

Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 4, 2013

B. Burman
M. Westerlund
Ericsson
January 31, 2013

Multi-Media Concepts and Relations
draft-burman-rtcweb-mmusic-media-structure-00

Abstract

There are currently significant efforts ongoing in IETF regarding more advanced multi-media functionalities, such as the work related to RTCWEB and CLUE. This work includes use cases for both multi-party communication and multiple media streams from an individual end-point. The usage of scalable encoding or simulcast encoding as well as different types of transport mechanisms have created additional needs to correctly identify different types of resources and describe their relations to achieve intended functionalities.

The different usages have both commonalities and differences in needs and behavior. This document attempts to review some usages and identify commonalities and needs. It then continues to highlight important aspects that need to be considered in the definition of these usages.

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 August 4, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Motivation	3
3. Use Cases	4
3.1. Existing RTP Usages	4
3.1.1. Basic VoIP call	4
3.1.2. Audio and Video Conference	5
3.1.3. Audio and Video Switched Conference	7
3.2. WebRTC	8
3.2.1. Mesh-based Multi-party	9
3.2.2. Multi-source Endpoints	10
3.2.3. Media Relaying	11
3.2.4. Usage of Simulcast	11
3.3. CLUE Telepresence	13
3.3.1. Telepresence Functionality	13
3.3.2. Distributed Endpoint	14
4. Discussion	14
4.1. Commonalities in Use Cases	14
4.1.1. Media Source	14
4.1.2. Encodings	16
4.1.3. Synchronization contexts	17
4.1.4. Distributed Endpoints	18
4.2. Identified WebRTC issues	18
4.3. Relevant to SDP evolution	19
5. IANA Considerations	20
6. Security Considerations	21
7. Informative References	21
Authors' Addresses	22

1. Introduction

This document concerns itself with the conceptual structures that can be found in different logical levels of a multi-media communication, from transport aspects to high-level needs of the communication application. The intention is to provide considerations and guidance that can be used when discussing how to resolve issues in the RTCWEB and CLUE related standardization. Typical use cases for those WG have commonalities that likely should be addressed similarly and in a way that allows to align them.

The document starts with going deeper in the motivation why this has become an important problem at this time. This is followed by studies of some use cases and what concepts they contain, and concludes with a discussion of observed commonalities and important aspects to consider.

2. Motivation

There has arisen a number of new needs and requirements lately from work such as WebRTC/RTCWEB [I-D.ietf-rtcweb-overview] and CLUE [I-D.ietf-clue-framework]. The applications considered in those WG has surfaced new requirements on the usage of both RTP [RFC3550] and existing signalling solutions.

The main application aspects that have created new needs are:

- o Multiple Media Streams from an end-point. The fact that an end-point may have multiple media capture devices, such as cameras or microphone mixes.
- o Group communications involving multiple end-points. This is realized using both mesh based connections as well as centralized conference nodes. These creating a need for dealing with multiple endpoints and/or multiple streams with different origins from a transport peer.
- o Media Stream Adaptation, both to adjust network resource consumption as well as to handle varying end-point capabilities in group communication.
- o Transport mechanisms including both higher levels of aggregation [I-D.ietf-mmusic-sdp-bundle-negotiation] [I-D.ietf-avtcore-multi-media-rtp-session] and the use of application-level transport repair mechanisms such as forward error correction (FEC) and/or retransmission.

The presence of multiple media resources or components creates a need to identify, handle and group those resources across multiple different instantiations or alternatives.

3. Use Cases

3.1. Existing RTP Usages

There are many different existing RTP usages. This section brings up some that we deem interesting in comparison to the other use cases.

3.1.1. Basic VoIP call

This use case is intended to function as a base-line to contrast against the rest of the use cases.

The communication context is an audio-only bi-directional communication between two users, Alice and Bob. This communication uses a single multi-media session that can be established in a number of ways, but let's assume SIP/SDP [RFC3261][RFC3264]. This multi-media session contains two end-points, one for Alice and one for Bob. Each end-point has an audio capture device that is used to create a single audio media source at each end-point.

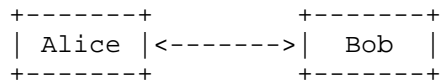


Figure 1: Point-to-point Audio

The session establishment (SIP/SDP) negotiates the intent to communicate over RTP using only the audio media type. Inherent in the application is an assumption of only a single media source in each direction. The boundaries for the encodings are represented using RTP Payload types in conjunction with the SDP bandwidth parameter (b=). The session establishment is also used to negotiate that RTP will be used, thus resulting in that an RTP session will be created for the audio. The underlying transport flows, in this case a single bi-directional UDP flow for RTP, another for RTCP, is configured by each end-point providing its' IP address and port, which becomes source or destination depending on in which direction the packet is sent.

The RTP session will have two RTP media streams, one in each direction, which carries the encoding of the media source the sending implementation has chosen based on the boundaries established by the RTP payload types and other SDP parameters, e.g. codec, and bit-

rates. The streams are in the RTP context identified by their SSRCs.

3.1.2. Audio and Video Conference

This use case is a multi-party use case with a central conference node performing media mixing. It also includes two media types, both audio and video. The high level topology of the communication session is the following:

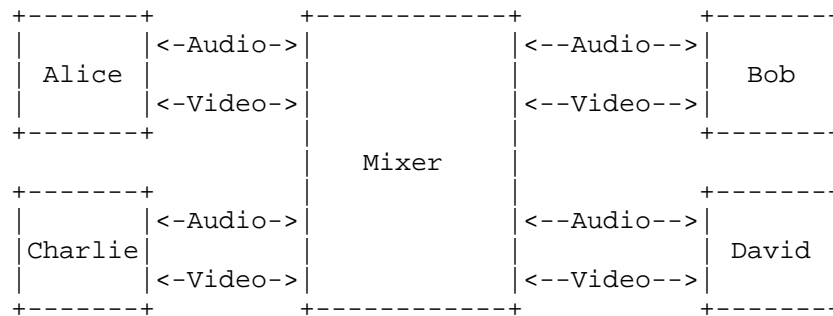


Figure 2: Audio and Video Conference with Centralized Mixing

The communication session is a multi-party conference including the four users Alice, Bob, Charlie, and David. This communication session contains four end-points and one middlebox (the Mixer). The communication session is established using four different multi-media sessions; one each between the user's endpoints and the middlebox. Each of these multi-media sessions uses a session establishment method, like SIP/SDP.

Looking at a single multi-media session between a user, e.g. Alice, and the Mixer, there exist two media types, audio and video. Alice has two capture devices, one video camera giving her a video media source, and an audio capture device giving an audio media source. These two media sources are captured in the same room by the same end-point and thus have a strong timing relationship, requiring inter-media synchronization at playback to provide the correct fidelity. Thus Alice's endpoint has a synchronization context that both her media sources use.

These two media sources are encoded using encoding parameters within the boundaries that has been agreed between the end-point and the Mixer using the session establishment. As has been common practice, each media type will use its own RTP session between the end-point and the mixer. Thus a single audio stream using a single SSRC will flow from Alice to the Mixer in the Audio RTP session and a single video stream will flow in the Video RTP session. Using this division

in separate RTP sessions, the bandwidth of both audio and video can be unambiguously and separately negotiated by the SDP bandwidth attributes exchanged between the end-points and the mixer. Each RTP session is using its own Transport Flows. The common synchronization context across Alice's two media streams is identified by binding both streams to the same CNAME, generated by Alice's endpoint.

The mixer does not have any physical capture devices, instead it creates conceptual media sources. It provides two media sources towards Alice; one audio being a mix of the audio from Bob, Charlie and David, the second one being a conceptual video source that contains a selection of one of the other video sources received from Bob, Charlie, or David depending on who is speaking. The Mixer's audio and video sources are provided in an encoding using a codec that is supported by both Alice's endpoint and the mixer. These streams are identified by a single SSRC in the respective RTP session.

The mixer will have its own synchronization context and it will inject the media from Bob, Charlie and David in a synchronized way into the mixer's synchronization context to maintain the inter-media synchronization of the original media sources.

The mixer establishes independent multimedia sessions with each of the participant's endpoints. The mixer will in most cases also have unique conceptual media sources for each of the endpoints. This as audio mixes and video selections typically exclude media sources originating from the receiving end-point. For example, Bob's audio mix will be a mix of Alice, Charlie and David, and will not contain Bob's own audio.

This use case may need unique user identities across the whole communication session. An example functionality of this is a participant list which includes audio energy levels showing who is speaking within the audio mix. If that information is carried in RTP using the RTP header extension for Mixer to audio clients [RFC6465] then contributing source identities in the form of CSRC need to be bound to the other end-point's media sources or user identities. This despite the fact that each RTP session towards a particular user's endpoint is terminated in the RTP mixer. This points out the need for identifiers that exist in multiple multi-media session contexts. In most cases this can easily be solved by the application having identities tailored specifically for its own needs, but some applications will benefit from having access to some commonly defined structure for media source identities.

3.1.3. Audio and Video Switched Conference

This use case is similar to the one above (Section 3.1.2), with the difference that the mixer does not mix media streams by decoding, mixing and re-encoding them, but rather switches a selection of received media more or less unmodified towards receiving end-points. This difference may not be very apparent to the end-user, but the main motivations to eliminate the mixing operation and switch rather than mix are:

- o Lower processing requirements in the mixer.
- o Lower complexity in the mixer.
- o Higher media quality at the receiver given a certain media bitrate.
- o Lower end-to-end media delay.

Without the mixing operation, the mixer has limited ability to create conceptual media sources that are customized for each receiver. The reasons for such customizations comes from sender and receiver differences in available resources and preferences:

- o Presenting multiple conference users simultaneously, like in a video mosaic.
- o Alignment of sent media quality to receivers presentation needs.
- o Alignment of codec type and configuration between sender and receiver.
- o Alignment of encoded bitrate to the available end-to-end link bandwidth.

To enable elimination of the mixing operation, media sent to the mixer must sufficiently well meet the above constraints for all intended receivers. There are several ways to achieve this. One way is to, by some system-wide design, ensure that all senders and receivers are basically identical in all the above aspects. This may however prove unrealistic when variations in conditions and end-points are too large. Another way is to let a sender provide a (small) set of alternative representations for each sent media source, enough to sufficiently well cover the expected range of variation. If those media source representations, encodings, are independent from one another, they constitute a Simulcast of the media source. If an encoding is instead dependent on and thus requires reception of one or more other encodings, the representation

of the media source jointly achieved by all dependent encodings is said to be Scalable. Simulcast and Scalable encoding can also be combined.

Both Simulcast and Scalable encodings result in that a single media source generates multiple RTP media streams of the same media type. The division of bandwidth between the Simulcast or Scalable streams for a single media source is application specific and will vary. The total bandwidth for a Simulcast or a Scalable source is the sum of all included RTP media streams. Since all streams in a Simulcast or Scalable source originate from the same capture device, they are closely related and should thus share synchronization context.

The first and second customizations listed above, presenting multiple conference users simultaneously, aligned with the presentation needs in the receiver, can also be achieved without mixing operation by simply sending appropriate quality media from those users individually to each receiver. The total bandwidth of this user presentation aggregate is the sum of all included RTP media streams. Audio and video from a single user share synchronization context and can be synchronized. Streams that originate from different users do not have the same synchronization context, which is acceptable since they do not need to be synchronized, but just presented jointly.

An actual mixer device need not be either mixing-only or switching-only, but may implement both mixing and switching and may also choose dynamically what to do for a specific media and a specific receiving user on a case-by-case basis or based on some policy.

3.2. WebRTC

This section brings up two different instantiations of WebRTC [ref-webrtc10] that stresses different aspects. But let's start with reviewing some important aspects of WebRTC and the MediaStream [ref-media-capture] API.

In WebRTC, an application gets access to a media source by calling `getUserMedia()`, which creates a `MediaStream` [ref-media-capture] (note the capitalization). A `MediaStream` consists of zero or more `MediaStreamTracks`, where each `MediaStreamTrack` is associated with a media source. These locally generated `MediaStreams` and their tracks are connected to local media sources, which can be media devices such as video cameras or microphones, but can also be files.

An WebRTC `PeerConnection` (PC) is an association between two endpoints that is capable of communicating media from one end to the other. The PC concept includes establishment procedures, including media negotiation. Thus a PC is an instantiation of a Multimedia Session.

When one end-point adds a MediaStream to a PC, the other endpoint will by default receive an encoded representation of the MediaStream and the active MediaStreamTracks.

3.2.1. Mesh-based Multi-party

This is a use case of WebRTC which establishes a multi-party communication session by establishing an individual PC with each participant in the communication session.

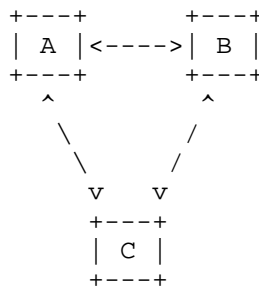


Figure 3: WebRTC Mesh-based Multi-party

Users A, B and C want to have a joint communication session. This communication session is created using a Web-application without any central conference functionality. Instead, it uses a mesh of PeerConnections to connect each participant's endpoint with the other endpoints. In this example, three double-ended connections are required to connect the three participants, and each endpoint has two PCs.

This is an audio and video communication and each end-point has one video camera and one microphone as media sources. Each endpoint creates its own MediaStream with one video MediaStreamTrack and one audio MediaStreamTrack. The endpoints add their MediaStream to both of their PCs.

Let's now focus on a single PC; in this case the one established between A and B. During the establishment of this PC, the two endpoints agree to use only a single transport flow for all media types, thus a single RTP session is created between A and B. A's MediaStream has one audio media source that is encoded according to the boundaries established by the PeerConnection establishment signalling, which includes the RTP payload types and thus Codecs supported as well as bit-rate boundaries. The encoding of A's media source is then sent in an RTP stream identified by a unique SSRC. In this case, as there are two media sources at A, two encodings will be created which will be transmitted using two different RTP streams

with their respective SSRC. Both these streams will reference the same synchronization context through a common CNAME identifier used by A. B will have the same configuration, thus resulting in at least four SSRC being used in the RTP session part of the A-B PC.

Depending on the configuration of the two PCs that A has, i.e. the A-B and the A-C ones, A could potentially reuse the encoding of a media source in both contexts, under certain conditions. First, a common codec and configuration needs to exist and the boundaries for these configurations must allow a common work point. In addition, the required bandwidth capacity needs to be available over the paths used by the different PCs. Both of those conditions are not always true. Thus it is quite likely that the endpoint will sometimes instead be required to produce two different encodings of the same media source.

If an application needs to reference the media from a particular endpoint, it can use the `MediaStream` and `MediaStreamTrack` as they point back to the media sources at a particular endpoint. This as the `MediaStream` has a scope that is not `PeerConnection` specific.

The programmer can however implement this differently while supporting the same use case. In this case the programmer creates two `MediaStreams` that each have `MediaStreamTracks` that share common media sources. This can be done either by calling `getUserMedia()` twice, or by cloning the `MediaStream` obtained by the only `getUserMedia()` call. In this example the result is two `MediaStreams` that are connected to different PCs. From an identity perspective, the two `MediaStreams` are different but share common media sources. This fact is currently not made explicit in the API.

3.2.2. Multi-source Endpoints

This section concerns itself with endpoints that have more than one media source for a particular media type. A straightforward example would be a laptop with a built in video camera used to capture the user and a second video camera, for example attached by USB, that is used to capture something else the user wants to show. Both these cameras are typically present in the same sound field, so it will be common to have only a single audio media source.

A possible way of representing this is to have two `MediaStreams`, one with the built in camera and the audio, and a second one with the USB camera and the audio. Each `MediaStream` is intended to be played with audio and video synchronized, but the user (local or remote) or application is likely to switch between the two captures.

It becomes important for a receiving endpoint that it can determine

that the audio in the two MediaStreams have the same synchronization context. Otherwise a receiver may playback the same media source twice, with some time overlap, at a switch between playing the two MediaStreams. Being able to determine that they are the same media source further allow for removing redundancy by having a single encoding if appropriate for both MediaStreamTracks.

3.2.3. Media Relaying

WebRTC endpoints can relay a received MediaStream from one PC to another by the simple API level maneuver of adding the received MediaStream to the other PC. To realize this in the implementation is more complex. This can also cause some issues from a media perspective. If an application spanning across multiple endpoints that relay media between each other makes a mistake, a media loop can be created. Media Loops could become a significant issue. For example could an audio echo be created, i.e. an endpoint receives its own media without detecting that it is its own media and plays it back with some delay. In case a WebRTC endpoint produces a conceptual media source by mixing incoming MediaStreams, if there is no loop detection, a feedback loop can be created.

RTP has loop detection to detect and handle such cases within a single RTP session. However, in the context of WebRTC, the RTP session is local to the PC and thus cannot rely on the RTP level loop detection. Instead, if this protection is needed on the WebRTC MediaStream level, it could for example be achieved by having media source identifiers that can be preserved between the different MediaStreams in the PCs.

When relaying media and in case one receives multiple encodings of the same source it is beneficial to know that. For example, if one encoding arrives with a delay of 80 ms and another with 450 ms, being able to choose the one with 80 ms and not be forced to delay all media sources from the same synchronization context to the most delayed source improves performance.

3.2.4. Usage of Simulcast

In this section we look at a use case applying simulcast from each user's endpoint to a central conference node to avoid the need for an individual encoding to each receiving endpoint. Instead, the central node chooses which of the available encodings that is forwarded to a particular receiver, like in Section 3.1.3.

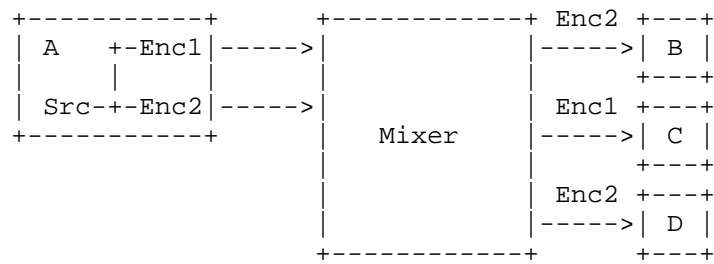


Figure 4

In this Communication Session there are four users with endpoints and one middlebox (The Mixer). This is an audio and video communication session. The audio source is not simulcasted and the endpoint only needs to produce a single encoding. For the video source, each endpoint will produce multiple encodings (Enc1 and Enc2 in Figure 4) and transfer them simultaneously to the mixer. The mixer picks the most appropriate encoding for the path from the mixer to each receiving client.

Currently there exists no specified way in WebRTC to realise the above, although use-cases and requirements discuss simulcast functionality. The authors believe there exist two possible solution alternatives in the WebRTC context:

Multiple Encodings within a PeerConnection: The endpoint that wants to provide a simulcast creates one or more MediaStreams with the media sources it wants to transmit over a particular PC. The WebRTC API provides functionality to enable multiple encodings to be produced for a particular MediaStreamTrack and have possibility to configure the desired quality levels and/or differences for each of the encodings.

Using Multiple PeerConnections: There exist capabilities to both negotiate and control the codec, bit-rate, video resolution, frame-rate, etc of a particular MediaStreamTrack in the context of one PeerConnection. Thus one method to provide multiple encodings is to establish multiple PeerConnections between A and the Mixer, where each PC is configured to provide the desired quality. Note that this solution comes in two flavors from an application perspective. One is that the same MediaStream object is added to the two PeerConnections. The second is that two different MediaStream objects, with the same number of MediaStreamTracks and representing the same sources, are created (e.g by cloning), one of them added to the first PeerConnection and the second one to the second PeerConnection.

Both of these solutions share a common requirement, the need to separate the received RTP streams not only based on media source, but also on the encoding. However, on an API level the solutions appear different. For Multiple Encodings within the context of a PC, the receiver will need new access methods for accessing and manipulating the different encodings. Using multiple PC instead requires that one can easily determine the shared (simulcasted) media source despite receiving it in multiple MediaStreams on different PCs. If the same MediaStream is added to both PC's the id's of the MediaStream and MediaStreamTracks will be the same, while they will be different if different MediaStream's (but representing the same sources) are added to the two PC's.

3.3. CLUE Telepresence

The CLUE framework [I-D.ietf-clue-framework] and use case [I-D.ietf-clue-telepresence-use-cases] documents make use of most, if not all, media concepts that were already discussed in previous sections, and adds a few more.

3.3.1. Telepresence Functionality

A communicating CLUE Endpoint can, compared to other types of Endpoints, be characterized by using multiple media resources:

- o Multiple capture devices, such as cameras or microphones, generating the media for a media source.
- o Multiple render devices, such as displays or speakers.
- o Multiple Media Types, such as audio, video and presentation streams.
- o Multiple remote Endpoints, since conference is a typical use case.
- o Multiple Encodings (encoded representations) of a media source.
- o Multiple Media Streams representing multiple media sources.

To make the multitude of resources more manageable, CLUE introduces some additional structures. For example, related media sources in a multimedia session are grouped into Scenes, which can generally be represented in different ways, described by alternative Scene Entries. CLUE explicitly separates the concept of a media source from the encoded representations of it and a single media source can be used to create multiple Encodings. It is also possible in CLUE to account for constraints in resource handling, like limitations in possible Encoding combinations due to physical device implementation.

The number of media resources typically differ between Endpoints. Specifically, the number of available media resources of a certain type used for sending at the sender side typically does not match the number of corresponding media resources used for receiving at the receiver side. Some selection process must thus be applied either at the sender or the receiver to select a subset of resources to be used. Hence, each resource that need to be part of that selection process must have some identification and characterization that can be understood by the selecting party. In the CLUE model, the sender (Provider) announces available resources and the receiver (Consumer) chooses what to receive. This choice is made independently in the two directions of a bi-directional communication.

3.3.2. Distributed Endpoint

The definition of a single CLUE Endpoint in the framework [I-D.ietf-clue-framework] says it can consist of several physical devices with source and sink media streams. This means that each logical node of such distributed Endpoint can have a separate transport interface, and thus that media sources originating from the same Endpoint can have different transport addresses.

4. Discussion

This section discusses some conclusions the authors make based on the use cases. First we will discuss commonalities between use cases. Secondly we will provide a summary of issues we see affect WebRTC. Lastly we consider aspects that need to be considered in the SDP evolution that is ongoing.

4.1. Commonalities in Use Cases

The above use cases illustrate a couple of concepts that are not well defined, nor have they fully specified standard mechanisms or behaviors. This section contains a discussion of such concepts, which the authors believe are useful in more than one context and thus should be defined to provide a common function when needed by multi-media communication applications.

4.1.1. Media Source

In several of the above use cases there exist a need for a separation between the media source, the particular encoding and its transport stream. In vanilla RTP there exist a one-to-one mapping between these; one media source is encoded in one particular way and transported as one RTP stream using a single SSRC in a particular RTP session.

The reason for not keeping a strict one-to-one mapping, allowing the media source to be identified separately from the RTP media stream (SSRC), varies depending on the application's needs and the desired functionalities:

Simulcast: Simulcast is a functionality to provide multiple simultaneous encodings of the same media source. As each encoding is independent of the other, in contrast to scalable encoding, independent transport streams for each encoding is needed. The receiver of a simulcast stream will need to be able to explicitly identify each encoding upon reception, as well as which media source it is an encoding of. This is especially important in a context of multiple media sources being provided from the same endpoint.

Mesh-based communication: When a communication application implements multi-party communication through a mesh of transport flows, there exist a need for tracking the original media source, especially when relaying between nodes is possible. It is likely that the encodings provided over the different transports are different. If an application uses relaying between different transports, an endpoint may, intentionally or not, receive multiple encodings of the same media source over the same or different transports. Some applications can handle the needed identification, but some can benefit from a standardized method to identify sources.

The second argument above can be generalized into a common need in applications that utilize multiple multimedia sessions, such as multiple PeerConnections or multiple SIP/SDP-established RTP sessions, to form a larger communication session between multiple endpoints. These applications commonly need to track media sources that occur in more than one multimedia session.

Looking at both CLUE and WebRTC, they appear to contain their own variants of the concept that was above denoted a media source. In CLUE it is called Media Capture. In WebRTC each MediaStreamTrack is identifiable, however, several MediaStreamTracks can share the actual source, and there is no way for the application to realize this currently. The identification of sources is being discussed, and there is a proposal [ref-leithead] that introduces the concept 'Track Source'. Thus, in this document we see the media source as the generalized commonality between these two concepts. Giving each media source a unique identifier in the communication session/context that is reused in all the PeerConnections or SIP/SDP-established RTP sessions would enable loop detection, correctly associate alternative encodings and provide a common name across the endpoints for application logic to reference the actual media source rather than a

particular encoding or transport stream.

It is arguable if the application should really know a long term persistent source identification, such as based on hardware identities, for example due to fingerprinting issues, and it would likely be better to use an anonymous identification that is still unique in a sufficiently wide context, for example within the communication application instance.

4.1.2. Encodings

An Encoding is a particular encoded representation of a particular media source. In the context of RTP and Signalling, a particular encoding must fit the established parameters, such as RTP payload types, media bandwidths, and other more or less codec-specific media constraints such as resolution, frame-rate, fidelity, audio bandwidth, etc.

In the context of an application, it appears that there are primarily two considerations around the use of multiple encodings.

The first is how many and what their defining parameters are. This may require to be negotiated, something the existing signalling solutions, like SDP, currently lack support for. For example in SDP, there exist no way to express that you would like to receive three different encodings of a particular video source. In addition, if you for example prefer these three encodings to be 720p/25 Hz, 360p/25 Hz and 180p/12.5 Hz, and even if you could define RTP payload types with these constraints, they must be linked to RTP streams carrying the encodings of the particular source. Also, for some RTP payload types there exist difficulties to express encoding characteristics with the desired granularity. The number of RTP payload types that can be used for a particular potential encoding can also be a constraint, especially as a single RTP payload type could well be used for all three target resolutions and frame rates in the example. Using multiple encodings might even be desirable for multi-party conferences that switches video, rather than composites and re-encodes it. It might be that SDP is not the most suitable place to negotiate this. From an application perspective, utilizing clients that have standardized APIs or protocols to control them, there exist a need for the application to express what it prefers in number of encodings as well as what their primary target parameters are.

Secondly, some applications may need explicit indication of what encoding a particular stream represents. In some cases this can be deduced based on information such as RTP payload types and parameters received in the media stream, but such implicit information will not

always be detailed enough and it may also be time-consuming to extract. For example, in SDP there is currently limitations for binding the relevant information about a particular encoding to the corresponding RTP stream, unless only a single RTP stream is defined per media description (m= line).

The CLUE framework explicitly discusses encodings as constraints that are applied when transforming a media source (capture) into what CLUE calls a capture encoding. This includes both explicit identification as well as a set of boundary parameters such as maximum width, height, frame rate as well as bandwidth. In WebRTC nothing related has yet been defined, and we note this as an issue that needs to be resolved. This as the authors expect that support for multiple encodings will be required to enable simulcast and scalability.

4.1.1.3. Synchronization contexts

The shortcomings around synchronization contexts appears rather limited. In RTP, each RTP media stream is associated with a particular synchronization context through the CNAME session description item. The main concerns here are likely twofold.

The first concern is to avoid unnecessary creation of new contexts, and rather correctly associate with the contexts that actually exist. For example, WebRTC MediaStreams are defined so that all MediaStreamTracks within a particular MediaStream shall be synchronized. An easy method for meeting this would be to assign a new CNAME for each MediaStream. However, that would ignore the fact that several media sources from the same synchronization context may appear in different combinations across several MediaStreams. Thus all these MediaStreams should share synchronization context to avoid playback glitches, like playing back different instantiations of a single media source out of sync because the media source was shared between two different MediaStreams.

The second problem is that synchronization context identification in RTP, i.e. CNAME, is overloaded as an endpoint identifier. As an example, consider an endpoint that has two synchronization contexts; one for audio and video in the room and another for an audio and video presentation stream, like the output of an DVD player. Relying on that an endpoint has only a single synchronization context and CNAME may be incorrect and could create issues that an application designer as well as RTP and signalling extension specifications need to watch out for.

CLUE discusses so far quite little about synchronization, but clearly intends to enable lip synchronization between captures that have that relation. The second issue is however quite likely to be encountered

in CLUE due to explicit inclusion of the Scene concept, where different Scenes do not require to share the same synchronization context, but is rather intended for situations where Scenes cannot share synchronization context.

4.1.4. Distributed Endpoints

When an endpoint consists of multiple nodes, the added complexity is often local to that endpoint, which is appropriate. However, some few properties of distributed endpoints needs to be tolerated by all entities in a multimedia communication session. The main item is to not assume that a single endpoint will only use a single network address. This is a dangerous assumption even for non-distributed endpoints due to multi-homing and the common deployment of NATs, especially large scale NATs which in worst case uses multiple addresses for a single endpoint's transport flows.

Distributed endpoints are brought up in the CLUE context. They are not specifically discussed in the WebRTC context, instead the desire for transport level aggregation makes such endpoints problematic. However, WebRTC does allow for fallback to media type specific transport flows and can thus without issues support distributed endpoints.

4.2. Identified WebRTC issues

In the process of identifying commonalities and differences between the different use cases we have identified what to us appears to be issues in the current specification of WebRTC that needs to be reviewed.

1. If simulcast or scalability are to be supported at all, the WebRTC API will need to find a method to deal more explicitly with the existence of different encodings and how these are configured, accessed and referenced. For simulcast, the authors see a quite straightforward solution where each PeerConnection is only allowed to contain a single encoding for a specific media source and the desired quality level can be negotiated for the full PeerConnection. When multiple encodings are desired, multiple PeerConnections with differences in configuration are established. That would only require that the underlying media source can explicitly be indicated and tracked by the receiver.
2. The current API structure allows to have multiple MediaStreams with fully or partially overlapping media sources. This, combined with multiple PeerConnections and the likely possibility to do relaying, there appears to exist a significant need to determine the underlying media source, despite receiving

different MediaStreams with particular media sources encoded in different ways. It is proposed that MediaSources are made possible to identify uniquely across multiple PeerConnections in the context of the communication application. It is however likely that while being unique in a sufficiently large context, the identification should also be anonymous to avoid fingerprinting issues, similar to the situation discussed in Section 4.1.1.

3. Implementations of the MediaStream API must be careful in how they name and deal with synchronization contexts, so that the actual underlying synchronization context is preserved when possible. It should be noted that cannot be done when a MediaStream is created that contains media sources from multiple synchronization contexts. This will instead require resynchronization of contributing sources, creation of a new synchronization context, and inserting the sources into that synchronization context.

These issues need to be discussed and an appropriate way to resolve them must be chosen.

4.3. Relevant to SDP evolution

The joint MMUSIC / RTCWeb WGs interim meeting in February 2013 will discuss a number of SDP related issues around the handling of multiple sources; the aggregation of multiple media types over the same RTP session as well as RTP sharing its transport flow not only with ICE/STUN but also with the WebRTC data channel using SCTP/DTLS/UDP. These issues will potentially result in a significant impact on SDP. It may also impact other ongoing work as well as existing usages and applications, making these discussions difficult.

The above use cases and discussion points to the existence of a number of commonalities between WebRTC and CLUE, and that a solution should preferably be usable by both. It is a very open question how much functionality CLUE requires from SDP, as CLUE WG plans to develop a protocol with a different usage model. The appropriate division in functionality between SDP and this protocol is currently unknown.

Based on this document, it is possible to express some protocol requirements when negotiating multimedia sessions and their media configurations. Note that this is written as requirements to consider, given that one believes this functionality is needed in SDP.

The Requirements:

Encoding negotiation: For Simulcast and Scalability in applications, it must be possible to negotiate the number and the boundary conditions for the desired encodings created from a particular media source.

Media Resource Identification: SDP-based applications that need explicit information about media sources, multiple encodings and their related RTP media streams could benefit from a common way of providing this information. This need can result in multiple different actual requirements. Some require a common, explicit identification of media sources across multiple signalling contexts. Some may require explicit indication of which set of encodings that has the same media source and thus which sets of RTP media streams (SSRCs) that are related to a particular media source.

RTP media stream parameters: With a greater heterogeneity of the possible encodings and their boundary conditions, situations may arise where some or sets of RTP media streams will need to have specific sets of parameters associated with them, compared to other (sets of) RTP media streams.

The above are general requirements and in some cases the appropriate point to address the requirement may not even be SDP. For example, media source identification could primarily be put in an RTCP Session Description (SDS) item, and only when so required by the application also be included in the signalling.

The discussion in this document has impact on the high level decision regarding how to relate RTP media streams to SDP media descriptions. However, as it is currently presenting concepts rather than giving concrete proposals on how to enable these concepts as extensions to SDP or other protocols, it is difficult to determine the actual impact that a high level solution will have. However, the authors are convinced that neither of the directions will prevent the definition of suitable concepts in SDP.

5. IANA Considerations

This document makes no request of IANA.

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

6. Security Considerations

The realization of the proposed concepts and the resolution will have security considerations. However, at this stage it is unclear if any has not already common considerations regarding preserving privacy, confidentiality and ensure integrity to prevent denial of service or quality degradations.

7. Informative References

- [I-D.ietf-avtcore-multi-media-rtp-session]
Westerlund, M., Perkins, C., and J. Lennox, "Multiple Media Types in an RTP Session", draft-ietf-avtcore-multi-media-rtp-session-01 (work in progress), October 2012.
- [I-D.ietf-clue-framework]
Duckworth, M., Pepperell, A., and S. Wenger, "Framework for Telepresence Multi-Streams", draft-ietf-clue-framework-08 (work in progress), December 2012.
- [I-D.ietf-clue-telepresence-use-cases]
Romanow, A., Botzko, S., Duckworth, M., Even, R., and I. Communications, "Use Cases for Telepresence Multi-streams", draft-ietf-clue-telepresence-use-cases-04 (work in progress), August 2012.
- [I-D.ietf-mmusic-sdp-bundle-negotiation]
Holmberg, C. and H. Alvestrand, "Multiplexing Negotiation Using Session Description Protocol (SDP) Port Numbers", draft-ietf-mmusic-sdp-bundle-negotiation-01 (work in progress), August 2012.
- [I-D.ietf-rtcweb-overview]
Alvestrand, H., "Overview: Real Time Protocols for Browser-based Applications", draft-ietf-rtcweb-overview-05 (work in progress), December 2012.
- [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.

- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC6465] Iovov, E., Marocco, E., and J. Lennox, "A Real-time Transport Protocol (RTP) Header Extension for Mixer-to-Client Audio Level Indication", RFC 6465, December 2011.
- [ref-leithead]
Microsoft, "Proposal: Media Capture and Streams Settings API v6, https://dvcs.w3.org/hg/dap/raw-file/tip/media-stream-capture/proposals/SettingsAPI_proposal_v6.html", December 2012.
- [ref-media-capture]
"Media Capture and Streams, <http://dev.w3.org/2011/webrtc/editor/getusermedia.html>", December 2012.
- [ref-webrtc10]
"WebRTC 1.0: Real-time Communication Between Browsers, <http://dev.w3.org/2011/webrtc/editor/webrtc.html>", January 2013.

Authors' Addresses

Bo Burman
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

Phone: +46 10 714 13 11
Email: bo.burman@ericsson.com

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

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

AVTEXT WG
Internet-Draft
Updates: RFC5104 (if approved)
Intended status: Standards Track
Expires: July 28, 2013

R. Even
>Huawei Technologies
January 24, 2013

Pausing an RTP Media Stream
draft-even-avtext-flow-control-to-zero-00.txt

Abstract

In Real-time multimedia applications using multiple media streams in point to point and multipoint calls can benefit from options that will enable them to pause and resume media streams as well as to indicate a mute state. This document describes the difference between pause and mute and describes how to provide this required functionality. This document updates RFC5104.

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 July 28, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. Pause and Resume	4
4. Mute and Not Rendered	4
5. Acknowledgements	5
6. IANA Considerations	5
7. Security Considerations	5
8. References	5
8.1. Normative References	5
8.2. Informative References	5
Author's Address	5

1. Introduction

Real-time multimedia communication topologies are typical point to point or multipoint. The multipoint call may use an MCU or an RTP mixer as specified in [RFC5117]. the call one of the parties may want to stop receiving one of the media stream temporarily. In order to do it the receiver will want the sender to stop sending media.

In the point to point case the receiver can ask the sender to reduce the media rate to zero and later ask for a new bit rate.

In the multipoint case, the central mixer if not using one of the streams may ask the sender to stop sending similar to the point to point case. A multipoint conference participant receiving streams from the MCU or RTP mixer behave like a point to point receiver in this case asking the MCU or RTP mixer to stop sending media. For a discussion on the use case look at section 3 of [I-D.westerlund-avtext-rtp-stream-pause].

During a point to point or multipoint call a sender may mute his microphone or camera but still continue to send media which may be some syntactic media. The receivers will benefit if they receive an indication about this state so it can also indicate to the user that the other side is muted. If the receiver is not rendering a stream it may indicate to the sender that he is not being seen or heard at the moment even though he is receiving the media stream.

To summarize there is a need for a pause and resume commands and for mute and not rendered indications.

All this messages are used for temporary flow change during the call and are intended to go end to end. The recommended proposal is to use RTCP Codec Control Messages [RFC5104] to achieve this functionality.

The document updates the current TMMBR message in [RFC5104] and adds new indications for the mute and not rendered states.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119[RFC2119] and indicate requirement levels for compliant RTP implementations.

3. Pause and Resume

TMMBR as specified in [RFC5104] is used by video conferencing systems for flow control. It will be useful if the same method can be used for pause and resume.

For the pause request using TMMBR with bit rate "0" will make an RTP media sender stop sending an RTP stream if it is used by others that have not requested a pause. This is not a problem for the point to point case and in the RTP media receiver to MCU/RTP mixer case in section 3 of [I-D.westerlund-avtext-rtp-stream-pause] since there is a single receiver on each call. The MCU / RTP mixer may send a pause request to the RTP media sender if he is not using the RTP stream for creating an outgoing stream.

RFC5104 [RFC5104] provides guidelines on how to apply an increase in the temporary rate change when there are multiple receivers. It recommends delaying the rate increase allowing all receivers to agree with the change. The major reason for it is that RFC5104 [RFC5104] allows using TMMBR to request a bandwidth that is higher than the current one negotiated using SDP "b" attribute.

This document recommends that in order to create consistency between the SDP negotiated bandwidth and TMMBR it will not be allowed to ask in TMMBR for a bandwidth that is higher than the SDP negotiated one. This may also be important for intermediaries that monitor bandwidth usage based on SDP. Based on this change there is also no need for the single receiver case to delay the increase in bandwidth when resuming the media stream in the point to point or RTP media receiver to MCU/ Mixer case. Note that TMMBR is request for maximum and the MCU/ Mixer may send at a lower rate if he cannot provide a higher rate based on the bit rate of the sender of this RTP media stream.

The MCU / Mixer may negotiate a higher bit rate using TMMBR / TMBBN based on mixing /transcoding model supported.

The RTP media stream receiver will signal Pause by TMMBR with zero value and Resume using TMMBR with the maximum bit rate that the receiver wants.

4. Mute and Not Rendered

Place holder to add new indications if people feel that these indications will be useful.

5. Acknowledgements

place holder

6. IANA Considerations

TBD

7. Security Considerations

TBD.

8. References

8.1. Normative References

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

[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.

8.2. Informative References

[I-D.westerlund-avtext-rtp-stream-pause]
Akram, A., Burman, B., Grondal, D., and M. Westerlund,
"RTP Media Stream Pause and Resume",
draft-westerlund-avtext-rtp-stream-pause-03 (work in
progress), October 2012.

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

Author's Address

Roni Even
>Huawei Technologies
Tel Aviv,
Israel

Email: roni.even@mail01.huawei.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 22, 2013

J. Lennox
Vidyo
K. Gross
AVA
February 18, 2013

A Taxonomy of Grouping Semantics and Mechanisms for Real-Time Transport
Protocol (RTP) Sources
draft-lennox-raiarea-rtp-grouping-taxonomy-00

Abstract

The terminology about, and associations among, Real-Time Transport Protocol (RTP) sources can be complex and somewhat opaque. This document describes a number of existing and proposed relationships among RTP sources, and attempts to define common terminology for discussing protocol entities and their relationships.

This document is still very rough, but is submitted in the hopes of making future discussion productive.

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 August 22, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
2. RTP Terminology	4
2.1. Synchronization Source	4
2.2. RTP Session	4
2.3. Contributing Source	5
2.4. Multimedia Session	5
2.5. Media Source	5
3. Terminology used other protocols and groups	5
3.1. SDP	5
3.1.1. SDP Multimedia Session	6
3.1.2. CLUE	6
3.1.3. WebRTC	7
4. Grouping of RTP sources	8
4.1. Overview of hierarchical relationships among groups	8
4.2. Existing and proposed source group semantics	8
4.2.1. Relationships among media sources	8
4.2.2. Alternative representations of media sources	9
4.2.3. Robustness and Repair	9
4.2.4. Reporting Associations	9
5. Existing and proposed mechanisms for describing groups	9
6. Security Considerations	10
7. Open Issues	10
8. IANA Considerations	10
9. Informative References	10
Appendix A. RTP terminology	11
A.1. Capture	11
A.2. CNAME	11
A.3. End system	11
A.4. Group	11
A.5. Multimedia session	12
A.6. Multi-stream transmission	12
A.7. Receiver	12
A.8. Repair stream	12
A.9. RTP endpoint	12
A.10. RTP session	12
A.11. Scene	12
A.12. Sender	12
A.13. Simulcast	12
A.14. Source	12
A.15. Media stream	13
A.16. Switched capture	13
A.17. Track	13
Authors' Addresses	13

1. Introduction

Terminology and understanding about sources in RTP, and how they relate, is very confused. Different protocols use the same terms differently, and complexities addressed at one layer are often glossed over or ignored at another.

This document attempts to provide some clarity by reviewing the semantics of various aspects of sources in RTP. As an organizing mechanism, it approaches this by describing various ways that RTP sources can be grouped and associated together.

2. RTP Terminology

2.1. Synchronization Source

In RTP, the smallest unit of media transport is the "synchronization source" (SSRC). Defined in [RFC3550], a synchronization source is the source of a stream of RTP packets, identified by a 32-bit numeric SSRC identifier carried in the RTP header of every RTP packet. All packets from a synchronization source form part of the same timing and sequence number space, so a receiver groups packets by synchronization source for playback. In the most common cases, a synchronization source is a single flow of encoded media that can be fed to a single decoding process.

In some more complex cases, a synchronization source is also used to encode repair flows and sub-flows of a layered encoding. See Section 4.2.2 and Section 4.2.3 below.

(Note: as the definition above mentions, "synchronization source" in [RFC3550] technically refers to the **source** of the stream of RTP packets, not the packets themselves. A better term for the stream could be helpful, and the authors are open to one; however, finding a good term that doesn't collide with terms used in other standards is very difficult. In particular, both "stream" and "flow" are overloaded enough that their use would probably confuse more than it clarifies.)

2.2. RTP Session

The RTP protocol transmits sources in an "RTP session". An RTP session is an association among a group of participants communicating with RTP. It is a group communications channel which carries a number of synchronization sources. Within an RTP session, every participant finds out meta-data and control information (over RTCP) about all the synchronization sources in the RTP session, all

synchronization source identifiers are unique in the RTP session, and the bandwidth of the RTCP control channel is shared by all the synchronization sources in the session.

2.3. Contributing Source

In an RTP session, in some cases, the media flows of some synchronization sources are not forwarded by RTP middleboxes (known as mixers and translators). In the case of mixers, each of the synchronization sources that are used to create a mixed stream of media becomes visible instead as a "contributing source" (CSRC), whose media is no longer directly visible in some parts of the RTP session, but whose meta-data is still known by all participants through RTCP.

2.4. Multimedia Session

A group of RTP sessions can be grouped into a "multimedia session", a group of concurrent, associated RTP sessions among a common group of participants. A multimedia session defines logical relationships among sources that appear in multiple RTP sessions.

2.5. Media Source

At a higher level than this is the "media source", a single piece of media that can be independently rendered. While in the simplest cases a synchronization source uniquely encodes a media source, many more complicated cases, discussed in Section 4.2.2 and Section 4.2.3 below, associate multiple synchronization sources together to provide various methods and options for encoding a media source. (In some of these cases, these multiple synchronization sources are even sent in separate RTP sessions, though always in the same multimedia session.)

Unlike the other terminology defined in this section, "media source" is not a well-established term; existing protocols use a number of different terms to define similar concepts, described below. ("Media source" may not be the best term in for this concept; the authors of this document are open to better ones.)

3. Terminology used other protocols and groups

3.1. SDP

The Session Description Protocol [RFC4566] provides a mechanism for describing -- and, with SDP Offer/Answer [RFC3264], negotiation -- multimedia sessions.

3.1.1. SDP Multimedia Session

SDP uses the term "multimedia session" in essentially the same manner as RTP.

3.1.1.1. Media Stream

"Media stream" is perhaps one of the most confusing terms in SDP. It is never explicitly defined in [RFC4566]; its meaning can only be inferred from descriptions of its properties, the SDP syntax used to describe it, and its usage in other protocols.

In SDP, a media stream is described by a "media description", a block of SDP starting with an "m=" line and followed by other SDP fields which are scoped to that media description.

When SDP is used to describe a media stream that uses RTP, the attributes used (in the core SDP protocol) all define attributes of an RTP session. Furthermore, [RFC4566] does note that "Media streams can be many-to-many." Thus, when SDP describes a media stream that uses RTP, the RTP-level protocol element that is described is an RTP session, which can carry any number of synchronization sources (and thus media sources).

However, for a number of reasons, many implementations of SDP, notably SIP devices, instead interpret SDP media streams as containing only a single media source in each direction (assuming unicast flows). There are a number of reasons this implementation decision was chosen, including the lack of definition of "media stream" in the SDP specification; the poorly chosen name (the term "media stream" seems like something that should refer to what this document calls a "media source"); the fact that core SDP, and offer/answer, provide no description or negotiation mechanisms for anything more fine-grained than a media source; and the fact that most existing implementations have no need for more than one media source and received of each media type.

Other than in discussions of other existing terminology, this document attempts to avoid the use of the word "stream".

3.1.2. CLUE

The CLUE working group is ongoing work in the IETF to define protocols and mechanism for interoperable telepresence systems. Its architecture and terms are defined in [I-D.ietf-clue-framework].

3.1.2.1. CLUE Capture Scene

In CLUE, a "Capture Scene" is a structure representing a spatial region containing one or more Capture Devices, each capturing media representing a portion of the region. The spatial region represented by a scene may or may not correspond to a real region in physical space, such as a room. A capture scene includes attributes and one or more capture scene entries, with each entry including one or more media captures.

3.1.2.2. CLUE Capture Scene Entry

In CLUE, a "Capture Scene Entry" is a list of Media Captures of the same media type that together form one way to represent the entire Capture scene.

3.1.2.3. CLUE Capture

In CLUE, a "Media Capture" is a source of Media, such as from one or more Capture Devices or constructed from other Media streams.

The "Capture" is CLUE's term that is most closely equivalent to what this document calls a "media source"; however, some usages (such as "dynamic sources" of "switched captures") may be more complex, and are still actively being defined.

3.1.2.4. CLUE Capture Encoding

In CLUE, a "Capture Encoding" is a specific encoding of a Media Capture, to be sent by a Media Provider to a Media Consumer via RTP.

In simulcast scenarios (see Section 4.2.2), multiple capture encodings can be used simultaneously to transmit a single capture.

3.1.3. WebRTC

WebRTC is ongoing work in the IETF and W3C to define mechanisms and APIs to allow Javascript applications in web browsers to participate in multimedia sessions. An overview can be found in [I-D.ietf-rtcweb-overview].

3.1.3.1. WebRTC RtcMediaStreamTrack

RtcMediaStreamTrack

An "RtcMediaStreamTrack" is an object in WebRTC that maps most closely to what this document calls a "media source", though some of the details remain to be defined. However, as WebRTC is defining an

API, the `RtcMediaStreamTrack` object also includes a good deal of meta-data about the media source, such as an ID, enabled/disabled state, and so forth.

3.1.3.2. WebRTC `RtcMediaStream`

An "`RtcMediaStream`" in WebRTC is an object that groups a set of `RtcMediaStreamTracks` that are synchronized with each other.

4. Grouping of RTP sources

Synchronization sources can have a large variety of relationships among them. These relationships can apply both between sources within a single session, and between sources that occur in multiple sessions.

Ways of relating synchronization sources typically involve groups: a set of synchronization sources has some relationship that applies to all those in the group, and no others. (Relationships that involve arbitrary non-grouping associations among synchronization sources, such that e.g., A relates to B and B to C, but A and C are unrelated, are uncommon if not nonexistent.)

In many cases, the semantics of groups are not simply that the sources form an undifferentiated group, but rather that members of the group have certain roles.

4.1. Overview of hierarchical relationships among groups

Many (though not all) associations among groups can be described hierarchically.

These hierarchical groups can be divided into three large categories: associations among independent media sources; alternative representations of media sources; and mechanisms for robustness and repair of a particular representation of a media source.

4.2. Existing and proposed source group semantics

We can divide group semantics into several broad categories.

(TODO: each of the items below needs to be described in detail.)

4.2.1. Relationships among media sources

Synchronization Contexts -- things identified by a CNAME.

CLUE Scenes (and their substructures, down to Captures.).

WebRTC MediaStreams.

Clock source signaling.

4.2.2. Alternative representations of media sources

Simulcast.

Multi-Stream Transmission of Layered Codecs.

The original FID description in RFC 5888 (grouping) Section 8.

4.2.3. Robustness and Repair

Forward Error Correction (FEC). (But in some cases this can also be across multiple media sources!)

Retransmission.

4.2.4. Reporting Associations

RTP also uses SSRC values to identify receivers of media, the source of feedback about synchronization sources (and thus, also media sources). In some cases these receivers are also grouped. However, these associations are not associations of senders of media, and thus are not the same type of relationship as the ones described above.

Inter-Domain Media Synchronization

Reporting groups.

5. Existing and proposed mechanisms for describing groups

SDP groups

SDP source groups

SRCNAME

MSID

CLUE

Application-specific SDP attributes

SDP "number of ports" field to create multiple sessions with SSRC alignment.

6. Security Considerations

Different for each way of grouping. Is there anything we can generalize?

Hopefully having a well-defined common terminology and understanding of the complexities of the RTP architecture will help lead us to better standards, avoiding security problems.

7. Open Issues

Much of the terminology is still a matter of dispute.

It might be useful to distinguish between a single endpoint's view of a source, or RTP session, or multimedia session, versus the full set of sessions and every endpoint that's communicating in them, with the signaling that established them.

(Sure to be many more...)

8. IANA Considerations

This document makes no request of IANA.

9. Informative References

[I-D.ietf-clue-framework]

Duckworth, M., Pepperell, A., and S. Wenger, "Framework for Telepresence Multi-Streams", draft-ietf-clue-framework-08 (work in progress), December 2012.

[I-D.ietf-rtcweb-overview]

Alvestrand, H., "Overview: Real Time Protocols for Browser-based Applications", draft-ietf-rtcweb-overview-05 (work in progress), December 2012.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC6222] Begen, A., Perkins, C., and D. Wing, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)", RFC 6222, April 2011.

Appendix A. RTP terminology

(TODO: This list of definitions needs to be cleaned up and expanded, if it's useful. Possibly its content should simply be merged into the appropriate sections of the main document, instead.)

A.1. Capture

"Capture" is the term used in the IETF CLUE Telepresence Framework to refer to what this document calls a "media source".

A.2. CNAME

An RTP canonical name (RTP CNAME or CNAME) is a persistent transport-level identifier for an RTP endpoint (Appendix A.9). CNAMEs must be unique within an RTP session (Section 2.2). The RTCP CNAME can be either persistent across different RTP sessions for an RTP endpoint or unique per session, meaning that an RTP endpoint chooses a different CNAME for each session.[RFC6222] receivers (Appendix A.7) require the CNAME to keep track of each participant. Receivers may also require the CNAME to associate multiple data streams from a given participant in a set of related RTP sessions, for example to synchronize audio and video.

A.3. End system

An application that generates the content to be sent in RTP packets and/or consumes the content of received RTP packets. An end system can produce one or more synchronization sources (Section 2.1) in a particular RTP session (Section 2.2).[RFC3550]

A.4. Group

A.5. Multimedia session

A set of concurrent, associated RTP sessions (Section 2.2) among a common group of participants.[RFC3550]

A.6. Multi-stream transmission

Multi-stream transmission (MST) is a mechanism by which different portions of a layered encoding of a media stream are sent using separate synchronization sources (sometimes in separate RTP sessions). MSTs are useful for receiver control of layered media.

A.7. Receiver

A.8. Repair stream

A repair stream is a secondary stream used for error recovery. The repair mechanisms available include

- o duplication of the original stream,
- o duplication of the original stream with a time offset,
- o forward error correction (FEC) techniques, and.
- o retransmission of lost packets (either globally or selectively).

A.9. RTP endpoint

A.10. RTP session

An RTP session or simply, session is an association among a set of participants communicating with RTP.[RFC3550]

A.11. Scene

A scene is associated with IETF CLUE and describes a collection of captures (Appendix A.1) and the metadata required to make associations (e.g. relative spacial orintation of cameras) between the captures.

A.12. Sender

A.13. Simulcast

A.14. Source

A synchronization source (SSRC) is the source of a stream of RTP packets, identified by a 32-bit numeric SSRC identifier carried in the RTP header so as not to be dependent upon the network address. All packets from a synchronization source form part of the same timing and sequence number space, so a receiver groups packets by

synchronization source for playback.[RFC3550]

A.15. Media stream

A.16. Switched capture

A switched capture is used in teleconference applications and is a stream (Appendix A.15) whose content is selected from one of a selection of sources. The source is selected based on an algorithm in the equipment producing the stream for example, a closeup of the person currently speaking in a conference room.

A.17. Track

A track is associated with W3C WebRTC and IETF RTCWeb projects. A track is defined by SSRC and CNAME. A track is the smallest media entity that may be independently rendered. A single audio channel is a track.

Authors' Addresses

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

Email: jonathan@vidyo.com

Kevin Gross
AVA Networks, LLC
Boulder, CO
US

Email: kevin.gross@avanw.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2013

A. Akram
B. Burman
D. Grondal
M. Westerlund
Ericsson AB
October 22, 2012

RTP Media Stream Pause and Resume
draft-westerlund-avtext-rtp-stream-pause-03

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 adding a group of new real-time feedback messages used to pause and resume RTP data streams.

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 April 25, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
2. Definitions	5
2.1. Abbreviations	5
2.2. Terminology	5
2.3. Requirements Language	6
3. Use Cases	6
3.1. Point to Point	6
3.2. RTP Mixer to Media Sender	7
3.3. Media Receiver to RTP Mixer	8
3.4. Media Receiver to Media Sender Across RTP Mixer	8
4. Design Considerations	9
4.1. Real-time Nature	9
4.2. Message Direction	9
4.3. Apply to Individual Sources	9
4.4. Consensus	9
4.5. Acknowledgements	10
4.6. Retransmitting Requests	10
4.7. Sequence Numbering	10
5. Relation to Other Solutions	11
5.1. Signaling Technology Performance Comparison	11
5.2. SDP "inactive" Attribute	19
5.3. CCM TMMBR / TMMBN	19
5.4. Media Stream Selection	20
5.5. Media Source Selection in SDP	21
5.6. Codec Operation Point	21
5.7. Conclusion	22
6. Solution Overview	22
6.1. Expressing Capability	23
6.2. Requesting to Pause	23
6.3. Media Sender Pausing	24
6.4. Requesting to Resume	25
7. Participant States	26
7.1. Playing State	27
7.2. Pausing State	27
7.3. Paused State	28
7.3.1. RTCP BYE Message	28
7.3.2. SSRC Time-out	28
7.4. Local Paused State	29

8. Message Format	29
9. Message Details	31
9.1. PAUSE	31
9.2. PAUSED	32
9.3. RESUME	33
9.4. REFUSE	34
9.5. Transmission Rules	34
10. Signalling	35
10.1. Offer-Answer Use	37
10.2. Declarative Use	38
11. Examples	39
11.1. Offer-Answer	39
11.2. Point-to-Point Session	40
11.3. Point-to-multipoint using Mixer	43
11.4. Point-to-multipoint using Translator	45
12. IANA Considerations	48
13. Security Considerations	49
14. Acknowledgements	49
15. References	49
15.1. Normative References	49
15.2. Informative References	50
Authors' Addresses	52

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 and demands control over 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] media stream from a specific source 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, which either uses logic based on voice activity, other measurements, user input over proprietary 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. If the number of conference participants are greater than what the conference logic has chosen to present simultaneously to receiving participants, some participant media streams sent to the RTP Mixer may not need to be forwarded to any other participant. Those media 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 media streams required at any given point in time is highly dynamic in such scenarios, 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 appropriate. No currently existing RTCP feedback message supports pausing and resuming an incoming data 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 should thus be defined as a Codec Control Message (CCM) extension.

2. Definitions

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, PAUSED as Indication and REFUSE as Notification.

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 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 receiver which sends a PAUSE request, defined in this specification, 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.



Figure 1: 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 be 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.

RTCWEB WG's use case and requirements document [I-D.ietf-rtcweb-use-cases-and-requirements] defines the following

API requirement derived in the most basic usage "Simple Video Communication Service" (Section 4.2.1 in [I-D.ietf-rtcweb-use-cases-and-requirements]) :

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.

The PAUSED indication in this document can be used to indicate to the media receiver that the stream delivery is deliberately paused due to a sender side mute operation.

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.

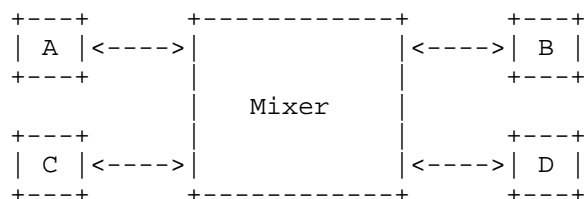


Figure 2: RTP Mixer

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 be that the number of senders 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 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 scalable encoded (for Multi-Stream Transmission), 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 in Figure 2 could potentially request to pause the delivery of a given media stream. Possible reasons include the ones in the point to point case (Section 3.1) above.

3.4. 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 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 its stream indications. Secondly, an end-point cannot rely on the CSRC to correctly identify the media stream to be paused when the delivered media is some type of mix. A more elaborate media stream identification solution is needed to support this in the general case. Thirdly, the end-point cannot determine if a given media stream is still needed by the RTP Mixer to deliver to another session participant.

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 RTP media stream the selected remote RTP media stream is to be provided gives richer functionality.

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

4.3. Apply to Individual Sources

The PAUSE and RESUME messages apply to single RTP 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. PAUSED indications refer to a single one of the sender's own, paused SSRC.

4.4. Consensus

An RTP media stream sender should not pause an SSRC that some receiver still wishes to receive. 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 single-handedly cause the media 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. Acknowledgements

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 media stream sender. When PAUSE or RESUME reaches the RTP media stream sender and are effective, it is immediately seen from the arrival or non-arrival of RTP packets for that RTP media stream. Thus, no explicit acknowledgements are required in this case.

In some cases when a PAUSE or RESUME message reaches the media sender, it will not be able to pause or resume the stream due to some local consideration. This error condition, a negative acknowledgement, is needed to avoid unnecessary retransmission of requests (Section 4.6).

4.6. Retransmitting Requests

When the media stream is not affected as expected by a PAUSE or RESUME request, it 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 media 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

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:

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

- o Mobile wireless access according to 3GPP LTE [3GPP.36.201], 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 [3GPP.25.308] (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:

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

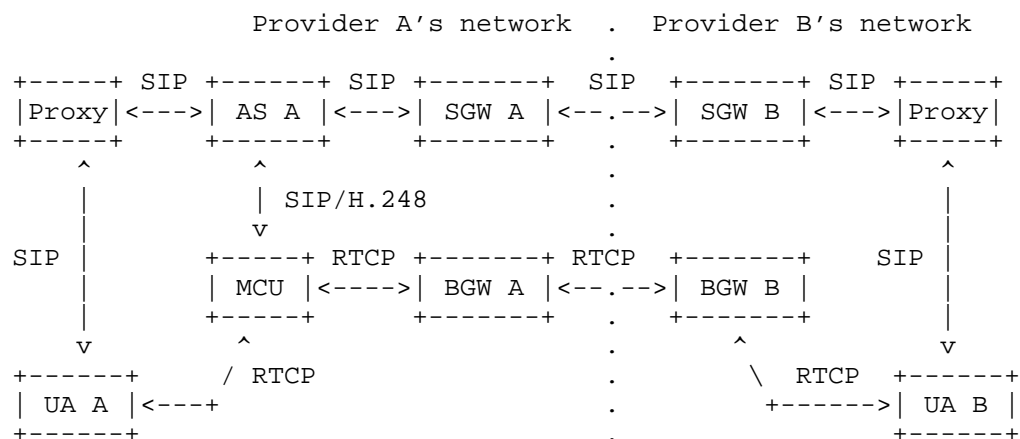


Figure 3: Comparison Signaling Topology

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:

- o 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 [3GPP.26.114], 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.
- o 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.

- o 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.
- o 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.
- o 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:

- o 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.
- o 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.
- o Since there are random delay factors in RTCP transmission, the expected, most probable value is used in the estimates.
- o 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.
- o 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 ($T_{\text{dither_max}}$ 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 $T_{\text{dither_max}}$, 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.
- o 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.

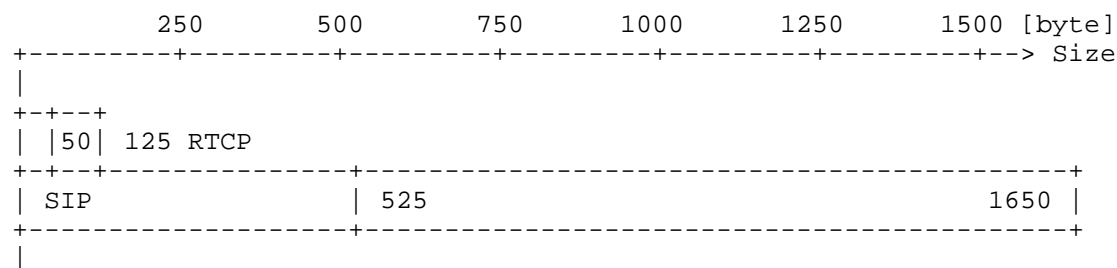


Figure 4: Message Size Comparison

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.

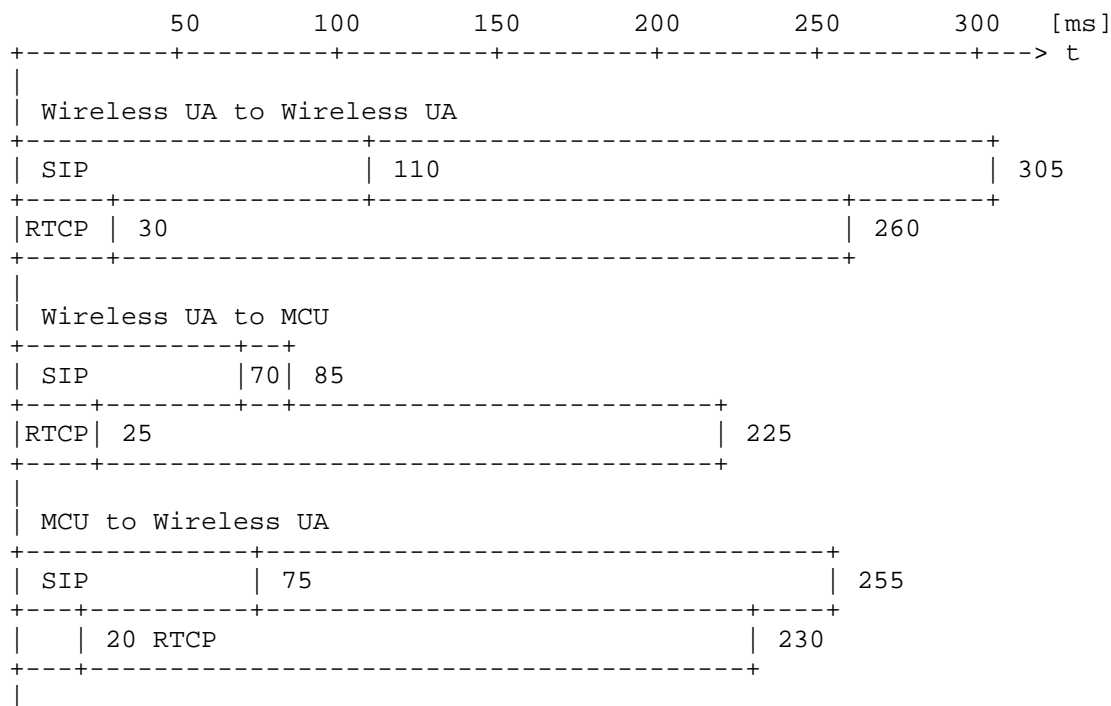


Figure 5: Mobile Access Transmission Delay Comparison

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 3.3. 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.

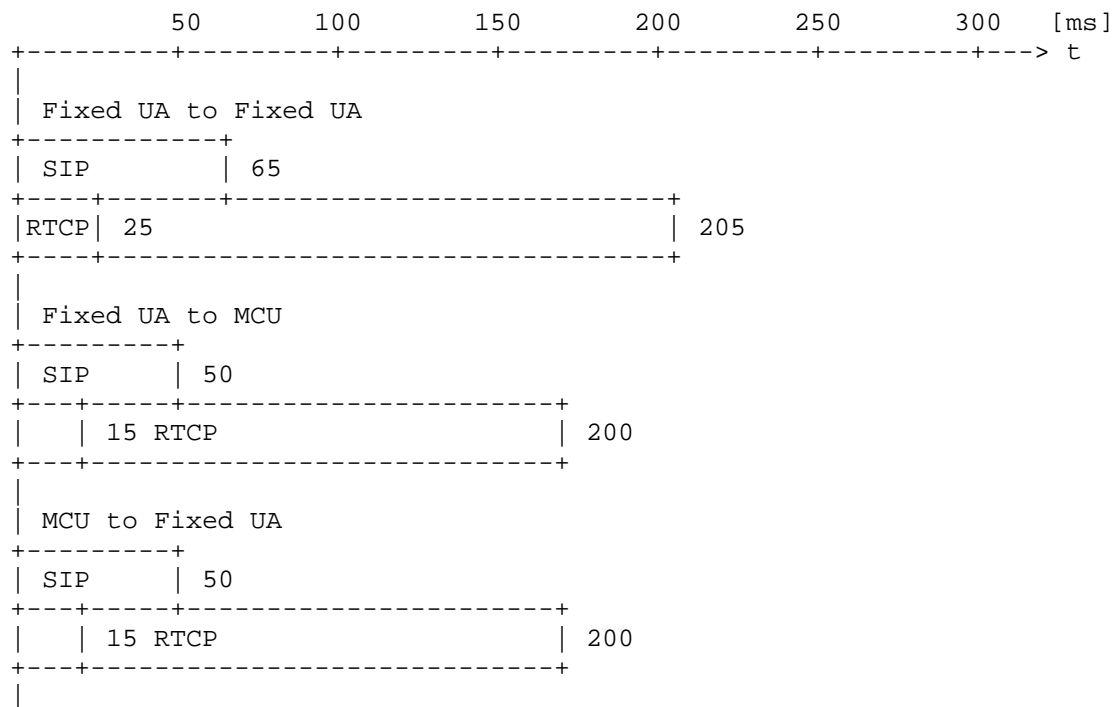


Figure 6: Fixed Access Transmission Delay Comparison

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 reconsideration. 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 [RFC3356]. For RTP media 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 media 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. 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 media 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.

5.3. 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 seemingly provide some of the necessary functionality, using a bitrate value of 0 as PAUSE request and PAUSED indication, respectively. 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).

The defined semantics for TMMBR differ significantly from what is required for PAUSE. When there are several receivers of the same

media 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. The TMMBR semantics are the opposite; the first media receiver that sends TMMBR 0 will pause the stream for all receivers. This does not matter in the point-to-point case since there is only a single receiver, but will not provide the desired functionality in other cases.

A possibly even more significant aspect is the guard period present in TMMBR when increasing the bandwidth. The equivalent of a RESUME request using TMMBR would be to send a TMMBR message with a non-zero value, likely at the level of maximum for the session. However, TMMBR/TMMBN semantics (Section 4.2.1.2 of [RFC5104]) requires a media 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 media sender knows in the RTP session. This applies independently of topology, i.e. also for a point to point session. Compared to the proposed solution, this adds several delays. First, there is the delay between the media sender receiving the TMMBR until it can send a TMMBN, then there is the above delay for the guard period before the media sender resumes transmission. This delay before resuming transmission is the most time critical operation in this solution, making use of TMMBR according to the defined semantics infeasible in practice.

A TMMBN message could arguably be used as an acknowledgement (Section 4.5) of either PAUSE or RESUME (depending on zero or non-zero bitrate parameter), but will not be able to provide any sequence number functionality (Section 4.7) and will thus risk mis-interpretation due to race situations.

5.4. Media Stream Selection

The Media Stream Selection draft [I-D.westerlund-dispatch-stream-selection] includes a functionality that allows to include and exclude a specified stream from a received set of streams, which arguably gives similar results as pausing a stream. The functionality described in that specification is mainly transport agnostic, but the proposed implementation is to extend BFCP [RFC4582], which would likely give a performance somewhere in between RTCP and SIP.

The semantics differ between exclude / include and pause / resume for a stream in topologies other than point-to-point. For example, in RTP Receiver to Mixer (Section 3.3), 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 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 stream (CSRC) to be included in the mix again.

It would in fact be possible to use media stream selection for SSRC from the mixer itself, not CSRC, to achieve pause and resume functionality when UA and mixer are connected point-to-point, but that would not work with Translator or in multipoint, and neither would it provide any RTP level indication that the stream is paused.

In the Mixer to RTP sender (Section 3.2) case, the topology is sufficiently similar to point-to-point to make it possible to use Media Stream Selection for pause and resume functionality, but would still not provide any RTP level indication that the stream is paused.

In topologies where the same RTP media stream is received by several receivers, Media Stream Selection does not provide any functionality to achieve consensus (Section 4.4) and will need modification to be possible to use.

5.5. Media Source Selection in SDP

There is also a similar 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 5.2). This suffers partly from the same mis-match in semantics and lack of functionality for consensus as the section above (Section 5.4), and would likely also suffer from lower real-time performance (Section 4.1), especially when implementing necessary signaling to reach consensus (Section 4.4).

5.6. Codec Operation Point

The draft on Codec Operation Point (COP) [I-D.westerlund-avtext-codec-operation-point] includes functionality to request a stream to be encoded at a certain bandwidth, including 0. That could also be used to pause and resume a stream. Since that draft is also based on CCM [RFC5104], the performance should be very similar to this specification and it should be possible to achieve sufficiently low signaling delay (Section 4.1).

The message semantics of COP (Section 4.4) also suits the purpose of this specification, and it would be possible to use COP for the sole purpose of controlling bandwidth in a way that effectively constitutes pause and resume. COP also has a functionality that this specification does not, in that it can set the bandwidth individually for sub-streams within a single SSRC, for example in scalable Single Stream Transmission (SST).

The authors however believe that there will be applications that can make good use of pause and resume functionality, but that will not be able to motivate a COP implementation, not even just the 'bitrate' Parameter Type (support for each Parameter Type is individually negotiable in COP). It is thus believed that pause and resume functionality is motivated as a separate specification.

5.7. 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 this 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.

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 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, 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 media streams.

The PAUSED functionality is also defined in such a way that it can be used standalone by the media 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).

6.2. Requesting to Pause

An RTP media stream receiver can choose to request PAUSE at any time, subject to AVPF timing rules.

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 media stream (SSRC).

When a non-paused RTP media stream sender receives the PAUSE request, it continues to send media while waiting for some time to allow other RTP media 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 media 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.

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

If the PAUSE request is lost before it reaches the RTP media stream sender, it will be discovered by the RTP media stream receiver because it continues to receive the RTP media stream. It will also not see any PAUSED indication (Section 6.3) for the stream. The same condition can be caused by the RTP media stream sender having received a disapproving RESUME for the PAUSE request, but that the

PAUSE sender did not receive the RESUME and may instead think that the PAUSE was lost. In both cases, the PAUSE request can be re-transmitted using the same PauseID.

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 media stream sender receiving a PAUSE not using the available PauseID informs the RTP media 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 media 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 media stream receivers that wish to receive the stream.

A similar situation where an ineffective PauseID is chosen can appear when a new RTP media 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 media 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.

If the RTP media stream sender receives several identical PAUSE for an RTP media 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 media 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. 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 media 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 media 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 media 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 media 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 media 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 media 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 media stream receiver can request to resume a stream at any time, subject to AVPF timing rules.

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

A pausing RTP media 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 media 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 media stream receivers SHOULD then avoid sending further RESUME requests for some reasonable amount of time, to allow the condition to clear.

If the RTP media stream sender receives several identical RESUME for an RTP media 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 media 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 media sender, it requires the media 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 media sender. If temporal dependency is not used when media 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 media sender when resuming, and if the media receiver did not save any sample from before the pause, the media 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.

7. Participant States

This document introduces three new states for a media stream in an RTP sender, according to the figure and sub-sections below.

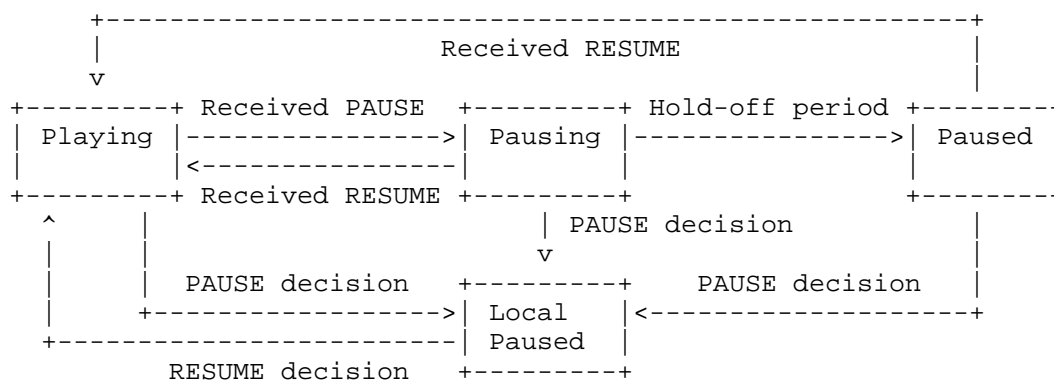


Figure 7: RTP Pause States

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 media sender has received at least one PAUSE message for the stream in question. The media sender SHALL wait during a hold-off period for the possible reception of RESUME messages for the RTP media stream being paused before actually pausing media transmission. The period to wait SHALL be long enough to allow another media 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 media 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 media 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 media 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 media sender SHALL send a PAUSED indication to all known media 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,

- o If a paused sender sends an RTCP BYE message, receivers observing this SHALL NOT send further PAUSE or RESUME requests to it.
- o 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 media stream to pause. If that receiver sends an RTCP BYE message observed by the sender, the sender SHALL resume the RTP media 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 media sender. As for Paused State (Section 7.3), the media sender SHALL send a PAUSED indication to all known media 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 media receivers that sent PAUSE for that stream, effectively clearing the media sender's memory for that media 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 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. 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:

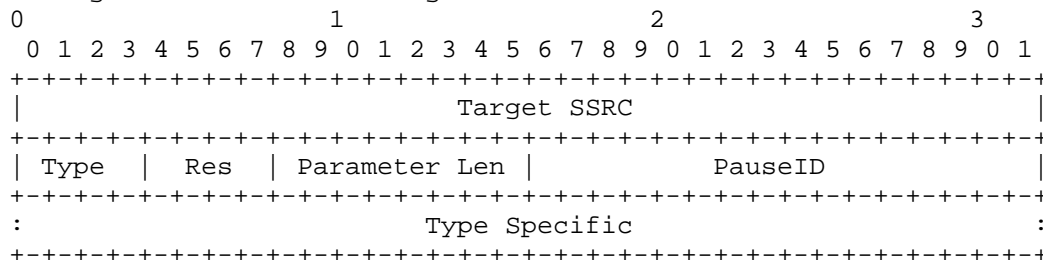


Figure 8: Syntax of FCI Entry in the PAUSE and RESUME message

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.

9.1. PAUSE

An RTP media 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 learnt from 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 media sender. The media 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 $(\text{PauseID} + 1) \bmod 2^{16}$ and $(\text{PauseID} + 2^{14}) \bmod 2^{16}$, where "MOD" is the modulo operator. Similarly, a PauseID smaller than the current one is defined as having a value between and including $(\text{PauseID} - 2^{15}) \bmod 2^{16}$ and $(\text{PauseID} - 1) \bmod 2^{16}$.

If an RTP media 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 media 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 $2 * \text{RTT} + \text{T_dither_max}$, the PAUSE MAY be scheduled for retransmission, using the same PauseID. RTT is the observed round-trip to the RTP media stream sender and T_dither_max is defined in section 3.4 of [RFC4585].

When an RTP media 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 media stream, and act accordingly.

If an RTP media 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 media 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 media 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 media 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 media 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 learnt from a returned REFUSE (Section 9.4).

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

When an RTP media 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 media 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 media stream sender sends a REFUSE message as specified below.

If an RTP media 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 media 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 media 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 media 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 media 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 media 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:

- o PAUSE SHOULD use Early or Immediate timing, except for retransmissions that SHOULD use Regular timing.
- o 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.
- o RESUME SHOULD always use Immediate or Early timing.
- o 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. Signalling

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.

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

- o "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.
- o "paused" represents the functionality of supporting the playing and local paused states and generate PAUSED FCI when a media 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 media 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.

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' media 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 media 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 media streams sent by the node. The node intends to send PAUSE and RESUME requests for other nodes' media 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 media delivery in any of its media streams. It has no need to receive PAUSED indications itself.

recvonly: The node desires to receive PAUSED indications whenever any media 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 media 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 9: 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 media delivery in any of its media 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 media direction, while the directionality of paused is the same as the media 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 signalling 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' media 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 10: 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 11: 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.

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.

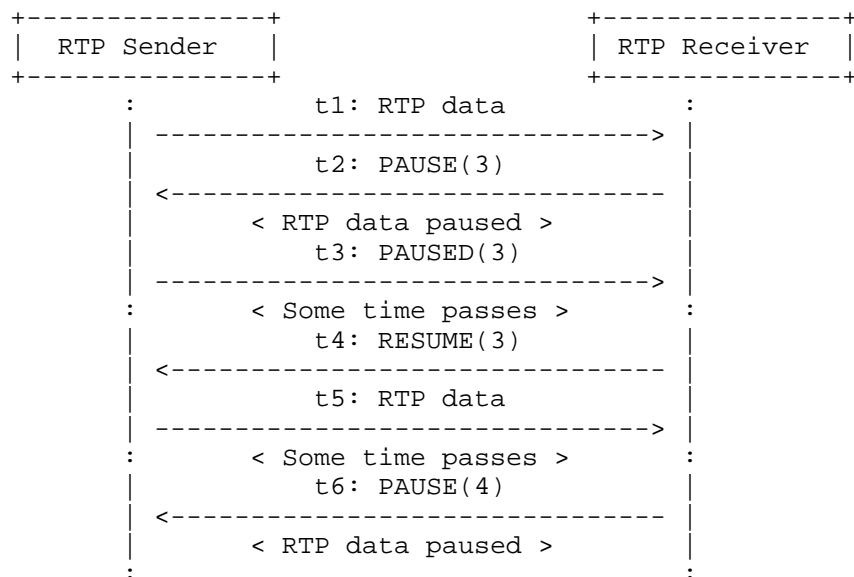


Figure 12: Pause and Resume Operation in Point-to-Point

Figure 12 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).

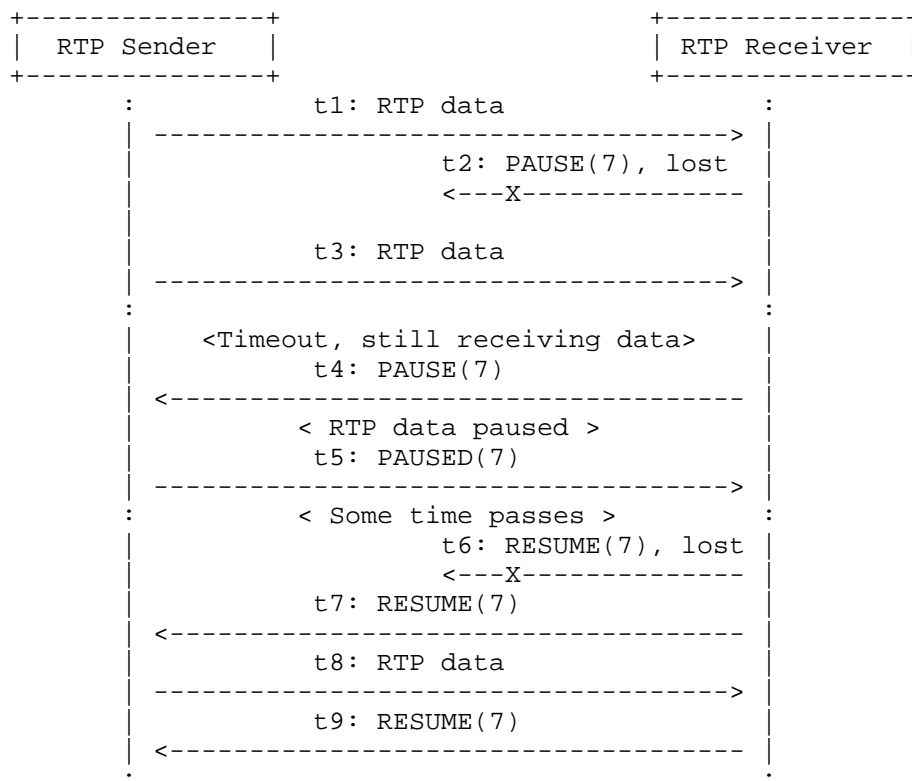


Figure 13: Pause and Resume Operation With Messages Lost

Figure 13 describes what happens if a PAUSE message from an RTP media stream receiver does not reach the RTP media stream sender. After sending a PAUSE message, the RTP media stream receiver waits for a time-out to detect if the RTP media 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 media stream receiver. When the timer expires, the RTP media stream receiver schedules a retransmission of the PAUSE message, which is sent at time t4. If the PAUSE message now reaches the RTP media stream sender, it pauses the RTP media stream and replies with PAUSED.

At time t6, the RTP media 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 media 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 media 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 media 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 media stream sender.

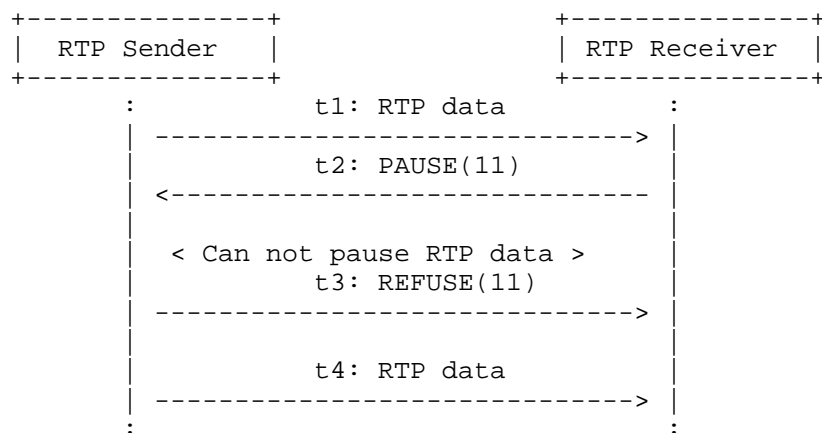


Figure 14: Pause Request is Refused in Point-to-Point

In Figure 14, 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 media 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 media 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.

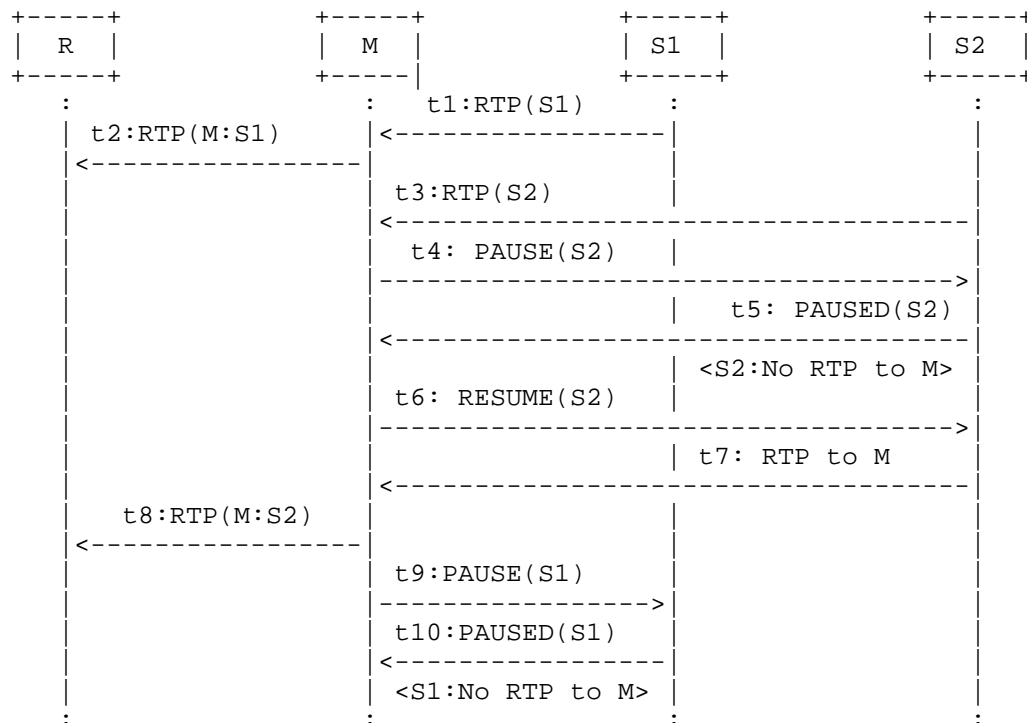


Figure 15: Pause and Resume Operation for a Voice Activated Mixer

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 media to the Mixer. As S2 has less voice activity than 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 media delivery to M. When the media

from t7 arrives at the Mixer, it switches this media 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 media 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.

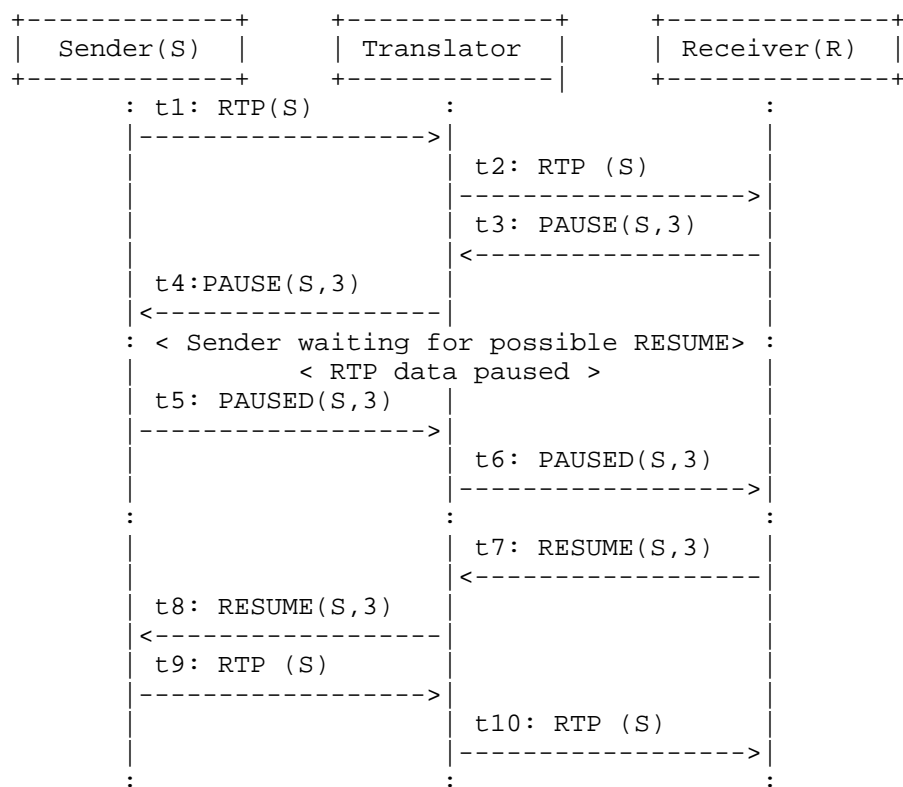


Figure 16: Pause and Resume Operation Between Two Participants Using a Translator

Figure 16 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 media sender, all messages uses a single PauseID, in this example 3.

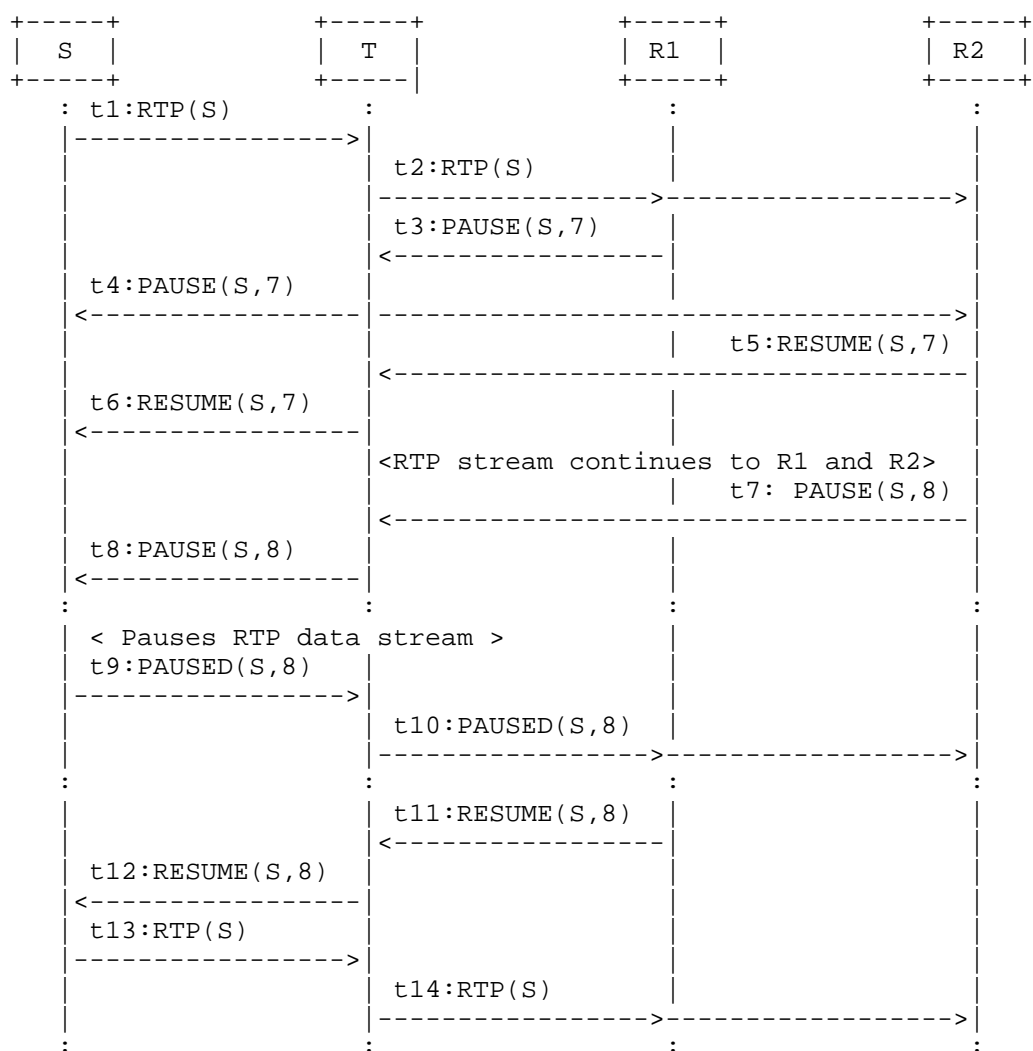


Figure 17: Pause and Resume Operation Between One Sender and Two Receivers Through Translator

Figure 17 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 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 the source streams 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.

14. Acknowledgements

15. References

15.1. Normative References

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

15.2. Informative References

- [3GPP.25.308]
3GPP, "High Speed Downlink Packet Access (HSDPA); Overall description; Stage 2", 3GPP TS 25.308 10.6.0, December 2011.
- [3GPP.26.114]
3GPP, "IP Multimedia Subsystem (IMS); Multimedia telephony; Media handling and interaction", 3GPP TS 26.114 10.4.0, June 2012.
- [3GPP.36.201]
3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA); LTE physical layer; General description", 3GPP TS 36.201 10.0.0, December 2010.
- [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-09 (work in progress), June 2012.
- [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-04 (work in progress), March 2012.
- [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 (work in progress), October 2011.
- [I-D.westerlund-avtext-codec-operation-point]
Westerlund, M., Burman, B., and L. Hamm, "Codec Operation Point RTCP Extension", draft-westerlund-avtext-codec-operation-point-00 (work in progress), March 2012.
- [I-D.westerlund-dispatch-stream-selection]
Grondal, D., Westerlund, M., and B. Burman, "Media Stream

- Selection (MESS)",
draft-westerlund-dispatch-stream-selection-00 (work in progress), October 2011.
- [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.
- [RFC3356] Fishman, G. and S. Bradner, "Internet Engineering Task Force and International Telecommunication Union - Telecommunications Standardization Sector Collaboration Guidelines", RFC 3356, August 2002.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC4582] Camarillo, G., Ott, J., and K. Drage, "The Binary Floor Control Protocol (BFCP)", RFC 4582, November 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.
- [RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, January 2008.
- [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.

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

AVTEXT Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 25, 2013

J. Xia
R. Huang
Huawei
February 21, 2013

RTP/RTCP extension for RTP Splicing Notification
draft-xia-avtext-splicing-notification-01

Abstract

Content splicing is a process that replaces the content of a main multimedia stream with other multimedia content, and delivers the substitutive multimedia content to the receivers for a period of time. The RTP mixer is designed to handle RTP splicing in [RFC6828], but how the RTP mixer knows when to start and end the splicing is still unspecified.

This memo defines two RTP/RTCP extensions to indicate the splicing related information to the RTP mixer: an RTP header extension that conveys the information in-band and an RTCP packet that conveys the information out-of-band.

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 August 23, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. Overview of RTP Splicing Notification	4
4. Conveying Splicing Metadata in RTP/RTCP extensions	5
4.1. RTP Header Extension	5
4.2. RTCP Splicing Notification Message	6
5. Reducing Splicing Latency	7
6. Failure Cases	8
7. SDP Signaling	8
8. Security Considerations	9
9. IANA considerations	10
9.1. SDP Attribute Registration	10
9.2. RTCP Control Packet Types	10
9.3. RTP Compact Header Extensions	10
10. Acknowledgments	11
11. References	11
11.1. Normative References	11
11.2. Informative References	11
Authors' Addresses	12

1. Introduction

Splicing is a process that replaces some multimedia content with other multimedia content and delivers the substitutive multimedia content to the receivers during specific, pre-designated time slot. Certain timing information about when to start and end the splicing must be first acquired by the splicer to start the splicing. This document refers to this information as Splicing Metadata.

[SCTE35] provides a method that encapsulates the Splicing Metadata inside the MPEG2-TS layer in cable TV systems. But in RTP splicing scenario described in [RFC6828], the mixer has to decode the RTP packets, search and solve the Splicing Metadata inside the payloads. The need for such processing enhances the workload of the mixer and limits the size of RTP sessions the mixer can support.

The document defines an RTP header extension [RFC5285] through which the main RTP sender can provide the Splicing Metadata by including it in the RTP packets.

Nevertheless, the Splicing Metadata conveyed in the RTP header extension might not reach the mixer successfully, any splicing unaware middlebox on the path between the RTP sender and the mixer might strip the RTP header extension.

To increase robustness against above case, the document also defines a new RTCP packet type in a complementary fashion to carry the Splicing Metadata to the mixer even though RTCP is inherently unreliable too.

2. Terminology

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

Most terminology defined in "Content Splicing for RTP Sessions" [RFC6828] applies to this document except the following one:

Splicing Metadata

A set of certain metadata that allows the mixer to know when to start and end the RTP splicing. The information consists of a couple of NTP-format timestamps on the splicing in point and on the splicing out point.

3. Overview of RTP Splicing Notification

According to RTP Splicing draft [RFC6828], a mixer is designed to do splicing on the RTP layer, but it cannot insert the substitutive content randomly but only do that at the reserved time slots set by the main RTP sender. This implies the mixer must first know the Splicing Metadata from the main RTP sender before splicing starts.

When a new splicing is forthcoming, the main RTP sender **MUST** send the Splicing Metadata to the mixer. Usually, the Splicing Metadata **SHOULD** be sent more than once to against the possible packet loss. To enable the mixer to get the substitutive content before the splicing starts, the main RTP sender **MUST** send the Splicing Metadata far enough in advance. Alternatively, the main RTP sender can estimate when to send the Splicing Metadata based on the round-trip time (RTT) following the mechanisms in section 6.4.1 of [RFC3550] when the mixer sends RTCP RR to the main sender.

The substitutive sender also needs to learn the Splicing Metadata from the main RTP sender in advance, and thus estimates when to transfer the substitutive content to the mixer. The Splicing Metadata could be transmitted from the main RTP sender to the substitutive content using some out-of-band mechanisms, the details how to achieve that are beyond the scope of this memo. To ensure the Splicing Metadata is valid to the main RTP sender and the substitutive RTP sender, the two senders **MUST** share a common reference clock, so the mixer can achieve accurate splicing.

In this document, the main RTP sender uses a couple of NTP-format timestamps, derived from the common reference clock, to indicate when to start and end the splicing to the mixer: the timestamp of the first substitutive RTP packet on the splicing in point, followed by the timestamp of the first main RTP packet on the splicing out point.

When the substitutive RTP sender gets the Splicing Metadata, it must prepare the substitutive stream. The RTP timestamp of the first substitutive RTP packet that would be presented on the receivers **MUST** correspond to the same time instant as the former NTP timestamp in the Splicing Metadata. To enable mixer to know the first substitutive RTP packet it begins to output, the substitutive RTP sender **MUST** enable the mixer to know above RTP timestamp in advance, e.g., from prior receipt of RTCP SR message.

When the splicing will end, the RTP timestamp of the first main RTP packet that would be presented on the receivers **MUST** correspond to the same time instant as the latter NTP timestamp in the Splicing Metadata.

4. Conveying Splicing Metadata in RTP/RTCP extensions

This memo defines two backwards compatible RTP extensions to convey the Splicing Metadata to the mixer: an RTP header extension and an RTCP splicing notification message.

4.1. RTP Header Extension

The RTP header extension mechanism defined in [RFC5285] can be adapted to carry the Splicing Metadata consisting of a couple of NTP-format timestamps.

One variant is defined for this header extension. The variant carries the lower 24-bit part of the Seconds of a NTP-format timestamp and the 32 bits of the Fraction of a NTP-format timestamp as defined in [RFC5905]. The format is shown in Figures 1.

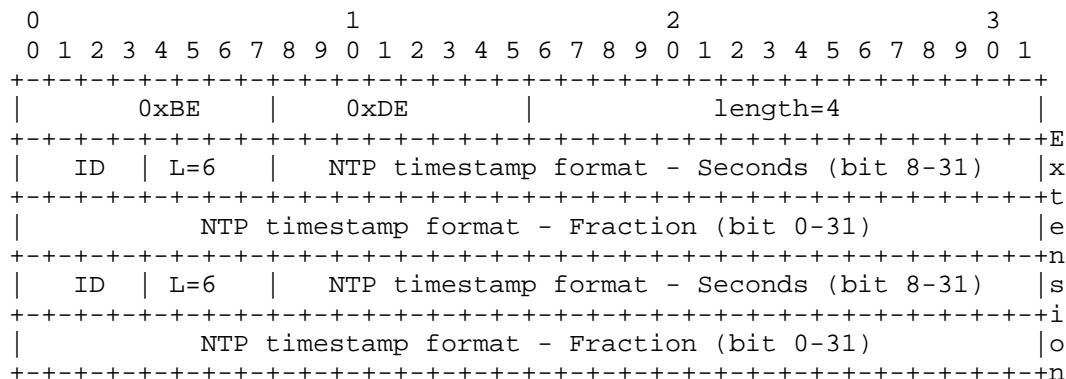


Figure 1: Sample 56-Bit NTP Encoding
Using the One-Byte Header Format

Note that the inclusion of an RTP header extension will reduce the efficiency of RTP header compression. It is RECOMMENDED that the main sender begins to insert the RTP header extensions into a number of RTP packets in advance of the splicing starting, while leaving the remain RTP packets unmarked.

After the mixer intercepts the RTP header extension and derives the Splicing Metadata, it will generate its own stream and could not include the RTP header extension in outgoing packets to reduce header overhead.

Furthermore, whether the in-band NTP-format timestamps are included or not, RTCP splicing notification message in next section MUST be

sent to provide robustness in the case of any splicing-unaware middlebox that might strip RTP header extensions.

When SDP is used, the use of the RTP header extensions defined above MUST be indicated as specified in [RFC5285]. Therefore, the following URI MUST be used:

- o 1 The URI used for signalling the use of Variant B/56-bit NTP RTP header extension in SDP is "urn:ietf:params:rtp-hdrex: splicing-metadata-56".

4.2. RTCP Splicing Notification Message

Besides the RTP header extension, the main RTP sender includes the Splicing Metadata in an RTCP splicing notification message.

The RTCP splicing notification message is a new RTCP packet type defined with PT = 211. It has a fix header followed by a couple of NTP-format timestamps:

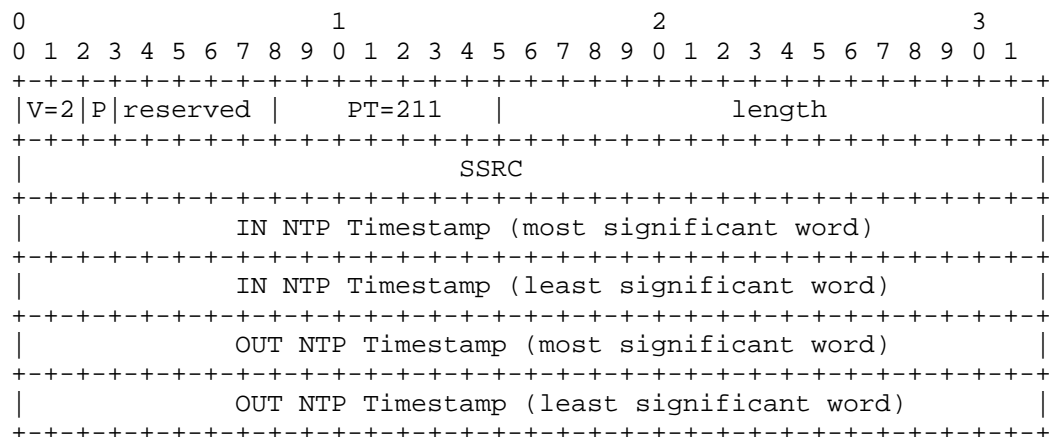


Figure 2: RTCP Splicing Notification Message

The RSI packet includes the following fields:

Length: 16 bits

As defined in [RFC3550], the length of the RTCP packet in 32-bit words minus one, including the header and any padding.

SSRC: 32 bits

The SSRC of the Main RTP Sender.

Timestamp: 64 bits

Indicates the wallclock time when this splicing starts and ends. The full-resolution NTP timestamp is used, which is a 64-bit, unsigned, fixed-point number with the integer part in the first 32 bits and the fractional part in the last 32 bits. This format is similar to RTCP Sender Report (Section 6.4.1 of [RFC3550]).

A basic assumption is that the main RTP sender and the mixers are in an SSM group with the source being the main RTP sender. RTCP splicing notification message from the main RTP sender is sent via multicast to the mixers.

The RTCP splicing notification message can be appended to RTCP SR the main RTP sender generates in compound RTCP packets, and hence follows the compound RTCP rules defined in Section 6.1 in [RFC3550].

If the use of non-compound RTCP [RFC5506] was previously negotiated between the sender and the mixer, the RTCP splicing notification message may be sent as non-compound RTCP packets.

When the mixer intercepts the RTCP splicing notification message, it MAY NOT forward the message to the receivers in order to reduce RTCP bandwidth consumption or to avoid downstream receivers from detecting splicing defined in Section 4.5 in [RFC6828].

5. Reducing Splicing Latency

When splicing starts or ends, the mixer outputs the multimedia content from another sender to the receivers. Given that the receivers must first acquire certain information ([RFC6285] refers to this information as Reference Information) to start processing the multimedia data, either the main RTP sender or the substitutive sender SHOULD provide the Reference Information align with its multimedia content to reduce the delay caused by acquiring the Reference Information. The means by which the Reference Information is distributed to the receivers is out of scope of this memo.

Another latency element is synchronization caused delay. The receivers must receive enough synchronization metadata prior to synchronizing the separate components of the multimedia streams when splicing starts or ends. Either the main RTP sender or the

substitutive sender SHOULD send the synchronization metadata early enough so that the receivers can play out the multimedia in a synchronized fashion. The mechanisms defined in [RFC6051] are RECOMMENDED to be adopted to reduce the possible synchronization delay.

6. Failure Cases

This section examines the implications of losing RTCP splicing notification message and other failure case, e.g., the RTP header extension is stripped on the path.

Given there may be splicing un-aware middlebox on the path between the main RTP sender and the mixer, one heuristic will be used to verify whether or not the Splicing Metadata reaches the mixers.

If the mixer does not get the Splicing Metadata when the splicing starts, it will still output the main content to the downstream receivers and forward the RTCP RR packets sent from downstream receivers to the main RTP sender. In such case, the main RTP sender can learn the splicing failed.

In a similar manner, the substitutive sender can learn the splicing failed if it does not receive any RTCP RR packets from downstream receivers when the splicing starts.

Upon the detection of a failure, the main RTP sender or the substitutive sender SHOULD check the path to the failed mixer, or fallback to the payload specific mechanisms, e.g., MPEG-TS splicing solution defined in [SCTE35].

7. SDP Signaling

This document defines the "rtp-splicing" media attribute, which is used for indicating whether the main RTP sender can provide time-slots within its RTP flows for RTP splicing. This attribute should be used in a declarative manner. If this attribute is included in the SDP description, the mixer can receive the Splicing Metadata and implement RTP splicing.

This document also reuses the Flow Identification (FID) semantics defined in SDP Grouping Framework [RFC5888] to represent the relationship between the main RTP stream and the substitutive RTP stream.

The next example shows how the "group" attribute used with FID

semantics can indicate RTP splicing support on RTP sender.

```
v=0
o=xia 1122334455 1122334466 IN IP4 splicing.example.com
s=RTP Splicing Example
t=0 0
a=group:FID 1 2
m=video 30000 RTP/AVP 100
i=Main RTP Stream
c=IN IP4 233.252.0.1/127
a=rtpmap:100 MP2T/90000
a=rtp-splicing
a=mid: 1
m= video 30001 RTP/AVP 100
i=Substitutive RTP Stream
c=IN IP4 233.252.0.2/127
a=sendonly
a=mid: 2
```

Figure 3: Example SDP for a single-channel splicing scenario

The mixer receiving the SDP message above receives one MPEG2-TS stream (payload 100) from the main RTP sender (with multicast destination address of 233.252.0.1) on port 30000, and/or receives another MPEG2-TS stream from the substitutive RTP sender (with multicast destination address of 233.252.0.2) on port 30001. But at a particular point in time, the mixer only selects one stream and output the content from the chosen stream to the downstream receivers.

8. Security Considerations

The security considerations of the RTP specification [RFC3550], the general mechanism for RTP header extensions [RFC5285] and the security considerations of the RTP splicing specification [RFC6828] apply.

The RTP header extension defined in Section 4.1 include two NTP-format timestamps. In the Secure Real-time Transport Protocol (SRTP)[RFC3711], RTP header extensions are authenticated but not encrypted. A malicious endpoint could choose to set the values in this header extension falsely, so as to falsely claim the splicing time.

In scenarios where this is a concern, additional mechanisms MUST be used to protect the confidentiality of the header extension. This

mechanism could be header extension encryption [SRTP-ENCR-HDR], or a lower-level security and authentication mechanism such as IPsec [RFC4301].

9. IANA considerations

9.1. SDP Attribute Registration

Following the guidelines in [RFC4566], the IANA is requested to register one new SDP attribute:

- o Contact name, email address and telephone number: Authors of RFCXXXX
- o Attribute-name: rtp-splicing
- o Long-form: Support RTP Splicing
- o Type of attribute: media-level
- o Subject to charset: no

This attribute is used to signal that the main RTP stream can provide time slots for RTP splicing. It is a property attribute, which does not take a value.

9.2. RTCP Control Packet Types

Based on the guidelines suggested in [RFC5226], a new RTCP packet format has been registered with the RTCP Control Packet Type (PT) Registry:

Name: SNM

Long name: Splicing Notification Message

Value: TBA1

Reference: This document

9.3. RTP Compact Header Extensions

The IANA has also registered a new RTP Compact Header Extension [RFC5285], according to the following:

Extension URI: urn:ietf:params:rtp-hdext:splicing-metadata-56

Description: Splicing metadata: 56-bit timestamp format

Contact: Jinwei Xia <xiajinwei@huawei.com>

Reference: This document

10. Acknowledgments

TBD

11. References

11.1. 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.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, July 2008.
- [RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.
- [RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.
- [RFC6051] Perkins, C. and T. Schierl, "Rapid Synchronisation of RTP Flows", RFC 6051, November 2010.
- [RFC6828] Xia, J., "Content Splicing for RTP Sessions", RFC 6828, January 2013.

11.2. Informative References

- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)",

RFC 3711, March 2004.

- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.
- [RFC6285] Ver Steeg, B., Begen, A., Van Caenegem, T., and Z. Vax, "Unicast-Based Rapid Acquisition of Multicast RTP Sessions", RFC 6285, June 2011.
- [SRTP-ENCR-HDR] Lennox, J., "Encryption of Header Extensions in the Secure Real-Time Transport Protocol (SRTP)", draft-ietf-avtcore-srtp-encrypted-header-ext-04 (work in progress), January 2013.
- [SCTE35] Society of Cable Telecommunications Engineers (SCTE), "Digital Program Insertion Cueing Message for Cable", 2011.

Authors' Addresses

Jinwei Xia
Huawei
101 Software Avenue
Nanjing, Yuhua District 210012
China

Phone: +86-025-84565890
Email: xiajinwei@huawei.com

Rachel Huang
Huawei
101 Software Avenue
Nanjing, Yuhua District 210012
China

Phone: +86-025-84565890
Email: rachel.huang@huawei.com

