

Network Working Group
Internet-Draft
Intended status: Informational
Expires: December 29, 2014

J. Lennox
Vidyo
K. Gross
AVA
S. Nandakumar
G. Salgueiro
Cisco Systems
B. Burman
Ericsson
June 27, 2014

A Taxonomy of Grouping Semantics and Mechanisms for Real-Time Transport
Protocol (RTP) Sources
draft-ietf-avtext-rtp-grouping-taxonomy-02

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.

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 December 29, 2014.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

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

Table of Contents

1. Introduction	4
2. Concepts	4
2.1. Media Chain	4
2.1.1. Physical Stimulus	8
2.1.2. Media Capture	8
2.1.3. Raw Stream	8
2.1.4. Media Source	8
2.1.5. Source Stream	9
2.1.6. Media Encoder	9
2.1.7. Encoded Stream	10
2.1.8. Dependent Stream	11
2.1.9. Media Packetizer	11
2.1.10. RTP Stream	11
2.1.11. Media Redundancy	12
2.1.12. Redundancy RTP Stream	12
2.1.13. Media Transport	13
2.1.14. Media Transport Sender	14
2.1.15. Sent RTP Stream	14
2.1.16. Network Transport	14
2.1.17. Transported RTP Stream	14
2.1.18. Media Transport Receiver	14
2.1.19. Received RTP Stream	15
2.1.20. Received Redundancy RTP Stream	15
2.1.21. Media Repair	15
2.1.22. Repaired RTP Stream	15
2.1.23. Media Depacketizer	15
2.1.24. Received Encoded Stream	16
2.1.25. Media Decoder	16
2.1.26. Received Source Stream	16
2.1.27. Media Sink	16
2.1.28. Received Raw Stream	17
2.1.29. Media Render	17
2.2. Communication Entities	17
2.2.1. End Point	18
2.2.2. RTP Session	18
2.2.3. Participant	19
2.2.4. Multimedia Session	20
2.2.5. Communication Session	20

3.	Relations at Different Levels	21
3.1.	Synchronization Context	22
3.1.1.	RTCP CNAME	22
3.1.2.	Clock Source Signaling	22
3.1.3.	Implicitly via RtcMediaStream	22
3.1.4.	Explicitly via SDP Mechanisms	22
3.2.	End Point	22
3.3.	Participant	23
3.4.	RtcMediaStream	23
3.5.	Single- and Multi-Session Transmission of SVC	23
3.6.	Multi-Channel Audio	24
3.7.	Simulcast	24
3.8.	Layered Multi-Stream	25
3.9.	RTP Stream Duplication	27
3.10.	Redundancy Format	27
3.11.	RTP Retransmission	28
3.12.	Forward Error Correction	29
3.13.	RTP Stream Separation	31
3.14.	Multiple RTP Sessions over one Media Transport	32
4.	Mapping from Existing Terms	32
4.1.	Audio Capture	32
4.2.	Capture Device	32
4.3.	Capture Encoding	32
4.4.	Capture Scene	33
4.5.	Endpoint	33
4.6.	Individual Encoding	33
4.7.	Multipoint Control Unit (MCU)	33
4.8.	Media Capture	33
4.9.	Media Consumer	33
4.10.	Media Description	33
4.11.	Media Provider	34
4.12.	Media Stream	34
4.13.	Multimedia Session	34
4.14.	Recording Device	34
4.15.	RtcMediaStream	34
4.16.	RtcMediaStreamTrack	35
4.17.	RTP Sender	35
4.18.	RTP Session	35
4.19.	SSRC	35
4.20.	Stream	35
4.21.	Video Capture	35
5.	Security Considerations	35
6.	Acknowledgement	36
7.	Contributors	36
8.	IANA Considerations	36
9.	Informative References	36
Appendix A.	Changes From Earlier Versions	38
A.1.	Modifications Between WG Version -01 and -02	38

A.2. Modifications Between WG Version -00 and -01	39
A.3. Modifications Between Version -02 and -03	40
A.4. Modifications Between Version -01 and -02	40
A.5. Modifications Between Version -00 and -01	40
Authors' Addresses	40

1. Introduction

The existing taxonomy of sources in RTP is often regarded as confusing and inconsistent. Consequently, a deep understanding of how the different terms relate to each other becomes a real challenge. Frequently cited examples of this confusion are (1) how different protocols that make use of RTP use the same terms to signify different things and (2) how the 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.

All non-specific references to ControLling mUltiple streams for tElepresence (CLUE) in this document map to [I-D.ietf-clue-framework] and all references to Web Real-Time Communications (WebRTC) map to [I-D.ietf-rtcweb-overview].

2. Concepts

This section defines concepts that serve to identify and name various transformations and streams in a given RTP usage. For each concept an attempt is made to list any alternate definitions and usages that co-exist today along with various characteristics that further describes the concept. These concepts are divided into two categories, one related to the chain of streams and transformations that media can be subject to, the other for entities involved in the communication.

2.1. Media Chain

In the context of this memo, Media is a sequence of synthetic or Physical Stimulus (Section 2.1.1) (sound waves, photons, key-strokes), represented in digital form. Synthesized Media is typically generated directly in the digital domain.

This section contains the concepts that can be involved in taking Media at a sender side and transporting it to a receiver, which may recover a sequence of physical stimulus. This chain of concepts is of two main types, streams and transformations. Streams are time-

based sequences of samples of the physical stimulus in various representations, while transformations changes the representation of the streams in some way.

The below examples are basic ones and it is important to keep in mind that this conceptual model enables more complex usages. Some will be further discussed in later sections of this document. In general the following applies to this model:

- o A transformation may have zero or more inputs and one or more outputs.
- o A stream is of some type.
- o A stream has one source transformation and one or more sink transformations (with the exception of Physical Stimulus (Section 2.1.1) that may lack source or sink transformation).
- o Streams can be forwarded from a transformation output to any number of inputs on other transformations that support that type.
- o If the output of a transformation is sent to multiple transformations, those streams will be identical; it takes a transformation to make them different.
- o There are no formal limitations on how streams are connected to transformations, this may include loops if required by a particular transformation.

It is also important to remember that this is a conceptual model. Thus real-world implementations may look different and have different structure.

To provide a basic understanding of the relationships in the chain we below first introduce the concepts for the sender side (Figure 1). This covers physical stimulus until media packets are emitted onto the network.

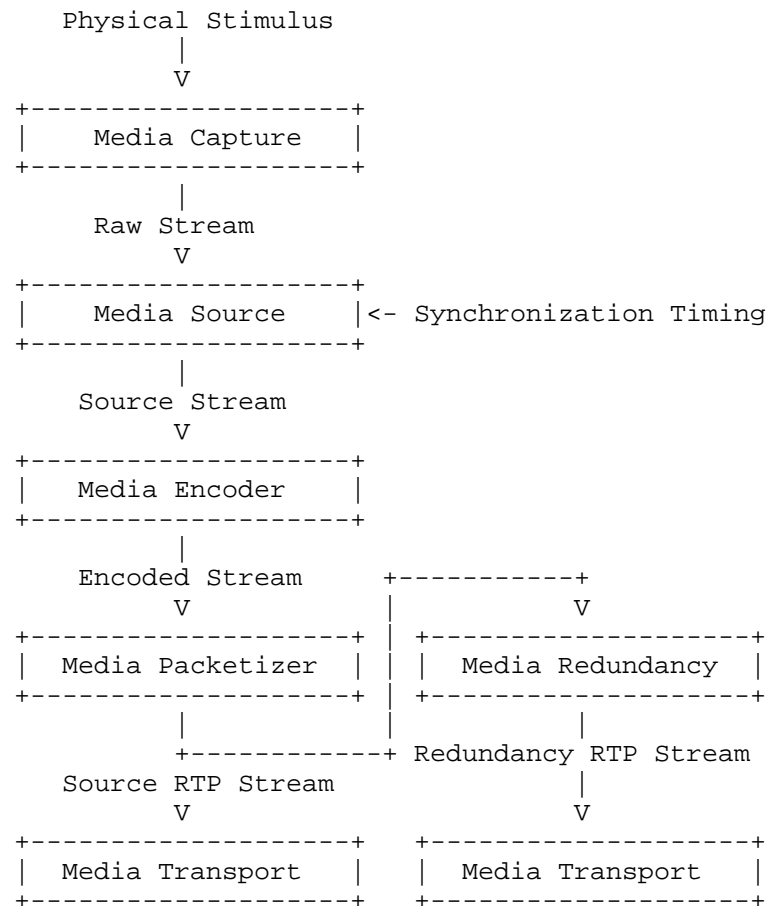


Figure 1: Sender Side Concepts in the Media Chain

In Figure 1 we have included a branched chain to cover the concepts for using redundancy to improve the reliability of the transport. The Media Transport concept is an aggregate that is decomposed below in Section 2.1.13.

Below we review a receiver media chain (Figure 2) matching the sender side to look at the inverse transformations and their attempts to recover possibly identical streams as in the sender chain. Note that the streams out of a reverse transformation, like the Source Stream out the Media Decoder are in many cases not the same as the corresponding ones on the sender side, thus they are prefixed with a "Received" to denote a potentially modified version. The reason for not being the same lies in the transformations that can be of irreversible type. For example, lossy source coding in the Media

Encoder prevents the Source Stream out of the Media Decoder to be the same as the one fed into the Media Encoder. Other reasons include packet loss or late loss in the Media Transport transformation that even Media Repair, if used, fails to repair. It should be noted that some transformations are not always present, like Media Repair that cannot operate without Redundancy RTP Streams.

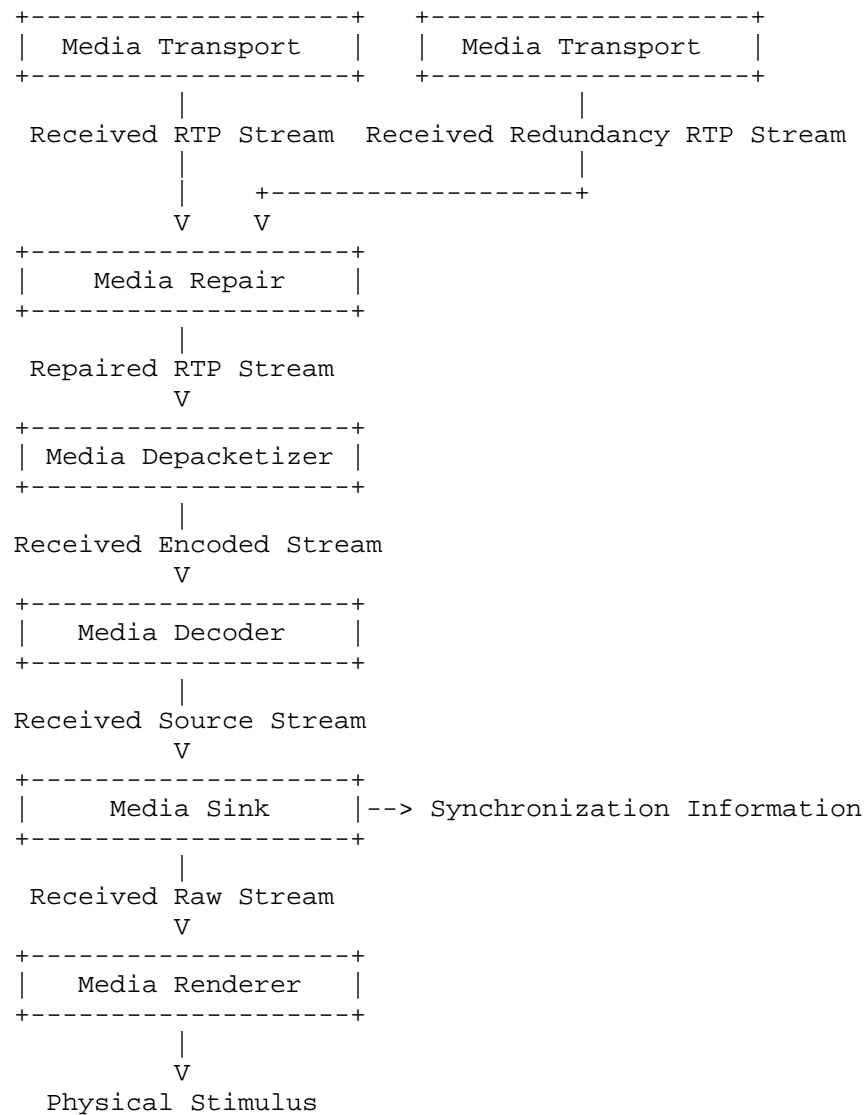


Figure 2: Receiver Side Concepts of the Media Chain

2.1.1. Physical Stimulus

The physical stimulus is a physical event that can be measured and converted to digital form by an appropriate sensor or transducer. This include sound waves making up audio, photons in a light field that is visible, or other excitations or interactions with sensors, like keystrokes on a keyboard.

2.1.2. Media Capture

Media Capture is the process of transforming the Physical Stimulus (Section 2.1.1) into digital Media using an appropriate sensor or transducer. The Media Capture performs a digital sampling of the physical stimulus, usually periodically, and outputs this in some representation as a Raw Stream (Section 2.1.3). This data is due to its periodical sampling, or at least being timed asynchronous events, some form of a stream of media data. The Media Capture is normally instantiated in some type of device, i.e. media capture device. Examples of different types of media capturing devices are digital cameras, microphones connected to A/D converters, or keyboards.

Characteristics:

- o A Media Capture is identified either by hardware/manufacture ID or via a session-scoped device identifier as mandated by the application usage.
- o A Media Capture can generate an Encoded Stream (Section 2.1.7) if the capture device support such a configuration.

2.1.3. Raw Stream

The time progressing stream of digitally sampled information, usually periodically sampled and provided by a Media Capture (Section 2.1.2). A Raw Stream can also contain synthesized Media that may not require any explicit Media Capture, since it is already in an appropriate digital form.

2.1.4. Media Source

A Media Source is the logical source of a reference clock synchronized, time progressing, digital media stream, called a Source Stream (Section 2.1.5). This transformation takes one or more Raw Streams (Section 2.1.3) and provides a Source Stream as output. This output has been synchronized with some reference clock, even if just a system local wall clock.

The output can be of different types. One type is directly associated with a particular Media Capture's Raw Stream. Others are more conceptual sources, like an audio mix of multiple Raw Streams (Figure 3), a mixed selection of the three loudest inputs regarding speech activity, a selection of a particular video based on the current speaker, i.e. typically based on other Media Sources.

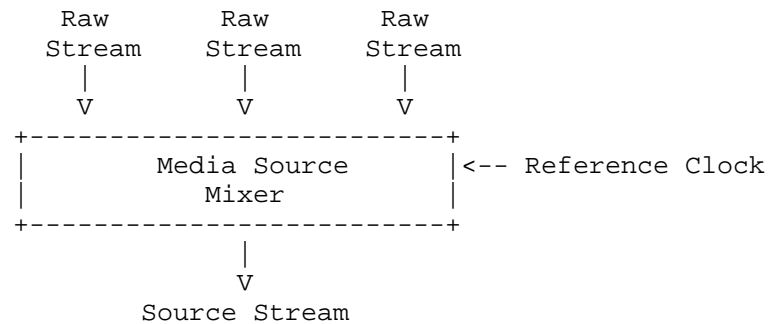


Figure 3: Conceptual Media Source in form of Audio Mixer

Characteristics:

- o At any point, it can represent a physical captured source or conceptual source.

2.1.5. Source Stream

A time progressing stream of digital samples that has been synchronized with a reference clock and comes from particular Media Source (Section 2.1.4).

2.1.6. Media Encoder

A Media Encoder is a transform that is responsible for encoding the media data from a Source Stream (Section 2.1.5) into another representation, usually more compact, that is output as an Encoded Stream (Section 2.1.7).

The Media Encoder step commonly includes pre-encoding transformations, such as scaling, resampling etc. The Media Encoder can have a significant number of configuration options that affects the properties of the encoded stream. This include properties such as bit-rate, start points for decoding, resolution, bandwidth or other fidelity affecting properties. The actually used codec is also an important factor in many communication systems, not only its parameters.

Scalable Media Encoders need special mentioning as they produce multiple outputs that are potentially of different types. A scalable Media Encoder takes one input Source Stream and encodes it into multiple output streams of two different types; at least one Encoded Stream that is independently decodable and one or more Dependent Streams (Section 2.1.8) that requires at least one Encoded Stream and zero or more Dependent Streams to be possible to decode. A Dependent Stream's dependency is one of the grouping relations this document discusses further in Section 3.8.

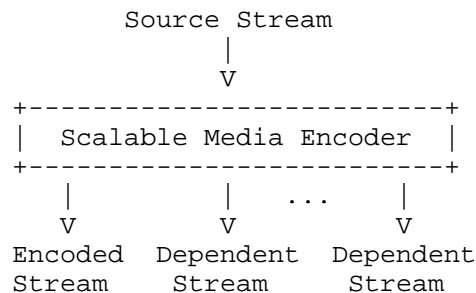


Figure 4: Scalable Media Encoder Input and Outputs

There are also other variants of encoders, like so-called Multiple Description Coding (MDC). Such Media Encoder produce multiple independent and thus individually decodable Encoded Streams that are possible to combine into a Received Source Stream that is somehow a better representation of the original Source Stream than using only a single Encoded Stream.

Characteristics:

- o A Media Source can be multiply encoded by different Media Encoders to provide various encoded representations.

2.1.1.7. Encoded Stream

A stream of time synchronized encoded media that can be independently decoded.

Characteristics:

- o Due to temporal dependencies, an Encoded Stream may have limitations in where decoding can be started. These entry points, for example Intra frames from a video encoder, may require identification and their generation may be event based or configured to occur periodically.

2.1.1.8. Dependent Stream

A stream of time synchronized encoded media fragments that are dependent on one or more Encoded Streams (Section 2.1.7) and zero or more Dependent Streams to be possible to decode.

Characteristics:

- o Each Dependent Stream has a set of dependencies. These dependencies must be understood by the parties in a multi-media session that intend to use a Dependent Stream.

2.1.1.9. Media Packetizer

The transformation of taking one or more Encoded (Section 2.1.7) or Dependent Stream (Section 2.1.8) and put their content into one or more sequences of packets, normally RTP packets, and output Source RTP Streams (Section 2.1.10). This step includes both generating RTP payloads as well as RTP packets.

The Media Packetizer can use multiple inputs when producing a single RTP Stream. One such example is SST packetization when using SVC (Section 3.5).

The Media Packetizer can also produce multiple RTP Streams, for example when Encoded and/or Dependent Streams are distributed over multiple RTP Streams. One example of this is MST packetization when using SVC (Section 3.5).

Characteristics:

- o The Media Packetizer will select which Synchronization source(s) (SSRC) [RFC3550] in which RTP sessions that are used.
- o Media Packetizer can combine multiple Encoded or Dependent Streams into one or more RTP Streams.

2.1.1.10. RTP Stream

A stream of RTP packets containing media data, source or redundant. The RTP Stream is identified by an SSRC belonging to a particular RTP session. The RTP session is identified as discussed in Section 2.2.2.

A Source RTP Stream is a RTP Stream containing at least some content from an Encoded Stream. Source material is any media material that is produced for transport over RTP without any additional redundancy

applied to cope with network transport losses. Compare this with the Redundancy RTP Stream (Section 2.1.12).

Characteristics:

- o Each RTP Stream is identified by a unique Synchronization source (SSRC) [RFC3550] that is carried in every RTP and RTP Control Protocol (RTCP) packet header in a specific RTP session context.
- o At any given point in time, a RTP Stream can have one and only one SSRC. SSRC collision and clock rate change [RFC7160] are examples of valid reasons to change SSRC for a RTP Stream, since the RTP Stream itself is not changed in any significant way, only the identifying SSRC number.
- o Each RTP Stream defines a unique RTP sequence numbering and timing space.
- o Several RTP Streams may map to a single Media Source via the source transformations.
- o Several RTP Streams can be carried over a single RTP Session.

2.1.11. Media Redundancy

Media redundancy is a transformation that generates redundant or repair packets sent out as a Redundancy RTP Stream to mitigate network transport impairments, like packet loss and delay.

The Media Redundancy exists in many flavors; they may be generating independent Repair Streams that are used in addition to the Source Stream (RTP Retransmission [RFC4588] and some FEC [RFC5109]), they may generate a new Source Stream by combining redundancy information with source information (Using XOR FEC [RFC5109] as a redundancy payload [RFC2198]), or completely replace the source information with only redundancy packets.

2.1.12. Redundancy RTP Stream

A RTP Stream (Section 2.1.10) that contains no original source data, only redundant data that may be combined with one or more Received RTP Stream (Section 2.1.19) to produce Repaired RTP Streams (Section 2.1.22).

2.1.13. Media Transport

A Media Transport defines the transformation that the RTP Streams (Section 2.1.10) are subjected to by the end-to-end transport from one RTP sender to one specific RTP receiver (an RTP session may contain multiple RTP receivers per sender). Each Media Transport is defined by a transport association that is identified by a 5-tuple (source address, source port, destination address, destination port, transport protocol). Each transport association normally contains only a single RTP session, although a proposal exists for sending multiple RTP sessions over one transport association [I-D.westerlund-avtcore-transport-multiplexing].

Characteristics:

- o Media Transport transmits RTP Streams of RTP Packets from a source transport address to a destination transport address.

The Media Transport concept sometimes needs to be decomposed into more steps to enable discussion of what a sender emits that gets transformed by the network before it is received by the receiver. Thus we provide also this Media Transport decomposition (Figure 5).

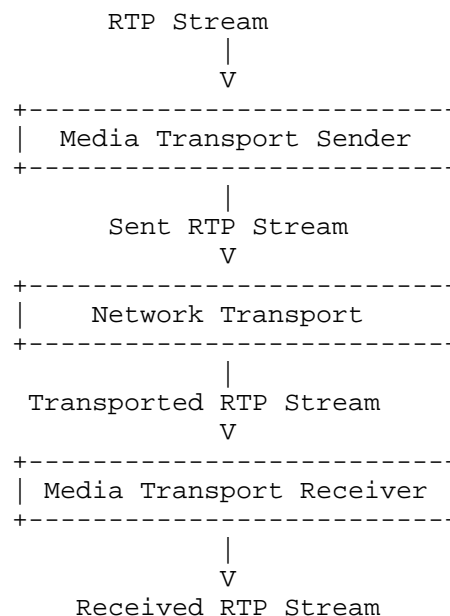


Figure 5: Decomposition of Media Transport

2.1.14. Media Transport Sender

The first transformation within the Media Transport (Section 2.1.13) is the Media Transport Sender, where the sending End-Point (Section 2.2.1) takes a RTP Stream and emits the packets onto the network using the transport association established for this Media Transport thus creating a Sent RTP Stream (Section 2.1.15). In this process it transforms the RTP Stream in several ways. First, it gains the necessary protocol headers for the transport association, for example IP and UDP headers, thus forming IP/UDP/RTP packets. In addition, the Media Transport Sender may queue, pace or otherwise affect how the packets are emitted onto the network. Thus adding delay, jitter and inter packet spacings that characterize the Sent RTP Stream.

2.1.15. Sent RTP Stream

The Sent RTP Stream is the RTP Stream as entering the first hop of the network path to its destination. The Sent RTP Stream is identified using network transport addresses, like for IP/UDP the 5-tuple (source IP address, source port, destination IP address, destination port, and protocol (UDP)).

2.1.16. Network Transport

Network Transport is the transformation that the Sent RTP Stream (Section 2.1.15) is subjected to by traveling from the source to the destination through the network. These transformations include, loss of some packets, varying delay on a per packet basis, packet duplication, and packet header or data corruption. These transformations produces a Transported RTP Stream (Section 2.1.17) at the exit of the network path.

2.1.17. Transported RTP Stream

The RTP Stream that is emitted out of the network path at the destination, subjected to the Network Transport's transformation (Section 2.1.16).

2.1.18. Media Transport Receiver

The receiver End-Point's (Section 2.2.1) transformation of the Transported RTP Stream (Section 2.1.17) by its reception process that result in the Received RTP Stream (Section 2.1.19). This transformation includes transport checksums being verified and if non-matching, causing discarding of the corrupted packet. Other transformations can include delay variations in receiving a packet on the network interface and providing it to the application.

2.1.19. Received RTP Stream

The RTP Stream (Section 2.1.10) resulting from the Media Transport's transformation, i.e. subjected to packet loss, packet corruption, packet duplication and varying transmission delay from sender to receiver.

2.1.20. Received Redundancy RTP Stream

The Redundancy RTP Stream (Section 2.1.12) resulting from the Media Transport transformation, i.e. subjected to packet loss, packet corruption, and varying transmission delay from sender to receiver.

2.1.21. Media Repair

A Transformation that takes as input one or more Source RTP Streams (Section 2.1.10) as well as Redundancy RTP Streams (Section 2.1.12) and attempts to combine them to counter the transformations introduced by the Media Transport (Section 2.1.13) to minimize the difference between the Source Stream (Section 2.1.5) and the Received Source Stream (Section 2.1.26) after Media Decoder (Section 2.1.25). The output is a Repaired RTP Stream (Section 2.1.22).

2.1.22. Repaired RTP Stream

A Received RTP Stream (Section 2.1.19) for which Received Redundancy RTP Stream (Section 2.1.20) information has been used to try to re-create the RTP Stream (Section 2.1.10) as it was before Media Transport (Section 2.1.13).

2.1.23. Media Depacketizer

A Media Depacketizer takes one or more RTP Streams (Section 2.1.10) and depacketizes them and attempts to reconstitute the Encoded Streams (Section 2.1.7) or Dependent Streams (Section 2.1.8) present in those RTP Streams.

It should be noted that in practical implementations, the Media Depacketizer and the Media Decoder may be tightly coupled and share information to improve or optimize the overall decoding process in various ways. It is however not expected that there would be any benefit in defining a taxonomy for those detailed (and likely very implementation-dependent) steps.

2.1.24. Received Encoded Stream

The received version of an Encoded Stream (Section 2.1.7).

2.1.25. Media Decoder

A Media Decoder is a transformation that is responsible for decoding Encoded Streams (Section 2.1.7) and any Dependent Streams (Section 2.1.8) into a Source Stream (Section 2.1.5).

It should be noted that in practical implementations, the Media Decoder and the Media Depacketizer may be tightly coupled and share information to improve or optimize the overall decoding process in various ways. It is however not expected that there would be any benefit in defining a taxonomy for those detailed (and likely very implementation-dependent) steps.

Characteristics:

- o A Media Decoder is the entity that will have to deal with any errors in the encoded streams that resulted from corruptions or failures to repair packet losses. This as a media decoder generally is forced to produce some output periodically. It thus commonly includes concealment methods.

2.1.26. Received Source Stream

The received version of a Source Stream (Section 2.1.5).

2.1.27. Media Sink

The Media Sink receives a Source Stream (Section 2.1.5) that contains, usually periodically, sampled media data together with associated synchronization information. Depending on application, this Source Stream then needs to be transformed into a Raw Stream (Section 2.1.3) that is sent in synchronization with the output from other Media Sinks to a Media Render (Section 2.1.29). The media sink may also be connected with a Media Source (Section 2.1.4) and be used as part of a conceptual Media Source.

Characteristics:

- o The Media Sink can further transform the Source Stream into a representation that is suitable for rendering on the Media Render as defined by the application or system-wide configuration. This include sample scaling, level adjustments etc.

2.1.28. Received Raw Stream

The received version of a Raw Stream (Section 2.1.3).

2.1.29. Media Render

A Media Render takes a Raw Stream (Section 2.1.3) and converts it into Physical Stimulus (Section 2.1.1) that a human user can perceive. Examples of such devices are screens, D/A converters connected to amplifiers and loudspeakers.

Characteristics:

- o An End Point can potentially have multiple Media Renders for each media type.

2.2. Communication Entities

This section contains concept for entities involved in the communication.

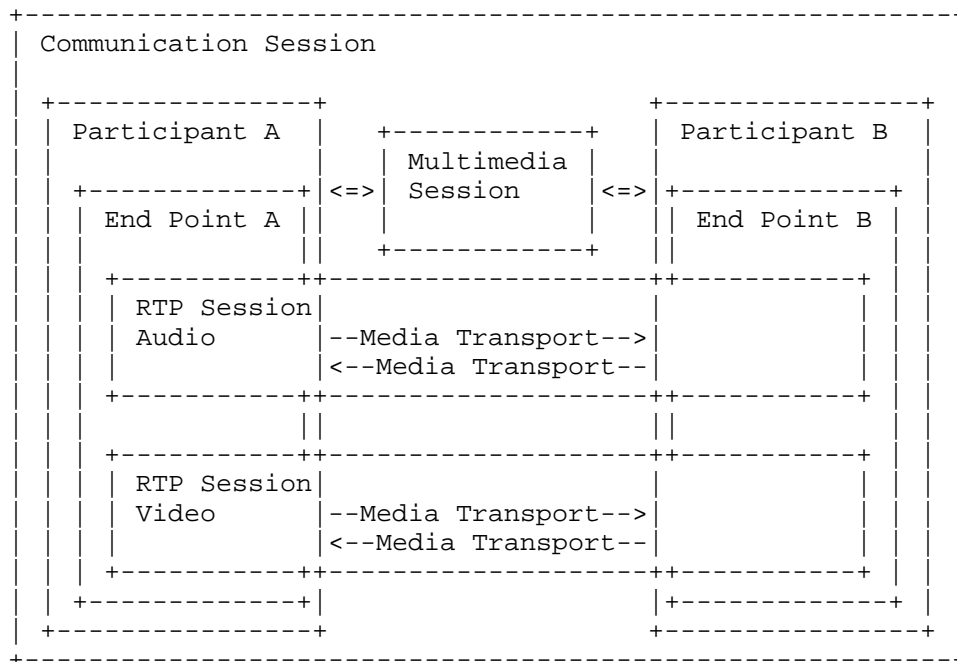


Figure 6: Example Point to Point Communication Session with two RTP Sessions

The figure above shows a high-level example representation of a very basic point-to-point Communication Session between Participants A and B. It uses two different audio and video RTP Sessions between A's and B's End Points, using separate Media Transports for those RTP Sessions. The Multimedia Session shared by the participants can for example be established using SIP (i.e., there is a SIP Dialog between A and B). The terms used in that figure are further elaborated in the sub-sections below.

2.2.1. End Point

Editor's note: Consider if a single word, "Endpoint", is preferable

A single addressable entity sending or receiving RTP packets. It may be decomposed into several functional blocks, but as long as it behaves as a single RTP stack entity it is classified as a single "End Point".

Characteristics:

- o End Points can be identified in several different ways. While RTP Canonical Names (CNAMEs) [RFC3550] provide a globally unique and stable identification mechanism for the duration of the Communication Session (see Section 2.2.5), their validity applies exclusively within a Synchronization Context (Section 3.1). Thus one End Point can handle multiple CNAMEs, each of which can be shared among a set of End Points belonging to the same Participant (Section 2.2.3). Therefore, mechanisms outside the scope of RTP, such as application defined mechanisms, must be used to ensure End Point identification when outside this Synchronization Context.
- o An End Point can be associated with at most one Participant (Section 2.2.3) at any single point in time.
- o In some contexts, an End Point would typically correspond to a single "host".

2.2.2. RTP Session

Editor's note: Re-consider if this is really a Communication Entity, or if it is rather an existing concept that should be described in Section 4.

An RTP session is an association among a group of participants communicating with RTP. It is a group communications channel which can potentially carry a number of RTP Streams. Within an RTP session, every participant can find meta-data and control information

(over RTCP) about all the RTP Streams in the RTP session. The bandwidth of the RTCP control channel is shared between all participants within an RTP Session.

Characteristics:

- o Typically, an RTP Session can carry one or more RTP Streams.
- o An RTP Session shares a single SSRC space as defined in RFC3550 [RFC3550]. That is, the End Points participating in an RTP Session can see an SSRC identifier transmitted by any of the other End Points. An End Point can receive an SSRC either as SSRC or as a Contributing source (CSRC) in RTP and RTCP packets, as defined by the endpoints' network interconnection topology.
- o An RTP Session uses at least two Media Transports (Section 2.1.13), one for sending and one for receiving. Commonly, the receiving one is the reverse direction of the same one as used for sending. An RTP Session may use many Media Transports and these define the session's network interconnection topology. A single Media Transport can normally not transport more than one RTP Session, unless a solution for multiplexing multiple RTP sessions over a single Media Transport is used. One example of such a scheme is Multiple RTP Sessions on a Single Lower-Layer Transport [I-D.westerlund-avtcore-transport-multiplexing].
- o Multiple RTP Sessions can be related.

2.2.3. Participant

A Participant is an entity reachable by a single signaling address, and is thus related more to the signaling context than to the media context.

Characteristics:

- o A single signaling-addressable entity, using an application-specific signaling address space, for example a SIP URI.
- o A Participant can have several Multimedia Sessions (Section 2.2.4).
- o A Participant can have several associated End Points (Section 2.2.1).

2.2.4. Multimedia Session

A multimedia session is an association among a group of participants engaged in the communication via one or more RTP Sessions (Section 2.2.2). It defines logical relationships among Media Sources (Section 2.1.4) that appear in multiple RTP Sessions.

Characteristics:

- o A Multimedia Session can be composed of several parallel RTP Sessions with potentially multiple RTP Streams per RTP Session.
- o Each participant in a Multimedia Session can have a multitude of Media Captures and Media Rendering devices.
- o A single Multimedia Session can contain media from one or more Synchronization Contexts (Section 3.1). An example of that is a Multimedia Session containing one set of audio and video for communication purposes belonging to one Synchronization Context, and another set of audio and video for presentation purposes (like playing a video file) with a separate Synchronization Context that has no strong timing relationship and need not be strictly synchronized with the audio and video used for communication.

2.2.5. Communication Session

A Communication Session is an association among group of participants communicating with each other via a set of Multimedia Sessions.

Characteristics:

- o Each participant in a Communication Session is identified via an application-specific signaling address.
- o A Communication Session is composed of at least one Multimedia Session per participant, involving one or more parallel RTP Sessions with potentially multiple RTP Streams per RTP Session.

For example, in a full mesh communication, the Communication Session consists of a set of separate Multimedia Sessions between each pair of Participants. Another example is a centralized conference, where the Communication Session consists of a set of Multimedia Sessions between each Participant and the conference handler.

3. Relations at Different Levels

This section uses the concepts from previous section and look at different types of relationships among them. These relationships occur at different levels and for different purposes. The section is organized such as to look at the level where a relation is required. The reason for the relationship may exist at another step in the media handling chain. For example, using Simulcast (discussed in Section 3.7) needs to determine relations at RTP Stream level, however the reason to relate RTP Streams is that multiple Media Encoders use the same Media Source, i.e. to be able to identify a common Media Source.

Media Sources (Section 2.1.4) are commonly grouped and related to an End Point (Section 2.2.1) or a Participant (Section 2.2.3). This occurs for several reasons; both due to application logic as well as for media handling purposes.

At RTP Packetization time, there exists a possibility for a number of different types of relationships between Encoded Streams (Section 2.1.7), Dependent Streams (Section 2.1.8) and RTP Streams (Section 2.1.10). These are caused by grouping together or distributing these different types of streams into RTP Streams.

The resulting RTP Streams will thus also have relations. This is a common relation to handle in RTP due to that RTP Streams are separate and have their own SSRC, implying independent sequence numbers and timestamp spaces. The underlying reasons for the RTP Stream relationships are different, as can be seen in the sub-sections below.

RTP Streams may be protected by Redundancy RTP Streams during transport. Several approaches listed below can be used to create Redundancy RTP Streams;

- o Duplication of the original RTP Stream
- o Duplication of the original RTP Stream with a time offset,
- o Forward Error Correction (FEC) techniques, and
- o Retransmission of lost packets (either globally or selectively).

The different RTP Streams can be transported within the same RTP Session or in different RTP Sessions to accomplish different transport goals. This explicit separation of RTP Streams is further discussed in Section 3.13.

3.1. Synchronization Context

A Synchronization Context defines a requirement on a strong timing relationship between the Media Sources, typically requiring alignment of clock sources. Such relationship can be identified in multiple ways as listed below. A single Media Source can only belong to a single Synchronization Context, since it is assumed that a single Media Source can only have a single media clock and requiring alignment to several Synchronization Contexts (and thus reference clocks) will effectively merge those into a single Synchronization Context.

3.1.1. RTCP CNAME

RFC3550 [RFC3550] describes Inter-media synchronization between RTP Sessions based on RTCP CNAME, RTP and Network Time Protocol (NTP) [RFC5905] formatted timestamps of a reference clock. As indicated in [I-D.ietf-avtcore-clksrc], despite using NTP format timestamps, it is not required that the clock be synchronized to an NTP source.

3.1.2. Clock Source Signaling

[I-D.ietf-avtcore-clksrc] provides a mechanism to signal the clock source in SDP both for the reference clock as well as the media clock, thus allowing a Synchronization Context to be defined beyond the one defined by the usage of CNAME source descriptions.

3.1.3. Implicitly via RtcMediaStream

The WebRTC WG defines "RtcMediaStream" with one or more "RtcMediaStreamTracks". All tracks in a "RtcMediaStream" are intended to be possible to synchronize when rendered.

3.1.4. Explicitly via SDP Mechanisms

RFC5888 [RFC5888] defines m=line grouping mechanism called "Lip Synchronization (LS)" for establishing the synchronization requirement across m=lines when they map to individual sources.

RFC5576 [RFC5576] extends the above mechanism when multiple media sources are described by a single m=line.

3.2. End Point

Some applications requires knowledge of what Media Sources originate from a particular End Point (Section 2.2.1). This can include such decisions as packet routing between parts of the topology, knowing the End Point origin of the RTP Streams.

In RTP, this identification has been overloaded with the Synchronization Context (Section 3.1) through the usage of the RTCP source description CNAME (Section 3.1.1) item. This works for some usages, but sometimes it breaks down. For example, if an End Point has two sets of Media Sources that have different Synchronization Contexts, like the audio and video of the human participant as well as a set of Media Sources of audio and video for a shared movie. Thus, an End Point may have multiple CNAMEs. The CNAMEs or the Media Sources themselves can be related to the End Point.

3.3. Participant

In communication scenarios, it is commonly needed to know which Media Sources that originate from which Participant (Section 2.2.3). Thus enabling the application to for example display Participant Identity information correctly associated with the Media Sources. This association is currently handled through the signaling solution to point at a specific Multimedia Session where the Media Sources may be explicitly or implicitly tied to a particular End Point.

Participant information becomes more problematic due to Media Sources that are generated through mixing or other conceptual processing of Raw Streams or Source Streams that originate from different Participants. This type of Media Sources can thus have a dynamically varying set of origins and Participants. RTP contains the concept of Contributing Sources (CSRC) that carries such information about the previous step origin of the included media content on RTP level.

3.4. RtcMediaStream

An RtcMediaStream in WebRTC is an explicit grouping of a set of Media Sources (RtcMediaStreamTracks) that share a common identifier and a single Synchronization Context (Section 3.1).

3.5. Single- and Multi-Session Transmission of SVC

Scalable Video Coding [RFC6190] has a mode of operation called Single Session Transmission (SST), where Encoded Streams and Dependent Streams from the SVC Media Encoder are sent in a single RTP Session (Section 2.2.2) using the SVC RTP Payload format. There is another mode of operation where Encoded Streams and Dependent Streams are distributed across multiple RTP Sessions, called Multi-Session Transmission (MST). SST denotes one or more RTP Streams (SSRC) per Media Source in a single RTP Session. MST denotes one or more RTP Streams (SSRC) per Media Source in each of multiple RTP Sessions. This is not always clear from the SVC payload format text [RFC6190], but is what existing deployments of that RFC have implemented.

To elaborate, what could be called SST-SingleStream (SST-SS) uses a single RTP Stream in a single RTP Session to send all Encoded and Dependent Streams from a single Media Source. Similarly, SST-MultiStream (SST-MS) uses a single RTP Stream per Media Source in a single RTP Session to send the Encoded and Dependent Streams. MST-SS uses a single RTP Stream in each of multiple RTP Sessions, where each RTP Stream can originate from any one of possibly multiple Media Sources. Finally, MST-MS uses multiple RTP Streams in each of the multiple RTP Sessions, where each RTP Stream can originate from any one of possibly multiple Media Sources. This is summarized below:

RTP Streams per Media Source	Single RTP Session	Multiple RTP Sessions
Single	SST-SS	MST-SS
Multiple	SST-MS	MST-MS

Table 1: SST / MST Summary

3.6. Multi-Channel Audio

There exist a number of RTP payload formats that can carry multi-channel audio, despite the codec being a mono encoder. Multi-channel audio can be viewed as multiple Media Sources sharing a common Synchronization Context. These are independently encoded by a Media Encoder and the different Encoded Streams are then packetized together in a time synchronized way into a single Source RTP Stream using the used codec's RTP Payload format. Example of such codecs are, PCMA and PCMU [RFC3551], AMR [RFC4867], and G.719 [RFC5404].

3.7. Simulcast

A Media Source represented as multiple independent Encoded Streams constitutes a simulcast of that Media Source. Figure 7 below represents an example of a Media Source that is encoded into three separate and different Simulcast streams, that are in turn sent on the same Media Transport flow. When using Simulcast, the RTP Streams may be sharing RTP Session and Media Transport, or be separated on different RTP Sessions and Media Transports, or be any combination of these two. It is other considerations that affect which usage is desirable, as discussed in Section 3.13.

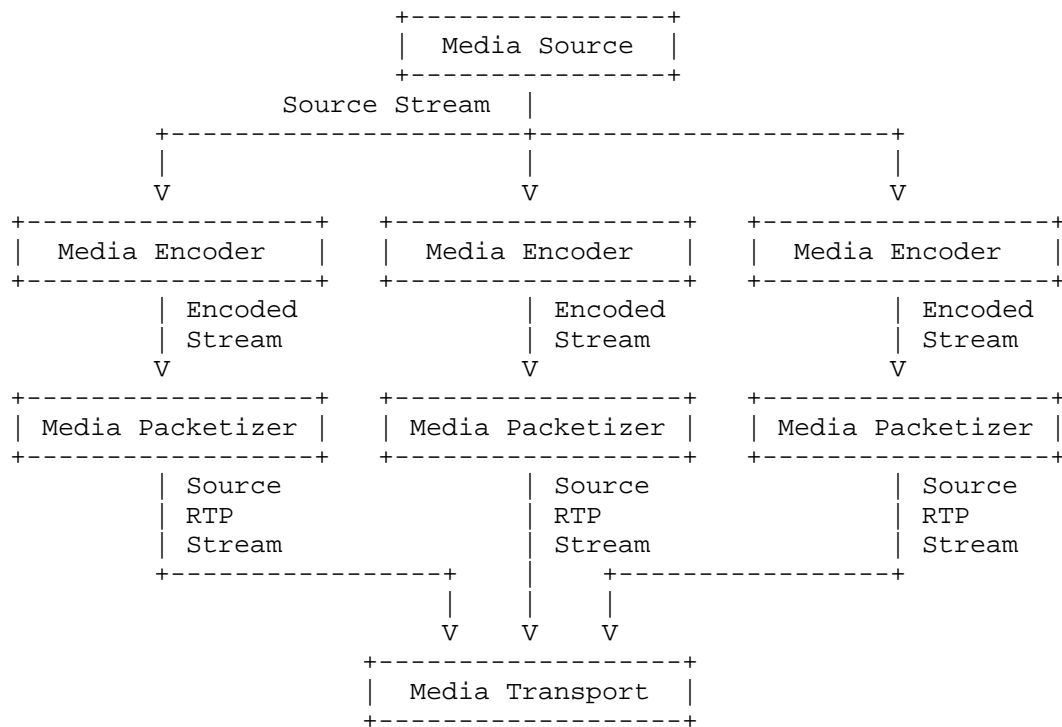


Figure 7: Example of Media Source Simulcast

The simulcast relation between the RTP Streams is the common Media Source. In addition, to be able to identify the common Media Source, a receiver of the RTP Stream may need to know which configuration or encoding goals that lay behind the produced Encoded Stream and its properties. This to enable selection of the stream that is most useful in the application at that moment.

3.8. Layered Multi-Stream

Layered Multi-Stream (LMS) is a mechanism by which different portions of a layered encoding of a Source Stream are sent using separate RTP Streams (sometimes in separate RTP Sessions). LMSs are useful for receiver control of layered media.

A Media Source represented as an Encoded Stream and multiple Dependent Streams constitutes a Media Source that has layered dependencies. The figure below represents an example of a Media Source that is encoded into three dependent layers, where two layers are sent on the same Media Transport using different RTP Streams,

i.e. SSRCs, and the third layer is sent on a separate Media Transport, i.e. a different RTP Session.

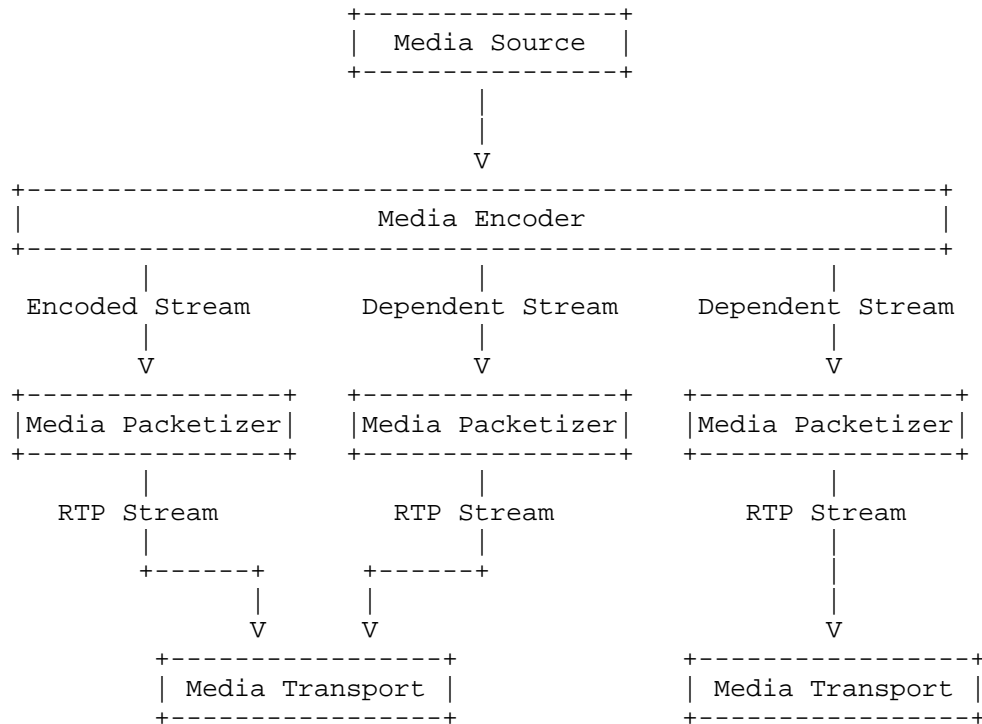


Figure 8: Example of Media Source Layered Dependency

As an example, the SVC MST (Section 3.5) relation needs to identify the common Media Encoder origin for the Encoded and Dependent Streams. The SVC RTP Payload RFC is not particularly explicit about how this relation is to be implemented. When using different RTP Sessions, thus different Media Transports, and as long as there is only one RTP Stream per Media Encoder and a single Media Source in each RTP Session (MST-SS (Section 3.5)), common SSRC and CNAMEs can be used to identify the common Media Source. When multiple RTP Streams are sent from one Media Encoder in the same RTP Session (SST-MS), then CNAME is the only currently specified RTP identifier that can be used. In cases where multiple Media Encoders use multiple Media Sources sharing Synchronization Context, and thus having a common CNAME, additional heuristics need to be applied to create the MST relationship between the RTP Streams.

3.9. RTP Stream Duplication

RTP Stream Duplication [RFC7198], using the same or different Media Transports, and optionally also delaying the duplicate [RFC7197], offers a simple way to protect media flows from packet loss in some cases. It is a specific type of redundancy and all but one Source RTP Stream (Section 2.1.10) are effectively Redundancy RTP Streams (Section 2.1.12), but since both Source and Redundant RTP Streams are the same it does not matter which is which. This can also be seen as a specific type of Simulcast (Section 3.7) that transmits the same Encoded Stream (Section 2.1.7) multiple times.

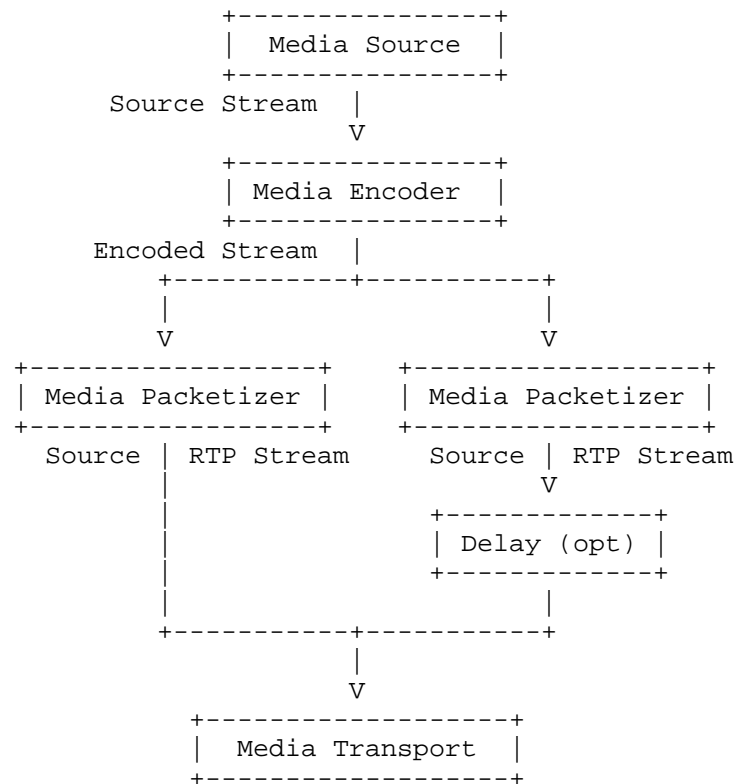


Figure 9: Example of RTP Stream Duplication

3.10. Redundancy Format

The RTP Payload for Redundant Audio Data [RFC2198] defines how one can transport redundant audio data together with primary data in the same RTP payload. The redundant data can be a time delayed version of the primary or another time delayed Encoded Stream using a

different Media Encoder to encode the same Media Source as the primary, as depicted below in Figure 10.

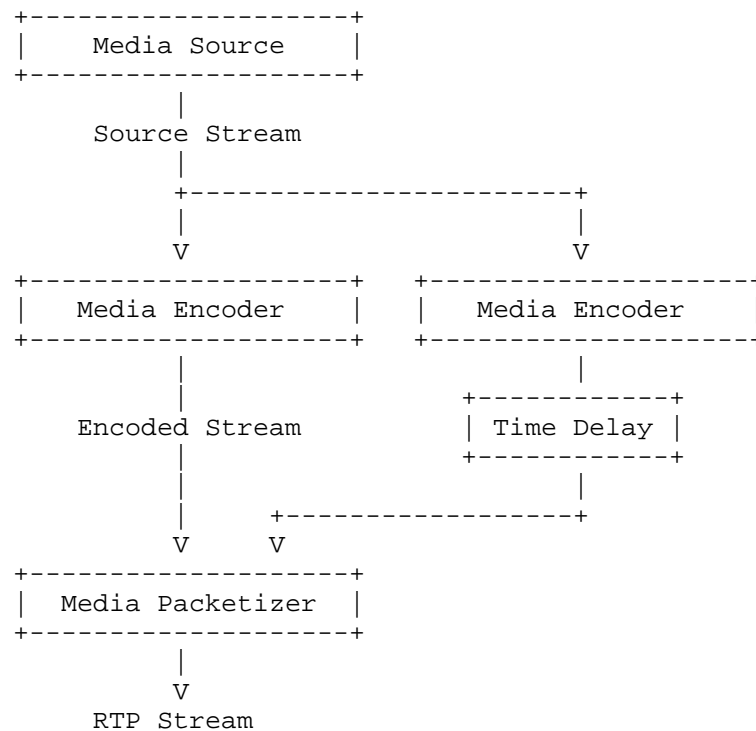


Figure 10: Concept for usage of Audio Redundancy with different Media Encoders

The Redundancy format is thus providing the necessary meta information to correctly relate different parts of the same Encoded Stream, or in the case depicted above (Figure 10) relate the Received Source Stream fragments coming out of different Media Decoders to be able to combine them together into a less erroneous Source Stream.

3.11. RTP Retransmission

The figure below (Figure 11) represents an example where a Media Source's Source RTP Stream is protected by a retransmission (RTX) flow [RFC4588]. In this example the Source RTP Stream and the Redundancy RTP Stream share the same Media Transport.

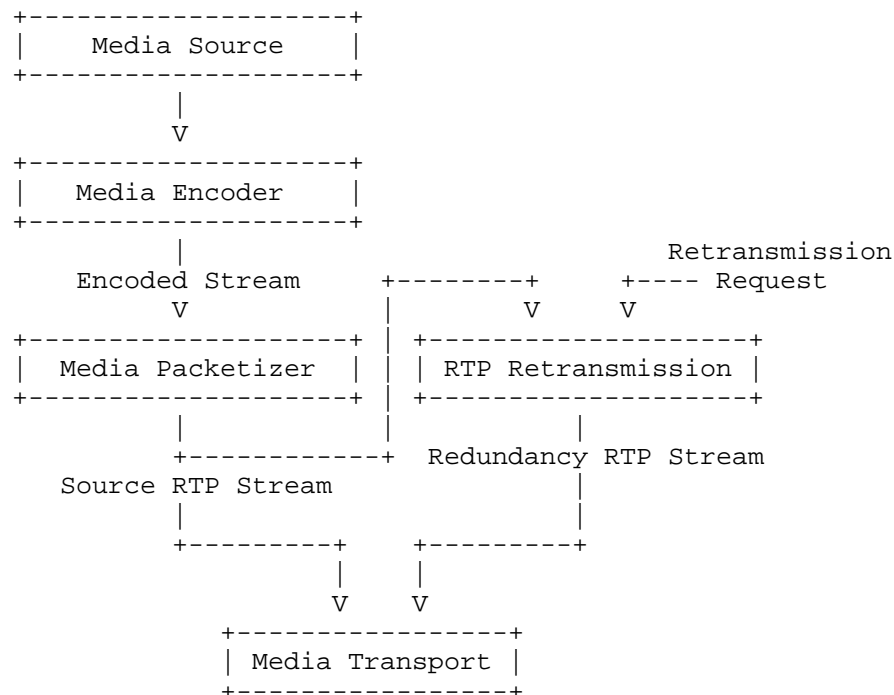


Figure 11: Example of Media Source Retransmission Flows

The RTP Retransmission example (Figure 11) helps illustrate that this mechanism works purely on the Source RTP Stream. The RTP Retransmission transform buffers the sent Source RTP Stream and upon requests emits a retransmitted packet with some extra payload header as a Redundancy RTP Stream. The RTP Retransmission mechanism [RFC4588] is specified so that there is a one to one relation between the Source RTP Stream and the Redundancy RTP Stream. Thus a Redundancy RTP Stream needs to be associated with its Source RTP Stream upon being received. This is done based on CNAME selectors and heuristics to match requested packets for a given Source RTP Stream with the original sequence number in the payload of any new Redundancy RTP Stream using the RTX payload format. In cases where the Redundancy RTP Stream is sent in a separate RTP Session from the Source RTP Stream, these sessions are related, e.g. using the SDP Media Grouping's [RFC5888] FID semantics.

3.12. Forward Error Correction

The figure below (Figure 12) represents an example where two Media Sources' Source RTP Streams are protected by FEC. Source RTP Stream A has a Media Redundancy transformation in FEC Encoder 1. This

produces a Redundancy RTP Stream 1, that is only related to Source RTP Stream A. The FEC Encoder 2, however takes two Source RTP Streams (A and B) and produces a Redundancy RTP Stream 2 that protects them together, i.e. Redundancy RTP Stream 2 relate to two Source RTP Streams (a FEC group). FEC decoding, when needed due to packet loss or packet corruption at the receiver, requires knowledge about which Source RTP Streams that the FEC encoding was based on.

In Figure 12 all RTP Streams are sent on the same Media Transport. This is however not the only possible choice. Numerous combinations exist for spreading these RTP Streams over different Media Transports to achieve the communication application's goal.

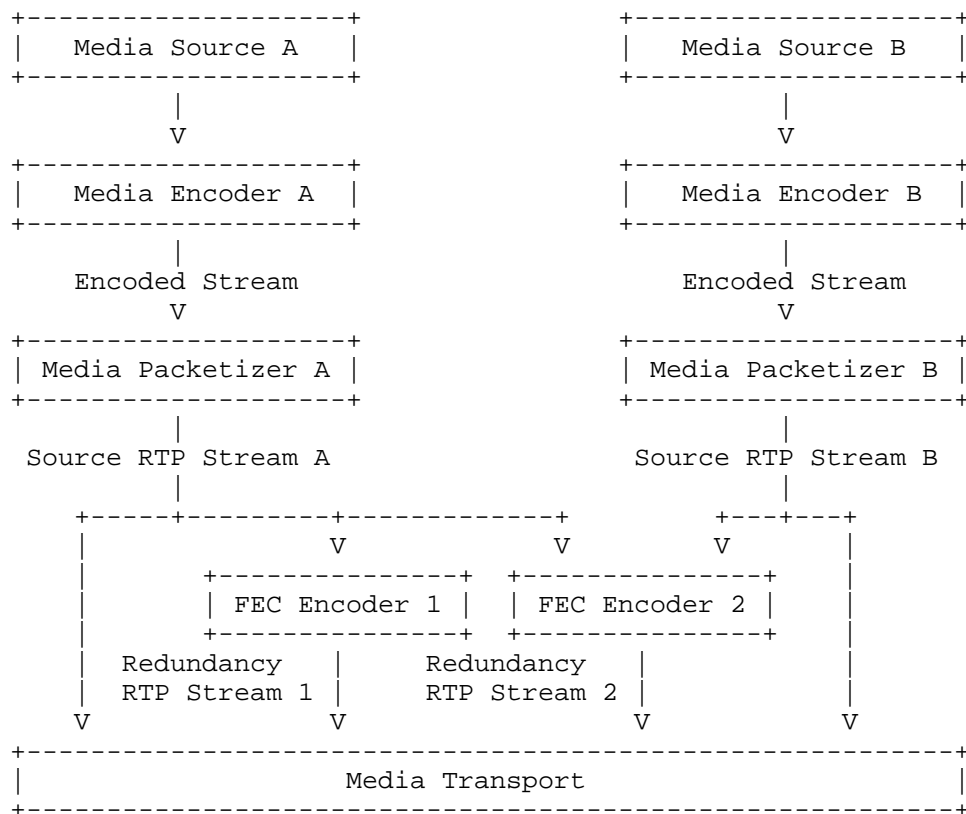


Figure 12: Example of FEC Flows

As FEC Encoding exists in various forms, the methods for relating FEC Redundancy RTP Streams with its source information in Source RTP Streams are many. The XOR based RTP FEC Payload format [RFC5109] is defined in such a way that a Redundancy RTP Stream has a one to one

relation with a Source RTP Stream. In fact, the RFC requires the Redundancy RTP Stream to use the same SSRC as the Source RTP Stream. This requires to either use a separate RTP session or to use the Redundancy RTP Payload format [RFC2198]. The underlying relation requirement for this FEC format and a particular Redundancy RTP Stream is to know the related Source RTP Stream, including its SSRC.

3.13. RTP Stream Separation

RTP Streams can be separated exclusively based on their SSRCs, at the RTP Session level, or at the Multi-Media Session level.

When the RTP Streams that have a relationship are all sent in the same RTP Session and are uniquely identified based on their SSRC only, it is termed an SSRC-Only Based Separation. Such streams can be related via RTCP CNAME to identify that the streams belong to the same End Point. [RFC5576]-based approaches, when used, can explicitly relate various such RTP Streams.

On the other hand, when RTP Streams that are related but are sent in the context of different RTP Sessions to achieve separation, it is known as RTP Session-based separation. This is commonly used when the different RTP Streams are intended for different Media Transports.

Several mechanisms that use RTP Session-based separation rely on it to enable an implicit grouping mechanism expressing the relationship. The solutions have been based on using the same SSRC value in the different RTP Sessions to implicitly indicate their relation. That way, no explicit RTP level mechanism has been needed, only signaling level relations have been established using semantics from Grouping of Media lines framework [RFC5888]. Examples of this are RTP Retransmission [RFC4588], SVC Multi-Session Transmission [RFC6190] and XOR Based FEC [RFC5109]. RTCP CNAME explicitly relates RTP Streams across different RTP Sessions, as explained in the previous section. Such a relationship can be used to perform inter-media synchronization.

RTP Streams that are related and need to be associated can be part of different Multimedia Sessions, rather than just different RTP sessions within the same Multimedia Session context. This puts further demand on the scope of the mechanism(s) and its handling of identifiers used for expressing the relationships.

3.14. Multiple RTP Sessions over one Media Transport

[I-D.westerlund-avtcore-transport-multiplexing] describes a mechanism that allow several RTP Sessions to be carried over a single underlying Media Transport. The main reasons for doing this are related to the impact of using one or more Media Transports. Thus using a common network path or potentially have different ones. There is reduced need for NAT/FW traversal resources and no need for flow based QoS.

However, Multiple RTP Sessions over one Media Transport makes it clear that a single Media Transport 5-tuple is not sufficient to express which RTP Session context a particular RTP Stream exists in. Complexities in the relationship between Media Transports and RTP Session already exist as one RTP Session contains multiple Media Transports, e.g. even a Peer-to-Peer RTP Session with RTP/RTCP Multiplexing requires two Media Transports, one in each direction. The relationship between Media Transports and RTP Sessions as well as additional levels of identifiers need to be considered in both signaling design and when defining terminology.

4. Mapping from Existing Terms

This section describes a selected set of terms from some relevant IETF RFC and Internet Drafts (at the time of writing), using the concepts from previous sections.

4.1. Audio Capture

Telepresence specifications from CLUE WG uses this term to describe an audio Media Source (Section 2.1.4).

4.2. Capture Device

Telepresence specifications from CLUE WG use this term to identify a physical entity performing a Media Capture (Section 2.1.2) transformation.

4.3. Capture Encoding

Telepresence specifications from CLUE WG uses this term to describe an Encoded Stream (Section 2.1.7) related to CLUE specific semantic information.

4.4. Capture Scene

Telepresence specifications from CLUE WG uses this term to describe a set of spatially related Media Sources (Section 2.1.4).

4.5. Endpoint

Telepresence specifications from CLUE WG use this term to describe exactly one Participant (Section 2.2.3) and one or more End Points (Section 2.2.1).

4.6. Individual Encoding

Telepresence specifications from CLUE WG use this term to describe the configuration information needed to perform a Media Encoder (Section 2.1.6) transformation.

4.7. Multipoint Control Unit (MCU)

This term is commonly used to describe the central node in any type of star topology [I-D.ietf-avtcore-rtp-topologies-update] conference. It describes a device that includes one Participant (Section 2.2.3) (usually corresponding to a so-called conference focus) and one or more related End Points (Section 2.2.1) (sometimes one or more per conference participant).

4.8. Media Capture

Telepresence specifications from CLUE WG uses this term to describe either a Media Capture (Section 2.1.2) or a Media Source (Section 2.1.4), depending on in which context the term is used.

4.9. Media Consumer

Telepresence specifications from CLUE WG use this term to describe the media receiving part of an End Point (Section 2.2.1).

4.10. Media Description

A single Source Description Protocol (SDP) [RFC4566] media description (or media block; an m-line and all subsequent lines until the next m-line or the end of the SDP) describes part of the necessary configuration and identification information needed for a Media Encoder transformation, as well as the necessary configuration and identification information for the Media Decoder to be able to correctly interpret a received RTP Stream.

A Media Description typically relates to a single Media Source. This is for example an explicit restriction in WebRTC. However, nothing prevents that the same Media Description (and same RTP Session) is re-used for multiple Media Sources [I-D.ietf-avtcore-rtp-multi-stream]. It can thus describe properties of one or more RTP Streams, and can also describe properties valid for an entire RTP Session (via [RFC5576] mechanisms, for example).

4.11. Media Provider

Telepresence specifications from CLUE WG use this term to describe the media sending part of an End Point (Section 2.2.1).

4.12. Media Stream

RTP [RFC3550] uses media stream, audio stream, video stream, and stream of (RTP) packets interchangeably, which are all RTP Streams.

4.13. Multimedia Session

SDP [RFC4566] defines a multimedia session as a set of multimedia senders and receivers and the data streams flowing from senders to receivers, which would correspond to a set of End Points and the RTP Streams that flow between them. In this memo, Multimedia Session also assumes those End Points belong to a set of Participants that are engaged in communication via a set of related RTP Streams.

RTP [RFC3550] defines a multimedia session as a set of concurrent RTP Sessions among a common group of participants. For example, a video conference may contain an audio RTP Session and a video RTP Session. This would correspond to a group of Participants (each using one or more End Points) sharing a set of concurrent RTP Sessions. In this memo, Multimedia Session also defines those RTP Sessions to have some relation and be part of a communication among the Participants.

4.14. Recording Device

WebRTC specifications use this term to refer to locally available entities performing a Media Capture (Section 2.1.2) transformation.

4.15. RtcMediaStream

A WebRTC RtcMediaStreamTrack is a set of Media Sources (Section 2.1.4) sharing the same Synchronization Context (Section 3.1).

4.16. RtcMediaStreamTrack

A WebRTC RtcMediaStreamTrack is a Media Source (Section 2.1.4).

4.17. RTP Sender

RTP [RFC3550] uses this term, which can be seen as the RTP protocol part of a Media Packetizer (Section 2.1.9).

4.18. RTP Session

Within the context of SDP, a single m=line can map to a single RTP Session or multiple m=lines can map to a single RTP Session. The latter is enabled via multiplexing schemes such as BUNDLE [I-D.ietf-mmusic-sdp-bundle-negotiation], for example, which allows mapping of multiple m=lines to a single RTP Session.

Editor's note: Consider if the contents of Section 2.2.2 should be moved here, or if this section should be kept and refer to the above.

4.19. SSRC

RTP [RFC3550] defines this as "the source of a stream of RTP packets", which indicates that an SSRC is not only a unique identifier for the Encoded Stream (Section 2.1.7) carried in those packets, but is also effectively used as a term to denote a Media Packetizer (Section 2.1.9).

4.20. Stream

Telepresence specifications from CLUE WG use this term to describe an RTP Stream (Section 2.1.10).

4.21. Video Capture

Telepresence specifications from CLUE WG uses this term to describe a video Media Source (Section 2.1.4).

5. Security Considerations

This document simply tries to clarify the confusion prevalent in RTP taxonomy because of inconsistent usage by multiple technologies and protocols making use of the RTP protocol. It does not introduce any new security considerations beyond those already well documented in the RTP protocol [RFC3550] and each of the many respective specifications of the various protocols making use of it.

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.

6. Acknowledgement

This document has many concepts borrowed from several documents such as WebRTC [I-D.ietf-rtcweb-overview], CLUE [I-D.ietf-clue-framework], Multiplexing Architecture [I-D.westerlund-avtcore-transport-multiplexing]. The authors would like to thank all the authors of each of those documents.

The authors would also like to acknowledge the insights, guidance and contributions of Magnus Westerlund, Roni Even, Paul Kyzivat, Colin Perkins, Keith Drage, Harald Alvestrand, and Alex Eleftheriadis.

7. Contributors

Magnus Westerlund has contributed the concept model for the media chain using transformations and streams model, including rewriting pre-existing concepts into this model and adding missing concepts. The first proposal for updating the relationships and the topologies based on this concept was also performed by Magnus.

8. IANA Considerations

This document makes no request of IANA.

9. Informative References

- [I-D.ietf-avtcore-clksrc]
Williams, A., Gross, K., Brandenburg, R., and H. Stokking,
"RTP Clock Source Signalling", draft-ietf-avtcore-
clksrc-11 (work in progress), March 2014.
- [I-D.ietf-avtcore-rtp-multi-stream]
Lennox, J., Westerlund, M., Wu, W., and C. Perkins,
"Sending Multiple Media Streams in a Single RTP Session",
draft-ietf-avtcore-rtp-multi-stream-04 (work in progress),
May 2014.
- [I-D.ietf-avtcore-rtp-topologies-update]
Westerlund, M. and S. Wenger, "RTP Topologies", draft-
ietf-avtcore-rtp-topologies-update-02 (work in progress),
May 2014.

- [I-D.ietf-clue-framework]
Duckworth, M., Pepperell, A., and S. Wenger, "Framework for Telepresence Multi-Streams", draft-ietf-clue-framework-15 (work in progress), May 2014.
- [I-D.ietf-mmusic-sdp-bundle-negotiation]
Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", draft-ietf-mmusic-sdp-bundle-negotiation-07 (work in progress), April 2014.
- [I-D.ietf-rtcweb-overview]
Alvestrand, H., "Overview: Real Time Protocols for Browser-based Applications", draft-ietf-rtcweb-overview-10 (work in progress), June 2014.
- [I-D.westerlund-avtcore-transport-multiplexing]
Westerlund, M. and C. Perkins, "Multiplexing Multiple RTP Sessions onto a Single Lower-Layer Transport", draft-westerlund-avtcore-transport-multiplexing-07 (work in progress), October 2013.
- [RFC2198] Perkins, C., Kouvelas, I., Hodson, O., Hardman, V., Handley, M., Bolot, J., Vega-Garcia, A., and S. Fosse-Parisis, "RTP Payload for Redundant Audio Data", RFC 2198, September 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.
- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.
- [RFC4867] Sjöberg, J., Westerlund, M., Lakaniemi, A., and Q. Xie, "RTP Payload Format and File Storage Format for the Adaptive Multi-Rate (AMR) and Adaptive Multi-Rate Wideband (AMR-WB) Audio Codecs", RFC 4867, April 2007.

- [RFC5109] Li, A., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, December 2007.
- [RFC5404] Westerlund, M. and I. Johansson, "RTP Payload Format for G.719", RFC 5404, January 2009.
- [RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.
- [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.
- [RFC6190] Wenger, S., Wang, Y., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, May 2011.
- [RFC7160] Petit-Huguenin, M. and G. Zorn, "Support for Multiple Clock Rates in an RTP Session", RFC 7160, April 2014.
- [RFC7197] Begen, A., Cai, Y., and H. Ou, "Duplication Delay Attribute in the Session Description Protocol", RFC 7197, April 2014.
- [RFC7198] Begen, A. and C. Perkins, "Duplicating RTP Streams", RFC 7198, April 2014.

Appendix A. Changes From Earlier Versions

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

A.1. Modifications Between WG Version -01 and -02

- o Major re-structure
- o Moved media chain Media Transport detailing up one section level
- o Collapsed level 2 sub-sections of section 3 and thus moved level 3 sub-sections up one level, gathering some introductory text into the beginning of section 3
- o Added that not only SSRC collision, but also a clock rate change [RFC7160] is a valid reason to change SSRC value for an RTP stream

- o Added a sub-section on clock source signaling
- o Added a sub-section on RTP stream duplication
- o Elaborated a bit in section 2.2.1 on the relation between End Points, Participants and CNAMEs
- o Elaborated a bit in section 2.2.4 on Multimedia Session and synchronization contexts
- o Removed the section on CLUE scenes defining an implicit synchronization context, since it was incorrect
- o Clarified text on SVC SST and MST according to list discussions
- o Removed the entire topology section to avoid possible inconsistencies or duplications with draft-ietf-avtcore-rtp-topologies-update, but saved one example overview figure of Communication Entities into that section
- o Added a section 4 on mapping from existing terms with one sub-section per term, mainly by moving text from sections 2 and 3
- o Changed all occurrences of Packet Stream to RTP Stream
- o Moved all normative references to informative, since this is an informative document
- o Added references to RFC 7160, RFC 7197 and RFC 7198, and removed unused references

A.2. Modifications Between WG Version -00 and -01

- o WG version -00 text is identical to individual draft -03
- o Amended description of SVC SST and MST encodings with respect to concepts defined in this text
- o Removed UML as normative reference, since the text no longer uses any UML notation
- o Removed a number of level 4 sections and moved out text to the level above

A.3. Modifications Between Version -02 and -03

- o Section 4 rewritten (and new communication topologies added) to reflect the major updates to Sections 1-3
- o Section 8 removed (carryover from initial -00 draft)
- o General clean up of text, grammar and nits

A.4. Modifications Between Version -01 and -02

- o Section 2 rewritten to add both streams and transformations in the media chain.
- o Section 3 rewritten to focus on exposing relationships.

A.5. Modifications Between Version -00 and -01

- o Too many to list
- o Added new authors
- o Updated content organization and presentation

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

Suhas Nandakumar
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
US

Email: snandaku@cisco.com

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

Email: gsalguei@cisco.com

Bo Burman
Ericsson
Kistavagen 25
SE-164 80 Kista
Sweden

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

Network Working Group
Internet-Draft
Updates: 5104 (if approved)
Intended status: Standards Track
Expires: April 30, 2015

B. Burman
A. Akram
Ericsson
R. Even
Huawei Technologies
M. Westerlund
Ericsson
October 27, 2014

RTP Stream Pause and Resume
draft-ietf-avtext-rtp-stream-pause-05

Abstract

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

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 30, 2015.

Copyright Notice

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

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

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	4
2. Definitions	5
2.1. Abbreviations	5
2.2. Terminology	6
2.3. Requirements Language	7
3. Use Cases	7
3.1. Point to Point	7
3.2. RTP Mixer to Media Sender	8
3.3. RTP Mixer to Media Sender in Point-to-Multipoint	9
3.4. Media Receiver to RTP Mixer	10
3.5. Media Receiver to Media Sender Across RTP Mixer	10
4. Design Considerations	11
4.1. Real-time Nature	11
4.2. Message Direction	11
4.3. Apply to Individual Sources	12
4.4. Consensus	12
4.5. Message Acknowledgments	12
4.6. Request Retransmission	13
4.7. Sequence Numbering	13

4.8. Relation to Other Solutions	13
5. Solution Overview	14
5.1. Expressing Capability	15
5.2. Requesting to Pause	15
5.3. Media Sender Pausing	16
5.4. Requesting to Resume	18
5.5. TMMBR/TMMBN Considerations	19
6. Participant States	19
6.1. Playing State	20
6.2. Pausing State	20
6.3. Paused State	21
6.3.1. RTCP BYE Message	21
6.3.2. SSRC Time-out	22
6.4. Local Paused State	22
7. Message Format	23
8. Message Details	25
8.1. PAUSE	25
8.2. PAUSED	26
8.3. RESUME	27
8.4. REFUSED	28
8.5. Transmission Rules	28
9. Signaling	29
9.1. Offer-Answer Use	32
9.2. Declarative Use	33
10. Examples	33
10.1. Offer-Answer	34
10.2. Point-to-Point Session	35
10.3. Point-to-Multipoint using Mixer	39
10.4. Point-to-Multipoint using Translator	41
11. IANA Considerations	44
12. Security Considerations	45
13. Contributors	45
14. Acknowledgements	45
15. References	46
15.1. Normative References	46
15.2. Informative References	46
Appendix A. Changes From Earlier Versions	47
A.1. Modifications Between Version -04 and -05	47
A.2. Modifications Between Version -03 and -04	47
A.3. Modifications Between Version -02 and -03	48
A.4. Modifications Between Version -01 and -02	48
A.5. Modifications Between Version -00 and -01	48
Authors' Addresses	49

1. Introduction

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

Multimedia conferencing may involve many participants; each has its own preferences for the communication session, not only at the start but also during the session. This document describes several scenarios in multimedia communication where a conferencing node or participant chooses to temporarily pause an incoming RTP [RFC3550] stream and later resume it when needed. The receiver does not need to terminate or inactivate the RTP session and start all over again by negotiating the session parameters, for example using SIP [RFC3261] with SDP Offer/Answer [RFC3264].

Centralized nodes, like RTP Mixers or MCUs, which either uses logic based on voice activity, other measurements, or user input could reduce the resources consumed in both the sender and the network by temporarily pausing the RTP streams that aren't required by the RTP Mixer. If the number of conference participants are greater than what the conference logic has chosen to present simultaneously to receiving participants, some participant RTP streams sent to the RTP Mixer may not need to be forwarded to any other participant. Those RTP streams could then be temporarily paused. This becomes especially useful when the media sources are provided in multiple encoding versions (Simulcast) [I-D.westerlund-avtcore-rtp-simulcast] or with Multi-Session Transmission (MST) of scalable encoding such as SVC [RFC6190]. There may be some of the defined encodings or combination of scalable layers that are not used or cannot be used all of the time, for example due to temporarily limited network or processing resources, and a centralized node may choose to pause such RTP streams without being requested to do so, but anyway send an explicit indication that the stream is paused.

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

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

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

2. Definitions

2.1. Abbreviations

3GPP: 3rd Generation Partnership Project

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

BGW: Border Gateway

CCM: Codec Control Messages (RFC 5104)

CNAME: Canonical Name (RTCP SDES)

CSRC: Contributing Source (RTP)

FB: Feedback (AVPF)

FCI: Feedback Control Information (AVPF)

FIR: Full Intra Refresh (CCM)

FMT: Feedback Message Type (AVPF)

LTE: Long-Term Evolution (3GPP)

MCU: Multipoint Control Unit

MTU: Maximum Transfer Unit

PT: Payload Type (RTP)

RTP: Real-time Transport Protocol (RFC 3550)

RTCP: RTP Control Protocol (RFC 3550)

RTCP RR: RTCP Receiver Report

SDP: Session Description Protocol (RFC 4566)

SGW: Signaling Gateway

SIP: Session Initiation Protocol (RFC 3261)

SSRC: Synchronization Source (RTP)

SVC: Scalable Video Coding

TCP: Transmission Control Protocol (RFC 793)

TMMBR: Temporary Maximum Media Bitrate Request (CCM)

TMMBN: Temporary Maximum Media Bitrate Notification (CCM)

UA: User Agent (SIP)

UDP: User Datagram Protocol (RFC 768)

2.2. Terminology

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

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

PAUSE Request from an RTP stream receiver to pause a stream

RESUME Request from an RTP stream receiver to resume a paused stream

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

REFUSED Indication from an RTP stream sender that a PAUSE or RESUME request will not be honored

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

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

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

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

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

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

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

2.3. Requirements Language

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

3. Use Cases

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

3.1. Point to Point

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



Figure 1: Point to Point

The usage of RTP stream pause in this use case is to temporarily halt delivery of streams that the sender provides but the receiver does not currently use. This can for example be due to minimized applications where the video stream is not actually shown on any

display, and neither is it used in any other way, such as being recorded.

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

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

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

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

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

3.2. RTP Mixer to Media Sender

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

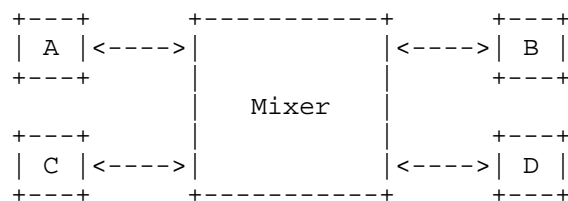


Figure 2: RTP Mixer in Unicast-only

Which streams that are delivered to a given receiver, A, can depend on several things. It can either be the RTP Mixer's own logic and

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

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

In some cases, especially when the Mixer sends multiple RTP streams per receiving client, there may be situations that makes it desirable to the Mixer to pause some of its sent RTP streams, even without being explicitly asked to do so by the receiving client. Such situations can for example be caused by a temporary lack of available Mixer network or processing resources. An RTP stream receiver that no longer receives an RTP stream could interpret this as an error condition and try to take action to re-establish the RTP stream. Such action would likely be undesirable if the RTP stream was in fact deliberately paused by the Mixer. Undesirable RTP stream receiver actions could be avoided if the Mixer is able to explicitly indicate that an RTP stream is deliberately paused.

Just as for point-to-point (Section 3.1), there is only a single receiver of the stream, the RTP Mixer, and pausing or resuming a stream does not affect anyone else than the sender and single receiver of that stream.

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

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

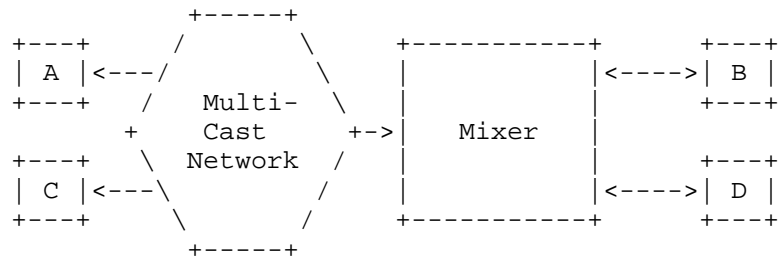


Figure 3: RTP Mixer in Point-to-Multipoint

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

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

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

3.4. Media Receiver to RTP Mixer

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

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

However, when the End Point requesting stream pause is connected to the RTP Mixer through a multicast network, such as A or C in Figure 3, the use case instead becomes identical to the one in Section 3.3, only with reverse direction of the streams and pause/resume requests.

3.5. Media Receiver to Media Sender Across RTP Mixer

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

1. The Mixer might not include CSRC in it's stream indications.
2. An End Point cannot rely on the CSRC to correctly identify the stream to be paused when the delivered media is some type of mix. A more elaborate stream identification solution is needed to support this in the general case.
3. The End Point cannot determine if a given stream is still needed by the RTP Mixer to deliver to another session participant.

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

4. Design Considerations

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

4.1. Real-time Nature

The first section (Section 1) of this specification describes some possible reasons why a receiver may pause an RTP sender. Pausing and resuming is time-dependent, i.e. a receiver may choose to pause an RTP stream for a certain duration, after which the receiver may want the sender to resume. This time dependency means that the messages related to pause and resume must be transmitted to the sender in real-time in order for them to be purposeful. The pause operation is arguably not very time critical since it mainly provides a reduction of resource usage. Timely handling of the resume operation is however likely to directly impact the end-user's perceived quality experience, since it affects the availability of media that the user expects to receive more or less instantly. It may also be highly desirable for a receiver to quickly learn that an RTP stream is intentionally paused on the RTP sender's own behalf.

4.2. Message Direction

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

4.3. Apply to Individual Sources

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

4.4. Consensus

An RTP stream sender should not pause an SSRC that some receiver still wishes to receive. The reason is that in RTP topologies where the stream is shared between multiple receivers, a single receiver on that shared network, independent of it being multicast, a mesh with joint RTP session or a transport Translator based, must not single-handedly cause the stream to be paused without letting all other receivers to voice their opinions on whether or not the stream should be paused. A consequence of this is that a newly joining receiver, for example indicated by an RTCP Receiver Report containing both a new SSRC and a CNAME that does not already occur in the session, firstly needs to learn the existence of paused streams, and secondly should be able to resume any paused stream. Any single receiver wanting to resume a stream should also cause it to be resumed. An important exception to this is when the RTP stream sender is aware of conditions that makes it desirable or even necessitates to pause the RTP stream on its own behalf, without being explicitly asked to do so. Such local consideration in the RTP sender takes precedence over RTP receiver wishes to receive the stream.

4.5. Message Acknowledgments

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

In some cases when a PAUSE or RESUME message reaches the RTP stream sender, it will not be able to pause or resume the stream due to some local consideration, for example lack of data to transmit. This

error condition, a negative acknowledgment, may be needed to avoid unnecessary retransmission of requests (Section 4.6).

4.6. Request Retransmission

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

When it comes to resume requests or unsolicited paused indications that are more time critical, the best performance may be achieved by repeating the message as often as possible until a sufficient number have been sent to reach a high probability of message delivery, or at an explicit indication that the message was delivered. For resume requests, such explicit indication can be delivery of the RTP stream being requested to be resumed.

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.

4.8. Relation to Other Solutions

A performance comparison between SIP/SDP and RTCP signaling technologies was made and included in draft versions of this specification. Using SIP and SDP [RFC4566] to carry pause and resume information means that it will need to traverse the entire signaling path to reach the signaling destination (either the remote End Point or the entity controlling the RTP Mixer), across any signaling proxies that potentially also has to process the SDP content to determine if they are expected to act on it. The amount of bandwidth required for a SIP/SDP-based signaling solution is in the order of at least 10 times more than an RTCP-based solution. Especially for UA sitting on mobile wireless access, this will risk introducing delays

that are too long (Section 4.1) to provide a good user experience, and the bandwidth cost may also be considered infeasible compared to an RTCP-based solution. RTCP data is sent through the media path, which is likely shorter (contains fewer intermediate nodes) than the signaling path, 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 considerations, RTCP is chosen as signaling protocol for the pause and resume functionality.

5. Solution Overview

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

It is proposed to re-use CCM TMMBR and TMMBN [RFC5104] to the extent possible, and to define a small set of new RTCP feedback messages where new semantics is needed.

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

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

Special considerations that apply when using TMMBR/TMMBN for pause and resume purposes are described in Section 5.5. This specification applies to both the new messages defined in herein as well as their TMMBR/TMMBN counterparts, except when explicitly stated otherwise. An obvious exception are any reference to the message parameters that are only available in the messages defined here. For example, any reference to PAUSE in the text below is equally applicable to TMMBR 0, and any reference to PAUSED is equally applicable to TMMBN 0. Therefore and for brevity, TMMBR/TMMBN will not be mentioned in the text, unless there is specific reason to do so.

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.

5.1. 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 only a subset of messages is possible, to support partial implementation, which is specifically believed to be feasible for the RTP Mixer to Media Sender use case (Section 3.2).

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

5.2. Requesting to Pause

An RTP 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 stream (SSRC).

When a non-paused RTP stream sender receives the PAUSE request, it continues to send the RTP stream while waiting for some time to allow other RTP stream receivers in the same RTP session that saw this PAUSE request to disapprove by sending a RESUME (Section 5.4) for the same stream and with the same PauseID as in the disapproved PAUSE. If such disapproving RESUME arrives at the RTP stream sender during the hold-off period before the stream is paused, the pause is not performed. In point-to-point configurations, the hold-off period may be set to zero. Using a hold-off period of zero is also appropriate when using TMMBR 0 and in line with the semantics for that message.

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

If the PAUSE request is lost before it reaches the RTP stream sender, it will be discovered by the RTP stream receiver because it continues to receive the RTP stream. It will also not see any PAUSED indication (Section 5.3) for the stream. The same condition can be caused by the RTP stream sender having received a disapproving RESUME from a stream receiver A for a PAUSE request sent by a stream sender B, but that the PAUSE sender (B) did not receive the RESUME (from A)

and may instead think that the PAUSE was lost. In both cases, a PAUSE request can be re-transmitted using the same PauseID. If using TMMBR 0 the request MAY be re-transmitted when the requester fails to receive a TMMBN 0 confirmation.

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

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

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

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

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

5.3. Media Sender Pausing

An RTP stream sender can choose to pause the stream at any time. This can either be as a result of receiving a PAUSE, or be based on some local sender consideration. When it does, it sends a PAUSED indication, containing the available PauseID. Note that PauseID is

incremented when sending an unsolicited PAUSED (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.

There is no reply to a PAUSED indication; it is simply an explicit indication of the fact that an RTP stream is paused. This can be helpful for the RTP stream receiver, for example to quickly understand that transmission is deliberately and temporarily suspended and no specific corrective action is needed.

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

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

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

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

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

5.4. Requesting to Resume

An RTP stream receiver can request to resume a stream with a RESUME request at any time, subject to AVPF timing rules. The RTP stream receiver must include the available PauseID in the RESUME request for it to be effective.

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

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

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

A pausing RTP stream sender can apply local considerations and MAY resume a paused RTP stream at any time. If TMMBR 0 was used to pause the RTP stream, it cannot be resumed due to local considerations, unless the RTP stream is paused only due to local considerations (Section 5.3) and thus no RTP stream receiver has requested to pause the stream with TMMBR 0.

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

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

5.5. TMMBR/TMMBN Considerations

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

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

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

TMMBR >0: Corresponds to RESUME when the RTP stream was previously paused with TMMBR 0. Since there is only a single RTP stream receiver, there is no need for the RTP stream sender to delay resuming the stream until after sending TMMBN >0, or to apply the hold-off period specified in [RFC5104] before increasing the bitrate from zero. The bitrate value used when resuming after pausing with TMMBR 0 is either according to known limitations, or based on starting a stream with the configured maximum for the stream or session, for example given by b-parameter in SDP.

TMMBN 0: Corresponds to PAUSED when the RTP stream was paused with TMMBR 0, but may, just as PAUSED, also be used unsolicited. An unsolicited RTP stream pause based on local sender considerations uses the RTP stream's own SSRC as TMMBR restriction owner in the TMMBN message bounding set. Also corresponds to a REFUSED indication when a stream is requested to be resumed with TMMBR >0.

TMMBN >0: Cannot be used as REFUSED indication when a stream is requested to be paused with TMMBR 0, for reasons stated in Section 5.2.

6. Participant States

This document introduces three new states for a stream in an RTP sender, according to the figure and sub-sections below. Any references to PAUSE, PAUSED, RESUME and REFUSED in this section SHALL be taken to apply to the extent possible also when TMMBR/TMMBN are used (Section 5.5) for this functionality.

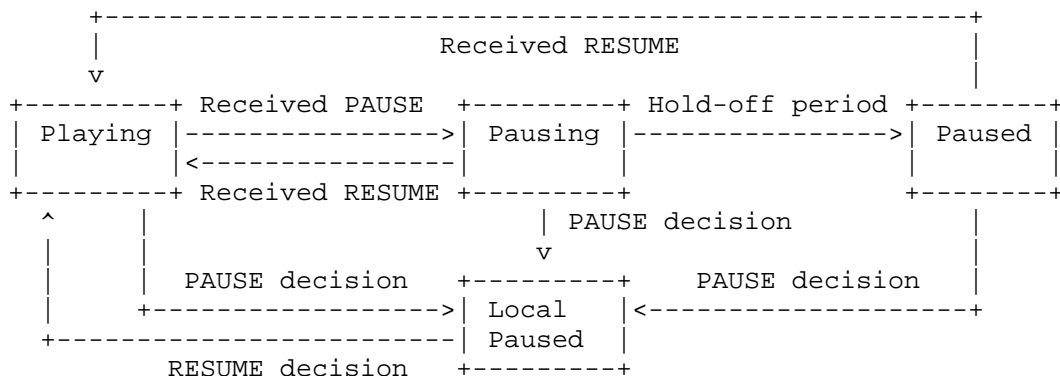


Figure 4: RTP Pause States

6.1. 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, meaning the RTP Time Stamp difference compared to before the pause reflects the time the RTP stream was paused.

6.2. Pausing State

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

$$2 * RTT + T_dither_max,$$

where RTT is the longest round trip known to the RTP stream sender and T_dither_max is defined in section 3.4 of [RFC4585]. The hold-off period MAY be set to 0 by some signaling (Section 9) means when it can be determined that there is only a single receiver, for example in point-to-point or some unicast situations.

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

6.3. Paused State

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

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

Pausing an RTP stream MUST NOT affect the sending of RTP keepalive [RFC6263][RFC5245] applicable to that RTP stream.

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.

6.3.1. 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 stream to pause. If that receiver sends an RTCP BYE message observed by the sender, the sender SHALL resume the RTP stream.

6.3.2. 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.

6.4. Local Paused State

This state can be entered at any time, based on local decision from the RTP stream sender. As for Paused State (Section 6.3), the RTP stream sender SHALL send a PAUSED indication to all known RTP stream receivers, when entering the state, and repeat it a sufficient number of times to reach a high probability that the message is correctly delivered, unless the stream was already in paused state (Section 6.3).

Editor's note: Consider specifying an explicit PAUSED ACK message that stops this message retransmission.

When using TMMBN 0 as PAUSED indication, being in paused state, and entering local paused state, the RTP stream sender SHALL send TMMBN 0 with itself included in the TMMBN bounding set.

As indicated in Figure 4, this state has higher precedence than paused state (Section 6.3) and RESUME messages alone cannot resume a paused RTP stream as long as the local decision still applies.

Pausing an RTP stream MUST NOT affect the sending of RTP keepalive [RFC6263][RFC5245] applicable to that RTP stream.

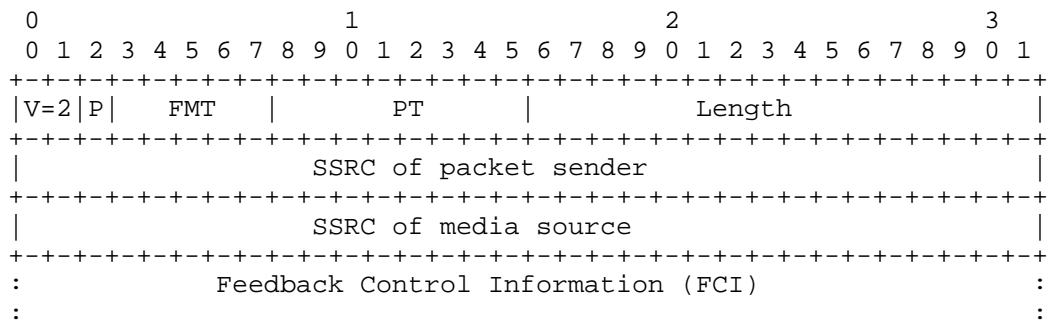
When leaving the state, the stream state SHALL become Playing, regardless whether or not there were any RTP stream receivers that sent PAUSE for that stream, effectively clearing the RTP stream sender's memory for that stream. This does however not apply when the stream was paused by a TMMBN 0, either before entering or during the Local Paused State, in which case leaving Local Paused State just removes the RTP sender from the TMMBN bounding set, and a new TMMBN with the updated bounding set MUST be sent accordingly. The stream state can become Playing only when there is no entry with a bitrate value of 0 in the stream's bounding set.

7. Message Format

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

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

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



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

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

PT: Payload Type = 205 (RTPFB)

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

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

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

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

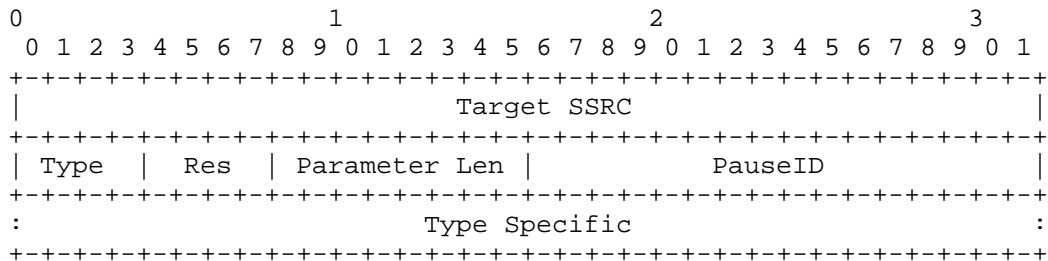


Figure 5: Syntax of FCI Entry in the PAUSE and RESUME message

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 REFUSED, 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: REFUSED 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.

8. 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 8.5.

Any references to PAUSE, PAUSED, RESUME and REFUSED in this section SHALL be taken to apply to the extent possible also when TMMBR/TMMBN are used (Section 5.5) for this functionality. TMMBR/TMMBN MAY be used instead of the messages defined in this specification when the effective topology is point-to-point. If either sender or receiver learns that the topology is not point-to-point, TMMBR/TMMBN MUST NOT be used for pause/resume functionality. If the messages defined in this specification are supported in addition to TMMBR/TMMBN, pause/resume signaling MUST use messages from this specification. If the topology is not point-to-point and the messages defined in this specification are not supported, pause/resume functionality with TMMBR/TMMBN MUST NOT be used.

8.1. PAUSE

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

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

PauseID SHOULD be the available PauseID, as indicated by PAUSED (Section 8.2) or implicitly determined by previously received PAUSE or RESUME (Section 8.3) requests. A randomly chosen PauseID MAY be used if it was not possible to retrieve PauseID information, in which case the PAUSE will either succeed, or the correct PauseID can be found in the returned REFUSED (Section 8.4). A PauseID that is matching the available PauseID is henceforth also called a valid PauseID.

PauseID needs to be incremented by one, in modulo arithmetic, for each PAUSE request that is not a retransmission, compared to what was

used in the last PAUSED indication sent by the media sender. This is to ensure that the PauseID matches what is the current available PauseID at the RTP stream sender. The RTP stream sender increments what it considers to be the available PauseID when entering Playing State (Section 6.1).

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

If the targeted RTP stream does not pause, if no PAUSED indication with a larger PauseID than the one used in PAUSE, and if no REFUSED 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 stream sender and T_dither_max is defined in section 3.4 of [RFC4585].

When an RTP stream sender in Playing State (Section 6.1) receives a valid PAUSE, and unless local considerations currently makes it impossible to pause the stream, it SHALL enter Pausing State (Section 6.2) when reaching an appropriate place to pause in the stream, and act accordingly.

If an RTP stream sender receives a valid PAUSE while in Pausing, Paused (Section 6.3) or Local Paused (Section 6.4) States, the received PAUSE SHALL be ignored.

8.2. PAUSED

The PAUSED indication MUST be sent whenever entering Paused State (Section 6.3) as a result of receiving a valid PAUSE (Section 8.1) request, or when entering Local Paused State (Section 6.4) based on a RTP stream sender local decision.

PauseID MUST contain the available, valid value to be included in a subsequent RESUME (Section 8.3).

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

After having entered Paused or Local Paused State and thus having sent PAUSED once, PAUSED MUST also be included in the next two regular RTCP reports, given that the pause condition is then still effective.

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

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

8.3. RESUME

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

PauseID SHOULD be the valid PauseID, as indicated by PAUSED (Section 8.2) or implicitly determined by previously received PAUSE (Section 8.1) or RESUME requests. A randomly chosen PauseID MAY be used if it was not possible to retrieve PauseID information, in which case the RESUME will either succeed, or the correct PauseID can be found in a returned REFUSED (Section 8.4).

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

When an RTP stream sender in Pausing (Section 6.2), Paused (Section 6.3) or Local Paused State (Section 6.4) receives a valid RESUME, and unless local considerations currently makes it impossible to resume the stream, it SHALL enter Playing State (Section 6.1) and act accordingly. If the RTP stream sender is incapable of honoring the RESUME request with a valid PauseID, or receives a RESUME request with an invalid PauseID while in Paused or Pausing state, the RTP stream sender sends a REFUSED message as specified below.

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

8.4. REFUSED

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

If an RTP stream sender receives a valid PAUSE (Section 8.1) or RESUME (Section 8.3) request that cannot be fulfilled by the sender due to some local consideration, it SHALL schedule transmission of a REFUSED indication containing the valid PauseID from the rejected request.

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

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

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

An RTP stream receiver that sent a PAUSE or RESUME request and receives a REFUSED 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 REFUSED to clear.

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

8.5. 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, as well as TMMBR/TMMBN used for pause/resume purposes (Section 5.5), 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. Unsolicited PAUSED (sent when entering Local Paused State (Section 6.4)) SHOULD always use Immediate or Early timing, until PAUSED for that PauseID is considered delivered at least once to all receivers of the paused RTP stream, after which it SHOULD use Regular timing.

Editor's note: Consider specifying a PAUSED ACK message as explicit indication of reception.

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

9. Signaling

The capability of handling messages defined in this specification MAY be exchanged at a higher layer such as SDP. This document extends the rtcp-fb attribute defined in section 4 of AVPF [RFC4585] to include the request for pause and resume. This specification follows all the rules defined in AVPF [RFC4585] and CCM [RFC5104] for an rtcp-fb attribute relating to payload type in a session description.

This specification defines a new parameter "pause" to the "ccm" feedback value defined in CCM [RFC5104], representing the capability to understand the RTCP feedback message and all of the defined FCIs of PAUSE, RESUME, PAUSED and REFUSED.

Note: When TMMBR 0 / TMMBN 0 are used to implement pause and resume functionality (with the restrictions described in this specification), signaling rtcp-fb attribute with ccm tmmbr parameter is sufficient and no further signaling is necessary. There is however no guarantee that TMMBR/TMMBN implementations pre-dating this specification work exactly as described here when used with a bitrate value of 0.

The "pause" parameter has two optional attributes, "nowait" and "config":

- o "nowait" indicates that the hold-off period defined in Section 6.2 can be set to 0, reducing the latency before the stream can be paused after receiving a PAUSE request. This condition occurs when there will be only a single receiver per direction in the RTP session, for example in point-to-point sessions. It is also possible to use in scenarios using unidirectional media. The conditions that allow "nowait" to be set also indicate that it would be possible to use CCM TMMBR/TMMBN as pause/resume signaling.
- o "config" allows for partial implementation of this specification according to the different roles in the use cases section (Section 3), and takes a value that describes what sub-set is implemented:
 - 1 Full implementation of this specification. This is the default configuration. A missing config attribute MUST be treated equivalent to providing a config value of 1.
 - 2 The implementation intends to send PAUSE and RESUME requests for received RTP streams and is thus also capable of receiving PAUSED and REFUSED. It does not support receiving PAUSE and RESUME requests, but may pause sent RTP streams due to local considerations and then intends to send PAUSED for them.
 - 3 The implementation supports receiving PAUSE and RESUME requests targeted for RTP streams it sends. It will send PAUSED and REFUSED as needed. The node will not send any PAUSE and RESUME requests, but supports and desires receiving PAUSED if received RTP streams are paused.
 - 4 The implementation intends to send PAUSE and RESUME requests for received RTP streams and is thus also capable of receiving PAUSED and REFUSED. It cannot pause any RTP streams it sends, and thus does not support receiving PAUSE and RESUME requests, and also does not support sending PAUSED indications.
 - 5 The implementation supports receiving PAUSE and RESUME requests targeted for RTP streams it sends. It will send PAUSED and REFUSED as needed. It does not support sending PAUSE and RESUME requests to pause received RTP streams, and also does not support receiving PAUSED indications.
 - 6 The implementation supports sent and received RTP streams being paused due to local considerations, and thus supports sending and receiving PAUSED indications.
 - 7 The implementation supports and desires to receive PAUSED indications for received RTP streams, but does not pause or

send PAUSED indications for sent RTP streams. It does not support any other messages defined in this specification.

- 8 The implementation supports pausing sent RTP streams and sending PAUSED indications for them, but does not support receiving PAUSED indications for received RTP streams. It does not support any other messages defined in this specification.

When signaling a config value other than 1, an implementation MAY ignore non-supported messages on reception, and MAY omit sending non-supported messages. The below table summarizes per-message send and receive support for the different config attribute values ("X" indicating support and "-" indicating non-support):

#	Send				Receive			
	PAUSE	RESUME	PAUSED	REFUSED	PAUSE	RESUME	PAUSED	REFUSED
1	X	X	X	X	X	X	X	X
2	X	X	X	-	-	-	X	X
3	-	-	X	X	X	X	X	-
4	X	X	-	-	-	-	X	X
5	-	-	X	X	X	X	-	-
6	-	-	X	-	-	-	X	-
7	-	-	-	-	-	-	X	-
8	-	-	X	-	-	-	-	-

Figure 6: Supported messages for different config values

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]
pause-attr       = [pause-config] [SP "nowait"] [SP byte-string]
pause-config     = "config=" pause-config-value
pause-config-value = %x31-38
; byte-string as defined in RFC 4566, for future extensions
```

Figure 7: ABNF

An endpoint implementing this specification and using SDP to signal capability SHOULD indicate the new "pause" parameter with ccm signaling, but MAY use existing ccm tmmbr signaling [RFC5104] if the limitations in functionality as described in this specification coming from such usage are considered acceptable. The messages from

this specification SHOULD NOT be used towards receivers that did not declare capability to receive those messages.

There MUST NOT be more than one "a=rtcp-fb" line with "pause" 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" specification.

9.1. Offer-Answer Use

An offerer implementing this specification needs to include "pause" CCM parameter with suitable configuration attribute ("config") in the SDP, according to what messages it intends to send and desires to receive in the session.

In SDP offer/answer, the "config" attribute and its message directions are interpreted based on the agent providing the SDP. The offerer is described in an offer, and the answerer is described in an answer.

An answerer receiving an offer with a "pause" CCM parameter and a config attribute with a certain value, describing a certain capability to send and receive messages, MAY change the config attribute value in the answer to another configuration. The permitted answers are listed in the below table.

SDP Offer config value	Permitted SDP Answer config values
1	1, 2, 3, 4, 5, 6, 7, 8
2	3, 4, 5, 6, 7, 8
3	2, 4, 5, 6, 7, 8
4	5, 6, 7, 8
5	4, 6, 7, 8
6	6, 7, 8
7	8
8	7

Figure 8: Config values in Offer/Answer

An offer or answer omitting the config attribute, MUST be interpreted as equivalent to config=1. In all cases the answerer MAY also completely remove any "pause" CCM parameter to indicate that it does not understand or desire to use any pause functionality for the affected payload types.

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 "pause" line in the offer. If both offer and answer contained a "nowait" parameter, then the hold-off period is configured to 0 at both offerer and answerer.

9.2. Declarative Use

In declarative use, the SDP is used to configure the node receiving the SDP. This has implications on the interpretation of the SDP signaling extensions defined in this specification.

First, the "config" attribute and its message directions are interpreted based on the node receiving the SDP.

Second, 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.

10. 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

10.1. 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
```

Figure 9: SDP Offer With Pause and Resume Capability

The offerer supports all of the messages defined in this specification, leaving out the optional config attribute. 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 the "pause" parameter.

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 config=2
```

Figure 10: SDP Answer With Pause and Resume Capability

The answerer will not allow its sent streams to be paused or resumed and thus restricts the answer to indicate config=2. It also supports pausing its own RTP streams due to local considerations, which is why config=2 is chosen rather than config=4. 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 period when being requested to pause the stream.

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

10.2. 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 period before pausing a stream is 0.

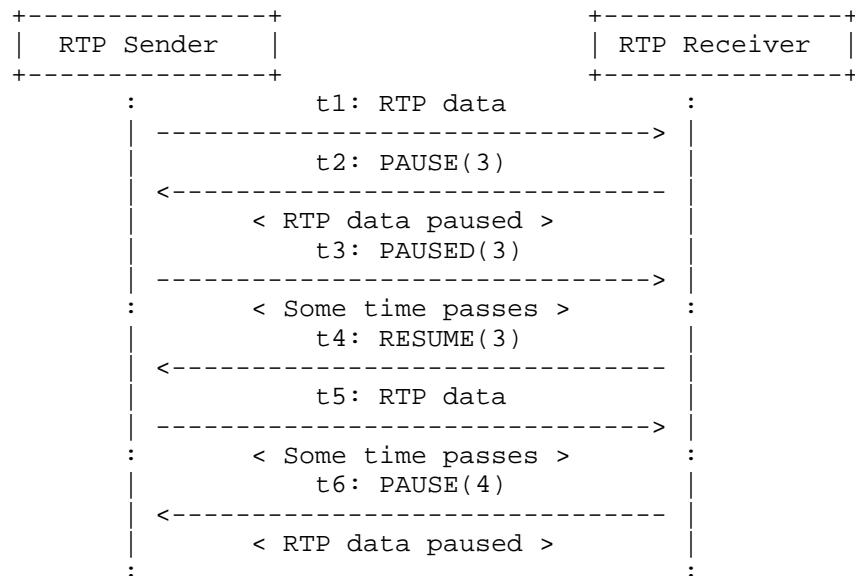


Figure 11: Pause and Resume Operation in Point-to-Point

Figure 11 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).

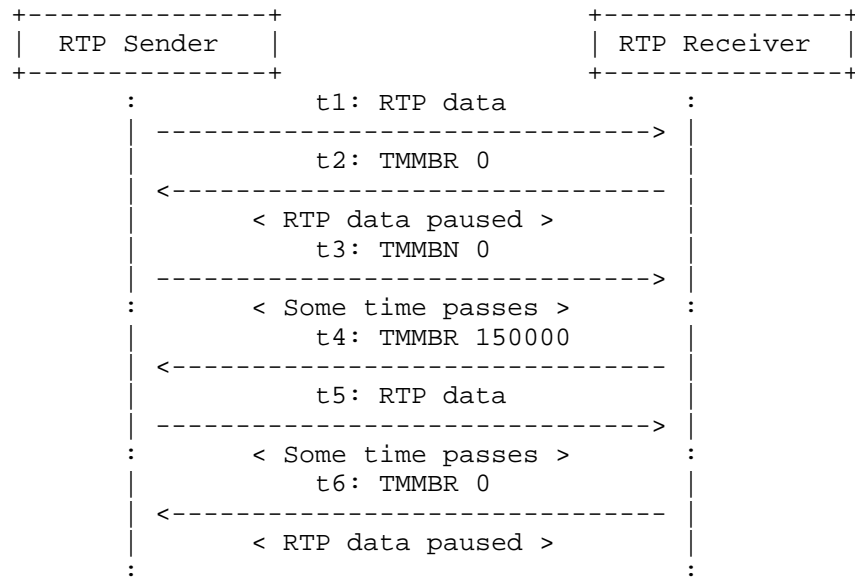


Figure 12: TMMBR Pause and Resume in Point-to-Point

Figure 12 describes the same point-to-point scenario as above, but using TMMBR/TMMBN signaling.

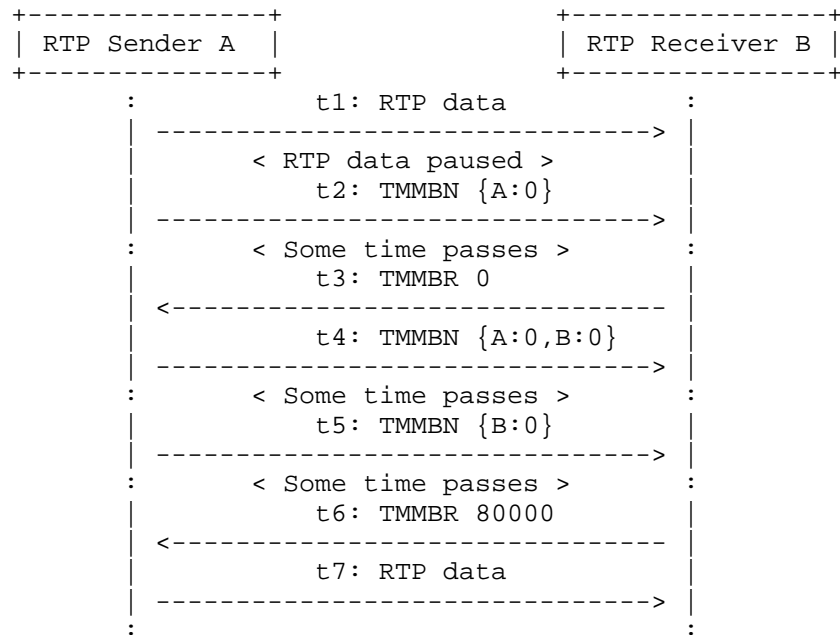


Figure 13: Unsolicited PAUSED using TMMBN

Figure 13 describes the case when an RTP stream sender (A) chooses to pause an RTP stream due to local considerations. Both the RTP stream sender (A) and the RTP stream receiver (B) use TMMBR/TMMBN signaling for pause/resume purposes. A decides to pause the RTP stream at time t2 and uses TMMBN 0 to signal PAUSED, including itself in the TMMBN bounding set. At time t3, despite the fact that the RTP stream is still paused, B decides that it is no longer interested to receive the RTP stream and signals PAUSE by sending a TMMBR 0. As a result of that, the bounding set now contains both A and B, and A sends out a new TMMBN reflecting that. After a while, at time t5, the local considerations that caused A to pause the RTP stream no longer apply, causing it to remove itself from the bounding set and to send a new TMMBN indicating this. At time t6, B decides that it is now interested to receive the RTP stream again and signals RESUME by sending a TMMBR containing a bitrate value greater than 0, causing A to resume sending RTP data.

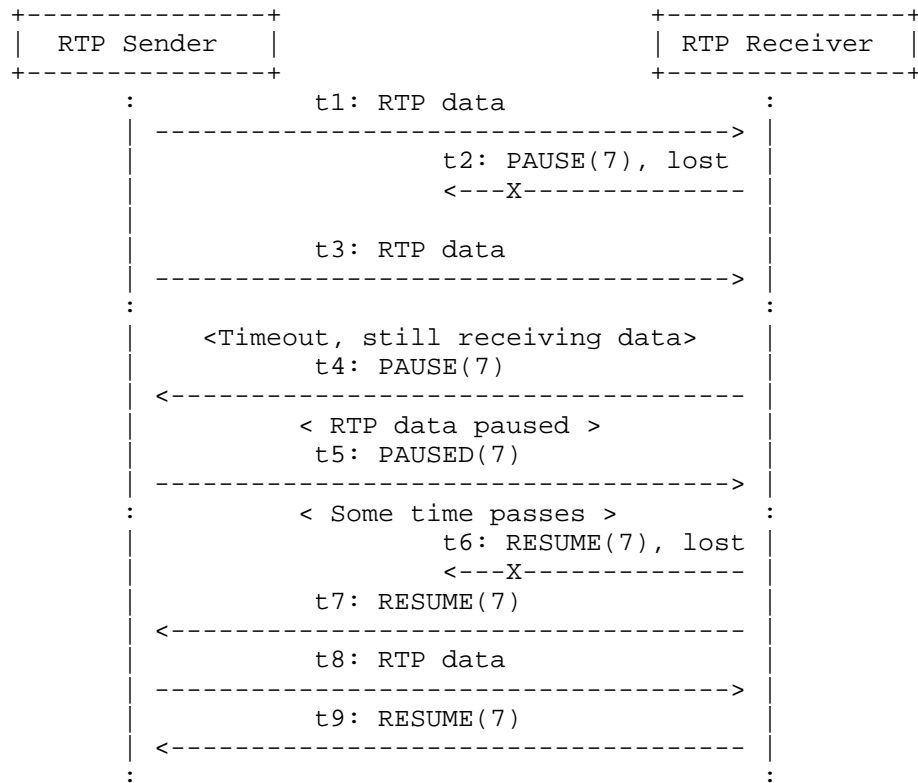


Figure 14: Pause and Resume Operation With Messages Lost

Figure 14 describes what happens if a PAUSE message from an RTP stream receiver does not reach the RTP stream sender. After sending a PAUSE message, the RTP stream receiver waits for a time-out to detect if the RTP stream sender has paused the data transmission or has sent PAUSED indication according to the rules discussed in Section 6.3. As the PAUSE message is lost on the way (at time t2), RTP data continues to reach to the RTP stream receiver. When the timer expires, the RTP stream receiver schedules a retransmission of the PAUSE message, which is sent at time t4. If the PAUSE message now reaches the RTP stream sender, it pauses the RTP stream and replies with PAUSED.

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

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

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

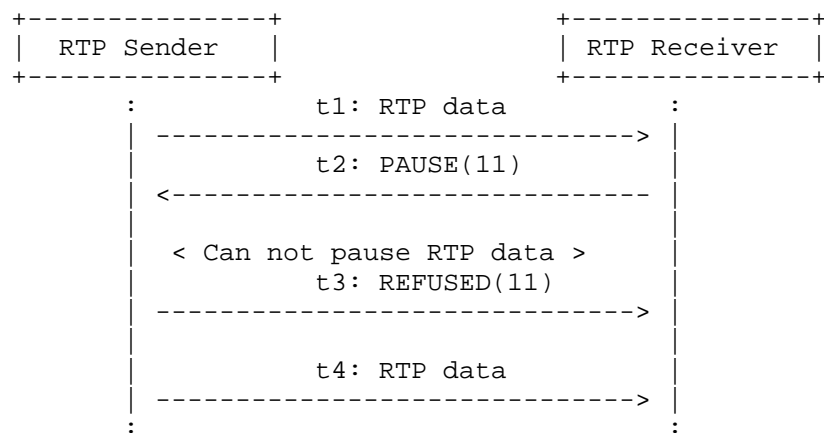


Figure 15: Pause Request is Refused in Point-to-Point

In Figure 15, the receiver requests to pause the sender, which refuses to pause due to some consideration local to the sender and responds with a REFUSED message.

10.3. Point-to-Multipoint using Mixer

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

The Mixer keeps track of all the streams delivered to the Mixer and how they are currently used. In this example, it selects the video

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

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

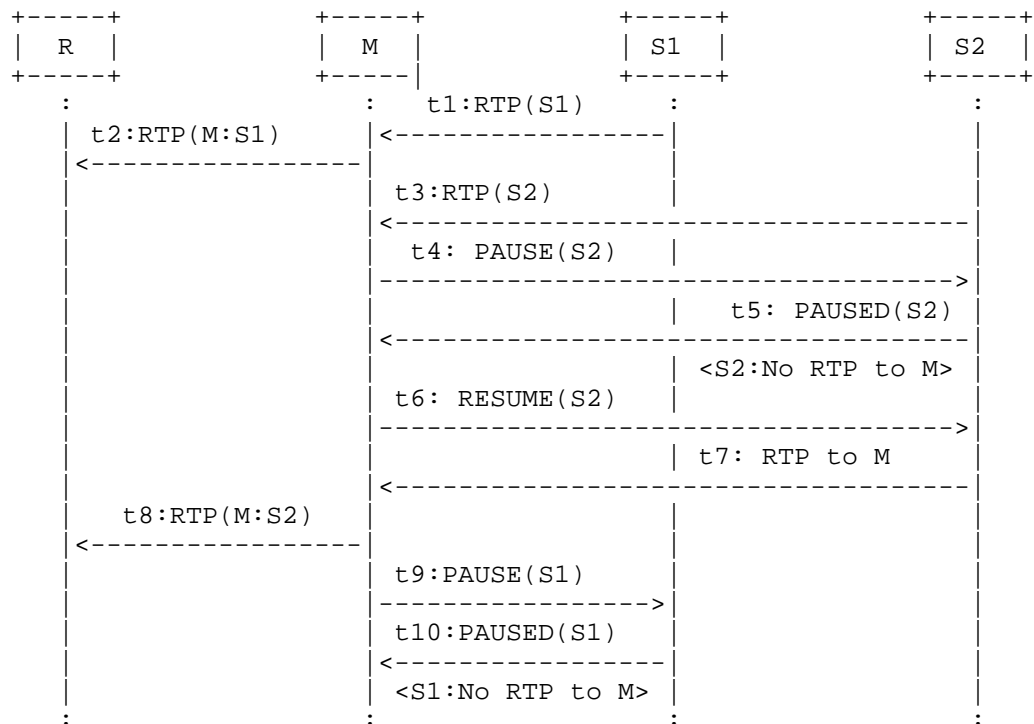


Figure 16: Pause and Resume Operation for a Voice Activated Mixer

The session starts at t1 with S1 being the most active speaker and thus being selected as the single video stream to be delivered to R (t2) using the Mixer SSRC but with S1 as CSRC (indicated after the colon in the figure). Then S2 joins the session at t3 and starts delivering an RTP stream to the Mixer. As S2 has less voice activity 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 stream delivery to M. When the RTP

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

10.4. 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.

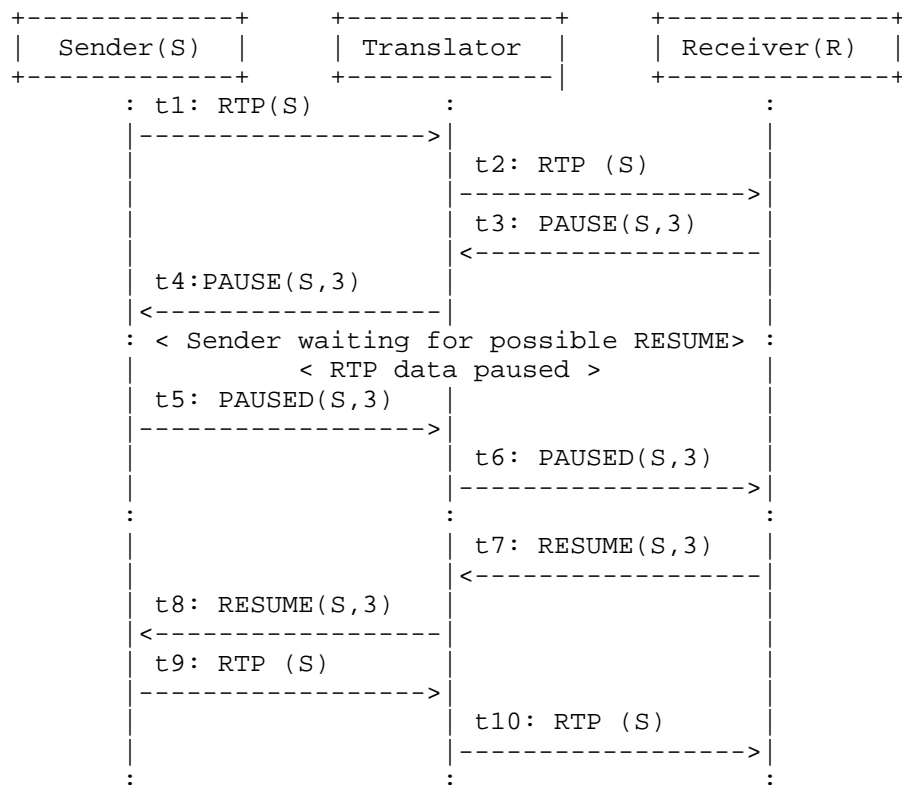


Figure 17: Pause and Resume Operation Between Two Participants Using a Translator

Figure 17 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 period to see if there is any other receiver that wants to receive the data, does not receive any disapproving RESUME, hence pauses itself and replies with PAUSED. Similarly the receiver resumes the sender by sending RESUME request through Translator. Since this describes only a single pause operation for a single RTP stream sender, all messages uses a single PauseID, in this example 3.

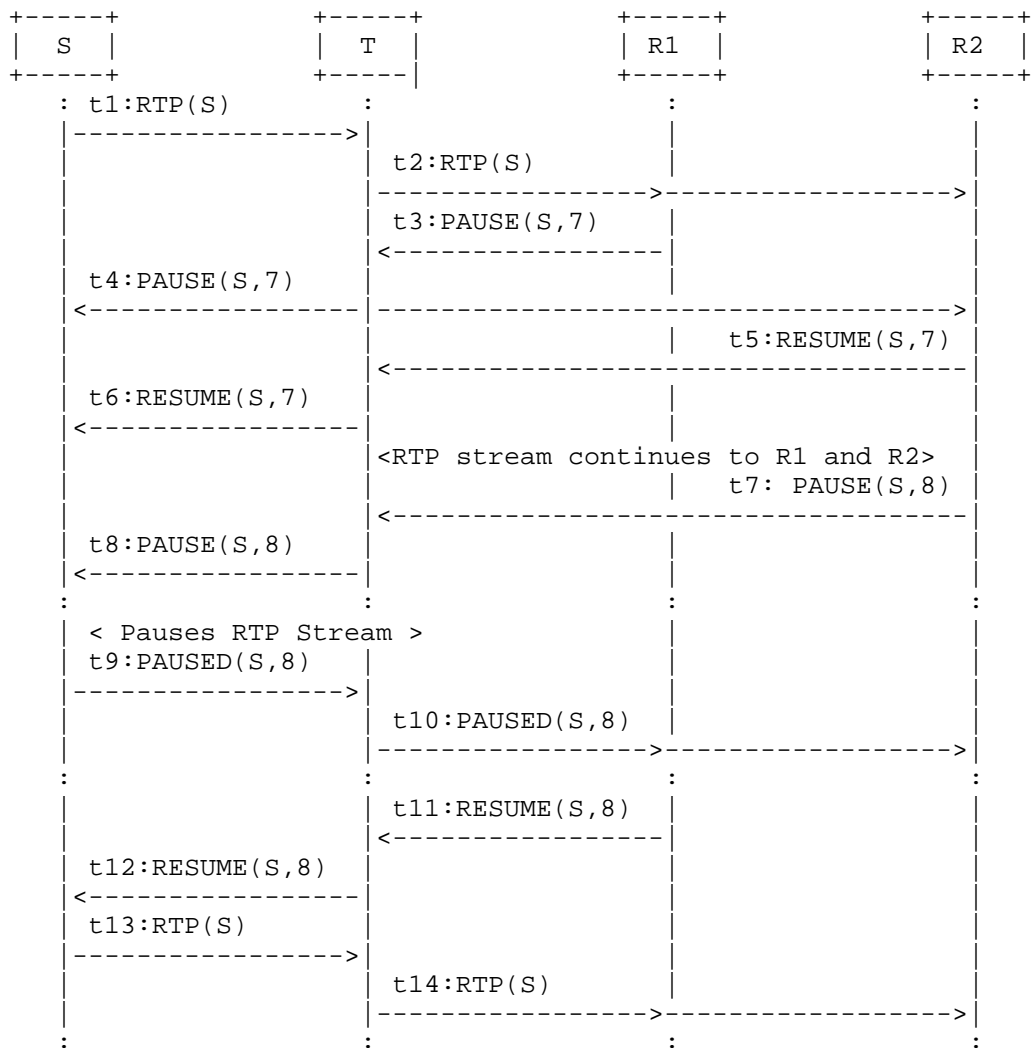


Figure 18: Pause and Resume Operation Between One Sender and Two Receivers Through Translator

Figure 18 explains the pause and resume operations when a transport Translator is involved between a sender and two receivers in an RTP session. Each message exchange is represented by the time it happens. At time t1, Sender (S) starts sending an RTP stream to the Translator, which is forwarded to R1 and R2 through the Translator, T. R1 and R2 receives RTP data from Translator at t2. At this

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

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

11. IANA Considerations

This specification requests the following registrations from IANA:

1. A new value for media stream pause / resume to be registered with IANA in the "FMT Values for RTPFB Payload Types" registry located at the time of publication at: <http://www.iana.org/assignments/rtp-parameters/rtp-parameters.xhtml#rtp-parameters-8>

Value: TBA1

Name: PAUSE-RESUME

Long Name: Media Pause / Resume

Reference: This RFC

2. A new value "pause" to be registered with IANA in the "Codec Control Messages" registry located at the time of publication at: <http://www.iana.org/assignments/sdp-parameters/sdp-parameters.xhtml#sdp-parameters-19>

Value Name: pause

Long Name: Media Pause / Resume

Usable with: ccm

Reference: This RFC

12. Security Considerations

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

1. Identity spoofing - An attacker can spoof him/herself as an authenticated user and can falsely pause or resume any source transmission. In order to prevent this type of attack, a strong authentication and integrity protection mechanism is needed.
2. Denial of Service (DoS) - An attacker can falsely pause all source streams which MAY result in Denial of Service (DoS). An Authentication protocol may prevent this attack.
3. Man-in-Middle Attack (MiMT) - The pausing and resuming of an RTP source is prone to a Man-in-Middle attack. Public key authentication may be used to prevent MiMT.

13. Contributors

Daniel Grondal contributed in the creation and writing of early versions of this specification. Christian Groves contributed significantly to the SDP config attribute and its use in Offer/Answer.

14. Acknowledgements

Daniel Grondal made valuable contributions during the initial versions of this draft. Emil Ivov, Christian Groves and Bernard Aboba provided valuable review comments.

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.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.
- [RFC6263] Marjou, X. and A. Sollaud, "Application Mechanism for Keeping Alive the NAT Mappings Associated with RTP / RTP Control Protocol (RTCP) Flows", RFC 6263, June 2011.

15.2. Informative References

- [I-D.ietf-avtcore-rtp-topologies-update]
Westerlund, M. and S. Wenger, "RTP Topologies", draft-ietf-avtcore-rtp-topologies-update-04 (work in progress), August 2014.
- [I-D.ietf-avtext-rtp-grouping-taxonomy]
Lennox, J., Gross, K., Nandakumar, S., and G. Salgueiro, "A Taxonomy of Grouping Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", draft-ietf-avtext-rtp-grouping-taxonomy-02 (work in progress), June 2014.

- [I-D.ietf-rtcweb-use-cases-and-requirements]
Holmberg, C., Hakansson, S., and G. Eriksson, "Web Real-Time Communication Use-cases and Requirements", draft-ietf-rtcweb-use-cases-and-requirements-14 (work in progress), February 2014.
- [I-D.westerlund-avtcore-rtp-simulcast]
Westerlund, M. and S. Nandakumar, "Using Simulcast in RTP Sessions", draft-westerlund-avtcore-rtp-simulcast-04 (work in progress), July 2014.
- [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.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC6190] Wenger, S., Wang, Y., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, May 2011.

Appendix A. Changes From Earlier Versions

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

A.1. Modifications Between Version -04 and -05

- o Added text in sections 4.1, 4.6, 6.4 and 8.5 on retransmission and timing of unsolicited PAUSED, to improve the message timeliness and probability of reception.

A.2. Modifications Between Version -03 and -04

- o Change of Copyright boilerplate

A.3. Modifications Between Version -02 and -03

- o Changed the section on SDP signaling to be more explicit and clear in what is supported, replacing the 'paused' parameter and the 'dir' attribute with a 'config' parameter that can take a value, and an explicit listing of what each value means.
- o Added a sentence in section on paused state (Section 6.3) that pause must not affect RTP keepalive.
- o Replaced REFUSE message name with REFUSED throughout, to better indicate that it is not a command but a notification.
- o Added text in a few places, clarifying that PAUSED message may be used unsolicited due to RTP sender local considerations, and also clarified the interaction between this usage and an RTP stream receiver pausing the stream. Also added an example describing this case.
- o Clarified that when TMMBN 0 is used as PAUSED message, and when sent unsolicited due to RTP sender local considerations, the TMMBN message includes the RTP stream sender itself as part of the bounding set.
- o Clarified that there is no reply to a PAUSED indication.
- o Improved the IANA section.
- o Editorial improvements.

A.4. Modifications Between Version -01 and -02

- o Replaced most text on relation with other signaling technologies in previous section 5 with a single, summarizing paragraph, as discussed at IETF 90 in Toronto, and placed it as the last subsection of section 4 (design considerations).
- o Removed unused references.

A.5. Modifications Between Version -00 and -01

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

Authors' Addresses

Bo Burman
Ericsson
Kistavagen 25
SE - 164 80 Kista
Sweden

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

Azam Akram
Ericsson
Farogatan 6
SE - 164 80 Kista
Sweden

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

Roni Even
Huawei Technologies
Tel Aviv
Israel

Email: roni.even@mail01.huawei.com

Magnus Westerlund
Ericsson
Farogatan 6
SE- 164 80 Kista
Sweden

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

AVTEXT Working Group
INTERNET-DRAFT
Intended Status: Standards Track
Expires: January 30, 2015

J. Xia
R. Even
R. Huang
Huawei
L. Deng
China Mobile
July 29, 2014

RTP/RTCP Extension for RTP Splicing Notification
draft-ietf-avtext-splicing-notification-00

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 to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

Copyright and License Notice

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

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

Table of Contents

1	Introduction	3
1.1	Terminology	3
2	Overview of RTP Splicing Notification	4
3	Conveying Splicing Interval in RTP/RTCP extensions	5
3.1	RTP Header Extension	5
3.2	RTCP Splicing Notification Message	6
4	Reducing Splicing Latency	7
5	Failure Cases	7
6	SDP Signaling	8
7	Security Considerations	9
8	IANA Considerations	9
8.1	RTCP Control Packet Types	9
8.2	RTP Compact Header Extensions	10
9	Acknowledges	10
10	References	10
10.1	Normative References	10
10.2	Informative References	11
	Authors' Addresses	11

1 Introduction

Splicing is a process that replaces some multimedia content with other multimedia content and delivers the substitutive multimedia content to the receivers for a period of time. In some predictable splicing cases, e.g., advertisement insertion, the splicing duration MUST be inside of the specific, pre-designated time slot. Certain timing information about when to start and end the splicing must be first acquired by the mixer to start the splicing. This document refers to this information as Splicing Interval.

[SCTE35] provides a method that encapsulates the Splicing Interval 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 Interval 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 Interval by including it in the RTP packets.

Nevertheless, the Splicing Interval 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 Interval to the mixer even though RTCP is inherently unreliable too.

1.1 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 RFC 2119 [RFC2119].

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

Splicing Interval:

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.

2 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 Interval from the main RTP sender before splicing starts.

When a new splicing is forthcoming, the main RTP sender **MUST** send the Splicing Interval to the mixer. Usually, the Splicing Interval **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 Interval far enough in advance. Alternatively, the main RTP sender can estimate when to send the Splicing Interval 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 Interval from the main RTP sender in advance, and thus estimates when to transfer the substitutive content to the mixer. The Splicing Interval 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 Interval 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, and the timestamp of the first main RTP packet on the splicing out point.

When the substitutive RTP sender gets the Splicing Interval, 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 Interval. 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 Interval.

3 Conveying Splicing Interval in RTP/RTCP extensions

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

3.1 RTP Header Extension

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

One variant is defined for this header extension. It carries the 7 octets splicing-out NTP timestamp (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]), followed by the 8 octets splicing-in NTP timestamp (64-bit NTP-format timestamp as defined in [RFC5905]). The top 8 bits of the splicing-out NTP timestamp are referred from the top 8 bits of the splicing-in NTP timestamp, under the consumption that the splicing-out time is after the splicing-in time, and the splicing interval is less than 2^{25} seconds, this order allows full resolution for splicing-in NTP timestamp while keeping 4 octets alignment.

The format is shown in Figures 1.

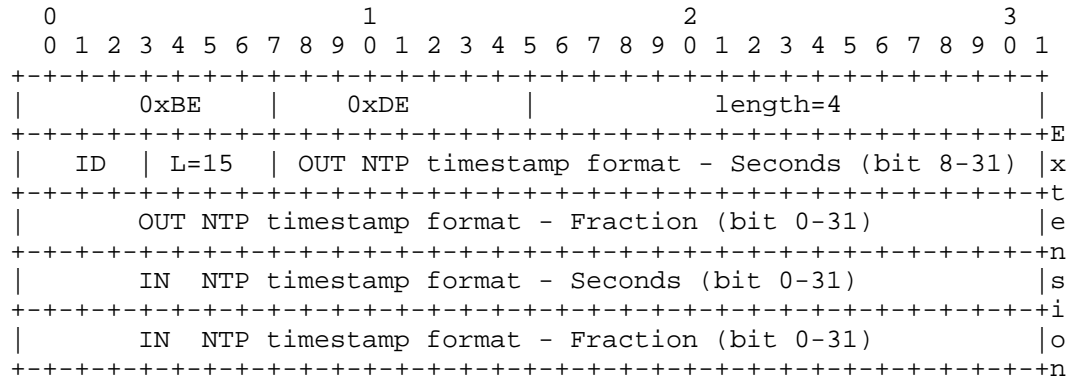


Figure 1: Sample hybrid NTP Encoding Using the One-Byte Header Format

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 Interval, 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.

3.2 RTCP Splicing Notification Message

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

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

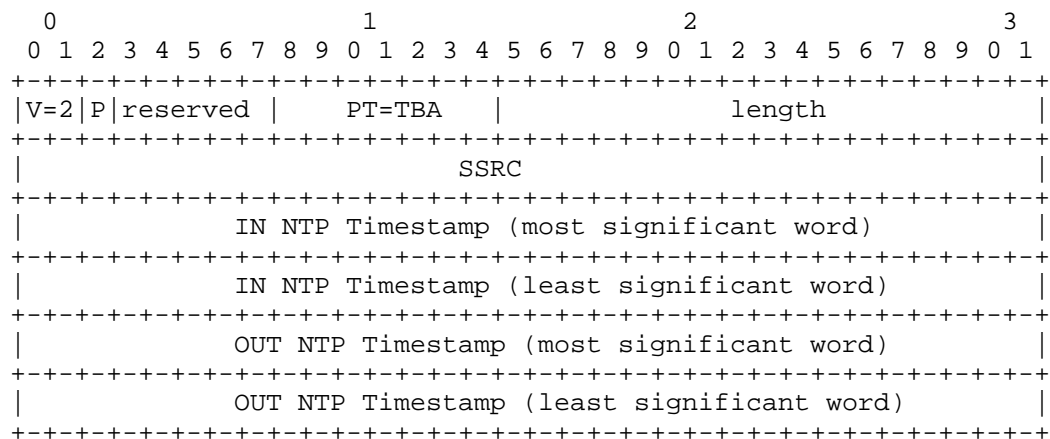


Figure 2: RTCP Splicing Notification Message

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]).

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].

4 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.

5 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 Interval reaches the mixers.

If the mixer does not get the Splicing Interval 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].

6 SDP Signaling

This document defines the URI for declaring this header extension in an extmap attribute to be "urn:ietf:params:rtp-hdext:splicing-interval".

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=extmap:1 urn:ietf:params:rtp-hdext:splicing-interval
```

```
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.

7 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].

8 IANA Considerations

8.1 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: TBA

Reference: This document

8.2 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-interval

Description: Splicing Interval

Contact: Jinwei Xia <xiajinwei@huawei.com>

Reference: This document

9 Acknowledges

TBD

10 References

10.1 Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [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., Ed., 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.

10.2 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.

[RFC6904] Lennox, J., "Encryption of Header Extensions in the Secure Real-Time Transport Protocol (SRTP)", April 2013.

[SCTE35] Society of Cable Telecommunications Engineers (SCTE), "Digital Program Insertion Cueing Message for Cable", 2011.

Authors' Addresses

Jinwei Xia
Huawei

Email: xiajinwei@huawei.com

Roni Even
Huawei

Email: ron.even.tlv@gmail.com

Rachel Huang
Huawei

Email: rachel.huang@huawei.com

Lingli Deng
China Mobile

Email: denglingli@chinamobile.com

Payload Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 30, 2015

J. Uberti
S. Holmer
M. Flodman
Google
J. Lennox
Vidyo
October 27, 2014

RTP Payload Format for VP9 Video
draft-uberti-payload-vp9-00

Abstract

This memo describes an RTP payload format for the VP9 video codec. The payload format has wide applicability, as it supports applications from low bit-rate peer-to-peer usage, to high bit-rate video conferences. It includes provisions for temporal and spatial scalability.

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 30, 2015.

Copyright Notice

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

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

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

Table of Contents

1. Introduction	2
2. Conventions, Definitions and Acronyms	2
3. Media Format Description	3
4. Payload Format	4
4.1. RTP Header Usage	4
4.2. VP9 Payload Description	6
4.2.1. Scalability Structure (SS):	8
4.2.2. Scalability Structure Update (SU):	9
4.3. VP9 Payload Header	10
4.4. Frame Fragmentation	10
4.5. Examples of VP9 RTP Stream	10
5. Using VP9 with RPSI and SLI Feedback	10
5.1. RPSI	10
5.2. SLI	11
5.3. Example	11
6. Layer Intra Request	13
7. Payload Format Parameters	14
7.1. Media Type Definition	14
7.2. SDP Parameters	15
7.2.1. Mapping of Media Subtype Parameters to SDP	16
7.2.2. Offer/Answer Considerations	16
8. Security Considerations	16
9. Congestion Control	17
10. IANA Considerations	17
11. References	17
Authors' Addresses	18

1. Introduction

This memo describes an RTP payload specification applicable to the transmission of video streams encoded using the VP9 video codec [I-D.grange-vp9-bitstream]. The format described in this document can be used both in peer-to-peer and video conferencing applications.

TODO: VP9 description. Please see [I-D.grange-vp9-bitstream].

2. Conventions, Definitions and Acronyms

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

3. Media Format Description

The VP9 codec can maintain up to eight reference frames, of which up to three can be referenced or updated by any new frame.

VP9 also allows a reference frame to be resampled and used as a reference for another frame of a different resolution. This allows internal resolution changes without requiring the use of keyframes.

These features together enable an encoder to implement various forms of coarse-grained scalability, including temporal, spatial, and quality scalability modes, as well as combinations of these, without the need for explicit spatially scalable encoding modes.

This payload format specification defines how such scalability modes can be encoded and communicated. In this payload, three separate types of layers are defined: temporal, spatial, and quality.

Temporal layers define different frame rates of video; spatial and quality layers define different, dependent representations of a single picture. Spatial layers allow a picture to be encoded at different resolutions, whereas quality layers allow a picture to be encoded at the same resolution but at different bitrates (and thus with different amounts of coding error).

Layers are designed (and MUST be encoded) such that if any layer, and all higher layers, are removed from the bitstream along any of the three dimensions, the remaining bitstream is still correctly decodable.

For terminology, this document uses the term "frame" to refer to a single encoded VP9 image, and "picture" to refer to all the representations of frames at a single instant in time. A picture thus can consist of multiple frames, encoding different spatial and/or quality layers.

[Editor's Note: Are separate spatial and quality layers necessary and useful? We could simplify by only defining a single sequence of frames within a picture.

Two modes of describing layer information are possible: "non-flexible mode" and "flexible mode". An encoder can freely switch between the two as appropriate.

In non-flexible mode, an SS message, which defines the layer hierarchy, is sent in the beginning of the stream together with the key frame. Each packet will have a picture id and reference indices, which in conjunction with the SS and the RTP sequence number can be

used to determine if the packet is decodable or not. An SU message can be sent by the sending client, or an MCU, to notify the receiver about what subset of the SS it will actually be receiving.

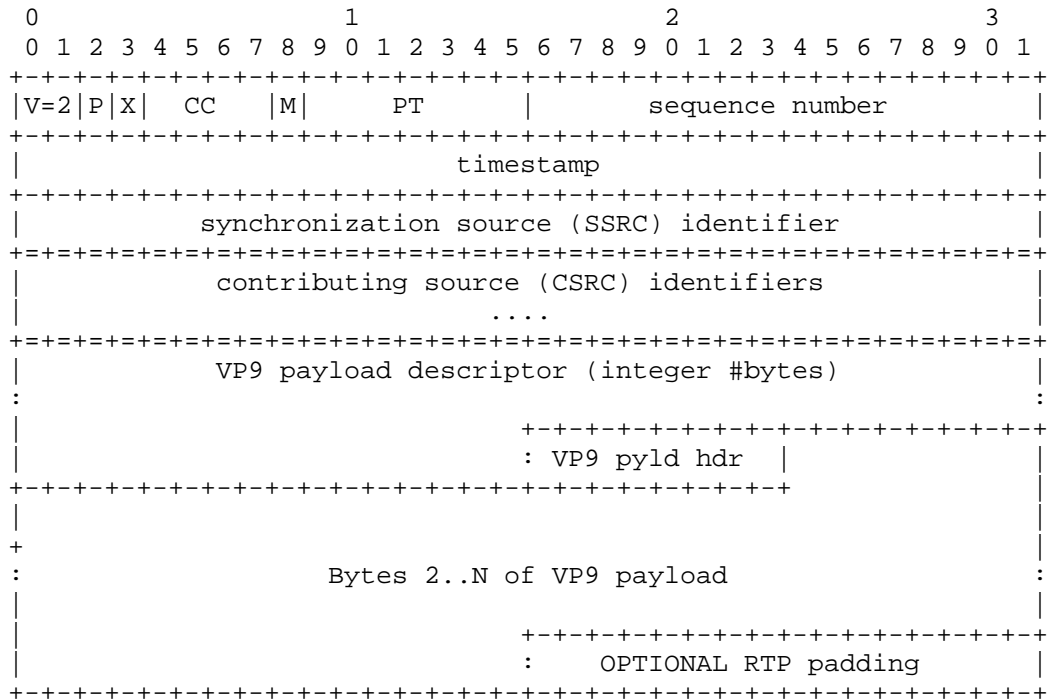
In the flexible mode each packet contains 1-4 reference indices, which identifies all frames referenced by the frame transmitted in the current packet. This enables a receiver to identify if a frame is decodable or not and helps it understand the layer structure so that it can drop packets as it sees fit. Since this is signaled in each packet it makes it possible to have more flexible layer hierarchies and patterns which are changing dynamically.

4. Payload Format

This section describes how the encoded VP9 bitstream is encapsulated in RTP. To handle network losses usage of RTP/AVPF [RFC4585] is RECOMMENDED. All integer fields in the specifications are encoded as unsigned integers in network octet order.

4.1. RTP Header Usage

The general RTP payload format for VP9 is depicted below.



The VP9 payload descriptor and VP9 payload header will be described in the next section. OPTIONAL RTP padding MUST NOT be included unless the P bit is set.

Figure 1

Marker bit (M): MUST be set for the final packet of each encoded frame. This enables a decoder to finish decoding the frame, where it otherwise may need to wait for the next packet to explicitly know that the frame is complete. Note that, if spatial or quality scalability is in use, more frames from the same picture may follow; see the description of the E bit below.

Timestamp: The RTP timestamp indicates the time when the frame was sampled, at a clock rate of 90 kHz. If a picture is encoded with multiple frames, all of the frames of the picture have the same timestamp.

Sequence number: The sequence numbers are monotonically increasing in order of the encoded bitstream.

The remaining RTP header fields are used as specified in [RFC3550].

4.2. VP9 Payload Description

The first octets after the RTP header are the VP9 payload descriptor, with the following structure.

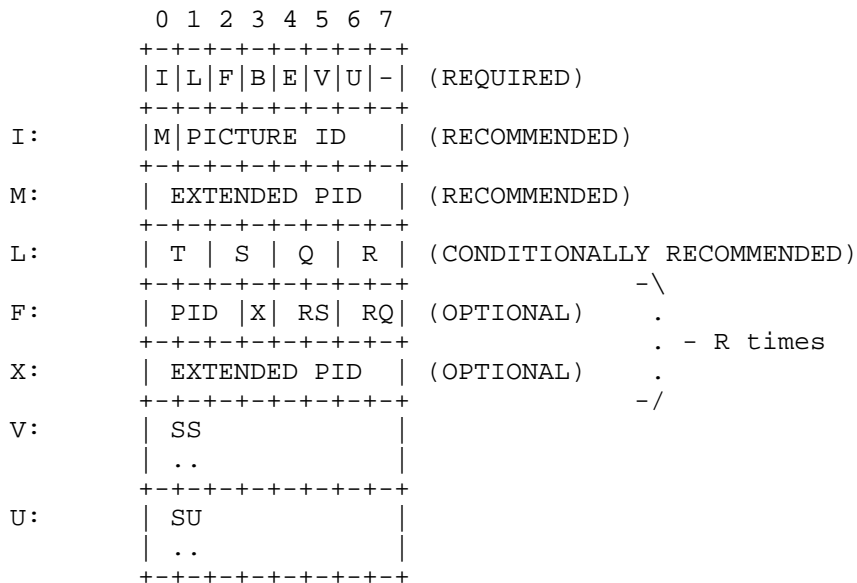


Figure 2

- I: PictureID present. When set to one, the OPTIONAL PictureID MUST be present after the mandatory first octet and specified as below. Otherwise, PictureID MUST NOT be present.
- L: Layer indices present. When set to one, the octets following the first octet and the extended Picture ID (if present) are as described by "Layer indices" below.
- F: Reference indices present. When set to one, the octets following the first octet and the extended Picture ID (if present) are as described by "Reference indices" below. This MUST only be set if L is also 1; if L is 0 then this MUST be set to zero and ignored by receivers.
- B: Start of VP9 frame. MUST be set to 1 if the first payload octet of the RTP packet is the beginning of a new VP9 frame, and MUST

NOT be 1 otherwise. Note that this frame might not be the first frame of the picture.

- E: End of picture. MUST be set to 1 for the final RTP packet of a VP9 picture, and 0 otherwise. Unless spatial or quality scalability is in use for this picture, this will have the same value as the marker bit in the RTP header.
- V: Scalability Structure (SS) present. When set to one, the OPTIONAL Scalability Structure MUST be present in the payload descriptor. Otherwise, the Scalability Structure MUST NOT be present.
- U: Scalability Structure Update (SU) present. When set to one, the OPTIONAL Scalability Structure Update MUST be present in the payload descriptor. Otherwise, the Scalability Structure Update MUST NOT be present.
- : Bit reserved for future use. MUST be set to zero and MUST be ignored by the receiver.

After the extension bit field follow the extension data fields that are enabled.

- M: The most significant bit of the first octet is an extension flag. The field MUST be present if the I bit is equal to one. If set the PictureID field MUST contain 16 bits else it MUST contain 8 bits including this MSB, see PictureID.

PictureID: 8 or 16 bits including the M bit. This is a running index of the frames. The field MUST be present if the I bit is equal to one. The 7 following bits carry (parts of) the PictureID. If the extension flag is one, the PictureID continues in the next octet forming a 15 bit index, where the 8 bits in the second octet are the least significant bits of the PictureID. If the extension flag is zero, there is no extension, and the PictureID is the 7 remaining bits of the first (and only) octet. The sender may choose 7 or 15 bits index. The PictureID SHOULD start on a random number, and MUST wrap after reaching the maximum ID. The receiver MUST NOT assume that the number of bits in PictureID stay the same through the session.

Layer indices: This byte is optional, but recommended whenever encoding with layers. T, S and Q are 2-bit indices for temporal, spatial, and quality layers, respectively. S and Q start at zero for each picture, and increment consecutively (with Q incrementing before S). These can help MCUs measure bitrates per layer and can help them make a quick decision on whether to relay a packet or not. They can also help receivers determine what layers they are

currently decoding. If "F" is set in the initial octet, R is 2 bits representing the number of reference fields this frame refers to. R MAY be zero, indicating a keyframe. The layer indices field will be followed by R reference indices. If "F" is not set, R MUST be set to zero and ignored by receivers.

Reference indices. These bytes are optional, but recommended when encoding with layers in the flexible mode. They are also recommended in the non-flexible mode when sending frames which are out of sync with the pattern signaled with the SS, for instance when encoding a layer synchronization frame in response to a LIR.

PID: The relative Picture ID referred to by this frame. I.e., PID=3 on a packet containing the frame with Picture ID 112 means that the frame refers back to the frame with picture ID 109. This calculation is done modulo the size of the Picture ID field, i.e. either 7 or 15 bits. For most layer structures a 3-bit relative Picture ID will be enough; however, the X bit can be used to refer to pictures with Picture IDs more than 7 previously.

RS and RQ: The spatial and quality layer IDs of the frame referred to by this frame, in the picture identified by the relative Picture ID.

X: 1 if this layer index has an extended relative Picture ID.

These 1-2 bytes are repeated R times, defined by the two R bits in the layer indices field.

4.2.1. Scalability Structure (SS):

The Scalability Structure data describes the pattern of scalable frames that will be used in a scalable stream. If the VP9 payload header's "V" bit is set, the scalability structure (SS) is present in the position indicated in Figure 2.

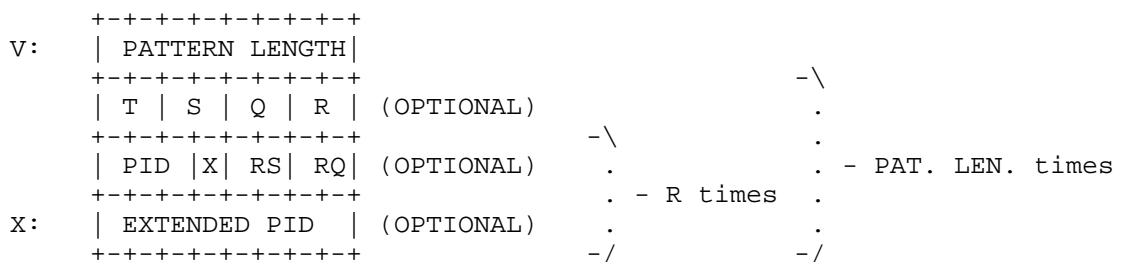


Figure 3

The scalability structure allows the structure of the VP9 stream to be predeclared, rather than indicating it on the fly with every frame as with the layer indices.

Its structure consists of a sequence of frames, encoded as with the layer indices. It begins with PATTERN LENGTH, indicating the number of frames in the pattern; it is then followed by that many instances of data encoded using the same semantics as the layer indices.

TODO: add frame resolution information.

In a scalable stream sent with a fixed pattern, the scalability structure SHOULD be included in the first packet of every keyframe picture, and also in the first packet of the first picture in which the scalability structure changes. If a SS is included in a picture with TID not equal to 0, it MUST also be repeated in the first packet the first frame with a lower TID, until TID equals 0.

If PATTERN LENGTH is 0, it indicates that no fixed scalability information is present going forward in the bitstream. An SS with a PATTERN LENGTH of 0 allows a bitstream to be changed from non-flexible to flexible mode.

4.2.2. Scalability Structure Update (SU):

TODO

4.3. VP9 Payload Header

TODO: need to describe VP9 payload header.

4.4. Frame Fragmentation

VP9 frames are fragmented into packets, in RTP sequence number order, beginning with a packet with the B bit set, and ending with a packet with the RTP marker bit set. There is no mechanism for finer-grained access to parts of a VP9 frame.

4.5. Examples of VP9 RTP Stream

TODO

5. Using VP9 with RPSI and SLI Feedback

The VP9 payload descriptor defined in Section 4.2 above contains an optional PictureID parameter. One use of this parameter is included to enable use of reference picture selection index (RPSI) and slice loss indication (SLI), both defined in [RFC4585].

5.1. RPSI

TODO: Update to indicate which frame within the picture.

The reference picture selection index is a payload-specific feedback message defined within the RTCP-based feedback format. The RPSI message is generated by a receiver and can be used in two ways. Either it can signal a preferred reference picture when a loss has been detected by the decoder -- preferably then a reference that the decoder knows is perfect -- or, it can be used as positive feedback information to acknowledge correct decoding of certain reference pictures. The positive feedback method is useful for VP9 used as unicast. The use of RPSI for VP9 is preferably combined with a special update pattern of the codec's two special reference frames -- the golden frame and the altref frame -- in which they are updated in an alternating leapfrog fashion. When a receiver has received and correctly decoded a golden or altref frame, and that frame had a PictureID in the payload descriptor, the receiver can acknowledge this simply by sending an RPSI message back to the sender. The message body (i.e., the "native RPSI bit string" in [RFC4585]) is simply the PictureID of the received frame.

5.2. SLI

TODO: Update to indicate which frame within the picture.

The slice loss indication is another payload-specific feedback message defined within the RTCP-based feedback format. The SLI message is generated by the receiver when a loss or corruption is detected in a frame. The format of the SLI message is as follows [RFC4585]:

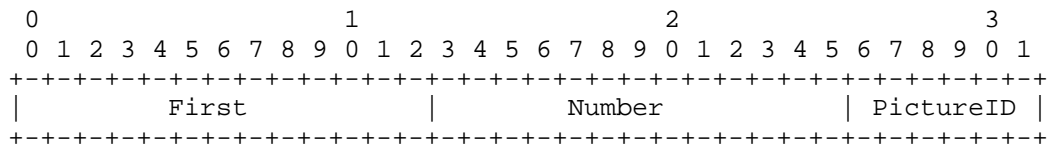


Figure 4

Here, First is the macroblock address (in scan order) of the first lost block and Number is the number of lost blocks. PictureID is the six least significant bits of the codec-specific picture identifier in which the loss or corruption has occurred. For VP9, this codec-specific identifier is naturally the PictureID of the current frame, as read from the payload descriptor. If the payload descriptor of the current frame does not have a PictureID, the receiver MAY send the last received PictureID+1 in the SLI message. The receiver MAY set the First parameter to 0, and the Number parameter to the total number of macroblocks per frame, even though only parts of the frame is corrupted. When the sender receives an SLI message, it can make use of the knowledge from the latest received RPSI message. Knowing that the last golden or altref frame was successfully received, it can encode the next frame with reference to that established reference.

5.3. Example

TODO: this example is copied from the VP8 payload format specification, and has not been updated for VP9. It may be incorrect.

The use of RPSI and SLI is best illustrated in an example. In this example, the encoder may not update the altref frame until the last sent golden frame has been acknowledged with an RPSI message. If an update is not received within some time, a new golden frame update is sent instead. Once the new golden frame is established and acknowledged, the same rule applies when updating the altref frame.

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

Event	Sender	Receiver	Established reference
1000	Send golden frame PictureID = 0	Receive and decode golden frame	golden
1001		Send RPSI(0)	
1002	Receive RPSI(0)		
...	(sending regular frames)		
1100	Send altref frame PictureID = 100	Altref corrupted or lost	golden
1101		Send SLI(100)	golden
1102	Receive SLI(100)		
1103	Send frame with reference to golden	Receive and decode frame (decoder state restored)	golden
...	(sending regular frames)		
1200	Send altref frame PictureID = 200	Receive and decode altref frame	golden
1201		Send RPSI(200)	
1202	Receive RPSI(200)		altref
...	(sending regular		

	frames)		
1300	Send golden frame PictureID = 300	Receive and decode golden frame	altref
1301		Send RPSI(300)	altref
1302	RPSI lost		
1400	Send golden frame PictureID = 400	Receive and decode golden frame	altref
1401		Send RPSI(400)	
1402	Receive RPSI(400)		golden

Table 1: Example signaling between sender and receiver

Note that the scheme is robust to loss of the feedback messages. If the RPSI is lost, the sender will try to update the golden (or altref) again after a while, without releasing the established reference. Also, if an SLI is lost, the receiver can keep sending SLI messages at any interval allowed by the RTCP sending timing restrictions as specified in [RFC4585], as long as the picture is corrupted.

6. Layer Intra Request

Editor's Note: The message described in this section is applicable to other codecs beyond just VP9. In the future it will be likely be split out into another document.

TODO: details of how this is encoded in RTCP.

A synchronization frame can be requested by sending a LIR, which is an RTCP feedback message asking the encoder to encode a frame which makes it possible to upgrade to a higher layer. The LIR message contains two tuples, {T1,S1,Q1} and {T2,S2,Q2}, where the first tuple is the currently highest layer the decoder can decode, while the second tuple is the layer the decoder wants to upgrade to.

Identification of an upgrade frame can be derived from the reference IDs of each frame by backtracking the dependency chain until reaching a point where only decodable frames are being referenced. Therefore it's recommended both for both the flexible and the non-flexible mode that, when upgrade frames are being encoded in response to a LIR, those packets should contain layer indices and the reference fields so that the decoder or an MCU can make this derivation.

Example:

LIR {1,1,0}, {1,2,1} is sent by an MCU when it is currently relaying {1,1,0} to a receiver and which wants to upgrade to {1,2,1}. In response the encoder should encode the next frames in layers {1,1,1} and {1,2,1} by only referring to frames in {1,1,0}, {1,0,0} or {0,0,0}.

In the non-flexible mode, periodic upgrade frames can be defined by the layer structure of the SS, thus periodic upgrade frames can be automatically identified by the picture ID.

7. Payload Format Parameters

This payload format has two required parameters.

7.1. Media Type Definition

This registration is done using the template defined in [RFC6838] and following [RFC4855].

Type name: video

Subtype name: VP9

Required parameters:

These parameters MUST be used to signal the capabilities of a receiver implementation. These parameters MUST NOT be used for any other purpose.

max-fr: The value of max-fr is an integer indicating the maximum frame rate in units of frames per second that the decoder is capable of decoding.

max-fs: The value of max-fs is an integer indicating the maximum frame size in units of macroblocks that the decoder is capable of decoding.

The decoder is capable of decoding this frame size as long as the width and height of the frame in macroblocks are less than

$\text{int}(\sqrt{\text{max-fs} * 8})$ - for instance, a max-fs of 1200 (capable of supporting 640x480 resolution) will support widths and heights up to 1552 pixels (97 macroblocks).

Optional parameters: none

Encoding considerations:

This media type is framed in RTP and contains binary data; see Section 4.8 of [RFC6838].

Security considerations: See Section 8 of RFC xxxx.

[RFC Editor: Upon publication as an RFC, please replace "XXXX" with the number assigned to this document and remove this note.]

Interoperability considerations: None.

Published specification: VP9 bitstream format

[I-D.grange-vp9-bitstream] and RFC XXXX.

[RFC Editor: Upon publication as an RFC, please replace "XXXX" with the number assigned to this document and remove this note.]

Applications which use this media type:

For example: Video over IP, video conferencing.

Additional information: None.

Person & email address to contact for further information:

TODO [Pick a contact]

Intended usage: COMMON

Restrictions on usage:

This media type depends on RTP framing, and hence is only defined for transfer via RTP [RFC3550].

Author: TODO [Pick a contact]

Change controller:

IETF Payload Working Group delegated from the IESG.

7.2. SDP Parameters

The receiver MUST ignore any fmp parameter unspecified in this memo.

7.2.1. Mapping of Media Subtype Parameters to SDP

The media type video/VP9 string is mapped to fields in the Session Description Protocol (SDP) [RFC4566] as follows:

- o The media name in the "m=" line of SDP MUST be video.
- o The encoding name in the "a=rtpmap" line of SDP MUST be VP9 (the media subtype).
- o The clock rate in the "a=rtpmap" line MUST be 90000.
- o The parameters "max-fs", and "max-fr", MUST be included in the "a=fmtp" line of SDP. These parameters are expressed as a media subtype string, in the form of a semicolon separated list of parameter=value pairs.

7.2.1.1. Example

An example of media representation in SDP is as follows:

```
m=video 49170 RTP/AVPF 98
a=rtpmap:98 VP9/90000
a=fmtp:98 max-fr=30; max-fs=3600;
```

7.2.2. Offer/Answer Considerations

TODO: Update this for VP9

8. Security Considerations

RTP packets using the payload format defined in this specification are subject to the security considerations discussed in the RTP specification [RFC3550], and in any applicable RTP profile. The main security considerations for the RTP packet carrying the RTP payload format defined within this memo are confidentiality, integrity and source authenticity. Confidentiality is achieved by encryption of the RTP payload. Integrity of the RTP packets through suitable cryptographic integrity protection mechanism. Cryptographic system may also allow the authentication of the source of the payload. A suitable security mechanism for this RTP payload format should provide confidentiality, integrity protection and at least source authentication capable of determining if an RTP packet is from a member of the RTP session or not. Note that the appropriate mechanism to provide security to RTP and payloads following this memo may vary. It is dependent on the application, the transport, and the signaling protocol employed. Therefore a single mechanism is not sufficient, although if suitable the usage of SRTP [RFC3711] is

recommended. This RTP payload format and its media decoder do not exhibit any significant non-uniformity in the receiver-side computational complexity for packet processing, and thus are unlikely to pose a denial-of-service threat due to the receipt of pathological data. Nor does the RTP payload format contain any active content.

9. Congestion Control

Congestion control for RTP SHALL be used in accordance with RFC 3550 [RFC3550], and with any applicable RTP profile; e.g., RFC 3551 [RFC3551]. The congestion control mechanism can, in a real-time encoding scenario, adapt the transmission rate by instructing the encoder to encode at a certain target rate. Media aware network elements MAY use the information in the VP9 payload descriptor in Section 4.2 to identify non-reference frames and discard them in order to reduce network congestion. Note that discarding of non-reference frames cannot be done if the stream is encrypted (because the non-reference marker is encrypted).

10. IANA Considerations

The IANA is requested to register the following values:

- Media type registration as described in Section 7.1.

11. References

- [I-D.grange-vp9-bitstream]
Grange, A. and H. Alvestrand, "A VP9 Bitstream Overview", draft-grange-vp9-bitstream-00 (work in progress), February 2013.
- [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.
- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.
- [RFC4855] Casner, S., "Media Type Registration of RTP Payload Formats", RFC 4855, February 2007.
- [RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, January 2013.

Authors' Addresses

Justin Uberti
Google, Inc.
747 6th Street South
Kirkland, WA 98033
USA

Email: justin@uberti.name

Stefan Holmer
Google, Inc.
Kungsbron 2
Stockholm 111 22
Sweden

Magnus Flodman
Google, Inc.
Kungsbron 2
Stockholm 111 22
Sweden

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

Email: jonathan@vidyo.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 22, 2015

M. Westerlund
B. Burman
Ericsson
R. Even
Huawei Technologies
M. Zanaty
Cisco Systems
July 21, 2014

RTP Header Extension for RTCP Source Description Items
draft-westerlund-avtext-sdes-hdr-ext-02

Abstract

Source Description (SDS) items are normally transported in RTP control protocol (RTCP). In some cases it can be beneficial to speed up the delivery of these items. Mainly when a new source (SSRC) joins an RTP session and the receivers needs this source's relation to other sources and its synchronization context, which are fully or partially identified using SDS items. To enable this optimization, this document specifies a new RTP header extension that can carry any type of SDS items.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 22, 2015.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

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

Table of Contents

1. Introduction	2
2. Definitions	3
2.1. Requirements Language	3
2.2. Terminology	3
3. Motivation	3
4. Specification	5
4.1. SDES Item Header Extension	5
4.1.1. One-Byte Format	5
4.1.2. Two-Byte Format	5
4.2. Usage of the SDES Item Header Extension	6
4.2.1. One or Two Byte Headers	6
4.2.2. MTU and Packet Expansion	6
4.2.3. Transmission Considerations	7
4.2.4. Different Usages	8
4.2.5. SDES Items in RTCP	8
5. IANA Considerations	9
6. Security Considerations	10
7. Acknowledgements	10
8. References	10
8.1. Normative References	10
8.2. Informative References	11
Authors' Addresses	12

1. Introduction

This specification defines an RTP header extension [RFC3550][RFC5285] that can carry RTCP source description (SDES) items. By including selected SDES items in an header extension the determination of relationship and synchronization context for new RTP streams (SSRCs) in an RTP session can be speeded up. Which relationship and what information depends on the SDES items carried. This becomes a complement to using only RTCP for SDES Item delivery.

First, some requirements language is defined. The following section motivates why this header extension is sometimes required or at least provides a significant improvement compared to waiting for regular RTCP packet transmissions of the information. This is followed by a specification of the header extension. Next, a sub-space of the

header-extension URN is defined to be used for existing and future SDES items, and the existing SDES items are registered.

2. Definitions

2.1. Requirements Language

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

2.2. Terminology

This document uses terminology defined in "A Taxonomy of Grouping Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources" [I-D.ietf-avtext-rtp-grouping-taxonomy] . In particular the following definitions:

Media Source

RTP Stream

Media Encoder

Encoded Stream

Participant

3. Motivation

Source Description (SDES) items are being associated with a particular SSRC and thus RTP stream. The source description items provide various meta data associated with the SSRC. How important it is to have this data no later than when receiving the first RTP packets depends on the item itself. The CNAME item is one item that is commonly needed if not at reception of the first RTP packet for this SSRC, so at least by the time the first media can be played out. If not, the synchronization context cannot be determined and thus any related streams cannot be correctly synchronized. Thus, this is a great example for the need to have this information early when a new RTP stream is received.

The main reason for new SSRCs in an RTP session is that a media sources are added. This either because an end-point is adding a new actual media source, or additional participants in a multi-party session being added to the session. Another reason for a new SSRC can be an SSRC collision that forces the colliding parties to select a new SSRC.

Returning to the case of rapid media synchronization, there exist an RTP header extension for Rapid Synchronization of RTP Flows [RFC6051]. That header extension carries the clock information present in the RTCP sender report (SR) packets. It however assumes that the CNAME binding is known, which can be provided via signaling in some cases, but not all. Thus an RTP header extension for carrying SDES items like CNAME is a powerful combination to enable rapid synchronization in all cases.

The Rapid Synchronization of RTP Flows specification does provide an analysis of the initial synchronization delay for different sessions depending on number of receivers as well as on session bandwidth (Section 2.1 of [RFC6051]). These results are applicable also for other SDES items that have a similar time dependency until the information can be sent using RTCP. Thus the benefit for reduction of initial delay before information is available can be determined for some use cases from these figures.

That document also discusses the case of late joiners, and defines an RTCP Feedback format to request synchronization information, which is another potential use case for SDES items in RTP header extension. It would for example be natural to include CNAME SDES item with the header extension containing the NTP formatted reference clock to ensure synchronization.

Some new SDES items are currently proposed, which can all benefit from timely delivery:

SRCNAME: This is a media source and encoding identifier to enable support for simulcast and improve some scalable encoding usages [I-D.westerlund-avtext-rtcp-sdes-srcname]. This SDES item could be used both for new sources and late joiners.

APPID: This SDES item provides an application specific identifier dynamically assigned to a particular RTP stream. The intention is to provide a receiver with information about the current role of the received RTP stream or its usage in an application [I-D.even-mmusic-application-token]. Thus a particular ID can be reassigned many times during the lifetime of an RTP session. This puts additional timing requirements, not only for new sources and late joiners, but also whenever the Application token is reassigned to another stream.

Based on the above, there appear to be good reasons why an RTP header extension for SDES items is worthwhile to pursue.

4. Specification

This section first specifies the SDES item RTP header extension format, followed by some usage considerations.

4.1. SDES Item Header Extension

The RTP header extension scheme that allows for multiple extensions to be included is defined in "A General Mechanism for RTP Header Extensions" [RFC5285]. That specification defines both short and long item headers. The short headers (One-byte) are restricted to 1 to 16 bytes of data, while the long format (Two-byte) supports a data length of 0 to 255 bytes. Thus that RTP header extension format is capable of supporting any SDES item from a data length perspective.

The ID field, independent of short or long format, identifies both the type of RTP header extension and, in the case of the SDES item header extension, the type of SDES item. The mapping is done in signaling by identifying the header extension and SDES item type using a URN, which is defined in the IANA consideration (Section 5) for all existing SDES items.

4.1.1. One-Byte Format

The one-byte header format for an SDES item extension element consists of the One-Byte header (defined in Section 4.2 of [RFC5285]), which consists of a 4-bit ID followed by a 4-bit length field (len) that identifies how many bytes (len value +1) of data that follows the header. The data part consists of len+1 bytes of UTF-8 text. The type of text is determined by the ID field value and its mapping to the type of SDES item.

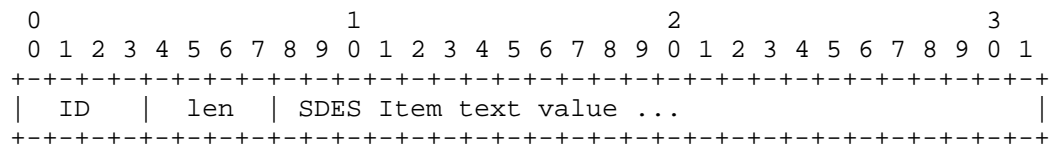


Figure 1

4.1.2. Two-Byte Format

The two-byte header format for an SDES item extension element consists of the two-byte header (defined in Section 4.3 of [RFC5285]), which consists of an 8-bit ID followed by an 8-bit length field (len) that identifies how many bytes of data that follows the header. The data part consists of len bytes of UTF-8 text. The type

of text is determined by the ID field value and its mapping to the type of SDES item.

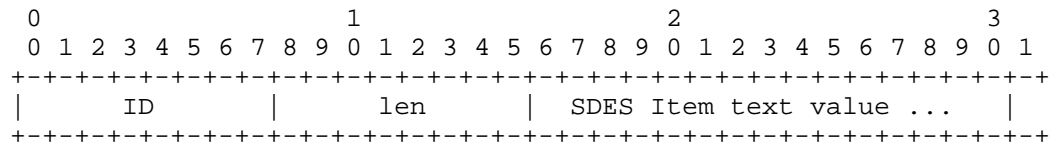


Figure 2

4.2. Usage of the SDES Item Header Extension

This section discusses various usage considerations; which form of header extension to use, the packet expansion, and when to send SDES items in header extension.

4.2.1. One or Two Byte Headers

The RTP header extensions for SDES items MAY use either the one-byte or two-byte header formats, depending on the text value size for the used SDES items. The one-byte header SHOULD be used when all non SDES item header extensions supports the one-byte format and all SDES item text values contain at most 16 bytes. Note that the RTP header extension specification does not allow mixing one-byte and two-byte headers for the same RTP stream (SSRC), so if the value size of any of the SDES items value requires the two-byte header, the all other header extensions MUST also use the two-byte header format.

For example using CNAMEs that are generated according to "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)" [RFC7022], using short term persistent values, and if 96-bit random values prior to base64 encoding are sufficient, then they will fit into the One-Byte header format.

4.2.2. MTU and Packet Expansion

The RTP packet size will clearly increase when they include the header extension. How much depends on which header extensions and their data parts. The SDES items can vary in size. There are also some use-cases which require transmitting multiple SDES items in the same packet to ensure that all relevant data reaches the receiver. An example of that is when you need both the CNAME, a SRCNAME and an appId plus the rapid time synchronization extension from RFC 6051. Such a combination is quite likely to result in at least 16+3+1+8 bytes of data plus the headers, which will be another 8 bytes for one-byte headers, thus in total 36 bytes.

The packet expansion can cause an issue when it cannot be taken into account when producing the RTP payload. Thus an RTP payload that is created to meet a particular IP level Maximum Transmission Unit (MTU), taking the addition of IP/UDP/RTP headers into account but excluding RTP header extensions suddenly exceeds the MTU, resulting in IP fragmentation. IP fragmentation is known to negatively impact the loss rate due to middleboxes unwilling or not capable of dealing with IP fragments.

As this is a real issue, the media encoder and payload packetizer should be flexible and be capable of handling dynamically varying payload size restrictions to counter the packet expansion caused by header extensions. If that is not possible, some reasonable worst case packet expansion should be calculated and used to reduce the RTP payload size of all RTP packets the sender transmits.

4.2.3. Transmission Considerations

The general recommendation is to only send header extensions when needed. This is especially true for SDES items that can be sent in periodic repetitions of RTCP throughout the whole session. Thus, the different usages (Section 4.2.4) have different recommendations. First some general considerations for getting the header extensions delivered to the receiver:

1. The probability for packet loss and burst loss determine how many repetitions of the header extensions will be required to reach a targeted delivery probability, and if burst loss is likely what dispersion would be needed to avoid getting multiple header extensions lost in a single burst.
2. How early the SDES item information is needed, from the first received RTP data or only after some set of packets are received, can guide if the header extension(s) should be in all of the first N packets or be included only once per set of packets, for example once per video frame.
3. The use of RTP level robustness mechanisms, such as RTP retransmission [RFC4588], or Forward Error Correction, e.g., [RFC5109] may treat packets differently from a robustness perspective, and SDES header extensions should be added to packets that get a treatment corresponding to the relative importance of receiving the information.

In summary, the number of header extension transmissions should be tailored to a desired probability of delivery taking the receiver population size into account. For the very basic case, N repetitions of the header extensions should be sufficient, but may not be

optimal. N is selected so that probability of delivery of at least one out of the N reaches the target value when calculating $1-P^N$, where P is the probability of packet loss. For point to point or small receiver populations, it might also be possible to use feedback, such as RTCP, to determine when the information in the header extensions has likely reached all receivers.

4.2.4. Different Usages

4.2.4.1. New SSRC

A new SSRC joins an RTP session. As this SSRC is completely new for everyone, the goal is to ensure that all receivers with high probability receives the information in the header extension. Thus header extension transmission strategies that allow some margins in the delivery probability should be considered.

4.2.4.2. Late Joiner

In a multi-party RTP session where one or a small number of receivers join a session where the majority of receivers already have all necessary information, the use of header extensions to deliver relevant information should be tailored to reach the new receivers. The trigger to send header extensions can for example either be RTCP from new receiver(s) or an explicit request like the Rapid Resynchronization Request defined in [RFC6051].

4.2.4.3. Information Change

In cases when the SDES item text value is changed and the new SDES information is tightly coupled to and thus needs to be synchronized with a related change in the RTP stream, use of a header extension is far superior to RTCP SDES. In this case it is equal or even more important with timely SDES information than in the case of new SSRCS (Section 4.2.4.1). Continued use of the old SDES information can lead to really undesired effects in the application. Application Token [I-D.even-mmusic-application-token] would be one such case. Thus, header extension transmission strategies with high probability of delivery should be chosen.

4.2.5. SDES Items in RTCP

As this RTP header extensions information, i.e. SDES Items can and will be sent also in RTCP it is worth some reflections on this interaction. There also exist the possibility to schedule a non-regular RTCP packet transmission containing important SDES items if one uses a RTP/AVPF based RTP profile. Depending on which mode ones RTCP feedback transmitter is working on extra RTCP packets may be

sent as immediate or early packets, enabling more timely deliver of SDES information.

There is however two aspects that differ between using RTP header extension and any non-regular transmission of RTCP packets. First, as the RTCP packet is a separate packet, there is no direct relation and also no fate sharing between the relevant media data and the SDES information. The order of arrival for the packets will matter. With a header-extension the SDES items can be ensured to arrive if the media data to played out arrives. Secondly, it is difficult to determine if an RTCP packet is actually delivered. This, as the RTCP packets lack both sequence number or a mechanism providing feedback on the RTCP packets themselves.

5. IANA Considerations

This IANA section firstly proposes to:

- o Reserve the SDES item RTP header extension defined in this document for use with current and future SDES items.
- o Register and assign the URN sub-space "urn:ietf:params:rtp-hdrex: sdes:" in the RTP Compact Header Extensions registry.

The reason to require registering a URN within that sub-space is that the name represent an RTCP Source Description item, where a specification is strongly recommended. The formal policy is maintained from the main space, i.e. Expert Review.

Secondly, it is proposed that all the current existing SDES items are registered for usage in the RTP Compact Header Extensions registry :

URN	SDES Item	Reference
=====		
urn:ietf:params:rtp-hdrex: sdes:cname	CNAME	[RFC3550]
urn:ietf:params:rtp-hdrex: sdes:name	NAME	[RFC3550]
urn:ietf:params:rtp-hdrex: sdes:email	EMAIL	[RFC3550]
urn:ietf:params:rtp-hdrex: sdes:phone	PHONE	[RFC3550]
urn:ietf:params:rtp-hdrex: sdes:loc	LOC	[RFC3550]
urn:ietf:params:rtp-hdrex: sdes:tool	TOOL	[RFC3550]
urn:ietf:params:rtp-hdrex: sdes:note	NOTE	[RFC3550]
urn:ietf:params:rtp-hdrex: sdes:priv	PRIV	[RFC3550]
urn:ietf:params:rtp-hdrex: sdes:h323-caddr	H323-CADDR	[Vineet_Kumar]
urn:ietf:params:rtp-hdrex: sdes:apsi	APSI	[RFC6776]

6. Security Considerations

Source Description items may contain data that are sensitive from a security perspective. There exist SDES items that are or may be sensitive from a user privacy perspective, like CNAME, NAME, EMAIL, PHONE, LOC and H323-CADDR. Others may contain sensitive information like NOTE and PRIV, while others may be sensitive from profiling implementations for vulnerability or other reasons, like TOOL. The CNAME sensitivity can vary depending on how it is generated and what persistence it has. A short term CNAME identifier generated using a random number generator may have minimal security implications, while one of the form user@host has privacy concerns and one generated from a MAC address has long term tracking potentials.

The above security concerns may have to be put in relation to needs of third party monitoring. In RTP sessions where any type of confidentiality protection is enabled, the SDES item header extensions SHOULD also be protected per default. This implies that to provide confidentiality, users of SRTP need to implement encrypted header extensions per [RFC6904]. Commonly, it is expected that the same security level is applied both RTCP packets carrying SDES items, as a RTP header extension containing a SDES item. If the security level is different it is important to consider the security properties as the worst in each aspect for the different configurations.

As the SDES items are used by the RTP based application to establish relationships between RTP streams or between an RTP stream and information about the originating Participant, there SHOULD be strong requirements on integrity and source authentication of the header extensions. If not, an attacker can modify the SDES item value to create erroneous relationship bindings in the receiving application.

7. Acknowledgements

The authors likes to thanks the following individuals for feedback and suggestions; Colin Perkins.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

- [RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, July 2008.
- [RFC6904] Lennox, J., "Encryption of Header Extensions in the Secure Real-time Transport Protocol (SRTP)", RFC 6904, April 2013.

8.2. Informative References

- [I-D.even-mmusic-application-token]
Even, R., Lennox, J., and Q. Wu, "The Session Description Protocol (SDP) Application Token Attribute", draft-even-mmusic-application-token-03 (work in progress), April 2014.
- [I-D.ietf-avtext-rtp-grouping-taxonomy]
Lennox, J., Gross, K., Nandakumar, S., and G. Salgueiro, "A Taxonomy of Grouping Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", draft-ietf-avtext-rtp-grouping-taxonomy-01 (work in progress), February 2014.
- [I-D.westerlund-avtext-rtcp-sdes-srcname]
Westerlund, M., "RTCP Source Description Item SRCNAME to Label Individual Media Sources", draft-westerlund-avtext-rtcp-sdes-srcname-03 (work in progress), October 2013.
- [RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.
- [RFC5109] Li, A., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, December 2007.
- [RFC6051] Perkins, C. and T. Schierl, "Rapid Synchronisation of RTP Flows", RFC 6051, November 2010.
- [RFC6776] Clark, A. and Q. Wu, "Measurement Identity and Information Reporting Using a Source Description (SDES) Item and an RTCP Extended Report (XR) Block", RFC 6776, October 2012.
- [RFC7022] Begen, A., Perkins, C., Wing, D., and E. Rescorla, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)", RFC 7022, September 2013.

Authors' Addresses

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Stockholm
Sweden

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

Bo Burman
Ericsson
Kistavagen 25
SE-164 80 Stockholm
Sweden

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

Roni Even
Huawei Technologies
Tel Aviv
Israel

Email: roni.even@mail01.huawei.com

Mo Zanaty
Cisco Systems
7100 Kit Creek
RTP, NC 27709
USA

Email: mzanaty@cisco.com