Network Working Group	M. Westerlund
Internet-Draft	B. Burman
Intended status: Standards Track	F. Jansson
Expires: April 26, 2012	Ericsson
	October 24, 2011

Multiple Synchronization sources (SSRC) in RTP Session Signaling
draft-westerlund-avtcore-max-ssrc-00

Abstract

RTP has always been a protocol that supports multiple participants each sending their own media streams in an RTP session. Unfortunately many implementations are designed only for point to point voice over IP with a single source in each end-point. Even client implementations aimed at video conferences have often been built with the assumption around central mixers that only deliver a single media stream per media type. Thus any application that wants to allow for more advance usage where multiple media streams are sent and received by an end-point has an issue with legacy implementations. This document describes the problem and proposes a solution for how to use multiple SSRCs within one RTP session and at the same time handle the legacy issues.

Table of Contents

1. Introduction

This document discusses the issues of non basic usage of RTP *[RFC3550]* where there is multiple media sources sent over an RTP session using the SSRC source identifier to distinguish between the sources. This include multiple sources from the same end-point, multiple end-points each having a source, or an application that sends or receive multiple encodings of a particular source.

1.1. Background

RTP sessions are a concept which most fundamental part is an SSRC space. This space can encompass a number of network nodes and interconnected transport flows between these nodes. Each node may have zero, one or more source identifiers (SSRCs) used to either identify a real media source such as a camera or a microphone, a conceptual source (like the most active speaker selected by an RTP mixer that switches between incoming media streams based on the media stream or additional information), or simply as an identifier for a receiver that provides feedback and reports on reception. There are also RTP nodes, like translators that are manipulating data, transport or session state without making their presence aware to the other session participants. RTP was designed with multiple participants in a session from the beginning. This was not restricted to multicast as many believe but also unicast using either multiple transport flows below RTP or a network node that redistributes the RTP packets, either unchanged in the form of a transport translator (relay) or modified in an RTP mixer. There is also the case where a single end-point have multiple media sources of the same media type, like multiple cameras or microphones. However, the most common use cases have been point to point Voice over IP (VoIP) or streaming applications where there have commonly not been more than one media source per end-point. Even in conferencing applications, especially voice only, the conference focus or bridge have provided a single stream being a mix of the other participants to each participant. Thus there has been little need for handling multiple SSRCs in implementations. This has resulted in an installed legacy base that is not fully RTP specification compliant and will have different issues if they receive multiple SSRCs of media, either simultaneously or in sequence. These issues will manifest themselves in various ways, either by software crashes, or simply in limited functionality, like only decoding and playing back the first or latest SSRC received and discarding any other SSRCs.

The signaling solutions around RTP, especially the SDP *[RFC4566]* based, have not considered the fundamental issues around an RTP session's theoretical support of up to 4 billion plus sources all sending media. No end-point has infinite processing resources to decode and mix any number of media sources. In addition the memory for storing related state, especially decoder state is limited, and the network bandwidth to receive multiple streams is also limited. Today, the most likely limitations are processing and network bandwidth although for some use cases memory or other limitations may also exist. The issue is that a given end-point will have some limitations in the number of streams it simultaneously can receive, decode and playback. These limitations need to be possible to expose and enabling the session participants to take them into account.

In similar ways there is a need for an end-point to express if it intends to produce one or more media streams in an RTP session. Todays SDP signaling support for this is basically the directionality

attribute which indicates an end-point intent to send media or not. There is however no way to indicate how many media streams will be sent.
Taking these things together there exist a clear need to enable the usage of multiple simultaneous media streams within an RTP session in a way that allows a system to take legacy implementations into account in addition to negotiate the actual capabilities around the multiple streams in an RTP session.

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 [RFC2119].

2.2. Terminology

The following terms and abbreviations are used in this document:

Encoding: A particular encoding is the choice of the media encoder (codec) that has been used to compress the media, the fidelity of that encoding through the choice of sampling, bit-rate and other configuration parameters.

Different encodings: An encoding is different when some parameter that characterize the encoding of a particular media source has been changed. Such changes can be one or more of the following parameters; codec, codec configuration, bit-rate, sampling.

3. Multiple Streams Issues

This section attempts to go a bit more in depth around the different issues when using multiple media streams in an RTP session to make it clear that although in theory multi-stream applications should already be possible to use, there are good reasons to create extensions for signaling. In addition, the RTP specification could benefit from clarifications on how certain mechanisms should be working when an RTP session contains more than two SSRCs.

3.1. Legacy Behaviors

It is a common assumption among many applications using RTP that they do not have a need to support more than one incoming and one outgoing media stream per RTP session. For a number of applications this assumption has been correct. For VoIP and Streaming applications it has been easiest to ensure that a given end-point only receives and/or sends a single stream. However, all end-points should support a source changing SSRC value during a session, e.g due to SSRC value collision

between participants in a conference and the requirement to always use unique SSRC values.
Some RTP extension mechanisms require the RTP stacks to handle additional SSRCs, like SSRC multiplexed RTP retransmission described in [RFC4588]. However, that still has only required handling a single media decoding chain.
There are however applications that clearly can benefit from receiving and using multiple media streams simultaneously. A very basic case would be T.140 conversational text, where the text characters are transmitted as a real-time media stream as you type. When used in a multi-party chat scenario, an end-point can receive input from multiple sending end-points where the T.140 RTP Payload Format *[RFC4103]* text media is both low bandwidth and where there is no obvious method to algorithmically distinguish between multiple sources of text, making simple multiplex and identification of separate sources through an identifier (SSRC) a good choice.
An RTP session that contains an end-point with more than two SSRCs actively sending media streams put some requirements on the receiving client which is not necessarily fulfilled by a legacy client:

1. The receiving client needs to handle receiving more than one stream simultaneously rather than replacing the already existing stream with the new one.

2. Be capable of decoding multiple streams simultaneously.

3. Be capable of rendering multiple streams simultaneously.

An application using multiple streams may be very similar to existing one media stream applications at signaling level. To avoid connecting two different implementations, one that is built to support multiple streams and one that is not, it is important that the capabilities are signaled. It is also the legacy that makes us use a basic assumption in the solution. Anyone that does not explicitly indicate capability to receive multiple media streams is assumed to only handle a single media, to avoid affecting legacy clients.

3.2. Receiver Limitations

An RTP end-point that intends to process the media in an RTP session needs to have sufficient resources to receive and process all the incoming streams. It is extremely likely that no receiver is capable to handle the theoretical upper limit of more than 4 billion media sources in an RTP session. Instead, one or more properties will limit the end-points' capabilities to handle simultaneous media streams. These properties are for example memory, processing, network bandwidth, memory bandwidth, or rendering estate to mention a few possible limitations.

We have also considered the issue of how many simultaneous non-active sources an end-point can handle. We cannot see that inactive media sending SSRCs result in significant resource consumption and there should thus be no need to limit them.
A potential issue that needs to be acknowledged is where a limited set of simultaneously active sources varies within a larger set of session members. As each media decoding chain may contain state, it is important that a receiver can flush a decoding state for an inactive source and if that source becomes active again it does not assume that this previous state exists. Thus, we see need for a signaling solution that allows a receiver to indicate its upper limit in terms of capability to handle simultaneous media streams. We see little need for an upper limitation of RTP session members. Applications will need to account for its own capability to use different codecs simultaneously when choosing general and payload specific limits.

3.3. Transmission Declarations

In an RTP based system where an end-point may either be legacy or has an explicit upper limit in the number of simultaneous streams, one will encounter situations where the end-point can not receive and process all simultaneous active streams in the session. Instead the sending end-points or central nodes, like RTP mixers, will provide the end-point with a selected set of streams based on various metrics, such as most active, most interesting, or user selected. In addition, the central node may combine multiple media streams using mixing or composition into a new media stream to enable an end-point to get sufficient source coverage in the session, despite existing limitations.
For such a system to be able to correctly determine the need for central processing, the capabilities needed for such a central processing node, and the potential need for an end-point to do sender side limitations, it is necessary for an end-point to declare how many simultaneous streams it may send. Thus, enabling negotiation of the number of streams an end-point sends.

4. Multiple Streams Extension

This section describes an extension of the media-level SDP attributes to support signaling of the end points multiple stream capabilities.

4.1. Signaling Support for Multiple Streams

A solution to the issues described in the previous section needs to:

* Enable signaling between the RTP sender and receiver how many simultaneous RTP streams that can be handled.

*Be able to handle the case where the number of RTP streams that can be sent from a client do not match the number of streams that can be received by the same client.

It is also a requirement that a multiple streams capable RTP sender MUST be able to adapt the number of sent streams to the RTP receiver capability.
For this purpose and for use in SDP, two new media-level SDP attributes are defined, max-send-ssrc and max-recv-ssrc, which can be used independently to establish a limit to the number of simultaneously active SSRCs for the send and receive directions, respectively. Active SSRCs are the ones counted as senders according to [RFC3550], i.e. they have sent RTP packets during the last two regular RTCP reporting intervals.
The syntax for the attributes is in ABNF [RFC5234]:

```
max-ssrc = "a="("max-send-ssrc:" / "max-recv-ssrc:") PT 1WSP limit
PT = "*" / 1*3DIGIT
limit = 1*8DIGIT
;WSP and DIGIT defined in [RFC5234]
```

A payload type-agnostic upper limit to the total number of simultaneous SSRCs that can be sent or received in this RTP session is signaled with a * instead of the payload type number. A value of 0 MAY be used as maximum number of SSRC, but it is then RECOMMENDED that this is also reflected using the sendonly or recvonly attribute. There MUST be at most one payload type-agnostic limit specified in each direction.
A payload type-specific upper limit to the total number of simultaneous SSRCs in the RTP session with that specific payload type is signaled with a defined payload type (static, or dynamic through rtpmap).
Multiple lines with max-send-ssrc or max-recv-ssrc attributes specifying a single payload type MAY be used, each line providing a limitation for that specific payload type. Payload types that are not defined in the media block MUST be ignored.
If a payload type-agnostic limit is present in combination with one or more payload type-specific ones, the total number of payload type-specific SSRCs are additionally limited by the payload type-agnostic number. When there are multiple lines with payload type-specific limits, the sender or receiver MUST be able to handle any combination of the SSRCs with different payload types that fulfill all of the payload type specific limitations, with a total number of SSRCs up to the payload type-agnostic limit.
When max-send-ssrc or max-recv-ssrc are not included in the SDP, it MUST be interpreted as equivalent to a limit of one, unless sendonly or recvonly attributes are specified, in which case the limit is implicitly zero for the corresponding unused direction.

4.2. Declarative Use

When used as a declarative media description, the specified limit in max-send-ssrc indicates the maximum number of simultaneous streams of the specified payload types that the configured end-point may send at any single point in time. Similarly, max-recv-ssrc indicates the maximum number of simultaneous streams of the specified payload types that may be sent to the configured end-point. Payload-agnostic limits MAY be used with or without additional payload-specific limits.

4.3. Use in Offer/Answer

When used in an offer [RFC3264], the specified limits indicate the agent's intent of sending and/or capability of receiving that number of simultaneous SSRCs. The answerer MUST reverse the directionality of recognized attributes such that max-send-ssrc becomes max-recv-ssrc and vice versa. The answerer SHOULD modify the offered limits in the answer to suit the answering client's capability and intentions. A sender MUST NOT send more simultaneous streams of the specified payload type than the receiver has indicated ability to receive, taking into account also any payload type-agnostic limit.
In case an answer fails to include any of the limitation attributes, the agent SHOULD be interpreted as capable of supporting only a single stream in the direction for which attributes are missing. If the offer lacks attributes it SHOULD be assumed that the offerer only supports a single stream in each direction. In case the offer lack both max-send-ssrc and max-recv-ssrc, they MUST NOT be included in the answer.

4.4. Examples

The SDP examples below are not complete. Only relevant parts have been included.

```
m=video 49200 RTP/AVP 99
a=rtpmap:99 H264/90000
a=max-send-ssrc:* 2
a=max-recv-ssrc:* 4
```

An offer with a stated intention of sending 2 simultaneous SSRCs and a capability to receive 4 simultaneous SSRCs.

```
m=video 50324 RTP/AVP 96 97
a=rtpmap:96 H264/90000
a=rtpmap:97 H263-2000/90000
a=max-recv-ssrc:96 2
a=max-recv-ssrc:97 5
a=max-recv-ssrc:* 5
```

An offer to receive at most 5 SSRCs, at most 2 of which using payload type 96 and the rest using payload type 97. By not including "max-send-ssrc" the value is implicitly set to 1.

```
m=video 50324 RTP/AVP 96 97 98
a=rtpmap:96 H264/90000
a=rtpmap:97 H263-2000/90000
a=rtpmap:98 H263/90000
a=max-recv-ssrc:96 2
a=max-recv-ssrc:97 3
a=max-recv-ssrc:98 5
a=max-recv-ssrc:* 5
```

An offer to receive at most 5 SSRCs, at most 2 of which using payload type 96, and at most 3 of which using payload type 97, and at most 5 using payload type 98. Permissible payload type combinations include those with no streams at all for one or more of the payload types, as well as a total number of SSRCs less than 5, e.g. two SSRCs with PT=96 and three SSRCs with PT=97, or one SSRC with PT=96, one with PT=97 and two with PT=98.

5. IANA Considerations

This document registers two media level SDP attributes.

6. Security Considerations

The SDP attributes defined in this document "a=max-recv-ssrc" and "a=max-send-ssrc" signals capabilities of the end-point. Thus they are vulnerable to attacks. The primary security concerns would be with third parties that modifies the values of the attributes or inserts the attributes in a signalling context. Thus changing the peers view of the others peers capabilities and proposals. A modification reducing either of send or receive values will degrade the service, potentially preventing the service all together. Increasing the value or inserting the attribute with a value different from 1 have the potential of being even more effective. It can result in that an end-point that only supports a single stream receives multiple streams. First of all potentially exposing software flaws regarding handling of multiple streams, thus causing crashes, less severe it can cause media degradation as the receiving entity flaps between media streams, or plays only a single one, where the other side assumes both will be played. In addition negotiation several streams has transport impact, potentially increasing the bit-rate consumed towards the end-point, and in addition forcing a adaptation response over a limited path thus degrading the media stream the end-point may play out.
To prevent third party manipulation of the SDP it should be source authenticated and integrity protected. The solution suitable for this depends on the signalling protocol being used. For SIP S/MIME *[RFC3261]*

are the ideal, and hop by hop TLS provides at least some protection, although not perfect. For SDP's retrieved using RTSP DESCRIBE *[RFC2326]* TLS would be the RECOMMENDED solution.

7. References

7.1. 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.
[RFC5234]	Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

7.2. Informative References

[RFC2326]	Schulzrinne, H., Rao, A. and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.
[RFC3264]	Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.
[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.
[RFC4103]	Hellstrom, G. and P. Jones, "RTP Payload for Text Conversation", RFC 4103, June 2005.
[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.

Authors' Addresses

Magnus Westerlund Westerlund Ericsson Farogatan 6 SE-164 80 Kista, Sweden Phone: +46 10 714 82 87 EMail: magnus.westerlund@ericsson.com

Bo Burman Burman Ericsson Farogatan 6 SE-164 80 Kista, Sweden Phone: +46 10 714 13 11 EMail: bo.burman@ericsson.com

Fredrik Jansson Jansson Ericsson
Farogatan 6 Kista, SE-164 80 Sweden Phone: +46 10 719 00 00 EMail: fredrik.k.jansson@ericsson.com