Network Working Group M. Westerlund
Internet-Draft B. Burman
Intended status: Standards Track Ericsson
Expires: January 5, 2012 July 4, 2011

RTP Multiple Stream Sessions and Simulcast
draft-westerlund-avtcore-multistream-and-simulcast-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 aimed only at 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 a problem with legacy. This issue is analyzed, and RTP clarifications and signalling extensions are proposed to handle this issue. A related issue is how to perform simulcast, in the meaning of sending multiple encodings or representations of the same media source, when using RTP for media transport. This is further analyzed and possible solutions discussed and we arrive at a conclusion for session multiplexing of simulcast versions. We also found a number of related issues when having multiple streams and simulcast.

Table of Contents

1. Introduction

This document looks at the issues of non basic usage of RTP where there is multiple media sources sent over an RTP session. This include multiple sources from the same end-point, multiple end-points each having a source, or due to an application that needs multiple encodings of a particular source. As will be shown these issues are interrelated and need a common discussion to ensure consistency.

After presenting the usages and the found issues the document goes on to discuss ways of solving the issues. These include both clarifications to the basic RTP behaviors and signalling extensions to be able to setup these session, also in the presence of legacy systems that are not assumed to have full support for multiple media streams within an RTP session.

This document proposes several general mechanisms that could be used independently in other use cases. We foresee that those proposals would in the end become independent but related documents in the relevant WGs of AVTCORE, AVTEXT and MMUSIC. However, at this stage when all these ideas are introduced we find it more useful to keep them together to ensure consistency and to make any relations clear, hopefully making it easier to find and resolve any issues in the area of multiple streams and simulcast.

1.1. Multiple Streams

RTP sessions are a concept which most fundamental part is a SSRC space. This space can encompass a number of network nodes and interconnect 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 a 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. In addition a single end-point may have multiple media sources of the same media type, like cameras or microphones.

However, the most common use cases has been point to point Voice over IP (VoIP) or streaming applications where there has commonly not been more than one media source per end-point. Even in conferencing applications, especially voice only, the conference focus or bridge has provided a single stream being a mix of the other participants to each participant. Thus there has been perceived little need for handling multiple SSRCs in implementations. This has resulted in an installed legacy base that isn't 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 signalling solutions around RTP, especially SDP based, hasn't considered the fundamental issues around 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 sources with media. 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 exist. The point is that a given end-point will have some limitations in the number of streams it simultaneously can receive, decode and playback. These limitations needs 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 stream. Todays SDP signalling support for this is basically the directionality attribute which indicates an end-point intend to send media or not. No indication of how many media streams.

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.

In addition to address the above set of issues we will also identify a number of issues related to multiple streams that should be addressed in the most suitable way. These include both obscurities in the RTP specification and short-comings in various signalling mechanisms that are exposed by multi-stream use cases.

1.2. Simulcast

Simulcast is the act of simultaneously sending multiple different versions of a media content. This can be done in several ways and for different purposes. This document focuses on the case where one wants to provide multiple different encodings towards a intermediary so that the intermediary can select which version to forward to other participants in the session. More discussion on the different ways of doing simulcast, which is the focus of this document in "Simulcast Usage and Applicability" (Section 3).

The different versions of a source content that can be simulcasted and that are considered in this document are:

Bit-rate: The primary difference is the amount of bits spent to encode the source and thus primarily affects the media signal to noise ratio (SNR).

Codec: Different media codecs are used to ensure that different receivers that do not have a common set of decoders can decode at least one of the versions. This includes codec configuration options that aren't compatible, like video encoder profiles, or the capability of receiving the transport packetization.

Sampling: Different sampling of media, in spatial as well as in temporal domain, may be used to suit different rendering capabilities or needs at receiving endpoints, as well as a method to achieve different bit-rates. For video streams, spatial sampling affects image resolution, and temporal sampling affects video framerate. For audio, spatial sampling relates to the number of audio channels, and temporal sampling affects audio bandwidth.

Different applications will have different reasons for providing a single media source in different versions. And as soon as an application have need for multiple versions for some reason, a potential need for simulcast is created. This need can arise even in media codecs that have scalability features built in to solve a set of variations.

The purpose of this document is to find the most suitable solution for the non-trivial variants of simulcast. To determine this, an analysis of different ways of multiplexing the different encodings are discussed in Section 6. Following the presentation of the alternatives, an analysis is performed in Section 7 on how different aspects like RTP mechanisms, signaling possibilities, and network features are affected by the alternatives.

The document ends with a recommendation for which solution is the most suitable and indicates what standardization work should be done if the WG agrees on the analysis and the suitability to define how simulcast should be done.

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

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. Simulcast Usage and Applicability

This section discusses different usage scenarios the term simulcast may refer to, and makes it clear which of those this document focuses on. It also reviews why simulcast and scalable codecs can be a useful combination.

3.1. Simulcasting to RTP Mixer

The usage here is in a multi-party session where one uses one or more central nodes to help facilitate the media transport between the session participants. Thus, this targets the RTP topology defined in [RFC5117] of RTP Mixer (Section 3.4: Topo-Mixer). This usage is one which is targeted for further discussion in this document.

Simulcasting different media encodings of video that has both different resolution and bit-rate is highly applicable to video conferencing scenarios. For example an RTP mixer selects the most active speaker and sends that participant’s media stream as a high

resolution stream to a receiver and in addition provides a number of small resolution video streams of any additional participants, thus enabling the receiving user to both see the current speaker in high quality and monitor the other participants. The active speaker gets a different combination of streams as it has limited use to get back the streams itself is sending. Thus, there can be several different combinations of high resolution and low resolution video in use simultaneously; requiring both a high and low resolution video from some sources at the same time.

For example, to provide both high and low resolution from an RTP Mixer there exist these potential alternatives:

Simulcast: The client sends one stream for the low resolution and another for the high resolution.

Scalable Video Coding: Using a video encoder that can provide one media stream that is both providing the high resolution and enables the mixer to extract a low resolution representation that has lower bit-rate than the full stream version.

Transcoding in the Mixer: The client transmits a high resolution stream to the RTP Mixer, which performs a transcoding to a lower resolution version of the video stream that is forwarded to the ones that need it.

The Transcoding requires that the mixer has sufficient amounts of transcoding resources to produce the number of low resolution versions required. This may in worst case be that all participants' streams needs transcoding. If the resources are not available, a different solution needs to be chosen.

The scalable video encoding requires a more complex encoder compared to non-scalable encoding. Also, if the resolution difference is big, the scalable codec may in fact be only marginally more bandwidth efficient, between the encoding client and the mixer, than a simulcast that sends the resolutions in separate streams, assuming equivalent video quality. At the same time, with scalable video encoding, the transmission of all but the lowest resolution will definitely consume more bandwidth from the mixer to the other participants than a non-scalable encoding, again assuming equivalent video quality.

Simulcasting has the benefit that it is conceptually simple. It enables use of any media codec that the participants agree on, allowing the mixer to be codec-agnostic. Considering today's video encoders, it is less bit-rate efficient in the path from the sending client to the mixer but more efficient in the mixer to receiver path

compared to Scalable Video Coding.

3.1.1. Simulcast Combined with Scalable Encoding

Scalable codecs are often used in arguments to motivate why simulcast isn't needed. A single media encoding that is sent as one joint media stream or divided up in base layers and enhancement layers over multiple transport is sufficient to achieve the desired functionality. As explained above in reality scalable codec is often not more efficient, especially in the path from the mixer to the receiver.

There are however, good reasons to combine simulcast with scalable encoding. By using simulcast to cover encoding variations where the scalable codec least efficient one can optimize the efficiency of the complete system. So a low number of simulcast working points, where each working point is in its turn a scalable codec configuration providing medium and/or fine grained scalability allowing a mixer to further tune the bit-rate to the available towards particular receivers using a combination of selecting simulcast versions and the number of extensions layers from that source.

A good example of this usage would be to send video encoded using SVC, where each simulcast version is a different resolution, and each SVC media stream uses temporal scalability and SNR scalability within that single media stream. If only resolution and temporal variations are needed, this can be implemented using H.264, as each simulcast version provides the different resolution, and each media stream within a simulcast encoding has temporal scalability using no-reference frames.

3.2. Simulcasting to Consuming End-Point

This usage is based on an RTP Transport Translator (Section 3.3: Topo-Trn-Translator) [RFC5117]. The transport translator functions as a relay and transmits all the streams received from one participant to all the other participants. In this case, one would do downlink simulcasting such that all receivers would receive all the versions. However, this clearly increases the bit-rate consumed on the paths to the client. The only benefit for the receiving client would be reduced decoding complexity when needing to only display a low resolution version. Otherwise a single stream application which only transmits the high resolution stream would allow the receiver to decode it and scale it down to the needed resolution.

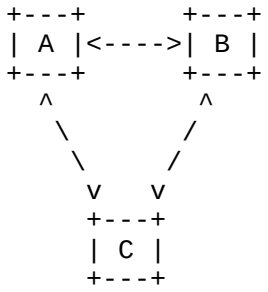
The usage of transport translator and simulcast becomes efficient if one allows each receiving client to control the relay to indicate

which version it wants to receive. However such a usage of RTP has some potential issues with RTCP. From the sending end-point it will look like the transmitted stream isn't received by a receiver that is known to receive other streams from the sender. Thus some consideration and mechanism are needed to support such a use case so that it doesn't break RTCP reception reporting.

This document will continue to consider this case but with less emphasis than on the RTP mixer case.

3.3. Same Encoding to Multiple Destinations

One interpretation of simulcast is when one encoding is sent to multiple receivers. This is well supported in RTP by simply copying all outgoing RTP and RTCP traffic to several transport destinations as long as the intention is to create a common RTP session. As long as all participants do the same, a full mesh is constructed and everyone in the multi party session has a similar view of the joint RTP session. This is analog to an Any Source Multicast (ASM) session but without the traffic optimization as multiple copies of the same content is likely to have to pass over the same link.



Full Mesh / Multi-unicast

As this type of simulcast is analog to ASM usage and RTP has good support for ASM sessions, no further consideration for this case is done.

3.4. Different Encoding to Independent Destinations

Another alternative interpretation of simulcast is with multiple destinations, where each destination gets a specifically tailored version, but where the destinations are independent. A typical example for this would be a streaming server distributing the same live session to a number of receivers, adapting the quality and resolution of the multi-media session to each receiver's capability and available bit-rate. This case can be solved in RTP by having

independent RTP sessions between the sender and the receivers. Thus this case is not considered further.

4. 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 signalling. 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.

4.1. Legacy behaviors

It is a common assumption among many applications using RTP that they don’t 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, they should support a source switching SSRC, e.g due to collision.

Some RTP extension mechanisms require the RTP stacks to handle additional SSRCs, like SSRC multiplexed RTP retransmission [RFC4588]. However, that still has only required handling a single media decoding chain.

However, there are applications that clearly can benefit from receiving and using multiple media streams simultaneously. A very basic case would be T.140 conversational text, which is both low bandwidth and where there is no simple method for mixing multiple sources of text that is supposed to be transmitted and displayed as you type. An RTP session that contains more than 2 SSRC actively sending media streams has the potential to confuse a legacy client in various ways:

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

These applications may be very similar to existing one media stream applications at signalling level. To avoid connecting two different

implementations, one that is built to support multiple streams and one that isn't, it is important that the capabilities are signalled. It is also the legacy that makes us use a basic assumption in the solution. Anyone that doesn't explicitly indicate capability to receive multiple media streams is assumed to only handle a single media, to avoid affecting legacy clients.

4.2. 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 an RTP session when it comes to more than 4 billion media sources. 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 this type of usage ensures 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 signalling 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 have some considerations around how they use codecs.

4.3. 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 will not receive all simultaneous active streams in the session. Instead the 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 a 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.

4.4. RTP and RTCP Issues

This section details a few RTP and RTCP issues identified in implementation work for supporting multiple streams.

4.4.1. Multiple Sender Reports in Compound

One potential interoperability issue is inclusion of multiple Sender Report blocks in the same RTCP compound packet. The RTP specification isn't clear if such stacking is allowed or not. Thus there might be RTCP receivers that might not correctly handle such message. There is also an uncertainty how one should calculate the RTCP transmission intervals in such cases.

4.4.2. Cross reporting within an end-point

When an end-point has more than one SSRC and sends media using them, a question arises if the different SSRCs needs to report on each other despite being local. It can be argued that it is needed due to that it might not be fully visible for any external observer that they are actually sent from the same end-point. Thus by reporting on each other there are no holes in the connectivity matrix between all sending SSRCs and all known SSRCs.

4.4.3. Which SSRC is providing feedback

When one has multiple SSRCs on an end-point and needs to send RTCP feedback messages some considerations around which SSRC is used as the source and if that is consistently used or not, may be needed.

4.5. SDP Signalling Issues

An existing issue with SDP is that the bandwidth parameters aren't specified to take asymmetric conditions into account. This becomes especially evident when we start using multiple streams in an RTP session. Such a use case can easily result in that an end-point maybe receive 5 streams of Full High Definition (HD) video but only sends one Standard Definition (SD) video stream. Thus easily having

a 10:1 asymmetry in bit-rate.

If one uses the current SDP bandwidth parameters then one likely needs to set the session bandwidth to the sum of the most consuming direction. This can result in that there is no way of negotiating an upper bound for the lower band-width direction media stream(s). In addition, an end-point may conclude that it can't support the bit-rate despite being capable of actually receiving the media streams being sent. Thus making clear what bandwidth limitations a single stream has compared to the whole RTP session is important.

In the cases there is QoS, either by end-point reservation or done by systems like IMS, the requested bandwidth based on the signalled value will not represent what is actually needed.

Asymmetry in itself also create an issue, as RTCP bandwidth may be derived from the session bandwidth. It is important that all end-points have a common view on what the RTCP bandwidth is. Otherwise if the bandwidth values are more than 5 times different, an end-point with the high bandwidth value may time out an end-point that has a low value as it's minimal reporting interval can become more than 5 times longer than for the other nodes.

5. Multi-Stream Extensions

5.1. Signaling Support for Multi-Stream

There is a need to signal between RTP sender and receiver how many simultaneous RTP streams can be handled. The number of RTP streams that can be sent from a client should not have to match the number of streams that can be received by the same client. A multi-stream 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 are in ABNF [RFC5234]:

```
max-ssrc = "a=" ("max-send-ssrc:" / "max-recv-ssrc:") PT 1*WSP limit
PT = "*" / 1*3DIGIT
limit = 1*8DIGIT
```

```
; WSP and DIGIT defined in [RFC5234]
```

A payload-agnostic upper limit to the total number of simultaneous SSRC that can be sent or received in this RTP session is signaled with a * payload type. 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-agnostic limit specified in each direction.

A payload-specific upper limit to the total number of simultaneous SSRC 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-agnostic limit is present in combination with one or more payload-specific ones, the total number of payload-specific SSRCs are additionally limited by the payload-agnostic number. When there are multiple lines with payload-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 specific limitations, with a total number of SSRCs up to the payload-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.

5.1.1. 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.

5.1.2. Use in Offer/Answer

When used in an offer, the specified limits indicates the agent’s intent of sending and/or capability of receiving that number of simultaneous SSRC. The answerer MUST reverse the directionality of recognized attributes such that max-send-ssrc becomes max-recv-ssrc

and vice versa. The answerer SHOULD decrease the offered limit in the answer to suit the answering client's capability. 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-agnostic limit.

In case an answer fails to include any of the limitation attributes, the agent MUST be interpreted as capable of supporting only a single stream in the direction for which attributes are missing. If the offer lacks attributes it MUST 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.

5.1.3. 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 SSRC, 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=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 SSRC, 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 SSRC less than 5, e.g. two SSRC with PT=96 and three SSRC with PT=97, or one SSRC with PT=96, one with PT=97 and two with PT=98.

5.2. Asymmetric SDP Bandwidth Modifiers

To resolve the issues around bandwidth, we propose new SDP bandwidth modifiers that supports directionality, possibility for payload specific values and clear semantics. A common problem for all the current SDP bandwidth modifiers is that they use a single bandwidth value without a clear specification. Uncertainty in how the bandwidth value is derived creates uncertainty on how bursty a media source can be.

Thus, we do consider what the design criteria are prior to providing a proposal for new SDP bandwidth attribute.

5.2.1. Design Criterias

The current b= SDP bandwidth syntax is very limited and only allows the following format:

```
bandwidth-fields  = *(%x62 "=" bwtype ":" bandwidth CRLF)
bwtype            = token
bandwidth         = 1*DIGIT
```

Thus we will need to specify a new SDP bandwidth attribute as that allows syntax of more complexity.

The functionalities we see from the new bandwidth attribute are the following:

Directionality: We need to be able to have different sets of attribute values depending on direction.

Bandwidth semantics: A semantics identifier so that new semantics can be defined in the future for other needed semantics. This part of the b= has been a very successful design feature. We do perceive a need for both single stream limitations and limitations for the aggregate of all streams in one direction.

Payload specific: The possibility to specify different bandwidth values for different RTP Payload types. This as some codecs have different characteristics and one may want to limit a specific codec and payload configuration to a particular bandwidth. Especially combined with codec negotiation there is a need to

express intentions and limitations on usage for that particular codec. In addition, payload agnostic information is also needed.

Bandwidth specification method: To have a clear specification of what any bit-rate values mean we propose that Token bucket parameters should be used, i.e. bucket depth and bucket fill rate, where appropriate for the semantics. If single values are to be specified, a clear definition on how to derive that value must be specified, including averaging intervals etc.

We will use these design criteria next in an actual proposal.

5.2.2. Attribute Specification

We define a new SDP attribute ("a=") as the bandwidth modifier line syntax can’t support the requirements and nor can it be changed in an interoperable way. Thus we define the "a=bw" attribute. This attribute is structured as follows. After the attribute name there is a directionality parameter, followed by a scope parameter and then a bandwidth semantics tag. The semantics tag defines what value(s) that follow and their interpretation.

The attribute is designed so that multiple instances of the line will be necessary to express the various bandwidth related configurations that are desired.

Scopes and semantics can be extended in the future at any point. To ensure that an end-point using SDP either in Offer/Answer or declarative truly understands these extensions, a required-prefix indicator ("!") can be added prior to any scope or semantics parameter.

5.2.2.1. Attribute Definition

The ABNF [RFC5234] for this attribute is the following:

```
bw-attrib     = "a=bw:" direction SP [req] scope SP
                [req] semantics ":" values
direction     = "send" / "recv" / "sendrecv"
scope         = payloadType / scope-ext
payloadType   = "PT=" ("*" / PT-value-list)
PT-value-list = PT-value *(";" PT-Value)
PT-value      = 1*3DIGIT
req           = "!"
semantics     = "SMT" / "AMT" / semantics-ext
values        = token-bucket / value-ext
token-bucket  = "tb=" br-value ":" bs-value
br-value      = 1*15DIGIT ; Bucket Rate
bs-value      = 1*15DIGIT ; Bucket Size

semantics-ext = token ; As defined in RFC 4566
scope-ext     = 1*VCHAR ; As defined in RFC 4566
value-ext     = 0*(WSP / VCHAR)
```

The a=bw attribute defines three possible directionalities:

send: In the send direction for SDP Offer/Answer agent or in case of declarative use in relation to the device that is being configured by the SDP.

recv: In the receiving direction for the SDP Offer/Answer agent providing the SDP or in case of declarative use in relation to the device that is being configured by the SDP.

sendrecv: The provided bandwidth values applies equally in send and recv direction, i.e. the values configures the directions symmetrically.

The Scope indicates what is being configured by the bandwidth semantics of this attribute line. This parameter is extensible and we begin with defining two different scopes based on payload type:

Payload Type: The bandwidth configuration applies to one or more specific payload type values.

PT=*: Applies independently of which payload type is being used.

This specification defines two semantics which are related. The Stream Maximum Token bucket based value (SMT) and the Aggregate Maximum Token bucket based value (AMT). Both semantics represent the bandwidth consumption of the stream or the aggregate as a token bucket. The token bucket values are the token bucket rate and the token bucket size, represented as two integer numbers. It is an open question exactly what this token bucket is measuring, if it is RTP

payload only, like TIAS, or if it includes all headers down to the IP level as most of the other bandwidth modifiers do.

The definition of the semantics in more detail are:

SMT: The maximum intended or allowed bandwidth usage for each individual source (SSRC) in an RTP session as specified by a token bucket. The token bucket values are the token rate in bits per second and the bucket size in bytes. This semantics may be used both symmetrically or in a particular direction. It can be used either to express the maximum for a particular payload type or for any payload type (PT=*).

AMT: The maximum intended or allowed bandwidth usage for sum of all sources (SSRC) in an RTP session according to the specified directionality as specified by a token bucket. The token bucket values are the token rate in bits per second and the bucket size in bytes. Thus if using the sendrecv directionality parameter, both send and receive streams SHALL be included in the generated aggregate. If only a send or recv, then only the streams present in that direction are included in the aggregate. It can be used either to express the maximum for a particular payload type or for any payload type (PT=*).

5.2.2.2. Offer/Answer Usage

The offer/answer negotiation is done for each bw attribute line individually with the scope and semantics immutable. If an answerer would like to add additional bw configurations using other directionality, scope, and semantics combination, it may add them.

An agent responding to an offer will need to consider the directionality and reverse them when responding to media streams using unicast. If the transport is multicast the directionality is not affected.

For media stream offers over unicast with directionality send, the answerer will reverse the directionality and indicate its reception bandwidth capability, which may be lower or higher than what the sender has indicated as its intended maximum.

For media stream offers over unicast with directionality receive, these do indicate an upper limit, the answerer will reverse the directionality and may only reduce the bandwidth when producing the answer indicating the answerer intended maximum.

[Need to define how the required "!" prefix is used in Offer/Answer]

5.2.2.3. Declarative Usage

In declarative usage the SDP attribute is interpreted from the perspective of the end-point being configured by the particular SDP. An interpreter MAY ignore a=bw attribute lines that contains unknown scope or semantics that does not start with the required ("!") prefix. If a "required" prefix is present at an unknown scope or semantics, the interpreter SHALL NOT use this SDP to configure the end-point.

5.2.2.4. Example

Declarative example with stream asymmetry.

```
m=video 50324 RTP/AVP 96 97 98
a=rtpmap:96 H264/90000
a=rtpmap:97 H263-2000/90000
a=rtpmap:98 MP4V-ES/90000
a=max-recv-ssrc:96 2
a=max-recv-ssrc:* 5
a=bw:send pt=* SMT:tb=1200000:16384
a=bw:recv pt=96 SMT:tb=1500000:16384
a=bw:recv pt=97:98 SMT:tb=2500000:16384
a=bw:recv pt=* AMT:tb=8000000:65535
```

In the above example the outgoing single stream is limited to bucket rate of 1.2 Mbps and bucket size of 16384 bytes. The up to 5 incoming streams can in total use maximum 8 Mbps bucket rate and with a bucket size of 65535 bytes. However, the individual streams maximum rate is depending on payload type. Payload type 96 (H.264) is limited to 1.5 Mbps with a bucket size of 16384 bytes, while the Payload types 97 (H.263) and 98 (MPEG-4) may use up top 2.5 Mbps with a bucket size of 16384 bytes.

5.3. Binding SSRCs Across RTP Sessions

When an end-point transmits multiple sources in the same RTP session there may be tight relations between two different media types and their SSRCs, for example a microphone and a camera that is co-located are tightly related. CNAME is not sufficient to express this relation although it is commonly inferred from end-points that has only one media stream per media type. CNAME primary use in multi-source usages is to indicate which end-point and what synchronization context a particular media stream relates to.

To enable a RTP session participant to determine that close binding across multiple sessions, despite the end-point sending multiple SSRCs a new method for identifying such sources are needed. We are

not relying on using the same SSRC in all sessions for a particular media source as it is not robust against SSRC collision and forces potentially cascading SSRC changes between sessions.

5.3.1. SDES Item SRCNAME

Source Descriptions are a method that should work with all RTP topologies (assuming that any intermediary node is supporting this item) and existing RTP extensions. Thus we propose one defines a new SDES item called the SRCNAME which identifies with an unique identifier a single multi-media source, like a camera and a co-located microphone, or a truly individual media source such as a camera. That way any one receiving the SDES information from a set of interlinked RTP sessions can determine which are the same source.

We proposes that the SRCNAME would commonly be per communication session unique random identifiers generated according to "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)" [RFC6222] with the addition that a local counter enumerating the sources on the host also are concatenated to the key in step 4 prior to calculating the hash.

This SRCNAME's relation to CNAME is the following. CNAME represents an end-point and a synchronization context. If the different sources identified by SRCNAMEs should be played out synchronized when receiving them in a multi-stream context, then the sources need to be in the same synchronization context. Thus in all cases, all SSRCs with the same SRCNAME will have the same CNAME. A given CNAME may contain multiple sets of sources using different SRCNAMEs.

5.3.2. SRCNAME in SDP

Source-Specific Media Attributes in the Session Description Protocol (SDP) [RFC5576] defines a way of declaring attributes for SSRC in each session in SDP. With a new SDES item, one can use this framework to define how also the SRCNAME can be provided for each SSRC in each RTP session, thus enabling an end-point to declare and learn the simulcast bindings ahead of receiving RTP/RTCP packets.

6. Simulcast Alternatives

Simulcast is the act of sending multiple alternative encodings of the same underlying media source. When transmitting multiple independent flows that originate from the same source, it could potentially be done in several different ways in RTP. The below sub-sections describe potential ways of achieving flow de-multiplexing and identification of which streams are alternative encodings of the same

source.

In the below descriptions we also include how this interacts with multiple sources (SSRCs) in the same RTP session for other reasons than simulcast. So multiple SSRCs may occur for various reasons such as multiple participants in multipoint topologies such as multicast, transport relays or full mesh transport simulcasting, multiple source devices, such as multiple cameras or microphones at one end-point, or RTP mechanisms in use, such as RTP Retransmission [RFC4588].

6.1. Payload Type Multiplexing

Payload multiplexing uses only the RTP payload type to identify the different alternatives. Thus all alternative streams would be sent in the same RTP session using only a single SSRC per actual media source. So when having multiple SSRCs, each SSRC would be unique media sources or RTP mechanism-related SSRC. Each RTP payload type would then need to both indicate the particular encoding and its configuration in addition to being a stream identifier. When considering a mechanism like RTP retransmission using SSRC multiplexing, an SSRC may either be a media source with multiple encodings as provided by the payload type, or a retransmission packet as identified also by the payload type.

As some encoders, like video, produce large payloads one can not expect that multiple payload encodings can fit in the same RTP packet payload. Instead a payload type multiplexed simulcast will need to send multiple different packets with one version in each packet or sequence of packets.

6.2. SSRC Multiplexing

The SSRC multiplexing idea is based on using a unique SSRC for each alternative encoding of one actual media source within the same RTP session. The identification of how flows are considered to be alternative needs an additional mechanism, for example using SSRC grouping [RFC5576] and a new SDES item such as SRCNAME proposed in Section 5.3.1 with a semantics that indicate them as alternatives of a particular media source. When one have multiple actual media sources in a session, each media source will use a number of SSRCs to represent the different alternatives it produces. For example, if all actual media sources are similar and produce the same number of simulcast versions, one will have n*m SSRCs in use in the RTP session, where n is the number of actual media sources and m the number of simulcast versions they can produce. Each SSRC can use any of the configured payload types for this RTP session. All session level attributes and parameters which are not source specific will apply and must function with all the alternative encodings intended

to be used.

6.3. Session Multiplexing

Session multiplexing means that each different version of an actual media source is transmitted in a different RTP session, using whatever session identifier to de-multiplex the different versions. This solution needs explicit session grouping [RFC5888] with a semantics that indicate them as alternatives. When there are multiple actual media sources in use, the SSRC representing a particular source will be present in the sessions for which it produces a simulcast version. It is also important to identify the SSRCs in the different sessions that are alternative encodings to each other, this can be accomplished using the same SSRC and/or a new SDES item identifying the media source across the session as the proposed SRCNAME SDES item (Section 5.3.1). Each RTP session will have its own set of configured RTP payload types where each SSRC in that session can use any of the configured ones. In addition all other attributes for sessions or sources can be used as normal to indicate the configuration of that particular alternative.

7. Simulcast Evaluation

This chapter evaluates the different multiplexing strategies in regard to several aspects.

7.1. Effects on RTP/RTCP

This section will be oriented around the different multiplexing mechanisms.

7.1.1. Payload Type Multiplexing

The simulcast solution needs to ensure that the negative impact on RTP/RTCP is minimal and that all the features of RTP/RTCP and its extensions can be used.

Payload type multiplexing for purposes like simulcast has well known negative effects on RTP. The basic issue is that all the different versions are being sent on the same SSRC, thus using the same timestamp and sequence number space. This has many effects:

1. Putting restraint between media encoding versions. For example, media encodings that uses different RTP timestamp rates cannot be combined as the timestamp values needs to be the same across all versions of the same media frame. Thus they are forced to use the same rate. When this is not possible, Payload Type

Multiplexing cannot be used.

2. Most RTP payload formats that may fragment a media object over multiple packets, like parts of a video frame, needs to determine the order of the fragments to correctly decode them. Thus it is important that one ensure that all fragments related to a frame or a similar media object are transmitted in sequence and without interruptions within the object. This can relatively simple be solved by ensuring that each version is sent in sequence.

3. Some media formats require uninterrupted sequence number space between media parts. These are media formats where any missing RTP sequence number will result in decoding failure or invoking of a repair mechanism within a single media context. The text/T140 payload format [RFC4103] is an example of such a format. These formats will be impossible to simulcast using payload multiplexing.

4. Sending multiple versions in the same sequence number space makes it more difficult to determine which version a packet loss may relate to. If one uses RTP Retransmission [RFC4588] one can ask for the missing packet. However, if the missing packet(s) do not belong to the version one is interested in, the retransmission request was in fact unnecessary.

5. The current RTCP feedback mechanisms are built around providing feedback on media streams based on stream ID (SSRC), packet (sequence numbers) and time interval (RTP Timestamps). There is almost never a field for indicating which payload type one is reporting on. Thus giving version specific feedback is difficult.

6. The current RTCP media control messages [RFC5104] specification is oriented around controlling particular media flows, i.e. requests are done addressing a particular SSRC. Thus such mechanisms needs to be redefined to support payload type multiplexing.

7. The number of payload types are inherently limited. Accordingly, using payload type multiplexing limits the number of simulcast streams and does not scale.

7.1.2. SSRC Multiplexing

As each version of the source has its own SSRC and thus explicitly unique flows, the negative effects above (Section 7.1.1) are not present for SSRC multiplexed simulcast.

The SSRC multiplexing of simulcast version requires a receiver to know that one is expected to only decode one of the versions and need not decode all of them simultaneously. This is currently a missing functionality as SDES CNAME cannot be used. The same CNAME has to be used for all flows connected to the same end-point and location. A clear example of this could be video conference where an end-point has 3 video cameras plus an audio mix being captured in the same room. As the media has a common timeline, it is important to be able to indicate that through the CNAME. Thus one cannot use CNAME to indicate that multiple SSRCs with the same CNAME are different versions of the same source. New semantics are required.

When one has all the versions in the same RTP session going to an RTP mixer and the mixer chooses to switch from forwarding one of the versions to forwarding another version, this creates an uncertainty in which SSRC one should use in the CSRC field (if used). As one is still delivering the same original source, such switch appears questionable to a receiver not having enabled simulcast in the direction to itself. Depending on what solution one chooses, one gets different effects here. If the CSRC is changed, then any message ensuring binding will need to be forwarded by the mixer, creating legacy issues. It has not been determined if there are downsides to not showing such a switch.

The impact of SSRC collisions on the SSRC multiplexing will be highly depending on what method is used to bind the SSRCs that provide different versions. Upon a collision and a forced change of the SSRC, a media sender will need to re-establish the binding to the other versions. By doing that, it will also likely be explicit when it comes to what the change was.

7.1.3. Session Multiplexing

Also session multiplexing does not have any of the negative effects that payload type multiplexing has (Section 7.1.1). As each flow is uniquely identified by RTP Session and SSRC, one can control and report on each flow explicitly. The great advantage of this method is that each RTP session appears just like if simulcast is not used thus minimal issues in RTP and RTCP including any extensions.

One potential downside of session multiplexing is that it becomes impossible without defining new RTCP message types to do truly synchronized media requests where one request goes to version A of source and another to version B of the same source. Due to the RTP session separation, one will be forced to send different RTCP packets to the different RTP session contexts, thus losing the ability to send two different RTCP packets in the same compound packet and RTP session context. This can be a minor inconvenience.

Using the same SSRC in all the RTP sessions allows for quick binding between the different versions. It also enables an RTP mixer that forwards one version to seamlessly decide to forward another version in a RTP session to a session participant that is not using simulcast in the direction from the mixer to the participant.

An SSRC collision forces a sender to change its SSRC in all sessions. Thus the collision-induced SSRC change may have bigger impact, as it affects all versions rather than a single version. But on the positive side, the binding between the versions will be immediate, rather than requiring additional signaling.

7.2. Signaling Impact

The method of multiplexing has significant impact on signaling functionality and how to perform it, especially if SDP [RFC4566] and SDP Offer/Answer [RFC3264] is used.

7.2.1. Negotiating the use of Simulcast

There will be a need for negotiating the usage of simulcast in general. For payload type multiplexing, one will need to indicate that different RTP payload types are intended as different simulcast versions. One likely has standalone SDP attributes that indicate the relation between the payload types, as one needs unique payload type numbers for the different versions. Thus, this increases the number of payload types needed within an RTP session. In worst case this may become a restriction as only 128 payload types are possible. This limitation is exacerbated if one uses solutions like RTP and RTCP multiplexing [RFC5761] where a number of payload types are blocked due to the overlap between RTP and RTCP.

SSRC multiplexing will likely use a standalone attribute to indicate the usage of simulcast. In addition, it may be possible to use a mechanism in SDP that binds the different SSRCs together. The first part is non-controversial. However the second one has significant impact on the signaling load in sessions with dynamic session participation. As each new participant joins a multiparty session, the existing participants that need to know the binding will need to receive an updated list of bindings. If that is done in SIP and SDP offer answer, a SIP re-Invite is required for each such transaction, invoking all the SIP nodes related to invites, and in systems like IMS also a number of policy nodes. If a receiver is required, which is likely, to receive the SSRC bindings prior to being able to decode any new source, then the signaling channel may introduce additional delay before a receiver can decode the media.

Session multiplexing results in one media description per version.

It will be necessary to indicate which RTP sessions are in fact simulcast versions. For example, using a Media grouping semantics specific for this. Each of these sessions will be focused on the particular version they intend to transport.

Legacy fallback, the impact on an end-point that isn't simulcast enabled, also needs to be considered. For a payload type multiplex solution, a legacy end-point that doesn't understand the indication that different RTP payload types are for different purpose may be slightly confused by the large amount of possibly overlapping or identical RTP payload types. In addition, as payload multiplexing isn't backwards compatible within a single media stream, the signalling needs to ensure that such a legacy client doesn't join a session using simulcast.

For an SSRC multiplexed session, a legacy end-point will ignore the SSRC binding signaling. From its perspective, this session will look like an ordinary session and it will setup to handle all the versions simultaneously. Thus, a legacy client is capable of decoding and rendering a simulcast enabled RTP session, but it will consume more resources and result in a duplication of the same source.

For session multiplexing, a legacy end-point will not understand the grouping semantic. It might either understand the grouping framework and thus determine that they are grouped for some purpose, or not understand grouping at all and then the offer simply looks like several different media sessions. This enables a simple fallback solution to exclude a legacy client from all simulcast versions except one, whichever is most suitable for the application.

7.2.2. Bandwidth negotation

The payload type multiplexed session cannot negotiate bandwidth for the individual versions without extensions. The regular SDP bandwidth attributes can only negotiate the overall bandwidth that all versions will consume. This makes it difficult to determine that one should drop one or more versions due to lack of bandwidth between the peers.

SSRC multiplexing suffers the same issues as payload type multiplexing, unless additional signaling (SSRC level attributes) is added.

Session multiplexing can negotiate bandwidth for each individual version and determine to exclude a particular version, and have the full knowledge on what it excludes to avoid consuming an excessive amount of bandwidth.

7.2.3. Negotation of media parameters

The negotiation and setting of the media codec, the codec parameters and RTP payload parameters for the payload type multiplexing is possible for each individual version as each has a unique payload type. The same is true for the session multiplexing where each version negotiates the parameters in the context of it's RTP session. The SSRC multiplexed version would need additional signaling to enable a binding between the payload types and which versions they are used for. Otherwise, the RTP payload types are negotiated without any context of which version intends to use which payload type.

However, the above assumes that there are no issues with defining different payload types for different alternative encodings. If that is not possible or it is intended to use the same payload type for multiple encodings, then additional signalling becomes necessary which isn't possible for payload multiplexing. For SSRC multiplexing, this signalling needs to redefine already existing session attributes, like imageattr [RFC6236] to have a per-SSRC scope. Session multiplexing can use existing attributes as they automatically get per-encoding scope thanks to the session multiplexing.

7.2.4. Negotation of RTP/RTCP Extensions

When one negotiates or configures the existing RTP and RTCP extensions, that can be done on either session level or in direct relation to one or several RTP payload types. They are not negotiated in the context of an SSRC. Thus payload type multiplexing will need to negotiate any session level extensions for all the versions without version specific consideration, unless extensions are deployed. It can also negotiate payload specific versions at a version individual level. SSRC multiplexing cannot negotiate any extension related to a certain version without extensions. Session multiplexing will have the full freedom of negotiating extensions for each version individually without any additional extensions.

7.3. Network Aspects

The multiplexing choice has impact on network level mechanisms.

7.3.1. Quality of Service

When it comes to Quality of Service mechanisms, they are either flow based or marking based. RSVP [RFC2205] is an example of a flow based mechanism, while Diff-Serv [RFC2474] is an example of a Marking based one. If one uses a marking based scheme, the method of multiplexing

will not affect the possibility to use QoS. However, if one uses a flow based one, there is a clear difference between the methods. Both Payload Type and SSRC multiplexing will result in all versions being part of the same 5-tuple (protocol, source address, destination address, source port, destination port) which is the most common selector for flow based QoS. Thus, separation of the level of QoS between versions is not possible. That is however possible if one uses session based multiplexing, where each different version will be in a different RTP context and thus commonly being sent over different 5-tuples.

7.3.2. NAT Traversal

Both the payload and SSRC multiplexing will have only one RTP session, not introducing any additional NAT traversal complexities compared to not using simulcast and only have a single version. The session multiplexing is using one RTP session per simulcast version. Thus additional lower layer transport flows will be required unless an explicit de-multiplexing layer is added between RTP and the transport protocol.

Below we analyzed and comment on the impact of requiring more underlying transport flows in the presence of NATs and Firewalls:

End-Point Port Consumption: A given IP address only has 65536 available local ports per transport protocol for any consumer of ports that exist on the machine. This is normally never an issue for a end-user machine. It can become an issue for servers that have large number of simultaneous flows. However, if the application uses ICE, which authenticated STUN requests, a server can serve multiple end-point from the same local port, and use the whole 5-tuple (source and destination address, source and destination port, protocol) as identifier of flows after having securely bound them to end-points using the STUN request. Thus in theory the minimal number of media server ports needed are the maximum number of simultaneous RTP sessions a single end-point may use, when in practice implementation will probably benefit from using more.

NAT State: If an end-point is behind a NAT each flow it generates to an external address will result in a state on that NAT. That state is a limited resource, either from memory or processing stand-point in home or SOHO NATs, or for large scale NATs serving many internal end-points, the available external ports run-out. We see this primarily as a problem for larger centralized NATs where end-point independent mapping do require each flow mapping to use one port for the external IP address. Thus affecting the maximum aggregation of internal users per external IP address.

However, we would like to point out that a real-time video conference session with audio and video are likely using less than 10 UDP flows, it is not like certain web applications that can result that 100+ TCP flows are opened to various servers from a single browser instance.

NAT Traversal taking additional time: When doing the NAT/FW traversal it takes additional time. And it takes time in a phase of communication between accepting to communicate and the media path being established which is fairly critical. The best case scenario for how much extra time it can take following the specified ICE procedures are: 1.5*RTT + Ta*(Additional_Flows-1), where Ta is the pacing timer, which ICE specifies to be no smaller than 20 ms. That assumes a message in one direction, and then an immediate triggered check back. This as ICE first finds one candidate pair that works prior to establish multiple flows. Thus, there are no extra time until one has found a working candidate pair. Based on that working pair the extra time it takes, is what it takes to in parallel establish the additional flows which in most case are 2-3 additional flows.

NAT Traversal Failure Rate: Due to that one need more than a single flow to be established through the NAT there is some risk that one succeed in establishing the first flow but fails with one or more of the additional flows. The risk that this happens are hard to quantify. However, that risk should be fairly low as one has just prior successfully established one flow from the same interfaces. Thus only rare events as NAT resource overload, or selecting particular port numbers that are filtered etc, should be reasons for failure.

As most simulcast solutions will anyway not use a very large number of simulcast versions due to the cost in encoding resources etc. one can discuss if the extra transport flows are a significant cost. We perceive the cost as low, if others are concluding that the cost is higher, a more generalized mechanism for multiplexing RTP sessions onto the same underlying transport flow should be considered.

7.4. Summary

It is quite clear from the analysis that payload type multiplexing is not at all a realistic option for using simulcast. It has many issues, especially on RTP/RTCP level. Thus, we will not consider it a viable solution in further discussions below.

Both SSRC and session multiplexing are viable to use. However, session multiplexing provides increased flexibility in usage, better support for network QoS, signalling flexibility, and support compared

to SSRC multiplexing, without defining additional extensions. Session multiplexing does however require additional NAT/FW pinholes to be opened or some other solution to allow multiple RTP sessions to share the same transport flow, but that is anyway something that already happens in today's applications.

The authors consider the impact on the signalling one of the most significant issues when it comes to SSRC multiplexing. For many use cases, selecting SSRC multiplexing will require us to define numerous signalling mechanisms to support binding such properties to specific SSRCs or encoding groups. This signalling already exists today for non simulcast RTP sessions or for simulcast in a session multiplexing context.

Session multiplexing is in the authors view clearly the best choice and is therefore recommended to be pursued as the single solution for simulcast.

8. Simulcast Extensions

This section discusses various extensions that either are required or could provide system performance gains if they where specified.

8.1. Signalling Support for Simulcast

To enable the usage of simulcast using session multiplexing some minimal signalling support is required. That support is discussed in this section. First of all, there is need for a mechanism to identify the RTP sessions carrying simulcast alternatives to each other. Secondly, a receiver needs to be able to identify the SSRC in the different sessions that are of the same media source but in different encodings.

Beyond the necessary signalling support for simulcast we look at some very useful optimizations in regards to the transmission of media streams and to help RTP mixers to select which stream alternatives to deliver to a specific client, or request a client to encode in a particular way.

8.1.1. Grouping Simulcast RTP Sessions

The proposal is to define a new grouping semantics for the session groupings framework [RFC5888]. There is a need to separate the semantics of intent to send simulcast streams from the capability to recognize and receive them. For that reason two new simulcast grouping tags are defined, "SimulCast Receive" (SCR) and "SimulCast Send" (SCS). They both act as an indicator that session level

simulcast is occurring and which sets of RTP sessions that carries simulcast alternatives to each other.

The grouping semantics SCR and SCS SHOULD be combined with the SDP attributes "a=max-send-ssrc" and "a=max-recv-ssrc" Section 5.1 to indicate the number of simultaneous streams of each encoding that may be sent or capable of receiving.

8.1.1.1. Declarative Use

When used as a declarative media description, SCR indicates the configured end-points required capability to recognize and receive a specified set of RTP streams as simulcast streams. In the same fashion, SCS request the end-point to send a specified set of RTP streams as simulcast streams. SCR and SCS MAY be used independently and at the same time and they need not specify the same or even the same number of RTP sessions in the group.

8.1.1.2. Offer/Answer Use

When used in an offer, SCS indicates the SDP providing agent’s intent of sending simulcast, and SCR indicates the agent’s capability of receiving simulcast streams. SCS and SCR MAY be used independently and at the same time and they need not specify the same or even the same number of RTP sessions in the group. The answerer MUST change SCS to SCR and SCR to SCS in the answer, given that it has and wants to use the corresponding (reverse) capability. An answerer not supporting the SCS or SCR direction, or not supporting SCS or SCR grouping semantics at all, will remove that grouping attribute altogether, according to [RFC5888]. An offerer that receives an answer indicating lack of simulcast support in one or both directions, where SCR and/or SCS grouping are removed, MUST NOT use simulcast in the non-supported direction(s).

8.1.2. Binding SSRCs Across RTP Sessions

When one performs simulcast, a transmitting end-point will for each actual media source have one SSRC in each session for which it currently provides an encoding alternative. As a receiver or a mixer will receive one or more of these, it is important that any RTP session participant beyond the sender can explicitly identify which SSRCs in the set of RTP sessions providing a simulcast service for a particular media type that originate from the same media source and thus belong together in the simulcast.

To accomplish this we extend the usage of SRCNAME as defined in Section 5.3.1. Within a particular media type the different RTP session carrying the different encodings will have the same SRCNAME

identifier. That way even if multiple encodings or representations are produced, any one receiving the SDES information from a set of interlinked RTP sessions can determine which are the same source.

8.2. Mixer Requests of Client streams

To increase the efficiency of simulcast systems, it is highly desirable that an RTP middlebox can signal to the client encoding and transmitting the streams if a particular stream is currently needed or not. This needs to be a quick and media plane oriented solution as it changes based on for example the user's speech activity or the user's selection in the user interface. Although several SIP and SDP-based methods would be possible, the required responsiveness suggests use of TMMBR from [RFC5104] with a bandwidth value of 0 to temporarily pause a certain SSRC and re-establishing transmission through TMMBR with a non-zero value.

8.3. Client to Mixer and Mixer to Client limiations

When a client has known limitations, for example based on local display layout between sources or if there is a better combination of streams from the available set of different encodings, then it is desirable to make these limitations known to the mixer delivering the streams. These limitations are also clearly dynamic, as sources may come or leave the session, making it prefer a different layout with another set of limitations in the delivered streams.

The Codec Control Messages in [RFC5104] defines some controls. However, with the addition of simulcast and scalable video there are more parameters that would be desired to control in a way similar to the Temporary Maximum Media Stream Bit Rate (TMMBR) messages, beyond just bit-rate. Factors such as largest image dimension and frame rate will also be needed, for example. In the context of simulcast, one also needs to consider if a limitation is not specific to an SSRC, but rather which encoding and scalability variation is most suitable from a particular media source (SRCNAME).

Thus we propose that new RTCP messages are defined to temporarily limit media source with respect to a combination of media stream properties such as for example bit-rate, frame-rate, image resolution, and audio channels. Such a message should be flexible enough to allow for additional limitation attributes.

8.4. Multiplexing Multiple RTP Sessions on Single Flow

It should be considered for RTP in non-legacy cases if multiple RTP sessions could be multiplexed in a standardized way on top of a single transport layer flow. That way the cost of opening additional

transport flows and the needed NAT/FW traversal would be avoided. We acknowledge that this has impact on use cases using a flow based QoS mechanism that needs differentiated service levels between sessions. Such a mechanism should thus be optional to use, but as there is likely a general interest in such a mechanism, work on this should be started.

8.5. Examples

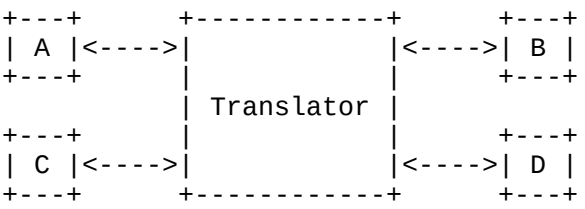
This section contains some SDP examples combining the proposals in this document to accomplish actual usages. We have skipped both NAT traversal tools as well as using the AVPF RTP profile [RFC4585] and Codec Control Messages [RFC5104] to save space in the SDPs, they are bulky enough. However, all these tools are likely to be part of a real SDP.

8.5.1. Multi-stream Signaling

This section contains examples of signalling for an application using multiple streams within an RTP session in two different contexts. In both these cases, the end-point that is involved in the signalling receives multiple streams, while only in the second case will the end-point transmit multiple streams.

8.5.1.1. Local Rendering in Video Conference Client

This example assumes a transport translator that enables the end-point to receive multiple streams from the other participants without using multiple destinations on transport level.



Four-party Translator-based Conference

Example of Media plane for RTP transport translator based multi-party conference with 4 participants.

Client A (Alice) in above figure is a desktop video conference client with a single camera and microphone. It uses a central transport translator to relay its media streams to the other participants, and in the same way it receives media streams from all other participants from the relay. This enables the client to locally render and

present other participants in a layout selected by the local client.

The network path between client A and the translator has certain known limitations, leading to a client needing to express its upper bounds in simultaneous streams that can be supported. That allows the conference server to know when it needs to tell the media plane relay to change its behavior from relaying to switching the media streams.

Alice invites herself into the conference by sending the following SDP offer:

```
v=0
o=alice 2890844526 2890842807 IN IP4 192.0.2.156
s=Multi stream Invite
c=IN IP4 192.0.2.156
b=AS:3530
t=0 0
m=audio 49200 RTP/AVP 96 97 9 8
b=AS:1450
a=rtpmap:96 G719/48000/2
a=rtpmap:97 G719/48000
a=rtpmap:9 G722/8000
a=rtpmap:8 PCMA/8000
a=bw:send pt=96 SMT:tb=128800:1500
a=bw:send pt=97 SMT:tb=64800:1500
a=bw:send pt=8;9 SMT:tb=64000:1500
a=bw:recv pt=* AMT:tb=1288000:1500
a=max-recv-ssrc:* 10
a=ssrc:834512974 cname:alice@foo.example.com
m=video 49300 RTP/AVP 96
b=AS:2080
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c01e
a=imageattr:* send [x=640,y=360] recv [x=640,y=360] [x=320,y=180]
a=bw:send pt=96 SMT:tb=500000:8192
a=bw:recv pt=96 SMT:tb=500000:8192
a=max-recv-ssrc:* 4
a=ssrc:451297483 cname:alice@foo.example.com
a=content:main
```

Alice Offer for a Multi-stream Conference

In the above SDP, Alice proposes one audio and one video RTP session. The audio session has 4 payload types being configured and the different payload configurations also show Alice’s intentions of their different bandwidth usage. For the audio receive direction, Alice accepts an aggregate bandwidth of 1288 kbps with a 1500 byte

bucket depth. This is sufficient bandwidth for 10 simultaneous streams. This limit of up to 10 streams being received is additionally indicated on SSRC level using the a=max-recv-ssrc attribute. The send limitation is implicitly set to one by excluding the a=max-send-ssrc attribute. Alice also declares the cname for the SSRC she intends to use.

The video session has only a single payload format using H.264. The configured profile and level is sufficient to support multiple resolutions of interest for the application. Alice indicates the intention to send 640x360 resolution and requests to receive either 640x360 or 320x180. The bandwidth for the video is expressed as the same 500 kbps upper limit in both send and receive directions, with an 8192 bytes bucket depth. There is no explicit limitation on the aggregate bandwidth. Alice does however express that she cannot handle receiving more than 4 simultaneous active SSRCs, so there is an implicit limit.

The application server controlling the conference receives the Offer and constructs a response based on knowledge about the conference and the available translator.

```
v=0
o=server 39451234544 39451234578 IN IP4 198.51.100.2
s=Multi stream Alice Answer
c=IN IP4 198.51.100.43
b=AS:2950
t=0 0
m=audio 49200 RTP/AVP 96 97 9
b=AS:870
a=rtpmap:96 G719/48000/2
a=rtpmap:97 G719/48000
a=rtpmap:9 G722/8000
a=bw:recv pt=96 SMT:tb=128800:1500
a=bw:recv pt=97 SMT:tb=64800:1500
a=bw:recv pt=9 SMT:tb=64000:1500
a=bw:send pt=* AMT:tb=500000:1500
a=max-send-ssrc:* 6
a=ssrc:239245219 cname:bob@foo.example.com
a=ssrc:986545121 cname:dave@foo.example.com
a=ssrc:2199983234 cname:fred@foo.example.com
m=video 49300 RTP/AVP 96
b=AS:2080
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c01e
a=imageattr:* recv [x=640,y=360] send [x=640,y=360] [x=320,y=180]
a=bw:recv pt=96 SMT:tb=500000:8192
a=bw:send pt=96 SMT:tb=500000:8192
a=max-send-ssrc:* 4
a=ssrc:924521923 cname:bob@foo.example.com
a=ssrc:654512198 cname:dave@foo.example.com
a=ssrc:3234219998 cname:fred@foo.example.com
a=content:main
```

SDP Answer to Alice from application server

The application server accepts both audio and video RTP sessions. It removed the a-law PCM format as it isn’t needed in this conference. It also reduces the number of simultaneous streams that may occur to 6 by setting the a=max-send-ssrc attribute to 6. The aggregate bandwidth that the client may receive, i.e. what the server declares as send, is limited down 500 kbps with a bucket depth of 1500 bytes. The SSRC values and their CNAMEs from the 3 already connected clients, bob, dave and fred are also included.

The video session is accepted as is, indicated by reversing the directions on the parts that indicates direction in the bw attribute and the imageattr. The max-recv-ssrc is changed to max-send-ssrc to indicate that there may be up to 4 simultaneous sources from the translator down to alice. The SSRCs and the corresponding CNAMEs are

also declared for video allowing for audio and video to be bound together, enabling synchronization before receiving the first RTCP sender reports.

8.5.1.2. Multiple Sources from Telepresence Room

In this use case Alice is an end-point which is a telepresence room. It has 3 cameras to cover different parts of the room's table. It also has directional microphones for each camera sector, such that it requests to send 3 streams of audio to maintain audio to screen bindings. If this is not possible, a stereo field sound mix can be provided instead that covers all three cameras.

Alice communicates directly with another single telepresence room end-point, Bob, but with only 2 cameras and microphones. However, Bob can receive 3 simultaneous streams and can use them in the local playout layout.

Alice invites herself into the conference by sending the following SDP offer:

```
v=0
o=alice 2890844526 2890842807 IN IP4 192.0.2.156
s=Telepresence Alice Invite
c=IN IP4 192.0.2.156
b=AS:8965
t=0 0
m=audio 49200 RTP/AVP 97 96
b=AS:725
a=rtpmap:96 G719/48000/2
a=rtpmap:97 G719/48000
a=bw:send pt=96 SMT:tb=128800:1500
a=bw:send pt=97 SMT:tb=64800:1500
a=bw:recv pt=* AMT:tb=644000:1500
a=max-recv-ssrc:* 5
a=max-send-ssrc:97 3
a=max-send-ssrc:96 1
a=ssrc:239245219 cname:alice@foo.example.com
a=ssrc:239245219 srcname:a3:d3:4b:f1:22:12
a=ssrc:986545121 cname:alice@foo.example.com
a=ssrc:986545121 srcname:12:3f:ab:d2:ec:32
a=ssrc:2199983234 cname:alice@foo.example.com
a=ssrc:2199983234 srcname:7f:12:db:87:2d:52
m=video 49300 RTP/AVP 96
b=AS:8240
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c01e
a=imageattr:* send [x=1280,y=720] recv [x=1280,y=720]
a=bw:send pt=96 SMT:tb=2500000:8192
a=bw:recv pt=96 SMT:tb=3000000:8192
a=bw:send pt=* AMT:tb=8000000:16384
a=max-recv-ssrc:* 5
a=max-send-ssrc:* 3
a=ssrc:245219239 cname:alice@foo.example.com
a=ssrc:245219239 srcname:a3:d3:4b:f1:22:12
a=ssrc:545121986 cname:alice@foo.example.com
a=ssrc:545121986 srcname:12:3f:ab:d2:ec:32
a=ssrc:199983234 cname:alice@foo.example.com
a=ssrc:199983234 srcname:7f:12:db:87:2d:52
a=content:main
```

Telepresence room Offer for a point to point session

Alice invites Bob into a session where Alice proposes one audio and one video RTP session, both with multiple streams. The audio session is proposing to use 3 mono streams of G.719 (pt=97) as being more prioritized than a single stereo G.719 (pt=96). It also states that it is willing to accept up to 5 simultaneous audio streams from Bob independent of payload type. The end-point also declares the SSRC it

intends to use with bindings to CNAME and SRCNAME, enabling Bob to bind together the audio and the video streams that come from the same part of the conference table.

The video session only configures H.264 payload format and states that it intends to send 1280x720 resolution and requests to receive the same. Alice also states that she will put the upper limit of the streams it sends to 2500 kbps with 8192 bytes bucket depth, while it will accept to receive individual streams that are up to 3000 kbps with 8192 bytes bucket depth. However, it also promises to limit the aggregate to no more than 8000 kbps and 16384 of bucket depth for the combination of all three streams it intends to send. Alice is willing to receive up to 5 streams of video simultaneous. Also here Alice informs Bob of the SSRC and their bindings to CNAME and SRCNAME.

Bob process this invite and constructs a SDP answer to be delivered to Alice. As Bob only has two cameras and microphones it will indicate this from its side. However, it is capable of receiving Alice 3 streams without any issues.

```
v=0
o=bob 2890847754 28908477889 IN IP4 198.51.100.21
s=Telepresence Bob Response
c=IN IP4 198.51.100.21
b=AS:8528
t=0 0
m=audio 49200 RTP/AVP 97 96
b=AS:288
a=rtpmap:96 G719/48000/2
a=rtpmap:97 G719/48000
a=bw:send pt=96 SMT:tb=128800:1500
a=bw:send pt=97 SMT:tb=64800:1500
a=bw:send pt=* AMT:tb=136000:1500
a=bw:recv pt=* AMT:tb=240000:1500
a=max-recv-ssrc:* 3
a=max-send-ssrc:97 2
a=max-send-ssrc:96 1
a=ssrc:52037639 cname:bob@foo.example.com
a=ssrc:52037639 srcname:37:ee:ca:38:01:3c
a=ssrc:820545843 cname:bob@foo.example.com
a=ssrc:820545843 srcname:20:85:17:48:75:a4
m=video 49300 RTP/AVP 96
b=AS:8240
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c01e
a=imageattr:* send [x=1280,y=720] recv [x=1280,y=720]
a=bw:recv pt=96 SMT:tb=2500000:8192
a=bw:send pt=96 SMT:tb=3000000:8192
a=bw:send pt=* AMT:tb=6000000:16384
a=bw:recv pt=* AMT:tb=8000000:16384
a=max-recv-ssrc:* 3
a=max-send-ssrc:* 2
a=ssrc:911548031 cname:bob@foo.example.com
a=ssrc:911548031 srcname:37:ee:ca:38:01:3c
a=ssrc:586599792 cname:bob@foo.example.com
a=ssrc:586599792 srcname:20:85:17:48:75:a4
a=content:main
```

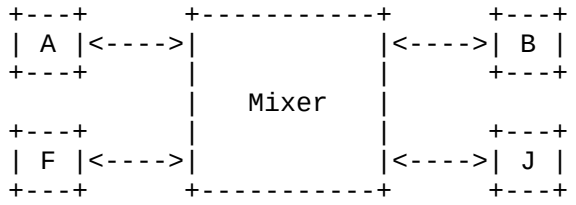
Telepresence room Answer for a point to point session

So Bob accepts the audio codec configurations but changes the aggregate bandwidths to what it is going to send itself and creates a limitation for Alice based on three mono streams. It confirms the number of streams Alice intends to be sending by including a=max-recv-ssrc:* 3. It also declares that it intends to send either two mono or one stereo stream. Bob also provides its configuration for SSRC and their mapping of CNAME and SRCNAME.

For video it is very similar, the number of streams Bob intends to send is stated as 2 and it also accept the 3 streams Alice intended to send in the max-recv-ssrc attribute. The bandwidth for these streams is accepted as suggested by Bob, keeping the upper limit for the individual streams at 3000 kbps and 8192 bytes depth. It also adds a total in Bob send direction that is twice the individual streams. It also confirms Alice's limitation for the aggregate. Finally the SSRCs for video are also declared and their bindings to CNAME and SRCNAME.

8.5.2. Simulcast Signaling

This example is for a case of client to video conference service using a centralized media topology with an RTP mixer. Alice, Bob calls into a conference server for a conference call with audio and video to the RTP mixer, these clients being capable to send a few video simulcast versions. The conference server also dials out to Fred, which is a legacy client resulting in fallback behavior. When dialing out to Joe more success is achieved as Joe is a client similar to Alice.



Four-party Mixer-based Conference

Example of Media plane for RTP mixer based multi-party conference with 4 participants.

8.5.2.1. Alice: Desktop Client

Alice is calling in to the mixer with an audiovisual single stream desktop client, only adding capability to send simulcast, announce SRCNAME and use of the new directional bandwidth attribute from Section 5.2 compared to a legacy client. The offer from Alice looks like

```
v=0
o=alice 2362969037 2362969040 IN IP4 203.0.113.156
s=Simulcast enabled Desktop Client
t=0 0
c=IN IP4 203.0.113.156
b=AS:825
a=group:SCS 2 3
m=audio 49200 RTP/AVP 96 97 9 8
b=AS:145
a=rtpmap:96 G719/48000/2
a=rtpmap:97 G719/48000
a=rtpmap:9 G722/8000
a=rtpmap:8 PCMA/8000
a=bw:send pt=96 SMT:tb=128800:1500
a=bw:send pt=97 SMT:tb=64800:1500
a=bw:send pt=8;9 SMT:tb=64000:1500
a=bw:recv pt=* AMT:tb=128800:1500
a=ssrc:521923924 cname:alice@foo.example.com
a=ssrc:521923924 srcname:a3:d3:4b:f1:22:12
a=mid:1
m=video 49300 RTP/AVP 96
b=AS:520
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c01e
a=imageattr:* send [x=640,y=360] recv [x=640,y=360] [x=320,y=180]
a=bw:send pt=96 SMT:tb=500000:8192
a=bw:recv pt=96 SMT:tb=500000:8192
a=ssrc:192392452 cname:alice@foo.example.com
a=ssrc:192392452 srcname:a3:d3:4b:f1:22:12
a=mid:2
a=content:main
m=video 49400 RTP/AVP 96
b=AS:160
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c00d
a=imageattr:96 send [x=320,y=180]
a=bw:send pt=96 SMT:tb=128000:4096
a=bw:recv pt=96 SMT:tb=128000:4096
a=ssrc:239245219 cname:alice@foo.example.com
a=ssrc:239245219 srcname:a3:d3:4b:f1:22:12
a=mid:3
a=sendonly
```

Alice Offer for a Simulcast Conference

As can be seen from the SDP, Alice has a simulcast-enabled client and offers two different session-multiplexed simulcast versions sent from her single camera, indicated by the SCS grouping tag and the two

media ID's (2 and 3). The first video version with media ID 2 prefers 360p resolution (signaled via imageattr) and the second video version with media ID 3 prefers 180p resolution. The first video media line also acts as the single receive video (making media line sendrecv), while the second video media line is only related to simulcast transmission and is thus offered sendonly. The two simulcast encoding streams and its related audio stream are bound together using SRCNAME SDES item. We also declare the end-point CNAME as all sources belong to the same synchronization context.

Alice uses the a=bw attribute defined in this document, but also uses the less exact, legacy b-line for interoperability. For video in this example, the client offers to send and receive a bandwidth lower than the video codec level maximum, which could for example have been set via some client or user preference, based on known transport limitations or knowledge what bandwidth is reasonable from a quality perspective given that specific codec at the proposed image resolution. The bitrates given in this example are supposed to be aligned with Section 5.2 and are thus based on the RTP payload level, but could also be designed based on another network layer according to the discussion in that section.

8.5.2.2. Bob: Telepresence Room

Bob is calling in to the mixer with a telepresence client that has capability for both sending multi-stream, receiving and local rendering of those multiple streams, as well as sending simulcast versions of the uplink video. More specifically, in this example the client has three cameras, each being sent in three different simulcast versions. In the receive direction, up to two main screens can show video from a (multi-stream) conference participant being active speaker, and still more screen estate can be used to show videos from up to 16 other conference listeners. Each camera has a corresponding (stereo) microphone that can also be negotiated down to mono by removing the stereo payload type from the answer.

```
v=0
o=bob 129384719 9834727 IN IP4 203.0.113.35
s=Simulcast enabled Multi stream Telepresence Client
t=0 0
c=IN IP4 203.0.113.35
b=AS:6035
a=group:SCS 2 3 4
m=audio 49200 RTP/AVP 96 97 9 8
b=AS:435
a=rtpmap:96 G719/48000/2
a=rtpmap:97 G719/48000
a=rtpmap:9 G722/8000
```

```
   a=rtpmap:8 PCMA/8000
   a=max-send-ssrc:* 3
   a=max-recv-ssrc:* 3
   a=bw:send pt=96 SMT:tb=128800:1500
   a=bw:send pt=97 SMT:tb=64800:1500
   a=bw:send pt=8;9 SMT:tb=64000:1500
   a=bw:send pt=* AMT:tb=386400:1500
   a=bw:recv pt=* AMT:tb=386400:1500
   a=ssrc:724847850 cname:bob@foo.example.com
   a=ssrc:724847850 srcname:37:ee:ca:38:01:3c
   a=ssrc:2847529901 cname:bob@foo.example.com
   a=ssrc:2847529901 srcname:20:85:17:48:75:a4
   a=ssrc:57289389 cname:bob@foo.example.com
   a=ssrc:57289389 srcname:1e:23:97:ab:9e:0c
   a=mid:1
   m=video 49300 RTP/AVP 96
   b=AS:4500
   a=rtpmap:96 H264/90000
   a=fmtp:96 profile-level-id=42c01f
   a=imageattr:* send [x=1280,y=720] recv [x=1280,y=720]
        [x=640,y=360] [x=320,y=180]
   a=max-send-ssrc:96 3
   a=max-recv-ssrc:96 2
   a=bw:send pt=96 SMT:tb=1500000:16384
   a=bw:send pt=* AMT:tb=4500000:16384
   a=bw:recv pt=96 SMT:tb=1500000:16384
   a=bw:recv pt=* AMT:tb=3000000:16384
   a=ssrc:75384768 cname:bob@foo.example.com
   a=ssrc:75384768 srcname:37:ee:ca:38:01:3c
   a=ssrc:2934825991 cname:bob@foo.example.com
   a=ssrc:2934825991 srcname:20:85:17:48:75:a4
   a=ssrc:3582594238 cname:bob@foo.example.com
   a=ssrc:3582594238 srcname:1e:23:97:ab:9e:0c
   a=mid:2
   a=content:main
   m=video 49400 RTP/AVP 96
   b=AS:1560
   a=rtpmap:96 H264/90000
   a=fmtp:96 profile-level-id=42c01e
   a=imageattr:* send [x=640,y=360]
   a=max-send-ssrc:96 3
   a=bw:send pt=96 SMT:tb=500000:8192
   a=ssrc:1371234978 cname:bob@foo.example.com
   a=ssrc:1371234978 srcname:37:ee:ca:38:01:3c
   a=ssrc:897234694 cname:bob@foo.example.com
   a=ssrc:897234694 srcname:20:85:17:48:75:a4
   a=ssrc:239263879 cname:bob@foo.example.com
   a=ssrc:239263879 srcname:1e:23:97:ab:9e:0c
```

```
a=mid:3
a=sendonly
m=video 49500 RTP/AVP 96
b=AS:420
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c00d
a=imageattr:96 send [x=320,y=180]
a=max-send-ssrc:96 3
a=bw:send pt=96 SMT:tb=128000:4096
a=ssrc:485723998 cname:bob@foo.example.com
a=ssrc:485723998 srcname:37:ee:ca:38:01:3c
a=ssrc:2345798212 cname:bob@foo.example.com
a=ssrc:2345798212 srcname:20:85:17:48:75:a4
a=ssrc:1295729848 cname:bob@foo.example.com
a=ssrc:1295729848 srcname:1e:23:97:ab:9e:0c
a=mid:4
a=sendonly
m=video 49600 RTP/AVP 96 97 98
b=AS:2600
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c01f
a=imageattr:96 recv [x=1280,y=720]
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=42c01e
a=imageattr:97 recv [x=640,y=360]
a=rtpmap:98 H264/90000
a=fmtp:98 profile-level-id=42c00d
a=imageattr:98 recv [x=320,y=180]
a=max-recv-ssrc:96 1
a=max-recv-ssrc:97 4
a=max-recv-ssrc:98 16
a=max-recv-ssrc:* 16
a=bw:recv pt=96 SMT:tb=1500000:16384
a=bw:recv pt=97 SMT:tb=500000:8192
a=bw:recv pt=98 SMT:tb=128000:4096
a=bw:recv pt=* AMT:tb=2500000:16384
a=mid:5
a=recvonly
a=content:alt
```

Bob Offer for a Multi-stream and Simulcast Telepresence Conference

Bob has a three-camera, three-screen, simulcast-enabled client with even higher performance than Alice’s and can additionally support 720p video, as well as multiple receive streams of various resolutions. The client implementor has thus decided to offer three simulcast streams for each camera, indicated by the SCS grouping tag and the three media ID’s (2, 3, and 4) in the SDP.

The first video media line with media ID 2 indicates the ability to send video from three simultaneous video sources (cameras) through the max-send-ssrc attribute with value 3. This media line is also marked as the main video by using the content attribute from [RFC4796]. Also the receive direction has declared ability to handle multiple video sources, and in this example it is 2. The interpretation of content:main for those two streams in the receive direction is that the client expects and can present (in prime position) at most two main (active speaker) video streams from another multi-camera client.

The second and third video media lines with media ID 3 and 4 are the sendonly simulcast streams. They can implicitly through the grouping be interpreted as also being content:main for the send direction, but is not marked as such since multiple media blocks with content:main could be confusing for a legacy client.

The fourth video media line with media ID 5 is recvonly and is marked with content:alt. That media line should, as was intended for that content attribute value, receive alternative content to the main speaker, such as "audience". In a multi-party conference, that could for example be the next-to-most-active speakers. The SDP describes that those streams can be presented in a set of different resolutions, indicated through the different payload types. The maximum number of streams per payload type is indicated through the max-recv-ssrc attribute. In this example, at most one stream can have payload type 96, preferably 720p, as indicated by the related imageattr line. Similarly, at most 4 streams can have payload type 97, preferably using 360p resolution, and at most 16 streams can have payload type 98, preferably of 180p resolution. In any case, there must never be more than 16 simultaneous streams of any payload type, but combinations of payload types may occur, such as for example two streams using payload type 97 and 8 streams using payload type 98.

To be able to relate the three cameras with the three microphones, all media lines that send audio or video use the ssrc attribute from [RFC5576], specifying the same SRCNAME from Section 5.3.2 for the audio and video versions that belong together. The use of this attribute is optional and the information can be retrieved from RTCP reporting, but it will then not be possible to correctly relate audio and video sources until the first RTCP report is received and participants may then seemingly make uncorrelated moves between screens and/or speakers when adjusting possible false correlation assumptions.

The legacy bandwidth reflects only the bandwidth in the receive direction, while the new bw attribute is very specific per direction and per media stream. We do note that the offered bandwidth for

transmission express as AS on session level woad be 6985. It is unclear what is the correct interpretation of the legacy bandwidth when there is bandwidth asymmetry.

The answer from a simulcast-enabled RTP mixer to this last SDP could look like:

```
v=0
o=server 238947290 239573929 IN IP4 198.51.100.2
s=Multi stream and Simulcast Telepresence Bob Answer
c=IN IP4 198.51.100.43
b=AS:7065
a=group:SCR 2 3 4
m=audio 49200 RTP/AVP 96
b=AS:435
a=rtpmap:96 G719/48000/2
a=max-send-ssrc:96 3
a=max-recv-ssrc:96 3
a=bw:send pt=96 SMT:tb=128800:1500
a=bw:recv pt=96 SMT:tb=128800:1500
a=bw:send pt=* AMT:tb=386400:1500
a=bw:recv pt=* AMT:tb=386400:1500
a=ssrc:4111848278 cname:server@conf1.example.com
a=ssrc:4111848278 srcname:87:e9:19:29:c1:bb
a=ssrc:835978294 cname:server@conf1.example.com
a=ssrc:835978294 srcname:1f:83:b3:85:62:7a
a=ssrc:2938491278 cname:server@conf1.example.com
a=ssrc:2938491278 srcname:99:76:b4:bb:90:52
a=mid:1
m=video 49300 RTP/AVP 96
b=AS:4650
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c01f
a=imageattr:* send [x=1280,y=720] [x=640,y=360] [x=320,y=180]
     recv [x=1280,y=720]
a=max-recv-ssrc:96 3
a=max-send-ssrc:96 2
a=bw:recv pt=96 SMT:tb=1500000:16384
a=bw:recv pt=* AMT:tb=4500000:16384
a=bw:send pt=96 SMT:tb=1500000:16384
a=bw:send pt=* AMT:tb=3000000:16384
a=ssrc:2938746293 cname:server@conf1.example.com
a=ssrc:2938746293 srcname:87:e9:19:29:c1:bb
a=ssrc:1207102398 cname:server@conf1.example.com
a=ssrc:1207102398 srcname:1f:83:b3:85:62:7a
a=mid:2
a=content:main
m=video 49400 RTP/AVP 96
```

```
   b=AS:1560
   a=rtpmap:96 H264/90000
   a=fmtp:96 profile-level-id=42c01e
   a=imageattr:* recv [x=640,y=360]
   a=max-recv-ssrc:96 3
   a=bw:recv pt=96 SMT:tb=500000:8192
   a=mid:3
   a=recvonly
   m=video 49500 RTP/AVP 96
   b=AS:420
   a=rtpmap:96 H264/90000
   a=fmtp:96 profile-level-id=42c00d
   a=imageattr:96 recv [x=320,y=180]
   a=max-recv-ssrc:96 3
   a=bw:recv pt=96 SMT:tb=128000:4096
   a=mid:4
   a=recvonly
   m=video 49600 RTP/AVP 96 97 98
   b=AS:2600
   a=rtpmap:96 H264/90000
   a=fmtp:96 profile-level-id=42c01f
   a=imageattr:96 send [x=1280,y=720]
   a=rtpmap:97 H264/90000
   a=fmtp:97 profile-level-id=42c01e
   a=imageattr:97 send [x=640,y=360]
   a=rtpmap:98 H264/90000
   a=fmtp:98 profile-level-id=42c00d
   a=imageattr:98 send [x=320,y=180]
   a=max-send-ssrc:96 1
   a=max-send-ssrc:97 4
   a=max-send-ssrc:98 8
   a=max-send-ssrc:* 8
   a=bw:send pt=96 SMT:tb=1500000:16384
   a=bw:send pt=97 SMT:tb=500000:8192
   a=bw:send pt=98 SMT:tb=128000:4096
   a=bw:send pt=* AMT:tb=2500000:16384
   a=ssrc:2981523948 cname:server@conf1.example.com
   a=ssrc:2938237 cname:server@conf1.example.com
   a=ssrc:1230495879 cname:server@conf1.example.com
   a=ssrc:74835983 cname:server@conf1.example.com
   a=ssrc:3928594835 cname:server@conf1.example.com
   a=ssrc:948753 cname:server@conf1.example.com
   a=ssrc:1293456934 cname:server@conf1.example.com
   a=ssrc:4134923746 cname:server@conf1.example.com
   a=mid:5
   a=sendonly
   a=content:alt
```

Server Answer for Bob Multi-stream and Simulcast Telepresence Conference

In this SDP answer, the grouping tag is changed to SCR, confirming that the sent simulcast streams will be received. The directionality of the streams themselves as well as the directionality of multi-stream and bandwidth attributes are changed. Note that the session level legacy bandwidth can be calculated more correctly with support from the bw attribute in the offer than would have been the case if only legacy media level bandwidth was present. Bandwidth bucket size can be adjusted down between the offer and the answer for streams sent from the answerer, indicating a more strict constant bitrate than really needed. The bucket size can be adjusted up or down for streams received by the answerer, indicating a more strict or flexible bitrate constraint, respectively, for the receiver compared to what the sender offered. The number of allowed streams in the content:alt video session has been reduced to 8 in the answer from 16 offered.

Note that the two video sources in the media block with mid:2 correspond to the two first audio sources (matching SRCNAME). The last audio source correspond to all video sources in the media block with mid:5, however SRCNAME can not be used to perform this binding as its semantic doesn’t match.

8.5.2.3. Fred: Dial-out to Legacy Client

Fred has a simple legacy client that know nothing of the new signaling means discussed in this document. In this example, the multi-stream and simulcast aware RTP mixer is calling out to Fred. Even though it is never actually sent, this would be Fred’s offer SDP, should he have called in. It is included here to improve the reader’s understanding of Fred’s response to the conference SDP.

```
v=0
o=fred 82342187 237429834 IN IP4 192.0.2.213
s=Legacy Client
t=0 0
c=IN IP4 192.0.2.213
m=audio 50132 RTP/AVP 9 8
a=rtpmap:9 G722/8000
a=rtpmap:8 PCMA/8000
m=video 50134 RTP/AVP 96 97
b=AS:405
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c00c
a=rtpmap:97 H263-2000/90000
a=fmtp:97 profile=0;level=30
```

Legacy Client Hypothetical Offer

Fred would offer a single mono audio and a single video, each with a couple of different codec alternatives.

The same conference server as in the previous example is calling out to Fred, offering the full set of multi-stream and simulcast features, with maximum stream and bandwidth limits based on what the server itself can support.

```
v=0
o=server 323439283 2384192332 IN IP4 198.51.100.2
s=Multi stream and Simulcast Dial-out Offer
c=IN IP4 198.51.100.43
b=AS:7065
a=group:SCR 2 3 4
m=audio 49200 RTP/AVP 96 97 9 8
b=AS:435
a=rtpmap:96 G719/48000/2
a=rtpmap:97 G719/48000
a=rtpmap:9 G722/8000
a=rtpmap:8 PCMA/8000
a=max-send-ssrc:* 4
a=max-recv-ssrc:* 3
a=bw:send pt=96 SMT:tb=128800:1500
a=bw:send pt=97 SMT:tb=64800:1500
a=bw:send pt=8;9 SMT:tb=64000:1500
a=bw:send pt=* AMT:tb=515200:1500
a=bw:recv pt=* AMT:tb=386400:1500
a=ssrc:3293472833 cname:server@conf1.example.com
a=ssrc:3293472833 srcname:28:23:54:39:7a:0e
a=ssrc:1734728348 cname:server@conf1.example.com
a=ssrc:1734728348 srcname:83:88:be:19:a6:15
a=ssrc:1054453769 cname:server@conf1.example.com
a=ssrc:1054453769 srcname:76:91:cc:23:02:68
a=ssrc:3923447729 cname:server@conf1.example.com
a=ssrc:3923447729 srcname:be:73:a6:03:00:82
a=mid:1
m=video 49300 RTP/AVP 96
b=AS:4650
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c01f
a=imageattr:* send [x=1280,y=720] [x=640,y=360] [x=320,y=180]
    recv [x=1280,y=720]
a=max-recv-ssrc:96 3
a=max-send-ssrc:96 3
a=bw:recv pt=96 SMT:tb=1500000:16384
a=bw:recv pt=* AMT:tb=4500000:16384
```

```
   a=bw:send pt=96 SMT:tb=1500000:16384
   a=bw:send pt=* AMT:tb=4500000:16384
   a=ssrc:78456398 cname:server@conf1.example.com
   a=ssrc:78456398 srcname:28:23:54:39:7a:0e
   a=ssrc:3284726348 cname:server@conf1.example.com
   a=ssrc:3284726348 srcname:83:88:be:19:a6:15
   a=ssrc:2394871293 cname:server@conf1.example.com
   a=ssrc:2394871293 srcname:76:91:cc:23:02:68
   a=mid:2
   a=content:main
   m=video 49400 RTP/AVP 96
   b=AS:1560
   a=rtpmap:96 H264/90000
   a=fmtp:96 profile-level-id=42c01e
   a=imageattr:* recv [x=640,y=360]
   a=max-recv-ssrc:96 3
   a=bw:recv pt=96 SMT:tb=500000:8192
   a=mid:3
   a=recvonly
   m=video 49500 RTP/AVP 96
   b=AS:420
   a=rtpmap:96 H264/90000
   a=fmtp:96 profile-level-id=42c00d
   a=imageattr:96 recv [x=320,y=180]
   a=max-recv-ssrc:96 3
   a=bw:recv pt=96 SMT:tb=128000:4096
   a=mid:4
   a=recvonly
   m=video 49600 RTP/AVP 96 97 98
   b=AS:2600
   a=rtpmap:96 H264/90000
   a=fmtp:96 profile-level-id=42c01f
   a=imageattr:96 send [x=1280,y=720]
   a=rtpmap:97 H264/90000
   a=fmtp:97 profile-level-id=42c01e
   a=imageattr:97 send [x=640,y=360]
   a=rtpmap:98 H264/90000
   a=fmtp:98 profile-level-id=42c00d
   a=imageattr:98 send [x=320,y=180]
   a=max-send-ssrc:96 1
   a=max-send-ssrc:97 4
   a=max-send-ssrc:98 8
   a=max-send-ssrc:* 8
   a=bw:send pt=96 SMT:tb=1500000:16384
   a=bw:send pt=97 SMT:tb=500000:8192
   a=bw:send pt=98 SMT:tb=128000:4096
   a=bw:send pt=* AMT:tb=2500000:16384
   a=ssrc:2342872394 cname:server@conf1.example.com
```

```
a=ssrc:1283741823 cname:server@conf1.example.com
a=ssrc:3294823947 cname:server@conf1.example.com
a=ssrc:1020408838 cname:server@conf1.example.com
a=ssrc:1999343791 cname:server@conf1.example.com
a=ssrc:2934192349 cname:server@conf1.example.com
a=ssrc:2234347728 cname:server@conf1.example.com
a=ssrc:3224283479 cname:server@conf1.example.com
a=mid:5
a=sendonly
a=content:alt
```

Server Dial-out Offer with Multi-stream and Simulcast

The answer from Fred to this offer would look like:

```
v=0
o=fred 9842793823 239482793 IN IP4 192.0.2.213
s=Legacy Client Answer to Server Dial-out
t=0 0
c=IN IP4 192.0.2.213
m=audio 50132 RTP/AVP 9
b=AS:80
a=rtpmap:9 G722/8000
m=video 50134 RTP/AVP 96
b=AS:405
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c00c
m=video 0 RTP/AVP 96
m=video 0 RTP/AVP 96
m=video 0 RTP/AVP 96
```

Legacy Client Answer to Server Dial-out

as can be seen from the hypothetical offer, Fred does not understand any of the multistream or simulcast attributes, and does also not understand the grouping framework. Thus, all those lines are removed from the answer SDP and any surplus video media blocks except for the first are rejected. The media bandwidth are adjusted down to what Fred actually accepts to receive.

8.5.2.4. Joe: Dial-out to Desktop Client

This example is almost identical to the one above, with the difference that the answering end-point has some limited simulcast and multi-stream capability. As above this is the offer SDP that Joe would have used, should he have called in.

```
v=0
```

```
   o=joe 82342187 237429834 IN IP4 192.0.2.213
   s=Simulcast and Multistream enabled Desktop Client
   t=0 0
   c=IN IP4 192.0.2.213
   b=AS:985
   a=group:SCS 2 3
   m=audio 49200 RTP/AVP 96 97 9 8
   b=AS:145
   a=rtpmap:96 G719/48000/2
   a=rtpmap:97 G719/48000
   a=rtpmap:9 G722/8000
   a=rtpmap:8 PCMA/8000
   a=bw:send pt=96 SMT:tb=128800:1500
   a=bw:send pt=97 SMT:tb=64800:1500
   a=bw:send pt=8;9 SMT:tb=64000:1500
   a=bw:recv pt=* AMT:tb=128800:1500
   a=ssrc:1223883729 cname:joe@foo.example.com
   a=ssrc:1223883729 srcname:12:88:07:cf:81:65
   a=mid:1
   m=video 49300 RTP/AVP 96
   b=AS:520
   a=rtpmap:96 H264/90000
   a=fmtp:96 profile-level-id=42c01e
   a=imageattr:96 send [x=640,y=360] recv [x=640,y=360] [x=320,y=180]
   a=bw:send pt=96 SMT:tb=500000:8192
   a=bw:recv pt=96 SMT:tb=500000:8192
   a=ssrc:3842394823 cname:joe@foo.example.com
   a=ssrc:3842394823 srcname:12:88:07:cf:81:65
   a=mid:2
   a=content:main
   m=video 49400 RTP/AVP 96
   b=AS:160
   a=rtpmap:96 H264/90000
   a=fmtp:96 profile-level-id=42c00d
   a=imageattr:96 send [x=320,y=180]
   a=bw:send pt=96 SMT:tb=128000:4096
   a=bw:recv pt=96 SMT:tb=128000:4096
   a=ssrc:1214232284 cname:joe@foo.example.com
   a=ssrc:1214232284 srcname:12:88:07:cf:81:65
   a=mid:3
   a=sendonly
   m=video 49300 RTP/AVP 96
   b=AS:320
   a=rtpmap:96 H264/90000
   a=fmtp:96 profile-level-id=42c00c
   a=imageattr:96 recv [x=320,y=180]
   a=max-recv-ssrc:* 2
   a=bw:recv pt=96 SMT:tb=128000:4096
```

```
a=bw:recv pt=96 AMT:tb=256000:4096
a=mid:4
a=recvonly
a=content:alt
```

Desktop Client Hypothetical Offer

Joe would send two versions of simulcast, 360p and 180p, from a single camera and can receive three sources of multi-stream, one 360p and two 180p streams.

Again, the same conference server is calling out to Joe and the offer SDP from the server would be almost identical to the one in the previous example. It is therefore not included here. The response from Joe would look like:

```
v=0
o=joe 239482639 4702341992 IN IP4 192.0.2.213
s=Answer from Desktop Client to Server Dial-out
t=0 0
c=IN IP4 192.0.2.213
b=AS:985
a=group:SCS 2 3
m=audio 49200 RTP/AVP 96
b=AS:145
a=rtpmap:96 G719/48000/2
a=bw:send pt=96 SMT:tb=128800:1500
a=bw:recv pt=* AMT:tb=128800:1500
a=ssrc:1223883729 cname:joe@foo.example.com
a=ssrc:1223883729 srcname:12:88:07:cf:81:65
a=mid:1
m=video 49300 RTP/AVP 96
b=AS:520
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c01e
a=imageattr:96 send [x=640,y=360] recv [x=640,y=360] [x=320,y=180]
a=bw:send pt=96 SMT:tb=500000:8192
a=bw:recv pt=96 SMT:tb=500000:8192
a=ssrc:3842394823 cname:joe@foo.example.com
a=ssrc:3842394823 srcname:12:88:07:cf:81:65
a=mid:2
a=content:main
m=video 0 RTP/AVP 96
a=mid:3
m=video 49400 RTP/AVP 96
b=AS:160
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c00d
```

```
a=imageattr:96 send [x=320,y=180]
a=bw:send pt=96 SMT:tb=128000:4096
a=bw:recv pt=96 SMT:tb=128000:4096
a=ssrc:1214232284 cname:joe@foo.example.com
a=ssrc:1214232284 srcname:12:88:07:cf:81:65
a=mid:4
a=sendonly
m=video 49300 RTP/AVP 96
b=AS:320
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42c00c
a=imageattr:96 recv [x=320,y=180]
a=max-recv-ssrc:* 2
a=bw:recv pt=96 SMT:tb=128000:4096
a=bw:recv pt=96 AMT:tb=256000:4096
a=mid:5
a=recvonly
a=content:alt
```

Desktop Client Answer to Server Dial-out

Since the RTP mixer support all of the features that Joe does and more, the SDP does not differ much from what it should have been in an offer. It can be noted that as stated in [RFC5888], all media lines need mid attributes, even the rejected ones, which is why mid:3 is present even though the mid quality simulcast version is rejected by Joe.

9. IANA Considerations

Following the guidelines in [RFC4566], in [RFC5888], and in [RFC3550], the IANA is requested to register:

1. The SID grouping tag to be used with the grouping framework, as defined in Section 8.1.1

2. A new SDES Item named SRCNAME, as defined in Section 5.3.1

3. The max-send-ssrc and max-recv-ssrc SDP attributes as defined in Section 5.1

4. The bw attribute as defined in Section 5.2

5. The bw attribute scope registry rules

6. The bw attribute semantics registry rules

10. Security Considerations

There is minimal difference in security between the simulcast solutions. Session multiplexing may have some additional overhead in the key-management, but that is minor as most key management schemes can be performed in parallel.

The multi-stream signalling has as other SDP based signalling issues with man in the middles that may modify the SDP as an attack on either the service in general or a particular end-point. This can as usual be resolved by a security mechanism that provides integrity and source authentication between the signalling peers.

The SDES SRCNAME being opaque identifiers could potentially carry additional meanings or function as overt channel. If the SRCNAME would be permanent between sessions, they have the potential for compromising the users privacy as they can be tracked between sessions. See RFC6222 for more discussion.

11. Acknowledgements

12. References

12.1. 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.

[RFC6222] Begen, A., Perkins, C., and D. Wing, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)", RFC 6222, April 2011.

12.2. Informative References

[RFC2205] Braden, B., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black,

"Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

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

[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.

[RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.

[RFC4796] Hautakorpi, J. and G. Camarillo, "The Session Description Protocol (SDP) Content Attribute", RFC 4796, February 2007.

[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.

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

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

[RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.

[RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.

[RFC6236] Johansson, I. and K. Jung, "Negotiation of Generic Image Attributes in the Session Description Protocol (SDP)", RFC 6236, May 2011.

Authors' Addresses

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

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

Bo Burman
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

Phone: +46 10 7141311
Email: bo.burman@ericsson.com