Simulcast and layered video coding support in WebRTC

don-garcia-simulcast-and-layered-video-webrtc-00

Abstract

This document describes the use cases and requirements for simulcast and layered video coding support in WebRTC. These techniques simplify the implementation of video stream adaptation to different participants in centralized conferencing solutions. This document also includes a proposal to expose these capabilities in the existing PeerConnection API by defining new media constraint properties.

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 February 06, 2014.

Copyright Notice

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

Video conferencing using a central server is one of the typical use cases for real-time communication capabilities in browsers [I-D.ietf-rtcweb-use-cases-and-requirements].

Most of today’s multiparty videoconference solutions make use of centralized servers to reduce the bandwidth and CPU consumption in the endpoints. Those servers receive streams from each participant and send the streams to rest of the participants, which usually have heterogeneous capabilities (screen size, CPU, bandwidth, etc.). One of the biggest issues is how to perform the adaption to different participants’ constraints with the minimum possible impact on video quality and server performance.
There are two approaches to adapt the streams to different destinations: one is transcoding (sometimes including mixing), and the other is switching between multiple streams or sub-streams received from the originator. The first solution is computationally expensive and can degrade video quality. The second solution makes a suboptimal use of network resources by sending redundant information, and in addition it is codec-specific.

The requirements and proposed API in this document are based on existing JSEP API version and VP8 capabilities. These are the technologies available in existing WebRTC browsers, but this proposal could be extended to other codecs or mapped to other APIs.

1.1. Browser support status

It is possible to use simulcast with existing WebRTC implementations. However, this requires the use of different PeerConnection objects, and all streams will have the same resolution.

Multi-layer encoding is implemented and working in existing WebRTC browsers, and it has been tested in prototypes, but currently there is no way for developers to enable it. In VP8 there is support for temporal scalability, while VP9 will include more advanced control and support for both temporal and spatial scalability.

1.2. 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 RFC 2119 [RFC2119].

2. Use-cases

The use cases envisioned for these new WebRTC capabilities are focused on centralized conferencing solutions.

2.1. Adaptation to devices with different capabilities

Some endpoints connected to a centralized conferencing server have small screens and do not need to receive high-resolution video, or the CPU power and battery consumption make it impossible to receive and decode high-resolution video in real-time.

In this situation, it is desirable to send lower-resolution video to those endpoints.

2.2. Adaptation to participants with different network conditions
Some endpoints connected to a centralized conferencing server do not have enough available bandwidth to receive high-quality video, while other endpoints have enough available bandwidth.

In this situation is desirable to send lower-bitrate video to those endpoints.

2.3. Recording

A conferencing server implements recording and wants to record video in the highest quality possible, while forwarding it in lower quality to endpoints.

2.4. Increasing video quality for active speaker

A videoconference application shows the video of the active speaker in a larger size than videos of the other participants.

It is desirable to increase the resolution and quality of that highlighted video stream, to maintain the perceived video quality. One possible implementation to increase the quality is to have a paused high-quality stream that resumes when voice activity is detected.

3. Requirements

This section contains the requirements for the API exposed in the browser, derived from the use-cases in Section 2.

Requirements on how and when to enable scalable video coding:

- REQ-1. It must be possible to enable and configure the scalable video coding before initiating a peer connection.
- REQ-2. It must be possible to enable and configure the scalable video coding before answering a peer connection.
- REQ-3. It must be possible to enable/disable and re-configure the scalable video coding to update a peer connection.

Requirements on the parameters that needs to be configurable:

- REQ-5. It must be possible to configure the number of simulcasted streams.
- REQ-6. It must be possible to configure the minimum and maximum bitrate of each simulcasted stream.
o REQ-7. It must be possible to configure the resolution of each simulcasted stream.

o REQ-8. It must be possible to configure the number of temporal layers (1 to 4). This should be the only mandatory parameter when enabling temporal scalability.

o REQ-9. It must be possible to configure the bitrate, frame rate decimation factor and membership of frames to layers for each temporal layer of the VP8 stream.

Requirements regarding RTP usage:

o REQ-10. Congestion control must be supported for all the simulcasted streams between the configured boundaries (min/max bitrate).

o REQ-11. Transmission of simulcasted streams must be signaled and negotiated in the SDP and transmitted in RTP sessions, making use of existing standard attributes [I-D.westerlund-avtcore-multistream-and-simulcast].

o REQ-12. Any endpoint should be prepared to receive VP8 multi-layered encoded video not requiring out of band negotiation in SDP.

Non functional requirements:

o REQ-13. The exposed API must be extensible to new codecs or new codec parameters.

4. Proposed API

The existing solution in the WebRTC API to modify settings of a PeerConnection is to use media constraints. This section defines some new media constrains to enable and configure the usage of simulcasted and layered video streams.

4.1. Simulcasted streams

Simulcast capabilities are codec-agnostic and do not require new media constraints. Existing media constrains for resolution, frame rate and bitrate can be reused, but the API needs to support receiving a list of them instead of just one.

4.2. Layered video coding
Multi-layer capabilities are codec-dependent. For VP8, these are the configuration parameters exposed in the codec, and that needs to be translated to media constraints (the descriptions are taken from VP8 source code):

- **tsNumberLayers**: This value specifies the number of coding layers to be used.
- **tsTargetBitrate**: These values specify the target coding bitrate for each coding layer.
- **tsRateDecimator**: These values specify the frame rate decimation factors to apply to each layer.
- **tsPeriodicity**: This value specifies the length of the sequence that defines the membership of frames to layers. For example, if tsPeriodicity=8 then frames are assigned to coding layers with a repeated sequence of length 8.
- **tsLayerId**: This array defines the membership of frames to coding layers. For a 2-layer encoding that assigns even numbered frames to one layer (0) and odd numbered frames to a second layer (1) with tsPeriodicity=8, then tsLayerId = (0,1,0,1,0,1,0,1).

### 4.3. Example

Example of media constraints to request two simulcasted streams, the first one with four temporal layers and default bitrate and the second one with a single layer and fixed bitrate.

```json
{
  video: [{
    width: 640,
    height: 480,
    codecs: {
      vp8: { tsNumberLayers: 4 }
    }
  },
  {
    width: 320,
    height: 240,
    bitrate: { min: 100000, max: 100000 }
  }
}
```
5. Acknowledgements

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

No security implications foreseen.

8. References

8.1. Normative References


8.2. Informative References


Garcia

Email: gustavo@tokbox.com