Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 16, 2012

A. Akram
B. Burman
D. Grondal
M. Westerlund
Ericsson AB
May 15, 2012

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

Abstract

With the increased popularity of real-time multimedia applications, users 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 adding a group of new real-time feedback messages used to pause and resume RTP data streams.

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.

Video conferencing MAY involve many participants; each has its own preferences and demands control over the communication session not only from the start but also during the session. This document describes several scenarios in multimedia communication where a participant chooses to temporarily pause incoming data from specific sources(s) and resuming it when needed. The receiver does not need to terminate the session from the source(s) 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, which either uses logic based on voice activity, other measurements, user input over properietary interfaces, or Media Stream Selection [I-D.westerlund-dispatch-stream-selection] could reduce the resources consumed in both the media sender and the network by temporarily pausing the media streams that aren't required by the RTP Mixer. This becomes especially useful when the media sources are provided in multiple encoding versions (Simulcast) [I-D.westerlund-avtcore-rtp-simulcast] or with 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 the media streams required at any given point is highly dynamic, using the out-of-band signalling channel for pausing and even more importantly resuming a media stream is difficult due to the performance requirements. Instead, the pause and resume signalling 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 approriate. No currently existing RTCP feedback message supports pausing and resuming an incoming data stream. As this is 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 should thus be defined as a Codec Control Message (CCM) extension.

2. Definition

2.1. Abbreviations

RTP Real-time Transport Protocol

RTCP Real-time Transport Control Protocol

SSRC Synchronization Source

CSRC Contributing Source

FB Feedback

AVPF Audio-Visual Profile with Feedback

FMT Feedback Message Type

PT Payload Type

CCM Codec Control Messages

MCU Multipoint Control Unit

2.2. Terminology

In addition to following, the definitions from RTP [RFC3550], AVPF [RFC4585] and CCM [RFC5104] also apply in this document.

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

Acknowledgement: The confirmation from receiver to sender that the message has been received.

Sender: The RTP entity that sends an RTP data stream.

Receiver: The RTP entity that receives an RTP data stream.

Mixer: The intermediate RTP node which receives a data stream from different nodes, combines them to make one stream and forwards to destinations, in the sense described in Topo-Mixer of RTP Topologies [RFC5117].

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

Paused Sender: An RTP sender which receives a PAUSE request, defined in this memo, from all other members in a communication session and stops its transmission, i.e. no other participant receives its RTP transmission at any given time.

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

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 media 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 has one or more SSRCs.

```
+---+         +---+
| A |<------->| B |
+---+         +---+
```

Point to Point

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

3.2. RTP Mixer to Media Sender

One of the most commonly used topologies in centralized conferencing is based on the RTP Mixer. 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 having its' own SSRCs and any media 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 media stream the Mixer provides can be an actual media mixing of multiple media sources, but it might also be as simple as selecting one of the underlying sources based on some Mixer policy or control signalling.

```
                 +---+      +------------+      +---+
                 | A |<---->|            |<---->| B |
                 +---+      |            |      +---+
                            |   Mixer    |
                 +---+      |            |      +---+
                 | C |<---->|            |<---->| D |
                 +---+      +------------+      +---+
```

Figure 1: RTP Mixer

The media streams being 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 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 listners may be prevented from breaking into the session unless approved by the moderator. The media selection could also be under the user's control using a protocol like Media Stream Selection [I-D.westerlund-dispatch-stream-selection].The media streams may also be simulcasted or scalably encoded, thus providing multiple versions that can be delivered by the media sender. These examples indicate that there are numerous reasons why a particular media 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 media 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 media stream from being delivered to it. It also needs to be able to resume delivery with minimal delay.

3.3. Media Receiver to RTP mixer

An end-point like A in Figure 1 could potentially request to pause the delivery of a given media 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 media stream. However, the authors are of the opinion that this is not a suitable solution.

First of all, the Mixer might not include CSRC in it's stream indications. Secondly, an end-point cannot rely on the CSRC to

correctly identify the media stream be paused when the delivered media is some type of mix. A media stream identification solution is needed to support this.

In addition, pause is only part of the semantics when it comes to selecting media streams. As can be seen in MESS [I-D.westerlund-dispatch-stream-selection], it can be beneficial to have both include and exclude semantics. In addition, substitution and possibility to control in what local media stream the selected media stream is to be provided gives richer functionality.

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

4. Design Considerations

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

4.1. Real-time Nature

The first section (Section 1) of this memo 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.

4.2. Message Direction

It is the responsibility of a receiver, who wants to pause or resume a stream from the sender(s), to transmit PAUSE and RESUME messages. A sender who likes to pause itself, can simply do it.

4.3. Apply to Individual Sources

The PAUSE and RESUME messages apply to single media 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.

4.4. Consensus

A sender must not pause an SSRC until all receivers that the sender knows of have requested it to be paused. The reason is that in RTP topologies where the media stream is shared between multiple

receivers, a single receiver on that shared network, independent of it being multicast or a transport Translator based, must not cause the media stream to be paused without the consent of all other receivers. A consequence of this is that a newly joining receiver needs to cause the sender to resume a paused stream. Any receiver wanting to resume a stream must also cause it to be resumed.

4.5. Acknowledgements

RTP 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 media sender. In some cases when a PAUSE or RESUME message reaches the media sender, it will not be able to pause the stream, instead the sender awaits requests from other receivers as well to fulfill the consensus requirement. In that case an RTP receiver MAY assume that previous PAUSE or RESUME message was lost and falsely retransmit it. In order to avoid this condition, the media sender target of a PAUSE or RESUME request needs to send an acknowledgement in response to each PAUSE and RESUME message.

4.6. Retransmitting Requests

As PAUSE or RESUME requests as well as Acknowledgments can be lost, the sender of a request will need to retransmit it in case no acknowlegement is received. 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, an acknowledgement has been received, or the media stream gets delivered.

4.7. Sequence Numbering

Every PAUSE and RESUME request message will need to have a sequence number to separate retransmissions from new requests. The sequence number is incremented by one every time a new request is transmitted. The PAUSE and RESUME message should share the same sequence number space. The advantage of using same sequence number space is to avoid the ambiguity which message to the request receiver should follow in case of retransmissions. For example, if an RTP sender receives both PAUSE and RESUME messages before deciding which message to respond to

(may be due to late packet arrival or any other reason), it can follow the message with higher sequence number.

Each acknowledgement will have the same sequence number as in the message (PAUSE or RESUME) it is responding to.

5. Solution Overview

The PAUSE and RESUME functionality is based on sending RTCP feedback messages from any RTP session participant that wants to pause or resume a media stream targeted at the media stream sender, as identified by the sender SSRC. A single Feedback message specification is used. The message consists of a number of Feedback Control Information (FCI) blocks, where each block can be a PAUSE request, a RESUME request or one of four different kinds of acknwoledgements. This structure allows a single feedback message to request pause or resume on a number of media streams.

To ensure reliability of the established state at the targeted media senders, acknowlegments are used. However, due to the requirement to not pause until all RTP session receivers, i.e. the ones that send RTCP Receiver Reports on the media sender's stream, are ok with it, most acknowlegements will NACK. This NACK says the session participant has established state for the media receiver that it desires a paused state, but it couldn't comply due to other session participants not having requested to pause the stream.

The transmission of any RTCP feedback messages follows the regular AVPF defined timing rules and depends on the session's mode of operation.

6. Participants States

This document introduces a new state the media stream in an RTP sender can have, a paused state.

6.1. Paused State

A media stream is in paused state when the sender pauses its transmission after receiving PAUSE requests from all other receiving participants in the session, which means no participant is willing to receive it's transmission. This requires the media stream sender to track all RTP session participants to determine that all have requested a pause state with the sender.

Following sub-sections discusses some potential issues when an RTP

sender goes into paused state.

6.1.1. RTCP BYE Message

When a participant leaves the communication 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 all receivers in a session, the sender keeps record of all the receivers which do and which do not want to receive its transmission. If a pausing receiver sends an RTCP BYE message observed by the sender, the sender SHALL NOT consider that receiver when it decides to pause its transmission.

These conditions are also valid if an RTP Translator is used in the communication. When an RTP Mixer implementing this memo 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 above conditions also apply to the sender and receiver parts of the RTP Mixer, respectively.

6.1.2. 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 other participant(s) for last five RTCP reporting intervals it removes that participant from the receiver list.

7. 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 acknowledgements 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 is defined by AVPF [RFC4585] is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|   FMT   |       PT      |          length               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  SSRC of packet sender                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  SSRC of media source                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:            Feedback Control Information (FCI)                 :
:                                                               :
```

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. he 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 request(s) or acknowledgments in the FCI.

SSRC of media source: Not used, SHALL be set to 0. The FCI identifies the SSRC the request is for or whose request the acknowlegement are on.

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

```
0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Target SSRC                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Sequence Number          | Type  |       Reserved        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Syntax of FCI Entry in the PAUSE and RESUME message

The FCI fields have the following definitions:

Target SSRC (32 bits): For a Request message, the "Target SSRC" value is the SSRC that this request is intended for. For any type of Acknowlegement type defined in this document, the SSRC is the SSRC who sent the request being acknowledged. A CSRC MUST NOT be used as a target as the interpretation of such a request is unclear.

Sequence Number (16 bits): Sequence number of the request that SHALL be incremented by one for each new request. Both PAUSE and RESUME messages SHALL share the same sequence number space. Each requesting SSRC has its own sequence number space with each target SSRC. In other words, A requesting B to PAUSE or RESUME has a different sequence number space than A and C. Also, B requesting of A to PAUSE or RESUME will have a different sequence number space.

Type (4 bits): The pause feedback type, i.e. either PAUSE or RESUME or their acknowledgements. The values are as follows,

0: PAUSE message

1: RESUME message

2: Pause-Acknowledgement (PACK)

3: Resume-Acknowledgement (RACK)

4: Negative-Acknowledgement (NACK)

5: REFUSE

6-15: Reserved for future use

Reserved: (12 bits): SHALL be ignored by receivers implementing this memo and MUST be set to 0 by senders implementing this memo.

7.1. Message Acknowledgements

To let the sender of PAUSE and RESUME requests verify the reception and the target's reaction to the request, the target of a PAUSE or RESUME request SHALL send an acknowledgment for each request received. All transmissions of request and acknowlegement are governed by the transmission rules as defined by Section 7.2. A request sender that hasn't received any acknowledgement after one Round-Trip Time (RTT) MAY retransmit the request again.

After having received an acknowledgement on a request, a receiver SHOULD avoid sending further requests of the same type to the same sender to avoid unnecessary bandwidth consumption. However, a receiver MAY repeat a request of the same type, e.g. if it is for some reason necessary to re-confirm the sender's opinion of the receiver's request status. Consequently, a sender SHALL respond with corresponding acknowledgement to all requests, even if the request seems unnecessary and does not cause the sender to change state.

Every acknowledgement SHALL have the same sequence number as the request message (PAUSE or RESUME) it acknowledges. The sender can respond to PAUSE or RESUME requests in four different ways.

7.1.1. Negative-Acknowledgement (NACK)

In order for the sender to pause its transmission, it MUST receive PAUSE request from all the receivers in a session. Consider there are N receiving participants in a session. When a sender receives a PAUSE request, it MUST check if it has received requests from N-1 participants. If the number of requesting participants are less than N-1 it replies with NACK, which is the indication to the requester that though the request has been received, the transmission can not be paused at this stage because there are still some receiver(s) in the session that want to receive it. If a pausing receiver is no longer interested in pausing the SSRC, it MAY send an RESUME request to the sender from which it has previously received a NACK. The sender shall then reply with RACK to that receiver Section 7.1.3.

The NACK MUST only be sent in response to a PAUSE request. The NACK MUST have the same sequence number as in the PAUSE request.

7.1.2. Pause-Acknowledgement (PACK)

When an RTP sender receives a PAUSE request from all the receivers in a session, it sends a Pause-Acknowledgement (PACK) to the receivers and enters into Paused state as discussed in Section 6.1. It means that if there are N participants in a session and the sender receives PAUSE request(s) from N-1th participant, it pauses its transmission and sends a PACK to all the PAUSE requesters. The other participants can detect that the media sender is paused based on it sending a PACK.

The PACK MUST only be sent in response to a PAUSE request. The PACK MUST contain the same sequence number as in the PAUSE request.

7.1.3. Resume-Acknowledgement (RACK)

When an RTP sender receives a RESUME request from any of the receivers in a session, it replies with Resume-Acknowledgement (RACK) and resumes its transmission, if it is in Paused state (discussed in Section 6.1).

The RACK MUST only be sent in response to a RESUME request. The RACK MUST match the sequence number in RESUME request.

7.1.4. REFUSE

If any PAUSE and/or RESUME request can not be fulfilled by the sender due to some reason, it replies with REFUSE acknowledgement.

The REFUSE MAY be sent in response to PAUSE or RESUME requests. The REFUSE MUST contain the same sequence number as in the PAUSE/RESUME request.

7.2. Transmission Rules

To be Written

8. Examples

Following are the use cases when there MAY be a need to use PAUSE and RESUME messages,

1. Point-to-Point session

2. Point-to-multipoint using Mixer

3. Point-to-multipoint using Translator

8.1. 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 message to the sender, which pauses or resumes transmission accordingly.

```
   +---------------+                   +---------------+
   |  RTP Sender   |                   | RTP Receiver  |
   +---------------+                   +---------------+
                      t1: RTP data
          | -------------------------------> |
          |           t2: PAUSE              |
          | <------------------------------- |
          |                                  |
          |       < RTP data paused >        |
          |           t3: PACK               |
          | -------------------------------> |
          |                                  |
          |           t4: RESUME             |
          | <------------------------------- |
          |           t5: RACK               |
          | -------------------------------> |
          |           t6: RTP data           |
          | -------------------------------> |
```

Figure 3: The pause and resume operation in Point-to-Point scenario

Figure 3 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. The sender pauses the data and replies with a Pause-Acknowledgement (PACK). Some time later (at time t4) the receiver requests the sender to resume, which resumes its transmission and replies with Resume-Acknowledgement (RACK).

```
+---------------+                       +---------------+
|  RTP Sender   |                       | RTP Receiver  |
+---------------+                       +---------------+
                 t1: RTP data
       | ------------------------------------> |
       |                 t2: PAUSE, lost       |
       |                 <---X---------------- |
       |                                       |
       |           t3: RTP data                |
       | ------------------------------------> |
       |                                       |
       |    <Timeout, still receiving data>    |
       |           t4: PAUSE                   |
       | <------------------------------------ |
       |           < RTP data paused >         |
       |           t5: PACK                    |
       | ------------------------------------> |
       |                                       |
       |           t6: RESUME                  |
       | <------------------------------------ |
       |           t7: RACK                    |
       | ------------------------------------> |
       |           t8: RTP data                |
       | ------------------------------------> |
```

Figure 4: The pause and resume operation with PAUSE lost

Figure 4 describes what happens if a PAUSE message from an RTP receiver does not reach the RTP sender. After sending a PAUSE message, the receiver waits for a time-out to detect if the sender has paused the data transmission or has sent any acknowledgement according to the rules discussed in Section 7.1. As the PAUSE message is lost on the way (at time t2), RTP data continues to reach to the receiver. When the timer expires, receiver schedules retransmit of the PAUSE message. If PAUSE message reaches to the RTP sender, it stops streaming and replies with PACK. The same rules apply to the RESUME message, i.e., the RTP receiver waits for a time-out value after sending the RESUME message until it gets the transmission or receives any acknowledgement.

```
   +---------------+                 +---------------+
   |  RTP Sender   |                 | RTP Receiver  |
   +---------------+                 +---------------+
          |           t1: RTP data          |
          | ------------------------------> |
          |           t2: PAUSE             |
          | <------------------------------ |
          |                                 |
          |  < Can not pause RTP data >     |
          |           t3: REFUSE            |
          | ------------------------------> |
          |                                 |
          |           t4: RTP data          |
          | ------------------------------> |
```

Figure 5: The pause request is refused in Point-to-Point scenario

In Figure 5, the receiver requests to pause the sender, which refuses to pause due to session policy and responds with REFUSE message.

8.2. Point-to-multipoint using Mixer

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

The Mixer keeps track of all the media streams delivered to the Mixer and how they currently are used. It selects the video stream to deliver to the receiver R based on the voice activity of the media senders. The video stream will be delivered to R using M's SSRC and with an CSRC indicating the orignal source.

```
   +-----+            +-----+            +-----+            +-----+
   |  R  |            |  M  |            | S1  |            | S2  |
   +-----+            +-----|            +-----+            +-----+
      |                  |   t1:RTP(S1)     |                  |
      | t2:RTP(M:S1)     |<-----------------|                  |
      |<-----------------|                  |                  |
      |                  | t3:RTP(S2)       |                  |
      |                  |<------------------------------------|
      |                  |  t4: PAUSE(S2)   |                  |
      |                  |------------------------------------>|
      |                  |                  |     t5: PACK(S2) |
      |                  |<------------------------------------|
      |                  |                  | <S2:No RTP to M> |
      |                  | t6: RESUME(S2)   |                  |
      |                  |------------------------------------->
      |                  |                  | t7: RTP to M     |
      |                  |<------------------------------------|
      |                  |                  | t8: RACK(S2)     |
      |                  |<------------------------------------|
      |   t9:RTP(M:S2)   |                  |                  |
      |<-----------------|                  |                  |
      |                  | t10:PAUSE(S1)    |                  |
      |                  |----------------->|                  |
      |                  | t11:PACK(S1)     |                  |
      |                  |<-----------------|                  |
      |                  | <S1:No RTP to M> |                  |
```

Figure 6: The pause and resume operations for an 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 the CSRC indicated after the colon in the figure. Then S2 joins the session at t3 and starts delivering media 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 PACK 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 sends a RESUME request to S2. At t7, S2 has received the RESUME request and acts on it by resuming RTP media delivery to M. It also schedules the transmission of a RACK, which is sent at t8. When the media from t7 arrives at the Mixer, it switches this media into its SSRC (M) at t9 and changes the CSRC to S2. As S1 now becomes unused, the Mixer issues a PAUSE request to S1 at t10, which is acknowledged at t11 with a PACK and the RTP media stream from S1 stops being delivered.

8.3. 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 and change the SSRC of the message. These examples are to show that the message can be safely used also in a transport Translator case.

```
+-------------+     +-------------+     +--------------+
|  Sender(S)  |     | Translator  |     | Receiver(R)  |
+-------------+     +-------------|     +--------------+
       | t1: RTP(S)        |                   |
       |------------------>|                   |
       |                   | t2: RTP (S)       |
       |                   |------------------>|
       |                   | t3: PAUSE(S)      |
       |                   |<------------------|
       | t4:PAUSE (S)      |                   |
       |<------------------|                   |
       |            < RTP data paused >        |
       | t5: PACK (S)      |                   |
       |------------------>|                   |
       |                   | t6: PACK (S)      |
       |                   |------------------>|
       |                   |                   |
       |                   | t7: RESUME(S)     |
       |                   |<------------------|
       |t8: RESUME(S)      |                   |
       |<------------------|                   |
       | t9: RACK (S)      |                   |
       |------------------>|                   |
       |                   | t10: RACK (S)     |
       |                   |------------------>|
       | t11: RTP(S)       |                   |
       |------------------>|                   |
       |                   | t12: RTP (S)      |
       |                   |------------------>|
```

Figure 7: The pause and resume operation between two participants using the Translator

Figure 7 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 PAUSE requests to the sender, which checks that there is no other receiver which wants to receive the data, hence pauses itself and replies with PACK. Similarly the receiver resumes the sender by sending RESUME request through

Translator.

```
   +-----+            +-----+            +-----+            +-----+
   |  S  |            |  T  |            | R1  |            | R2  |
   +-----+            +-----|            +-----+            +-----+
      | t1:RTP(S)        |                  |                  |
      |----------------->|                  |                  |
      |                  | t2:RTP(S)        |                  |
      |                  |----------------->------------------>|
      |                  | t3:PAUSE(S)      |                  |
      |                  |<-----------------|                  |
      | t4:PAUSE(S)      |                  |                  |
      |<-----------------|                  |                  |
      | t5:NACK(S)       |                  |                  |
      |----------------->|                  |                  |
      |                  | t6:NACK(S)       |                  |
      |                  |----------------->|                  |
      |                  |<RTP stream continues to R1 and R2>  |
      |                  |                  |     t7: PAUSE(S) |
      |                  |<------------------------------------|
      | t8:PAUSE(S)      |                  |                  |
      |<-----------------|                  |                  |
      | < Pauses RTP data stream >          |                  |
      | t9:PACK(S)       |                  |                  |
      |----------------->|                  |                  |
      |                  | t10:PACK(S)      |                  |
      |                  |----------------->------------------>|
      |                  | t11:RESUME(S)    |                  |
      |                  |<-----------------|                  |
      | t12:RESUME(S)    |                  |                  |
      |<-----------------|                  |                  |
      | t13:RACK(S)      |                  |                  |
      |----------------->|                  |                  |
      |                  | t14:RACK(S)      |                  |
      |                  |----------------->|                  |
      | t15:RTP(S)       |                  |                  |
      |----------------->|                  |                  |
      |                  | t16:RTP(S)       |                  |
      |                  |----------------->------------------>|
```

Figure 8: The pause and resume operation between one sender and two receivers through translator

Figure 8 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 media 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 media stream.

After some time (at t3), R1 chooses to pause the stream. On receiving the PAUSE request from R1, S checks if there are any other receiver which still wants to receive the data. At this time, S knows that R2 exists and has not indicated that it wants to pause the stream. The sender S replies with NACK to R1 and continues to send data to T which forwards to both R1 and R2. At t7, the receiver R2 also selects to pause the data by sending a PAUSE request. Now the sender S knows that no receiver (neither R1 nor R2) want the stream, it concludes that the stream must be paused. S now stops sending the stream and replies with PACK to R1 and R2. When any of the receivers (R1 or R2) choses to resume the stream from S, it sends a RESUME request to the sender. In reply, the RTP sender sends a RACK to the requesting RTP receiver and resumes streaming.

Consider an RTP session which includes one or more receivers, paused sender(s), and a Translator. 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) resumes the transmission since there is now a receiver in the session that did not pause the sender. It SHALL depend on the new receiver to pause or continue that stream(s).

9. Signalling

The capability of handling PAUSE and RESUME messages 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], this document recommends to use the rtcp-fb attribute at media level and it must not be used at session level. This memo follows all the rules defined in AVPF for rtcp-fb attribute relating to payload type in a session description.

Section 7.1 of CCM [RFC5104] defines a new feedback value "ccm", which indicates the support of codec control using RTCP feedback. The CCM [RFC5104] defines four different parameters which SHOULD be used with the feedback value "ccm" to indicate the specific codec control command.

This memo defines a new parameter, "pause", which aggregatively represent the PAUSE, RESUME messages and their acknowledgements (i.e., PACK, NACK, RACK and REFUSE). An endpoint implementing this

memo and using SDP to signal capability MUST use the new "pause" extension to ccm signaling. Similarly, a sender or receiver SHOULD NOT use the messages from this memo towards receivers that did not declare capability for it.

The below figure is an example how to show support for pausing and resuming the stream according to this memo:

```
v=0
o=alice 3203093520 3203093520 IN IP4 host.example.com
s=Pausing Media
t=0 0
c=IN IP4 host.example.com
m=audio 49170 RTP/AVPF 98
a=rtpmap:98 H263-1998/90000
a=rtcp-fb:98 ccm pause
```

Figure 9: An SDP example with pause and resume capability

10. IANA Considerations

As outlined in Section 7, this memo requests IANA to allocate

1. The 'pause' tag to be used with ccm under rtcp-fb AVPF attribute in SDP.

2. The FMT number TBA1 to be allocated to the PAUSE and RESUME functionality from this memo.

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

4. PAUSE, RESUME, PACK, RACK, NACK, and REFUSE with the listed numbers in the pause Type registry.

11. 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 paused all the source stream which MAY result in Denial of Service (DoS). An Authentication protocol MAY save from this attack.

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

12. Acknowledgements

13. References

13.1. 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.

13.2. Informative References

[I-D.westerlund-avtcore-rtp-simulcast] Westerlund, M., Burman, B., Lindqvist, M., and F. Jansson, "Using Simulcast in RTP sessions", draft-westerlund-avtcore-rtp-simulcast-00 (work in progress), October 2011.

[I-D.westerlund-dispatch-stream-selection] Grondal, D., Burman, B., and M. Westerlund, "Media Stream Selection (MESS)", draft-westerlund-dispatch-stream-selection-00 (work in progress), October 2011.

[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.

[RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, January 2008.

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

Authors' Addresses

Azam Akram
Ericsson AB
Farogatan 6
SE - 164 80 Kista,
Sweden

Phone: +46107142658
Fax: +46107175550
Email: muhammad.azam.akram@ericsson.com
URI: www.ericsson.com

Bo Burman
Ericsson AB
Farogatan 6
SE - 164 80 Kista,
Sweden

Phone: +46107141311
Fax: +46107175550
Email: bo.burman@ericsson.com
URI: www.ericsson.com

Daniel Grondal
Ericsson AB
Farogatan 6
SE - 164 80 Kista,
Sweden

Phone: +46107147505
Fax: +46107175550
Email: daniel.grondal@ericsson.com
URI: www.ericsson.com

Magnus Westerlund
Ericsson AB
Farogatan 6
SE- Kista 164 80,
Sweden

Phone: +46107148287
Fax:
Email: magnus.westerlund@ericsson.com
URI: www.ericsson.com