Path: The concatenation of multiple links, resulting in an end-to-end data transfer.

PtM: Point to Multipoint

PtP: Point to Point

SFU: Selective Forwarding Unit

SSM: Source-Specific Multicast

SSRC: Synchronization Source

3. Topologies

This subsection defines several topologies that are relevant for codec control but also RTP usage in other contexts. The section starts with point to point cases, with or without middleboxes. Then follows a number of different methods for establishing point to multipoint communication. These are structured around the most fundamental enabler, i.e., multicast, a mesh of connections, translators, mixers and finally MCUs and SFUs. The section ends by discussing de-composited endpoints, asymmetric middlebox behaviors and combining topologies.

The topologies may be referenced in other documents by a shortcut name, indicated by the prefix "Topo-".

For each of the RTP-defined topologies, we discuss how RTP, RTCP, and the carried media are handled. With respect to RTCP, we also discuss the handling of RTCP feedback messages as defined in [RFC4585] and [RFC5104].

3.1. Point to Point

Shortcut name: Topo-Point-to-Point

The Point to Point (PtP) topology (Figure 1) consists of two endpoints, communicating using unicast. Both RTP and RTCP traffic are conveyed endpoint-to-endpoint, using unicast traffic only (even if, in exotic cases, this unicast traffic happens to be conveyed over an IP-multicast address).

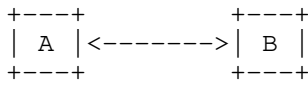


Figure 1: Point to Point

The main property of this topology is that A sends to B, and only B, while B sends to A, and only A. This avoids all complexities of handling multiple endpoints and combining the requirements stemming from them. Note that an endpoint can still use multiple RTP Synchronization Sources (SSRCs) in an RTP session. The number of RTP sessions in use between A and B can also be of any number, subject only to system level limitations like the number range of ports.

RTCP feedback messages for the indicated SSRCs are communicated directly between the endpoints. Therefore, this topology poses minimal (if any) issues for any feedback messages. For RTP sessions which use multiple SSRC per endpoint it can be relevant to implement support for cross-reporting suppression as defined in "Sending Multiple Media Streams in a Single RTP Session" [I-D.ietf-avtcore-rtp-multi-stream-optimisation].

3.2. Point to Point via Middlebox

This section discusses cases where two endpoints communicate but have one or more middleboxes involved in the RTP session.

3.2.1. Translators

Shortcut name: Topo-PtP-Translator

Two main categories of Translators can be distinguished; Transport Translators and Media translators. Both Translator types share common attributes that separate them from Mixers. For each media stream that the Translator receives, it generates an individual stream in the other domain. A translator keeps the SSRC for a stream across the translation, whereas a Mixer can select a single media stream, or send out multiple mixed media streams, but always under its own SSRC, possibly using the CSRC field to indicate the source(s) of the content. Mixers are more common in point to multipoint cases than in PtP. The reason is that in PtP use cases the primary focus is interoperability, such as transcoding to a codec the receiver supports, which can be done by a media translator.

As specified in Section 7.1 of [RFC3550], the SSRC space is common for all participants in the RTP session, independent of on which side of the Translator the session resides. Therefore, it is the responsibility of the participants to run SSRC collision detection,

and the SSRC is thus a field the Translator cannot change. Any SDES information associated with a SSRC or CSRC also needs to be forwarded between the domains for any SSRC/CSRC used in the different domains.

A Translator commonly does not use an SSRC of its own, and is not visible as an active participant in the session. One reason to have its own SSRC is when a Translator acts as a quality monitor that sends RTCP reports and therefore is required to have an SSRC. Another example is the case when a Translator is prepared to use RTCP feedback messages. This may, for example, occur in a translator configured to detect packet loss of important video packets and wants to trigger repair by the media sender, by sending feedback messages. While such feedback could use the SSRC of the target for the translator, this in turn would require translation of the targets RTCP reports to make them consistent. It may be simpler to expose an additional SSRC in the session. The only concern is endpoints failing to support the full RTP specification, thus having issues with multiple SSRCs reporting on the RTP streams sent by that endpoint.

In general, a Translator implementation should consider which RTCP feedback messages or codec-control messages it needs to understand in relation to the functionality of the Translator itself. This is completely in line with the requirement to also translate RTCP messages between the domains.

3.2.1.1. Transport Relay/Anchoring

There exist a number of different types of middleboxes that might be inserted between two RTP endpoints on the transport level, e.g., to perform changes on the IP/UDP headers, and are, therefore, basic transport translators. These middleboxes come in many variations including NAT [RFC3022] traversal by pinning the media path to a public address domain relay, network topologies where the media flow is required to pass a particular point for audit by employing relaying, or preserving privacy by hiding each peer's transport addresses to the other party. Other protocols or functionalities that provide this behavior are TURN [RFC5766] servers, Session Border Gateways and Media Processing Nodes with media anchoring functionalities.

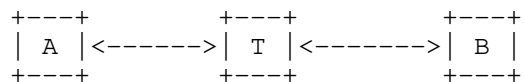


Figure 2: Point to Point with Translator

A common element in these functions is that they are normally transparent at the RTP level, i.e., they perform no changes on any RTP or RTCP packet fields and only affect the lower layers. They may affect, however, the path the RTP and RTCP packets are routed between the endpoints in the RTP session, and thereby only indirectly affect the RTP session. For this reason, one could believe that transport translator-type middleboxes do not need to be included in this document. This topology, however, can raise additional requirements in the RTP implementation and its interactions with the signalling solution. Both in signalling and in certain RTCP fields, network addresses other than those of the relay can occur since B has a different network address than the relay (T). Implementations that can not support this will also not work correctly when endpoints are subject to NAT.

The transport relay implementation also have some considerations, where security considerations are an important aspect. Source address filtering of incoming packets are usually important in relays, to prevent attackers to inject traffic into a session, which one peer will think comes from the other peer.

3.2.1.2. Transport Translator

Transport Translators (Topo-Trn-Translator) do not modify the media stream itself, but are concerned with transport parameters. Transport parameters, in the sense of this section, comprise the transport addresses (to bridge different domains such unicast to multicast) and the media packetization to allow other transport protocols to be interconnected to a session (in gateways). Of the transport Translators, this memo is primarily interested in those that use RTP on both sides, and this is assumed henceforth.

Translators that bridge between different protocol worlds need to be concerned about the mapping of the SSRC/CSRC (Contributing Source) concept to the non-RTP protocol. When designing a Translator to a non-RTP-based media transport, an important consideration is how to handle different sources and their identities. This problem space is not discussed henceforth.

The most basic transport translators that operate below the RTP level were already discussed in Section 3.2.1.1.

3.2.1.3. Media Translator

Media Translators (Topo-Media-Translator) modify the media stream itself. This process is commonly known as transcoding. The modification of the media stream can be as small as removing parts of the stream, and it can go all the way to a full decoding and re-

encoding (down to the sample level or equivalent) utilizing a different media codec. Media Translators are commonly used to connect entities without a common interoperability point in the media encoding.

Stand-alone Media Translators are rare. Most commonly, a combination of Transport and Media Translator is used to translate both the media stream and the transport aspects of a stream between two transport domains (or clouds).

When media translation occurs, the Translator's task regarding handling of RTCP traffic becomes substantially more complex. In this case, the Translator needs to rewrite B's RTCP Receiver Report before forwarding them to A. The rewriting is needed as the stream received by B is not the same stream as the other participants receive. For example, the number of packets transmitted to B may be lower than what A sends, due to the different media format and data rate. Therefore, if the Receiver Reports were forwarded without changes, the extended highest sequence number would indicate that B were substantially behind in reception, while most likely it would not be. Therefore, the Translator must translate that number to a corresponding sequence number for the stream the Translator received. Similar arguments can be made for most other fields in the RTCP Receiver Reports.

A media Translator may in some cases act on behalf of the "real" source and respond to RTCP feedback messages. This may occur, for example, when a receiver requests a bandwidth reduction, and the media Translator has not detected any congestion or other reasons for bandwidth reduction between the media source and itself. In that case, it is sensible that the media Translator reacts to the codec control messages itself, for example, by transcoding to a lower media rate.

A variant of translator behaviour worth pointing out is the one depicted in Figure 3 of an endpoint A sends a media flow to B. On the path there is a device T that on A's behalf does something with the media streams, for example adds an RTP session with FEC information for A's media streams. In this case, T needs to bind the new FEC streams to A's media stream, for example by using the same CNAME as A.

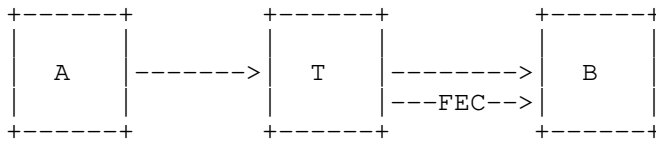


Figure 3: When De-composition is a Translator

This type of functionality where T does something with the media stream on behalf of A is covered under the media translator definition.

3.2.2. Back to Back RTP sessions

There exist middleboxes that interconnect two endpoints through themselves, but not by being part of a common RTP session. They establish instead two different RTP sessions, one between A and the middlebox and another between the middlebox and B. This topology is called Topo-Back-To-Back

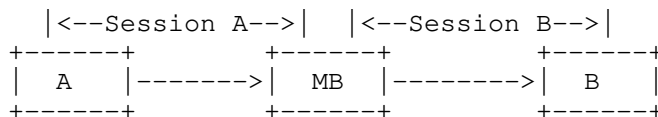


Figure 4: When De-composition is a Translator

The middlebox acts as an application-level gateway and bridges the two RTP sessions. This bridging can be as basic as forwarding the RTP payloads between the sessions, or more complex including media transcoding. The difference with the single RTP session context is the handling of the SSRCs and the other session-related identifiers, such as CNAMEs. With two different RTP sessions these can be freely changed and it becomes the middlebox's task to maintain the correct relations.

The signalling or other above-RTP level functionalities referencing RTP media streams may be what is most impacted by using two RTP sessions and changing identifiers. The structure with two RTP sessions also puts a congestion control requirement on the middlebox, because it becomes fully responsible for the media stream it sources into each of the sessions.

Adherence to congestion control can be solved locally or by bridging also statistics from the receiving endpoint. From an implementation point, however, this requires dealing with a number of inconsistencies. First, packet loss must be detected for an RTP flow sent from A to the middlebox, and that loss must be reported through

a skipped sequence number in the flow from the middlebox to B. This coupling and the resulting inconsistencies is conceptually easier to handle when considering the two flows as belonging to a single RTP session.

3.3. Point to Multipoint Using Multicast

Multicast is an IP layer functionality that is available in some networks. Two main flavors can be distinguished: Any Source Multicast (ASM) [RFC1112] where any multicast group participant can send to the group address and expect the packet to reach all group participants; and Source Specific Multicast (SSM) [RFC3569], where only a particular IP host sends to the multicast group. Both these models are discussed below in their respective sections.

3.3.1. Any Source Multicast (ASM)

Shortcut name: Topo-ASM (was Topo-Multicast)

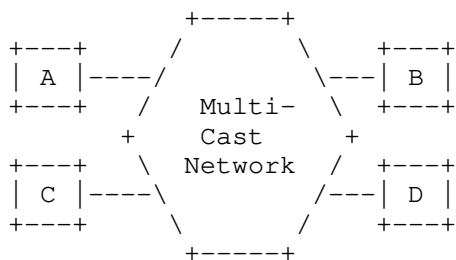


Figure 5: Point to Multipoint Using Multicast

Point to Multipoint (PtM) is defined here as using a multicast topology as a transmission model, in which traffic from any participant reaches all the other participants, except for cases such as:

- o packet loss, or
- o when a participant does not wish to receive the traffic for a specific multicast group and, therefore, has not subscribed to the IP multicast group in question. This scenario can occur, for example, where a multimedia session is distributed using two or more multicast groups and a participant is subscribed only to a subset of these sessions.

In the above context, "traffic" encompasses both RTP and RTCP traffic. The number of participants can vary between one and many, as RTP and RTCP scale to very large multicast groups (the theoretical

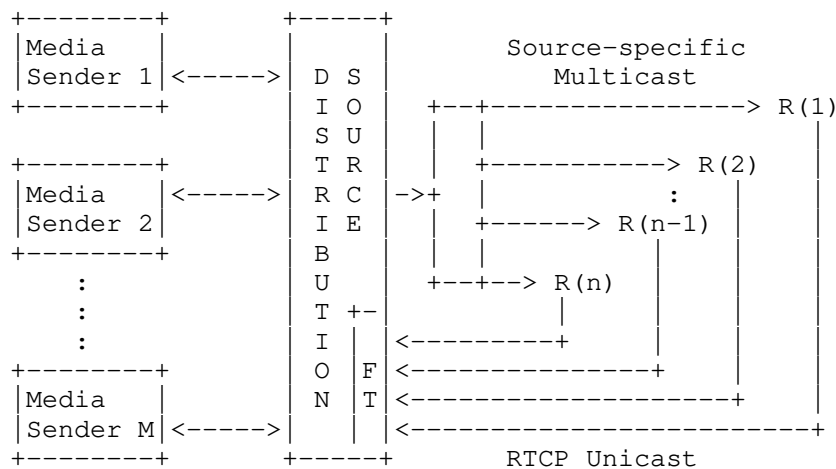
limit of the number of participants in a single RTP session is in the range of billions). The above can be realized using Any Source Multicast (ASM).

For feedback usage, it is useful to define a "small multicast group" as a group where the number of participants is so low (and other factors such as the connectivity is so good) that it allows the participants to use early or immediate feedback, as defined in AVPF [RFC4585]. Even when the environment would allow for the use of a small multicast group, some applications may still want to use the more limited options for RTCP feedback available to large multicast groups, for example when there is a likelihood that the threshold of the small multicast group (in terms of participants) may be exceeded during the lifetime of a session.

RTCP feedback messages in multicast reach, like media data, every subscriber (subject to packet losses and multicast group subscription). Therefore, the feedback suppression mechanism discussed in [RFC4585] is typically required. Each individual node needs to process every feedback message it receives, not only to determine if it is affected or if the feedback message applies only to some other participant, but also to derive timing restrictions for the sending of its own feedback messages, if any.

3.3.2. Source Specific Multicast (SSM)

In Any Source Multicast, any of the participants can send to all the other participants, by sending a packet to the multicast group. In contrast, Source Specific Multicast [RFC3569][RFC4607] refers to scenarios where only a single source (Distribution Source) can send to the multicast group, creating a topology that looks like the one below:



FT = Feedback Target

Transport from the Feedback Target to the Distribution Source is via unicast or multicast RTCP if they are not co-located.

Figure 6: Point to Multipoint using Source Specific Multicast

In the SSM topology (Figure 6) a number of RTP sources (1 to M) are allowed to send media to the SSM group. These sources send media to a dedicated distribution source, which forwards the media streams to the multicast group on behalf of the original senders. The media streams reach the Receivers (R(1) to R(n)). The Receivers' RTCP messages cannot be sent to the multicast group, as the SSM multicast group by definition has only a single IP sender. To support RTCP, an RTP extension for SSM [RFC5760] was defined. It uses unicast transmission to send RTCP from each of the receivers to one or more Feedback Targets (FT). The feedback targets relay the RTCP unmodified, or provide a summary of the participants RTCP reports towards the whole group by forwarding the RTCP traffic to the distribution source. Figure 6 only shows a single feedback target integrated in the distribution source, but for scalability the FT can be many and have responsibility for sub-groups of the receivers. For summary reports, however, there must be a single feedback aggregating all the summaries to a common message to the whole receiver group.

The RTP extension for SSM specifies how feedback (both reception information and specific feedback events) are handled. The more general problems associated with the use of multicast, where everyone receives what the distribution source sends needs to be accounted for.

Aforementioned situation results in common behavior for RTP multicast:

1. Multicast applications often use a group of RTP sessions, not one. Each endpoint needs to be a member of most or all of these RTP sessions in order to perform well.
2. Within each RTP session, the number of media sinks is likely to be much larger than the number of RTP sources.
3. Multicast applications need signalling functions to identify the relationships between RTP sessions.
4. Multicast applications need signalling functions to identify the relationships between SSRCs in different RTP sessions.

All multicast configurations share a signalling requirement: all of the participants need to have the same RTP and payload type configuration. Otherwise, A could, for example, be using payload type 97 to identify the video codec H.264, while B would identify it as MPEG-2.

Security solutions for this type of group communications are also challenging. First, the key-management and the security protocol must support group communication. Source authentication becomes more difficult and requires special solutions. For more discussion on this please review Options for Securing RTP Sessions [RFC7201].

3.3.3. SSM with Local Unicast Resources

[RFC6285] "Unicast-Based Rapid Acquisition of Multicast RTP Sessions" results in additional extensions to SSM Topology.

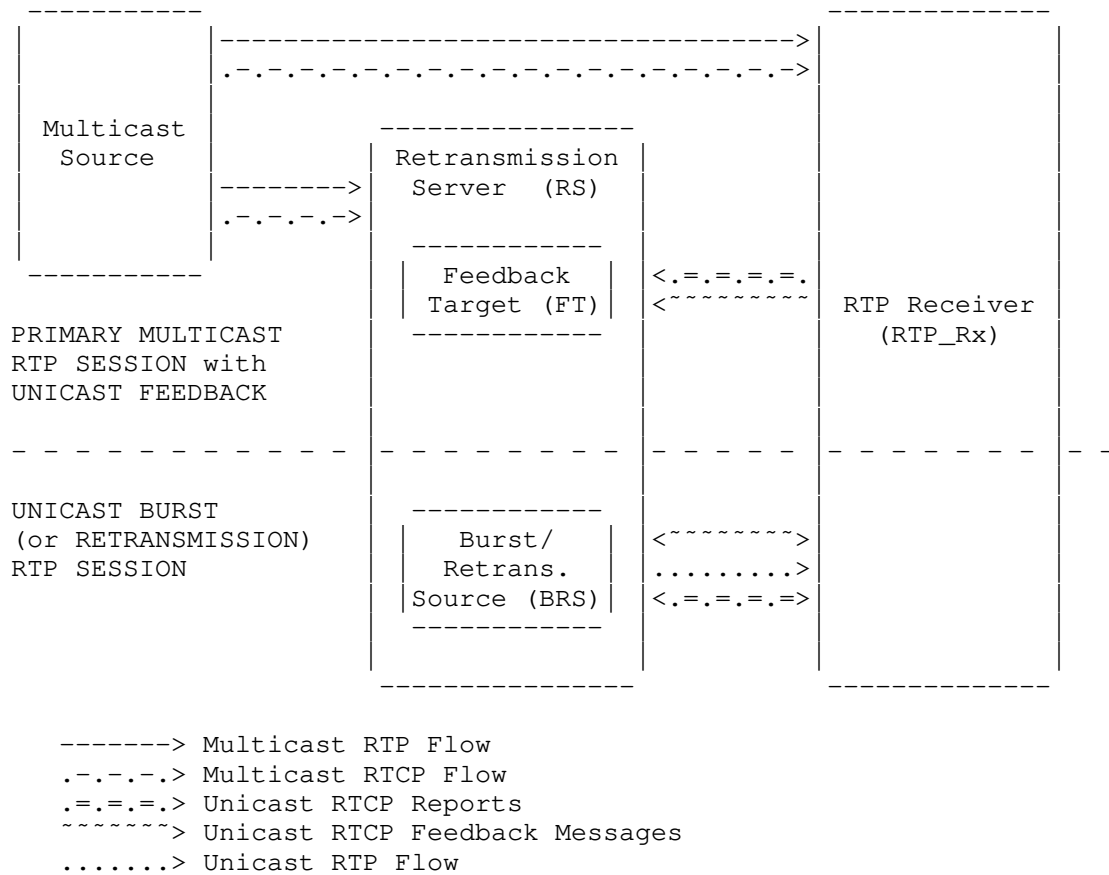


Figure 7

The Rapid acquisition extension allows an endpoint joining an SSM multicast session to request media starting with the last sync-point (from where media can be decoded without requiring context established by the decoding of prior packets) to be sent at high speed until such time where, after decoding of these burst-delivered media packets, the correct media timing is established, i.e. media packets are received within adequate buffer intervals for this application. This is accomplished by first establishing a unicast PtP RTP session between the Burst/Retransmission Source (BRS, Figure 7) and the RTP Receiver. The unicast session is used to transmit cached packets from the multicast group at higher than normal speed in order to synchronize the receiver to the ongoing multicast packet flow. Once the RTP receiver and its decoder have caught up with the multicast session's current delivery, the receiver switches over to receiving directly from the multicast group. The

(still existing) PtP RTP session is, in many deployed applications, be used as a repair channel, i.e., for RTP Retransmission traffic of those packets that were not received from the multicast group.

3.4. Point to Multipoint Using Mesh

Shortcut name: Topo-Mesh

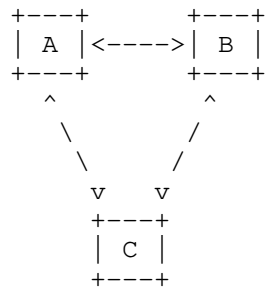


Figure 8: Point to Multi-Point using Mesh

Based on the RTP session definition, it is clearly possible to have a joint RTP session over multiple unicast transport flows like the above joint three endpoint session. In this case, A needs to send its' media streams and RTCP packets to both B and C over their respective transport flows. As long as all participants do the same, everyone will have a joint view of the RTP session.

This does not create any additional requirements beyond the need to have multiple transport flows associated with a single RTP session. Note that an endpoint may use a single local port to receive all these transport flows, or it might have separate local reception ports for each of the endpoints.

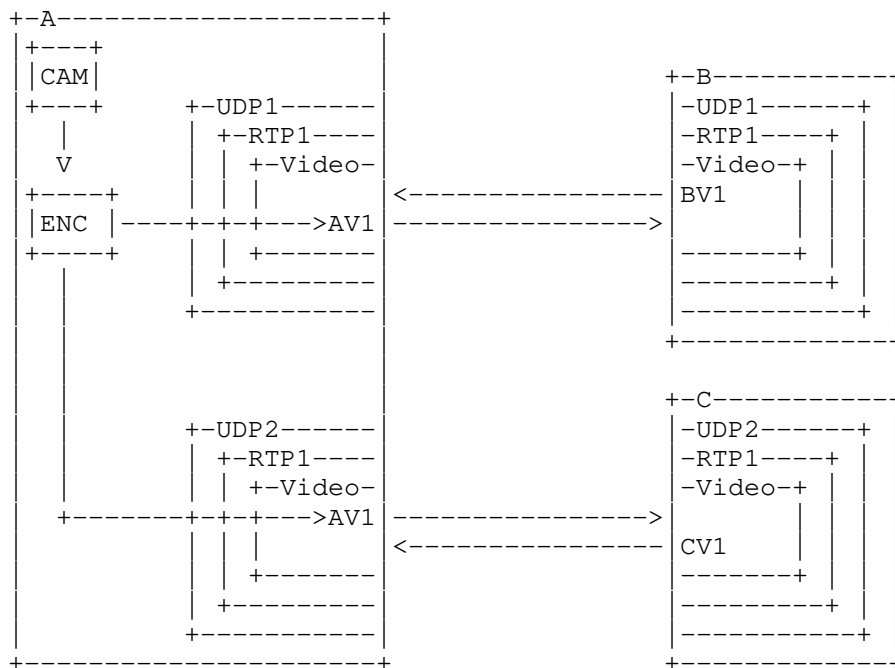


Figure 9: An Multi-unicast Mesh with a joint RTP session

A joint RTP session from A's perspective for the Mesh depicted in Figure 8 with a joint RTP session have multiple transport flows, here enumerated as UDP1 and UDP2. However, there is only one RTP session (RTP1). The media source (CAM) is encoded and transmitted over the SSRC (AV1) across both transport layers. However, as this is a joint RTP session, the two streams must be the same. Thus, an congestion control adaptation needed for the paths A to B and A to C needs to use the most restricting path's properties.

An alternative structure for establishing the above topology is to use independent RTP sessions between each pair of peers, i.e., three different RTP sessions. In some scenarios, the same RTP media stream may be sent from transmitting endpoint, however it also supports local adaptation taking place in one or more of the RTP media streams, rendering them non-identical.

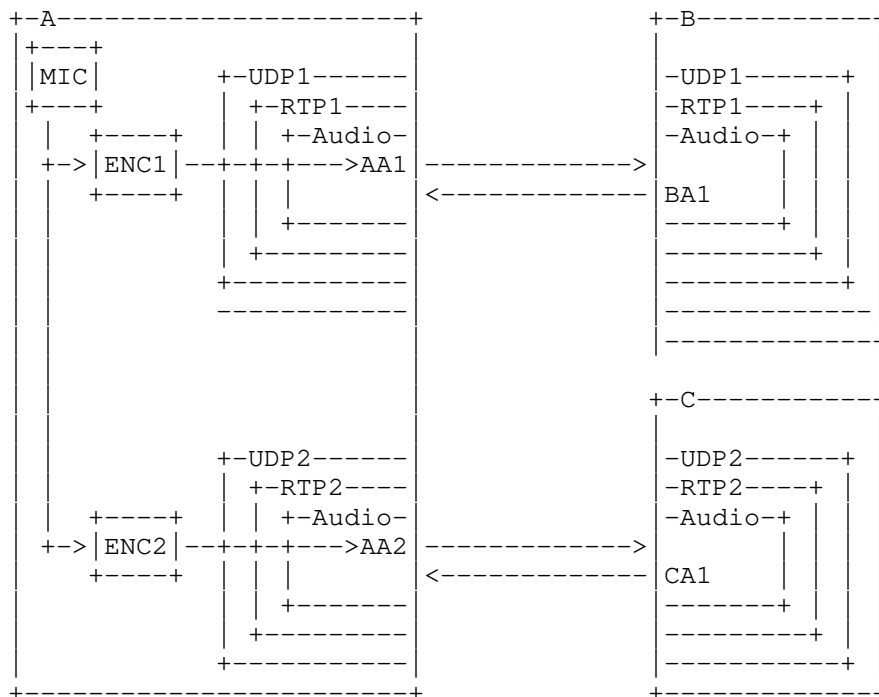


Figure 10: An Multi-unicast Mesh with independent RTP session

Lets review the topology when independent RTP sessions are used, from A's perspective in Figure 8 by considering both how the media is handled and the RTP sessions that are set-up in Figure 10. A's microphone is captured and the digital audio can then be feed into two different encoder instances, as each being associated with two independent RTP sessions (RTP1 and RTP2). The SSRCs (AA1 and AA2) in each RTP session will be completely independent and the media bit-rate produced by the encoders can also be tuned differently to address any congestion control requirements differing for the paths A to B compared to A to C.

From a topologies viewpoint, an important difference exists in the behavior around RTCP. First, when a single RTP session spans all three endpoints and their connecting flows, a common RTCP bandwidth is calculated and used for this single joint session. In contrast, when there are multiple independent RTP sessions, each RTP session has its local RTCP bandwidth allocation.

Further, when multiple sessions are used, endpoints not directly involved in a session, do not have any awareness of the conditions in those sessions. For example, in the case of the three endpoint

configuration in Figure 8, endpoint A has no awareness of the conditions occurring in the session between endpoints B and C (whereas, if a single RTP session were used, it would have such awareness).

Loop detection is also affected. With independent RTP sessions, the SSRC/CSRC cannot be used to determine when an endpoint receives its own media stream, or a mixed media stream including its own media stream (a condition known as a loop). The identification of loops and, in most cases, their avoidance, has to be achieved by other means, for example through signaling or the use of an RTP external name space binding SSRC/CSRC among any communicating RTP sessions in the mesh.

3.5. Point to Multipoint Using the RFC 3550 Translator

This section discusses some additional usages related to point to multipoint of Translators compared to the point to point only cases in Section 3.2.1.

3.5.1. Relay - Transport Translator

Shortcut name: Topo-PtM-Trn-Translator

This section discusses Transport Translator only usages to enable multipoint sessions.

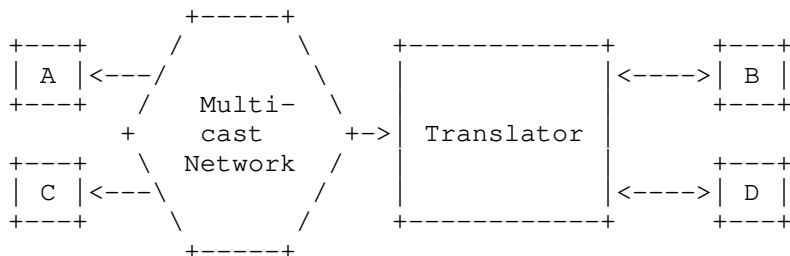


Figure 11: Point to Multipoint Using Multicast

Figure 11 depicts an example of a Transport Translator performing at least IP address translation. It allows the (non-multicast-capable) participants B and D to take part in an any source multicast session by having the Translator forward their unicast traffic to the multicast addresses in use, and vice versa. It must also forward B's traffic to D, and vice versa, to provide each of B and D with a complete view of the session.

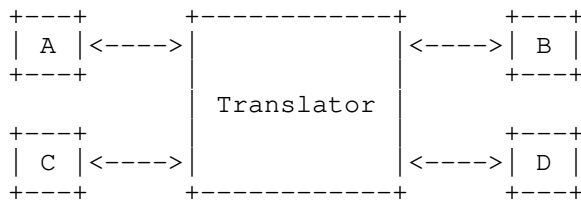


Figure 12: RTP Translator (Relay) with Only Unicast Paths

Another Translator scenario is depicted in Figure 12. The Translator in this case connects multiple users of a conference through unicast. This can be implemented using a very simple transport Translator which, in this document, is called a relay. The relay forwards all traffic it receives, both RTP and RTCP, to all other participants. In doing so, a multicast network is emulated without relying on a multicast-capable network infrastructure.

For RTCP feedback this results in a similar set of considerations to those described in the ASM RTP topology. It also puts some additional signalling requirements onto the session establishment; for example, a common configuration of RTP payload types is required.

Transport translators and relays should always consider doing source address filtering, to prevent attackers to inject traffic using the listening ports on the translator. The translator can however go one step further, and especially if explicit SSRC signalling is used, prevent other session participants to send SSRCs that are used by other participants in the session. This can improve the security properties of the session, despite the use of group keys that on cryptographic level allows anyone to impersonate another in the same RTP session.

A Translator that doesn't change the RTP/RTCP packets content can be operated without the requiring the translator to have access to the security contexts used to protect the RTP/RTCP traffic between the participants.

3.5.2. Media Translator

In the context of multipoint communications a Media Translator is not providing new mechanisms to establish a multipoint session. It is more of an enabler, or facilitator, that ensures one or some sub-set of session participants can participate in the session.

If B in Figure 11 were behind a limited network path, the Translator may perform media transcoding to allow the traffic received from the other participants to reach B without overloading the path. This

transcoding can help the other participants in the Multicast part of the session, by not requiring the quality transmitted by A to be lowered to the bitrates that B is actually capable of receiving.

3.6. Point to Multipoint Using the RFC 3550 Mixer Model

Shortcut name: Topo-Mixer

A Mixer is a middlebox that aggregates multiple RTP streams that are part of a session by generating one or more new RTP streams and, in most cases, by manipulating the media data. One common application for a Mixer is to allow a participant to receive a session with a reduced amount of resources.

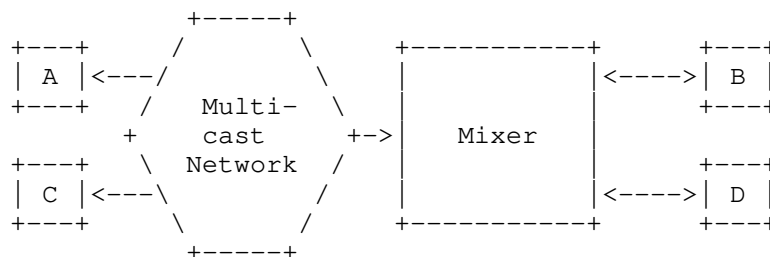


Figure 13: Point to Multipoint Using the RFC 3550 Mixer Model

A Mixer can be viewed as a device terminating the media streams received from other session participants. Using the media data from the received media streams, a Mixer generates media streams that are sent to the session participant.

The content that the Mixer provides is the mixed aggregate of what the Mixer receives over the PtP or PtM paths, which are part of the same conference session.

The Mixer is the content source, as it mixes the content (often in the uncompressed domain) and then encodes it for transmission to a participant. The CSRC Count (CC) and CSRC fields in the RTP header can be used to indicate the contributors to the newly generated stream. The SSRCs of the to-be-mixed streams on the Mixer input appear as the CSRCs at the Mixer output. That output stream uses a unique SSRC that identifies the Mixer's stream. The CSRC should be forwarded between the different conference participants to allow for loop detection and identification of sources that are part of the global session. Note that Section 7.1 of RFC 3550 requires the SSRC space to be shared between domains for these reasons. This also implies that any SDES information normally needs to be forwarded across the mixer.

The Mixer is responsible for generating RTCP packets in accordance with its role. It is a receiver and should therefore send receiver reports for the media streams it receives. In its role as a media sender, it should also generate sender reports for those media streams it sends. As specified in Section 7.3 of RFC 3550, a Mixer must not forward RTCP unaltered between the two domains.

The Mixer depicted in Figure 13 is involved in three domains that need to be separated: the any source multicast network (including participants A and C), participant B, and participant D. Assuming all four participants in the conference are interested in receiving content from each other participant, the Mixer produces different mixed streams for B and D, as the one to B may contain content received from D, and vice versa. However, the Mixer may only need one SSRC per media type in each domain where it is the receiving entity and transmitter of mixed content.

In the multicast domain, a Mixer still needs to provide a mixed view of the other domains. This makes the Mixer simpler to implement and avoids any issues with advanced RTCP handling or loop detection, which would be problematic if the Mixer were providing non-symmetric behavior. Please see Section 3.11 for more discussion on this topic. The mixing operation, however, in each domain could potentially be different.

A Mixer is responsible for receiving RTCP feedback messages and handling them appropriately. The definition of "appropriate" depends on the message itself and the context. In some cases, the reception of a codec-control message by the Mixer may result in the generation and transmission of RTCP feedback messages by the Mixer to the participants in the other domain(s). In other cases, a message is handled by the Mixer itself and therefore not forwarded to any other domain.

When replacing the multicast network in Figure 13 (to the left of the Mixer) with individual unicast paths as depicted in Figure 14, the Mixer model is very similar to the one discussed in Section 3.9 below. Please see the discussion in Section 3.9 about the differences between these two models.

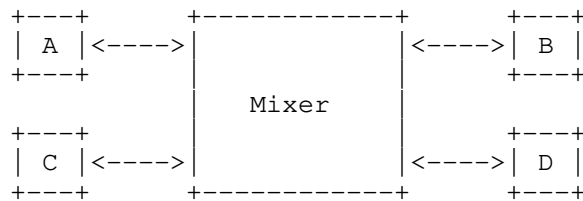


Figure 14: RTP Mixer with Only Unicast Paths

We now discuss in more detail the different mixing operations that a mixer can perform and how they can affect RTP and RTCP behavior.

3.6.1. Media Mixing

The media mixing mixer is likely the one that most think of when they hear the term "mixer". Its basic mode of operation is that it receives media streams from several participants and selects the stream(s) to be included in a media-domain mix. The selection can be through static configuration or by dynamic, content dependent means such as voice activation. The mixer then creates a single outgoing stream from this mix.

The most commonly deployed media mixer is probably the audio mixer, used in voice conferencing, where the output consists of a mixture of all the input streams; this needs minimal signalling to be successfully set up. Audio mixing is relatively straightforward and commonly possible for a reasonable number of participants. Assume, for example, that one wants to mix N streams from different participants. The mixer needs to decode those N streams, typically into the sample domain, and then produce N or $N+1$ mixes. Different mixes are needed so that each contributing source gets a mix of all other sources except its own, as this would result in an echo. When N is lower than the number of all participants one may produce a Mix of all N streams for the group that are currently not included in the mix, thus $N+1$ mixes. These audio streams are then encoded again, RTP packetized and sent out. In many cases, audio level normalization is also required before the actual mixing process.

In video, the term "mixing" has a different interpretation than audio. It is commonly used to refer to the process of spatially combining contributed video streams is known as "tiling". The reconstructed, appropriately scaled down videos can be spatially arranged in a set of tiles, each tile containing the video from a participant. Tiles can be of different sizes, so that, for example, a particularly important participant, or the loudest speaker, is being shown on in larger tile than other participants. A self-view picture can be included in the tiling, which can either be locally

produced or be a feedback from a received and reconstructed video image. Such remote loopback allows for confidence monitoring, i.e., it enables the participant to see himself/herself just as other participants see him/her. The tiling normally operates on reconstructed video in the sample domain. The tiled image is encoded, packetized, and sent by the mixer. It is possible that a middlebox with media mixing duties contains only a single mixer of the aforementioned type, in which case all participants necessarily see the same tiled video, even if it is being sent over different RTP streams. More common, however, are mixing arrangement where an individual mixer is available for each outgoing port of the middlebox, allowing individual compositions for each participant (a feature referred to as personalized layout).

One problem with media mixing is that it consumes both large amount of media processing (for the actual mixing process in the uncompressed domain) and encoding resources (for the encoding of the mixed signal). Another problem is the quality degradation created by decoding and re-encoding the media that is encapsulated in the RTP media stream, which is the result of the lossy nature of most commonly used media codecs. A third problem is the latency introduced by the media mixing, which can be substantial and annoyingly noticeable in case of video, or in case of audio if that mixed audio is lip-synchronized with high latency video. The advantage of media mixing is that it is straightforward for the clients to handle the single media stream (which includes the mixed aggregate of many sources), as they don't need to handle multiple decodings, local mixing and composition. In fact, mixers were introduced in pre-RTP times so that legacy, single stream receiving endpoints could successfully participate in what a user would recognize as a multiparty video conference.

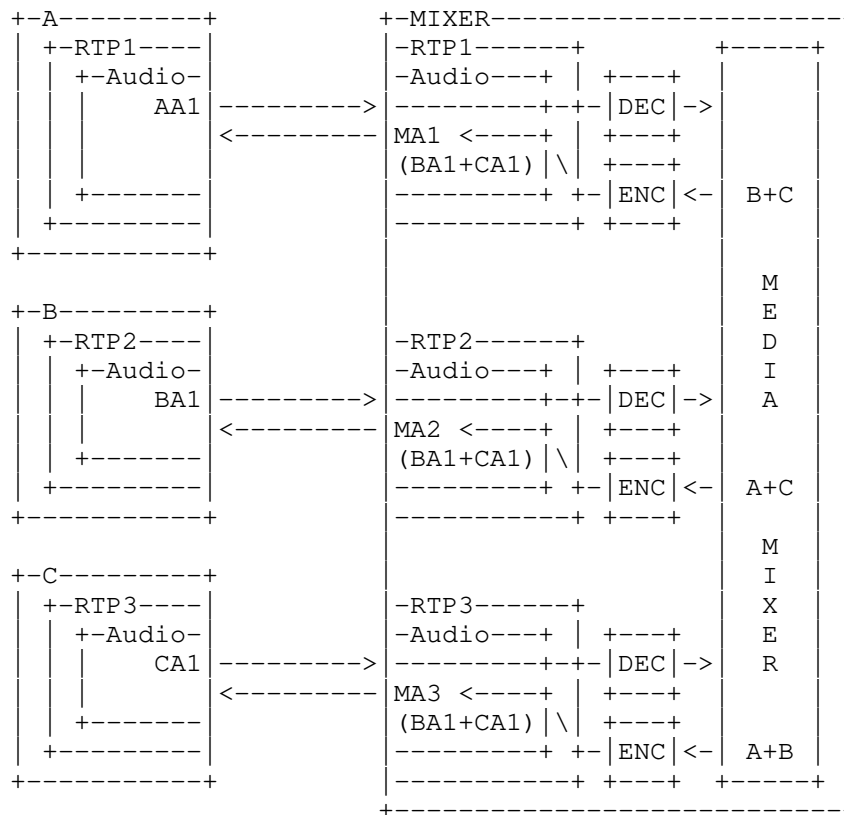


Figure 15: Session and SSRC details for Media Mixer

From an RTP perspective media mixing can be a very simple process, as can be seen in Figure 15. The mixer presents one SSRC towards the receiving client, e.g., MA1 to Peer A, where the associated stream is the media mix of the other participants. As each peer, in this example, receives a different version of a mix from the mixer, there is no actual relation between the different RTP sessions in terms of actual media or transport level information. There are, however, common relationships between RTP1-RTP3, namely SSRC space and identity information. When A receives the MA1 stream which is a combination of BA1 and CA1 streams, the mixer may include CSRC information in the MA1 stream to identify the contributing source BA1 and CA1, allowing the receiver to identify the contributing sources even if this were not possible through the media itself or through other signaling means.

The CSRC has, in turn, utility in RTP extensions, like the Mixer to Client audio levels RTP header extension [RFC6465]. If the SSRCs

from the endpoint to mixer paths are used as CSRCs in another RTP session, then RTP1, RTP2 and RTP3 become one joint session as they have a common SSRC space. At this stage, the mixer also needs to consider which RTCP information it needs to expose in the different paths. In the above scenario, a mixer would normally expose nothing more than the Source Description (SDES) information and RTCP BYE for a CSRC leaving the session. The main goal would be to enable the correct binding against the application logic and other information sources. This also enables loop detection in the RTP session.

3.6.2. Media Switching

Media switching mixers are used from limited functionality scenarios where no, or only very limited, concurrent presentation of multiple sources is required by the application to more complex multi-stream usages with receiver mixing or tiling, including combined with simulcast and/or scalability between source and mixer. An RTP Mixer based on media switching avoids the media decoding and encoding operations in the mixer, as it conceptually forwards the encoded media stream as it was being sent to the mixer. It does not avoid, however, the decryption and re-encryption cycle as it rewrites RTP headers. Forwarding media (in contrast to reconstructing-mixing-encoding media) reduces the amount of computational resources needed in the mixer and increases the media quality (both in terms of fidelity and reduced latency).

A media switching mixer maintains a pool of SSRCs representing conceptual or functional streams that the mixer can produce. These streams are created by selecting media from one of the RTP media streams received by the mixer and forwarded to the peer using the mixer's own SSRCs. The mixer can switch between available sources if that is required by the concept for the source, like the currently active speaker. Note that the mixer, in most cases, still needs to perform a certain amount of media processing, as many media formats do not allow to "tune into" the stream at arbitrary points of their bitstream.

To achieve a coherent RTP media stream from the mixer's SSRC, the mixer needs to rewrite the incoming RTP packet's header. First the SSRC field must be set to the value of the Mixer's SSRC. Second, the sequence number must be the next in the sequence of outgoing packets it sent. Third, the RTP timestamp value needs to be adjusted using an offset that changes each time one switches media source. Finally, depending on the negotiation of the RTP payload type, the value representing this particular RTP payload configuration may have to be changed if the different endpoint mixer paths have not arrived on the same numbering for a given configuration. This also requires that

the different endpoints support a common set of codecs, otherwise media transcoding for codec compatibility would still be required.

We now consider the operation of a media switching mixer that supports a video conference with six participants (A-F) where the two most recent speakers in the conference are shown to each participant. The mixer has thus two SSRCs sending video to each peer, and each peer is capable of locally handling two video streams simultaneously.

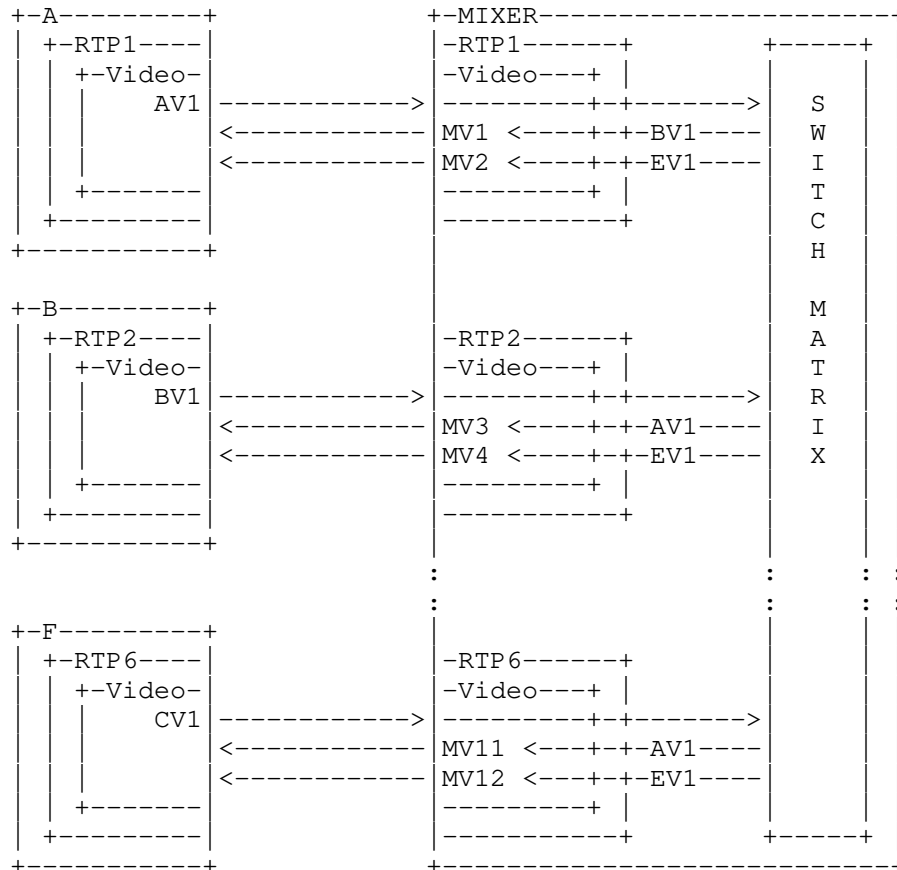


Figure 16: Media Switching RTP Mixer

The Media Switching RTP mixer can, similarly to the Media Mixing Mixer, reduce the bit-rate required for media transmission towards the different peers by selecting and forwarding only a sub-set of RTP media streams it receives from the conference participants. In cases the mixer receives simulcast transmissions or a scalable encoding of

the media source, the mixer has more degrees of freedom to select streams or sub-sets of stream to forward to a receiver, both based on transport or client restrictions as well as application logic.

To ensure that a media receiver can correctly decode the RTP media stream after a switch, a codec that uses temporal prediction needs to start its decoding from independent refresh points, or similar points in the bitstream. For some codecs, for example frame based speech and audio codecs, this is easily achieved by starting the decoding at RTP packet boundaries, as each packet boundary provides a refresh point (assuming proper packetization on the encoder side). For other codecs, particularly in video, refresh points are less common in the bitstream or may not be present at all without an explicit request to the respective encoder. The Full Intra Request [RFC5104] RTCP codec control message has been defined for this purpose.

In this type of mixer one could consider to fully terminate the RTP sessions between the different endpoint and mixer paths. The same arguments and considerations as discussed in Section 3.9 need to be taken into consideration and apply here.

3.7. Selective Forwarding Middlebox

Another method for handling media in the RTP mixer is to "project", or make available, all potential RTP sources (SSRCs) into a per-endpoint, independent RTP session. The middlebox can select which of the potential sources that are currently actively transmitting media will be sent to each of the endpoints. This is similar to the media switching Mixer but has some important differences in RTP details.

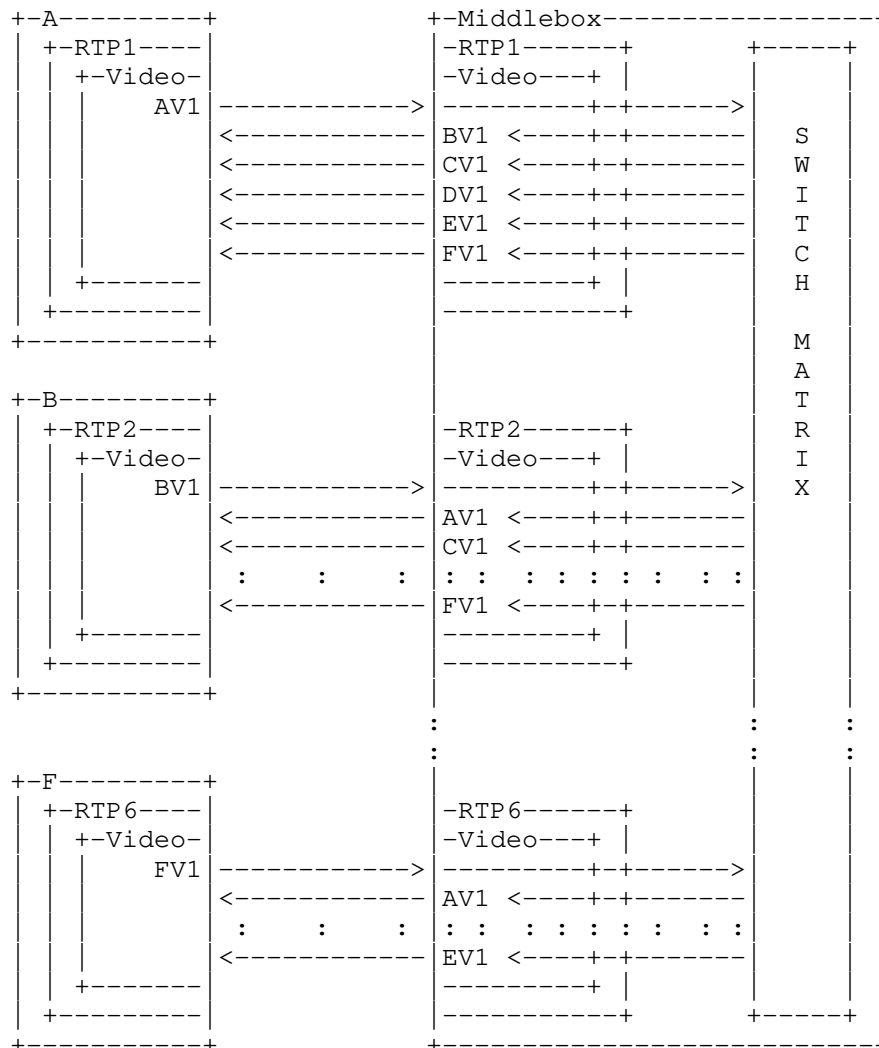


Figure 17: Selective Forwarding Middlebox

In the six participant conference depicted above in (Figure 17) one can see that end-point A is aware of five incoming SSRCs, BV1-FV1. If this middlebox intends to have a similar behavior as in Section 3.6.2 where the mixer provides the end-points with the two latest speaking end-points, then only two out of these five SSRCs need concurrently transmit media to A. As the middlebox selects the source in the different RTP sessions that transmit media to the end-points, each RTP media stream requires some rewriting of RTP header fields when being projected from one session into another. In

particular, the sequence number needs to be consecutively incremented based on the packet actually being transmitted in each RTP session. Therefore, the RTP sequence number offset will change each time a source is turned on in a RTP session. The timestamp (possibly offset) stays the same.

As the RTP sessions are independent, the SSRC numbers used can also be handled independently, thereby bypassing the requirement for SSRC collision detection and avoidance. On the other hand, tools such as remapping tables between the RTP sessions are required. For example, the stream that is being sent by endpoint B to the middlebox (BV1) may use an SSRC value of 12345678. When that media stream is sent to endpoint F by the middlebox, it can use any SSRC value, e.g. 87654321. As a result, each endpoint may have a different view of the application usage of a particular SSRC. Any RTP level identity information, such as SDES items also needs to update the SSRC referenced, if the included SDES items are intended to be global. Thus the application must not use SSRC as references to RTP media streams when communicating with other peers directly. This also affects loop detection which will fail to work, as there is no common namespace and identities across the different legs in the communication session on RTP level. Instead this responsibility falls onto higher layers.

The middlebox is also responsible to receive any RTCP codec control requests coming from an end-point, and decide if it can act on the request locally or needs to translate the request into the RTP session that contains the media source. Both end-points and the middlebox need to implement conference related codec control functionalities to provide a good experience. Commonly used are Full Intra Request to request from the media source to provide switching points between the sources, and Temporary Maximum Media Bit-rate Request (TMMBR) to enable the middlebox to aggregate congestion control responses towards the media source so to enable it to adjust its bit-rate (obviously only in case the limitation is not in the source to middlebox link).

The selective forwarding middlebox has been introduced in recently developed videoconferencing systems in conjunction with, and to capitalize on, scalable video coding as well as simulcasting. An example of scalable video coding is Annex G of H.264, but other codecs, including H.264 AVC and VP8 also exhibit scalability, albeit only in the temporal dimension. In both scalable coding and simulcast cases the video signal is represented by a set of two or more bitstreams, providing a corresponding number of distinct fidelity points. The middlebox selects which parts of a scalable bitstream (or which bitstream, in the case of simulcasting) to forward to each of the receiving endpoints. The decision may be

driven by a number of factors, such as available bit rate, desired layout, etc. Contrary to transcoding MCUs, these "Selective Forwarding Units" (SFUs) have extremely low delay, and provide features that are typically associated with high-end systems (personalized layout, error localization) without any signal processing at the middlebox. They are also capable of scaling to a large number of concurrent users, and--due to their very low delay--can also be cascaded.

This version of the middlebox also puts different requirements on the endpoint when it comes to decoder instances and handling of the RTP media streams providing media. As each projected SSRC can, at any time, provide media, the endpoint either needs to be able to handle as many decoder instances as the middlebox received, or have efficient switching of decoder contexts in a more limited set of actual decoder instances to cope with the switches. The application also gets more responsibility to update how the media provided is to be presented to the user.

Note that this topology could potentially be seen as a media translator which include an on/off logic as part of its media translation. The main difference would be a common global SSRC space in the case of the Media Translator and the mapped one used in the above. It also has mixer aspects, as the streams it provides are not basically translated version, but instead they have conceptual property assigned to them. Thus this topology appears to be some hybrid between the translator and mixer model.

The differences between selective forwarding middlebox and a switching mixer (Section 3.6.2) are minor, and they share most properties. The above requirement on having a large number of decoding instances or requiring efficient switching of decoder contexts, are one point of difference. The other is how the identification is performed, where the Mixer uses CSRC to provide info what is included in a particular RTP packet stream that represent a particular concept. Selective forwarding gets the source information through the SSRC, and instead have to use other mechanism to make clear the streams current purpose.

3.8. Point to Multipoint Using Video Switching MCUs

Shortcut name: Topo-Video-switch-MCU

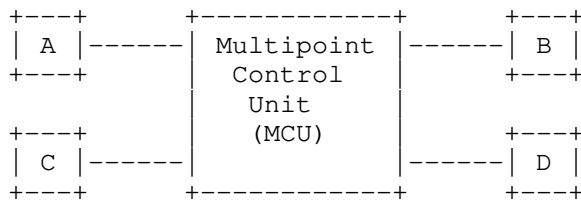


Figure 18: Point to Multipoint Using a Video Switching MCU

This PtM topology was popular in early implementations of multipoint videoconferencing systems due to its simplicity, and the corresponding middlebox design has been known as a "video switching MCU". The more complex RTCP-terminating MCUs, discussed in the next section, became the norm, however, when technology allowed implementations at acceptable costs.

A video switching MCU forwards to a participant a single media stream, selected from the available streams. The criteria for selection are often based on voice activity in the audio-visual conference, but other conference management mechanisms (like presentation mode or explicit floor control) are known to exist as well.

The video switching MCU may also perform media translation to modify the content in bit-rate, encoding, or resolution. However, it still may indicate the original sender of the content through the SSRC. In this case, the values of the CC and CSRC fields are retained.

If not terminating RTP, the RTCP Sender Reports are forwarded for the currently selected sender. All RTCP Receiver Reports are freely forwarded between the participants. In addition, the MCU may also originate RTCP control traffic in order to control the session and/or report on status from its viewpoint.

The video switching MCU has most of the attributes of a Translator. However, its stream selection is a mixing behavior. This behavior has some RTP and RTCP issues associated with it. The suppression of all but one media stream results in most participants seeing only a subset of the sent media streams at any given time, often a single stream per conference. Therefore, RTCP Receiver Reports only report on these streams. Consequently, the media senders that are not currently forwarded receive a view of the session that indicates their media streams disappear somewhere en route. This makes the use of RTCP for congestion control, or any type of quality reporting, very problematic.

To avoid the aforementioned issues, the MCU needs to implement two features. First, it needs to act as a Mixer (see Section 3.6) and forward the selected media stream under its own SSRC and with the appropriate CSRC values. Second, the MCU needs to modify the RTCP RRs it forwards between the domains. As a result, it is recommended that one implement a centralized video switching conference using a Mixer according to RFC 3550, instead of the shortcut implementation described here.

3.9. Point to Multipoint Using RTCP-Terminating MCU

Shortcut name: Topo-RTCP-terminating-MCU

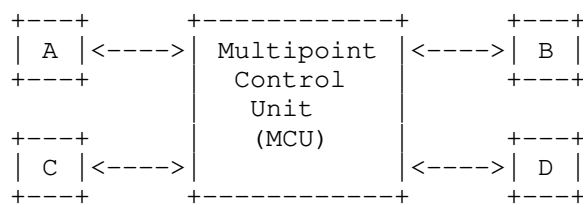


Figure 19: Point to Multipoint Using Content Modifying MCUs

In this PtM scenario, each participant runs an RTP point-to-point session between itself and the MCU. This is a very commonly deployed topology in multipoint video conferencing. The content that the MCU provides to each participant is either:

- a. a selection of the content received from the other participants, or
- b. the mixed aggregate of what the MCU receives from the other PtP paths, which are part of the same conference session.

In case (a), the MCU may modify the content in terms of bit-rate, encoding format, or resolution. No explicit RTP mechanism is used to establish the relationship between the original media sender and the version the MCU sends. In other words, the outgoing sessions typically use a different SSRC, and may well use a different payload type (PT), even if this different PT happens to be mapped to the same media type. This is a result of the individually negotiated session for each participant.

In case (b), the MCU is the content source as it mixes the content and then encodes it for transmission to a participant. According to RTP [RFC3550], the SSRC of the contributors are to be signalled using the CSRC/CC mechanism. In practice, today, most deployed MCUs do not implement this feature. Instead, the identification of the

participants whose content is included in the Mixer's output is not indicated through any explicit RTP mechanism. That is, most deployed MCUs set the CSRC Count (CC) field in the RTP header to zero, thereby indicating no available CSRC information, even if they could identify the content sources as suggested in RTP.

The main feature that sets this topology apart from what RFC 3550 describes is the breaking of the common RTP session across the centralized device, such as the MCU. This results in the loss of explicit RTP-level indication of all participants. If one were using the mechanisms available in RTP and RTCP to signal this explicitly, the topology would follow the approach of an RTP Mixer. The lack of explicit indication has at least the following potential problems:

1. Loop detection cannot be performed on the RTP level. When carelessly connecting two misconfigured MCUs, a loop could be generated.
2. There is no information about active media senders available in the RTP packet. As this information is missing, receivers cannot use it. It also deprives the client of information related to currently active senders in a machine-usable way, thus preventing clients from indicating currently active speakers in user interfaces, etc.

Note that deployed MCUs (and endpoints) rely on signalling layer mechanisms for the identification of the contributing sources, for example, a SIP conferencing package [RFC4575]. This alleviates, to some extent, the aforementioned issues resulting from ignoring RTP's CSRC mechanism.

3.10. Split Component Endpoint

Shortcut name: Topo-Split-Endpoint

The implementation of an application may desire to send a subset of the application's data to each of multiple devices, each with its own network address. A very basic use case for this would be to separate audio and video processing for a particular endpoint into different components. For example, in a video conference room system the endpoint could be considered as being composed of one device handling the audio and another handling the video, interconnected by some control functions allowing them to behave as a single endpoint in all aspects except for transport as depicted in Figure 20.

Which decomposition scheme is possible is highly dependent on the RTP session usage. It is not really feasible to decompose one logical end-point into two different transport nodes in one RTP session. A

third party monitor would report such an attempt as two entities being two different end-points with a CNAME collision. As a result, a fully RTP conformant de-composited endpoint is one where the different decomposed parts use separate RTP sessions to send and/or receive media streams intended for them.

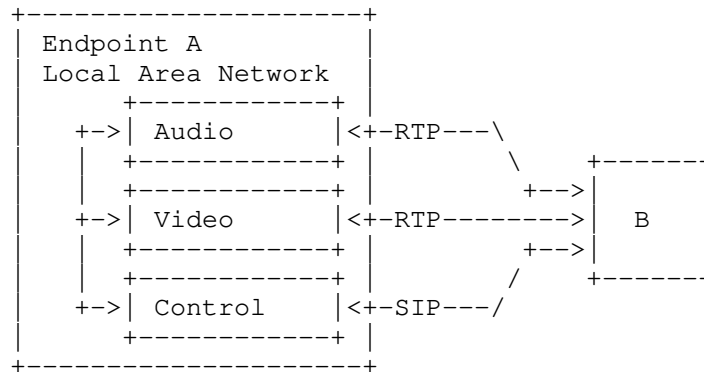


Figure 20: Split Component Endpoint

In the above usage, let us assume that the different RTP sessions are used for audio and video. The audio and video parts, however, use a common CNAME and also have a common clock to ensure that synchronization and clock drift handling works, despite the fact that the components are separated. Also, RTCP handling works correctly as long as only one part of the split endpoint is part of each RTP session. That way any differences in the path between A's audio entity and B and A's video and B are related to different SSRCs in different RTP sessions.

The requirement that can be derived from the above usage is that the transport flows for each RTP session might be under common control, but still are addressed to what looks like different endpoints (based on addresses and ports). This connection diagram cannot be accomplished using one RTP session and thus multiple RTP sessions are needed.

3.11. Non-Symmetric Mixer/Translators

Shortcut name: Topo-Asymmetric

It is theoretically possible to construct an MCU that is a Mixer in one direction and a Translator in another. The main reason to consider this would be to allow topologies similar to Figure 13, where the Mixer does not need to mix in the direction from B or D towards the multicast domains with A and C. Instead, the media

streams from B and D are forwarded without changes. Avoiding this mixing would save media processing resources that perform the mixing in cases where it isn't needed. However, there would still be a need to mix B's stream towards D. Only in the direction B -> multicast domain or D -> multicast domain would it be possible to work as a Translator. In all other directions, it would function as a Mixer.

The Mixer/Translator would still need to process and change the RTCP before forwarding it in the directions of B or D to the multicast domain. One issue is that A and C do not know about the mixed-media stream the Mixer sends to either B or D. Therefore, any reports related to these streams must be removed. Also, receiver reports related to A and C's media stream would be missing. To avoid A and C thinking that B and D aren't receiving A and C at all, the Mixer needs to insert locally generated reports reflecting the situation for the streams from A and C into B and D's Sender Reports. In the opposite direction, the Receiver Reports from A and C about B's and D's stream also need to be aggregated into the Mixer's Receiver Reports sent to B and D. Since B and D only have the Mixer as source for the stream, all RTCP from A and C must be suppressed by the Mixer.

This topology is so problematic and it is so easy to get the RTCP processing wrong, that it is not recommended for implementation.

3.12. Combining Topologies

Topologies can be combined and linked to each other using Mixers or Translators. However, care must be taken in handling the SSRC/CSRC space. A Mixer does not forward RTCP from sources in other domains, but instead generates its own RTCP packets for each domain it mixes into, including the necessary Source Description (SDES) information for both the CSRCs and the SSRCs. Thus, in a mixed domain, the only SSRCs seen will be the ones present in the domain, while there can be CSRCs from all the domains connected together with a combination of Mixers and Translators. The combined SSRC and CSRC space is common over any Translator or Mixer. It is important to facilitate loop detection, something that is likely to be even more important in combined topologies due to the mixed behavior between the domains. Any hybrid, like the Topo-Video-switch-MCU or Topo-Asymmetric, requires considerable thought on how RTCP is dealt with.

4. Comparing Topologies

The topologies discussed in Section 3 have different properties. This section first describes these properties and then analyzes how these properties are supported by the different topologies. Note that, even if a certain property is supported within a particular

topology concept, the necessary functionality may be optional to implement.

4.1. Topology Properties

4.1.1. All to All Media Transmission

To recapitulate, multicast, and in particular Any Source Multicast (ASM), provides the functionality that everyone may send to, or receive from, everyone else within the session. Source-specific Multicast (SSM) can provide a similar functionality by having anyone intending to participate as sender to send its media to the SSM distribution source. The SSM distribution source forwards the media to all receivers subscribed to the multicast group. Mesh, MCUs, Mixers, SFMs and Translators may all provide that functionality at least on some basic level. However, there are some differences in which type of reachability they provide.

Closest to true IP-multicast-based, all to all transmission comes perhaps the transport Translator function called "relay" in in Section 3.5, as well as the Mesh with joint RTP sessions. Media Translators, Mesh with independent RTP Sessions, Mixers, SFUs and the MCU variants do not provide a fully meshed forwarding on the transport level; instead, they only allow limited forwarding of content from the other session participants.

The "all to all media transmission" requires that any media transmitting entity considers the path to the least capable receiver. Otherwise, the media transmissions may overload that path. Therefore, a media sender needs to monitor the path from itself to any of the participants, to detect the currently least capable receiver, and adapt its sending rate accordingly. As multiple participants may send simultaneously, the available resources may vary. RTCP's Receiver Reports help performing this monitoring, at least on a medium time scale.

The resource consumption for performing all to all transmission varies depending with the topology. Both ASM and SSM have the benefit that only one copy of each packet traverses a particular link. Using a relay causes the transmission of one copy of a packet per client-to-relay path and packet transmitted. However, in most cases the links carrying the multiple copies will be the ones close to the relay (which can be assumed to be part of the network infrastructure with good connectivity to the backbone), rather than the clients (which may be behind slower access links). The Mesh causes N-1 streams of transmitted packets to traverse the first hop link from the client, in an N client mesh. How long the different paths are common, is highly situation dependent.

The transmission of RTCP by design adapts to any changes in the number of participants due to the transmission algorithm, defined in the RTP specification [RFC3550], and the extensions in AVPF [RFC4585] (when applicable). That way, the resources utilized for RTCP stay within the bounds configured for the session.

4.1.2. Transport or Media Interoperability

All Translators, Mixers, and RTCP-terminating MCU, and Mesh with individual RTP sessions, allow changing the media encoding or the transport to other properties of the other domain, thereby providing extended interoperability in cases where the participants lack a common set of media codecs and/or transport protocols. Selective Forwarding Middleboxes can adopt the transport, and (at least) selectively forward the encoded streams that match a receiver's capability. It requires an additional translator to change the media encoding if the encoded streams do not match the receiver's capabilities.

4.1.3. Per Domain Bit-Rate Adaptation

Participants are most likely to be connected to each other with a heterogeneous set of paths. This makes congestion control in a Point to Multipoint set problematic. For the ASM, SSM, Mesh with common RTP session, and Transport Relay scenario, each individual sender has to adapt to the receiver with the least capable path, yielding suboptimal quality for the receivers behind the more capable paths. This is no longer necessary when Media Translators, Mixers, SFM or MCUs are involved, as each participant only needs to adapt to the slowest path within its own domain. The Translator, Mixer, SFM, or MCU topologies all require their respective outgoing streams to adjust the bit-rate, packet-rate, etc., to adapt to the least capable path in each of the other domains. That way one can avoid lowering the quality to the least-capable participant in all the domains at the cost (complexity, delay, equipment) of the Mixer, SFM or Translator, and potentially media sender (multicast/layered encoding and sending the different representations).

4.1.4. Aggregation of Media

In the all to all media property mentioned above and provided by ASM, SSM, Mesh with common RTP session, and relay, all simultaneous media transmissions share the available bit-rate. For participants with limited reception capabilities, this may result in a situation where even a minimal acceptable media quality cannot be accomplished, because multiple media streams need to share the same resources. One solution to this problem is to provide for a Mixer, or MCU to aggregate the multiple streams into a single one, where the single

stream takes up less resources in terms of bit-rate. This aggregation can be performed according to different methods. Mixing or selection are two common methods. Selection is almost always possible and easy to implement. Mixing requires resources in the mixer, and may be relatively easy and not impairing the quality too badly (audio) or quite difficult (video tiling, which is not only computationally complex but also reduces the pixel count per stream, with corresponding loss in perceptual quality).

4.1.5. View of All Session Participants

The RTP protocol includes functionality to identify the session participants through the use of the SSRC and CSRC fields. In addition, it is capable of carrying some further identity information about these participants using the RTCP Source Descriptors (SDS). In topologies that provide an all-to-all functionality, i.e. ASM, Mesh with common RTP session, Relay a compliant RTP implementation offers the functionality directly as specified in RTP. In topologies that do not offer all-to-all communication, it is necessary that RTCP is handled correctly in domain bridging function. RTP includes explicit specification text for Translators and Mixers, and for SFMs the required functionality can be derived from that text. However, the MCU described in Section 3.8 cannot offer the full functionality for session participant identification through RTP means. The topologies that create independent RTP sessions per endpoint or pair of endpoints, like Back to Back RTP session, MESH with independent RTP sessions, and the RTCP terminating MCU RTCP terminating MCU (Section 3.9) do not support RTP based identification of session participants. In all those cases, other non-RTP based mechanisms need to be implemented if such knowledge is required or desirable.

4.1.6. Loop Detection

In complex topologies with multiple interconnected domains, it is possible to unintentionally form media loops. RTP and RTCP support detecting such loops, as long as the SSRC and CSRC identities are maintained and correctly set in forwarded packets. Loop detection will work in ASM, SSM, Mesh with joint RTP session, and Relay. It is likely that loop detection works for the video switching MCU Section 3.8, at least as long as it forwards the RTCP between the participants. However, the Back to Back RTP sessions, Mesh with independent RTP sessions, SFM, will definitely break the loop detection mechanism.

4.2. Comparison of Topologies

The table below attempts to summarize the properties of the different topologies. The legend to the topology abbreviations are: Topo-Point-to-Point (PtP), Topo-ASM (ASM), Topo-SSM (SSM), Topo-Trns-Translator (TT), Topo-Media-Translator (including Transport Translator) (MT), Topo-Mesh with joint session (MJS), Topo-Mesh with individual sessions (MIS), Topo-Mixer (Mix), Topo-Asymmetric (ASY), Topo-Video-switch-MCU (VSM), and Topo-RTCP-terminating-MCU (RTM), Selective Forwarding Middlebox (SFM). In the table below, Y indicates Yes or full support, N indicates No support, (Y) indicates partial support, and N/A indicates not applicable.

Property	PtP	ASM	SSM	TT	MT	MJS	MIS	Mix	ASY	VSM	RTM	SFM
All to All media Interoperability	N	Y	(Y)	Y	Y	Y	(Y)	(Y)	(Y)	(Y)	(Y)	(Y)
Per Domain Adaptation	N/A	N	N	Y	Y	Y	Y	Y	Y	N	Y	Y
Aggregation of media	N/A	N	N	N	Y	N	Y	Y	Y	N	Y	Y
Full Session View	N	N	N	N	N	N	N	Y	(Y)	Y	Y	N
Loop Detection	Y	Y	Y	Y	Y	Y	N	Y	Y	(Y)	N	N

Please note that the Media Translator also includes the transport Translator functionality.

5. Security Considerations

The use of Mixers, SFMs and Translators has impact on security and the security functions used. The primary issue is that both Mixers, SFMs and Translators modify packets, thus preventing the use of integrity and source authentication, unless they are trusted devices that take part in the security context, e.g., the device can send Secure Realtime Transport Protocol (SRTP) and Secure Realtime Transport Control Protocol (SRTCP) [RFC3711] packets to session endpoints. If encryption is employed, the media Translator, SFM and Mixer need to be able to decrypt the media to perform its function. A transport Translator may be used without access to the encrypted payload in cases where it translates parts that are not included in the encryption and integrity protection, for example, IP address and UDP port numbers in a media stream using SRTP [RFC3711]. However, in general, the Translator, SFM or Mixer needs to be part of the signalling context and get the necessary security associations (e.g., SRTP crypto contexts) established with its RTP session participants.

Including the Mixer, SFM and Translator in the security context allows the entity, if subverted or misbehaving, to perform a number of very serious attacks as it has full access. It can perform all the attacks possible (see RFC 3550 and any applicable profiles) as if

the media session were not protected at all, while giving the impression to the session participants that they are protected.

Transport Translators have no interactions with cryptography that works above the transport layer, such as SRTP, since that sort of Translator leaves the RTP header and payload unaltered. Media Translators, on the other hand, have strong interactions with cryptography, since they alter the RTP payload. A media Translator in a session that uses cryptographic protection needs to perform cryptographic processing to both inbound and outbound packets.

A media Translator may need to use different cryptographic keys for the inbound and outbound processing. For SRTP, different keys are required, because an RFC 3550 media Translator leaves the SSRC unchanged during its packet processing, and SRTP key sharing is only allowed when distinct SSRCs can be used to protect distinct packet streams.

When the media Translator uses different keys to process inbound and outbound packets, each session participant needs to be provided with the appropriate key, depending on whether they are listening to the Translator or the original source. (Note that there is an architectural difference between RTP media translation, in which participants can rely on the RTP Payload Type field of a packet to determine appropriate processing, and cryptographically protected media translation, in which participants must use information that is not carried in the packet.)

When using security mechanisms with Translators, SFMs and Mixers, it is possible that the Translator, SFM or Mixer could create different security associations for the different domains they are working in. Doing so has some implications:

First, it might weaken security if the Mixer/Translator accepts a weaker algorithm or key in one domain than in another. Therefore, care should be taken that appropriately strong security parameters are negotiated in all domains. In many cases, "appropriate" translates to "similar" strength. If a key management system does allow the negotiation of security parameters resulting in a different strength of the security, then this system should notify the participants in the other domains about this.

Second, the number of crypto contexts (keys and security related state) needed (for example, in SRTP [RFC3711]) may vary between Mixers, SFMs and Translators. A Mixer normally needs to represent only a single SSRCs per domain and therefore needs to create only one security association (SRTP crypto context) per domain. In contrast, a Translator needs one security association per participant it

translates towards, in the opposite domain. Considering Figure 11, the Translator needs two security associations towards the multicast domain, one for B and one for D. It may be forced to maintain a set of totally independent security associations between itself and B and D respectively, so as to avoid two-time pad occurrences. These contexts must also be capable of handling all the sources present in the other domains. Hence, using completely independent security associations (for certain keying mechanisms) may force a Translator to handle $N \times DM$ keys and related state; where N is the total number of SSRCs used over all domains and DM is the total number of domains.

The multicast based (ASM and SSM), Relay and Mesh with common RTP session are all topologies with multiple endpoints that requires knowledge about the different crypto contexts for the endpoints. These multi-party topologies have special requirements on the key-management as well as the security functions. Specifically source-authentication in these environments has special requirements.

There exist a number of different mechanisms to provide keys to the different participants. One example is the choice between group keys and unique keys per SSRC. The appropriate keying model is impacted by the topologies one intends to use. The final security properties are dependent on both the topologies in use and the keying mechanisms' properties, and need to be considered by the application. Exactly which mechanisms are used is outside of the scope of this document. Please review RTP Security Options [RFC7201] to get a better understanding of most of the available options.

6. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

7. Acknowledgements

The authors would like to thank Bo Burman, Umesh Chandra, Roni Even, Keith Lantz, Ladan Gharai, Geoff Hunt, Mark Baugher, and Alex Eleftheriadis for their help in reviewing this document.

8. References

8.1. Normative References

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

- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.
- [RFC4575] Rosenberg, J., Schulzrinne, H., and O. Levin, "A Session Initiation Protocol (SIP) Event Package for Conference State", RFC 4575, August 2006.
- [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, July 2006.

8.2. Informative References

- [I-D.ietf-avtcore-rtp-multi-stream-optimisation] Lennox, J., Westerlund, M., Wu, W., and C. Perkins, "Sending Multiple Media Streams in a Single RTP Session: Grouping RTCP Reception Statistics and Other Feedback", draft-ietf-avtcore-rtp-multi-stream-optimisation-02 (work in progress), February 2014.
- [RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, August 1989.
- [RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.
- [RFC3569] Bhattacharyya, S., "An Overview of Source-Specific Multicast (SSM)", RFC 3569, July 2003.
- [RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, August 2006.
- [RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, February 2008.
- [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.
- [RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.

- [RFC6285] Ver Steeg, B., Begen, A., Van Caenegem, T., and Z. Vax, "Unicast-Based Rapid Acquisition of Multicast RTP Sessions", RFC 6285, June 2011.
- [RFC6465] Ivov, E., Marocco, E., and J. Lennox, "A Real-time Transport Protocol (RTP) Header Extension for Mixer-to-Client Audio Level Indication", RFC 6465, December 2011.
- [RFC7201] Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", RFC 7201, April 2014.

Authors' Addresses

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

Phone: +46 10 714 82 87
Email: magnus.westerlund@ericsson.com

Stephan Wenger
Vidyo
433 Hackensack Ave
Hackensack, NJ 07601
USA

Email: stewe@stewe.org

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 05, 2015

N. Ismail
Cisco
R. Barnes
Mozilla
D. Benham
N. Buckles
Cisco
July 04, 2014

Requirements for Secure RTP Media Switching
draft-ismail-avtcore-media-req-00

Abstract

This draft outlines the requirements for enabling media switches to form a multimedia multi-user conferences without needing to have the keys used to provide confidentiality and integrity for the media in the conference.

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 January 05, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. Media Switching/RTFS Architecture	3
4. RTP header manipulation	5
5. Requirements	7
6. Example Scenario	7
7. Security Considerations	9
8. IANA Considerations	9
9. Acknowledgements	9
10. References	10
10.1. Normative References	10
10.2. Informative References	10
Authors' Addresses	10

1. Introduction

Modern audio and video conferencing systems include RTP middleboxes that can often "switch" video and audio streams without mixing them. When receivers have homogenous coding capabilities and can receive multiple streams each, such media switchers avoid the need to decode and re-encode media for the purpose of compositing video or mixing audio. Instead they can forward encoded media as it was sent by the transmitter. In this case, a media switching device can behave more like a media switching RTP Translator [I-D.ietf-avtcore-rtp-topologies-update], which we will label an RTP Translator Forwarding Switch (RTFS).

Modern audio and video conferencing systems have also decomposed switching infrastructure into a) a controller that deals with the signaling and keeps track of who is in the conference and b) one or more media switching devices that receive, rewrite headers and transmit streams to receivers. In scalable systems, media switching devices may be deployed in many distributed locations to optimize bandwidth or latency and may be rented on demand from third-parties to meet peak loading needs. Therefore, there is a need to locate switching devices in data centers and/or be operated by third-parties not otherwise trusted with decryption or encryption of audio and video media.

This draft outlines the requirements for enabling media switching/RTFS devices to perform only the functions they need to, including header rewrites and authenticating transmitters and receivers, without

having to acquire or use the keys to provide confidentiality and integrity for the media in SRTP. This enables deployments where the privacy of the media can be assured even when a third-party service is used for switching media.

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 [RFC2119].

3. Media Switching/RTFS Architecture

In traditional conferencing systems, the conferencing media infrastructure fully decrypts, decodes and processes RTP media streams received from one or more transmitters prior to forwarding the newly encoded (transcoded, composited and/or mixed) and encrypted RTP media streams to the rest of receivers. Media Switching Mixers, which may need to composite or mix media, maintain independent and persistent SRTP sessions with each endpoint [I-D.ietf-avtcore-rtp-topologies-update]. More specifically, each endpoint establishes a point-to-point SRTP session with conferencing media infrastructure, which has its own persistent SSRCs, SRTP keys and SRTP contexts (reference the figure below) [RFC7201].

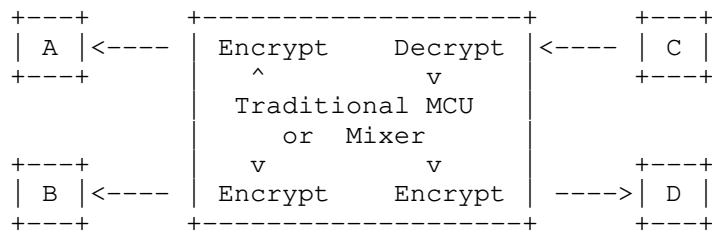


Figure 1: Traditional MCU or Mixer

When receivers have homogenous coding capabilities and can receive multiple streams each, a media switcher can avoid processing media and (selectively) forward streams while manipulating only the necessary parts of the RTP headers prior to forwarding to receivers. The RTP payload part of streams from transmitters is forwarded without any processing or changes.

In this case, a media switching device can behave more like a scalable RTP Translator Forwarding Switch (RTFS), maintaining the SSRCs of the transmitting endpoints rather than generating their own persistent SSRCs towards every receiving endpoint (reference the figure below). Though this is not the only viable embodiment of a

media switching architecture, this is the most relevant for the requirements discussed in this document.

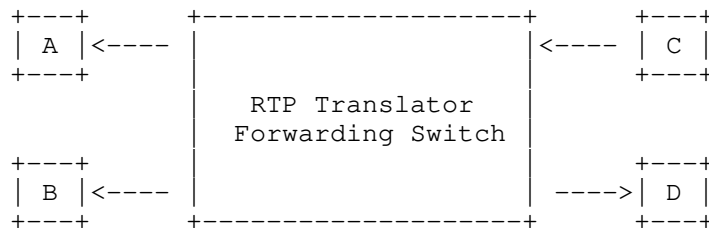
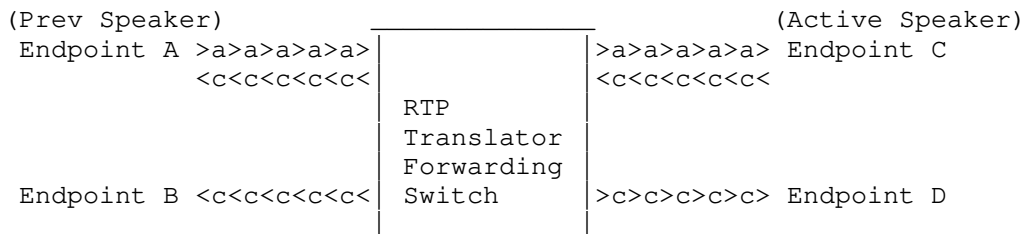


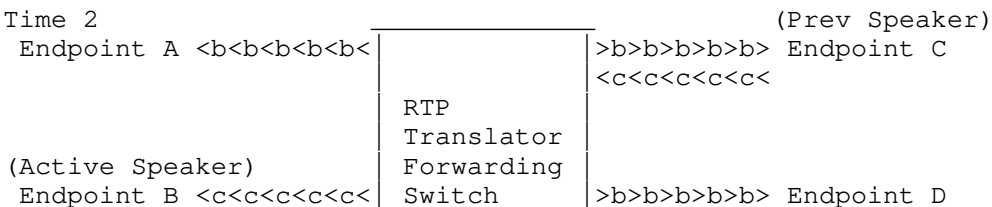
Figure 2: Scalable RTP Translator Forwarding Switch (RTFS)

These media switching/RTFS devices may selectively forward only certain transmitted stream(s) at any given time, such as the video and audio stream from the currently active speaker. In this case, endpoints receive different RTP video streams that are generated by different transmitters, each with its own SSRC, SRTP key and SRTP context. All these streams are rendered to the end user as a single video source representing the most active speaker. Moreover, endpoints do not receive the same RTP streams all the times. For example, in the figure below, endpoints A, B and D receive the video streams from endpoint C, the currently active speaker, which is actually receiving video from endpoint A, the previous active speaker. Later, when endpoint B becomes the active speaker, then endpoints A, C and D will start to receive video from B, which continues to receive video from endpoint C. In the final time slot, when Endpoint A becomes the active speaker, the process continues.

Time 1



Time 2



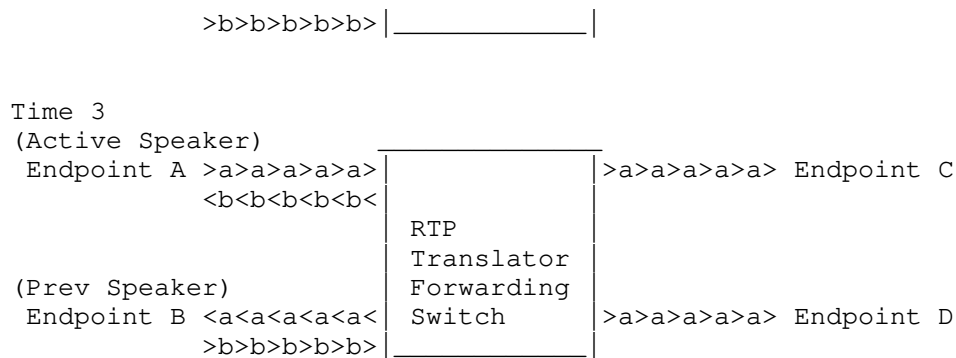


Figure 3: RTFS Media Flow for Active Speakers

Meeting the objective of scalability and simplicity in this media switching architecture starts with minimizing/eliminating the media processing performed by the media switching device, but can also to be extended to cryptography, where crypto processing and crypto state maintained by the media switching/RTFS devices are minimized. With the advent of cloud-based services, it is essential to enable deployments where the privacy of the media can be assured even when a third-party service is used for conference switching. Then enterprises can use cloud-based, third-party conferencing services while restricting such from accessing and manipulation of their media content. The ability to eliminate the need of media switching/RTFS devices to decrypt and re-encrypt packets is not merely a scalability and simplicity requirement, but is also a core security requirement in cloud-based conferencing services.

4. RTP header manipulation

A media switching/RTFS device might need to modify some of the RTP header fields to map between different values picked by different endpoints prior to switching. An example is the RTP payload type values which for SIP endpoints calling into the conference are picked by the endpoints. Different endpoints are likely to pick different values for the same media format. The media switching device is responsible for mapping between such different values. In the case of RTP payload types, the conference system might be able to send a SIP reinvite to renegotiate the RTP payload type value down to a shared value hence avoiding the remapping. This mechanism does not always work as endpoints can choose to use asymmetric payload types. Renegotiation also adds complexity and delays to the conferencing system. Other RTP header fields such as RTP extension headers can also be modified, deleted or added as they are negotiated separately with each participants.

On the other hand, two of the RTP fields must not be modified by media switches that do not have access to the media encryption keys. These two fields are the SSRC and the RTP sequence number. Both fields are used in the calculation of the SRTP cipher's IV, thus requiring a total re-encryption upon modification.

Below is the set of RTP header fields along with whether a media switching/RTFS device might modify them, unlikely to modify them or must not modify them.

- o Version (V): This field is unlikely to be modified by the media switching device
- o Padding marker (P): This field is unlikely to be modified by the media switching device
- o Extension (X): The media switching device might modify this field when it needs to add RTP extension headers where none existed or if it needs to delete existing RTP extension headers
- o Contributing sources count (CC): The media switching device is unlikely to modify this field
- o Marker bit (M): This field is unlikely to be modified by the media switching device
- o Payload Type (PT): The media switching device might modify this field to map between different RTP type values picked by different endpoints
- o Sequence Number (SEQ): The media switching device must not modify this field
- o Timestamp (TS): This field is unlikely to be modified by the media switching device
- o Synchronization Source (SSRC): This field must not be modified by the media switching device
- o Extension Header (ExtHDR): The media switching device is likely modify this field either to change its value or to delete it completely

5. Requirements

The following are the security solution requirements for media switching/RTFS device that enable media privacy to be maintained across participant endpoints.

1. Solution needs to maintain all current SRTP security properties.
2. Solution need to extend replay attacks protection to cover cross-participants replay prevention. Packets sent between the media switching device and participant A cannot be retransmitted to participant B undetected.
3. Keys used for encryption and authentication of RTP payloads and other information deemed unsuitable for accessibility by the media switching device must not be generated by or accessible to any of the media switching devices.
4. The media switching devices must be capable, if authorized, of changing any part of an RTP header except for the RTP sequence number and SSRC. This in turn mandates that the media switching devices must have access to the keys used for the authentication of RTP header fields other than SSRC and RTP sequence number when a proper authorization is in place.
5. The SRTP master keys must not be generated by the media switching devices
6. The media switching devices must not be involved in the distribution of the SRTP master keys to participants nor in the authentication of the participants identities for the purpose of key distribution
7. The media switching devices must be able to switch an already active SRTP stream to a new receiver while guaranteeing the timely synchronization between the SRTP context of the transmitter and its old and new receivers. Of special interest is the RoC part pf the SRTP context due to its dynamic nature. It is important to note that media switching devices can not change RTP sequence numbers as that would require packet re-encryption.

6. Example Scenario

The above requirements (especially 3 and 4) imply that there is a need for SRTP ciphersuites that allow a split key and split authentication model. Instead of the current single SRTP master key, this document requires two independent SRTP master keys. The first

is an end to end key that is used for the encryption of the RTP payload and other information requiring end-to-end encryption. The end to end key is also used for the authentication of the RTP payload, the RTP sequence number, RoC and SSRC as well as any other information requiring end-to-end authentication. The second key is hop-by-hop key used for the authentication of the RTP packet as well as any other information requiring hop by hop authentication (e.g. RTCP packet authentication). The hop-by-hop key can also be used for encryption of information that the switch is authorized to access and modify, such as encrypted RTCP packets.

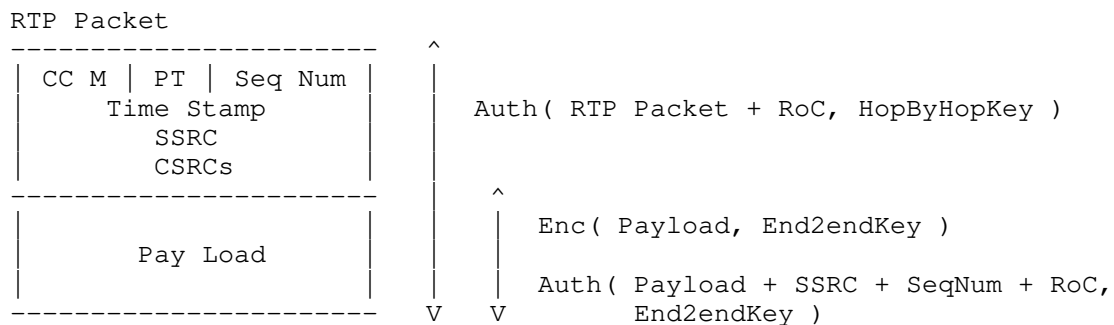
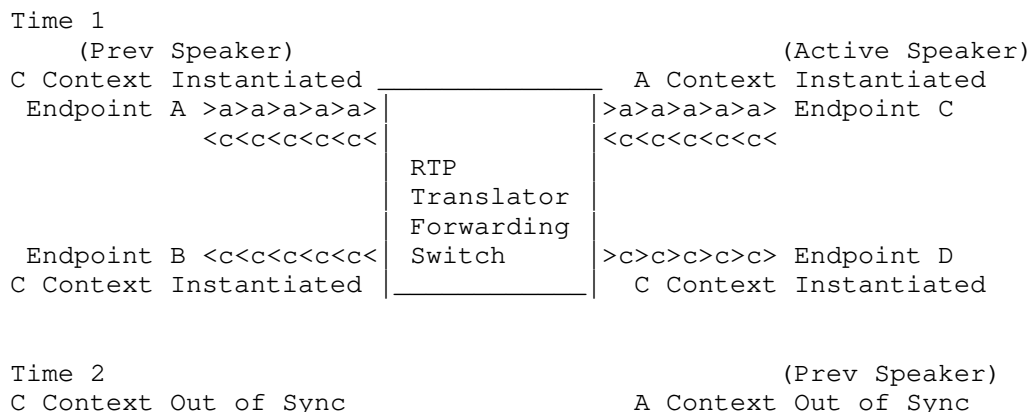


Figure 4: SRTP Split key-authentication model

The following figures illustrate how this split-context system could be used to accomplish the RTP forwarding objectives above. We do not show the control interactions that would be necessary to distribute the requisite keys among the participants.

TODO: Flesh out this example case further

Note that media from endpoints are flowing in direction of the arrows.



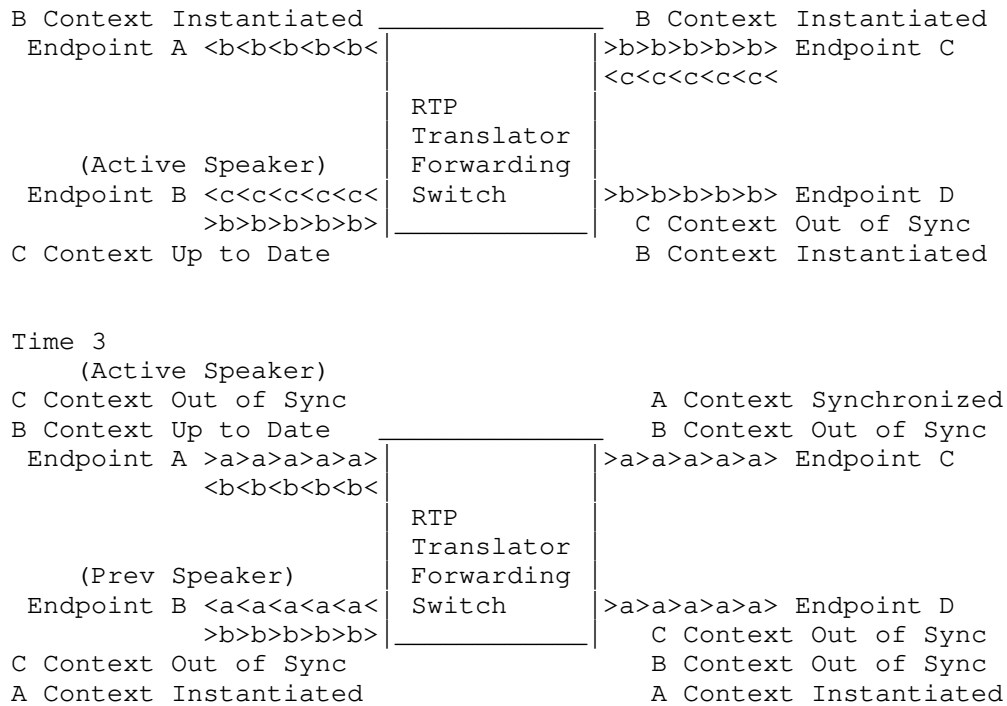


Figure 5: SRTP context synchronization

7. Security Considerations

This specification is all about new requirements for a system for securing RTP headers separately from the RTP body.

The requirements discussed above lead to a need for new SRTP cipher suites that split protection between hop-by-hop and end-to-end protections. This split may require new models for managing SRTP keys, e.g., extensions to DTLS-SRTP or EKT. We do not address requirements for key management in this document, since they would be accomplished at the control layer, rather than the RTP forwarding layer.

8. IANA Considerations

This document requires no actions from IANA.

9. Acknowledgements

The authors would like to thank Eric Rescorla and Cullen Jennings for their inputs. <GET YOUR NAME HERE - PLEASE SEND COMMENTS>.

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

10.2. Informative References

- [I-D.ietf-avtcore-rtp-topologies-update]
Westerlund, M. and S. Wenger, "RTP Topologies", draft-ietf-avtcore-rtp-topologies-update-02 (work in progress), May 2014.
- [I-D.ietf-rtcweb-security-arch]
Rescorla, E., "WebRTC Security Architecture", draft-ietf-rtcweb-security-arch-09 (work in progress), February 2014.
- [I-D.ietf-rtcweb-security]
Rescorla, E., "Security Considerations for WebRTC", draft-ietf-rtcweb-security-06 (work in progress), January 2014.
- [RFC7201] Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", RFC 7201, April 2014.

Authors' Addresses

Nermeen Ismail
Cisco
170 W Tasman Dr.
San Jose
US

Email: nermeen@cisco.com

Richard Barnes
Mozilla
331 E Evelyn Ave.
Mountain View
US

Email: rlb@ipv.sx

David Benham
Cisco
170 W Tasman Dr.
San Jose
US

Email: dbenham@cisco.com

Nathan Buckles
Cisco
170 W Tasman Dr.
San Jose
US

Email: nbuckles@cisco.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 5, 2015

J. Mattsson
Y. Cheng
Ericsson
July 4, 2014

Privacy Ensured Cloud Conferencing - Use Case and Goals
draft-mattsson-avtvore-cloud-conferencing-use-case-00

Abstract

The aim of this document is to describe the use case of privacy ensured cloud conferencing in a pervasive monitoring landscape and to point out goals for a solution mitigating the pervasive monitoring threat [RFC7258].

Virtualized cloud-based conferencing is happening and IETF should take action to make such services viable and trustworthy from a pervasive monitoring perspective.

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 January 5, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Background	3
3. Goals and Non-Goals	3
3.1. Goals	4
3.1.1. Ensure End-To-End(s) Confidentiality	4
3.1.2. Ensure End-To-End Source Authentication	4
3.1.3. Provide a more efficient service than Full-Mesh	4
3.1.4. Support Cloud Based Conferencing	4
3.2. Non-Goals	5
3.2.1. Securing the endpoints	5
3.2.2. Concealing that communication occurs	5
3.2.3. Individual Media Source Authentication	5
3.2.4. Preventing invited user to access content	5
4. Problems with Current Technology	6
5. Conclusions	7
6. Security Considerations	7
7. Acknowledgements	7
8. References	7
Authors' Addresses	8

1. Introduction

This document discusses the possibility to provide real-time conference communication services to enterprises and other organizations that try to ensure the privacy of their communication in a world with pervasive monitoring. This includes being able to purchase conferencing supporting network services, including cloud-based ones that are resistant to content monitoring.

This document starts with a background section discussing the development in the world that affects considerations for ensuring communication privacy. Next goals and non-goals for privacy ensured cloud conferencing are stated, followed by the considerations around using current technology and standards.

Some strategies for secure cloud systems can be found in [I-D.jennings-perpass-secure-rai-cloud].

2. Background

Within the field of real-time conferencing there is an ongoing transformation. A transformation towards more cloud based, virtualized and software based conferencing server implementations. The central conferencing server on dedicated hardware is under heavy competition from virtualized servers. One enabling factor for this is the increased capabilities of the end-points that allow them to decode and process multiple simultaneously received media streams. This in its turn has made the central conferring media nodes to switch from mixing or composing media in the decoded domain to instead perform the much less heavy-weight operation of selection, switching and forwarding of media streams, at least for video. Thus making virtualized cloud-based conferencing services viable.

This transformation to virtualization and cloud-based services increases the threats towards the confidentiality of the content of any conference. The reason is that an attacker interested in surveillance of a conference now also has the possibility to attack the cloud provider and attempt to get access to the actual hardware or virtualization layer as a method of accessing what happens within the conference services server instances.

From the pervasive monitoring debate we know that there are many organizations that are actively performing large scale pervasive monitoring, this includes national agencies, but also criminal organizations may be engaged in such activities. It has been revealed that several large service providers have been compromised, resulting in people questioning the impact and security of sourcing services for enterprise or governments to external providers. IETF has stated that pervasive monitoring is an attack and that it should be mitigated [RFC7258].

The trend of using virtualized cloud-based services (e.g. conferencing) has a number of positive effects on flexibility, CAPEX, ease of use, etc. IETF should take action to make such services viable and trustworthy from a pervasive monitoring perspective. One important part of pervasive monitoring is the passive pervasive monitoring [I-D.trammell-perpass-ppa].

3. Goals and Non-Goals

This section proposed goals and non-goals for the privacy ensured conferencing use case.

3.1. Goals

3.1.1. Ensure End-To-End(s) Confidentiality

The content of the communication and all its media needs to be confidential within the group of entities explicitly invited into the conference. An external monitoring adversary should be unable to deduce the human to human communication that actually occurred from capturing the media packets within a time frame for which the communication remains confidential.

3.1.2. Ensure End-To-End Source Authentication

In a conference system with multiple participants it is vital that the multi-media content presented to any of the participant humans are from the stated participants, and not an adversary that attempts to inject misleading content. Nor should an adversary be able to fool the system into becoming a trusted party in the conference, only explicitly invited parties shall be able to contribute content.

3.1.3. Provide a more efficient service than Full-Mesh

A multi-party conference that has the goals of confidentiality and source authentication can be established as a full MESH, i.e. each participating end-point directly addresses each of the other participants. However, this has a significant issue with the amount of consumed resources in both the uplink and the downlink from each participant. To reduce this issue one wants to consider other topologies, which implies network or centralized server functionalities.

3.1.4. Support Cloud Based Conferencing

To achieve a cost effective and scalable conferencing support the conference central nodes must be possible to run as instances in a cloud based virtualized environment.

From a security stand point this is a significant issue. In a virtual, possibly multi-tenant, cloud environment the ones running the conferencing supporting implementation instance can't trust that anything maintained at that instance is secure from an adversary. A pervasive monitoring entity may in fact have direct access to the hardware through vulnerabilities in the virtualization layer the instance runs in.

3.2. Non-Goals

3.2.1. Securing the endpoints

The security of a communication session requires that the endpoints are not compromised and that the users are trustworthy. If not, credentials and decrypted content may be shared with third parties. However this is hard to prevent through system design. Thus, it should be assumed that the endpoint is secure and the user is trustworthy, how to achieve this is out of scope.

3.2.2. Concealing that communication occurs

A non-goal is to attempt to prevent a pervasive monitoring adversary from knowing that the communication session has occurred. The reason for excluding this as a goal is that first of all it is extremely difficult to achieve, as a pervasive monitoring adversary can be expected to be able to have knowledge of all IP flows that enter or exit local ISPs, across links that straddle nation borders or internet exchange points. Thus the flows required to achieve the communication session need to be highly difficult to correlate between different legs of the communication. At this stage this is deemed too difficult to attempt and will need to be subject for future studies. Existing attempts include The Onion Router (TOR), which has been claimed to be possible to monitor, at least partially, by an adversary with sufficient reach.

3.2.3. Individual Media Source Authentication

Although the participants in the conference are authenticated, it should probably not be a goal to provide source authentication of the media at the individual user level, instead being satisfied with being able to authenticate media as coming from an invited conference participant or not.

There exist solutions that can provide individual media source authentication, e.g. TESLA. However they impact the performance or security properties they provide. Thus, further studies are required to determine impact and resulting security properties if desired to have individual source authentication.

3.2.4. Preventing invited user to access content

As an invited user will be provided with the content protection keys, an invited user can unless active measures are taken against this, decrypt content from the time periods prior and post the user is officially part of the conference.

If this is a concern the solution could be extended to re-key the content protection keys every time a user joins or leaves the conference so each particular set of conference participants uses a unique key. However, this also changes the trust level required on the conference roster handling at any point and how to keep that accurate and secured. Further the re-keying operations and their timely completion become an obstacle in system design.

4. Problems with Current Technology

If SRTP is used end-to-end, a multiparty conference where the middlebox/server duplicates packets and forwards the complete encrypted packets from a client to multiple participants, the RTP handling is problematic.

The RTP mixer will be forced to behave like a video switching MCU in RFC 5117. SRTP prevents the mixer from performing any type of RTP or RTCP rewrite. However, to keep the bitrate in check its switching decision will result in stopping RTP streams from reaching the client. This results in RTP sequences with large gaps in them. These gaps hide packet losses at the edges of the gaps, resulting in that the receiver has issues in determining if loss near switching point is intentional or not. This can cause repair attempts, buffering issues, and triggering bit-rate adaptation. In addition the congestion control mechanism has significant difficulties to act correctly in such an environment.

Further the above topology requires the RTP stacks to be capable of handling multiple remote peers, including for adaptation of congestion control. This has previously been limited to any-source multicast and transport relay topologies, not RTP mixer ones.

To enable this system's properties, enable RTP mixing, while not letting the mixer get access to content, it's required to specify a two level security mechanism. In any multi-vendor environments this will require a specification as it will affect the cipher operations and the data transmitted between participants. The application could handle automatic key-management in the group of authorized participants. One approach on how to do this is [I-D.cheng-srtp-cloud]

To support video switching/relaying knowing from which points in the video streams a receiving endpoint will be able to decode is important. Thus markers for where switching points in the media stream are will be required.

To enable the middle boxes to take local decisions on this, each sender will need to include some speaker activity indication;

preferably including some type of ranking of how likely this is to contain speech. However, this activity indication also needs to leak as little information as possible about the actual content of the speech.

5. Conclusions

Virtualized cloud-based conferencing is happening and IETF should take action to make such services viable and trustworthy from a pervasive monitoring perspective.

From the goals and discussion above it is clear that to provide effective cloud based conferencing while protecting from pervasive monitoring, two layers of security is needed. This is not supported by SRTP.

There is currently active work on secure objects in the form of JSON objects in the IETF WG JOSE. But this is not applicable to RTP based real-time media.

6. Security Considerations

The whole document is about making cloud-based conferencing viable and trustworthy from a pervasive monitoring perspective.

7. Acknowledgements

The authors would like to thank Magnus Westerlund for providing much of the input to this document as well as much of the text.

8. References

[I-D.cheng-srtp-cloud]

Cheng, Y., Mattsson, J., and Naslund, M., "Secure Real-time Transport Protocol (SRTP) for Cloud Services", draft-cheng-srtp-cloud-00 (work in progress), July 2014.

[I-D.jennings-perpass-secure-rai-cloud]

Jennings, C. and S. Nandakumar, "Trustable Cloud Systems - Strategies and Recommendations", draft-jennings-perpass-secure-rai-cloud-01 (work in progress), January 2014.

[I-D.trammell-perpass-ppa]

Trammell, B., Borkmann, D., and C. Huitema, "A Threat Model for Pervasive Passive Surveillance", draft-trammell-perpass-ppa-01 (work in progress), November 2013.

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, May 2014.

Authors' Addresses

John Mattsson
Ericsson AB
SE-164 80 Stockholm
Sweden

Phone: +46 10 71 43 501
Email: john.mattsson@ericsson.com

Yi Cheng
Ericsson
SE-164 80 Stockholm
Sweden

Phone: +46 10 71 17 589
Email: yi.cheng@ericsson.com

AVTCORE
Internet-Draft
Updates: 5764 (if approved)
Intended status: Standards Track
Expires: January 3, 2015

M. Petit-Huguenin
Jive Communications
G. Salgueiro
Cisco Systems
July 2, 2014

Multiplexing Scheme Updates for Secure Real-time Transport Protocol
(SRTP) Extension for Datagram Transport Layer Security (DTLS)
draft-petithuguenin-avtcore-rfc5764-mux-fixes-00

Abstract

This document defines how Datagram Transport Layer Security (DTLS), Real-time Transport Protocol (RTP), Real-time Transport Control Protocol (RTCP), Session Traversal Utilities for NAT (STUN), and Traversal Using Relays around NAT (TURN) packets are multiplexed on a single receiving socket. It overrides the guidance from SRTP Extension for DTLS [RFC5764], which suffered from three issues described and fixed in this document.

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 January 3, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Implicit Allocation of Codepoints for New STUN Methods .	3
1.2. Implicit Allocation of New Codepoints for TLS ContentTypes	4
1.3. Multiplexing of TURN Channels	4
2. Terminology	5
3. RFC 5764 Updates	5
4. Implementation Status	6
5. Security Considerations	7
6. IANA Considerations	7
6.1. STUN Methods	7
6.2. TLS ContentType	8
6.3. TURN Channel Numbers	8
7. Acknowledgements	8
8. References	9
8.1. Normative References	9
8.2. Informative References	9
Authors' Addresses	10

1. Introduction

Section 5.1.2 of Secure Real-time Transport Protocol (SRTP) Extension for DTLS [RFC5764] defines a scheme for a Real-time Transport Protocol (RTP) [RFC3550] receiver to demultiplex Datagram Transport Layer Security (DTLS) [RFC6347], Session Traversal Utilities for NAT (STUN) [RFC5389] and Secure Real-time Transport Protocol (SRTP)/Secure Real-time Transport Control Protocol (SRTCP) [RFC3711] packets that are arriving on the RTP port. Unfortunately, this demultiplexing scheme has created three problematic issues:

1. It implicitly allocated codepoints for new STUN methods without an IANA registry reflecting these new allocations.
2. It implicitly allocated codepoints for new Transport Layer Security (TLS) ContentTypes without an IANA registry reflecting these new allocations.
3. It did not take into account the fact that the Traversal Using Relays around NAT (TURN) usage of STUN can create TURN channels that also need to be demultiplexed with the other packet types explicitly mentioned in Section 5.1.2 of RFC 5764.

These flaws in the demultiplexing scheme were unavoidably inherited by other documents, such as [I-D.ietf-mmusic-udptl-dtls] and [I-D.ietf-mmusic-sdp-bundle-negotiation]. These will need to be corrected with the updates this document provides when it become normative.

1.1. Implicit Allocation of Codepoints for New STUN Methods

The demultiplexing scheme in [RFC5764] states that the receiver can identify the packet type by looking at the first byte. If the value of this first byte is 0 or 1, the packet is identified to be STUN. The problem that arises as a result of this implicit allocation is that this restricts the codepoints for STUN methods (as described in Section 18.1 of [RFC5389]) to values between 0x000 and 0x07F, which in turn reduces the number of possible STUN method codepoints assigned by IETF Review (i.e., the range from (0x000 - 0x7FF) from 2048 to only 128 and entirely obliterating those STUN method codepoints assigned by Designated Expert (i.e., the range 0x800 - 0xFFFF). In fact, RFC 5764 implicitly (and needlessly) allocated a very large range of STUN methods, but at a minimum the IANA STUN Methods registry should properly reflect this.

There are only a few STUN method codepoints currently allocated, but this is largely attributed to the fact that STUN did not see much deployment until the development of WebRTC. For this reason, simply marking the implicit allocations made by RFC 5764 in the STUN Method registry may create a shortage of codepoints at a time when interest in STUN and STUN Usages (especially TURN) is growing rapidly. Consequently, this document also changes the RFC 5764 packet identification algorithm to expand the range assigned to the STUN protocol from 0 - 1 to 0 - 19, as the values 2-19 are unused.

In addition to explicitly allocating STUN methods codepoints from 0x500 to 0xFFFF as Reserved values, this document also updates the IANA registry such that the STUN method codepoints assigned via IETF Review are in the 0x000-0x27F range and those assigned via Designated Expert are in the 0x280-0x4FF range. The proposed changes to the STUN Method Registry is:

OLD:

0x000-0x7FF	IETF Review
0x800-0xFFFF	Designated Expert

NEW:

0x000-0x27F	IETF Review
0x280-0x4FF	Designated Expert
0x500-0xFFF	Reserved

1.2. Implicit Allocation of New Codepoints for TLS ContentType

The demultiplexing scheme in [RFC5764] dictates that if the value of the first byte is between 20 and 63 (inclusive), then the packet is identified to be DTLS. The problem that arises is that this restricts the TLS ContentType codepoints (as defined in Section 12 of [RFC5246]) to this range, and by extension implicitly allocates ContentType codepoints 0 to 19 and 64 to 255. Unlike STUN, TLS is a mature protocol that is already well established and widely implemented and thus we expect only relatively few new codepoints to be assigned in the future. With respect to TLS packet identification, this document simply explicitly reserves the codepoints from 0 to 19 and from 64 to 255 so they are not inadvertently assigned in the future.

1.3. Multiplexing of TURN Channels

When used with ICE [RFC5245], an RFC 5764 implementation can receive packets on the same socket from three different paths, as shown in Figure 1:

1. Directly from the source
2. Through a NAT
3. Relayed by a TURN server

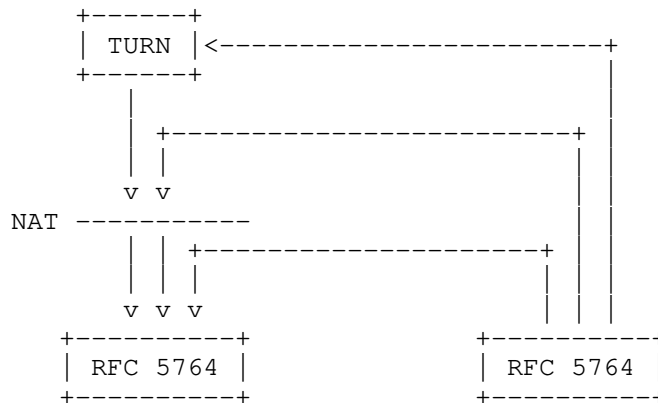


Figure 1: Packet Reception by an RFC 5764 Implementation

Even if the ICE algorithm succeeded in selecting a non-relayed path, it is still possible to receive data from the TURN server. For instance, when ICE is used with aggressive nomination the media path can quickly change until it stabilizes. Also, freeing ICE candidates is optional, so the TURN server can restart forwarding STUN connectivity checks during an ICE restart.

TURN channels are an optimization where data packets are exchanged with a 4-byte prefix, instead of the standard 36-byte STUN overhead (see Section 2.5 of [RFC5766]). The problem is that the RFC 5764 demultiplexing scheme does not define what to do with packets received over a TURN channel since these packets will start with a first byte whose value will be between 64 and 127 (inclusive). If the TURN server was instructed to send data over a TURN channel, then the current RFC 5764 demultiplexing scheme will reject these packets. Current implementations violate RFC 5764 for values 64 to 127 (inclusive) and they instead parse packets with such values as TURN. In order to prevent future documents from assigning values from the unused range to a new protocol, this document modifies the RFC 5764 demultiplexing algorithm to properly account for TURN channels.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] when they appear in ALL CAPS. When these words are not in ALL CAPS (such as "must" or "Must"), they have their usual English meanings, and are not to be interpreted as RFC 2119 key words.

3. RFC 5764 Updates

This document updates the text in Section 5.1.2 of [RFC5764] as follows:

OLD TEXT

The process for demultiplexing a packet is as follows. The receiver looks at the first byte of the packet. If the value of this byte is 0 or 1, then the packet is STUN. If the value is in between 128 and 191 (inclusive), then the packet is RTP (or RTCP, if both RTCP and RTP are being multiplexed over the same destination port). If the value is between 20 and 63 (inclusive), the packet is DTLS. This process is summarized in Figure 3.

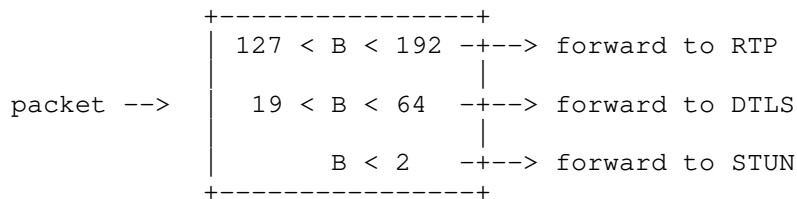


Figure 3: The DTLS-SRTP receiver's packet demultiplexing algorithm.
Here the field B denotes the leading byte of the packet.

END OLD TEXT

NEW TEXT

The process for demultiplexing a packet is as follows. The receiver looks at the first byte of the packet. If the value of this byte is in between 0 and 19 (inclusive), then the packet is STUN. If the value is in between 128 and 191 (inclusive), then the packet is RTP (or RTCP, if both RTCP and RTP are being multiplexed over the same destination port). If the value is between 20 and 63 (inclusive), the packet is DTLS. If the value is between 64 and 127 (inclusive), the packet is TURN Channel. This process is summarized in Figure 3.

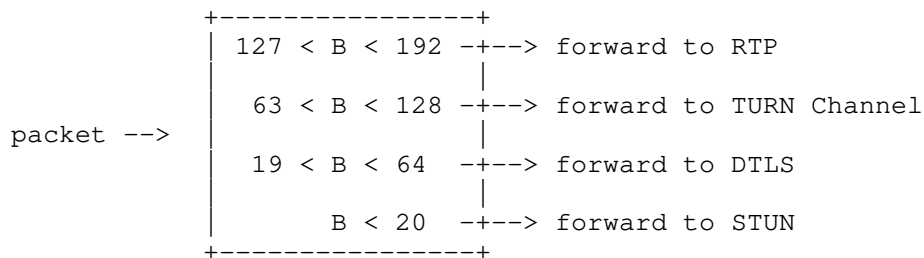


Figure 3: The DTLS-SRTP receiver's packet demultiplexing algorithm.
Here the field B denotes the leading byte of the packet.

END NEW TEXT

[[Note: we may want to use "<=" instead of "<" to make it easier on implementers.]]

4. Implementation Status

[[Note to RFC Editor: Please remove this section and the reference to [RFC6982] before publication.]]

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC6982]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC6982], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

Note that there is currently no implementation declared in this section, but the intent is to add RFC 6982 templates here from implementers that support the modifications in this document.

5. Security Considerations

This document simply updates existing IANA registries and does not introduce any specific security considerations beyond those detailed in [RFC5764].

6. IANA Considerations

6.1. STUN Methods

This specification contains the registration information for 2816 STUN Methods codepoints, as explained in Section 1.1 and in accordance with the procedures defined in Section 18.1 of [RFC5389].

Value: 0x500-0xFFF

Name: Reserved

Reference: RFC5764, RFCXXXX

This specification also reassigns the ranges in the STUN Methods Registry as follow:

Range: 0x000-0x27F

Registration Procedures: IETF Review

Range: 0x280-0x4FF

Registration Procedures: Designated Expert

6.2. TLS ContentType

This specification contains the registration information for 212 TLS ContentType codepoints, as explained in Section 1.2 and in accordance with the procedures defined in Section 12 of [RFC5246].

Value: 0-19

Description: Reserved

DTLS-OK: N/A

Reference: RFC5764, RFCXXXX

Value: 64-255

Description: Reserved

DTLS-OK: N/A

Reference: RFC5764, RFCXXXX

6.3. TURN Channel Numbers

This specification contains the registration information for 32768 TURN Channel Numbers codepoints, as explained in Section 1.3 and in accordance with the procedures defined in Section 18 of [RFC5766].

Value: 0x8000-0xFFFF

Name: Reserved

Reference: RFCXXXX

[RFC EDITOR NOTE: Please replace RFCXXXX with the RFC number of this document.]

7. Acknowledgements

The implicit STUN Method codepoint allocations problem was first reported by Martin Thomson in the RTCWEB mailing-list and discussed further with Magnus Westerlund.

Thanks to Simon Perreault and Colton Shields for the comments, suggestions, and questions that helped improve this document.

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.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.
- [RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, May 2010.
- [RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

8.2. Informative References

- [RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, July 2013.

[I-D.ietf-mmusic-udptl-dtls]

Holmberg, C., Sedlacek, I., and G. Salgueiro, "UDP Transport Layer (UDPTL) over Datagram Transport Layer Security (DTLS)", draft-ietf-mmusic-udptl-dtls-10 (work in progress), June 2014.

[I-D.ietf-mmusic-sdp-bundle-negotiation]

Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", draft-ietf-mmusic-sdp-bundle-negotiation-07 (work in progress), April 2014.

Authors' Addresses

Marc Petit-Huguenin
Jive Communications
1275 West 1600 North, Suite 100
Orem, UT 84057
USA

Email: marcph@getjive.com

Gonzalo Salgueiro
Cisco Systems
7200-12 Kit Creek Road
Research Triangle Park, NC 27709
US

Email: gsalguei@cisco.com

AVT Core Working Group
Internet-Draft
Intended status: Experimental
Expires: January 1, 2015

V. Singh
T. Karkkainen
J. Ott
S. Ahsan
Aalto University
L. Eggert
NetApp
June 30, 2014

Multipath RTP (MPRTP)
draft-singh-avtcore-mprtp-09

Abstract

The Real-time Transport Protocol (RTP) is used to deliver real-time content and, along with the RTP Control Protocol (RTCP), forms the control channel between the sender and receiver. However, RTP and RTCP assume a single delivery path between the sender and receiver and make decisions based on the measured characteristics of this single path. Increasingly, endpoints are becoming multi-homed, which means that they are connected via multiple Internet paths. Network utilization can be improved when endpoints use multiple parallel paths for communication. The resulting increase in reliability and throughput can also enhance the user experience. This document extends the Real-time Transport Protocol (RTP) so that a single session can take advantage of the availability of multiple paths between two endpoints.

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 January 1, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	38
1.1. Requirements Language	4
1.2. Terminology	4
1.3. Use-cases	4
2. Goals	5
2.1. Functional goals	5
2.2. Compatibility goals	6
3. RTP Topologies	6
4. MPRTTP Architecture	6
5. Example Media Flow Diagrams	8
5.1. Streaming use-case	8
5.2. Conversational use-case	9
6. MPRTTP Functional Blocks	10
7. Available Mechanisms within the Functional Blocks	11
7.1. Session Setup	11
7.1.1. Connectivity Checks	11
7.1.2. Adding New or Updating Interfaces	11
7.1.3. In-band vs. Out-of-band Signaling	12
7.2. Expanding RTP	13
7.3. Expanding RTCP	13
7.4. Failure Handling and Teardown	14
8. MPRTTP Protocol	14
8.1. Overview	15
8.1.1. Gather or Discovering Candidates	15
8.1.2. NAT Traversal	15
8.1.3. Choosing between In-band (in RTCP) and Out-of-band (in SDP) Interface Advertisement	16
8.1.4. In-band Interface Advertisement	16
8.1.5. Subflow ID Assignment	17
8.1.6. Active and Passive Subflows	17
8.2. RTP Transmission	17

8.3.	Playout Considerations at the Receiver	18
8.4.	Subflow-specific RTCP Statistics and RTCP Aggregation . .	18
8.5.	RTCP Transmission	19
9.	Packet Formats	19
9.1.	RTP Header Extension for MPRTCP	19
9.1.1.	MPRTCP RTP Extension for a Subflow	20
9.2.	RTCP Extension for MPRTCP (MPRTCP)	21
9.2.1.	MPRTCP Extension for Subflow Reporting	23
9.2.1.1.	MPRTCP for Subflow-specific SR, RR and XR	24
9.3.	MPRTCP Extension for Interface advertisement	26
10.	RTCP Timing reconsiderations for MPRTCP	27
11.	SDP Considerations	28
11.1.	Signaling MPRTCP Header Extension in SDP	28
11.2.	Signaling MPRTCP capability in SDP	28
11.3.	MPRTCP with ICE	29
11.4.	Increased Throughput	29
11.5.	Offer/Answer	30
11.5.1.	In-band Signaling Example	30
12.	IANA Considerations	31
12.1.	MPRTCP Header Extension	31
12.2.	MPRTCP Packet Type	31
12.3.	SDP Attributes	32
12.3.1.	"mprtcp" attribute	32
13.	Security Considerations	33
14.	Acknowledgements	33
15.	References	33
15.1.	Normative References	33
15.2.	Informative References	34
Appendix A.	Interoperating with Legacy Applications	36
Appendix B.	Change Log	36
B.1.	Changes in draft-singh-avtcore-mprtcp-09	36
B.2.	Changes in draft-singh-avtcore-mprtcp-08	36
B.3.	Changes in draft-singh-avtcore-mprtcp-06 and -07	36
B.4.	Changes in draft-singh-avtcore-mprtcp-05	36
B.5.	Changes in draft-singh-avtcore-mprtcp-04	37
B.6.	Changes in draft-singh-avtcore-mprtcp-03	37
B.7.	Changes in draft-singh-avtcore-mprtcp-02	37
B.8.	Changes in draft-singh-avtcore-mprtcp-01	38
Authors' Addresses	38

1. Introduction

Multi-homed endpoints are becoming common in today's Internet, e.g., devices that support multiple wireless access technologies such as 3G and Wireless LAN. This means that there is often more than one network path available between two endpoints. Transport protocols, such as RTP, have not been designed to take advantage of the availability of multiple concurrent paths and therefore cannot

benefit from the increased capacity and reliability that can be achieved by pooling their respective capacities.

Multipath RTP (MPRTP) is an OPTIONAL extension to RTP [RFC3550] that allows splitting a single RTP stream into multiple subflows that are transmitted over different paths. In effect, this pools the resource capacity of multiple paths. Multipath RTCP (MPRTCP) is an extension to RTCP, it is used along with MPRTP to report per-path sender and receiver characteristics.

Other IETF transport protocols that are capable of using multiple paths include SCTP [RFC4960], MPTCP [RFC6182] and SHIM6 [RFC5533]. However, these protocols are not suitable for real-time communications.

1.1. 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 [RFC2119].

1.2. Terminology

- o Endpoint: host either initiating or terminating an RTP flow.
- o Interface: logical or physical component that is capable of acquiring a unique IP address.
- o Path: sequence of links between a sender and a receiver. Typically, defined by a set of source and destination addresses.
- o Subflow: flow of RTP packets along a specific path, i.e., a subset of the packets belonging to an RTP stream. The combination of all RTP subflows forms the complete RTP stream. Typically, a subflow would map to a unique path, i.e., each combination of IP addresses and port pairs (5-tuple, including protocol) is a unique subflow.

1.3. Use-cases

The primary use-case for MPRTP is transporting high bit-rate streaming multimedia content between endpoints, where at least one is multi-homed. Such endpoints could be residential IPTV devices that connect to the Internet through two different Internet service providers (ISPs), or mobile devices that connect to the Internet through 3G and WLAN interfaces. By allowing RTP to use multiple paths for transmission, the following gains can be achieved:

- o Higher quality: Pooling the resource capacity of multiple Internet paths allows higher bit-rate and higher quality codecs to be used. From the application perspective, the available bandwidth between the two endpoints increases.
- o Load balancing: Transmitting an RTP stream over multiple paths reduces the bandwidth usage on a single path, which in turn reduces the impact of the media stream on other traffic on that path.
- o Fault tolerance: When multiple paths are used in conjunction with redundancy mechanisms (FEC, re-transmissions, etc.), outages on one path have less impact on the overall perceived quality of the stream.

A secondary use-case for MPRTTP is transporting Voice over IP (VoIP) calls to a device with multiple interfaces. Again, such an endpoint could be a mobile device with multiple wireless interfaces. In this case, little is to be gained from resource pooling, i.e., higher capacity or load balancing, because a single path should be easily capable of handling the required load. However, using multiple concurrent subflows can improve fault tolerance, because traffic can shift between the subflows when path outages occur. This results in very fast transport-layer handovers that do not require support from signaling.

2. Goals

This section outlines the basic goals that multipath RTP aims to meet. These are broadly classified as Functional goals and Compatibility goals.

2.1. Functional goals

Allow unicast RTP session to be split into multiple subflows in order to be carried over multiple paths. This may prove beneficial in case of video streaming.

- o Increased Throughput: Cumulative capacity of the two paths may meet the requirements of the multimedia session. Therefore, MPRTTP MUST support concurrent use of the multiple paths.
- o Improved Reliability: MPRTTP SHOULD be able to send redundant packets or re-transmit packets along any available path to increase reliability.

The protocol SHOULD be able to open new subflows for an existing session when new paths appear and MUST be able to close subflows when paths disappear.

2.2. Compatibility goals

MPRTP MUST be backwards compatible; an MPRTP stream needs to fall back to be compatible with legacy RTP stacks if MPRTP support is not successfully negotiated.

- o Application Compatibility: MPRTP service model MUST be backwards compatible with existing RTP applications, i.e., an MPRTP stack MUST be able to work with legacy RTP applications and not require changes to them. Therefore, the basic RTP APIs MUST remain unchanged, but an MPRTP stack MAY provide extended APIs so that the application can configure any additional features provided by the MPRTP stack.
- o Network Compatibility: individual RTP subflows MUST themselves be well-formed RTP flows, so that they are able to traverse NATs and firewalls. This MUST be the case even when interfaces appear after session initiation. Interactive Connectivity Establishment (ICE) [RFC5245] MAY be used for discovering new interfaces or performing connectivity checks.

3. RTP Topologies

RFC 5117 [RFC5117] describes a number of scenarios using mixers and translators in single-party (point-to-point), and multi-party (point-to-multipoint) scenarios. RFC 3550 [RFC3550] (Section 2.3 and 7.x) discuss in detail the impact of mixers and translators on RTP and RTCP packets. MPRTP assumes that if a mixer or translator exists in the network, then either all of the multiple paths or none of the multiple paths go via this component.

4. MPRTP Architecture

In a typical scenario, an RTP session uses a single path. In an MPRTP scenario, an RTP session uses multiple subflows that each use a different path. Figure 1 shows the difference.

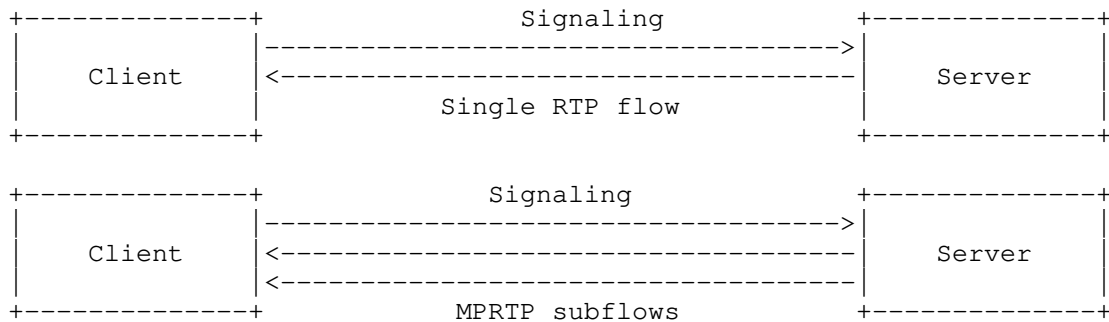


Figure 1: Comparison between traditional RTP streaming and MPRTTP

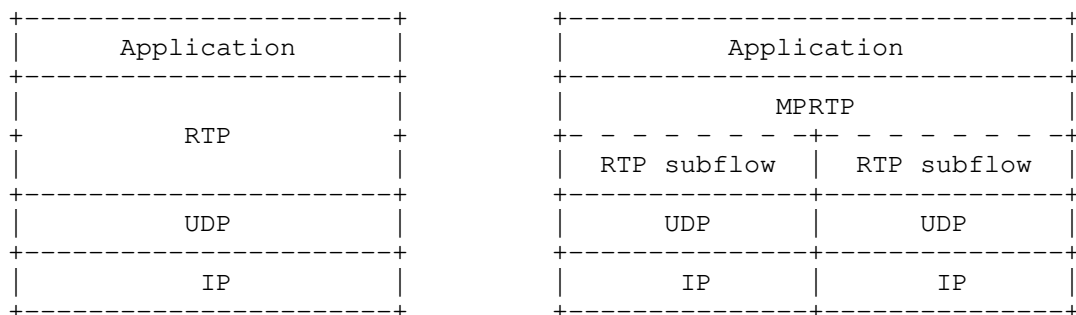


Figure 2: MPRTTP Architecture

Figure 2 illustrates the differences between the standard RTP stack and the MPRTTP stack. MPRTTP receives a normal RTP session from the application and splits it into multiple RTP subflows. Each subflow is then sent along a different path to the receiver. To the network, each subflow appears as an independent, well-formed RTP flow. At the receiver, the subflows are combined to recreate the original RTP session. The MPRTTP layer performs the following functions:

- o **Path Management:** The layer is aware of alternate paths to the other host, which may, for example, be the peer's multiple interfaces. This enables the endpoint to transmit differently marked packets along separate paths. MPRTTP also selects interfaces to send and receive data. Furthermore, it manages the port and IP address pair bindings for each subflow.
- o **Packet Scheduling:** the layer splits a single RTP flow into multiple subflows and sends them across multiple interfaces (paths). The splitting MAY BE done using different path characteristics.

- o Subflow recombination: the layer creates the original stream by recombining the independent subflows. Therefore, the multipath subflows appear as a single RTP stream to applications.

5. Example Media Flow Diagrams

There may be many complex technical scenarios for MPRTTP, however, this memo only considers the following two scenarios: 1) a unidirectional media flow that represents the streaming use-case, and 2) a bidirectional media flow that represents a conversational use-case.

5.1. Streaming use-case

In the unidirectional scenario, the receiver (client) initiates a multimedia session with the sender (server). The receiver or the sender may have multiple interfaces and both endpoints are MPRTTP-capable. Figure 3 shows this scenario. In this case, host A has multiple interfaces. Host B performs connectivity checks on host A's multiple interfaces. If the interfaces are reachable, then host B streams multimedia data along multiple paths to host A. Moreover, host B also sends RTCP Sender Reports (SR) for each subflow and host A responds with a normal RTCP Receiver Report (RR) for the overall session as well as the receiver statistics for each subflow. Host B distributes the packets across the subflows based on the individually measured path characteristics.

Alternatively, to reduce media startup time, host B may start streaming multimedia data to host A's initiating interface and then perform connectivity checks for the other interfaces. This method of updating a single path session to a multipath session is called "multipath session upgrade".

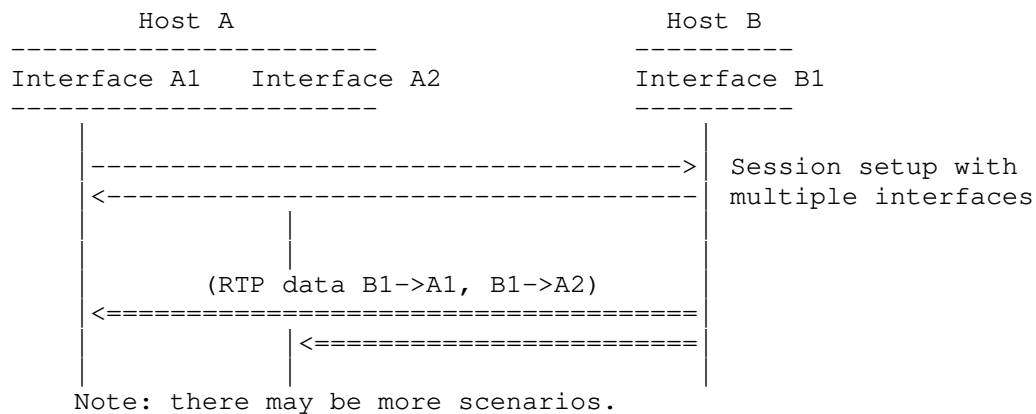


Figure 3: Unidirectional media flow

5.2. Conversational use-case

In the bidirectional scenario, multimedia data flows in both directions. The two hosts exchange their lists of interfaces with each other and perform connectivity checks. Communication begins after each host finds suitable address, port pairs. Interfaces that receive data send back RTCP receiver statistics for that path (based on the 5-tuple). The hosts balance their multimedia stream across multiple paths based on the per path reception statistics and its own volume of traffic. Figure 4 describes an example of a bidirectional flow.

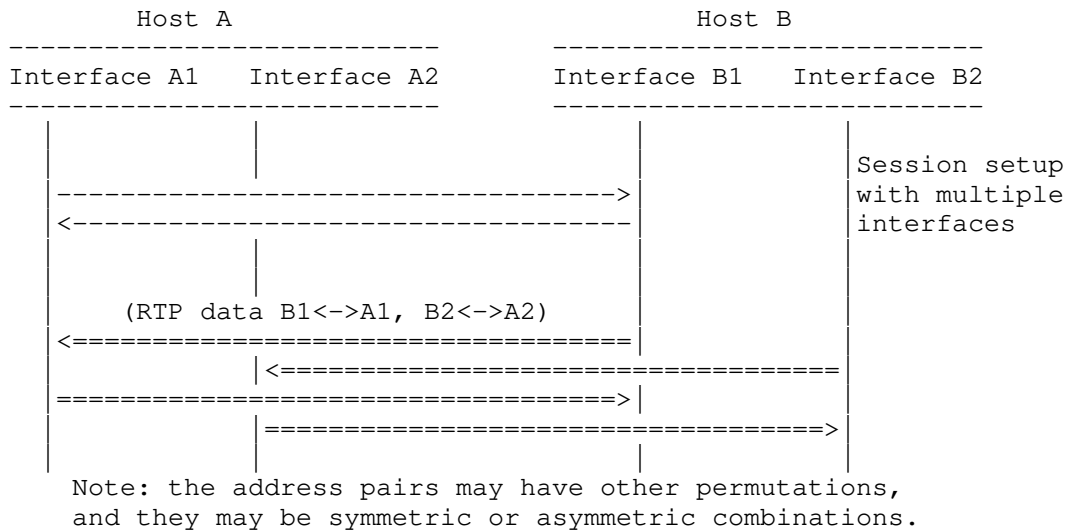


Figure 4: Bidirectional flow

6. MPRTTP Functional Blocks

This section describes some of the functional blocks needed for MPRTTP. We then investigate each block and consider available mechanisms in the next section.

1. **Session Setup:** Interfaces may appear or disappear at anytime during the session. To preserve backward compatibility with legacy applications, a multipath session **MUST** look like a bundle of individual RTP sessions. A multipath session may be upgraded from a typical single path session, as and when new interfaces appear or disappear. However, it is also possible to setup a multipath session from the beginning, if the interfaces are available at the start of the multimedia session.
2. **Expanding RTP:** For a multipath session, each subflow **MUST** look like an independent RTP flow, so that individual RTCP messages can be generated per subflow. Furthermore, MPRTTP splits the single multimedia stream into multiple subflows based on path characteristics (e.g. RTT, loss-rate, receiver rate, bandwidth-delay product etc.) and dynamically adjusts the load on each link.
3. **Adding Interfaces:** Interfaces on the host need to be regularly discovered and advertised. This can be done at session setup and/or during the session. Discovering interfaces is outside the scope of this document.

4. Expanding RTCP: MP RTP MUST provide per path RTCP reports for monitoring the quality of the path, for load balancing, or for congestion control, etc. To maintain backward compatibility with legacy applications, the receiver MUST also send aggregate RTCP reports along with the per-path reports.
 5. Maintenance and Failure Handling: In a multi-homed endpoint interfaces may appear and disappear. If this occurs at the sender, it has to re-adjust the load on the available links. On the other hand, if this occurs at the receiver, then the multimedia data transmitted by the sender to those interfaces is lost. This data may be re-transmitted along a different path i.e., to a different interface on the receiver. Furthermore, the endpoint has to either explicitly signal the disappearance of an interface, or the other endpoint has to detect it (by lack of media packets, RTCP feedback, or keep-alive packets).
 6. Teardown: The MP RTP layer releases the occupied ports on the interfaces.
7. Available Mechanisms within the Functional Blocks

This section discusses some of the possible alternatives for each functional block mentioned in the previous section.

7.1. Session Setup

MP RTP session can be set up in many possible ways e.g., during handshake, or upgraded mid-session. The capability exchange may be done using out-of-band signaling (e.g., Session Description Protocol (SDP) [RFC3264] in Session Initiation Protocol (SIP) [RFC3261], Real-Time Streaming Protocol (RTSP) [I-D.ietf-mmusic-rfc2326bis]) or in-band signaling (e.g., RTP/RTCP header extension, Session Traversal Utilities for NAT (STUN) messages).

7.1.1. Connectivity Checks

The endpoint SHOULD be capable of performing connectivity checks (e.g., using ICE [RFC5245]). If the IP addresses of the endpoints are reachable (e.g., globally addressable, same network etc) then connectivity checks are not needed.

7.1.2. Adding New or Updating Interfaces

Interfaces can appear and disappear during a session, the endpoint MUST be capable of advertising the changes in its set of usable interfaces. Additionally, the application or OS may define a policy on when and/or what interfaces are usable. However, MP RTP requires a

method to advertise or notify the other endpoint about the updated set of usable interfaces.

7.1.3. In-band vs. Out-of-band Signaling

MTRTP nodes will generally use a signaling protocol to establish their MPRTTP session. With the existence of such a signaling relationship, two alternatives become available to exchange information about the available interfaces on each side for extending RTP sessions to MPRTTP and for modifying MPRTTP sessions: in-band and out-of-band signaling.

In-band signaling refers to using mechanisms of RTP/RTCP itself to communicate interface addresses, e.g., a dedicated RTCP extensions along the lines of the one defined to communicate information about the feedback target for RTP over SSM [RFC5760]. In-band signaling does not rely on the availability of a separate signaling connection and the information flows along the same path as the media streams, thus minimizing dependencies. Moreover, if the media channel is secured (e.g., by means of SRTP), the signaling is implicitly protected as well if SRTCP encryption and authentication are chosen. In-band signaling is also expected to take a direct path to the peer, avoiding any signaling overlay-induced indirections and associated processing overheads in signaling elements -- avoiding such may be especially worthwhile if frequent updates may occur as in the case of mobile users. Finally, RTCP is usually sent sufficiently frequently (in point-to-point settings) to provide enough opportunities for transmission and (in case of loss) retransmission of the corresponding RTCP packets.

Examples for in-band signaling include RTCP extensions as noted above or suitable extensions to STUN [I-D.wing-mmusic-ice-mobility].

Out-of-band signaling refers to using a separate signaling connection (via SIP, RTSP, or HTTP) to exchange interface information, e.g., expressed in SDP. Clear benefits are that signaling occurs at setup time anyway and that experience and SDP syntax (and procedures) are available that can be re-used or easily adapted to provide the necessary capabilities. In contrast to RTCP, SDP offers a reliable communication channel so that no separate retransmissions logic is necessary. In SDP, especially when combined with ICE, connectivity check mechanisms including sophisticated rules are readily available. While SDP is not inherently protected, suitable security may need to be applied anyway to the basic session setup.

Examples for out-of-band signaling are dedicated extensions to SDP; those may be combined with ICE.

Both types of mechanisms have their pros and cons for middleboxes. With in-band signaling, control packets take the same path as the media packets and they can be directly inspected by middleboxes so that the elements operating on the signaling channel do not need to understand new SDP. With out-of-band signaling, only the middleboxes processing the signaling need to be modified and those on the data forwarding path can remain untouched.

Overall, it may appear sensible to provide a combination of both mechanisms: out-of-band signaling for session setup and initial interface negotiation and in-band signaling to deal with frequent changes in interface state (and for the potential case, albeit rather theoretical case of MP RTP communication without a signaling channel).

In its present version, this document explores both options to provide a broad understanding of how the corresponding mechanisms would look like.

7.2. Expanding RTP

RTCP [RFC3550] is generated per media session. However, with MP RTP, the media sender spreads the RTP load across several interfaces. The media sender SHOULD make the path selection, load balancing and fault tolerance decisions based on the characteristics of each path. Therefore, apart from normal RTP sequence numbers defined in [RFC3550], the MP RTP sender MUST add subflow-specific sequence numbers to help calculate fractional losses, jitter, RTT, playout time, etc., for each path, and a subflow identifier to associate the characteristics with a path. The RTP header extension for MP RTP is shown in Section 9.1.

7.3. Expanding RTCP

To provide accurate per path information an MP RTP endpoint MUST send (SR/RR) report for each unique subflow along with the overall session RTCP report. Therefore, the additional subflow reporting affects the RTCP bandwidth and the RTCP reporting interval. RTCP report scheduling for each subflow may cause a problem for RTCP recombination and reconstruction in cases when 1) RTCP for a subflow is lost, and 2) RTCP for a subflow arrives later than the other subflows. (There may be other cases as well.)

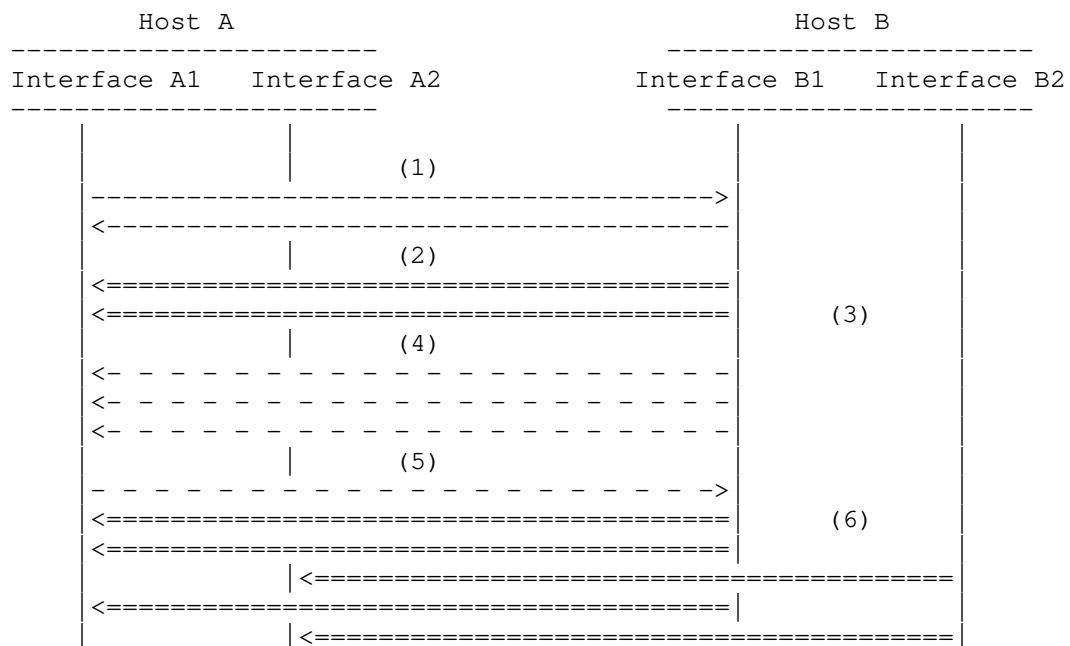
The sender distributes the media across different paths using the per path RTCP reports. However, this document doesn't cover algorithms for congestion control or load balancing.

7.4. Failure Handling and Teardown

An MPRTTP endpoint MUST keep alive subflows that have been negotiated but no media is sent on them. Moreover, using the information in the subflow reports, a sender can monitor an active subflow for failure (errors, unreachability, congestion) and decide not to use (make the active subflow passive), or teardown the subflow.

If an interface disappears, the endpoint MUST send an updated interface advertisement without the interface and release the the associated subflows.

8. MPRTTP Protocol



Key:

| Interface

---> Signaling Protocol

<=== RTP Packets

- -> RTCP Packet

Figure 5: MPRTTP New Interface

8.1. Overview

The bullet points explain the different steps shown in Figure 5 for upgrading a single path multimedia session to multipath session.

(1) The first two interactions between the hosts represents the establishment of a normal RTP session. This may performed e.g. using SIP or RTSP.

(2) When the RTP session has been established, host B streams media using its interface B1 to host A at interface A1.

(3) Host B supports sending media using MP RTP and becomes aware of an additional interface B2.

(4) Host B advertises the multiple interface addresses.

(5) Host A supports receiving media using MP RTP and becomes aware of an additional interface A2.

Side note, even if an MP RTP-capable host has only one interface, it MUST respond to the advertisement with its single interface.

(6) Each host receives information about the additional interfaces and the appropriate endpoints starts to stream the multimedia content using the additional paths.

If needed, each endpoint will need to independently perform connectivity checks (not shown in figure) and ascertain reachability before using the paths.

8.1.1. Gather or Discovering Candidates

The endpoint periodically polls the operating system or is notified when an additional interface appears. If the endpoint wants to use the additional interface for MP RTP it MUST advertise it to the other peers. The endpoint may also use ICE [RFC5245] to gather additional candidates.

8.1.2. NAT Traversal

After gathering their interface candidates, the endpoints decide internally if they wish to perform connectivity checks.

[note-iceornot]

If the endpoint chooses to perform connectivity checks then it MUST first advertise the gathered candidates as ICE candidates in SDP

during session setup and let ICE perform the connectivity checks. As soon as a sufficient number of connectivity checks succeed, the endpoint can use the successful candidates to advertise its MPRTTP interface candidates.

Alternatively, if the endpoint supports mobility extensions for ICE [I-D.wing-mmusic-ice-mobility], it can let the ICE agent gather and perform the connectivity checks. When the connectivity checks succeed, the ICE agent should notify the MPRTTP layer of these new paths (5-tuples), these new paths are then used by MPRTTP to send media packets depending on the scheduling algorithm.

If an endpoint supports Interface selection via PCP Flow Extension [I-D.reddy-mmusic-ice-best-interface-pcp], it will receive notifications when new interfaces become available and additionally when the link quality of a currently available interface changes. The application can advertise and perform connectivity checks with the new interface and in the case of changes in link quality pass the information to the scheduling algorithm for better application performance.

[Editors note: The interaction between the RTP agent and ICE agent is needed for implementing a scheduling algorithm or congestion control. See details of a scheduling algorithm in [ACM-MPRTTP].]

8.1.3. Choosing between In-band (in RTCP) and Out-of-band (in SDP) Interface Advertisement

If there is no media flowing at the moment and the application wants to use the interfaces from the start of the session, it should advertise them in SDP (See [I-D.singh-mmusic-mprtp-sdp-extension]). Alternatively, the endpoint can setup the session as a single path media session and upgrade the session to multipath by advertising the session in-band (See Section 8.1.4 or [I-D.wing-mmusic-ice-mobility]). Moreover, if an interface appears and disappears, the endpoint SHOULD prefer to advertise it in-band because the endpoint would not have to wait for a response from the other endpoint before starting to use the interface. However, if there is a conflict between an in-band and out-of-band advertisement, i.e., the endpoint receives an in-band advertisement while it has a pending out-of-band advertisement, or vice versa then the session is setup using out-of-band mechanisms.

8.1.4. In-band Interface Advertisement

To advertise the multiple interfaces in RTCP, an MPRTTP-capable endpoint MUST add the MPRTTP Interface Advertisement defined in Figure 13 with the RTCP Sender Report (SR). Each unique address is

encapsulated in an Interface Advertisement block and contains the IP address, RTP and RTCP port addresses. The Interface Advertisement blocks are ordered based on a decreasing priority level. On receiving the MPRTTP Interface Advertisement, an MPRTTP-capable receiver MUST respond with the set of interfaces (subset or all available) it wants to use.

If the sender and receiver have only one interface, then the endpoints MUST indicate the negotiated single path IP, RTP port and RTCP port addresses.

8.1.5. Subflow ID Assignment

After interface advertisements have been exchanged, the endpoint MUST associate a Subflow ID to each unique subflow. Each combination of sender and receiver IP addresses and port pairs (5-tuple) is a unique subflow. If the connectivity checks have been performed then the endpoint MUST only use the subflows for which the connectivity checks have succeeded.

8.1.6. Active and Passive Subflows

To send and receive data an endpoint MAY use any number of subflows from the set of available subflows. The subflows that carry media data are called active subflows, while those subflows that don't send any media packets (fallback paths) are called passive subflows.

An endpoint MUST multiplex the subflow specific RTP and RTCP packets on the same port to keep the NAT bindings alive. This is inline with the recommendation made in RFC6263[RFC6263]. Moreover, if an endpoint uses ICE, multiplexing RTP and RTCP reduces the number of components from 2 to 1 per media stream. If no MPRTTP or MPRTCP packets are received on a particular subflow at a receiver, the receiver SHOULD remove the subflow from active and passive lists and not send any MPRTCP reports for that subflow. It may keep the bindings in a dead-pool, so that if the 5-tuple or subflow reappears, it can quickly reallocate it based on past history.

8.2. RTP Transmission

If both endpoints are MPRTTP-capable and if they want to use their multiple interfaces for sending the media stream then they MUST use the MPRTTP header extensions. They MAY use normal RTP with legacy endpoints (see Appendix A).

An MPRTTP endpoint sends RTP packets with an MPRTTP extension that maps the media packet to a specific subflow (see Figure 8). The MPRTTP layer SHOULD associate an RTP packet with a subflow based on a

scheduling strategy. The scheduling strategy may either choose to augment the paths to create higher throughput or use the alternate paths for enhancing resilience or error-repair. Due to the changes in path characteristics, the endpoint should be able change its scheduling strategy during an ongoing session. The MP RTP sender **MUST** also populate the subflow specific fields described in the MP RTP extension header (see Section 9.1.1).

[ACM-MP RTP] describes and evaluates a scheduling algorithm for video over multiple interfaces. The video is encoded at a single target bit rate and is evaluated in various network scenarios, as discussed in [I-D.ietf-rmcat-eval-criteria].

8.3. Playout Considerations at the Receiver

A media receiver, irrespective of MP RTP support or not, should be able to playback the media stream because the received RTP packets are compliant to [RFC3550], i.e., a non-MP RTP receiver will ignore the MP RTP header and still be able to playback the RTP packets. However, the variation of jitter and loss per path may affect proper playout. The receiver can compensate for the jitter by modifying the playout delay (i.e., by calculating skew across all paths) of the received RTP packets. For example, an adaptive playout algorithm is discussed in [ACM-MP RTP].

8.4. Subflow-specific RTCP Statistics and RTCP Aggregation

Aggregate RTCP provides the overall media statistics and follows the normal RTCP defined in RFC3550 [RFC3550]. However, subflow specific RTCP provides the per path media statistics because the aggregate RTCP report may not provide sufficient per path information to an MP RTP scheduler. Specifically, the scheduler should be aware of each path's RTT and loss-rate, which an aggregate RTCP cannot provide.

The aggregate RTCP report (i.e., the regularly scheduled RTCP report) **MUST** be sent compounded as described in [RFC5506], however, the subflow-specific RTCP reports **MAY** be sent non-compounded. Further, each subflow and the aggregate RTCP report **MUST** follow the timing rules defined in [RFC4585].

The RTCP reporting interval is locally implemented and the scheduling of the RTCP reports may depend on the the behavior of each path. For instance, the RTCP interval may be different for a passive path than an active path to keep port bindings alive. Additionally, an endpoint may decide to share the RTCP reporting bit rate equally across all its paths or schedule based on the receiver rate on each path.

8.5. RTCP Transmission

The sender sends an RTCP SR on each active path. For each SR the receiver gets, it echoes one back to the same IP address-port pair that sent the SR. The receiver tries to choose the symmetric path and if the routing is symmetric then the per-path RTT calculations will work out correctly. However, even if the paths are not symmetric, the sender would at maximum, under-estimate the RTT of the path by a factor of half of the actual path RTT.

9. Packet Formats

In this section we define the protocol structures described in the previous sections.

9.1. RTP Header Extension for MPRTTP

The MPRTTP header extension is used to distribute a single RTP stream over multiple subflows.

The header conforms to the one-byte RTP header extension defined in [RFC5285]. The header extension contains a 16-bit length field that counts the number of 32-bit words in the extension, excluding the four-octet extension header (therefore zero is a valid length, see Section 5.3.1 of [RFC3550] for details).

The RTP header for each subflow is defined below:

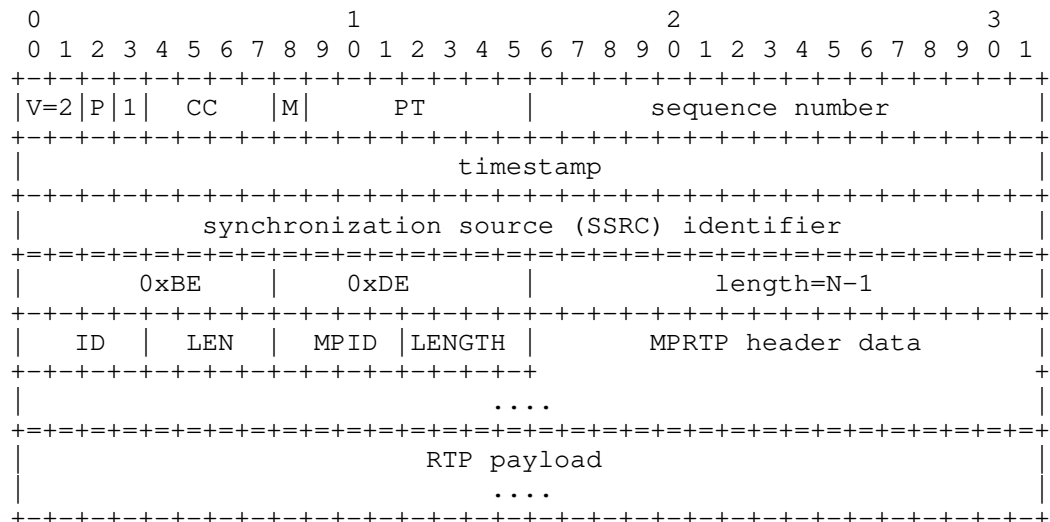


Figure 6: Generic MPRTTP header extension

MPID:

The MPID field corresponds to the type of MP RTP packet.
Namely:

MPID ID Value	Use
0x0	Subflow RTP Header. For this case the Length is set to 4

Figure 7: RTP header extension values for MP RTP (H-Ext ID)

length

The 4-bit length field is the length of extension data in bytes not including the H-Ext ID and length fields. The value zero indicates there is no data following.

MP RTP header data

Carries the MPID specific data as described in the following sub-sections.

9.1.1. MP RTP RTP Extension for a Subflow

The RTP header for each subflow is defined below:

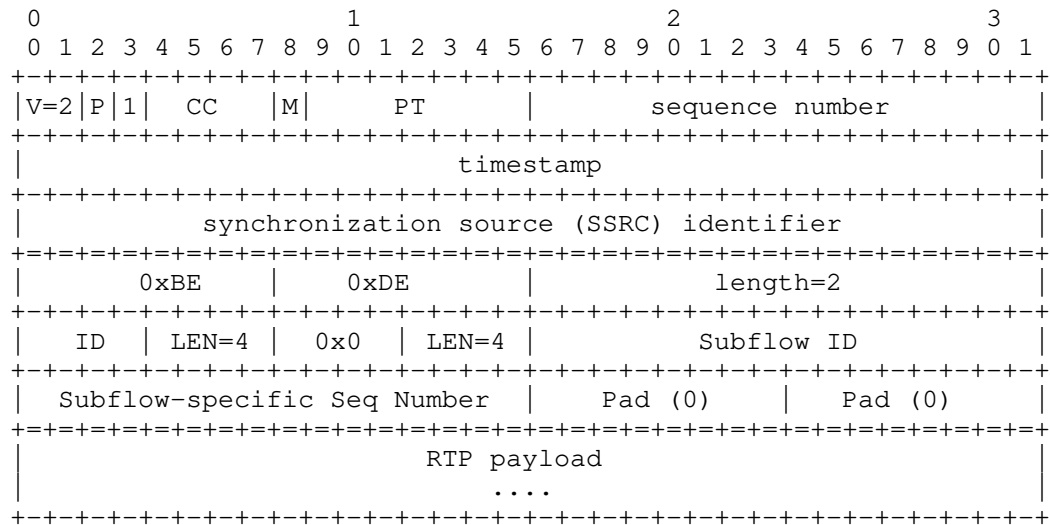


Figure 8: MPRTP header for subflow

MP ID = 0x0

Indicates that the MPRTP header extension carries subflow specific information.

length = 4

Subflow ID: Identifier of the subflow. Every RTP packet belonging to the same subflow carries the same unique subflow identifier.

Flow-Specific Sequence Number (FSSN): Sequence of the packet in the subflow. Each subflow has its own strictly monotonically increasing sequence number space.

9.2. RTCP Extension for MPRTP (MPRTCP)

The MPRTP RTCP header extension is used to 1) provide RTCP feedback per subflow to determine the characteristics of each path, and 2) advertise each interface.

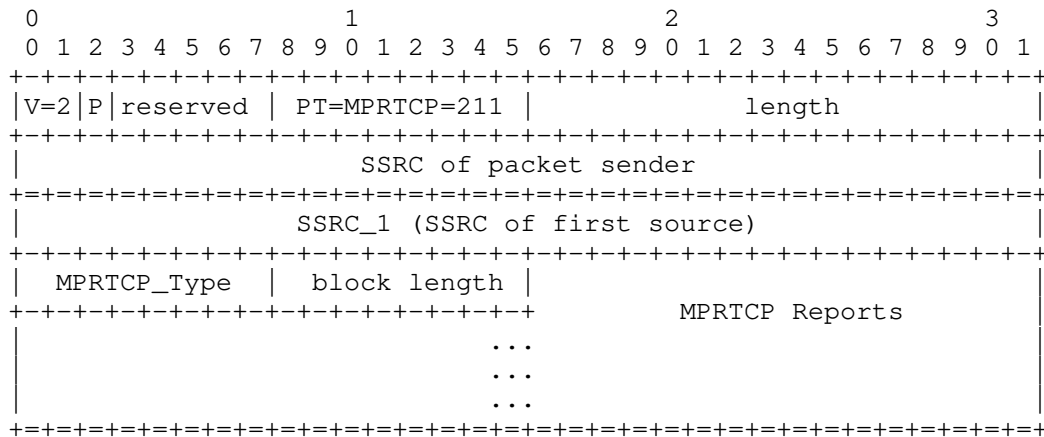


Figure 9: Generic RTCP Extension for MPRTCP (MPRTCP) [appended to normal SR/RR]

MPRTCP: 8 bits

Contains the constant 211 to identify this as an Multipath RTCP packet.

length: 16 bits

As described for the RTCP packet (see Section 6.4.1 of the RTP specification [RFC3550]), the length of this is in 32-bit words minus one, including the header and any padding.

MPRTCP_Type: 8-bits

The MPRTCP_Type field corresponds to the type of MPRTCP RTCP packet. Namely:

MPRTCP_Type Value	Use
0	Subflow Specific Report
1	Interface Advertisement (IPv4 Address)
2	Interface Advertisement (IPv4 Address)
3	Interface Advertisement (DNS Address)

Figure 10: RTP header extension values for MPRTCP (MPRTCP_Type)

block length: 8-bits

The 8-bit length field is the length of the encapsulated MPRTCP reports in 32-bit word length (i.e., length * 4 bytes) including the MPRTCP_Type and length fields. The value zero is invalid and the report block MUST be ignored.

MPRTCP Reports: variable size

Defined later in 9.2.1 and 9.3.

9.2.1. MPRTCP Extension for Subflow Reporting

When sending a report for a specific subflow the sender or receiver MUST add only the reports associated with that 5-tuple. Each subflow is reported independently using the following MPRTCP Feedback header.

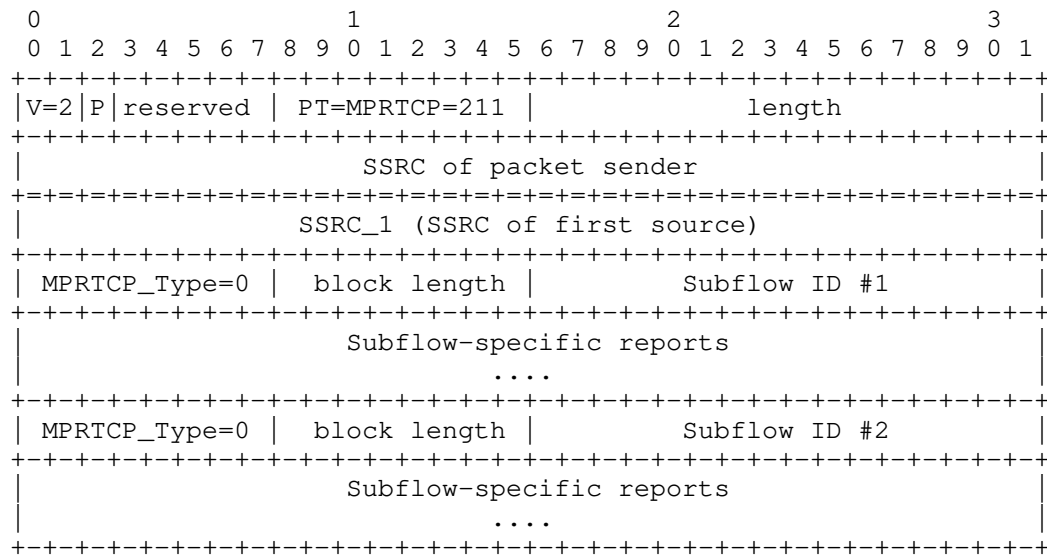


Figure 11: MPRTCP Subflow Reporting Header

MPRTCP_Type: 0

The value indicates that the encapsulated block is a subflow report.

block length: 8-bits

The 8-bit length field is the length of the encapsulated subflow-specific reports in 32-bit word length not including the MPRTCP_Type and length fields.

Subflow ID: 16 bits

Subflow identifier is the value associated with the subflow the endpoint is reporting about. If it is a sender it MUST use the Subflow ID associated with the 5-tuple. If it is a receiver it MUST use the Subflow ID received in the Subflow-specific Sender Report.

Subflow-specific reports: variable

Subflow-specific report contains all the reports associated with the Subflow ID. For a sender, it MUST include the Subflow-specific Sender Report (SSR). For a receiver, it MUST include Subflow-specific Receiver Report (SRR). Additionally, if the receiver supports subflow-specific extension reports then it MUST append them to the SRR.

9.2.1.1. MPRTCP for Subflow-specific SR, RR and XR

[note-rtcp-reuse]

Example:

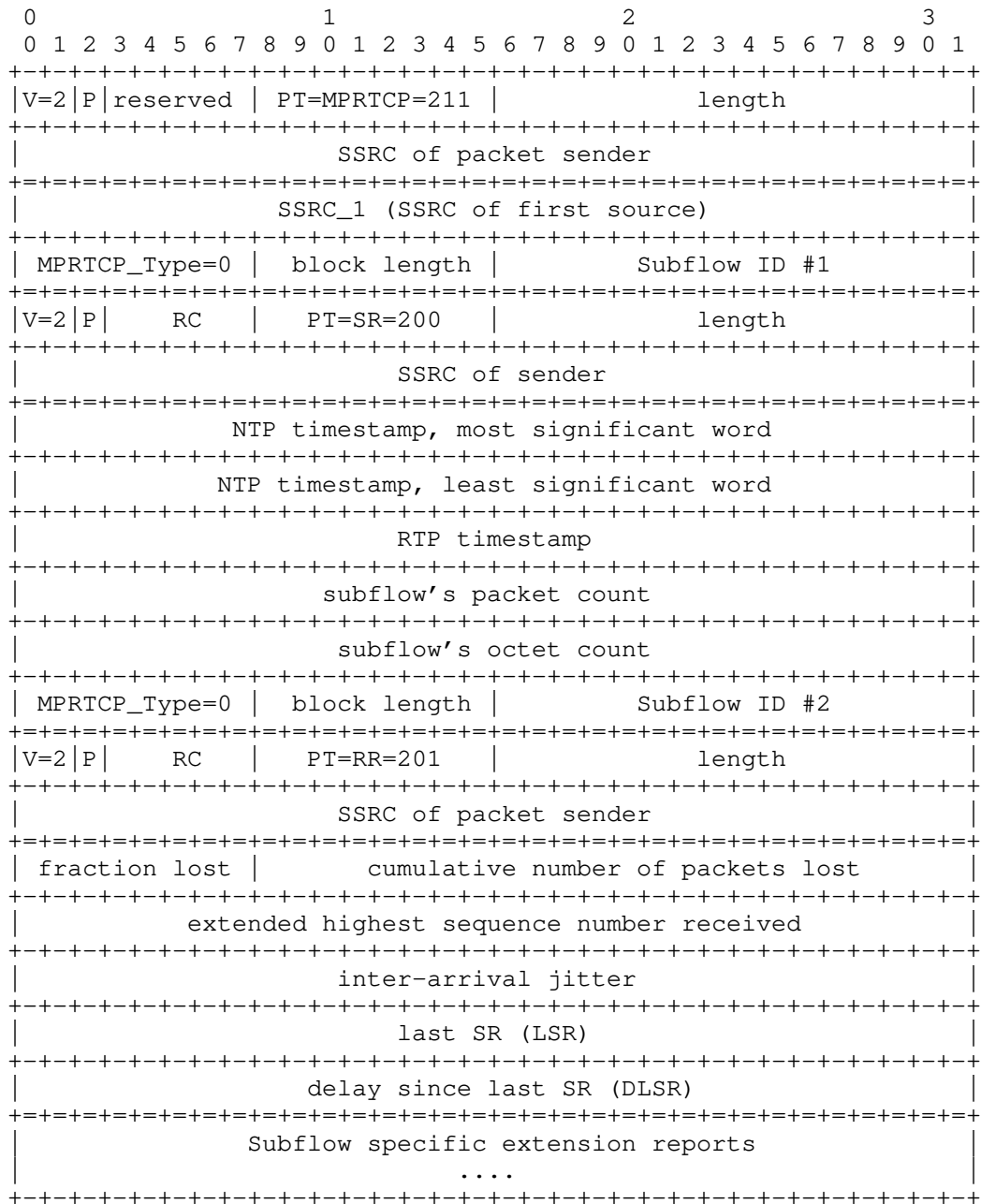


Figure 12: Example of reusing RTCP SR and RR inside an MPRTCP header (Bi-directional use-case, in case of uni-directional flow the subflow will only send an SR or RR).

9.3. MPRTCP Extension for Interface advertisement

This sub-section defines the RTCP header extension for in-band interface advertisement by the receiver. The interface advertisement block describes a method to represent IPv4, IPv6 and generic DNS-type addresses in a block format. It is based on the sub-reporting block in [RFC5760]. The interface advertisement SHOULD immediately follow the Receiver Report. If the Receiver Report is not present, then it MUST be appended to the Sender Report.

The endpoint MUST advertise the interfaces it wants to use whenever an interface appears or disappears and also when it receives an Interface Advertisement.

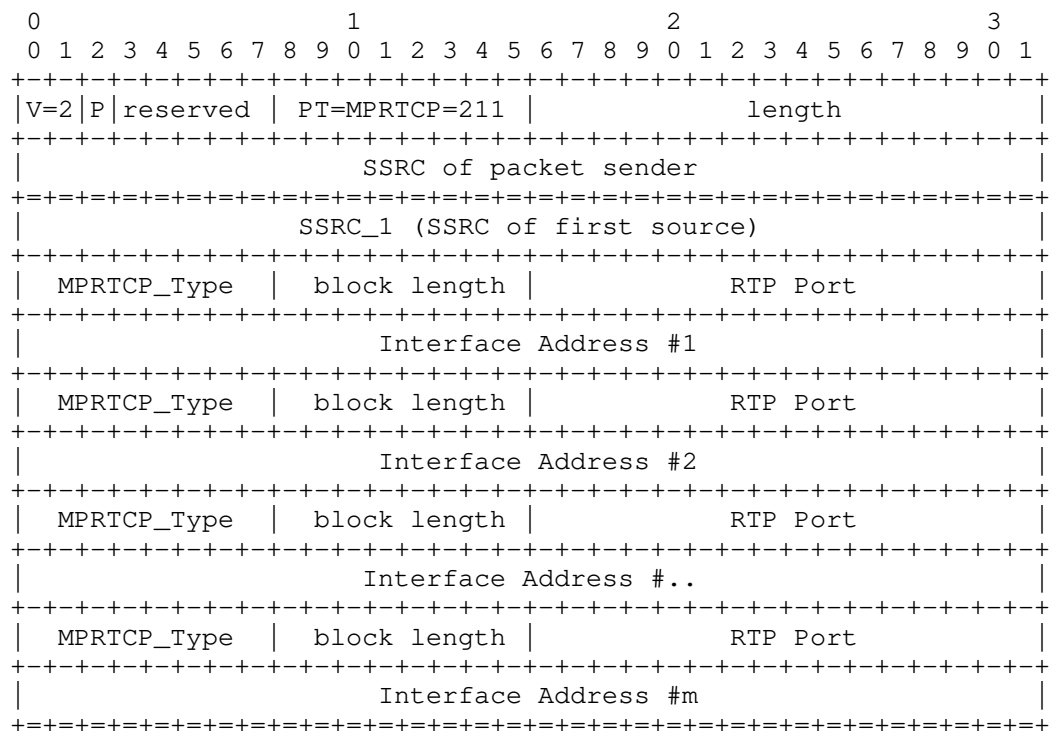


Figure 13: MPRTCP Interface Advertisement. (appended to SR/RR)

MPRTCP_Type: 8 bits

The MPRTCP_Type corresponds to the type of interface address.
Namely:

1: IPv4 address

2: IPv6 address

3: DNS name

block length: 8 bits

The length of the Interface Advertisement block in 32-bit word lengths.

For an IPv4 address, this should be 2 (8 octets including = 2 + 2 + 4).

For an IPv6 address, this should be 5 (20 octets = 2 + 2 + 16).

For a DNS name, the length field indicates the number of word lengths making up the address string plus 1 (32-bit word for the 2 byte port number and 2 byte header).

RTP Port: 2 octets

The port number to which the sender sends RTP data. A port number of 0 is invalid and MUST NOT be used.

Interface Address: 4 octets (IPv4), 16 octets (IPv6), or n octets (DNS name)

The address to which receivers send feedback reports. For IPv4 and IPv6, fixed-length address fields are used. A DNS name is an arbitrary-length string. The string MAY contain Internationalizing Domain Names in Applications (IDNA) domain names and MUST be UTF-8 [RFC3629] encoded.

10. RTCP Timing reconsiderations for MPRTCP

MPRTP endpoints MUST conform to the timing rule imposed in [RFC4585], i.e., the total RTCP rate between the participants MUST NOT exceed 5% of the media rate. For each endpoint, a subflow MUST send the aggregate and subflow-specific report. The endpoint SHOULD schedule the RTCP reports for the active subflows based on the share of the transmitted or received bit rate to the average media bit rate, this method ensures fair sharing of the RTCP bandwidth. Alternatively, the endpoint MAY schedule the reports in round-robin.

11. SDP Considerations

11.1. Signaling MPRTTP Header Extension in SDP

To indicate the use of the MPRTTP header extensions (see Section 9) in SDP, the sender MUST use the following URI in extmap: urn:ietf:params:rtp-hdext:mprttp. This is a media level parameter. Legacy RTP (non-MPRTTP) clients will ignore this header extension, but can continue to parse and decode the packet (see Appendix A).

Example:

```
v=0
o=alice 2890844526 2890844527 IN IP4 192.0.2.1
s=
c=IN IP4 192.0.2.1
t=0 0
m=video 49170 RTP/AVP 98
a=rtpmap:98 H264/90000
a=fmtp:98 profile-level-id=42A01E;
a=extmap:1 urn:ietf:params:rtp-hdext:mprttp
```

11.2. Signaling MPRTTP capability in SDP

A participant of a media session MUST use SDP to indicate that it supports MPRTTP. Not providing this information will make the other endpoint ignore the RTCP extensions.

```
mprttp-attr = "a=" "mprttp" [
    SP mprttp-optional-parameter]
CRLF ; flag to enable MPRTTP
```

The endpoint MUST use 'a=mprttp', if it is able to send and receive MPRTTP packets. Generally, senders and receivers MUST indicate this capability if they support MPRTTP and would like to use it in the specific media session being signaled. To exchange the additional interfaces, the endpoint SHOULD use the in-band signaling (See Section 9.3). Alternatively, advertise in SDP (See [I-D.singh-mmusic-mprttp-sdp-extension]).

MPRTTP endpoint multiplexes RTP and RTCP on a single port, sender MUST indicate support by adding "a=rtcp-mux" in SDP [RFC5761]. If an endpoint receives an SDP without "a=rtcp-mux" but contains "a=mprttp", then the endpoint MUST infer support for multiplexing.

MPRTTP endpoint uses Reduced-Size RTCP [RFC5506] for reporting path characteristics per subflow (MPRTCP). An MPRTTP endpoint MUST indicate support by adding "a=rtcp-rsize" in SDP [RFC5761]. If an

endpoint receives an "a=rtcp-rsize" but contains "a=mp RTP", then the endpoint MUST infer support for Reduced-Size RTCP.

[note-rtp-rtcp-mux]

11.3. MP RTP with ICE

If the endpoints intend to use ICE [RFC5245] for discovering interfaces and running connectivity checks then the endpoint MUST advertise its ICE candidates in the initial OFFER, as defined in ICE [RFC5245]. Thereafter the endpoints run connectivity checks.

When an endpoint uses ICE's regular nomination [RFC5245] procedure, it chooses the best ICE candidate as the default path. In the case of an MP RTP endpoint, if more than one ICE candidate succeeded the connectivity checks then an MP RTP endpoint MAY advertise (some of) these in-band in RTCP as MP RTP interfaces.

When an endpoint uses ICE's aggressive nomination [RFC5245] procedure, the selected candidate may change as more ICE checks complete. Instead of sending updated offers as additional ICE candidates appear (transience), the endpoint it MAY use in-band signaling to advertise its interfaces, as defined in Section 9.3.

If the default interface disappears and the paths used for MP RTP are different from the one in the c= and m= lines then the an alternate interface for which the ICE checks were successful should be promoted to the c= and m= lines in the updated offer.

When a new interface appears then the application/endpoint should internally decide if it wishes to use it and sends an updated offer with ICE candidates of the all its interfaces including the new interface. The receiving endpoint responds to the offer with all its ICE candidates in the answer and starts connectivity checks between all its candidates and the offerer's ICE candidates. Similarly, the initiating endpoint starts connectivity checks between the its interfaces (incl. the new interface) and all the received ICE candidates in the answer. If the connectivity checks succeed, the initiating endpoint MAY use in-band signaling (See Section 9.3) to advertise its new set of interfaces.

11.4. Increased Throughput

The MP RTP layer MAY choose to augment paths to increase throughput. If the desired media rate exceeds the current media rate, the endpoints MUST renegotiate the application specific ("b=AS:xxx") [RFC4566] bandwidth.

11.5. Offer/Answer

When SDP [RFC4566] is used to negotiate MPRTTP sessions following the offer/answer model [RFC3264], the "a=mp RTP" attribute (see Section 11.2) indicates the desire to use multiple interfaces to send or receive media data. The initial SDP offer MUST include this attribute at the media level. If the answerer wishes to also use MPRTTP, it MUST include a media-level "a=mp RTP" attribute in the answer. If the answer does not contain an "a=mp RTP" attribute, the offerer MUST NOT stream media over multiple paths and the offerer MUST NOT advertise additional MPRTTP interfaces in-band or out-of-band.

When SDP is used in a declarative manner, the presence of an "a=mp RTP" attribute signals that the sender can send or receive media data over multiple interfaces. The receiver SHOULD be capable to stream media to the multiple interfaces and be prepared to receive media from multiple interfaces.

The following sections shows examples of SDP offer and answer for in-band and out-of-band signaling.

11.5.1. In-band Signaling Example

The following offer/answer shows that both the endpoints are MPRTTP capable and SHOULD use in-band signaling for interfaces advertisements.

Offer:

```
v=0
o=alice 2890844526 2890844527 IN IP4 192.0.2.1
s=
c=IN IP4 192.0.2.1
t=0 0
m=video 49170 RTP/AVP 98
a=rtpmap:98 H264/90000
a=fmtp:98 profile-level-id=42A01E;
a=rtcp-mux
a=mp RTP
```

Answer:

```
v=0
o=bob 2890844528 2890844529 IN IP4 192.0.2.2
s=
c=IN IP4 192.0.2.2
t=0 0
m=video 4000 RTP/AVP 98
a=rtpmap:98 H264/90000
a=fmtp:98 profile-level-id=42A01E;
a=rtcp-mux
a=mprtp
```

The endpoint MAY now use in-band RTCP signaling to advertise its multiple interfaces. Alternatively, it MAY make another offer with the interfaces in SDP (out-of-band signaling) [I-D.singh-mmusic-mprtp-sdp-extension].

12. IANA Considerations

The following contact information shall be used for all registrations in this document:

```
Contact:    Varun Singh
            mailto:varun.singh@iki.fi
            tel:+358-9-470-24785
```

Note to the RFC-Editor: When publishing this document as an RFC, please replace "RFC XXXX" with the actual RFC number of this document and delete this sentence.

12.1. MPRTCP Header Extension

This document defines a new extension URI to the RTP Compact Header Extensions sub-registry of the Real-Time Transport Protocol (RTP) Parameters registry, according to the following data:

```
Extension URI: urn:ietf:params:rtp-hdext:mprtp
Description:  Multipath RTP
Reference:    RFC XXXX
```

12.2. MPRTCP Packet Type

A new RTCP packet format has been registered with the RTCP Control Packet type (PT) Registry:

Name: MPRTCP
Long name: Multipath RTCP
Value: 211
Reference: RFC XXXX.

This document defines a substructure for MPRTCP packets. A new sub-registry has been set up for the sub-report block type (MPRTCP_Type) values for the MPRTCP packet, with the following registrations created initially:

Name: Subflow Specific Report
Long name: Multipath RTP Subflow Specific Report
Value: 0
Reference: RFC XXXX.

Name: IPv4 Address
Long name: IPv4 Interface Address
Value: 1
Reference: RFC XXXX.

Name: IPv6 Address
Long name: IPv6 Interface Address
Value: 2
Reference: RFC XXXX.

Name: DNS Name
Long name: DNS Name indicating Interface Address
Value: 3
Reference: RFC XXXX.

Further values may be registered on a first come, first served basis. For each new registration, it is mandatory that a permanent, stable, and publicly accessible document exists that specifies the semantics of the registered parameter as well as the syntax and semantics of the associated sub-report block. The general registration procedures of [RFC4566] apply.

12.3. SDP Attributes

This document defines a new SDP attribute, "mprtp", within the existing IANA registry of SDP Parameters.

12.3.1. "mprtp" attribute

- o Attribute Name: MPRTCP
- o Long Form: Multipath RTP

- o Type of Attribute: media-level
- o Charset Considerations: The attribute is not subject to the charset attribute.
- o Purpose: This attribute is used to indicate support for Multipath RTP. It can also provide one of many possible interfaces for communication. These interface addresses may have been validated using ICE procedures.
- o Appropriate Values: See Section 11.2 of RFC XXXX.

13. Security Considerations

TBD

All drafts are required to have a security considerations section. See RFC 3552 [RFC3552] for a guide.

14. Acknowledgements

Varun Singh, Saba Ahsan, and Teemu Karkkainen are supported by Trilogy (<http://www.trilogy-project.org>), a research project (ICT-216372) partially funded by the European Community under its Seventh Framework Program. The views expressed here are those of the author(s) only. The European Commission is not liable for any use that may be made of the information in this document.

The work of Varun Singh and Joerg Ott from Aalto University has been partially supported by the European Institute of Innovation and Technology (EIT) ICT Labs activity RCLD 11882.

Thanks to Roni Even , Miguel A. Garcia , Ralf Globisch , Christer Holmberg , and Frederic Maze for providing valuable feedback on earlier versions of this draft.

15. References

15.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

- [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.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.
- [RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, July 2008.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.
- [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, July 2006.
- [RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.

15.2. Informative References

- [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.
- [RFC6182] Ford, A., Raiciu, C., Handley, M., Barre, S., and J. Iyengar, "Architectural Guidelines for Multipath TCP Development", RFC 6182, March 2011.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.
- [RFC5533] Nordmark, E. and M. Bagnulo, "Shim6: Level 3 Multihoming Shim Protocol for IPv6", RFC 5533, June 2009.
- [RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, January 2008.

- [I-D.ietf-mmusic-rfc2326bis]
Schulzrinne, H., Rao, A., Lanphier, R., Westerlund, M.,
and M. Stiemerling, "Real Time Streaming Protocol 2.0
(RTSP)", draft-ietf-mmusic-rfc2326bis-40 (work in
progress), February 2014.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston,
A., Peterson, J., Sparks, R., Handley, M., and E.
Schooler, "SIP: Session Initiation Protocol", RFC 3261,
June 2002.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model
with Session Description Protocol (SDP)", RFC 3264, June
2002.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session
Description Protocol", RFC 4566, July 2006.
- [RFC6263] Marjou, X. and A. Sollaud, "Application Mechanism for
Keeping Alive the NAT Mappings Associated with RTP / RTP
Control Protocol (RTCP) Flows", RFC 6263, June 2011.
- [I-D.singh-mmusic-mprtp-sdp-extension]
Singh, V., Ott, J., Karkkainen, T., Globisch, R., and T.
Schierl, "Multipath RTP (MPRTP) attribute in Session
Description Protocol", draft-singh-mmusic-mprtp-sdp-
extension-03 (work in progress), January 2014.
- [I-D.reddy-mmusic-ice-best-interface-pcp]
Reddy, T., Wing, D., Steeg, B., Penno, R., and V. Varun,
"Improving ICE Interface Selection Using Port Control
Protocol (PCP) Flow Extension", draft-reddy-mmusic-ice-
best-interface-pcp-00 (work in progress), October 2013.
- [I-D.wing-mmusic-ice-mobility]
Wing, D., Reddy, T., Patil, P., and P. Martinsen,
"Mobility with ICE (MICE)", draft-wing-mmusic-ice-
mobility-06 (work in progress), February 2014.
- [I-D.ietf-rmcat-eval-criteria]
Singh, V. and J. Ott, "Evaluating Congestion Control for
Interactive Real-time Media", draft-ietf-rmcat-eval-
criteria-00 (work in progress), January 2014.
- [ACM-MPRTTP]
Singh, V., Ahsan, S., and J. Ott, "MPRTTP: multipath
considerations for real-time media", in Proc. of ACM
Multimedia Systems, MMSys, 2013.

Appendix A. Interoperating with Legacy Applications

Some legacy endpoints may abort processing incoming packets, if they are received from different source address. This may occur due to the loop detection algorithm. Additionally, it is also possible for the receiver to reset processing of the stream if the jump in packet sequence numbers received over multiple interface is large. Both of these errors are based on implementation-specific details.

An MPRTTP sender can use its multiple interfaces to send media to a legacy RTP client. The legacy receiver will ignore the subflow RTP header extension and the receiver's de-jitter buffer will attempt to compensate for any mismatch in per-path latency. However, the receiver will only send the overall or aggregate RTCP report which may be insufficient for an MPRTTP sender to adequately schedule packets over multiple paths or detect if a path disappeared.

An MPRTTP receiver can only use one of its interface when communicating with a legacy sender.

Appendix B. Change Log

Note to the RFC-Editor: please remove this section prior to publication as an RFC.

B.1. Changes in draft-singh-avtcore-mprtp-09

- o Editorial updates based on review comments.
- o Clarified use of a=rtcp-rsize.
- o Fixed bug in block length of interface advertisements.

B.2. Changes in draft-singh-avtcore-mprtp-08

- o Added reference to use of PCP for discovering new interfaces.

B.3. Changes in draft-singh-avtcore-mprtp-06 and -07

- o Added reference to Mobility ICE.

B.4. Changes in draft-singh-avtcore-mprtp-05

- o SDP extensions moved to draft-singh-mmusic-mprtp-sdp-extension-00. Kept only the basic 'a=mprtp' attribute in this document.
- o Cleaned up ICE procedures for advertising only using in-band signaling.

B.5. Changes in draft-singh-avtcore-mprtp-04

- o Fixed missing 0xBEDE header in MPRTCP header format.
- o Removed connectivity checks and keep-alives from in-band signaling.
- o MPRTCP and MPRTCP are multiplexed on a single port.
- o MPRTCP packet headers optimized.
- o Made ICE optional
- o Updated Sections: 7.1.2, 8.1.x, 11.2, 11.4, 11.6.
- o Added how to use MPRTCP in RTSP (Section 12).
- o Updated IANA Considerations section.

B.6. Changes in draft-singh-avtcore-mprtp-03

- o Added this change log.
- o Updated section 6, 7 and 8 based on comments from MMUSIC.
- o Updated section 11 (SDP) based on comments of MMUSIC.
- o Updated SDP examples with ICE and non-ICE in out-of-band signaling scenario.
- o Added Appendix A on interop with legacy.
- o Updated IANA Considerations section.

B.7. Changes in draft-singh-avtcore-mprtp-02

- o MPRTCP protocol extensions use only one PT=210, instead of 210 and 211.
- o RTP header uses 1-byte extension instead of 2-byte.
- o Added section on RTCP Interval Calculations.
- o Added "mprtp-interface" attribute in SDP considerations.

B.8. Changes in draft-singh-avtcore-mprtp-01

- o Added MPRTTP and MPRTCP protocol extensions and examples.
- o WG changed from -avt to -avtcore.

Editorial Comments

[note-iceornot] Editor: Legacy applications do not require ICE for session establishment, therefore, MPRTTP should not require it as well.

[note-rtcp-reuse] Editor: inside the context of subflow specific reports can we reuse the payload type code for Sender Report (PT=200), Receiver Report (PT=201), Extension Report (PT=207). Transport and Payload specific RTCP messages are session specific and SHOULD be used as before.

[note-rtp-rtcp-mux] Editor: If a=mprtp is indicated, does the endpoint need to indicate a=rtcp-mux and a=rtcp-rsize? because MPRTTP mandates the use of RTP and RTCP multiplexing, and Reduced-Size RTCP.

Authors' Addresses

Varun Singh
Aalto University
School of Electrical Engineering
Otakaari 5 A
Espoo, FIN 02150
Finland

Email: varun@comnet.tkk.fi
URI: <http://www.netlab.tkk.fi/~varun/>

Teemu Karkkainen
Aalto University
School of Electrical Engineering
Otakaari 5 A
Espoo, FIN 02150
Finland

Email: teemuk@comnet.tkk.fi

Joerg Ott
Aalto University
School of Electrical Engineering
Otakaari 5 A
Espoo, FIN 02150
Finland

Email: jo@comnet.tkk.fi

Saba Ahsan
Aalto University
School of Electrical Engineering
Otakaari 5 A
Espoo, FIN 02150
Finland

Email: saba.ahsan@aalto.fi

Lars Eggert
NetApp
Sonnenallee 1
Kirchheim 85551
Germany

Phone: +49 151 12055791
Email: lars@netapp.com
URI: <http://eggert.org/>