Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 17, 2012

M. Westerlund
B. Burman
Ericsson

May 16, 2012

Codec Control for WebRTC
draft-westerlund-rtcweb-codec-control-00

Abstract

This document proposes how WebRTC should handle media codec control between peers. With media codec control we mean such parameters as video resolution and frame-rate. This includes both initial establishment of capabilities using the SDP based JSEP signalling and during ongoing real-time interactive sessions in response to user and application events. The solution uses SDP for initial boundary establishment that are rarely, if ever changed. During the session the RTCP based Codec Operations Point (COP) signaling solution is used for dynamic control of parameters enabling timely and responsive controls.

Status of this Memo

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

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

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

This Internet-Draft will expire on November 17, 2012.

Table of Contents

1. Introduction

In WebRTC there exist need for codec control to improve the efficiency and user experience of its real-time interactive media transported over a PeerConnection. The fundamentals of the codec control is that the media receiver provides preference for how it would like the media to be encoded to best suit the receiver's consumption of the media stream. This includes parameters such as video resolution and frame-rate, and for audio number of channels and audio bandwidth. It also includes non media specific properties such as how to provision available transmission bit-rates between different media streams.

This document proposes a specific solution for how to accomplish codec control that meets the goals and requirements. It is based on establishing the outer boundaries, when it comes to codec support and capabilities, at PeerConnection establishment using JSEP [I-D.ietf-rtcweb-jsep] and SDP [RFC4566]. During an ongoing session the preferred parameters are signalled using the Codec Operation Point RTCP Extension (COP) [I-D.westerlund-avtext-codec-operation-point]. The java script Application will primarily make its preferences made clear through its usage of the media elements, like selecting the size of the rendering area for video. But it can also use the constraints concept in the API to indicate preferences that the browser can weigh into its decision to request particular preferred parameters.

This document provides a more detailed overview of the solution. Then it discusses the use cases and requirements that motivates the solution, followed by an analysis of the benefits and downsides of the proposed solution. This is followed by a proposed specification of how WebRTC should use SDP and COP.

2. Definitions

2.1. Abrevations

The following Abbreviations are used in this document.

COP: Codec Operation Point RTCP Extension, the solution for codec control defined in [I-D.westerlund-avtext-codec-operation-point].

JSEP: Java script Session Establishment Protocol [I-D.ietf-rtcweb-jsep].

RTP: Real-time Transport Protocol [RFC3550].

SDP: Session Description Protocol [RFC4566].

2.2. Requirements Language

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

3. Overview

The basic idea in this proposal is to use JSEP to establish the outer limits for behavior and then use Codec Operation Point (COP) [I-D.westerlund-avtext-codec-operation-point] proposal to handle dynamic changes during the session.

Boundary conditions are typically media type specific and in some cases also codec specific. Relevant for video are highest resolution, frame-rate and maximum complexity. These can be expressed in JSEP SDP for H.264 using the H.264 RTP payload format [RFC6184] specifying the profile and level concept. The authors expect something similar for the VP8 payload format [I-D.ietf-payload-vp8].

During the session the browser implementation detects when there is need to use COP to do any of the following things.

a. Request new target values for codec operation, for example based on that the GUI element displaying a video has changed due to window resize or purpose change. This includes parameters such as resolution, frame-rate, and picture aspect ratio.

b. Change parameters due to changing display screen attached to the device. Affected parameters include resolution, picture aspect ratio and sample aspect ratio.

c. Indicate when the end-point changes encoding parameters in its role as sender.

d. Change important parameters affecting the transport for media streams such as a maximum media bit-rate, token bucket size (to control the burstiness of the sender), used RTP payload type, maximum RTP packet size, application data unit Aggregation (to control amount of audio frames in the same RTP packet).

e. Affect the relative prioritization of media streams.

The receiving client may send a COP request in RTCP to request some set of parameters to be changed according to the receiving client's preferences. The applications preferences are primarily indicated through its usage of the media elements. But there exist cases and properties where the application will have to provide additional preference information for example using the constraints. The browser implementation takes all these information into account when expressing preference using a set of parameters.

The media sender evaluates the request and weights it against other potential receiver's requests and may update one or more (if scalability is supported) codec operation points to better suit the receivers. Any new operation point(s) are announced using a COP Notification. Independently if the codec operation point(s) are changed or not, the COP request is acknowledged using a COP status message.

Using RTCP and "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)" [RFC5124] the COP message can in most cases be sent immediately or with a very small delay. As the message travels in the media plane it will reach the peer or the next middlebox that are part of the media path directly.

4. Requirements and Motivations

This section discusses the use cases and the requirements for codec control. This includes both the ones explicitly discussed in the use case document but also derived ones. This is followed by a discussion why the proposed mechanism is considered the most suitable for WebRTC.

4.1. Use Cases and Requirements

There are use cases and derived requirements in "Web Real-Time Communication Use-cases and Requirements" [I-D.ietf-rtcweb-use-cases-and-requirements].

A Selection of interesting Use Cases and the description parts that are most applicable to Codec Control are:

4.2.1 - Simple Video Communication Service: Two or more users have loaded a video communication web application into their browsers, provided by the same service provider, and logged into the service it provides. The web service publishes information about user login status by pushing updates to the web application in the

browsers. When one online user selects a peer online user, a 1-1 video communication session between the browsers of the two peers is initiated. The invited user might accept or reject the session.

During session establishment a self-view is displayed, and once the session has been established the video sent from the remote peer is displayed in addition to the self-view. During the session, each user can select to remove and re-insert the self-view as often as desired. Each user can also change the sizes of his/her two video displays during the session. Each user can also pause sending of media (audio, video, or both) and mute incoming media

The two users may be using communication devices of different makes, with different operating systems and browsers from different vendors.

4.2.10 - Multiparty video communication: In this use-case is the Simple Video Communication Service use-case (Section 4.2.1) is extended by allowing multiparty sessions. No central server is involved - the browser of each participant sends and receives streams to and from all other session participants. The web application in the browser of each user is responsible for setting up streams to all receivers.

In order to enhance intelligibility, the web application pans the audio from different participants differently when rendering the audio. This is done automatically, but users can change how the different participants are placed in the (virtual) room. In addition the levels in the audio signals are adjusted before mixing.

Another feature intended to enhance the use experience is that the video window that displays the video of the currently speaking peer is highlighted.

Each video stream received is by default displayed in a thumbnail frame within the browser, but users can change the display size.

Note: What this use-case adds in terms of requirements is capabilities to send streams to and receive streams from several peers concurrently, as well as the capabilities to render the video from all received streams and be able to spatialize, level adjust and mix the audio from all received streams locally in the browser. It also adds the capability to measure the audio level/ activity.

4.3.3 - Video conferencing system with central server: An organization uses a video communication system that supports the establishment of multiparty video sessions using a central conference server.

The browser of each participant send an audio stream (type in terms of mono, stereo, 5.1, ... depending on the equipment of the participant) to the central server. The central server mixes the audio streams (and can in the mixing process naturally add effects such as spatialization) and sends towards each participant a mixed audio stream which is played to the user.

The browser of each participant sends video towards the server. For each participant one high resolution video is displayed in a large window, while a number of low resolution videos are displayed in smaller windows. The server selects what video streams to be forwarded as main- and thumbnail videos respectively, based on speech activity. As the video streams to display can change quite frequently (as the conversation flows) it is important that the delay from when a video stream is selected for display until the video can be displayed is short.

Note: This use-case adds requirements on support for fast stream switches F7. There exist several solutions that enable the server to forward one high resolution and several low resolution video streams: a) each browser could send a high resolution, but scalable stream, and the server could send just the base layer for the low resolution streams, b) each browser could in a simulcast fashion send one high resolution and one low resolution stream, and the server just selects or c) each browser sends just a high resolution stream, the server transcode into low resolution streams as required.

The derived requirements that applies to codec control are:

F3: Transmitted streams MUST be rate controlled.

F6: The browser MUST be able to handle high loss and jitter levels in a graceful way.

F7: The browser MUST support fast stream switches.

F24: The browser MUST be able to take advantage of capabilities to prioritize voice and video appropriately.

F25: The browser SHOULD use encoding of streams suitable for the current rendering (e.g. video display size) and SHOULD change parameters if the rendering changes during the session.

It might not be obvious how some of the above requirements actually have impact on the question of controlling the media encoder in a transmitter so let's go through what the document authors consider be its applicability. But let's start with reviewing the topologies that exist.

Peer to Peer: This is the basic topology used in use case "Simple Video Communication Service". Two end-points communicating directly with each other. A PeerConnection directly connects the source and the sink of the media stream. Thus in this case it is simple and straightforward to feed preferences from the sink into the source's media encoder to produce the best possible match that the source is capable of, given the preferences.

Peer to Multiple Peers: A given source have multiple PeerConnections going from the source to a number of receivers, i.e. sinks as described by use case "Multiparty video communication". In some implementations this will be implemented as Peer to Peer topology where only the source for the raw media is common between the different PeerConnections. On more resource constrained devices that can't afford individual media encodings for each PeerConnection the media stream is to be delivered over, there exist a need to merge the different preferences from the different receivers into a single or a set of fewer configurations that can be produced. For codecs that has scalability features, it might be possible to produce multiple actual operation points in a single encoding and media stream. For example multiple frame rates can be produced by H.264 by encoding using a frame structure where some frames can be removed to produce a lower bit-rate and lower frame rate version of the stream. Thus possibly allowing multiple concurrent operation points to be produced to meet the demands for an even larger number of preferred operation points.

Centralized Conferencing: This topology consists of a central server to which each conference participant connects his PeerConnection(s). Over that PeerConnection the participant will receive all the media streams the conference service thinks should be sent and delivered. The actual central node can work in several different modes for media streams. It can be a very simple relay node (RTP transport translator [RFC5117]), where it forwards all media streams arriving to it to the other participants, forming a common RTP session among all participants with full visibility. Another mode of operation would be an RTP mixer that forwards selected media streams using a set of SSRC the

RTP mixer has. The third mode is to perform actual media mixing such as where audio is mixed and video is composited into a new video image and encoded again.

This results in two different behaviors in who needs to merge multiple expressed preferences. For a simple relay central node, the merge of preferences may be placed on the end-point, similar to the resource constrained peer to multiple peer case above. The second alternative is to let the central node merge the preferences into a single set of preferences, which is then signalled to the media source end-point.

Note: In the above it might be possible to establish multiple PeerConnections between an end-point and the central node. The different PeerConnections would then be used to express different preferences for a given media stream. This enables simulcast delivery to the central node so that it can use more than a single operation point to meet the preferences expressed by the multiple receiving participants. That approach can improve the media quality for end-points capable of receiving and using a higher media quality, since they can avoid being constrained by the lowest common denominator of a single operation point.

Peer Relayed: This is not based on an explicit use case in the use case document. It is based on a usage that appears possible to support, and for which there has been interest. The topology is that Peer A sources a media stream and sends it over a PeerConnection to B. B in its turn has a PeerConnection to Peer C. B chooses to relay the incoming media stream from A to C. To maintain quality, it is important that B does not decode and re-encode the media stream (transcoding). Thus a case arises where B will have to merge the preferences from itself and C into the preferences it signals to A.

Comments on the applicability of the requirement on the codec control:

F3: This requirement requires rate-control on the media streams. There will also be multiple media streams being sent to or from a given end-point. Combined, this creates a potential issue when it comes to prioritization between the different media streams and what policy to use to increase and decrease the bit-rate provided for each media stream. The application's preferences combined with other properties such as current resolution and frame-rate affects which parameter that is optimal to use when bit-rate needs to be changed. The other aspect is if one media stream is less relevant so that reducing that stream's quality or even terminating the transmission while keeping others unchanged is the

best choice for the application. In other cases, applying the cheese cutter principle and reduce all streams in equal proportion is the most desirable. Another aspect is the potential for requesting aggregation of multiple audio frames in the same RTP packet to reduce the overhead and thus lower the bit-rate for some increased delay and packet loss sensitivity.

F6: The browser MUST be able to handle high loss and jitter levels in a graceful way. When such conditions are encountered, it will be highly beneficial for the receiver to be able to indicate that the sender should try to combat this by changing the encoding and media packetization. For example for audio it might be beneficial to aggregate several frames together and apply additional levels of FEC on those fewer packets that are produced to reduce the residual audio frame loss.

F7: The browser MUST support fast stream switches. Fast stream switches occur in several ways in WebRTC. One is in the centralized conferencing when relay based central nodes turn on and off individual media streams depending on the application's current needs. Another is RTP mixers that switches input sources for a given outgoing SSRC owned by the mixer. This should have minimal impact on a receiver as there is no SSRC change. Along the same lines, the application can cause media stream changes by changing their usage in the application. By changing the usage of a media stream from being the main video to become a thumbnail of one participant in the session, there exist a need to rapidly switch the video resolution to enable high efficiency and avoid increased bit-rate usage.

F24: The browser MUST be able to take advantage of capabilities to prioritize voice and video appropriately. This requirement comes from the QoS discussion present in use case 4.2.6 (Simple Video Communication Service, QoS). This requirement assumes that the application has a preference for one media type over another. Given this assumption, the same prioritization can actually occur for a number of codec parameters when there exist multiple media streams and one can individually control these media streams. This is another aspect of the discussion for requirement F3.

F25: The browser SHOULD use encoding of streams suitable for the current rendering (e.g. video display size) and SHOULD change parameters if the rendering changes during the session. This requirement comes from a very central case that a receiving application changes the display layout and where it places a given media stream. Thus changing the amount of screen estate that the media stream is viewed on also changes what resolution that would be the optimal to use from the media sender. However, this

discussion should not only apply to video resolution. Additional application preferences should result in preferences being expressed to the media sender also for other properties, such as video frame-rate. For audio, number of audio channels and the audio bandwidth are relevant properties.

The authors hope this section has provided a sufficiently clear picture that there exist both multiple topologies with different behaviors, and different points where preferences might need to be merged. The discussion of the requirements also provides a view that there are multiple parameters that needs to be expressed, not only video resolution.

4.2. Motivations

This section discusses different types of motivations for this solution. It includes comparison to the solution described in "RTCWEB Resolution Negotiation" [I-D.alvestrand-rtcweb-resolution].

4.2.1. Performance

The proposed solution has the following performance characteristics. The initial phase, establishing the boundaries, is done in parallel with the media codec negotiation and establishment of the PeerConnection. Thus using the signalling plane is optimal as this process does not create additional delay or significant overhead.

During an ongoing communication session, using COP messages in RTCP has the following properties:

Path Delay: The COP messages are contained in the RTCP packets being sent over the PeerConnection, i.e. the most optimal peer to peer path that ICE has managed to get to work. Thus one can expect this path to be equal or shorter in delay than the signalling path being used between the PeerConnection end-points. If the signalling message is instead sent over the PeerConnection's data channel, it will be using the same path. In almost all cases, the direct path between two peers will also be shorter than a path going via the webserver.

Media Plane: The COP messages will always go to the next potential RTP/RTCP processing point, i.e. the one on the other side of the PeerConnection. Even for multiparty sessions using centralized servers, the COP message may need to be processed in the middle to perform the merger of the different participant's preferences.

Overhead: An RTCP COP message can be sent as reduced size RTCP message [RFC5506] thus having minimal unnecessary baggage. For example a COP Request message requesting a new target resolution from a single SSRC will be 29 bytes. Using reduced size RTCP keeps the average RTCP size down and enables rapid recovery of the early allowed flag in early mode and in more cases enable the immediate mode.

Minimal Blocking: Using RTCP lets the transmission of COP messages be governed by RTCP's transmission rules. As WebRTC will be using the SAVPF profile it is possible to use the early mode, allowing an early transmission of an RTCP packet carrying a feedback event, like a COP request, to be sent with little delay. It might even be possible to determine that the immediate mode of operation can be enabled, thus allowing the RTCP feedback events to be sent immediate in all cases while using the mode. The small overhead and usage of reduced size RTCP will help ensure that the times when transmission of a COP message will be blocked is a rare event and will quickly be released.

The next aspect of RTCP blocking is that we expect that the application will need to rapidly switch back and forth between codec parameters. Thus requiring both a protocol that allows quick setting of parameters and also the possibility to revert back to previous preferences while the request is outstanding. COP has support for such updated requests, even if the first request is in flight.

If the above is compared to the properties that Harald Alvestrand's proposal [I-D.alvestrand-rtcweb-resolution] has, the following differences are present. When it comes to signalling path delay, a signalling plane based solution will in almost all cases at best have the same path delay as a media plane solution, achieved by using the data channel to carry the signalling. There the only difference will be the message size, which will only incur a minor difference in transfer times. But in cases where the application has not implemented use of the data channel, the signalling path will be worse, possibly significantly.

Using the signalling plane for solutions based on centralized conference mixers can easily result in that the request message needs to be processed in the webserver before being forwarded to the mixer node that actually processes the media, followed by the central mixer triggering additional signalling messages to other end-points that also needs to react. This can be avoided assuming that the data channel is used for signalling transport. Using the media plane for such signalling will be equal or better in almost all cases.

When it comes to blocking, there exist a significant issue with using JSEP to carry this type of messages. Someone that has sent an SDP offer in an offer/answer exchange is blocked from sending a new update until it has received a final or provisional answer to that offer. Here COP has a great advantage as the design has taken rapid change of parameters into consideration and allows multiple outstanding requests.

4.2.2. Ease of Use

We see a great benefit in that COP can be allowed to be mainly driven by the browser implementation and its knowledge of how media elements are currently being used by the application. For example the video resolution of the display area can be determined by the browser, further determining that the resource consumption would be reduced and the image quality improved or at least maintained by requesting another target resolution better suiting the current size. There are also other metrics or controls that exist in the browser space, like the congestion control that can directly use the COP signalling to request more suitable parameters given the situation.

Certain application preferences can't be determined based solely on the usage. Thus using the constraints mechanism to indicate preferences is a very suitable solution for most such properties. For example the relative priority of media streams, or a desire for lower frame rate to avoid reductions in resolution or image quality SNR are likely to need constraints.

This type of operation results in better performance for simple applications where the implementor isn't as knowledged about the need to initiate signalling to trigger a change of video resolution. Thus providing good performance in more cases and having less amount of code in their applications.

Still more advanced applications should have influence on the behavior. This can be realized in several ways. One is to use the constraints to inform the browser about the application's preferences how to treat the different media streams, thus affecting how COP is used. If required, it is possible to provide additional API functionalities for the desired controls.

The authors are convinced that providing ease of use for the simple application is important. Providing more advanced controls for the advanced applications is desirable.

5. SDP Usage

SDP SHALL be used to establish boundaries and capabilities for the media codec control in WebRTC. This includes the following set of capabilities that is possible to express in SDP:

Codec Capabilities: For all media codecs it is needed to determine what capabilities that are available if there exist optional functionalities. This concerns both media encoding and the RTP payload format as codec control can affect both. For codecs where the span of complexities are large there might exist need to express the level of complexity supported. For Video codecs like H.264 this can be expressed by the profile level ID. These capabilities are expected to be defined by the RTP payload format or in SDP attributes defined in the RTP payload formats to be used.

COP Parameters Supported: SDP SHALL be used to negotiate the set of COP parameters that the peers can express preferences for and for which they will send notification on their sets of parameter values used.

6. COP Usage

An WebRTC end-point SHALL implement Codec Operation Point RTCP Extension [I-D.westerlund-avtext-codec-operation-point]. The following COP parameters SHALL be supported:

- Payload Type
- Bitrate
- Token Bucket Size
- Framerate
- Horizontal Pixels
- Vertical Pixels
- Maximum RTP Packet Size
- Maximum RTP Packet Rate
- Application Data Unit Aggregation

Please note that also the ALT and ID parameters must be implemented

in COP for COP to correctly function.

To make COP usage efficient the end-point SHALL implement Reduced size RTCP packets [RFC5506].

To provide in addition to requesting specific frame-rates also the RTCP Codec Control Messages "Temporal-Spatial Trade-off Request and Notification" [RFC5104] . This enables a receiver to make a relative indication of their preferred trade-off between spatial and temporal quality. This provides an highly useful indication to the media sender about what the receiver prefer in a relative sense. The COP framerate or resolution parameters can be used to further provides target, max or min values to further indicate within which set of parameters the sender should find this relative trade-off.

To enable an receiver to temporarily halt or pause delivery of a given media stream an WebRTC end-point SHALL also implement "RTP Media Stream Pause and Resume" [I-D.westerlund-avtext-rtp-stream-pause]. This is important COP related features as described by the use case and motivations to enable the receiver to indicate that it prefers to have a given media stream halted if the aggregate media bit-rate is reduced. It can also be used to recover aggregate media bit-rate when the application has no current use of a given media stream, but may rapidly need it again due to interactions in the application or with other instances.

7. IANA Considerations

This document makes no request of IANA.

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

8. Security Considerations

The usage of COP and its security issues are described in [I-D.westerlund-avtext-codec-operation-point]. The main threats to this usage of COP are the following things:

a. That the SDP based codec boundary signalling and COP parameter negotiation could be intercepted and modified. Thus enabling denial of service attacks on the end-points reducing the scope of the COP usage and the media codec parameters to provide sub-optimal quality or block certain features. To prevent this the SDP needs to be authenticated and integrity protected.

b. The COP messages themselves could be modified to affect the negotiated codec parameters. This could have sever impact on the media quality as media streams can be completely throttled, or configured to very reduced framerate or resolution. To prevent this source authentication and integrity protection must be applied to the RTCP compound packets.

c. In multi-party applications of COP an entity may need to combine multiple sets of requested parameters. In these multi-party cases a particular participant may target the other participants and actively try to degrade their experience. Any COP entity merging sets will need to consider if a particular participant is actively harmful to the others and can chose to ignore that entities request.

9. Acknowledgements

10. References

10.1. Normative References

[I-D.ietf-rtcweb-jsep]
Uberti, J. and C. Jennings, "Javascript Session Establishment Protocol", draft-ietf-rtcweb-jsep-00 (work in progress), March 2012.

[I-D.westerlund-avtext-codec-operation-point]
Westerlund, M., Burman, B., and L. Hamm, "Codec Operation Point RTCP Extension", draft-westerlund-avtext-codec-operation-point-00 (work in progress), March 2012.

[I-D.westerlund-avtext-rtp-stream-pause]
Akram, A., Burman, B., Grondal, D., and M. Westerlund, "RTP Media Stream Pause and Resume", draft-westerlund-avtext-rtp-stream-pause-01 (work in progress), May 2012.

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

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session

Description Protocol", RFC 4566, July 2006.

[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.

[RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.

10.2. Informative References

[I-D.alvestrand-rtcweb-resolution]
Alvestrand, H., "RTCWEB Resolution Negotiation", draft-alvestrand-rtcweb-resolution-00 (work in progress), April 2012.

[I-D.ietf-payload-vp8]
Westin, P., Lundin, H., Glover, M., Uberti, J., and F. Galligan, "RTP Payload Format for VP8 Video", draft-ietf-payload-vp8-04 (work in progress), March 2012.

[I-D.ietf-rtcweb-use-cases-and-requirements]
Holmberg, C., Hakansson, S., and G. Eriksson, "Web Real-Time Communication Use-cases and Requirements", draft-ietf-rtcweb-use-cases-and-requirements-07 (work in progress), April 2012.

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

[RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, February 2008.

[RFC6184] Wang, Y., Even, R., Kristensen, T., and R. Jesup, "RTP Payload Format for H.264 Video", RFC 6184, 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 10 714 13 11
Email: bo.burman@ericsson.com