Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 27, 2014

H. Alvestrand
Google

August 26, 2013

Resolution Constraints in Web Real Time Communications
draft-alvestrand-constraints-resolution-03

Abstract

This document specifies the constraints necessary for a Javascript application to successfully indicate to a browser that supports WebRTC what resolutions it desires on a video stream.

It also discusses the possible use of SDP to carry that information between browsers.

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].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

There are a number of scenarios where it's useful for a WebRTC application to indicate to the WebRTC implementation in the supported browser what the desired characteristics of a video stream are. These include, but are not limited to:

- o Specifying a minimum desired resolution for a given application, in order to control the user experience or resource tradeoffs made by the browser to favour a particular stream

- o Specifying a maximum desired resolution for a given stream, in order to save some resource (bandwidth, CPU....), possibly outside of the browser where the browser can't tell that it's exceeding a constraint

- o Specifying resolutions that are a reasonable fit for the current usage of the video stream, for instance fitting with the number of pixels available on the part of a device's display surface that is devoted to displaying this video stream

- o Specifying the shape of a video stream, in order to fit the video onto a display surface without the need for black bars or image distortion

Similar considerations apply for framerate.

1.1. Disposition of this text

This draft is written in order to get something specific out to refer to during spec-writing and implementation. Some text may eventually get merged into the JSEP specification, [I-D.ietf-rtcweb-jsep].

2. Usage Scenarios

Consider the following (simplified) model of a video stream through a WebRTC application:

```
   |<-------------- Browser A -------------------->|
  Camera ---> MediaStream A ---> Peerconnection A ------+
          |<------- Application A ---------->|          |
                          v  ^                          v
                 Signalling channel              Internet (media)
                          v  ^                          |
          |<------- Application B ---------->|          |
  <video> tag <-- MediaStream B <--- Peerconnection B --+
   |<-------------Browser B ----------------------->|
```

Both applications are running in browsers, with Application A connected to a camera that is able to deliver video streams up to HD quality (1280x720).

2.1. Scenario: Resolution change

At one particular moment in time, the <video> tag in Application B is rendered as a thumbnail, and video is flowing to it in a 160x100

resolution; there is no need to send any more data, since no more pixels are available for its display anyway.

Then the user of Application B hits the "full-screen" button. There are now 1600x1200 pixels available for display.

Initially, Application B will splay the 160x100 image across the larger surface, because there is no other choice, but it will desire to have as many pixels as possible available to provide a high quality image.

2.2. Scenario: Constrained bandwidth

At one particular moment in time, the camera is generating 1280x720, resulting in a 2 Mbits/second data flow from A to B. Congestion control signals that this data rate is no longer available; rather than letting the browser reduce the bandwidth of some flow of its choice, Application A decides that the high definition video is the feature that is least valuable. It can then apply a new constraint to Mediastream A, specifying that resolution should be at most 640x360; browser A is then responsible for making sure this decision is communicated to browser B (if it needs to be).

2.3. Scenario: Limited processing capacity

If application B is running on a slow machine (2000-class PC or 2010-class mobile phone), the maximum capacity of the video decoder may be 320x200 - Application B may then wish to indicate that application A should limit the stream sent across the network to that resolution - sending more bits isn't useful, because the receiver doesn't have enough capacity to decode and downscale the video stream.

3. Models for resolution manipulation

As specified in the "Media Capture And Streams" document [W3C.WD-mediacapture-streams-20130516] the consumer of a video track in a MediaStream will have a "native resolution", which indicates what size video it's useful to push to it. The application can also set (and change) constraints on the video MediaStreamTrack, indicating which range of properties it sees useful for the purposes of the application.

In SDP, the "a=imageattr" attribute is available to provide information on the resolution of video streams described by an SDP m-line.

If both mechanisms are available, the choices available to the writer of application B in the "increase screen area" above are:

1. Signal (by non-standard means) to Application A that more pixels are needed. Application A will then modify the constraints on Mediastream A to say that the desired (not mandatory) min resolution is 1600x1200; Browser A will then reconfigure the camera to generate the closest available resolution, which is 1280x720.

2. Apply a new constraint set to Mediastream B's video track, saying that the desired resolution is now 1600x1200. Browser B will then have to figure out that this is an incoming track via Peerconnection B, and that the resolution needs to be signalled; it will then fire a NegotationNeeded event at Application B, which will then renegotiate the desired resolutions using an SDP exchange with Browser A; Browser A will then figure out from the SDP that it's useful to generate a higher resolution video stream, and reconfigure the camera as above.

3. Execute a renegotiation with Application A, adding attributes as described in Section 5 by modifying the SDP generated by CreateOffer, and triggering the behaviour in the previous alternative inside Browser A. API-wise, this is perhaps the most complex method.

The advantage of the first method is that it does not require any SDP parsing or generation.

The advantage of the second method is that it will work when appliation A and application B are different applications; there is no need for them to have any private agreement on how to set bitrate. It does require both the implementation of constraints and that browser B has the ability to generate the proper constraints in the SDP.

The third method requires SDP parsing in browser A, but not SDP generation in browser B. It does require SDP manipulation in Javascript at application B.

3.1. Sender-side constraint manipulation

The following Javascript code (somewhat pseudocode) will achieve the "increase screen area" according to method 1 above.

The examples use APIs from "Media Capture And Streams as well as from WebRTC [W3C.WD-webrtc-20120821].

```
// B side
function needResolutionToChange(newWidth, newHeight) {
   message = makeMessage("resolution", newWidth, newHeight);
   remote.send(message)
}

// A side
function handleMessage(message) {
  if (message.verb === "resolution") {
     constraints = video.constraints();
     // The function below can be a polyfill.
     constraints.replaceOrAddOptConstraint("width", message.arg1);
     constraints.replaceOrAddOptConstraint("height", message.arg2);
   }
}
```

3.2. Receiver-side constraint manipulation

This implements version 2 of the constraint manipulation above. Note that the handing of "onnegotiationneeded" is the same as for any other renegotiation.

```
// B side
function needResolutionToChange(newWidth, newHeight) {
  constraints = video.constraints();
  constraints.replaceOrAddOptConstraint("width",
                                        { "max": message.arg1 });
  constraints.replaceOrAddOptConstraint("height",
                                        { "max": message.arg2 });
  video.applyCoinstraints(constraints);
}

pc.onnegotiationneeded = function() {
  offer = pc.CreateOffer();
  message = makeMessage("offer", offer);
  remote.send(message);
}

// Functions to handle answer from A side omitted.

// A side
function handleMessage(message) {
  if (message.verb === "offer") {
    pc.SetRemoteDescription(message.arg1, success, failure)
  }
```

```
}
// Functions to return answer to B side omitted.
```

4. Constraints for specifying resolution

All constraints needed are registered in the IANA registry by [W3C.WD-mediacapture-streams-20130516]. In summary, they are:

- o width - unsigned long or MinMaxConstraint - ie width: { min: 640, max: 1024 }
- o height - unsigned long or MinMaxConstraint
- o frameRate - float or MinMaxConstraint

5. Syntax and Mapping Examples

See Section 6 for the actual definition of the constraints used here.

5.1. Examples with GetUserMedia

A constraint saying that we absolutely must have a minimum resolution of 1024x768:

```
getUserMedia({
   video: { mandatory: { width: { min: 1024 }, height: { min: 768 }}}
}, successCallback, errorCallback);
```

A constraint saying that we'd prefer 60 frames per second, if available, and if we can get that, we'd like to lock the resolution to 640x480, but in all cases, the screen must be clamped to a 4:3 aspect ratio - 16:9 or odd aspect ratios are not acceptable to this application:

```
getUserMedia({
   video: {
     mandatory: { aspectRatio: { min: 1.333, max: 1.334 } },
     optional [
       { frameRate: 60 },
       { width: 640 },
       { heigth: 480 }
     ]
```

```
      }
   }, successCallback, errorCallback);
```

5.2. SDP mappings

The examples below are based on [I-D.roach-mmusic-unified-plan] and [I-D.ietf-mmusic-msid].

An optional constraint has been applied to an incoming stream where both upper and lower are constrained to 320x200. The stream has been assigned to a hardware video decoder that can decode most resolutions up to 1024x768, in any aspect ratio, but only if all divisions are divisible by 16. The incoming stream has MediaStream ID aaaa, and MediaStreamTrack id bbbb.

Escaped line breaks are added for readability.

```
m=video
a=imageattr:* [x=320,y=200,q=1.0] \
                  [x=[120:16:1024],y=[100:16:768],q=0.2]
a=msid: aaaa bbbb
```

6. IANA Considerations

This document makes no requests of IANA:

Note to RFC Editor: This section can be deleted before publication as an RFC.

7. Security Considerations

No security considerations particular to these specific constraints have so far been identified.

8. Acknowledgements

Special thanks are given to Dan Burnett, Cullen Jennings, the IETF RTCWEB WG and the W3C WEBRTC WG for strongly influencing this memo, and to Per Kjellander for being the first to implement the constraints in getUserMedia.

9. References

9.1. Normative References

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

9.2. Informative References

[I-D.ietf-mmusic-msid]
Alvestrand, H., "Cross Session Stream Identification in the Session Description Protocol", draft-ietf-mmusic-msid-01 (work in progress), August 2013.

[I-D.ietf-rtcweb-jsep]
Uberti, J. and C. Jennings, "Javascript Session Establishment Protocol", draft-ietf-rtcweb-jsep-03 (work in progress), February 2013.

[I-D.roach-mmusic-unified-plan]
Roach, A., Uberti, J., and M. Thomson, "A Unified Plan for Using SDP with Large Numbers of Media Flows", draft-roach-mmusic-unified-plan-00 (work in progress), July 2013.

[W3C.WD-mediacapture-streams-20130516]
Burnett, D., Bergkvist, A., Jennings, C., and A. Narayanan, "Media Capture and Streams", World Wide Web Consortium WD WD-mediacapture-streams-20130516, May 2013, <http://www.w3.org/TR/2013/WD-mediacapture-streams-20130516>.

[W3C.WD-webrtc-20120821]
Bergkvist, A., Burnett, D., Narayanan, A., and C. Jennings, "WebRTC 1.0: Real-time Communication Between Browsers", World Wide Web Consortium WD WD-webrtc-20120821, August 2012, <http://www.w3.org/TR/2012/WD-webrtc-20120821>.

Appendix A. Changes from -00 to -01

Added the "Usage Scenarios" chapter.

Repointed the eventual target to be incorporation in the JSEP draft.

Made sure the constraints are consistently spelled in camelCase, with a small initial letter.

Appendix B. Changes from -01 to -02

Moved a bit of the text around between sections, and referred to the "settings API" proposal from the Media Capture task force.

Appendix C. Changes from -02 to -03

Retargeted document to be a "here's how you can do it" draft.

Updated constraints format to be as per the May 16 W3C draft of "media capture and streams".

Author's Address

Harald Alvestrand
Google

Email: harald@alvestrand.no