Network Working Group
Internet-Draft
Updates: 5104 (if approved)
Intended status: Standards Track
Expires: January 5, 2015

B. Burman
A. Akram
Ericsson
R. Even
Huawei Technologies
M. Westerlund
Ericsson

July 4, 2014

RTP Stream Pause and Resume
draft-ietf-avtext-rtp-stream-pause-01

Abstract

With the increased popularity of real-time multimedia applications, it is desirable to provide good control of resource usage, and users also demand more control over communication sessions. This document describes how a receiver in a multimedia conversation can pause and resume incoming data from a sender by sending real-time feedback messages when using Real-time Transport Protocol (RTP) for real time data transport. This document extends the Codec Control Messages (CCM) RTCP feedback package by explicitly allowing and describing specific use of existing CCM messages and adding a group of new real-time feedback messages used to pause and resume RTP data streams. This document updates RFC 5104.

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.

Table of Contents

1. Introduction

As real-time communication attracts more people, more applications

are created; multimedia conversation applications being one example. Multimedia conversation further exists in many forms, for example, peer-to-peer chat application and multiparty video conferencing controlled by central media nodes, such as RTP Mixers.

Multimedia conferencing may involve many participants; each has its own preferences for the communication session, not only at the start but also during the session. This document describes several scenarios in multimedia communication where a conferencing node or participant chooses to temporarily pause an incoming RTP [RFC3550] stream and later resume it when needed. The receiver does not need to terminate or inactivate the RTP session and start all over again

by negotiating the session parameters, for example using SIP [RFC3261] with SDP Offer/Answer [RFC3264].

Centralized nodes, like RTP Mixers or MCUs, which either uses logic based on voice activity, other measurements, or user input could reduce the resources consumed in both the sender and the network by temporarily pausing the RTP streams that aren't required by the RTP Mixer. If the number of conference participants are greater than what the conference logic has chosen to present simultaneously to receiving participants, some participant RTP streams sent to the RTP Mixer may not need to be forwarded to any other participant. Those RTP streams could then be temporarily paused. This becomes especially useful when the media sources are provided in multiple encoding versions (Simulcast) [I-D.westerlund-avtcore-rtp-simulcast] or with Multi-Session Transmission (MST) of scalable encoding such as SVC [RFC6190]. There may be some of the defined encodings or combination of scalable layers that are not used all of the time.

As the RTP streams required at any given point in time is highly dynamic in such scenarios, using the out-of-band signaling channel for pausing, and even more importantly resuming, an RTP stream is difficult due to the performance requirements. Instead, the pause and resume signaling should be in the media plane and go directly between the affected nodes. When using RTP [RFC3550] for media transport, using Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF) [RFC4585] appears appropriate. No currently existing RTCP feedback message explicitly supports pausing and resuming an incoming RTP stream. As this affects the generation of packets and may even allow the encoding

process to be paused, the functionality appears to match Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF) [RFC5104] and it is proposed to define the solution as a Codec Control Message (CCM) extension.

The Temporary Maximum Media Bitrate Request (TMMBR) message of CCM is used by video conferencing systems for flow control. It is desirable to be able to use that method with a bitrate value of zero for pause and resume, whenever possible.

2. Definitions

2.1. Abbreviations

3GPP: 3rd Generation Partnership Project

AVPF: Audio-Visual Profile with Feedback (RFC 4585)

BGW: Border Gateway

CCM: Codec Control Messages (RFC 5104)

CNAME: Canonical Name (RTCP SDES)

CSRC: Contributing Source (RTP)

FB: Feedback (AVPF)

FCI: Feedback Control Information (AVPF)

FIR: Full Intra Refresh (CCM)

FMT: Feedback Message Type (AVPF)

LTE: Long-Term Evolution (3GPP)

MCU: Multipoint Control Unit

MTU: Maximum Transfer Unit

PT: Payload Type (RTP)

RTP: Real-time Transport Protocol (RFC 3550)

RTCP: RTP Control Protocol (RFC 3550)

RTCP RR: RTCP Receiver Report

SDP: Session Description Protocol (RFC 4566)

SGW: Signaling Gateway

SIP: Session Initiation Protocol (RFC 3261)

SSRC: Synchronization Source (RTP)

SVC: Scalable Video Coding

TCP: Transmission Control Protocol (RFC 793)

TMMBR: Temporary Maximum Media Bitrate Request (CCM)

TMMBN: Temporary Maximum Media Bitrate Notification (CCM)

UA: User Agent (SIP)

UDP: User Datagram Protocol (RFC 768)

2.2. Terminology

In addition to following, the definitions from RTP [RFC3550], AVPF [RFC4585], CCM [RFC5104], and RTP Taxonomy [I-D.ietf-avtext-rtp-grouping-taxonomy] also apply in this document.

Feedback Messages: CCM [RFC5104] categorized different RTCP feedback messages into four types, Request, Command, Indication and Notification. This document places the PAUSE and RESUME messages into Request category, PAUSED as Indication and REFUSE as Notification.

PAUSE Request from an RTP stream receiver to pause a stream

RESUME Request from an RTP stream receiver to resume a paused

stream

PAUSED Indication from an RTP stream sender that a stream is paused

REFUSE Notification from an RTP stream sender that a PAUSE or RESUME request will not be honored

Mixer: The intermediate RTP node which receives an RTP stream from different end points, combines them to make one RTP stream and forwards to destinations, in the sense described in Topo-Mixer of RTP Topologies [I-D.ietf-avtcore-rtp-topologies-update].

Participant: A member which is part of an RTP session, acting as receiver, sender or both.

Paused sender: An RTP stream sender that has stopped its transmission, i.e. no other participant receives its RTP transmission, either based on having received a PAUSE request, defined in this specification, or based on a local decision.

Pausing receiver: An RTP stream receiver which sends a PAUSE request, defined in this specification, to other participant(s).

Stream: Used as a short term for RTP stream, unless otherwise noted.

Stream receiver: Short for RTP stream receiver; the RTP entity responsible for receiving an RTP stream, usually a Media Depacketizer.

Stream sender: Short for RTP stream sender; the RTP entity responsible for creating an RTP stream, usually a Media Packetizer.

2.3. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. Use Cases

This section discusses the main use cases for RTP stream pause and resume.

3.1. Point to Point

This is the most basic use case with an RTP session containing two End Points. Each End Point sends one or more streams.

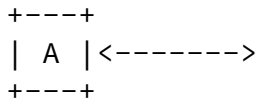


Figure 1: Point to Point

The usage of RTP stream pause in this use case is to temporarily halt delivery of streams that the sender provides but the receiver does not currently use. This can for example be due to minimized applications where the video stream is not actually shown on any display, and neither is it used in any other way, such as being recorded.

In this case, since there is only a single receiver of the stream, pausing or resuming a stream does not impact anyone else than the sender and the single receiver of that stream.

RTCWEB WG's use case and requirements document [I-D.ietf-rtcweb-use-cases-and-requirements] defines the following API requirements in Appendix A, used also by W3C WebRTC WG:

A8 The Web API must provide means for the web application to mute/ unmute a stream or stream component(s). When a stream is sent to a peer mute status must be preserved in the stream received by the peer.

A9 The Web API must provide means for the web application to cease the sending of a stream to a peer.

This memo provides means to optimize transport usage by stop sending muted streams and start sending again when unmuting.

3.2. RTP Mixer to Media Sender

One of the most commonly used topologies in centralized conferencing is based on the RTP Mixer [I-D.ietf-avtcore-rtp-topologies-update]. The main reason for this is that it provides a very consistent view of the RTP session towards each participant. That is accomplished through the Mixer originating its' own streams, identified by SSRC, and any RTP streams sent to the participants will be sent using those SSRCs. If the Mixer wants to identify the underlying media sources for its' conceptual streams, it can identify them using CSRC. The stream the Mixer provides can be an actual mix of multiple media sources, but it might also be switching received streams as described in Sections 3.6-3.8 of [I-D.ietf-avtcore-rtp-topologies-update].

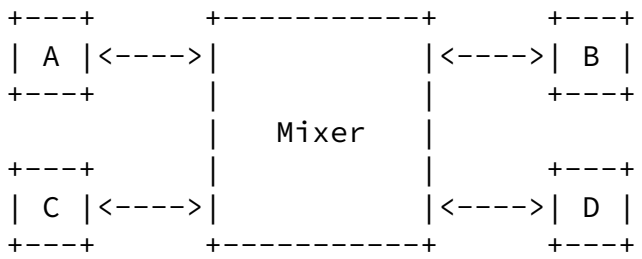


Figure 2: RTP Mixer in Unicast-only

Which streams that are delivered to a given receiver, A, can depend on several things. It can either be the RTP Mixer's own logic and measurements such as voice activity on the incoming audio streams. It can be that the number of sent media sources exceed what is reasonable to present simultaneously at any given receiver. It can also be a human controlling the conference that determines how the media should be mixed; this would be more common in lecture or similar applications where regular listeners may be prevented from breaking into the session unless approved by the moderator. The streams may also be part of a Simulcast [I-D.westerlund-avtcore-rtp-simulcast] or scalable encoded (for Multi-Stream Transmission) [RFC6190], thus providing multiple versions that can be delivered by the RTP stream sender. These examples indicate that there are numerous reasons why a particular stream would not currently be in use, but must be available for use at very short notice if any dynamic event occurs that causes a different stream selection to be done in the Mixer.

Because of this, it would be highly beneficial if the Mixer could request to pause a particular stream from being delivered to it. It also needs to be able to resume delivery with minimal delay.

Just as for point-to-point (Section 3.1), there is only a single receiver of the stream, the RTP Mixer, and pausing or resuming a

stream does not affect anyone else than the sender and single receiver of that stream.

3.3. RTP Mixer to Media Sender in Point-to-Multipoint

This use case is similar to the previous section, however the RTP Mixer is involved in three domains that need to be separated; the Multicast Network (including participants A and C), participant B, and participant D. The difference from above is that A and C share a multicast domain, which is depicted below.

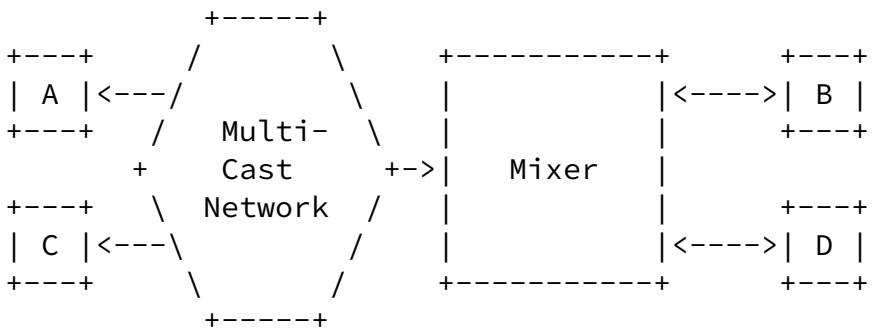


Figure 3: RTP Mixer in Point-to-Multipoint

If the RTP Mixer pauses a stream from A, it will not only pause the stream towards itself, but will also stop the stream from arriving to C, which C is heavily impacted by, might not approve of, and should thus have a say on.

If the Mixer resumes a paused stream from A, it will be resumed also towards C. In this case, if C is not interested it can simply ignore the stream and is not impacted as much as above.

In this use case there are several receivers of a stream and special care must be taken as not to pause a stream that is still wanted by some receivers.

3.4. Media Receiver to RTP Mixer

An End Point in Figure 2 could potentially request to pause the delivery of a given stream. Possible reasons include the ones in the point to point case (Section 3.1) above.

When the RTP Mixer is only connected to individual unicast paths, the use case and any considerations are identical to the point to point use case.

However, when the End Point requesting stream pause is connected to the RTP Mixer through a multicast network, such as A or C in

Figure 3, the use case instead becomes identical to the one in Section 3.3, only with reverse direction of the streams and pause/ resume requests.

3.5. Media Receiver to Media Sender Across RTP Mixer

An End Point, like A in Figure 2, could potentially request to pause the delivery of a given stream, like one of B's, over any of the SSRCs used by the Mixer by sending a pause request for the CSRC identifying the stream. However, the authors are of the opinion that this is not a suitable solution, for several reasons:

1. The Mixer might not include CSRC in it's stream indications.

2. An End Point cannot rely on the CSRC to correctly identify the stream to be paused when the delivered media is some type of mix. A more elaborate stream identification solution is needed to support this in the general case.

3. The End Point cannot determine if a given stream is still needed by the RTP Mixer to deliver to another session participant.

Due to the above reasons, we exclude this use case from further consideration.

4. Design Considerations

This section describes the requirements that this specification needs to meet.

4.1. Real-time Nature

The first section (Section 1) of this specification describes some possible reasons why a receiver may pause an RTP sender. Pausing and resuming is time-dependent, i.e. a receiver may choose to pause an RTP stream for a certain duration, after which the receiver may want the sender to resume. This time dependency means that the messages related to pause and resume must be transmitted to the sender in real-time in order for them to be purposeful. The pause operation is

arguably not very time critical since it mainly provides a reduction of resource usage. Timely handling of the resume operation is however likely to directly impact the end-user's perceived quality experience, since it affects the availability of media that the user expects to receive more or less instantly.

4.2. Message Direction

It is the responsibility of an RTP stream receiver, who wants to pause or resume a stream from the sender(s), to transmit PAUSE and RESUME messages. An RTP stream sender who likes to pause itself, can simply do it. Any indication that an RTP stream is paused is the responsibility of the RTP stream sender and may in some cases not even be needed by the stream receiver.

4.3. Apply to Individual Sources

The PAUSE and RESUME messages apply to single RTP streams identified by their SSRC, which means the receiver targets the sender's SSRC in the PAUSE and RESUME requests. If a paused sender starts sending with a new SSRC, the receivers will need to send a new PAUSE request in order to pause it. PAUSED indications refer to a single one of the sender's own, paused SSRC.

4.4. Consensus

An RTP stream sender should not pause an SSRC that some receiver still wishes to receive. The reason is that in RTP topologies where the stream is shared between multiple receivers, a single receiver on that shared network, independent of it being multicast, a mesh with joint RTP session or a transport Translator based, must not single-handedly cause the stream to be paused without letting all other receivers to voice their opinions on whether or not the stream should be paused. A consequence of this is that a newly joining receiver, for example indicated by an RTCP Receiver Report containing both a new SSRC and a CNAME that does not already occur in the session, firstly needs to learn the existence of paused streams, and secondly should be able to resume any paused stream. Any single receiver

wanting to resume a stream should also cause it to be resumed.

4.5. Acknowledgments

RTP and RTCP does not guarantee reliable data transmission. It uses whatever assurance the lower layer transport protocol can provide. However, this is commonly UDP that provides no reliability guarantees. Thus it is possible that a PAUSE and/or RESUME message transmitted from an RTP End Point does not reach its destination, i.e. the targeted RTP stream sender. When PAUSE or RESUME reaches the RTP stream sender and are effective, i.e., an active RTP stream sender pauses, or a resuming RTP stream sender have media data to transmit, it is immediately seen from the arrival or non-arrival of RTP packets for that RTP stream. Thus, no explicit acknowledgments are required in this case.

In some cases when a PAUSE or RESUME message reaches the RTP stream sender, it will not be able to pause or resume the stream due to some local consideration, for example lack of data to transmit. This error condition, a negative acknowledgment, may be needed to avoid unnecessary retransmission of requests (Section 4.6).

4.6. Retransmitting Requests

When the stream is not affected as expected by a PAUSE or RESUME request, the request may have been lost and the sender of the request will need to retransmit it. The retransmission should take the round trip time into account, and will also need to take the normal RTCP bandwidth and timing rules applicable to the RTP session into account, when scheduling retransmission of feedback.

When it comes to resume requests that are more time critical, the best resume performance may be achieved by repeating the request as often as possible until a sufficient number have been sent to reach a high probability of request delivery, or the stream gets delivered.

4.7. Sequence Numbering

A PAUSE request message will need to have a sequence number to separate retransmissions from new requests. A retransmission keeps the sequence number unchanged, while it is incremented every time a

new PAUSE request is transmitted that is not a retransmission of a previous request.

Since RESUME always takes precedence over PAUSE and are even allowed to avoid pausing a stream, there is a need to keep strict ordering of PAUSE and RESUME. Thus, RESUME needs to share sequence number space with PAUSE and implicitly references which PAUSE it refers to. For the same reasons, the explicit PAUSED indication also needs to share sequence number space with PAUSE and RESUME.

5. Relation to Other Solutions

This section compares other possible solutions to achieve a similar functionality, along with motivations why the current solution is chosen.

5.1. Signaling Technology Performance Comparison

Editor's note: This section is related to the motivation for selecting RTCP as signaling technology rather than SIP/SDP and should be considered to be removed or at least significantly reduced if and when this draft is adopted as a working group

draft, since there now seems to be consensus that RTCP is the preferred technology.

This section contains what is thought to be a realistic estimate of one-way data transmission times for signaling implementing functionalities of this specification.

Two signaling protocols are compared. SIP is chosen to represent signaling in the control plane and RTCP is chosen to represent signaling in the media plane. For the sake of the comparison, each of these two protocols are listed with one favorable and one unfavorable condition to give the reader a hint of what range of delays that can be expected. The favorable condition is chosen as good as possible, while still realistic. The unfavorable condition is also chosen to be realistically occurring, and is not the worst possible or imaginable. Actual delays can in most cases be expected to lie somewhere between those two values.

It would also be possible to include a signaling protocol using a some dedicated signaling channel, separate from SIP and RTCP, into the comparison. Such signaling protocol can be expected to show performance somewhere in the range covered by the SIP and RTCP comparison below. The protocol can either use UDP as transport, like RTCP, or it can use TCP, like SIP, when the messages becomes too large for the MTU. The data sent on such channel can either be text based, in which case the amount of data can be similar to SIP, or it can be binary, in which case the amount of data can be similar to RTCP. Therefore, the dedicated signaling channel case is not described further in this specification.

Two different access technologies are compared:

- Wired, fixed access is chosen as a representative low-delay alternative.

- Mobile wireless access according to 3GPP LTE [TS36.201], also known as "4G", is chosen as a representative high-delay alternative.

NOTE: LTE is at the time of writing the most recent and best performing mobile wireless access. If an earlier mobile wireless access was to be used instead, the estimated transmission times would be considerably increased. For example, it is estimated that using 3GPP HSPA [TS25.308] (evolved 3G, just previous to LTE) would increase RTCP signaling times somewhat and significantly increase signaling times for SIP, although those estimates are too preliminary to provide any values here.

The target scenario includes two UA, residing in two different provider's (operator's) network. Those networks are assumed to be geographically close, that is no inter-continental transmission delays are included in the estimates.

Three signaling alternatives are compared:

- Wireless UA to wireless UA, including two wireless links, uplink and downlink.

- Wireless UA to media server (MCU), including a single wireless

uplink.

o Media server (MCU) to wireless UA, including a single wireless downlink.

The reason to include separate results for wireless uplink and downlink is that delay times can differ significantly.

The targeted topology is outlined in the following figure.

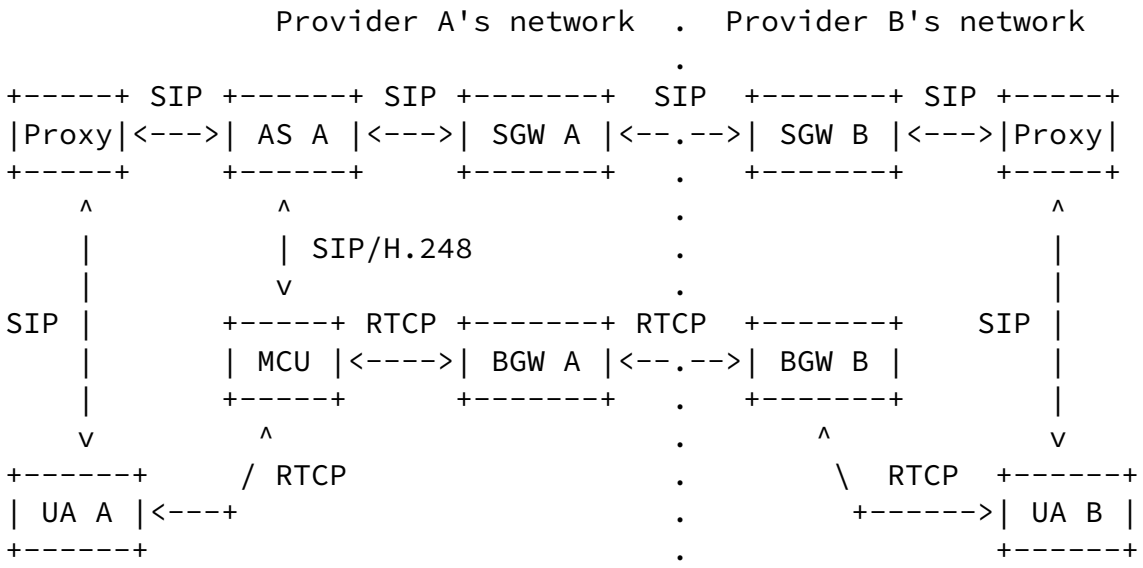


Figure 4: Comparison Signaling Topology

In the figure above, UA is a SIP User Agent, Proxy is a SIP Proxy, AS is an Application Server, MCU is a Multipoint Conference Unit, SGW a Signaling GateWay, and BGW a media Border GateWay.

It can be noted that when either one or both UAs use call forwarding or have roamed into yet another provider's network, several more signaling path nodes and a few more media path nodes could be included in the end-to-end signaling path.

The MCU is assumed to be located in one of the provider's network. Signaling delays between the MCU and a UA are presented as the average of MCU and UA being located in the same and different provider's networks.

These assumptions are used for SIP signaling:

- A SIP UPDATE is used within an established session to dynamically impact individual streams to achieve the pause and resume functionality. The offer and answer SDP contains one audio and one video media, compliant with what is suggested in 3GPP MTSI [TS26.114], with the addition of SDP feedback message indication outlined in this specification (Section 10). A more complex media session with more streams would significantly add to the SDP size.

- UDP is used as transport, except when risking to exceed MTU, in which case TCP is used instead. This is evaluated on a per-message basis.

- Only SIP forward direction is included in the delay estimate, that is, delays needed to receive a response such as 200 OK are not included.

- Favorable case:

  * SIP SigComp [RFC5049] in dynamic mode is used for SIP and SDP signaling on the mobile link, reducing the SIP message size to approximately 1/3 of the original size.

- Unfavorable case:

  * SIP message is not compressed on the mobile link.

  * SIP signaling on the mobile link uses a dedicated mobile wireless access radio channel that was idle for some time, has entered low power state and thus has to be re-established by radio layer signaling before any data can be sent.

These assumptions are used for RTCP signaling:

- A minimal compound RTCP feedback packet is used, including one SR and one SDES with only the CNAME item present, with the addition of the feedback message outlined in Section 8.

- RTCP bandwidth is chosen based on a 200 kbit/s session, which is considered to be a low bandwidth for media that would be worth pausing, and using the default 5% of this for RTCP traffic results

in 10 kbit/s. This low bandwidth makes RTCP scheduling delays be a significant factor in the unfavorable case.

- Since there are random delay factors in RTCP transmission, the expected, most probable value is used in the estimates.

- The mobile wireless access channel used for RTCP will always be active, that is there will be sufficient data to send at any time such that the radio channel will never have to be re-established. This is considered reasonable since it is assumed that the same channel is not only used for the messages defined in this specification, but also for other RTP and RTCP data.

- Favorable case:

  * It is assumed that AVPF Early or Immediate mode can always be used for the signaling described in this specification, since such signaling will be small in size and only occur occasionally in RTCP time scale.

  * Early mode does not use dithering of send times (T_dither_max is set to 0), that is, sender and receiver of the message are connected point-to-point. It can be noted that in case of a multiparty session where multiple End Points can see each others' messages, and unless the number of End Points is very large, it is very unlikely that more than a single End Point has the desire to send the same message (defined in this specification) as another End Point, and at almost exactly the same time. It is therefore arguably not very meaningful for messages in this specification to try to do feedback suppression by using a non-zero T_dither_max, even in multiparty sessions, but AVPF does not allow for any exemption from that rule.

  * Reduced-size RTCP is used, which is considered appropriate for the type of messages defined in this specification.

  * RTP/RTCP header compression [RFC5225] is not used, not even on the mobile link.

- Unfavorable case:

  * The expected, regular AVPF RTCP interval is used, including an expected value for timer re-consideration.

  * A full, not reduced-size, minimal compound RTCP feedback packet without header compression is always used. No reduction of

scheduling delays from the use of reduced-size RTCP is included

in the evaluation, since that would also require a reasonable estimate of the mix of compound and non-compound RTCP, which was considered too difficult for this study. The given unfavorable delays are thus an over-estimate compared to a more realistic case.

Common to both SIP and RTCP signaling estimates is that no UA processing delays are included. The reason for that decision is that processing delays are highly implementation and UA dependent. It is expected that wireless UA will be more limited than fixed UA by processing, but they are also constantly and quickly improving so any estimate will very quickly be outdated. More realistic estimates will however have to add such delays, which can be expected to be in the order of a few to a few tens of milliseconds. It is expected that SIP will be more penalized than RTCP by including processing delays, since it has larger and more complex messages. The processing may also include SigComp [RFC5049] compression and decompression in the favorable cases.

As a partial result, the message sizes can be compared, based on the messages defined in this specification (Section 8) and a SIP UPDATE with contents (Section 10) as discussed above. Favorable and unfavorable message sizes are presented as stacked bars in the figure below. Message sizes include IPv4 headers but no lower layer data, are rounded to the nearest 25 bytes, and the bars are to scale.

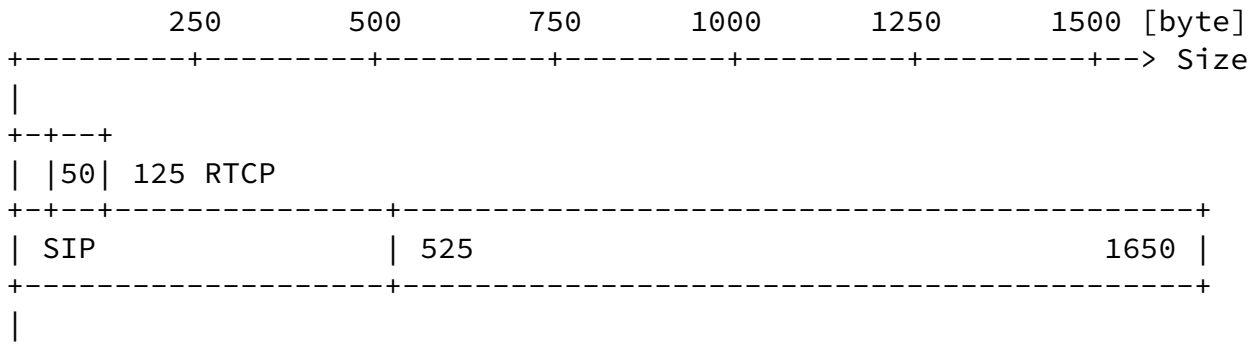


Figure 5: Message Size Comparison

The signaling delay results of the study are summarized in the

following two figures. Favorable and unfavorable values are presented as stacked bars. Since there are many factors that impact the calculations, including some random processes, there are uncertainty in the calculations and delay values are thus rounded to nearest 5 ms. The bars are to scale.

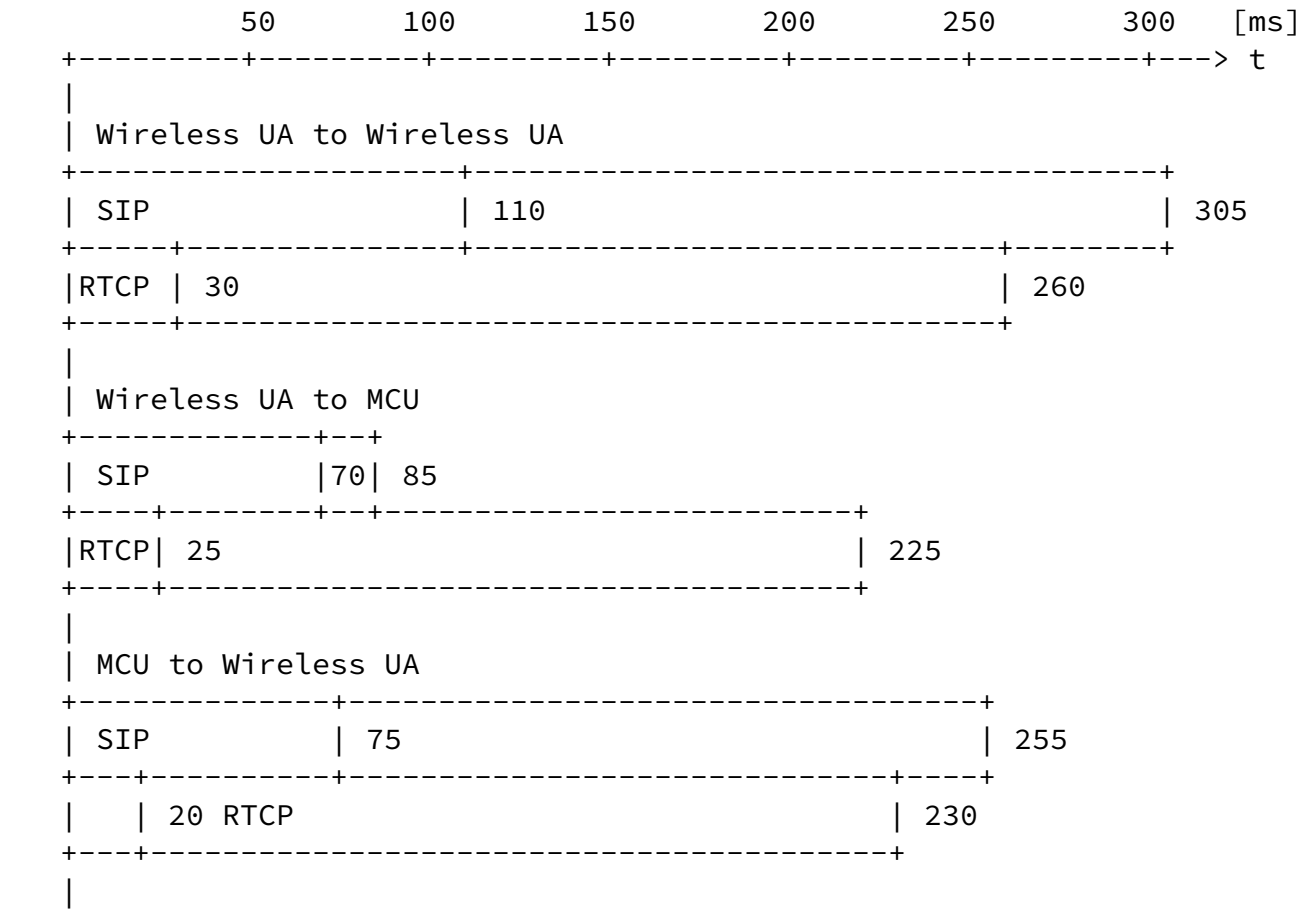


Figure 6: Mobile Access Transmission Delay Comparison

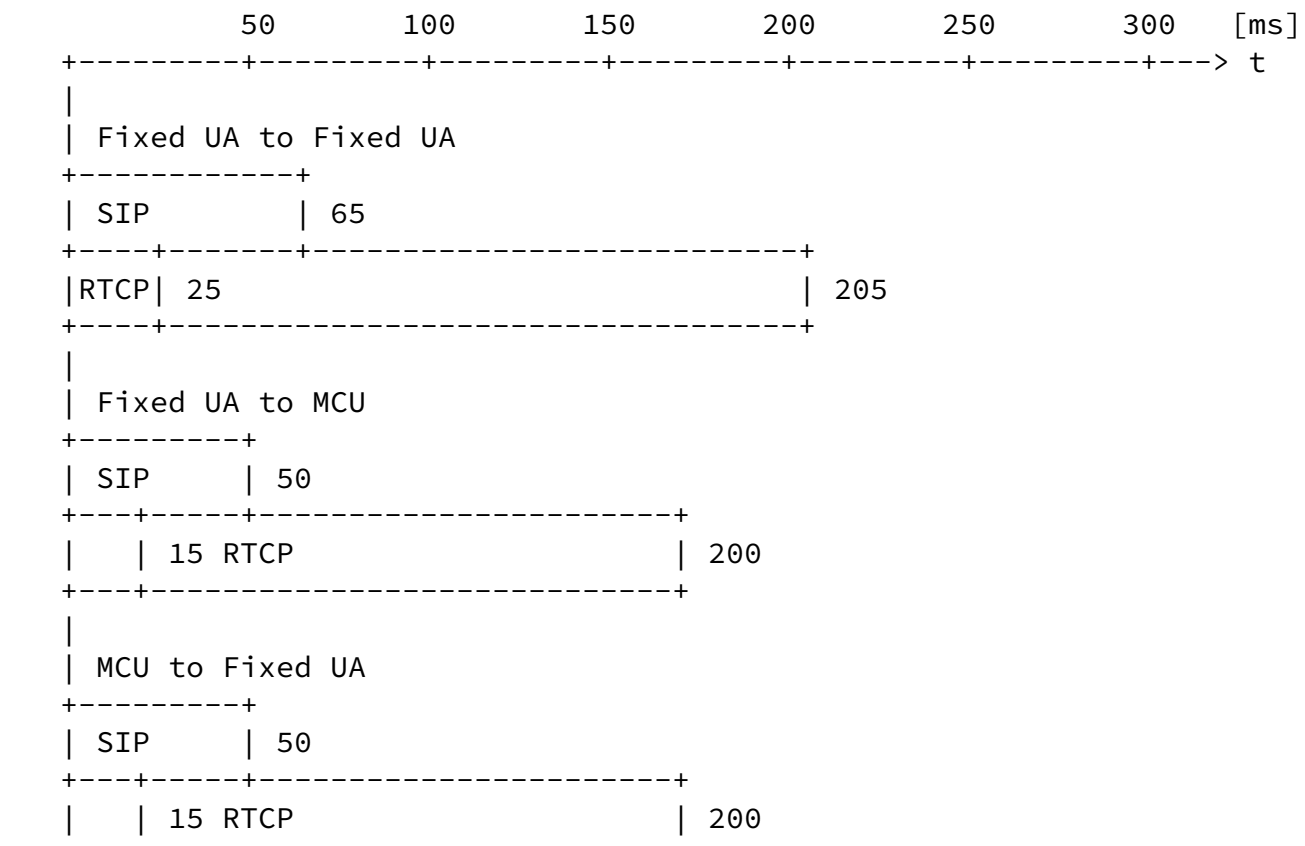
As can be seen, RTCP has a smaller signaling delay than SIP in a majority of cases for this mobile access. Non-favorable RTCP is however always worse than favorable SIP.

The UA to MCU signaling corresponds to the use case in Section 3.4.

The reason that unfavorable SIP is more beneficial than unfavorable RTCP in this case comes from the fact that latency is fairly short to re-establish an uplink radio channel (as was assumed needed for unfavorable SIP), while unfavorable RTCP does not benefit from this since the delay is mainly due to RTCP Scheduling.

The MCU to UA signaling corresponds to the use case in Section 3.2. It has an unfavorable SIP signaling case with much longer delay than UA to MCU above, because the mixer cannot re-establish a downlink radio channel as quickly as the UA can establish an uplink. This case is applicable when an MCU wants to resume a paused stream, which is likely the most delay sensitive functionality, as discussed in Section 4.1.

Below are the same cases for fixed access depicted. Although delays are generally shorter, scales are kept the same for easy comparison with the previous figure.



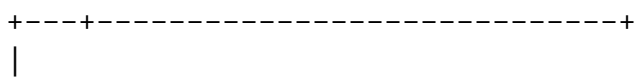


Figure 7: Fixed Access Transmission Delay Comparison

For fixed access, favorable RTCP is still significantly better than SIP, but unfavorable RTCP is significantly worse than SIP. There is no difference between favorable and unfavorable SIP, since in fixed access there is no channel that needs to be re-established.

Regarding the unfavorable values above, it should be possible with reasonable effort to design UA and network nodes that show favorable delays in a majority of cases.

For SIP, the major delays in the unfavorable cases above comes from re-establishing a radio bearer that has entered low power state due to inactivity, and large size SIP messages. The inactivity problem can be removed by using for example SIP keep-alive [RFC5626], at the cost of reduced battery life to keep the signaling radio bearer active, and some very minimal amount of extra data transmission. The large SIP messages can to some extent be reduced by SIP SigComp

[RFC5049]. It may however prove harder to reduce delays that comes from forwarding the SDP many times between different signaling nodes.

For RTCP, the major delays comes from low RTCP bandwidth and not being able to use Immediate or Early mode, including use of timer re-consideration. UAs and network nodes can explicitly allocate an appropriate amount of RTCP bandwidth through use of the b=RS and b=RR RTCP bandwidth SDP attributes [RFC3556]. For RTP streams of higher bandwidth than the 200 kbit/s used in this comparison, which will be even more interesting to pause, RTCP bandwidth will per default also be higher, significantly reducing the signaling delays. For example, using a 1000 kbit/s stream instead of a 200 kbit/s stream will reduce the unfavorable RTCP delays from 260 ms to 115 ms for Wireless-Wireless, from 225 ms to 80 ms for Wireless-MCU, and from 230 ms to 80 ms for MCU-Wireless.

5.2. CCM TMMBR / TMMBN

The Codec Control Messages specification [RFC5104] contains two

messages, Temporary Maximum Media Bitrate Request (TMMBR) and Temporary Maximum Media Bitrate Notification (TMMBN), which could provide some of the necessary functionality. TMMBR with a bitrate value of 0 could effectively constitute a PAUSE request and TMMBN 0 could effectively be a PAUSED indication, and there are already implementations making use of TMMBR 0 in this way. It is possible to signal per SSRC (Section 4.3) and using the media path for signaling (AVPF) [RFC4585] will in most cases provide the shortest achievable signaling delay (Section 4.1). However, in some cases the defined semantics for TMMBR differ from what is required for PAUSE.

When there is only a single receiver of a stream, TMMBR 0 and PAUSE are effectively identical.

When there are several receivers of the same stream, the stream must not be paused until there are no receiver that desires to receive it (Section 4.4), for example there is no disapproving RESUME for a PAUSE. In the presence of several simultaneous receivers, the TMMBR semantics is the opposite; the first RTP stream receiver that sends TMMBR 0 will pause the stream for all receivers.

When there is only a single receiver of a stream that is paused, TMMBR with a bitrate greater than 0 can effectively function as a RESUME, resuming the stream immediately as needed (Section 4.4).

For the case of multiple simultaneous receivers, TMMBR specifies to use a guard period when increasing the bandwidth. In this case, TMMBR/TMMBN semantics (Section 4.2.1.2 of [RFC5104]) requires an RTP stream sender to wait 2*RTT+T_dither_max after having sent a TMMBN,

indicating the intention to increase the bandwidth, before it actually increases its bandwidth usage. The RTT is specified to be the longest the RTP stream sender knows in the RTP session. So, there is both the delay between the RTP stream sender receiving the TMMBR until it can send a TMMBN, and the above delay for the guard period before the RTP stream sender is allowed to resume transmission. This delay before resuming transmission is the most time critical operation in this solution, making use of TMMBR as RESUME according to the defined semantics infeasible in practice when there are multiple simultaneous stream receivers.

5.3. SDP "inactive" Attribute

In SDP [RFC4566], an "inactive" attribute is defined on media level and session level. The attribute is intended to be used to put media "on hold", either at the beginning of a session or as a result of session re-negotiation [RFC3264], for example using SIP re-INVITE [RFC3261], possibly in combination with ITU-T H.248 media gateway control.

This attribute is only possible to specify with media level resolution, is not possible to signal per individual stream (SSRC) (Section 4.3), and is thus not usable for RTP sessions containing more than a single SSRC.

There is a per-ssrc attribute defined in [RFC5576], but that does currently not allow to set an individual stream (SSRC) inactive.

Using "inactive" does thus not provide sufficient functionality for the purpose of this specification.

5.4. Media Source Selection in SDP

There is a draft that selects sources based on SDP [I-D.lennox-mmusic-sdp-source-selection] information. It builds on the per-ssrc attribute [RFC5576] discussed above (Section 5.3).

The semantics differ between selecting a media source and pause / resume for a stream in topologies other than point-to-point. For example, in RTP Receiver to Mixer (Section 3.4), pausing a stream (SSRC) from the mixer should stop it being received altogether, while excluding a stream (CSRC) from the mix would just avoid that specific media source being included in the stream from the mixer. There is a similar difference between resuming a stream (SSRC) from the mixer and allowing a media source (CSRC) to be included in the mix again. This suffers from a lack of functionality for consensus (Section 4.4) and would likely also suffer from lower real-time performance (Section 4.1).

5.5. Conclusion

As can be seen from Section 5.1, using SIP and SDP to carry pause and resume information means that it will need to traverse the entire signaling path to reach the signaling destination (either the remote

End Point or the entity controlling the RTP Mixer), across any signaling proxies that potentially also has to process the SDP content to determine if they are expected to act on it. The amount of bandwidth required for a SIP/SDP-based signaling solution is in the order of at least 10 times more than an RTCP-based solution.

Especially for UA sitting on mobile wireless access, this will risk introducing delays that are too long (Section 4.1) to provide a good user experience, and the bandwidth cost may also be considered infeasible compared to an RTCP-based solution.

As seen in the same section, the RTCP data is sent through the media path, which is likely shorter (contains fewer intermediate nodes) than the signaling path but may anyway have to traverse a few intermediate nodes. The amount of processing and buffering required in intermediate nodes to forward those RTCP messages is however believed to be significantly less than for intermediate nodes in the signaling path.

Based on those reasons, RTCP is proposed as signaling protocol for the pause and resume functionality. Much of the wanted functionality can in a point-to-point case be achieved with the existing TMMBR/ TMMBN CCM messages [RFC5104], but they cannot be used when the stream is sent to multiple simultaneous receivers.

6. Solution Overview

The proposed solution implements PAUSE and RESUME functionality based on sending AVPF RTCP feedback messages from any RTP session participant that wants to pause or resume a stream targeted at the stream sender, as identified by the sender SSRC.

It is proposed to re-use CCM TMMBR and TMMBN [RFC5104] to the extent possible, and to define a small set of new RTCP feedback messages where new semantics is needed. Considerations that apply when using TMMBR/TMMBN for pause and resume purposes are also described.

A single Feedback message specification is used to implement the new messages. The message consists of a number of Feedback Control Information (FCI) blocks, where each block can be a PAUSE request, a RESUME request, PAUSED indication, a REFUSE response, or an extension to this specification. This structure allows a single feedback message to handle pause functionality on a number of streams.

The PAUSED functionality is also defined in such a way that it can be used standalone by the RTP stream sender to indicate a local decision to pause, and inform any receiver of the fact that halting media delivery is deliberate and which RTP packet was the last transmitted.

This section is intended to be explanatory and therefore intentionally contains no mandatory statements. Such statements can instead be found in other parts of this specification.

6.1. Expressing Capability

An End Point can use an extension to CCM SDP signaling to declare capability to understand the messages defined in this specification. Capability to understand PAUSED indication is defined separately from the others to support partial implementation, which is specifically believed to be feasible for the RTP Mixer to Media Sender use case (Section 3.2).

For the case when TMMBR/TMMBN are used for pause and resume purposes, it is possible to explicitly express joint support for TMMBR and TMMBN, but not for TMMBN only.

6.2. Requesting to Pause

An RTP stream receiver can choose to request PAUSE at any time, subject to AVPF timing rules. This also applies when using TMMBR 0 in the point-to-point case.

The PAUSE request contains a PauseID, which is incremented by one (in modulo arithmetic) with each PAUSE request that is not a re-transmission. The PauseID is scoped by and thus a property of the targeted RTP stream (SSRC).

When a non-paused RTP stream sender receives the PAUSE request, it continues to send the RTP stream while waiting for some time to allow other RTP stream receivers in the same RTP session that saw this PAUSE request to disapprove by sending a RESUME (Section 6.4) for the same stream and with the same PauseID as in the disapproved PAUSE. If such disapproving RESUME arrives at the RTP stream sender during the wait period before the stream is paused, the pause is not performed. In point-to-point configurations, the wait period may be set to zero. Using a wait period of zero is also appropriate when using TMMBR 0 and in line with the semantics for that message.

If the RTP stream sender receives further PAUSE requests with the available PauseID while waiting as described above, those additional requests are ignored.

If the PAUSE request or TMMBR 0 is lost before it reaches the RTP stream sender, it will be discovered by the RTP stream receiver because it continues to receive the RTP stream. It will also not see any PAUSED indication (Section 6.3) or TMMBN 0 for the stream. The same condition can be caused by the RTP stream sender having received a disapproving RESUME from a stream receiver A for a PAUSE request sent by a stream sender B, but that the PAUSE sender (B) did not receive the RESUME (from A) and may instead think that the PAUSE was lost. In both cases, a PAUSE request can be re-transmitted using the same PauseID. If using TMMBR 0 the request MAY be re-transmitted when the requester fails to receive a TMMBN 0 confirmation.

If the pending stream pause is aborted due to a disapproving RESUME, the PauseID from the disapproved PAUSE is invalidated by the RESUME and any new PAUSE must use an incremented PauseID (in modulo arithmetic) to be effective.

An RTP stream sender receiving a PAUSE not using the available PauseID informs the RTP stream receiver sending the ineffective PAUSE of this condition by sending a REFUSE response that contains the next available PauseID value. This REFUSE also informs the RTP stream receiver that it is probably not feasible to send another PAUSE for some time, not even with the available PauseID, since there are other RTP stream receivers that wish to receive the stream.

A similar situation where an ineffective PauseID is chosen can appear when a new RTP stream receiver joins a session and wants to PAUSE a stream, but does not yet know the available PauseID to use. The REFUSE response will then provide sufficient information to create a valid PAUSE. The required extra signaling round-trip is not considered harmful, since it is assumed that pausing a stream is not time-critical (Section 4.1).

There may be local considerations making it impossible or infeasible to pause the stream, and the RTP stream sender can then respond with a REFUSE. In this case, if the used PauseID would otherwise have been effective, the REFUSE contains the same PauseID as in the PAUSE request, and the PauseID is kept as available. Note that when using TMMBR 0 as PAUSE, that request cannot be refused (TMMBN > 0) due to the existing restriction in section 4.2.2.2 of [RFC5104] that TMMBN SHALL contain the current bounding set, and the fact that a TMMBR 0 will always be the most restrictive point in any bounding set.

If the RTP stream sender receives several identical PAUSE for an RTP stream that was already at least once responded with REFUSE and the condition causing REFUSE remains, those additional REFUSE should be sent with regular RTCP timing. A single REFUSE can respond to several identical PAUSE requests.

6.3. Media Sender Pausing

An RTP stream sender can choose to pause the stream at any time. This can either be as a result of receiving a PAUSE, or be based on some local sender consideration. When it does, it sends a PAUSED indication, containing the available PauseID. If the stream was paused by a TMMBR 0, TMMBN 0 is used as PAUSED indication. What is said on PAUSED in the rest of this paragraph apply also to the use of TMMBN 0, except for PAUSED message parameters. Note that PauseID is incremented when pausing locally (without having received a PAUSE). It also sends the PAUSED indication in the next two regular RTCP reports, given that the pause condition is then still effective.

The RTP stream sender may want to apply some local consideration to exactly when the stream is paused, for example completing some media unit or a forward error correction block, before pausing the stream.

The PAUSED indication also contains information about the RTP extended highest sequence number when the pause became effective. This provides RTP stream receivers with first hand information allowing them to know whether they lost any packets just before the stream paused or when the stream is resumed again. This allows RTP stream receivers to quickly and safely take into account that the stream is paused, in for example retransmission or congestion control algorithms.

If the RTP stream sender receives PAUSE requests with the available PauseID while the stream is already paused, those requests are ignored.

As long as the stream is being paused, the PAUSED indication MAY be sent together with any regular RTCP SR or RR. Including PAUSED in this way allows RTP stream receivers joining while the stream is paused to quickly know that there is a paused stream, what the last sent extended RTP sequence number was, and what the next available

PauseID is to be able to construct valid PAUSE and RESUME requests at a later stage.

When the RTP stream sender learns that a new End Point has joined the RTP session, for example by a new SSRC and a CNAME that was not previously seen in the RTP session, it should send PAUSED indications for all its paused streams at its earliest opportunity. It should in addition continue to include PAUSED indications in at least two regular RTCP reports.

6.4. Requesting to Resume

An RTP stream receiver can request to resume a stream with a RESUME request at any time, subject to AVPF timing rules. If the stream was paused with TMMBR 0, resuming the stream is made with TMMBR containing a bitrate value larger than 0. The bitrate value used when resuming after a PAUSE with TMMBR 0 is either according to known limitations, or the configured maximum for the stream or session. What is said on RESUME in the rest of this paragraph apply also to the use of TMMBR with a bitrate value larger than 0, except for RESUME message parameters.

The RTP stream receiver must include the available PauseID in the RESUME request for it to be effective.

A pausing RTP stream sender that receives a RESUME including the correct available PauseID resumes the stream at the earliest opportunity. Receiving RESUME requests for a stream that is not paused does not require any action and can be ignored.

There may be local considerations, for example that the media device is not ready, making it temporarily impossible to resume the stream at that point in time, and the RTP stream sender MAY then respond with a REFUSE containing the same PauseID as in the RESUME. When receiving such REFUSE with a PauseID identical to the one in the sent RESUME, RTP stream receivers SHOULD then avoid sending further RESUME requests for some reasonable amount of time, to allow the condition to clear.

If the RTP stream sender receives several identical RESUME for an RTP stream that was already at least once responded with REFUSE and the condition causing REFUSE remains, those additional REFUSE should be sent with regular RTCP timing. A single REFUSE can respond to several identical RESUME requests.

When resuming a paused stream, especially for media that makes use of temporal redundancy between samples such as video, the temporal dependency between samples taken before the pause and at the time instant the stream is resumed may not be appropriate to use in the encoding. Should such temporal dependency between before and after the media was paused be used by the RTP stream sender, it requires the RTP stream receiver to have saved the sample from before the pause for successful continued decoding when resuming. The use of this temporal dependency is left up to the RTP stream sender. If temporal dependency is not used when the RTP stream is resumed, the first encoded sample after the pause will not contain any temporal dependency to samples before the pause (for video it may be a so-called intra picture). If temporal dependency to before the pause is

used by the RTP stream sender when resuming, and if the RTP stream receiver did not save any sample from before the pause, the RTP stream receiver can use a FIR request [RFC5104] to explicitly ask for a sample without temporal dependency (for video a so-called intra picture), even at the same time as sending the RESUME.

6.5. TMMBR/TMMBN Considerations

As stated, TMMBR/TMMBN may be used to provide pause and resume functionality for the point-to-point case. If the topology is not point-to-point, TMMBR/TMMBN cannot safely be used for pause or resume.

This is a brief summary of what functionality is provided when using TMMBR/TMMBN:

TMMBR 0: Corresponds to PAUSE, without the requirement for any hold-off period to wait for RESUME before pausing the stream.

TMMBR >0: Corresponds to RESUME when the stream was previously paused with TMMBR 0. Since there is only a single RTP stream

receiver, there is no need for the RTP stream sender to delay resuming the stream until after sending TMMBN >0, or to apply the hold-off period specified in [RFC5104] before increasing the bitrate from zero.

TMMBN 0: Corresponds to PAUSED. Also corresponds to a REFUSE indication when a stream is requested to be resumed with TMMBR >0.

TMMBN >0: Cannot be used as REFUSE indication when a stream is requested to be paused with TMMBR 0, for reasons stated in Section 6.2.

7. Participant States

This document introduces three new states for a stream in an RTP sender, according to the figure and sub-sections below. Any references to PAUSE, PAUSED, RESUME and REFUSE in this section SHALL be taken to apply to the extent possible also when TMMBR/TMMBN are used (Section 6.5) for this functionality.

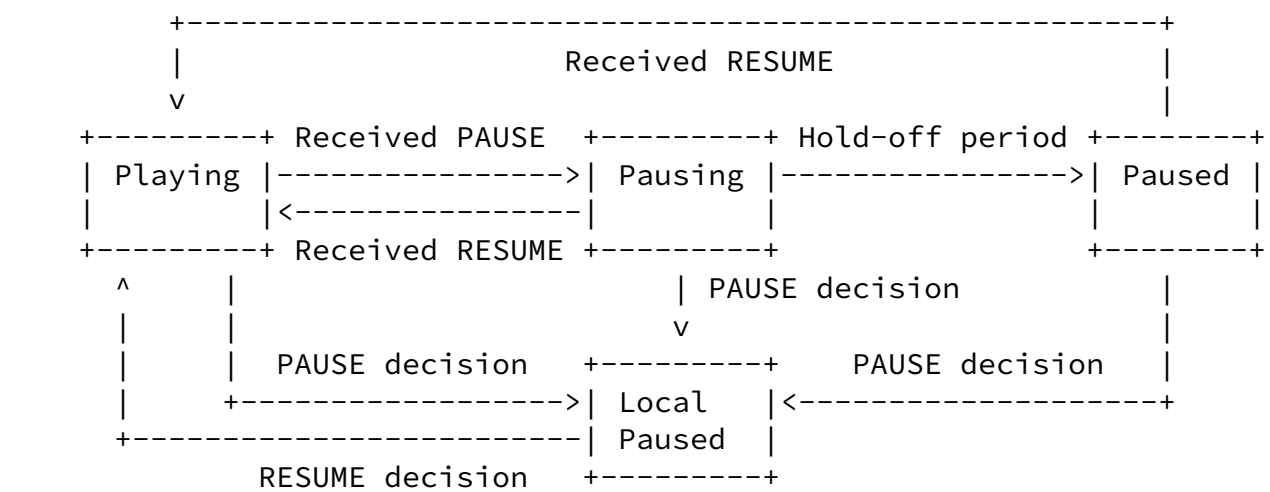


Figure 8: RTP Pause States

7.1. Playing State

This state is not new, but is the normal media sending state from [RFC3550]. When entering the state, the PauseID MUST be incremented by one in modulo arithmetic. The RTP sequence number for the first packet sent after a pause SHALL be incremented by one compared to the highest RTP sequence number sent before the pause. The first RTP Time Stamp for the first packet sent after a pause SHOULD be set according to capture times at the source.

7.2. Pausing State

In this state, the RTP stream sender has received at least one PAUSE message for the stream in question. The RTP stream sender SHALL wait during a hold-off period for the possible reception of RESUME messages for the RTP stream being paused before actually pausing RTP stream transmission. The period to wait SHALL be long enough to allow another RTP stream receiver to respond to the PAUSE with a RESUME, if it determines that it would not like to see the stream paused. This delay period (denoted by 'Hold-off period' in the figure) is determined by the formula:

```
2 * RTT + T_dither_max,
```

where RTT is the longest round trip known to the RTP stream sender and T_dither_max is defined in section 3.4 of [RFC4585]. The hold-off period MAY be set to 0 by some signaling (Section 10) means when it can be determined that there is only a single receiver, for example in point-to-point or some unicast situations.

If the RTP stream sender has set the hold-off period to 0 and receives information that it was an incorrect decision and that there are in fact several receivers of the stream, for example by RTCP RR, it MUST change the hold-off to instead be based on the above formula.

7.3. Paused State

An RTP stream is in paused state when the sender pauses its transmission after receiving at least one PAUSE message and the hold-off period has passed without receiving any RESUME message for that stream.

When entering the state, the RTP stream sender SHALL send a PAUSED indication to all known RTP stream receivers, and SHALL also repeat PAUSED in the next two regular RTCP reports.

Following sub-sections discusses some potential issues when an RTP sender goes into paused state. These conditions are also valid if an RTP Translator is used in the communication. When an RTP Mixer implementing this specification is involved between the participants (which forwards the stream by marking the RTP data with its own SSRC), it SHALL be a responsibility of the Mixer to control sending PAUSE and RESUME requests to the sender. The below conditions also apply to the sender and receiver parts of the RTP Mixer, respectively.

7.3.1. RTCP BYE Message

When a participant leaves the RTP session, it sends an RTCP BYE message. In addition to the semantics described in section 6.3.4 and 6.3.7 of RTP [RFC3550], following two conditions MUST also be considered when an RTP participant sends an RTCP BYE message,

- If a paused sender sends an RTCP BYE message, receivers observing this SHALL NOT send further PAUSE or RESUME requests to it.

- Since a sender pauses its transmission on receiving the PAUSE requests from any receiver in a session, the sender MUST keep record of which receiver that caused the RTP stream to pause. If that receiver sends an RTCP BYE message observed by the sender, the sender SHALL resume the RTP stream.

7.3.2. SSRC Time-out

Section 6.3.5 in RTP [RFC3550] describes the SSRC time-out of an RTP participant. Every RTP participant maintains a sender and receiver list in a session. If a participant does not get any RTP or RTCP packets from some other participant for the last five RTCP reporting

intervals it removes that participant from the receiver list. Any streams that were paused by that removed participant SHALL be resumed.

7.4. Local Paused State

This state can be entered at any time, based on local decision from the RTP stream sender. As for Paused State (Section 7.3), the RTP stream sender SHALL send a PAUSED indication to all known RTP stream receivers, when entering the state, and repeat it in the next two regular RTCP reports.

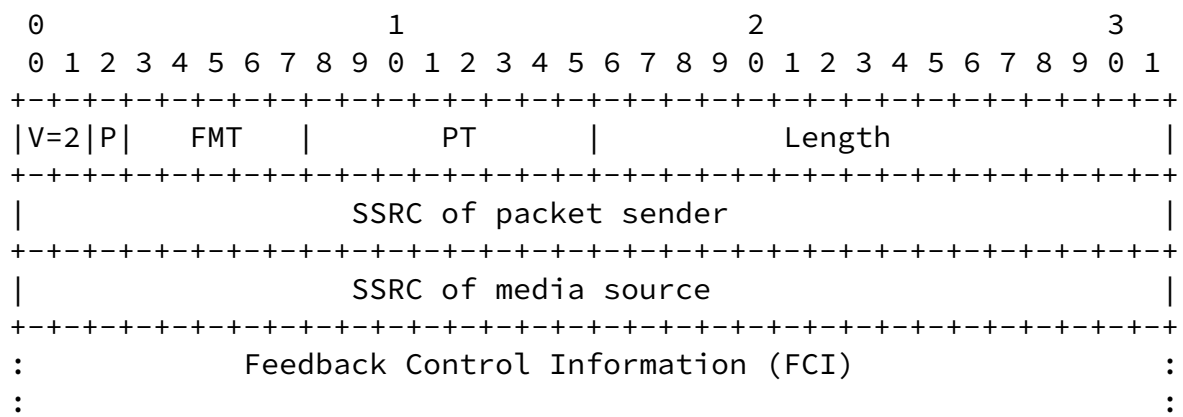
When leaving the state, the stream state SHALL become Playing, regardless whether or not there were any RTP stream receivers that sent PAUSE for that stream, effectively clearing the RTP stream sender's memory for that stream.

8. Message Format

Section 6 of AVPF [RFC4585] defines three types of low-delay RTCP feedback messages, i.e. Transport layer, Payload-specific, and Application layer feedback messages. This document defines a new Transport layer feedback message, this message is either a PAUSE request, a RESUME request, or one of four different types of acknowledgments in response to either PAUSE or RESUME requests.

The Transport layer feedback messages are identified by having the RTCP payload type be RTPFB (205) as defined by AVPF [RFC4585]. The PAUSE and RESUME messages are identified by Feedback Message Type (FMT) value in common packet header for feedback message defined in section 6.1 of AVPF [RFC4585]. The PAUSE and RESUME transport feedback message is identified by the FMT value = TBA1.

The Common Packet Format for Feedback Messages defined by AVPF [RFC4585] is:



For the PAUSE and RESUME messages, the following interpretation of the packet fields will be:

FMT: The FMT value identifying the PAUSE and RESUME message: TBA1

PT: Payload Type = 205 (RTPFB)

Length: As defined by AVPF, i.e. the length of this packet in 32-bit words minus one, including the header and any padding.

SSRC of packet sender: The SSRC of the RTP session participant sending the messages in the FCI. Note, for End Points that have multiple SSRCs in an RTP session, any of its SSRCs MAY be used to send any of the pause message types.

SSRC of media source: Not used, SHALL be set to 0. The FCI identifies the SSRC the message is targeted for.

The Feedback Control Information (FCI) field consist of one or more PAUSE, RESUME, PAUSED, REFUSE, or any future extension. These messages have the following FCI format:

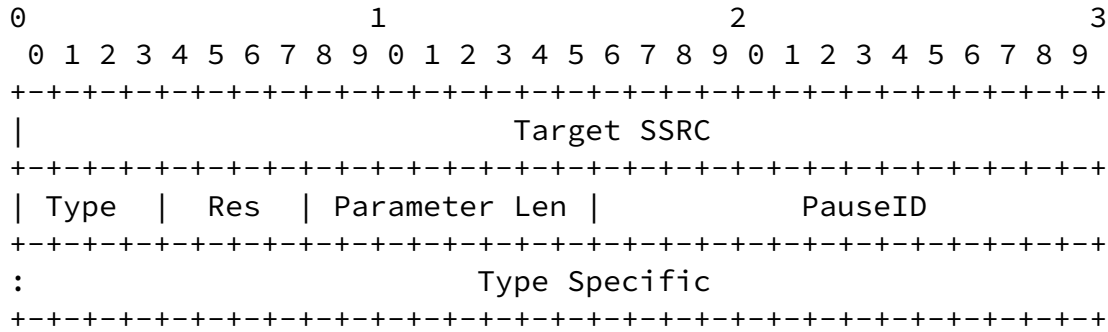


Figure 9: Syntax of FCI Entry in the PAUSE and RESUME message

The FCI fields have the following definitions:

Target SSRC (32 bits): For a PAUSE and RESUME messages, this value is the SSRC that the request is intended for. For PAUSED, it MUST be the SSRC being paused. If pausing is the result of a PAUSE request, the value in PAUSED is effectively the same as Target SSRC in a related PAUSE request. For REFUSE, it MUST be the Target SSRC of the PAUSE or RESUME request that cannot change

state. A CSRC MUST NOT be used as a target as the interpretation of such a request is unclear.

Type (4 bits): The pause feedback type. The values defined in this specification are as follows,

0: PAUSE request message

1: RESUME request message

2: PAUSED indication message

3: REFUSE indication message

4-15: Reserved for future use

Res: (4 bits): Type specific reserved. SHALL be ignored by receivers implementing this specification and MUST be set to 0 by senders implementing this specification.

Parameter Len: (8 bits): Length of the Type Specific field in 32-bit words. MAY be 0.

PauseID (16 bits): Message sequence identification. SHALL be incremented by one modulo 2^16 for each new PAUSE message, unless the message is re-transmitted. The initial value SHOULD be 0. The PauseID is scoped by the Target SSRC, meaning that PAUSE, RESUME, and PAUSED messages therefore share the same PauseID space for a specific Target SSRC.

Type Specific: (variable): Defined per pause feedback Type. MAY be empty.

9. Message Details

This section contains detailed explanations of each message defined in this specification. All transmissions of request and indications are governed by the transmission rules as defined by Section 9.5.

Any references to PAUSE, PAUSED, RESUME and REFUSE in this section SHALL be taken to apply to the extent possible also when TMMBR/TMMBN are used (Section 6.5) for this functionality. TMMBR/TMMBN MAY be

used instead of the messages defined in this specification when the effective topology is point-to-point. If either sender or receiver learns that the topology is not point-to-point, TMMBR/TMMBN MUST NOT be used for pause/resume functionality. If the messages defined in this specification are supported in addition to TMMBR/TMMBN, pause/resume signaling MUST revert to use those instead. If the topology is not point-to-point and the messages defined in this specification are not supported, pause/resume functionality with TMMBR/TMMBN MUST NOT be used.

9.1. PAUSE

An RTP stream receiver MAY schedule PAUSE for transmission at any time.

PAUSE has no defined Type Specific parameters and Parameter Len MUST be set to 0.

PauseID SHOULD be the available PauseID, as indicated by PAUSED (Section 9.2) or implicitly determined by previously received PAUSE or RESUME (Section 9.3) requests. A randomly chosen PauseID MAY be used if it was not possible to retrieve PauseID information, in which case the PAUSE will either succeed, or the correct PauseID can be found in the returned REFUSE (Section 9.4). A PauseID that is matching the available PauseID is henceforth also called a valid PauseID.

PauseID needs to be incremented by one, in modulo arithmetic, for each PAUSE request that is not a retransmission, compared to what was used in the last PAUSED indication sent by the media sender. This is to ensure that the PauseID matches what is the current available PauseID at the RTP stream sender. The RTP stream sender increments what it considers to be the available PauseID when entering Playing State (Section 7.1).

For the scope of this specification, a PauseID larger than the current one is defined as having a value between and including (PauseID + 1) MOD 2^16 and (PauseID + 2^14) MOD 2^16, where "MOD" is the modulo operator. Similarly, a PauseID smaller than the current

one is defined as having a value between and including (PauseID - 2^15) MOD 2^16 and (PauseID - 1) MOD 2^16.

If an RTP stream receiver that sent a PAUSE with a certain PauseID receives a RESUME with the same PauseID, it is RECOMMENDED that it refrains from sending further PAUSE requests for some appropriate time since the RESUME indicates that there are other receivers that still wishes to receive the stream.

If the targeted RTP stream does not pause, if no PAUSED indication with a larger PauseID than the one used in PAUSE, and if no REFUSE is received within 2 * RTT + T_dither_max, the PAUSE MAY be scheduled for retransmission, using the same PauseID. RTT is the observed round-trip to the RTP stream sender and T_dither_max is defined in section 3.4 of [RFC4585].

When an RTP stream sender in Playing State (Section 7.1) receives a valid PAUSE, and unless local considerations currently makes it impossible to pause the stream, it SHALL enter Pausing State

(Section 7.2) when reaching an appropriate place to pause in the stream, and act accordingly.

If an RTP stream sender receives a valid PAUSE while in Pausing, Paused (Section 7.3) or Local Paused (Section 7.4) States, the received PAUSE SHALL be ignored.

9.2. PAUSED

The PAUSED indication MAY be sent either as a result of a valid PAUSE (Section 9.1) request, when entering Paused State (Section 7.3), or based on a RTP stream sender local decision, when entering Local Paused State (Section 7.4).

PauseID MUST contain the available, valid value to be included in a subsequent RESUME (Section 9.3).

PAUSED SHALL contain a 32 bit parameter with the RTP extended highest sequence number valid when the RTP stream was paused. Parameter Len MUST be set to 1.

After having entered Paused or Local Paused State and thus having

sent PAUSED once, PAUSED MUST also be included in the next two regular RTCP reports, given that the pause condition is then still effective.

While remaining in Paused or Local Paused States, PAUSED MAY be included in all regular RTCP reports.

When in Paused or Local Paused States, It is RECOMMENDED to send PAUSED at the earliest opportunity and also to include it in the next two regular RTCP reports, whenever the RTP stream sender learns that there are End Points that did not previously receive the stream, for example by RTCP reports with an SSRC and a CNAME that was not previously seen in the RTP session.

9.3. RESUME

An RTP stream receiver MAY schedule RESUME for transmission whenever it wishes to resume a paused stream, or to disapprove a stream from being paused.

PauseID SHOULD be the valid PauseID, as indicated by PAUSED (Section 9.2) or implicitly determined by previously received PAUSE (Section 9.1) or RESUME requests. A randomly chosen PauseID MAY be used if it was not possible to retrieve PauseID information, in which case the RESUME will either succeed, or the correct PauseID can be found in a returned REFUSE (Section 9.4).

RESUME has no defined Type Specific parameters and Parameter Len MUST be set to 0.

When an RTP stream sender in Pausing (Section 7.2), Paused (Section 7.3) or Local Paused State (Section 7.4) receives a valid RESUME, and unless local considerations currently makes it impossible to resume the stream, it SHALL enter Playing State (Section 7.1) and act accordingly. If the RTP stream sender is incapable of honoring the RESUME request with a valid PauseID, or receives a RESUME request with an invalid PauseID while in Paused or Pausing state, the RTP stream sender sends a REFUSE message as specified below.

If an RTP stream sender in Playing State receives a RESUME containing either a valid PauseID or a PauseID that is less than the valid PauseID, the received RESUME SHALL be ignored.

9.4. REFUSE

REFUSE has no defined Type Specific parameters and Parameter Len MUST be set to 0.

If an RTP stream sender receives a valid PAUSE (Section 9.1) or RESUME (Section 9.3) request that cannot be fulfilled by the sender due to some local consideration, it SHALL schedule transmission of a REFUSE indication containing the valid PauseID from the rejected request.

If an RTP stream sender receives PAUSE or RESUME requests with a non-valid PauseID it SHALL schedule a REFUSE response containing the available, valid PauseID, except if the RTP stream sender is in Playing State and receives a RESUME with a PauseID less than the valid one, in which case the RESUME SHALL be ignored.

If several PAUSE or RESUME that would render identical REFUSE responses are received before the scheduled REFUSE is sent, duplicate REFUSE MUST NOT be scheduled for transmission. This effectively lets a single REFUSE respond to several invalid PAUSE or RESUME requests.

If REFUSE containing a certain PauseID was already sent and yet more PAUSE or RESUME messages are received that require additional REFUSE with that specific PauseID to be scheduled, and unless the PauseID number space has wrapped since REFUSE was last sent with that PauseID, further REFUSE messages with that PauseID SHOULD be sent in regular RTCP reports.

An RTP stream receiver that sent a PAUSE or RESUME request and receives a REFUSE containing the same PauseID as in the request

SHOULD refrain from sending an identical request for some appropriate time to allow the condition that caused REFUSE to clear.

An RTP stream receiver that sent a PAUSE or RESUME request and receives a REFUSE containing a PauseID different from the request MAY schedule another request using the PauseID from the REFUSE indication.

9.5. Transmission Rules

The transmission of any RTCP feedback messages defined in this specification MUST follow the normal AVPF defined timing rules and depends on the session's mode of operation.

All messages defined in this specification MAY use either Regular, Early or Immediate timings, taking the following into consideration:

- PAUSE SHOULD use Early or Immediate timing, except for retransmissions that SHOULD use Regular timing.
- The first transmission of PAUSED for each (non-wrapped) PauseID SHOULD be sent with Immediate or Early timing, while subsequent PAUSED for that PauseID SHOULD use Regular timing.
- RESUME SHOULD always use Immediate or Early timing.
- The first transmission of REFUSE for each (non-wrapped) PauseID SHOULD be sent with Immediate or Early timing, while subsequent REFUSE for that PauseID SHOULD use Regular timing.

10. Signaling

The capability of handling messages defined in this specification MAY be exchanged at a higher layer such as SDP. This document extends the rtcp-fb attribute defined in section 4 of AVPF [RFC4585] to include the request for pause and resume. Like AVPF [RFC4585] and CCM [RFC5104], it is RECOMMENDED to use the rtcp-fb attribute at media level and it MUST NOT be used at session level. This specification follows all the rules defined in AVPF for rtcp-fb attribute relating to payload type in a session description.

> Note: When TMMBR 0 / TMMBN 0 are used to implement pause and resume functionality (with the restrictions described in this memo), signaling rtcp-fb attribute with ccm tmmbr parameter is sufficient and no further signaling is necessary.

This specification defines two new parameters to the "ccm" feedback value defined in CCM [RFC5104], "pause" and "paused".

- "pause" represents the capability to understand the RTCP feedback

message and all of the defined FCIs of PAUSE, RESUME, PAUSED and REFUSE. A direction sub-parameter is used to determine if a given node desires to issue PAUSE or RESUME requests, can respond to PAUSE or RESUME requests, or both.

- "paused" represents the functionality of supporting the playing and local paused states and generate PAUSED FCI when a stream delivery is paused. A direction sub-parameter is used to determine if a given node desires to receive these indications, intends to send them, or both.

The reason for this separation is to make it possible for partial implementation of this specification, according to the different roles in the use cases section (Section 3).

A sub-parameter named "nowait", indicating that the hold-off time defined in Section 7.2 can be set to 0, reducing the latency before the stream can paused after receiving a PAUSE request. This condition occurs when there will be only a single receiver per direction in the RTP session, for example in point-to-point sessions. It is also possible to use in scenarios using unidirectional media. The conditions that allow "nowait" to be set also indicate that it would be possible to use CCM TMMBR/TMMBN as pause/resume signaling.

A sub-parameter named "dir" is used to indicate in which directions a given node will use the pause or paused functionality. The node being configured or issuing an offer or an answer uses the directionality in the following way. Note that pause and paused have separate and different definitions.

Direction ("dir") values for "pause" is defined as follows:

sendonly: The node intends to send PAUSE and RESUME requests for other nodes' streams and is thus also capable of receiving PAUSED and REFUSE. It will not support receiving PAUSE and RESUME requests.

recvonly: The node supports receiving PAUSE and RESUME requests targeted for streams sent by the node. It will send PAUSED and REFUSE as needed. The node will not send any PAUSE and RESUME requests.

sendrecv: The node supports receiving PAUSE and RESUME requests targeted for streams sent by the node. The node intends to send PAUSE and RESUME requests for other nodes' streams. Thus the node is capable of sending and receiving all types of pause messages.

This is the default value. If the "dir" parameter is omitted, it MUST be interpreted to represent this value.

Direction values for "paused" is defined as follows:

sendonly: The node intends to send PAUSED indications whenever it pauses RTP stream delivery in any of its streams. It has no need to receive PAUSED indications itself.

recvonly: The node desires to receive PAUSED indications whenever any stream sent by another node is paused. It does not intend to send any PAUSED indications.

sendrecv: The nodes desires to receive PAUSED indications and intends to send PAUSED indications whenever any stream is paused. This is the default value. If the "dir" parameter is omitted, it MUST be interpreted to represent this value.

This is the resulting ABNF [RFC5234], extending existing ABNF in section 7.1 of CCM [RFC5104]:

```
rtcp-fb-ccm-param =/ SP "pause" *(SP pause-attr)
                     / SP "paused" *(SP paused-attr)
pause-attr         = direction
                   / "nowait"
                   / token ; for future extensions
paused-attr        = direction
                   / token ; for future extensions
direction          = "dir=" direction-alts
direction-alts     = "sendonly" / "recvonly" / "sendrecv"
```

Figure 10: ABNF

An endpoint implementing this specification and using SDP to signal capability SHOULD indicate both of the new "pause" and "paused" parameters with ccm signaling. When negotiating usage, it is possible select either of them, noting that "pause" contain the full "paused" functionality. A sender or receiver SHOULD NOT use the messages from this specification towards receivers that did not declare capability for it.

There MUST NOT be more than one "a=rtcp-fb" line with "pause" and one with "paused" applicable to a single payload type in the SDP, unless the additional line uses "*" as payload type, in which case "*" SHALL be interpreted as applicable to all listed payload types that does

not have an explicit "pause" or "paused" specification.

There MUST NOT be more than a single direction sub-parameter per "pause" and "paused" parameter. There MUST NOT be more than a single "nowait" sub-parameter per "pause" parameter.

10.1. Offer-Answer Use

An offerer implementing this specification needs to include "pause" and/or "paused" CCM parameters with suitable directionality parameter ("dir") in the SDP, according to what messages it intends to send and desires or is capable to receive in the session. It is RECOMMENDED to include both "pause" and "paused" if "pause" is supported, to enable at least the "paused" functionality if the answer only supports "paused" or different directionality for the two functionalities. The "pause" and "paused" functionalities are negotiated independently, although the "paused" functionality is part of the "pause" functionality. As a result, an answerer MAY remove "pause" or "paused" lines from the SDP depending on the agreed mode of functionality.

In offer/answer, the "dir" parameter is interpreted based on the agent providing the SDP. The node described in the offer is the offerer, and the answerer is described in an answer. In other words, an offer for "paused dir=sendonly" means that the offerer intends to send PAUSED indications whenever it pauses RTP stream delivery in any of its streams.

An answerer receiving an offer with a "pause" parameter with dir=sendrecv MAY remove the pause line in its answer, respond with pause keeping sendrecv for full bi-directionality, or it may change dir value to either sendonly or recvonly based on its capabilities and desired functionality. An offer with a "pause" parameter with dir=sendonly or dir=recvonly is either completely removed or accepted with reverse directionality, i.e. sendonly becomes recvonly or recvonly becomes sendonly.

An answer receiving an offer with "paused" has the same choices as for "pause" above. It should be noted that the directionality of pause is the inverse of RTP stream direction, while the directionality of paused is the same as the RTP stream direction.

If the offerer believes that itself and the intended answerer are likely the only End Points in the RTP session, it MAY include the "nowait" sub-parameter on the "pause" line in the offer. If an answerer receives the "nowait" sub-parameter on the "pause" line in the SDP, and if it has information that the offerer and itself are not the only End Points in the RTP session, it MUST NOT include any "nowait" sub-parameter on its "pause" line in the SDP answer. The answerer MUST NOT add "nowait" on the "pause" line in the answer

unless it is present on the "paused" line in the offer. If both offer and answer contained a "nowait" parameter, then the hold-off time is configured to 0 at both offerer and answerer.

10.2. Declarative Use

In declarative use, the SDP is used to configure the node receiving the SDP. This has implications on the interpretation of the SDP signaling extensions defined in this draft. First, it is normally only necessary to include either "pause" or "paused" parameter to indicate the level of functionality the node should use in this RTP session. Including both is only necessary if some implementations only understands "paused" and some other can understand both. Thus indicating both means use pause if you understand it, and if you only understand paused, use that.

The "dir" directionality parameter indicates how the configured node should behave. For example "pause" with sendonly:

sendonly: The node intends to send PAUSE and RESUME requests for other nodes' streams and is thus also capable of receiving PAUSED and REFUSE. It will not support receiving PAUSE and RESUME requests.

In this example, the configured node should send PAUSE and RESUME requests if has reason for it. It does not need to respond to any PAUSE or RESUME requests as that is not supported.

The "nowait" parameter, if included, is followed as specified. It is the responsibility of the declarative SDP sender to determine if a configured node will participate in a session that will be point to point, based on the usage. For example, a conference client being

configured for an any source multicast session using SAP [RFC2974] will not be in a point to point session, thus "nowait" cannot be included. An RTSP [RFC2326] client receiving a declarative SDP may very well be in a point to point session, although it is highly doubtful that an RTSP client would need to support this specification, considering the inherent PAUSE support in RTSP.

11. Examples

The following examples shows use of PAUSE and RESUME messages, including use of offer-answer:

1. Offer-Answer

2. Point-to-Point session

3. Point-to-Multipoint using Mixer

4. Point-to-Multipoint using Translator

11.1. Offer-Answer

The below figures contains an example how to show support for pausing and resuming the streams, as well as indicating whether or not the hold-off period can be set to 0.

```
v=0
o=alice 3203093520 3203093520 IN IP4 alice.example.com
s=Pausing Media
t=0 0
c=IN IP4 alice.example.com
m=audio 49170 RTP/AVPF 98 99
a=rtpmap:98 G719/48000
a=rtpmap:99 PCMA/8000
a=rtcp-fb:* ccm pause nowait
a=rtcp-fb:* ccm paused
```

Figure 11: SDP Offer With Pause and Resume Capability

The offerer supports all of the messages defined in this

specification and offers a sendrecv stream. The offerer also believes that it will be the sole receiver of the answerer's stream as well as that the answerer will be the sole receiver of the offerer's stream and thus includes the "nowait" sub-parameter for both "pause" and "paused" parameters.

This is the SDP answer:

```
v=0
o=bob 293847192 293847192 IN IP4 bob.example.com
s=-
t=0 0
c=IN IP4 bob.example.com
m=audio 49202 RTP/AVPF 98
a=rtpmap:98 G719/48000
a=rtcp-fb:98 ccm pause dir=sendonly
a=rtcp-fb:98 ccm paused
```

Figure 12: SDP Answer With Pause and Resume Capability

The answerer will not allow its sent streams to be paused or resumed and thus support pause only in sendonly mode. It does support paused

and intends to send it, and also desires to receive PAUSED indications. Thus paused in sendrecv mode is included in the answer. The answerer somehow knows that it will not be a point-to-point RTP session and has therefore removed "nowait" from the "pause" line, meaning that the offerer must use a non-zero hold-off time when being requested to pause the stream.

When using TMMBR 0 / TMMBN 0 to achieve pause and resume functionality, there are no differences in SDP compared to CCM [RFC5104] and therefore no such examples are included here.

11.2. Point-to-Point Session

This is the most basic scenario, which involves two participants, each acting as a sender and/or receiver. Any RTP data receiver sends PAUSE or RESUME messages to the sender, which pauses or resumes transmission accordingly. The hold-off time before pausing a stream is 0.

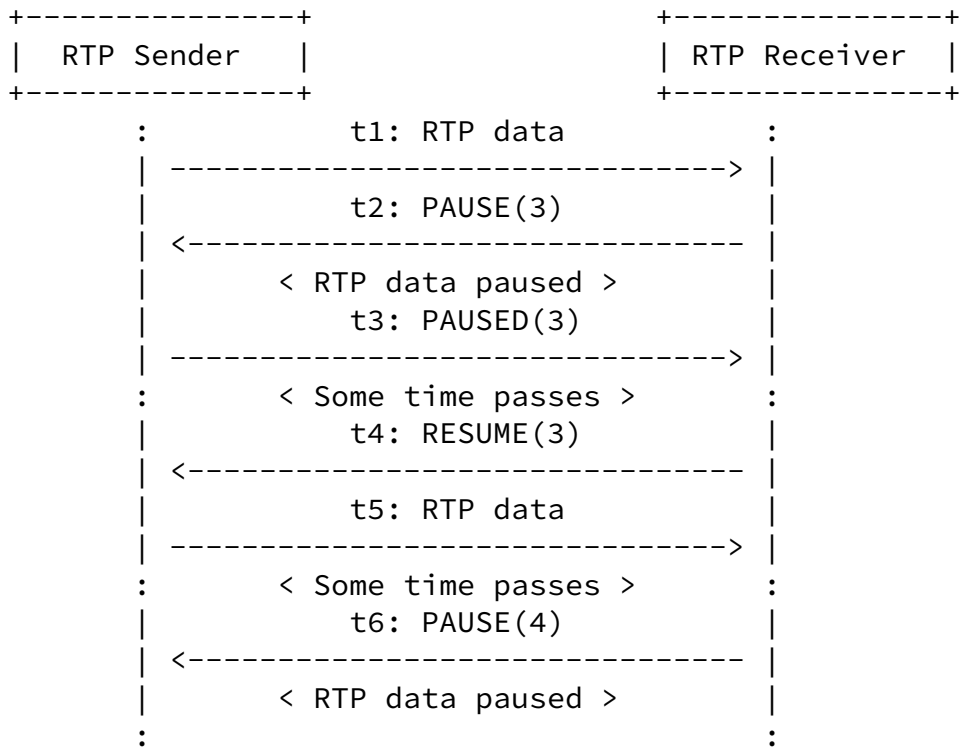
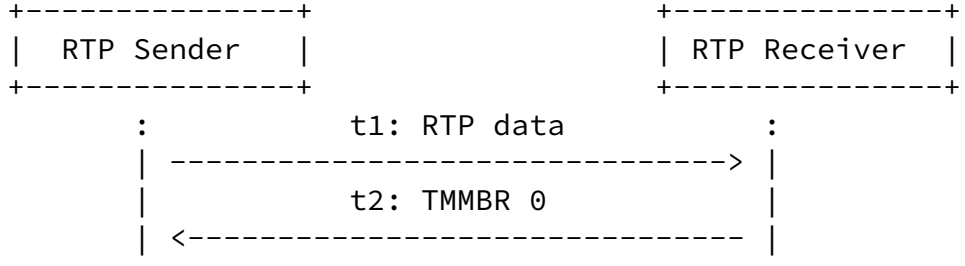


Figure 13: Pause and Resume Operation in Point-to-Point

Figure 13 shows the basic pause and resume operation in Point-to-Point scenario. At time t1, an RTP sender sends data to a receiver. At time t2, the RTP receiver requests the sender to pause the stream, using PauseID 3 (which it knew since before in this example). The sender pauses the data and replies with a PAUSED containing the same

PauseID. Some time later (at time t4) the receiver requests the sender to resume, which resumes its transmission. The next PAUSE, sent at time t6, contains an updated PauseID (4).



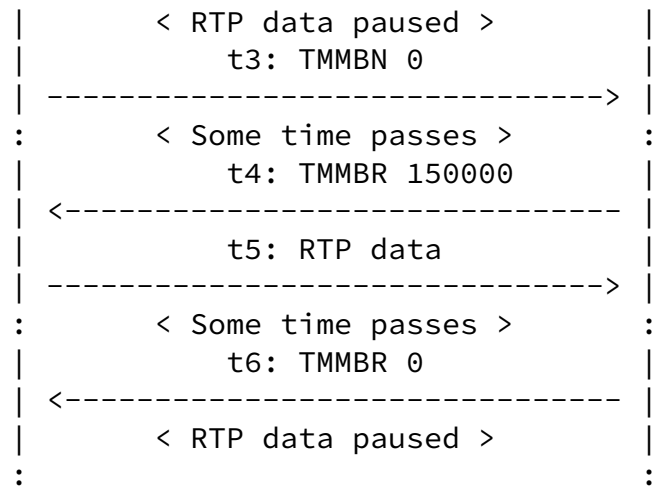
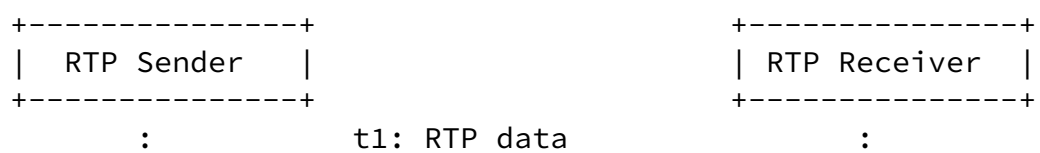


Figure 14: TMMBR Pause and Resume in Point-to-Point

Figure 14 describes the same point-to-point scenario as above, but using TMMBR/TMMBN signaling.



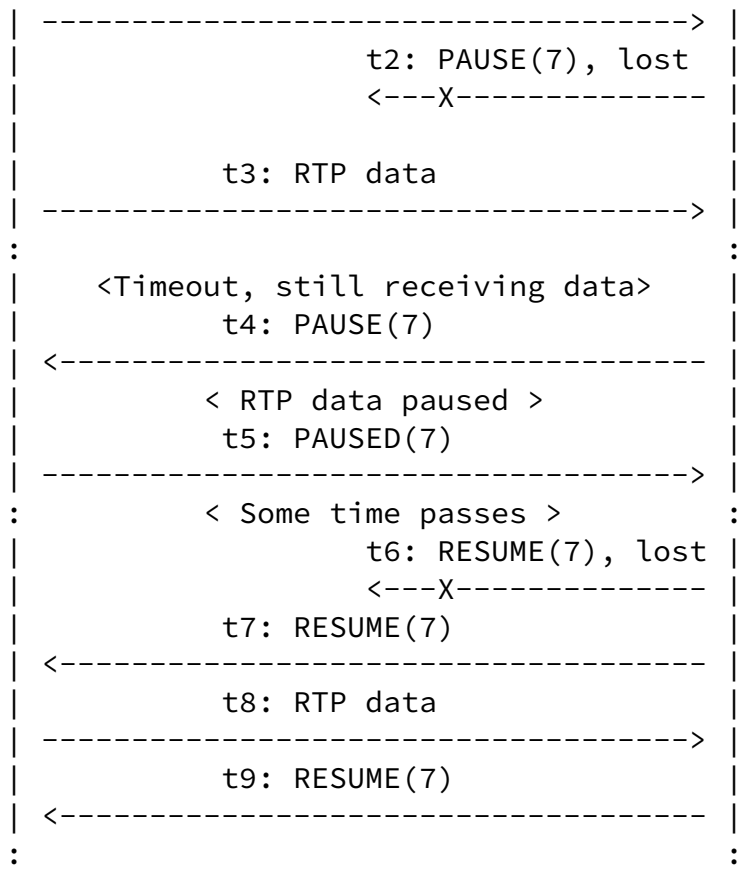


Figure 15: Pause and Resume Operation With Messages Lost

Figure 15 describes what happens if a PAUSE message from an RTP stream receiver does not reach the RTP stream sender. After sending a PAUSE message, the RTP stream receiver waits for a time-out to detect if the RTP stream sender has paused the data transmission or has sent PAUSED indication according to the rules discussed in Section 7.3. As the PAUSE message is lost on the way (at time t2), RTP data continues to reach to the RTP stream receiver. When the timer expires, the RTP stream receiver schedules a retransmission of the PAUSE message, which is sent at time t4. If the PAUSE message now reaches the RTP stream sender, it pauses the RTP stream and replies with PAUSED.

At time t6, the RTP stream receiver wishes to resume the stream again and sends a RESUME, which is lost. This does not cause any severe effect, since there is no requirement to wait until further RESUME are sent and another RESUME are sent already at time t7, which now reaches the RTP stream sender that consequently resumes the stream at

time t8. The time interval between t6 and t7 can vary, but may for example be one RTCP feedback transmission interval as determined by the AVPF rules.

The RTP stream receiver did not realize that the RTP stream was resumed in time to stop yet another scheduled RESUME from being sent at time t9. This is however harmless since the RESUME PauseID is less than the valid one and will be ignored by the RTP stream sender. It will also not cause any unwanted resume even if the stream was paused based on a PAUSE from some other receiver before receiving the RESUME, since the valid PauseID is now larger than the one in the stray RESUME and will only cause a REFUSE containing the new valid PauseID from the RTP stream sender.

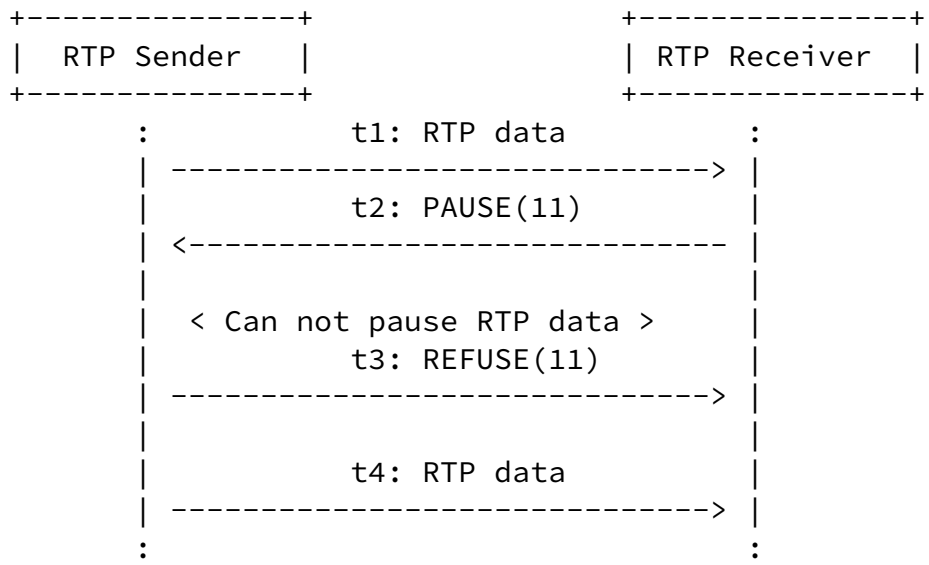


Figure 16: Pause Request is Refused in Point-to-Point

In Figure 16, the receiver requests to pause the sender, which refuses to pause due to some consideration local to the sender and responds with a REFUSE message.

11.3. Point-to-Multipoint using Mixer

An RTP Mixer is an intermediate node connecting different transport-level clouds. The Mixer receives streams from different RTP sources, selects or combines them based on the application's needs and forwards the generated stream(s) to the destination. The Mixer typically puts its' own SSRC(s) in RTP data packets instead of the original source(s).

The Mixer keeps track of all the streams delivered to the Mixer and

how they are currently used. In this example, it selects the video

stream to deliver to the receiver R based on the voice activity of the RTP stream senders. The video stream will be delivered to R using M's SSRC and with an CSRC indicating the original source.

Note that PauseID is not of any significance for the example and is therefore omitted in the description.

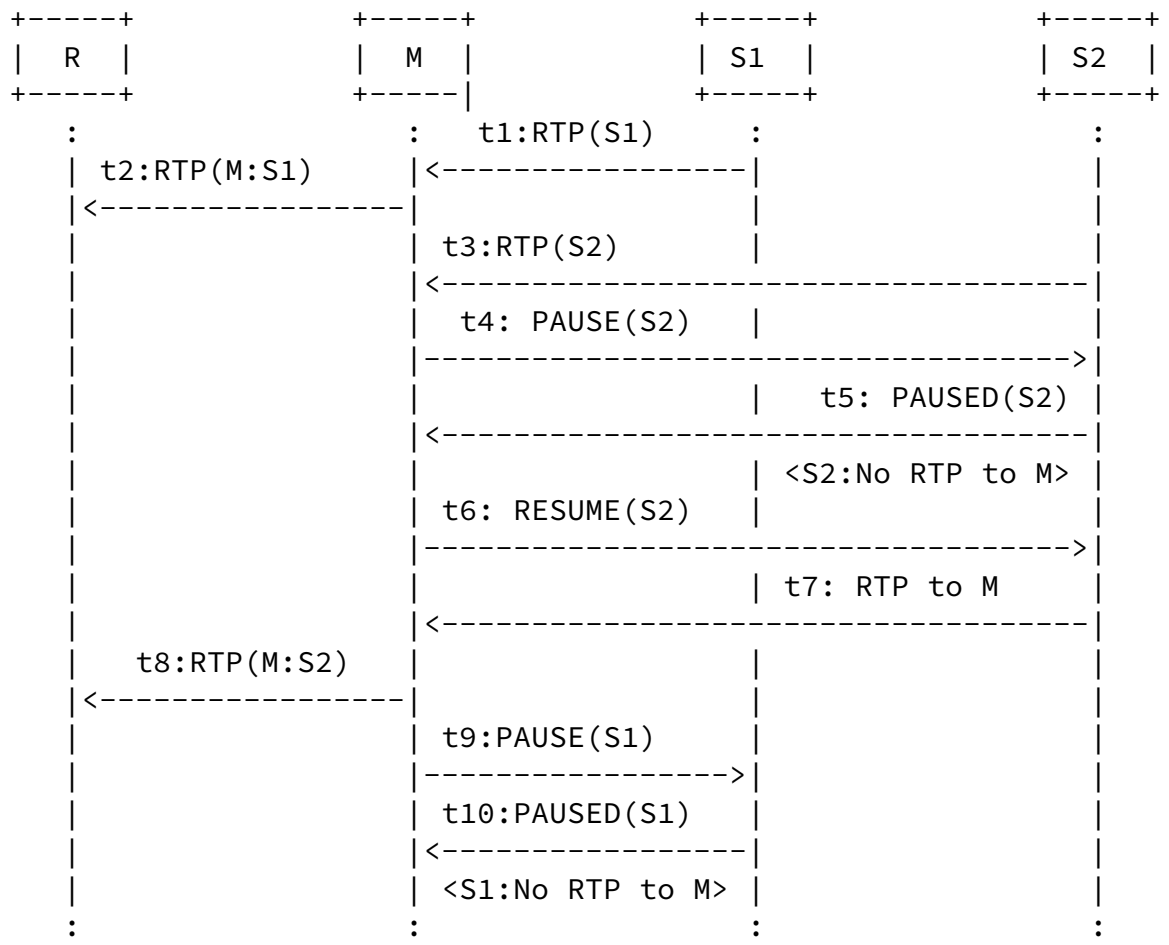


Figure 17: Pause and Resume Operation for a Voice Activated Mixer

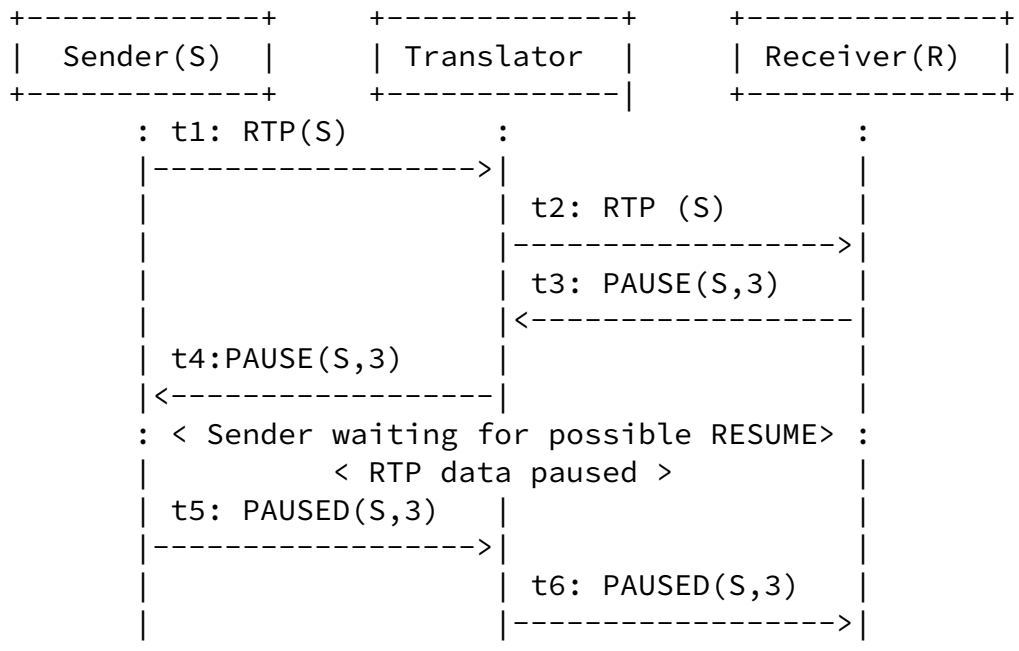
The session starts at t1 with S1 being the most active speaker and thus being selected as the single video stream to be delivered to R (t2) using the Mixer SSRC but with S1 as CSRC (indicated after the colon in the figure). Then S2 joins the session at t3 and starts

delivering an RTP stream to the Mixer. As S2 has less voice activity then S1, the Mixer decides to pause S2 at t4 by sending S2 a PAUSE request. At t5, S2 acknowledges with a PAUSED and at the same instant stops delivering RTP to the Mixer. At t6, the user at S2 starts speaking and becomes the most active speaker and the Mixer decides to switch the video stream to S2, and therefore quickly sends a RESUME request to S2. At t7, S2 has received the RESUME request and acts on it by resuming RTP stream delivery to M. When the RTP

stream from t7 arrives at the Mixer, it switches this RTP stream into its SSRC (M) at t8 and changes the CSRC to S2. As S1 now becomes unused, the Mixer issues a PAUSE request to S1 at t9, which is acknowledged at t10 with a PAUSED and the RTP stream from S1 stops being delivered.

11.4. Point-to-Multipoint using Translator

A transport Translator in an RTP session forwards the message from one peer to all the others. Unlike Mixer, the Translator does not mix the streams or change the SSRC of the messages or RTP media. These examples are to show that the messages defined in this specification can be safely used also in a transport Translator case. The parentheses in the figures contains (Target SSRC, PauseID) information for the messages defined in this specification.



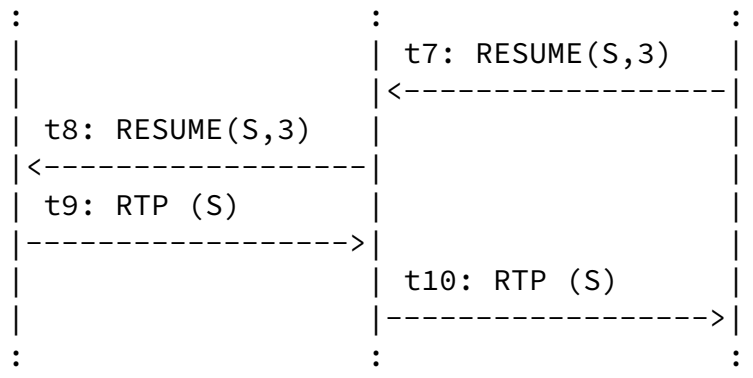
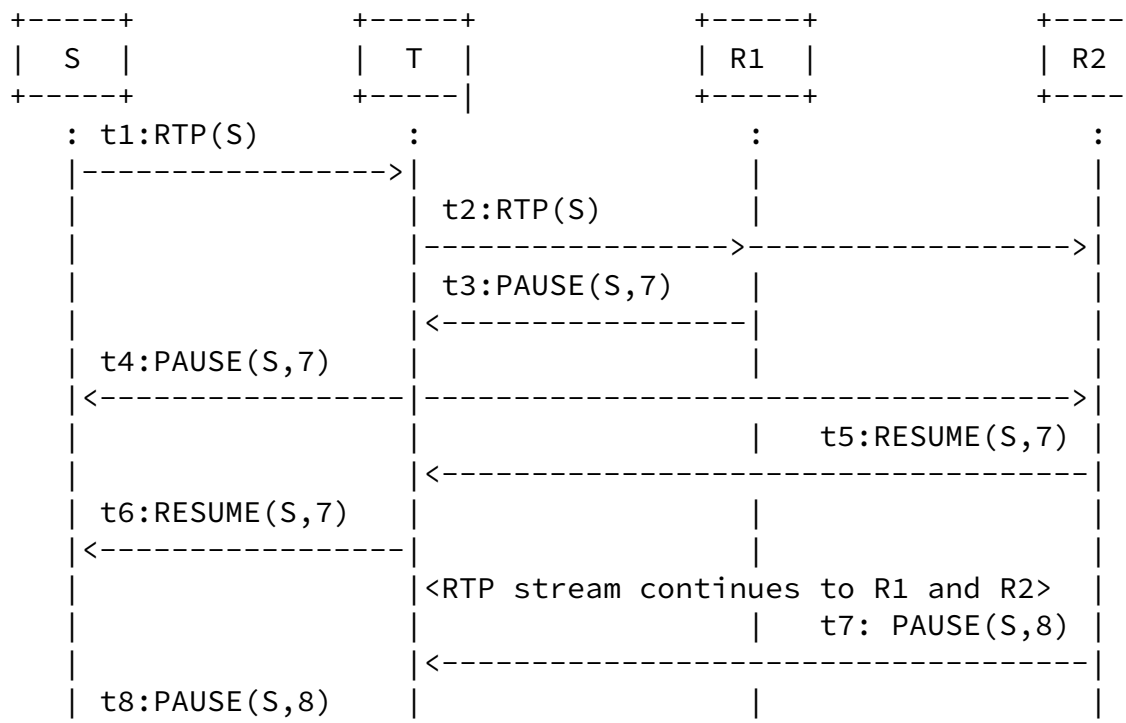


Figure 18: Pause and Resume Operation Between Two Participants Using a Translator

Figure 18 describes how a Translator can help the receiver in pausing and resuming the sender. The sender S sends RTP data to the receiver R through Translator, which just forwards the data without modifying the SSRCs. The receiver sends a PAUSE request to the sender, which in this example knows that there may be more receivers of the stream and waits a non-zero hold-off time to see if there is any other receiver that wants to receive the data, does not receive any disapproving RESUME, hence pauses itself and replies with PAUSED. Similarly the receiver resumes the sender by sending RESUME request through Translator. Since this describes only a single pause operation for a single RTP stream sender, all messages uses a single PauseID, in this example 3.


S
T
R1
R2
t1:RTP(S)
t2:RTP(S)
t3:PAUSE(S,7)
t4:PAUSE(S,7)
t5:RESUME(S,7)
t6:RESUME(S,7)
<RTP stream continues to R1 and R2>
t7: PAUSE(S,8)
t8:PAUSE(S,8)

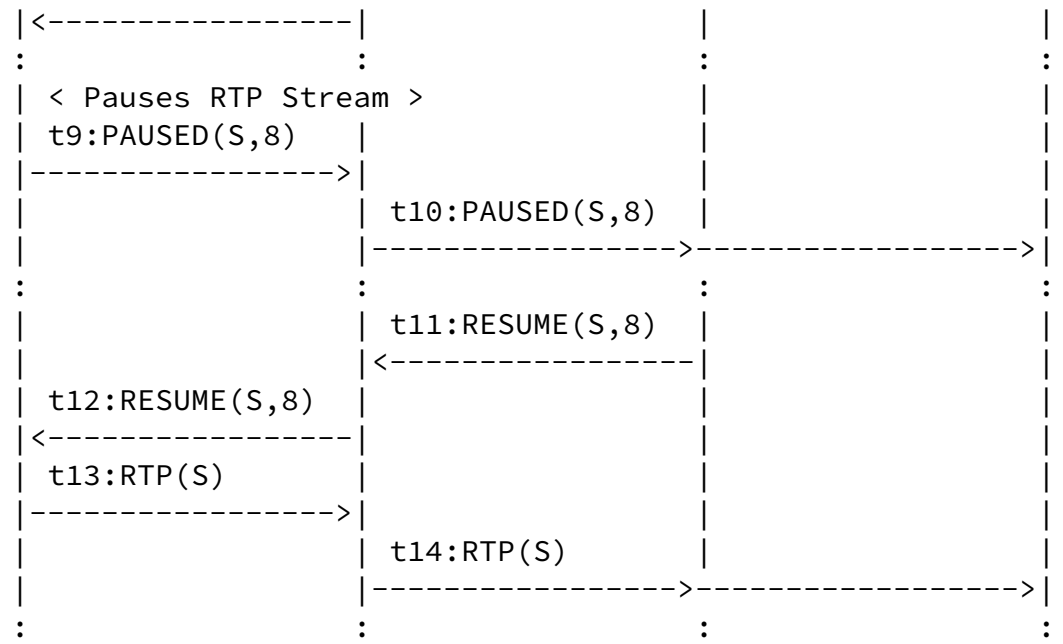


Figure 19: Pause and Resume Operation Between One Sender and Two Receivers Through Translator

Figure 19 explains the pause and resume operations when a transport Translator is involved between a sender and two receivers in an RTP session. Each message exchange is represented by the time it happens. At time t1, Sender (S) starts sending an RTP stream to the Translator, which is forwarded to R1 and R2 through the Translator, T. R1 and R2 receives RTP data from Translator at t2. At this

point, both R1 and R2 will send RTCP Receiver Reports to S informing that they receive S's stream.

After some time (at t3), R1 chooses to pause the stream. On receiving the PAUSE request from R1 at t4, S knows that there are at least one receiver that may still want to receive the data and uses a non-zero hold-off period to wait for possible RESUME messages. R2 did also receive the PAUSE request at time t4 and since it still wants to receive the stream, it sends a RESUME for it at time t5, which is forwarded to the sender S by the translator T. The sender S sees the RESUME at time t6 and continues to send data to T which forwards to both R1 and R2. At t7, the receiver R2 chooses to pause

the stream by sending a PAUSE request with an updated PauseID. The sender S still knows that there are more than one receiver (R1 and R2) that may want the stream and again waits a non-zero hold-off time, after which and not having received any disapproving RESUME, it concludes that the stream must be paused. S now stops sending the stream and replies with PAUSED to R1 and R2. When any of the receivers (R1 or R2) chooses to resume the stream from S, in this example R1, it sends a RESUME request to the sender. The RTP sender immediately resumes the stream.

Consider also an RTP session which includes one or more receivers, paused sender(s), and a Translator. Further assume that a new participant joins the session, which is not aware of the paused sender(s). On receiving knowledge about the newly joined participant, e.g. any RTP traffic or RTCP report (i.e. either SR or RR) from the newly joined participant, the paused sender(s) immediately sends PAUSED indications for the paused streams since there is now a receiver in the session that did not pause the sender(s) and may want to receive the streams. Having this information, the newly joined participant has the same possibility as any other participant to resume the paused streams.

12. IANA Considerations

As outlined in Section 8, this specification requests IANA to allocate

1. The FMT number TBA1 to be allocated to the PAUSE and RESUME functionality from this specification.

2. The 'pause' and 'paused' tags to be used with ccm under rtcp-fb AVPF attribute in SDP.

3. The 'nowait' parameter to be used with the 'pause' and 'paused' tags in SDP.

4. A registry listing registered values for 'pause' Types.

5. PAUSE, RESUME, PAUSED, and REFUSE with the listed numbers in the pause Type registry.

13. Security Considerations

This document extends the CCM [RFC5104] and defines new messages, i.e. PAUSE and RESUME. The exchange of these new messages MAY have some security implications, which need to be addressed by the user. Following are some important implications,

1. Identity spoofing - An attacker can spoof him/herself as an authenticated user and can falsely pause or resume any source transmission. In order to prevent this type of attack, a strong authentication and integrity protection mechanism is needed.

2. Denial of Service (DoS) - An attacker can falsely pause all source streams which MAY result in Denial of Service (DoS). An Authentication protocol may prevent this attack.

3. Man-in-Middle Attack (MiMT) - The pausing and resuming of an RTP source is prone to a Man-in-Middle attack. Public key authentication may be used to prevent MiMT.

14. Contributors

Daniel Grondal contributed in the creation and writing of earlier versions of this specification.

15. Acknowledgements

Daniel Grondal made valuable contributions during the initial versions of this draft.

16. References

16.1. 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.

[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.

[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.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

16.2. 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-avtext-rtp-grouping-taxonomy]
Lennox, J., Gross, K., Nandakumar, S., and G. Salgueiro, "A Taxonomy of Grouping Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", draft-ietf-avtext-rtp-grouping-taxonomy-01 (work in progress), February 2014.

[I-D.ietf-rtcweb-use-cases-and-requirements]
Holmberg, C., Hakansson, S., and G. Eriksson, "Web Real-Time Communication Use-cases and Requirements", draft-ietf-rtcweb-use-cases-and-requirements-14 (work in progress), February 2014.

[I-D.lennox-mmusic-sdp-source-selection]
Lennox, J. and H. Schulzrinne, "Mechanisms for Media Source Selection in the Session Description Protocol (SDP)", draft-lennox-mmusic-sdp-source-selection-05 (work in progress), October 2012.

[I-D.westerlund-avtcore-rtp-simulcast]
Westerlund, M. and S. Nandakumar, "Using Simulcast in RTP Sessions", draft-westerlund-avtcore-rtp-simulcast-03 (work in progress), October 2013.

[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.

[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.

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

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[RFC5049] Bormann, C., Liu, Z., Price, R., and G. Camarillo, "Applying Signaling Compression (SigComp) to the Session Initiation Protocol (SIP)", RFC 5049, December 2007.

[RFC5225] Pelletier, G. and K. Sandlund, "RObust Header Compression Version 2 (ROHCv2): Profiles for RTP, UDP, IP, ESP and UDP-Lite", RFC 5225, April 2008.

[RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.

[RFC5626] Jennings, C., Mahy, R., and F. Audet, "Managing Client-Initiated Connections in the Session Initiation Protocol (SIP)", RFC 5626, October 2009.

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

[TS25.308]
3GPP, "High Speed Downlink Packet Access (HSDPA); Overall description; Stage 2", 3GPP TS 25.308 10.6.0, December 2011, <http://www.3gpp.org/ftp/Specs/html-info/25308.htm>.

[TS26.114]

3GPP, "IP Multimedia Subsystem (IMS); Multimedia telephony; Media handling and interaction", 3GPP TS 26.114 10.7.0, June 2013, <http://www.3gpp.org/ftp/Specs/html-info/26114.htm>.

[TS36.201] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA); LTE physical layer; General description", 3GPP TS 36.201 10.0.0, December 2010, <http://www.3gpp.org/ftp/Specs/html-info/36201.htm>.

Appendix A. Changes From Earlier Versions

NOTE TO RFC EDITOR: Please remove this section prior to publication.

A.1. Modifications Between Version -00 and -01

- Corrected text in section 6.5 and 6.2 to indicate that a PAUSE signaled via TMMBR 0 cannot be REFUSEd using TMMBN > 0
- Improved alignment with RTP Taxonomy draft, including the change of Packet Stream to RTP Stream
- Editorial improvements

Authors' Addresses

Bo Burman
Ericsson
Kistavagen 25
SE - 164 80 Kista
Sweden

Phone: +46107141311
Email: bo.burman@ericsson.com
URI: www.ericsson.com

Azam Akram
Ericsson

Farogatan 6
SE - 164 80 Kista
Sweden

Phone: +46107142658
Email: muhammad.azam.akram@ericsson.com
URI: www.ericsson.com

Roni Even
Huawei Technologies
Tel Aviv
Israel

Email: roni.even@mail01.huawei.com

Magnus Westerlund
Ericsson
Farogatan 6
SE- Kista 164 80
Sweden

Phone: +46107148287
Email: magnus.westerlund@ericsson.com