W3C WebRTC WG TPAC Meeting

September 19-20, 2019
Fukuoka, Japan
https://meet.google.com/vxf-wgai-rjn

Chairs: Bernard Aboba
Harald Alvestrand
Jan-Ivar Bruaroey

W3C WG IPR Policy

- This group abides by the W3C Patent Policy https://www.w3.org/Consortium/Patent-Policy/
- Only people and companies listed at https://www.w3.org/2004/01/pp-impl/47318/status are allowed to make substantive contributions to the WebRTC specs

Welcome!

- Welcome to the Thursday meeting of the W3C WebRTC WG at TPAC 2019!
- During this meeting, we hope to make progress on bringing WG specifications to CR and PR.

About these Meetings

Information on the meeting:

- Room: Sakura, 3F
- Meeting info:
  - https://www.w3.org/2011/04/webrtc/wiki/September_19-20_2019
- Link to latest drafts:
  - https://w3c.github.io/mediacapture-main/
  - https://w3c.github.io/mediacapture-output/
  - https://w3c.github.io/mediacapture-screen-share/
  - https://w3c.github.io/mediacapture-record/
  - https://w3c.github.io/mediacapture-fromelement/
  - https://w3c.github.io/webrtc-pc/
  - https://w3c.github.io/webrtc-stats/
  - https://w3c.github.io/webrtc-svc/
  - https://www.w3.org/TR/mst-content-hint/
  - https://w3c.github.io/webrtc-nv-use-cases/
  - https://w3c.github.io/webrtc-dscp-exp/
- Link to Slides has been published on WG wiki
- Scribe? IRC http://irc.w3.org/ Channel: #webrtc
- The meeting is being recorded.

Morning Agenda for Thursday, September 19

8:30 AM - 9:00 AM State of the WEBRTC WG (Harald)
Status of specifications and implementations.

Special mention: MediaStream Recording API: https://w3c.github.io/mediacapture-record/

9:00 - 9:30 AM Test Status (Dr. Alex)

9:30 AM - 10:00 AM WebRTC-PC (Bernard)
WebRTC-PC: https://w3c.github.io/webrtc-pc/

10:00 AM - 10:30 AM Break

10:30 AM - 11:00 AM Next steps to PR for WebRTC-PC (Bernard)

11:00 AM - Noon Capture (Jan-Ivar)
Screen Capture: https://w3c.github.io/mediacapture-screen-share/
MediaCapture & Streams: https://w3c.github.io/mediacapture-main/
MediaStream Recording: https://w3c.github.io/mediacapture-record/MediaRecorder.html

noon - 1:00 PM Lunch

Afternoon Agenda for Thursday, September 19

1:00 PM - 2:00 PM WebRTC-Stats (Varun & Henrik)
Reference: https://w3c.github.io/webrtc-stats/

2:00 PM - 2:30 PM Content-hints (Harald)
Content-Hints: https://w3c.github.io/mst-content-hint/

2:30 PM - 3:00 PM Other current specifications (Harald and Peter)
DSCP API: https://www.w3.org/TR/webrtc-dscp/
WebRTC-ICE: https://github.com/w3c/webrtc-ice

3:00 PM - 3:30 PM Break

3:30 PM - 4:30 PM WebRTC-PC "Features at risk" (Jan-Ivar)

4:30 PM - 5:30 PM WEBRTC WG Developer Feedback Session

Agenda for Friday, September 20

8:30 AM - 9:15 AM Scalable Video Coding Extension for WebRTC (Bernard & Florent)
Reference: https://w3c.github.io/webrtc-svc/

9:15 AM - 10 AM Privacy Issues (Youennf)
MediaCapture & Streams: https://w3c.github.io/mediacapture-main/
Audio Output Devices API: https://w3c.github.io/mediacapture-output/
Media Capture from DOM Elements: https://w3c.github.io/mediacapture-fromelement/
WebRTC-PC: https://w3c.github.io/webrtc-pc/

10:00 AM - 10:30 AM Break
10:30 AM - 11:00 AM WebRTC NV use cases (Bernard)
Reference: https://w3c.github.io/webrtc-nv-use-cases/

11:00 AM - 12:00 PM WEBRTC WG Re-Charter (Dom)

12:00 PM - 1 PM Lunch
1 PM - 2 PM Joint meeting with Accessibility Platform Architectures WG (Bernard)
Reference: https://www.w3.org/WAI/APA/wiki/Accessible_RTC_Use_Cases

2 PM - 3 PM WebRTC-Stats (Varun & Henrik)
Reference: https://w3c.github.io/webrtc-stats/
3 PM - 3:30 PM Break
3:30 PM - 4:30 PM TBD
4:30 PM - 5:30 PM Wrapup and Next Steps (Harald)

State of the W3C WEBRTC WG (Harald, 30 minutes)

What we're chartered to do

- Finish WebRTC 1.0 (HIGH PRIORITY)
- Define an object-oriented API (based on ORTC)
- Describe requirements for new use cases
- Address those use cases
    - New protocols (and associated APIs)
    - New data access functions

What our environment demands

- WebRTC 1.0 should “just work”
  - Across all browsers
  - In all networks
- Low level data access
  - In a performant manner (example: link)
- Son of ORTC
  - Pressure seems to have decreased
  - WebTransport, WebCodec spinning out

Media Capture and Streams

- Candidate Recommendation (Oct 17)
- 26 open issues
  - 21 of which are > 3 months old
- Interoperability matrix shows lots of things working in ¾ of browsers
- Community sense seems to be “works”
- Promise: PR in Q4 2018 (that’s last year!)

Screen Capture

- Triggered by external event (chrome app store)
- Push to bring functionality up to par with existing implementations based on gUM.
- Security still troublesome, but can’t live without
- TAG review needed

WebRTC-1.0

- Candidate Recommendation
  - Renewed Sep 18 2017 (separated Identity spec)
- 32 open issues
  - 17 are > 3 months
- Interoperability matrix shows lots of issues, but also lots of interoperability
- Confluence map shows implementation progress across the board.
- Community sense “are we usable yet”?
- Promise: PR in Q3 2019 (that’s now!)

WebRTC-Identity

- Candidate Recommendation (split sept)
- 24 open issues
  - Newest issue is from 2017
- Test suite has been separated
- Promise: PR in Q3 2019 (as for webrtc-pc)
- Community sense: “Not much happening”

Resources available to WG

- Editors: 2 editors (Henrik and Youenn) currently active on mediacapture-streams, screen-capture and webrtc-pc (Jan-Ivar continues to write)
  - Some others contribute PRs - THANK YOU!
- Adam, Taylor and Dan have left the editor team since last TPAC
  - THANK YOU for all your efforts!
- Other drafts managed by other editors

Where resources come from

- People are motivated to get stuff done that they care about
- Organizations sponsor people to get stuff done that they care about
- W3C is a "gift economy" - to make something happen, volunteer to work on it!
- Careful balance of "polish" vs "new work" needed - otherwise, new work goes elsewhere

Other documents - active

- Capture from DOM - heavy use
  - Need to find new editor(s)
  - Need privacy/security,TAG review
  - 19 open issues
- Recorder - heavy use, updates
  - Also one suggested path for “media access”
  - Need to find new editor(s)
  - Need privacy/security,TAG review
- Stats identifiers - updates
  - Linked to webrtc-pc

Other documents - quiet

- Depth - quieted down?
- Audio output devices - at CR, in use, little activity
- Content hints - released, PRs merged, only a few open issues.
- DSCP - no code, no activity
- WebRTC-SVC - “intent to implement” in Chrome
- WebRTC-ICE - implemented in Chrome

We should eventually kill or finish these.

Other W3C activity

- WebCodec initiative - Media WG interest
- WebTransport initiative - replacing RTP
- Timed Text - Web and TV
- Media Timed Events - Web and TV
- Media WG in general
- Security and Privacy issues

Attention focus for this meeting

- Finish 1.0
  - Get all the bugs resolved
  - Figure out how to get to interop across the board
- Look at new APIs
  - Where what we have is not enough
  - Use cases and requirements are key!
- Attend to Raw Media
  - Because that's where we're being asked to go

WebRTC Test Status
(Dr. Alex, 30 minutes)

WebRTC 1.0 Testing Status
W3C TPAC 2019

Dr. Alex Gouaillard

Number of WPT tests and coverage

webrtc/	1318
webrtc-stats/	5
mediacapture-streams/	249
mediacapture-fromelement/	45
screen-capture/	21

Total Tests

	2016	2017	2018	2019
webrtc/	293	1296	1318	1588
coverage	N/A	69.57%	N/A	N/A

Yearly Progress

Coverage Status - TPAC 2017

- PR #8051: Add coverage report and tools for WebRTC tests
- Coverage = (total - todo) / total

```
$ cd webrtc/tools
$ node scripts/overview.js
Overall Coverage
====================
todo            |    248
tested          |    315
trivial         |    173
untestable      |     79
====================
total           |    815
coverage        | 69.57%
====================
```

4. Peer-to-peer connections	67.83%
5. RTP Media API	67.01%
6. Peer-to-peer Data API	71.87%
7. Peer-to-peer DTMF	93.54%
8. Statistics Model	100.00%
9. Identity	86.04%
10. Media Stream API Extensions for Network Use	35.71%

WebRTC WPT results - Oct 2018

Path	Chrome 70 Linux 18.04 @d44bc3ed38 Oct 20 2018	Edge 17 Windows 10 @d44bc3ed38 Oct 21 2018	Firefox 62 Linux 18.04 @d44bc3ed38 Oct 20 2018	Safari 11.1 macOS 10.13 @d44bc3ed38 Oct 21 2018
mediacapture-depth/	6 / 6	0 / 1	6 / 6	6 / 6
mediacapture-fromelement/	42 / 45	0 / 5	22 / 45	32 / 45
mediacapture-image/	129 / 177	0 / 20	64 / 177	82 / 173
mediacapture-record/	49 / 72	0 / 2	56 / 72	2 / 72
mediacapture-streams/	207 / 249	0 / 30	186 / 249	5 / 34
screen-capture/	8 / 21	0 / 2	8 / 21	7 / 11
webrtc/	579 / 1318	0 / 88	700 / 1318	226 / 555
webrtc-stats/	4 / 5	0 / 1	4 / 5	4 / 5

Path	Tests Passing in *X* / 4 Browsers				
	0 / 4	**1 / 4**	**2 / 4**	**3 / 4**	**4 / 4**
mediacapture-depth/	0 / 6	0 / 6	0 / 6	6 / 6	0 / 6
mediacapture-fromelement/	1 / 45	11 / 45	20 / 45	13 / 45	0 / 45
mediacapture-image/	48 / 177	46 / 177	34 / 177	49 / 177	0 / 177
mediacapture-record/	31 / 95	23 / 95	40 / 95	1 / 95	0 / 95
mediacapture-streams/	29 / 293	137 / 293	126 / 293	1 / 293	0 / 293
screen-capture/	13 / 21	0 / 21	1 / 21	7 / 21	0 / 21
webrtc/	509 / 1318	302 / 1318	387 / 1318	120 / 1318	0 / 1318
webrtc-stats/	1 / 5	0 / 5	0 / 5	4 / 5	0 / 5

2019 - WPT.fyi: more tests, better separated

Path	Chrome 78 Linux 18.04 3472889 Sep 18, 2019	Edge 78 Windows 10.0 3472889 Sep 18, 2019	Firefox 71 Linux 18.04 3472889 Sep 18, 2019	Safari 82 preview macOS 10.13 3472889 Sep 18, 2019
media-capabilities/	151 / 179	151 / 179	132 / 179	97 / 179
media-playback-quality/	21 / 21	21 / 21	21 / 21	9 / 21
media-source/	521 / 589	520 / 589	420 / 526	383 / 517
mediacapture-depth/	6 / 6	6 / 6	6 / 6	6 / 6
mediacapture-fromelement/	42 / 44	42 / 44	21 / 44	31 / 44
mediacapture-image/	141 / 181	138 / 181	66 / 181	84 / 180
mediacapture-record/	79 / 96	73 / 96	71 / 96	9 / 96
mediacapture-streams/	405 / 425	186 / 327	276 / 345	276 / 381
webrtc/	1303 / 1588	1266 / 1588	999 / 1588	877 / 1588
webrtc-identity/	3 / 22	3 / 22	11 / 22	3 / 22
webrtc-quic/	115 / 115	115 / 115	2 / 115	2 / 115
webrtc-stats/	15 / 16	15 / 16	15 / 16	15 / 16
webrtc-svc/	1 / 3	3 / 3	1 / 3	1 / 3

2018 ~ 2019 - WPT.fyi: much more passing tests

Path	Tests Passing in X / 4 Browsers				
	0 / 4	1 / 4	2 / 4	3 / 4	4 / 4
mediacapture-depth/	0 / 6	0 / 6	0 / 6	6 / 6	0 / 6
mediacapture-fromelement/	1 / 45	11 / 45	20 / 45	13 / 45	0 / 45
mediacapture-image/	48 / 177	46 / 177	34 / 177	49 / 177	0 / 177
mediacapture-record/	31 / 95	23 / 95	40 / 95	1 / 95	0 / 95
mediacapture-streams/	29 / 293	137 / 293	126 / 293	1 / 293	0 / 293
screen-capture/	13 / 21	0 / 21	1 / 21	7 / 21	0 / 21
webrtc/	509 / 1318	302 / 1318	387 / 1318	120 / 1318	0 / 1318
webrtc-stats/	1 / 5	0 / 5	0 / 5	4 / 5	0 / 5

2018

media-capabilities/	19 / 179	9 / 179	17 / 179	48 / 179	86 / 179
media-playback-quality/	0 / 21	0 / 21	0 / 21	12 / 21	9 / 21
media-source/	27 / 589	40 / 589	81 / 589	123 / 589	318 / 589
mediacapture-depth/	0 / 6	0 / 6	0 / 6	0 / 6	6 / 6
mediacapture-fromelement/	1 / 44	0 / 44	11 / 44	14 / 44	18 / 44
mediacapture-image/	40 / 181	3 / 181	53 / 181	20 / 181	65 / 181
mediacapture-record/	32 / 119	11 / 119	15 / 119	53 / 119	8 / 119
mediacapture-streams/	1 / 425	85 / 425	95 / 425	108 / 425	136 / 425
webrtc/	125 / 1588	131 / 1588	359 / 1588	295 / 1588	678 / 1588
webrtc-identity/	11 / 22	8 / 22	0 / 22	0 / 22	3 / 22
webrtc-quic/	0 / 115	0 / 115	113 / 115	0 / 115	2 / 115
webrtc-stats/	1 / 16	0 / 16	0 / 16	0 / 16	15 / 16
webrtc-svc/	0 / 3	2 / 3	0 / 3	0 / 3	1 / 3

2019

2019 - WPT.fyi - /webrtc

Path	Chrome 78 Linux 18.04 3472889 Sep 18, 2019	Edge 78 Windows 10.0 3472889 Sep 18, 2019	Firefox 71 Linux 18.04 3472889 Sep 18, 2019	Safari 82 preview macOS 10.13 3472889 Sep 18, 2019
RTCRtpReceiver-getCapabilities.html	4 / 4	4 / 4	1 / 4	4 / 4
RTCRtpReceiver-getContributingSources.https.html	3 / 3	1 / 3	3 / 3	3 / 3
RTCRtpReceiver-getParameters.html	3 / 4	3 / 4	1 / 4	1 / 4
RTCRtpReceiver-getStats.https.html	1 / 3	1 / 3	1 / 3	1 / 3
RTCRtpReceiver-getSynchronizationSources.https.html	0 / 15	1 / 15	5 / 15	10 / 15
RTCRtpSender-getCapabilities.html	4 / 4	4 / 4	1 / 4	4 / 4
RTCRtpSender-getStats.https.html	1 / 3	1 / 3	1 / 3	1 / 3
RTCRtpSender-replaceTrack.https.html	8 / 10	8 / 10	10 / 10	10 / 10
RTCRtpSender-setParameters.html	1 / 2	1 / 2	2 / 2	2 / 2
RTCRtpSender-setStreams.https.html	6 / 6	6 / 6	1 / 6	1 / 6
RTCRtpSender-transport.https.html	9 / 9	1 / 9	1 / 9	1 / 9
RTCRtpTransceiver-direction.html	4 / 4	4 / 4	4 / 4	4 / 4
RTCRtpTransceiver-setCodecPreferences.html	14 / 14	14 / 14	1 / 14	1 / 14
RTCRtpTransceiver-stop.html	1 / 5	1 / 5	4 / 5	5 / 5
RTCRtpTransceiver.https.html	20 / 39	19 / 39	39 / 39	20 / 39
RTCSctpTransport-constructor.html	5 / 5	5 / 5	1 / 5	1 / 5
RTCSctpTransport-events.html	3 / 3	3 / 3	0 / 3	0 / 3
RTCSctpTransport-maxChannels.html	3 / 3	3 / 3	0 / 3	0 / 3
RTCSctpTransport-maxMessageSize.html	6 / 6	6 / 6	1 / 6	1 / 6
RTCTrackEvent-constructor.html	7 / 8	7 / 8	8 / 8	8 / 8
RTCTrackEvent-fire.html	8 / 10	8 / 10	10 / 10	5 / 10
datachannel-emptystring.html	1 / 2	1 / 2	1 / 2	1 / 2
getstats.html	2 / 2	2 / 2	1 / 2	2 / 2
historical.html	9 / 18	9 / 18	10 / 18	18 / 18
idlharness.https.window.html	470 / 509	470 / 509	318 / 509	322 / 509
legacy/	28 / 28	26 / 28	27 / 28	9 / 28
no-media-call.html	2 / 2	2 / 2	2 / 2	2 / 2
promises-call.html	2 / 2	2 / 2	2 / 2	2 / 2
protocol/	32 / 32	32 / 32	20 / 32	20 / 32
simplecall-no-ssrcs.https.html	2 / 2	2 / 2	2 / 2	2 / 2
simplecall.https.html	2 / 2	2 / 2	2 / 2	2 / 2

2018 - WPT.fyi & WPT.kite: mobile browsers

Path	Chrome 70 Linux 18.04 @d44bc3ed38 Oct 20 2018	Edge 17 Windows 10 @d44bc3ed38 Oct 21 2018	Firefox 62 Linux 18.04 @d44bc3ed38 Oct 20 2018	Safari 11.1 macOS 10.13 @d44bc3ed38 Oct 21 2018
mediacapture-depth/	6 / 6	0 / 1	6 / 6	6 / 6
mediacapture-fromelement/	42 / 45	0 / 5	22 / 45	32 / 45
mediacapture-image/	129 / 177	0 / 20	64 / 177	82 / 173
mediacapture-record/	49 / 72	0 / 2	56 / 72	2 / 72
mediacapture-streams/	207 / 249	0 / 30	186 / 249	5 / 34
screen-capture/	8 / 21	0 / 2	8 / 21	7 / 11
webrtc/	579 / 1318	0 / 88	700 / 1318	226 / 555
webrtc-stats/	4 / 5	0 / 1	4 / 5	4 / 5

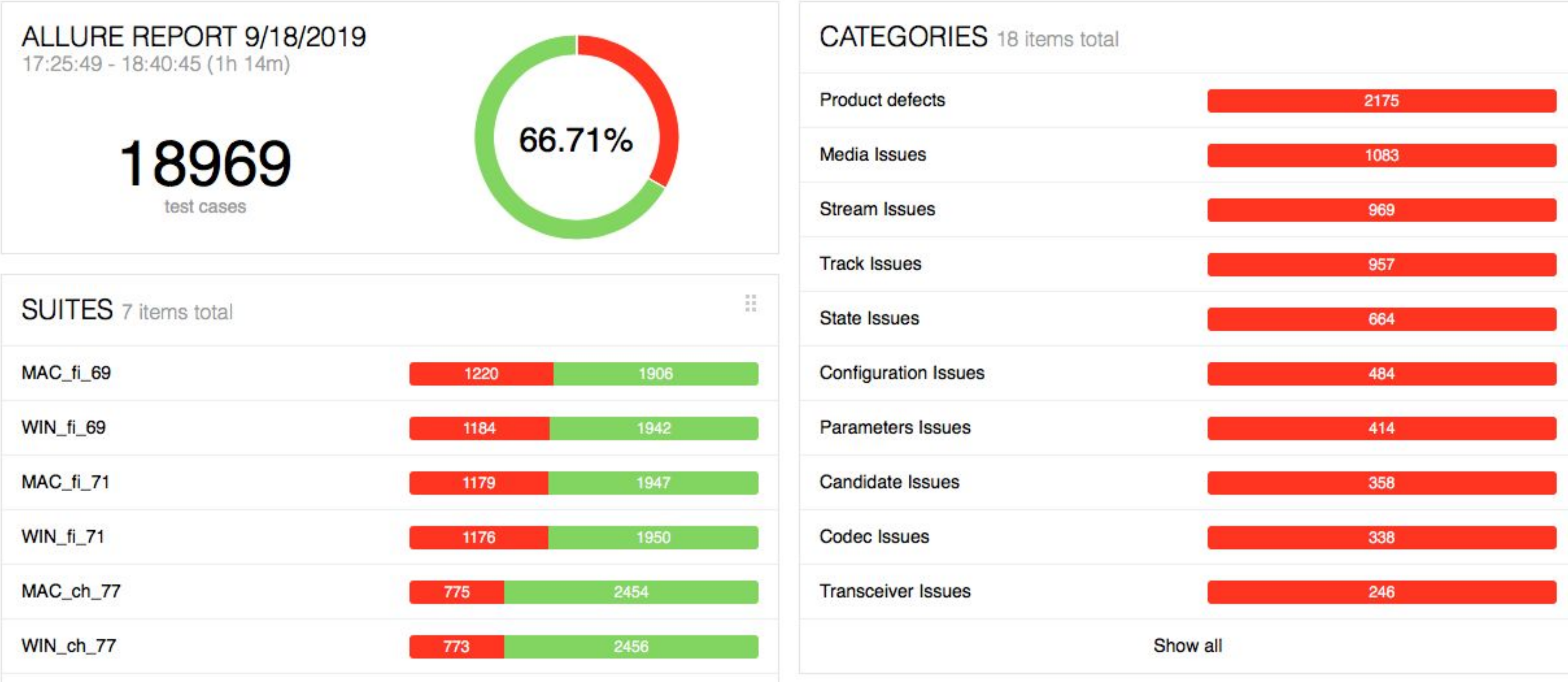
Dashboard / Web platform tests

webrtc	275/733	275/733	275/733	283/733	283/733	283/733	334/729	333/729	335/726	334/729	333/716	279/628	33/706	33/706	247/702	247/702	174/553	216/569
mediacapture-streams	49/58	49/58	49/58	47/58	47/58	49/58	28/57	28/57	28/57	28/57	34/56	34/57	36/58	36/58	29/58	32/58	7/49	39/54

2019 - WPT.fyi - /webrtc

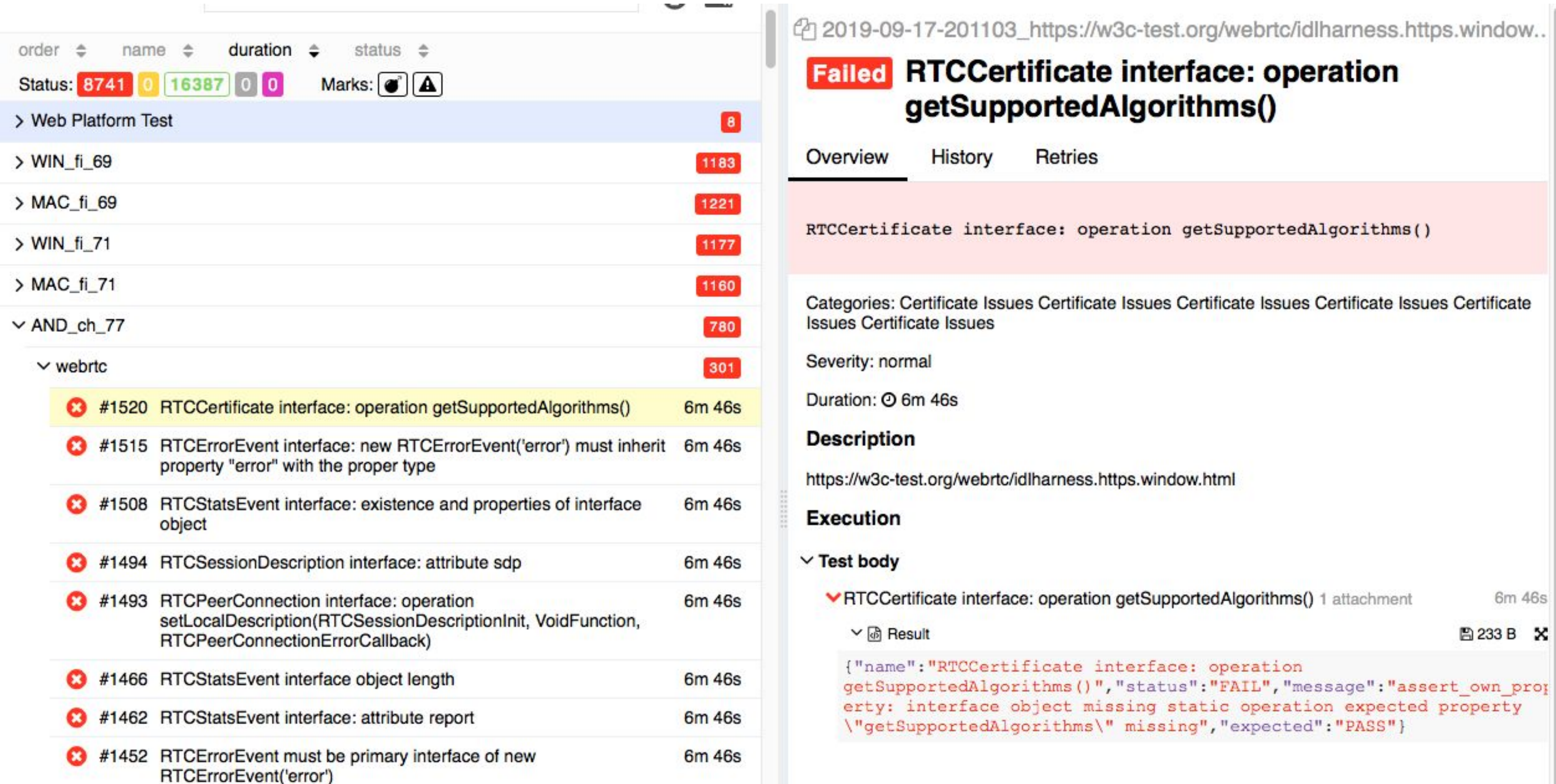
Subtest	Chrome 78 Linux 18.04 d473076 Sep 18, 2019	Edge 78 Windows 10.0 d473076 Sep 19, 2019	Firefox 71 Linux 18.04 d473076 Sep 18, 2019	Safari 82 preview macOS 10.13 d473076 Sep 19, 2019
Harness status	OK	OK	OK	OK
Check same-origin RTCCertificate serialization	PASS	PASS	FAIL message: promise_test: Unhandled rejection with value: object "TypeError: certificate2.getFingerprints is not a function"	PASS
Check cross-origin RTCCertificate serialization	PASS	PASS	FAIL message: promise_test: Unhandled rejection with value: object "TypeError: certificate2.getFingerprints is not a function"	PASS
Check cross-origin created RTCCertificate	FAIL message: assert_throws: function "() => { new RTCPeerConnection({certificates: [certificate2]}) }" did not throw	FAIL message: assert_throws: function "() => { new RTCPeerConnection({certificates: [certificate2]}) }" did not throw	FAIL message: assert_throws: function "() => { new RTCPeerConnection({certificates: [certificate2]}) }" did not throw	PASS

TPAC 2019 - WPT.kite



https://dashboard.cosmosoftware.io/wpt/aab8bf00bd40f392c06b88f8afd03a8b03691099/

TPAC 2019 - WPT.kite



Simulcast.kite

WPT can’t test p2p nor SFU based scenarios as needed by webrtc 1.0

=> KITE

=> tests

=> per SFUs

=> involve everyone => IETF (104, 105, …) Hackathon !

104 - Simulcast Testing with KITE

Specifically for this event, CoSMo created a gitHub repository with two automated kite interoperability tests. https://github.com/ManuCosmo/KITE-Hackathon

One test is a "typical" SFU test: KITE-Janus-Test is provided, which can be easily adapted to test any SFU, and should be the starting point for SFU developers wanting to automatically test against all the browser configuration CoSMo will provide for testing that week end:

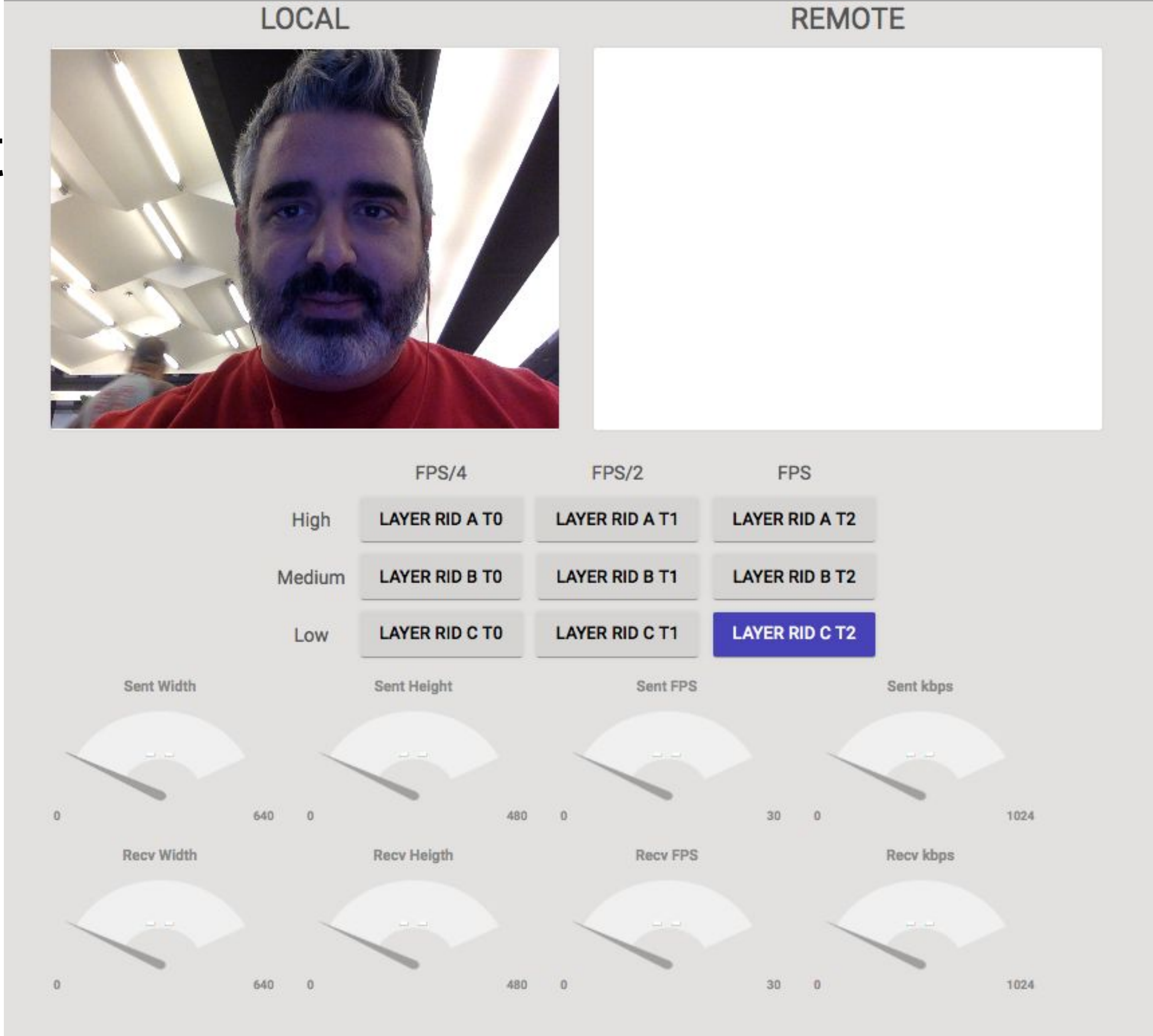
- Loopback setting to simplify testing configuration.
- SFU vendors to set-up and host SFus and loopback web-app
- KITE test to launch web browsers, connect to the web-app, run a scenario and report.
- At least one test written per bug found, to protect for future regression.

104 - Medooze test

KITE repo

Meedoze Playground

Choose a simulcast layer (High, Medium and Low) and a Temporal layer (not always present), and you can visually compare the sent and received Width, Height, FPS and kbps.



104 - Janus (VideoRoom/Echo) test

https://github.com/ManuCosmo/KITE-Hackathon

To test Janus, a test server is available:

- https://d10.conf.meetecho.com/ietf104/ (deployed locally in the IETF NOC)

The easier way to test simulcasting is to use the EchoTest plugin, which will allow you to choose which layers to send back. A couple of query strings are available to enable simulcast and force a specific codec:

- simulcast=true will enable old-style simulcasting (SDP munging for Chrome and Safari, rid-based for Firefox),
- simulcast2=true will enable the new rid-based simulcasting on Chrome M74 and M75;
- vcodec=X forces a specific codec (e.g., vcodec=h264).

Here's a couple more examples:

- https://d10.conf.meetecho.com/ietf104/echotest.html?simulcast=true
- https://d10.conf.meetecho.com/ietf104/echotest.html?simulcast2=true
- https://d10.conf.meetecho.com/ietf104/echotest.html?simulcast2=true&vcodec=h264

See annex E for a use case.

104 - Media Soup Test

https://github.com/ManuCosmo/KITE-Hackathon

To test Mediasoup, one can use https://v3demo.mediasoup.org

Some global variables in the browser console for debugging:

- PC1: the PeerConnection? that sends mic/webcam.
- PC2: the PeerConnection? that receives remote audio/video tracks using BUNDLE.
- CLIENT._micProducer and CLIENT._webcamProducer: mediasoup Producers, useful to check their rtpParameters that have been signaled to the SFU.
- sendSdps(): prints the local and remote SDP of the sending PeerConnection? (PC1).
- recvSdps(): prints the local and remote SDP of the sending PeerConnection? (PC2).

104 - Biggest WebRTC Hack session ever

19 registered

(13 listing ONLY WebRTC)

All main Browser vendors: MS, Google, Mozilla, Apple

Many SFU Tech Lead on-site: Meetecho, Medooze, …

Many SFU Tech Lead prepared tests: MediaSoup, ….

Status Report - Browser support card

			chrome 75 (canary)	chrome stable	Safari TP	Safari	firefox
Media Simulcast / ABR		h264 simulcast	yes - but bug pending	only via SDP mungling	yes	yes	no
		vp8 simulcast	yes	only via SDP mungling	yes	yes	yes
W3C Browsers APIs	RTCTransceiver	Have transceivers	yes - with unified plan	yes - unified plan	yes	yes	yes
	Stats API	Compliant Stats	yes - but bug pending	no	no	no	no
		Per layer Stats	no	no	no	no	no
	Simulcast enabling	Standard API + createOffer()	yes	no	no	no	yes - but old setParameter()
		legacy SDP mangling	yes	yes	yes	yes	no
IETF Internet protocols	Signalling (JSEP, SDP O/A)	Standard Unified Plan	yes	yes	yes	opt-in	yes
		Legacy Plan B	opt-in	opt-in	opt-in	yes	no
	Media Transport (RTP) simulcast features	rid	yes - if using addTransceiver	no	no	no	yes
		repairedId (RTX)	yes	no	no	no	no (no RTX at all)
		legacy ssrc in SDP	no - if using addTransceiver	yes	yes	yes	yes
	Bandwidth evaluation and congestion control	transport-wide-cc	yes	yes	yes	yes	no
		REMB	yes	yes	yes	yes	yes
	not all standards, but some IETF doc exists		vetted by henrik and harald		vetted by Youenn		vetted by nils

Status Report
SFU support table

		Open Source Media Servers								Commercial PaaS		
tested at IETF 104		Yes	Yes	No	Yes	No	No	No	No	No		
team member present at IETF 104		Yes	Yes	No	No	No	No	No	No	No		
Point of Contact		lorenzo	sergio	emil / boris / saul	inaki	?	?	micael gallego	Voluntas	gustavo garcia	?	?
Name		janus (VideoRoom plugin)	medooze	jitsi	mediasoup	INTEL	licode	openvidu / KMS	sora shiguredo	houseparty	tokbox	twilio
SDP Plan semantics	Plan B	yes	yes	Yes	yes		yes	yes	yes	yes		
	Unified Plan	yes	yes	One way only through convertion	yes		no	no	yes	no		
SDP O/A signaling	direct SDP signalling	yes	yes	no	ORTC: RTCRtpParameters on the wire, SDP O/A locally		yes	yes	yes	no		
	other	no	JSON on the wire, SDP locally	Jingle / COLIBRI on the wire, SDP locally			no	no	no	JSON on the wire SDP locally		
simulcast enabled via	SDP munging	yes	yes	yes	yes		yes	simulcast not supported	yes	yes		
	setParameter	yes	yes	no	yes		yes		no	no		
	addTransceiver	yes	yes	no	yes		no		no	no		
PC and stream handling	separate publisher and Subscriber PC	yes	no	no	yes.		no	yes	no	no		
	multiple PC	"master" => multiple,	no	no	no		it's flexible, depends on scalability: M multistream x N PC	?	no	no		
	single multi-stream PC	"unified-plan" => single multistream PC	yes	yes	sending (MID and RID), receiving (SSRCs), both with BUNDLE				yes	yes		
video codecs	VP8	yes + simulcast	yes + simulcast	Depends on configuration, but mainly VP8	yes + simulcast		yes + simulcast	yes	yes + simulcast	yes		
	H.264	yes + simulcast	yes + simulcast		yes + simulcast		yes	yes	yes + simulcast	no		
	VP9	yes + SVC	yes + SVC		yes		yes + SVC	no	no	no		
	rids supported	yes	yes	no	yes		yes	no	no	no		
	repairid supported	yes	yes	no	yes		no	no	yes	no		
	ssrc-less supported	yes (simulcast only)	yes	no	yes		no	no	no	no		
bandwidth congestion control	transport-wide-cc	yes - only receiver side	yes	yes	no		no	no	yes - only receiver side	yes		
	remb	yes	yes	yes	yes		yes	yes	yes			
bandwidth limitation on senders		REMB + SDP AS	no	simulcast layers dropping	proprietary client API		proprietary client API	Proprietary client API or settings	no	REMB		
mid rewritting		no	yes	no	no		no	no	yes	no		

									Commercial PaaS
tested at IETF 104		Yes	Yes	Yes	No	No	No	No	No
team member present at IETF 104		Yes	Yes	No	No	No	No	No	No
Point of Contact		sergio	lorenzo	inaki	Voluntas	emil / boris / saul	?	micael gallego	gustavo garcia
Name		medooze	janus (VideoRoom plugin)	mediasoup	sora shiguredo	jitsi	licode	openvidu / KMS	houseparty
SDP Plan semantics	Unified Plan	yes	yes	yes	yes	One way only through convertion	no	no	no
simulcast enabled via	addTransceiver	yes	yes	yes	no	no	no	simulcast not supported	no
PC and stream handling	single multi-stream PC	yes	sending (MID and RID), receiving (SSRCs), both with BUNDLE ("unified-plan" branch)	sending (MID and RID), receiving (SSRCs), both with BUNDLE	yes	yes	it's flexible, depends on scalability: M multistream x N PC	no	yes
video codecs	VP8	yes + simulcast	yes + simulcast	yes + simulcast	yes + simulcast	Depends on configuration, but mainly VP8	yes + simulcast	yes	yes
	H.264	yes + simulcast	yes + simulcast	yes + simulcast	yes + simulcast		yes	yes	no
IDs	rids supported	yes	yes	yes	yes	no	yes	no	no
	repairid supported	yes	yes	yes	yes	no	no	no	no
	ssrc-less supported	yes	yes (simulcast only)	yes	no	no	no	no	no

Annex: The Bandwidth allocation bug - case study

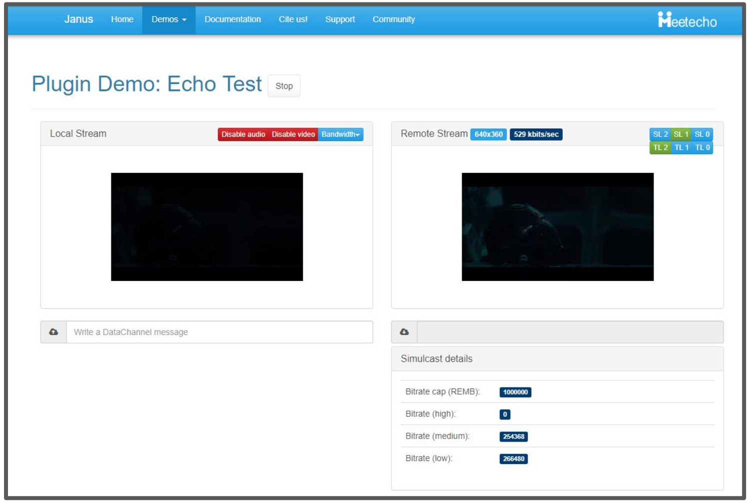
Set-up

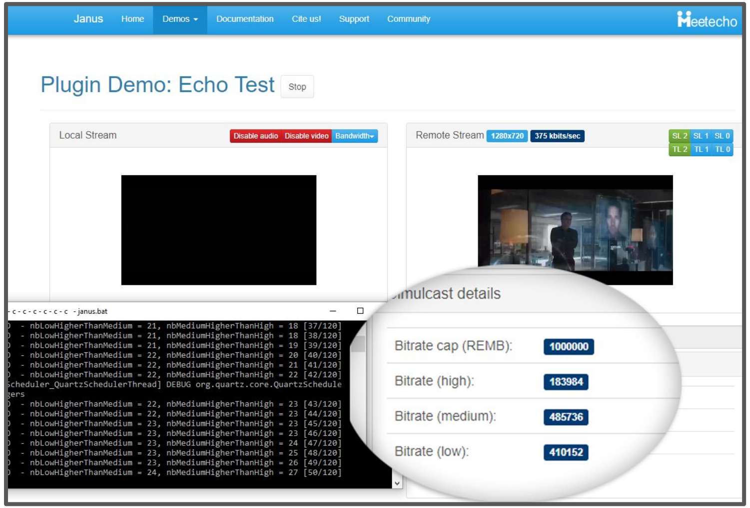
- Repo: https://github.com/ManuCosmo/KITE-Hackathon
- Config: https://github.com/ManuCosmo/KITE-Hackathon/blob/master/KITE-Simulcast-Test/configs/janus.simulcast.config.json
- App: Simulcast loop back page from Janus with VP8
  - https://d10.conf.meetecho.com/ietf104/echotest-cap.html?simulcast2=true&vcodec=vp8
- Browser(s) config(s):
  - Chrome m75 (canari) on Windows 10

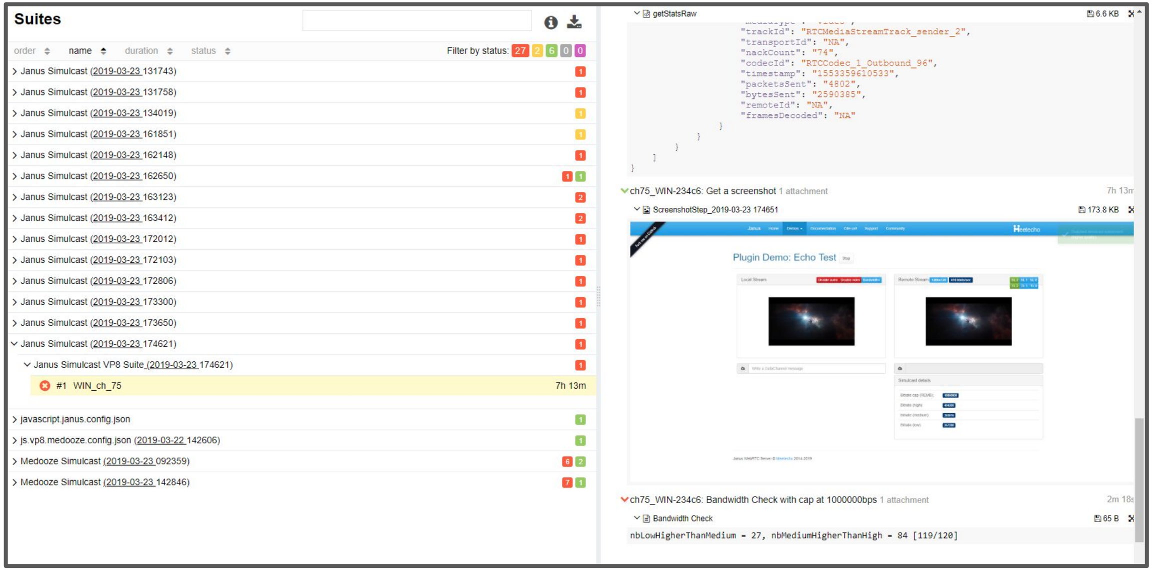
KITE Test: Steps & Checks

- open the page and checks that the call is established (video element display media)
- call getStats
- set the cap (REMB) to 1000000 bps
- every 1s for 120s, check the bitrates for low, medium and high simulcast profiles and:
- increment the `nbLowHigherThanMedium` if the low bitrate is higher than the medium bitrate
- increment the `nbMediumHigherThanHigh` if the low bitrate is higher than the medium bitrate

=> Fail the test if `nbLowHigherThanMedium` or `nbMediumHigherThanHigh` are higher than 0.







- Top left hand: the Echo Test demo page modified to illustrate simulcast and temporal layers selections, as well as bandwidth per simulcast layer.
- Bottom left hand: the app running in an instrumented browser thanks to KITE, you can see that the bandwidth is being allocated to the layers. You can also see it is not being consistent with what it should be (higher bandwidth allocation for higher resolution).
- Top right hand, the Dashboard view of things: list of tests, JSON output of the test, screenshots, and much more

IETF hack 105, 106,

105 - Montreal

=> Frozen Mountain

106 - Singapore

=> INTEL

Next steps to PR for WebRTC-PC (Bernard, 30 minutes)

W3C Requirements for PR

- Process: https://www.w3.org/2018/Process-20180201/#rec-pr
- Criteria:
  - *must* show adequate implementation experience except where an exception is approved by the Director,
  - *must* show that the document has received wide review,
  - *must* show that all issues raised during the Candidate Recommendation review period other than by Advisory Committee representatives acting in their formal AC representative role have been formally addressed,
  - *must* identify any substantive issues raised since the close of the Candidate Recommendation review period by parties other than Advisory Committee representatives acting in their formal AC representative role,
  - *may* have removed features identified in the Candidate Recommendation document as "at risk" without republishing the specification as a Candidate Recommendation.
- How can we remove the obstacles to reaching PR?

WebRTC Issues

- Good progress on closing specification issues since TPAC 2018.
- 31 Open Issues. Labels:
    - Editorial: 15
    - PR exists: 10
    - TPAC 2019: 5
    - Needs submitter action: 2
    - Question: 1
- New issue velocity: 8 in the last month
- Current fix velocity: 9 in the last month
- Merging existing PRs + addressing TPAC issues would leave less than 5 non-editorial issues remaining.

Simulcast: the Final (Implementation) Frontier

- Since TPAC 2018, Issues labeled “simulcast” have been resolved.
- However, implementation gaps and differences remain:
  - maxFramerate a “feature at risk” (no implementations)
  - Differences in simulcast SDP (SSRC support)
  - Support for RID/MID header extensions
- Potential solutions
  - Internet Draft: https://tools.ietf.org/html/draft-alvestrand-mmusic-simulcast-ssrc

WPT/WebRTC Status

- WPT status: https://wpt.fyi/webrtc
- More green, but still some yellow, orange and red.
  - 1+ implementations of all objects: RtpSender/Receiver, SctpTransport, DtlsTransport, IceTransport
- No WPT tests for simulcast
  - “Simulcast playground” only runs successfully in Firefox.
  - Potential for RID/MID loopback test
- Still some false negatives due to dependencies.

WPT Status: Orange is the new Pink

Path	Chrome 78 Linux 18.04 1719e88 Sep 13, 2019	Edge 78 Windows 10.0 1719e88 Sep 13, 2019	Firefox 71 Linux 18.04 1719e88 Sep 13, 2019	Safari 82 preview macOS 10.13 1719e88 Sep 13, 2019
RTCCertificate-postMessage.html	3 / 4	3 / 4	1 / 4	4 / 4
RTCCertificate.html	5 / 6	5 / 6	2 / 6	5 / 6
RTCConfiguration-bundlePolicy.html	16 / 16	16 / 16	8 / 16	16 / 16
RTCConfiguration-iceCandidatePoolSize.html	10 / 10	10 / 10	1 / 10	10 / 10
RTCConfiguration-iceServers.html	33 / 76	33 / 76	30 / 76	32 / 76
RTCConfiguration-iceTransportPolicy.html	14 / 17	14 / 17	11 / 17	17 / 17
RTCConfiguration-rtcpMuxPolicy.html	14 / 14	14 / 14	1 / 14	14 / 14
RTCDTMFSender-insertDTMF.https.html	7 / 8	6 / 8	8 / 8	1 / 8
RTCDTMFSender-ontonechange-long.https.html	2 / 2	2 / 2	2 / 2	1 / 2
RTCDTMFSender-ontonechange.https.html	12 / 14	12 / 14	14 / 14	1 / 14
RTCDataChannel-bufferedAmount.html	11 / 13	11 / 13	13 / 13	1 / 13
RTCDataChannel-id.html	5 / 5	5 / 5	5 / 5	1 / 5
RTCDataChannel-send-blob-order.html	1 / 2	1 / 2	1 / 2	1 / 2
RTCDataChannel-send.html	7 / 12	7 / 12	11 / 12	8 / 12
RTCDataChannelEvent-constructor.html	5 / 5	5 / 5	5 / 5	5 / 5
RTCDtlsTransport-getRemoteCertificates.html	2 / 2	2 / 2	1 / 2	1 / 2
RTCDtlsTransport-state.html	4 / 4	4 / 4	1 / 4	1 / 4
RTCError.html	24 / 24	24 / 24	1 / 24	1 / 24
RTCIceCandidate-constructor.html	19 / 19	19 / 19	17 / 19	8 / 19
RTCIceConnectionState-candidate-pair.https.html	2 / 2	1 / 2	2 / 2	2 / 2
RTCIceTransport-extension.https.html	29 / 30	30 / 30	1 / 30	1 / 30
RTCIceTransport.html	1 / 3	1 / 3	1 / 3	1 / 3
RTCPeerConnection-add-track-no-deadlock.https.html	2 / 2	2 / 2	2 / 2	2 / 2
RTCPeerConnection-addIceCandidate.html	17 / 30	17 / 30	30 / 30	11 / 30
RTCPeerConnection-addTrack.https.html	10 / 10	6 / 10	10 / 10	10 / 10
RTCPeerConnection-addTransceiver.https.html	11 / 13	9 / 13	11 / 13	11 / 13
RTCPeerConnection-canTrickleIceCandidates.html	1 / 4	1 / 4	4 / 4	1 / 4
RTCPeerConnection-connectionState.https.html	7 / 7	5 / 7	1 / 7	5 / 7
RTCPeerConnection-constructor.html	22 / 23	22 / 23	21 / 23	22 / 23
RTCPeerConnection-createAnswer.html	3 / 4	3 / 4	4 / 4	4 / 4
RTCPeerConnection-createDataChannel.html	37 / 42	37 / 42	35 / 42	26 / 42
RTCPeerConnection-createOffer.html	3 / 5	3 / 5	5 / 5	4 / 5
RTCPeerConnection-generateCertificate.html	7 / 9	7 / 9	9 / 9	9 / 9
RTCPeerConnection-getDefaultIceServers.html	1 / 2	1 / 2	1 / 2	1 / 2
RTCPeerConnection-getStats.https.html	9 / 14	8 / 14	8 / 14	6 / 14
RTCPeerConnection-getTransceivers.html	2 / 2	2 / 2	2 / 2	2 / 2
RTCPeerConnection-iceConnectionState-disconnected.https.html	2 / 2	1 / 2	2 / 2	2 / 2
RTCPeerConnection-iceConnectionState.https.html	12 / 12	7 / 12	8 / 12	7 / 12
RTCPeerConnection-iceGatheringState.html	3 / 4	3 / 4	3 / 4	3 / 4
RTCPeerConnection-mandatory-getStats.https.html	49 / 77	49 / 77	28 / 77	0 / 77
RTCPeerConnection-ondatachannel.html	5 / 9	5 / 9	7 / 9	4 / 9
RTCPeerConnection-onicecandidateerror.https.html	2 / 2	2 / 2	0 / 2	0 / 2

WPT Status (cont'd)

Path	Chrome 78 Linux 18.04 1719e88 Sep 13, 2019	Edge 78 Windows 10.0 1719e88 Sep 13, 2019	Firefox 71 Linux 18.04 1719e88 Sep 13, 2019	Safari 82 preview macOS 10.13 1719e88 Sep 13, 2019
RTCPeerConnection-onnegotiationneeded.html	14 / 14	14 / 14	13 / 14	5 / 14
RTCPeerConnection-onsignalingstatechanged.https.html	2 / 2	1 / 2	2 / 2	2 / 2
RTCPeerConnection-ontrack.https.html	6 / 6	6 / 6	6 / 6	6 / 6
RTCPeerConnection-remote-track-mute.https.html	1 / 6	1 / 6	3 / 6	0 / 6
RTCPeerConnection-removeTrack.https.html	13 / 14	5 / 14	14 / 14	13 / 14
RTCPeerConnection-restartIce-onnegotiationneeded.https.html	0 / 2	0 / 2	2 / 2	1 / 2
RTCPeerConnection-restartIce.https.html	11 / 14	11 / 14	13 / 14	1 / 14
RTCPeerConnection-setDescription-transceiver.html	3 / 7	3 / 7	7 / 7	4 / 7
RTCPeerConnection-setLocalDescription-answer.html	4 / 7	4 / 7	7 / 7	2 / 7
RTCPeerConnection-setLocalDescription-offer.html	6 / 8	6 / 8	8 / 8	3 / 8
RTCPeerConnection-setLocalDescription-pranswer.html	3 / 5	3 / 5	1 / 5	4 / 5
RTCPeerConnection-setLocalDescription-rollback.html	1 / 5	1 / 5	5 / 5	3 / 5
RTCPeerConnection-setLocalDescription.html	4 / 4	4 / 4	4 / 4	4 / 4
RTCPeerConnection-setRemoteDescription-answer.html	4 / 4	4 / 4	4 / 4	4 / 4
RTCPeerConnection-setRemoteDescription-nomsid.html	2 / 2	2 / 2	2 / 2	2 / 2
RTCPeerConnection-setRemoteDescription-offer.html	3 / 10	3 / 10	8 / 10	3 / 10
RTCPeerConnection-setRemoteDescription-pranswer.html	5 / 5	5 / 5	1 / 5	5 / 5
RTCPeerConnection-setRemoteDescription-replaceTrack.https.html	7 / 7	7 / 7	7 / 7	7 / 7
RTCPeerConnection-setRemoteDescription-rollback.html	1 / 6	1 / 6	6 / 6	2 / 6
RTCPeerConnection-setRemoteDescription-tracks.https.html	11 / 15	11 / 15	15 / 15	11 / 15
RTCPeerConnection-setRemoteDescription.html	5 / 6	5 / 6	6 / 6	6 / 6
RTCPeerConnection-track-stats.https.html	18 / 19	18 / 19	0 / 19	7 / 19
RTCPeerConnection-transceivers.https.html	45 / 45	45 / 45	44 / 45	42 / 45
RTCPeerConnectionIceEvent-constructor.html	7 / 9	7 / 9	7 / 9	9 / 9
RTCRtpParameters-codecs.html	7 / 7	7 / 7	1 / 7	1 / 7
RTCRtpParameters-degradationPreference.html	1 / 3	1 / 3	1 / 3	1 / 3
RTCRtpParameters-encodings.html	16 / 25	16 / 25	1 / 25	1 / 25
RTCRtpParameters-headerExtensions.html	2 / 2	2 / 2	1 / 2	1 / 2
RTCRtpParameters-rtcp.html	3 / 3	3 / 3	1 / 3	1 / 3
RTCRtpParameters-transactionId.html	6 / 6	6 / 6	1 / 6	1 / 6
RTCRtpReceiver-getCapabilities.html	4 / 4	4 / 4	1 / 4	4 / 4
RTCRtpReceiver-getContributingSources.https.html	3 / 3	1 / 3	3 / 3	3 / 3
RTCRtpReceiver-getParameters.html	3 / 4	3 / 4	1 / 4	1 / 4
RTCRtpReceiver-getStats.https.html	1 / 3	1 / 3	1 / 3	1 / 3
RTCRtpReceiver-getSynchronizationSources.https.html	0 / 15	1 / 15	5 / 15	10 / 15
RTCRtpSender-getCapabilities.html	4 / 4	4 / 4	1 / 4	4 / 4
RTCRtpSender-getStats.https.html	1 / 3	1 / 3	1 / 3	1 / 3
RTCRtpSender-replaceTrack.https.html	8 / 10	8 / 10	10 / 10	10 / 10
RTCRtpSender-setParameters.html	1 / 2	1 / 2	2 / 2	2 / 2
RTCRtpSender-setStreams.https.html	6 / 6	6 / 6	1 / 6	1 / 6
RTCRtpSender-transport.https.html	9 / 9	1 / 9	1 / 9	1 / 9
RTCRtpTransceiver-direction.html	4 / 4	4 / 4	4 / 4	4 / 4
RTCRtpTransceiver-setCodecPreferences.html	14 / 14	14 / 14	1 / 14	1 / 14

WPT Status (cont'd)

Path	Chrome 78 Linux 18.04 1719e88 Sep 13, 2019	Edge 78 Windows 10.0 1719e88 Sep 13, 2019	Firefox 71 Linux 18.04 1719e88 Sep 13, 2019	Safari 82 preview macOS 10.13 1719e88 Sep 13, 2019
RTCRtpTransceiver-direction.html	4 / 4	4 / 4	4 / 4	4 / 4
RTCRtpTransceiver-setCodecPreferences.html	14 / 14	14 / 14	1 / 14	1 / 14
RTCRtpTransceiver-stop.html	1 / 5	1 / 5	4 / 5	5 / 5
RTCRtpTransceiver.https.html	20 / 39	19 / 39	39 / 39	20 / 39
RTCSctpTransport-constructor.html	5 / 5	5 / 5	1 / 5	1 / 5
RTCSctpTransport-events.html	3 / 3	3 / 3	0 / 3	0 / 3
RTCSctpTransport-maxChannels.html	3 / 3	3 / 3	0 / 3	0 / 3
RTCSctpTransport-maxMessageSize.html	6 / 6	6 / 6	1 / 6	1 / 6
RTCTrackEvent-constructor.html	7 / 8	7 / 8	8 / 8	8 / 8
RTCTrackEvent-fire.html	8 / 10	8 / 10	10 / 10	5 / 10
datachannel-emptystring.html	1 / 2	1 / 2	1 / 2	1 / 2
getstats.html	2 / 2	2 / 2	1 / 2	2 / 2
historical.html	9 / 18	9 / 18	10 / 18	18 / 18
idlharness.https.window.html	470 / 509	470 / 509	318 / 509	322 / 509
legacy/	28 / 28	26 / 28	27 / 28	9 / 28
no-media-call.html	2 / 2	2 / 2	2 / 2	2 / 2
promises-call.html	2 / 2	2 / 2	2 / 2	2 / 2
protocol/	32 / 32	32 / 32	20 / 32	20 / 32
simplecall-no-ssrcs.https.html	2 / 2	2 / 2	2 / 2	2 / 2
simplecall.https.html	2 / 2	2 / 2	2 / 2	2 / 2

Confluence Status

- Web-platform-tests dashboard “does not contain useful metrics for evaluation or comparison of web platform features”
- Web confluence project:
    - Looks at properties and methods exposed by browsers:
    - https://web-confluence.appspot.com/#!/
    - Caveat: no guarantee that a widely-supported API is interoperable in its details, or will remain part of the web platform.
    - Tool that extracts data from the confluence tracker: https://dontcallmedom.github.io/webrtc-impl-tracker/?webrtc
- Overall status:
    - Fewer features with no implementations, compared with TPAC 2018.
    - Improved object model support
        - TPAC 2018: Current edge the only browser implementing DtlsTransport and IceTransport.

Confluence Status (cont'd)

Interface	Member		Chrome	Edge	Firefox	Safari
RTCPeerConnection	createOffer	4/4	40+	15+	45+	11+
	createAnswer	4/4	40+	15+	45+	11+
	setLocalDescription	4/4	40+	15+	45+	11+
	localDescription	4/4	43+	15+	45+	11+
	currentLocalDescription	3/4	70+		57+	11+
	pendingLocalDescription	3/4	70+		57+	11+
	setRemoteDescription	4/4	40+	15+	45+	11+
	remoteDescription	4/4	43+	15+	45+	11+
	currentRemoteDescription	3/4	70+		57+	11+
	pendingRemoteDescription	3/4	70+		57+	11+
	addIceCandidate	4/4	40+	15+	45+	11+
	signalingState	4/4	43+	15+	45+	11+
	iceGatheringState	4/4	43+	15+	45+	11+
	iceConnectionState	4/4	43+	15+	45+	11+
	connectionState	2/4	72+			11+
	canTrickleIceCandidates	2/4		15+	47+	
	getDefaultIceServers	?/4				
	getConfiguration	4/4	70+	15+	45+	11+
	setConfiguration	2/4	58+			11+
	close	4/4	40+	15+	45+	11+
	onnegotiationneeded	4/4	43+	15+	45+	11+
	onicecandidate	4/4	43+	15+	45+	11+
	onicecandidateerror	?/4				
	onsignalingstatechange	4/4	43+	15+	45+	11+
	oniceconnectionstatechange	4/4	43+	15+	45+	11+
	onicegatheringstatechange	4/4	59+	15+	53+	11+
	onconnectionstatechange	2/4	72+			11+
	generateCertificate	3/4	49+		45+	12+
	getSenders	3/4	64+		45+	11+
	getReceivers	3/4	59+		45+	11+
	getTransceivers	3/4	69+		59+	11+
	addTrack	3/4	64+		45+	11+
	removeTrack	3/4	64+		45+	11+
	addTransceiver	3/4	69+		59+	11+
	ontrack	3/4	64+		46+	11+
	sctp	1/4	76+			
	createDataChannel	3/4	40+		45+	11+
	ondatachannel	3/4	43+		45+	11+
	getStats	4/4	40+	15+	45+	11+
	onstatsended	?/4				

Confluence Status (cont'd)

RTCSessionDescription	type	4/4	43+	15+	45+	11+
	sdp	4/4	43+	15+	45+	11+
	toJSON	4/4	43+	15+	45+	11+
RTCIceCandidate	candidate	4/4	43+	15+	45+	11+
	sdpMid	4/4	43+	15+	45+	11+
	sdpMLineIndex	4/4	43+	15+	45+	11+
	foundation	1/4	74+			
	component	1/4	74+			
	priority	1/4	74+			
	address	1/4	74+			
	protocol	1/4	74+			
	port	1/4	74+			
	type	1/4	74+			
	tcpType	1/4	74+			
	relatedAddress	1/4	74+			
	relatedPort	1/4	74+			
	usernameFragment	2/4	74+		67+	
	toJSON	4/4	43+	15+	45+	11+
RTCPeerConnectionIceEvent	candidate	4/4	56+	15+	45+	12+
	url	1/4				12+
RTCPeerConnectionIceErrorEvent	hostCandidate	?/4				
	url	?/4				
	errorCode	?/4				
	errorText	?/4				
RTCCertificate	expires	3/4	49+		45+	12+
	getSupportedAlgorithms	?/4				
	getFingerprints	2/4	61+			12+
RTCRtpSender	track	4/4	64+	13+	45+	11+
	transport	2/4	75+	13+		
	rtcpTransport	2/4	75+	13+		
	getCapabilities	3/4	69+	13+		12+
	setParameters	3/4	68+		46+	12+
	getParameters	3/4	68+		46+	11+
	replaceTrack	3/4	65+		45+	11+
	setStreams	1/4	76+			
	getStats	3/4	67+		55+	12+
	dtmf	2/4	66+		52+	

Confluence Status (cont’d)

RTCRtpReceiver	track	4/4	59+	13+	45+	11+
	transport	2/4	75+	13+		
	rtcpTransport	2/4	75+	13+		
	getCapabilities	3/4	69+	13+		12+
	getParameters	2/4	73+			11+
	getContributingSources	4/4	59+	13+	59+	12+
	getSynchronizationSources	3/4	73+		59+	12+
	getStats	3/4	67+		55+	12+
RTCRtpTransceiver	mid	3/4	69+		59+	11+
	sender	3/4	69+		59+	11+
	receiver	3/4	69+		59+	11+
	stopped	3/4	69+		59+	11+
	direction	3/4	69+		59+	11+
	currentDirection	2/4	69+		59+	
	stop	2/4			59+	11+
	setCodecPreferences	1/4	76+			
RTCDtlsTransport	iceTransport	1/4	75+			
	state	2/4	75+	13+		
	getRemoteCertificates	2/4	76+	13+		
	onstatechange	1/4	75+			
	onerror	2/4	75+	13+		
RTCIceTransport	role	2/4	75+	13+		
	component	1/4		13+		
	state	3/4	75+	13+		11+
	gatheringState	2/4	75+			11+
	getLocalCandidates	1/4	75+			
	getRemoteCandidates	2/4	75+	13+		
	getSelectedCandidatePair	1/4	75+			
	getLocalParameters	1/4	75+			
	getRemoteParameters	2/4	75+	13+		
	onstatechange	1/4	75+			
	ongatheringstatechange	1/4	75+			
	onselectedcandidatepairchange	1/4	75+			
RTCTrackEvent	receiver	3/4	64+		46+	11+
	track	3/4	64+		46+	11+
	streams	3/4	64+		46+	11+
	transceiver	3/4	69+		59+	11+
RTCSctpTransport	transport	1/4	76+			
	state	1/4	76+			
	maxMessageSize	1/4	76+			
	maxChannels	1/4	76+			
	onstatechange	1/4	76+			

Confluence Status (cont'd)

RTCDataChannel	label	3/4	56+		60+	11+
	ordered	3/4	56+		60+	11+
	maxPacketLifeTime	2/4			62+	11+
	maxRetransmits	3/4	56+		62+	11+
	protocol	3/4	56+		60+	11+
	negotiated	3/4	56+		68+	11+
	id	3/4	56+		60+	11+
	priority	?/4				
	readyState	3/4	56+		60+	11+
	bufferedAmount	3/4	56+		60+	11+
	bufferedAmountLowThreshold	3/4	56+		60+	11+
	onopen	3/4	56+		60+	11+
	onbufferedamountlow	3/4	56+		60+	11+
	onerror	3/4	56+		60+	11+
	onclose	3/4	56+		60+	11+
	close	3/4	56+		60+	11+
	onmessage	3/4	56+		60+	11+
	binaryType	3/4	56+		60+	11+
	send	3/4	56+		60+	11+
RTCDataChannelEvent	channel	3/4	56+		45+	11+
RTCDTMFSender	insertDTMF	2/4	66+		52+	
	ontonechange	2/4	66+		52+	
	canInsertDTMF	1/4	66+			
	toneBuffer	2/4	66+		52+	
RTCDTMFToneChangeEvent	tone	3/4	66+	13+	52+	
RTCStatsEvent	report	?/4				
RTCError	errorDetail	1/4	74+			
	sdpLineNumber	1/4	74+			
	httpRequestStatusCode	1/4	74+			
	sctpCauseCode	1/4	74+			
	receivedAlert	1/4	74+			
	sentAlert	1/4	74+			
RTCErrorEvent	error	1/4	74+			

Simulcast Playground

- Single browser tests for simulcast, written by Fippo.
  - Enables testing of simulcast operation without a conferencing server.
  - Can determine if encoding attributes (such as maxBitrate, maxFramerate, active) is having the desired effect.
    - Assuming the specification defines the “desired effect”!
  - WebRTC Hacks article: https://webrtchacks.com/a-playground-for-simulcast-without-an-sfu/
- Repo: https://github.com/fippo/simulcast-playground
  - Separate page for each browser, since simulcast still isn’t interoperable enough (yet)
- Could be extended to allow tests between two browsers without a conferencing server.
  - Progress still needed to make this feasible.

WG decisions to be made

- What do we do to demonstrate simulcast interop?
  - How do we test simulcast interoperability?
- Handling of “features at risk”
  - Session this afternoon

Break
10:00 AM - 10:30 AM

WebRTC-PC (Bernard, 30 minutes)

Webrtc-PC (bernard)

- **Issues**
  - **Issue 1764: reducing audio packet rate while track is disabled (bernard)**
  - **Issue 2121/PR 2188: Constrainable properties on remote tracks are under-specified (Henrik)**
  - **Issue 2230: RTCPeerConnectionIceErrorEvent: host candidate clarification (Youenn)**
  - **Issue 2257: Consider making RTCCertificate throw when serialized when *forStorage* is false (Jan-Ivar)**

Issue 1764: reducing audio packet rate while track is disabled (bernard)

- **Section 5.2 says:**
  - If `track` is ended, or if the track's output is disabled, i.e. the track is disabled and/or muted, the `RTCRtpSender` *MUST* send silence (audio), black frames (video) or a zero-information-content equivalent. In the case of video, the `RTCRtpSender` *SHOULD* send one black frame per second. If `track` is null then the `RTCRtpSender` does not send.
- **In the case of audio, why isn't there equivalent guidance on packet rate reduction?**
  - Fippo: Shouldn't have to use replaceTrack(null) to reduce the audio bitrate.
  - Harald: Would like to encourage use of track.enabled = false to mute.
- **Proposed solution (Nils)**
  - Video keeps state on the screen, so it is better to send a black frame than to send nothing and have a frozen picture. This is not the case for audio.
  - Recommendation: send nothing regardless of CN or DTX (since they may not be negotiated).

Issue 2121/PR 2188: Constrainable properties on remote tracks are under-specified (Henrik)

Context: It's not clear which constraints apply to remote tracks. Each spec needs to explicitly say which constraints they support - it being listed in getUserMedia() is not sufficient.

Problem: Prior to clarifications about GUM constraints not being inherited by webrtc-pc, Chrome implemented the following constraints for remote tracks:

- width, height: Get current resolution or apply downscaling (limit resolution).
- frameRate: Get current frame rate or apply downsampling (limit frame rate).
- aspectRatio: Get current aspect ratio or apply cropping/downscaling.

At the April 11 Virtual Interim, a decision was made to create a PR defining which constraints make sense for remote tracks; the above was proposed but the PR never merged.

Proposal A:

- Specify the above behavior in webrtc-pc and clarify that other constraints are not applicable (PR 2188).

Proposal B:

- Specify that no constraints are applicable, making Chrome's implementation non-compliant.

Proposal C:

- Define constraints for getSettings(), but throw OverconstrainedError on applyConstraints()

Issue 2230: RTCPeerConnectionIceErrorEvent: host candidate clarification (Youenn)

- RTCPeerConnectionIceErrorEvent exposes host candidate address
  - Same filtering should be applied as elsewhere
  - Current rule is to return 0.0.0.0:0 if filtering is on
- Question: should mDNS names be placed here and in which cases?
- Answer: no need for mDNS names, no change needed
  - Mode 1 issue
- Potential improvements
  - Strengthen the 'relay' case: filter if the IP address is not already exposed to the web page
  - Return null instead of 0.0.0.0:0, simpler and more straightforward
  - Split IP address and port as two different fields

Issue 2257/PR 2297: Consider making RTCCertificate throw when serialized when *forStorage* is false (Jan-Ivar)

- **Based on feedback from annevk, allowing the passing of certificates around between processes might open them up to Spectre attacks.**
- **Original intent here seemed to be to allow storage.**

- **PR 2297 restricts RTCCertificates to page and process they're born in:**

`RTCCertificate` objects are serializable objects [HTML]. Their serialization steps, given *value*, *serialized*, and a *forStorage* boolean are:

1. If *forStorage* is `false`, throw a `SecurityError`.

- **And it removes various [[Origin]] internal slot.**
- **Is this OK, or are any web sites relying on being able to postMessage RTCCertificates?**

Capture (Jan-Ivar, 60 minutes)

Screen capture

- Issues
  - Issue 60: Define Tab Capture (Harald)
  - Is this ready for CR?

Issue 60: Define Tab Capture (Harald)

- Issue seems editorial
- Spec allows you to capture anything the UA wants to call a “display surface”
- Definition of “browser display surface”:
    - A browser display surface is the rendered form of a single document. This is not strictly limited to HTML [HTML5] documents, though the discussion in this document will address some specific concerns with the capture of HTML.
- This seems to describe tab capture, but what happens if the active document in a tab changes? Text doesn’t seem to say.
- Suggested text:
    - "The UA may choose to display the current document in a browser window, continuing to cast the current document into the same media stream track when the current document changes. This is commonly called tab capture."
- No normative changes needed.

Media Capture & Streams (Jan-Ivar)

- Issues
  - Issue 554: Specify a way for webdriver to add/remove/setup web capture devices (Youennf)
  - Issue 565: Should a devicechange event be fired when the list of devices remains the same? (Youennf)
  - Issue 584/PR 623: Resize mode (crop-and-scale) is under-specified (Henrik)
  - Issue 608: Is enumerateDevices list order significant? (Youennf)

Issue 554: Specify a way for webdriver to add/remove/setup web capture devices (Youennf)

- No way to automate testing of devicechange event
  - For Web Sites
  - For WPT testing
- WebKit has automated testing for these through an internal web API to add/remove capture devices
  - Support of setting some capacities to differentiate between the devices (sample rate, min/max video size, facing mode)
  - Other browsers might have similar support?
- Possibility for an extension spec to define a WebDriver API?
- If so, what would be the scope?
  - Control on a few capture properties
  - Media content customization is not a pre-requisite

Issue 565: Should a devicechange event be fired when the list of devices remains the same? (Youennf 1/2)

- Example:
  - User is reading a newspaper article
    - Newspaper article registers devicechange event
  - While reading an article, he gots a WebRTC call
    - User switches to the WebRTC tab
    - User plugs-in a camera
    - User does the call
    - User unplugs the camera
    - User goes back to reading the newspaper article
    - devicechange event is fired on the newspaper article
- Firing devicechange event in that case is:
  - Useless: devices did not change for the newspaper article
  - Potentially harmful: it leaks some information of user actions

Issue 565: Should a devicechange event be fired when the list of devices remains the same? (Youennf 2/2)

- **Spec says:**
  - If a browsing context later comes to meet the device information can be exposed check criteria (e.g. gains focus), the User Agent *MUST* execute the steps at that time.
  - The User Agent *MAY* combine firing multiple events into firing one event when several events are due or when multiple devices are added or removed at the same time, e.g. a camera with a microphone.
- **Proposal: Allow browsers to not fire devicechange event if the list of device is actually the same**
  - **Since last time enumerateDevices was called or devicechange event handler was set**
  - **When combining multiple events into one event lead to the same list of devices**

Issue 584/PR 623: Resize mode (crop-and-scale) is under-specified (Henrik)

Problem: VideoResizeModeEnum's "crop-and-scale" is underspecified:

- ***"This resolution is downscaled and/or cropped from a higher camera resolution by the user agent."***

We don't want to allow stretching or black borders. The final media should be a subset of the input.

Proposal:

- **Add: "*The media MUST NOT be upscaled, stretched or have fake data created that did not occur in the input source.*"**

Issue 608: Is enumerateDevices list order significant? (Youennf)

- Some websites want to use the 'default' devices
  - Some 'default' devices have better integration with the OS
- Current behavior (tested on MacOS only)
  - All browsers list the system default audio input device first in enumerateDevices
  - getUserMedia({audio: true}) uses the system default audio input device
  - Chrome fires a devicechange event when the system default audio input device is changed
- Can and should we specify this behavior?

Media Stream Recording

- **Issues**
  - **Issue 139: Does recording of remote A/V streams always imply re-encoding? (Henrik)**
  - **Issue 167/PR 186: Add replaceStream method to MediaRecorder (Henrik)**
    - **PR 187: Alternative B: replaceTrack method on MediaRecorder (jib)**
  -

Issue 139: Does recording of remote A/V streams always imply re-encoding? (Henrik)

Problem: When recording a remote track from WebRTC, the bitstream gets decoded and then re-encoded per MediaRecorder's MediaRecorderOptions.

Question 1: Can we avoid costly re-encoding?
Question 2: Can we get a dump of the raw encoded stream?

Proposal A:

- Specify that if the codec in mimeType is missing we must not re-encode remote tracks.
  - Example: "`video/webm;codecs=vp9,opus`" means re-encode both audio and video; "`video/webm;codecs=opus`" means don't re-encode video, but re-encode audio.
- Answer to both questions = YES!

Proposal B:

- Add a note that an implementation is allowed to avoid re-encoding of remote tracks, but don't mandate it.
- Answer to both questions = implementation-specific.

Issue 167/PR 186: Add replaceSteam method to MediaRecorder (Henrik)

Problem: MediaRecorder does not allow you to change the source of recording while recording. Modifying the set of live tracks of the recorded stream triggers failure.

We want to be able to do this seamlessly (no glitches!).

Proposal:

- `Promise<void> mediaRecorder.replaceStream(MediaStream newStream)`
  - **Require the same number of audio and video tracks in newStream.**
  - **Seamlessly start to gather frames from those tracks instead.**

[Issue 167](#)/[PR 187](#): Add replaceTrack method to MediaStream (Henrik/Jan-Ivar)

Proposal B (tentative):

- `void mediaRecorder.replaceTrack(MediaStreamTrack track, MediaStreakTrack withTrack)`
    - **No ordering issues.**
    - **Not possible to accidentally add/remove tracks to/from recording**
    - **No promise needed, since all checking can be done synchronously**
    - **Throws `NotFoundError` if `track` not found**
    - **Throws `InvalidModificationError` if `track.kind` doesn't match**
    - **Throws `SecurityError` if either isolation properties disallow recording**

In addition, we propose to no longer stop the recorder if a stream's track-set changes, because it no longer makes sense to react to those. Instead, we read `stream` in start() only (copy the track-set), and only stop once all tracks end.

Finally, we make stream attribute writeable. Useful for rec.stop(); rec.start();

Lunch
12:00 PM - 1:00 PM

Afternoon Agenda for Thursday, September 19

1:00 PM - 2:00 PM WebRTC-Stats (Varun & Henrik)
Reference: https://w3c.github.io/webrtc-stats/

2:00 PM - 2:30 PM Content-hints (Harald)
Content-Hints: https://w3c.github.io/mst-content-hint/

2:30 PM - 3:00 PM Other current specifications (Harald and Peter)
DSCP API: https://www.w3.org/TR/webrtc-dscp/
WebRTC-ICE: https://github.com/w3c/webrtc-ice

3:00 PM - 3:30 PM Break

3:30 PM - 4:30 PM WebRTC-PC "Features at risk" (Jan-Ivar)

4:30 PM - 5:30 PM WEBRTC WG Developer Feedback Session (Bernard)

WebRTC-Stats
(Varun & Henrik, 60 minutes)

webrtc-stats (Varun & Henrik) - 1/2

- “State of the Stats” Report (Henrik)

- Issue 361: Should track also contain RTCRtpReceiver stats? /
  Issue 452: Add receiving "track" stats to the obsolete section (Henrik)
- Issue 478: Should we have transceiver stats? /
  Issue 396: RTC[Audio/Video]SenderStats should have mid (Henrik/Varun)
- Issue 479: Should we move track/sender/receiver to the obsolete section? (Henrik)
- Issue 480: What should we do about onstatsended? /
  Issue 472: Issues with replaceTrack, will statsended fire or give me what I want (Henrik)
- Issue 365: Remove track/stream stats in favor of RTCMediaHandlerStats.mid /
  Issue 470: RTCMediaStreamStats can be made obsolete (Henrik)
- Issue 437: RTCStats.id should not be "predictable" (Henrik)
- Issue 395/PR 482: Add RTCOutboundRtpStreamStats.rid (Varun/Henrik)
- Issue 398: Add RTCEncodingParameterStats (Henrik)
- Issue 401: Add SVC stats in RTC[In/Out]boundRtpStreamStats (Henrik)

…

webrtc-stats (Varun & Henrik) - 2/2

…

- Issue 443: Detecting Video Playback glitches (Henrik)
- Issue 440: Clarify qualityLimitationReason when limited by multiple reasons (Henrik)
- Issue 391: audioLevel can be removed from "track"|"receiver"|"sender" stats (Henrik)
- Issue 448: Audio samples and channels (Henrik)
- Issue 358: identifying the ice generation of a candidate pair (Henrik)
- Issue 376: [DataChannels] Use RTT from sctp in the stats (Henrik)
- Issue 377: [DataChannels] Expose bandwidth or congestion window from the SCTP lib (Henrik)

Webrtc-stats implementation status

	Chrome - 77.0.3865.70	Firefox - 71.0a1	Safari - 13.1
codec	6/11	0/11	6/11
inbound-rtp	18/37	14/37	16/37
outbound-rtp	22/33	13/33	14/33
remote-inbound-rtp	11/23	10/23	0/23
remote-outbound-rtp	0/14	8/14	0/14
media-source	0/9	8/9	0/9
csrc	0/7	0/7	0/7
peer-connection	5/7	0/7	5/7
stream	5/5	0/5	0/5
track	17/23	0/23	10/23

	Chrome - 77.0.3865.70	Firefox - 71.0a1	Safari - 13.1
sender	0/23	0/23	0/23
receiver	0/30	0/30	0/30
transport	9/17	0/17	8/17
candidate-pair	18/30	12/30	17/30
local-candidate	11/13	9/13	10/13
remote-candidate	9/13	8/13	10/13
certificate	6/7	0/7	6/7
ice-server	0/10	0/10	0/10

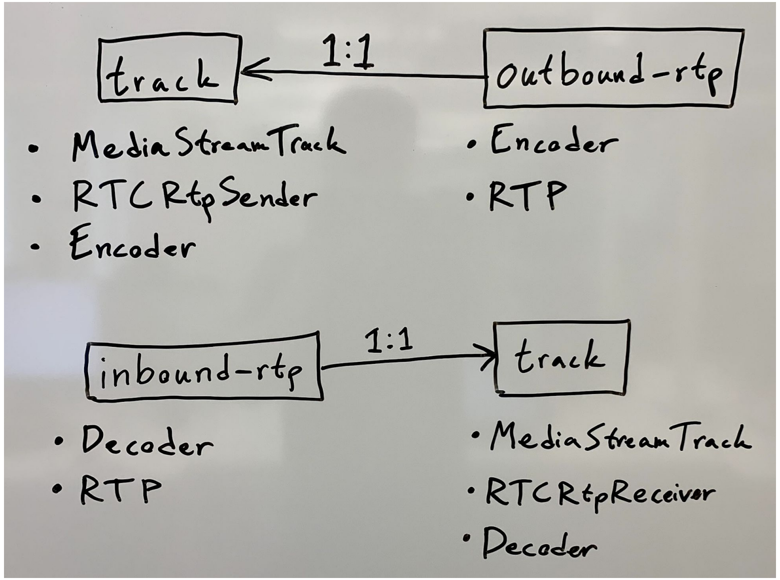
Validate the values from stats

- Not much progress, we should really look into this.

“State of the Stats” Report (Henrik) - 1/5

In the beginning…

- **We had “track” stats, which supposedly represented MediaStreamTrack, but were really a mix of track, encoder, decoder, sender and receiver stats.**
- **We also had “outbound-rtp” and “inbound-rtp”, representing a mix of RTP, encoder and decoder stats.**

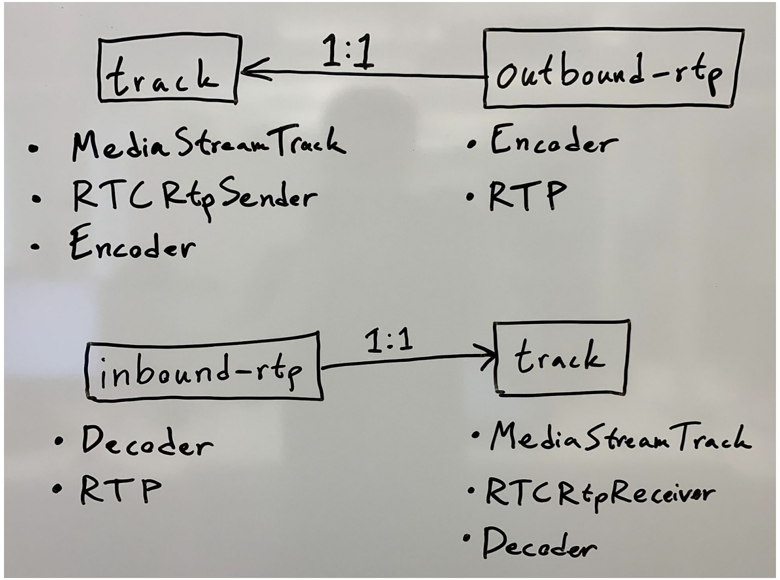


One “track” stats per sender or receiver, with one “outbound-rtp” or “inbound-rtp”

“State of the Stats” Report (Henrik) - 2/5

In the beginning…

- **We had “track” stats, which supposedly represented MediaStreamTrack, but were really a mix of track, encoder, decoder, sender and receiver stats.**
- **We also had “outbound-rtp” and “inbound-rtp”, representing a mix of RTP, encoder and decoder stats.**



One “track” stats per sender or receiver, with one “outbound-rtp” or “inbound-rtp”

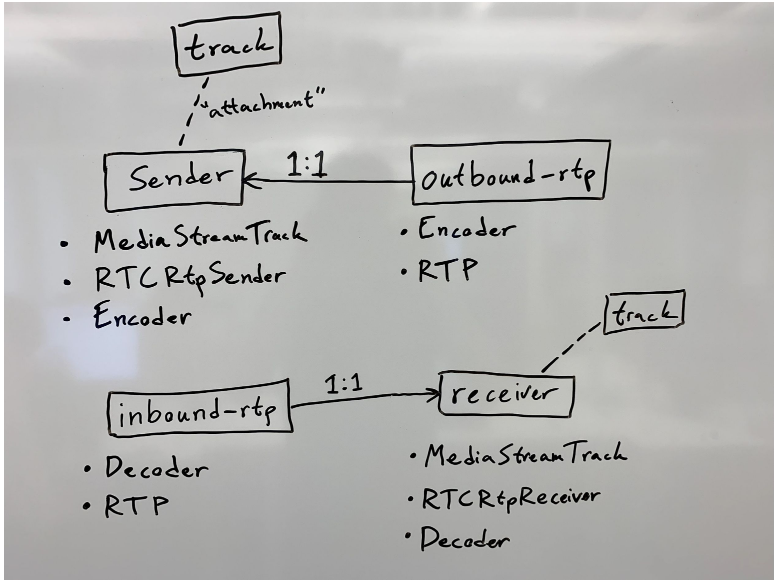
The transceiver model does not consist of “tracks”; it consists of sender-receiver pairs.

What happens to frame counters if I do replaceTrack()? If I have multiple senders sending the same track?

Need “sender” and “receiver” stats!
TPAC 2017: [#231]

“State of the Stats” Report (Henrik) - 3/5

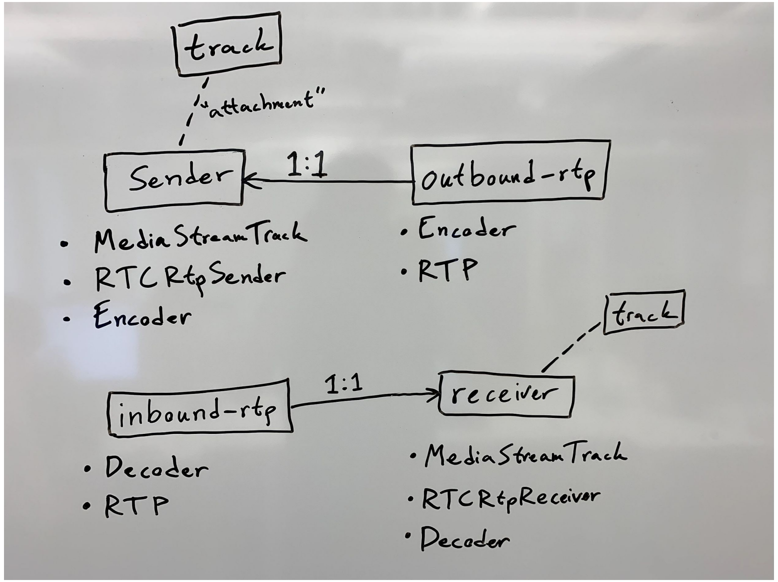
- **“sender” and “receiver” stats replace “track”**
- **“track” becomes a child dictionary of “sender” and “receiver”, which is essentially a copy of the “sender” and “receiver” stats.**
  - **But when replaceTrack() happens, the “sender” frame counters keep increasing, but the “track” stats object is replaced by a new object whose frame counters start at zero.**



PROBLEM SOLVED!

“State of the Stats” Report (Henrik) - 4/5

- “sender” and “receiver” stats replace “track”
- “track” becomes a child dictionary of “sender” and “receiver”, which is essentially a copy of the “sender” and “receiver” stats.
    - But when replaceTrack() happens, the “sender” frame counters keep increasing, but the “track” stats object is replaced by a new object whose frame counters start at zero.



PROBLEM SOLVED!

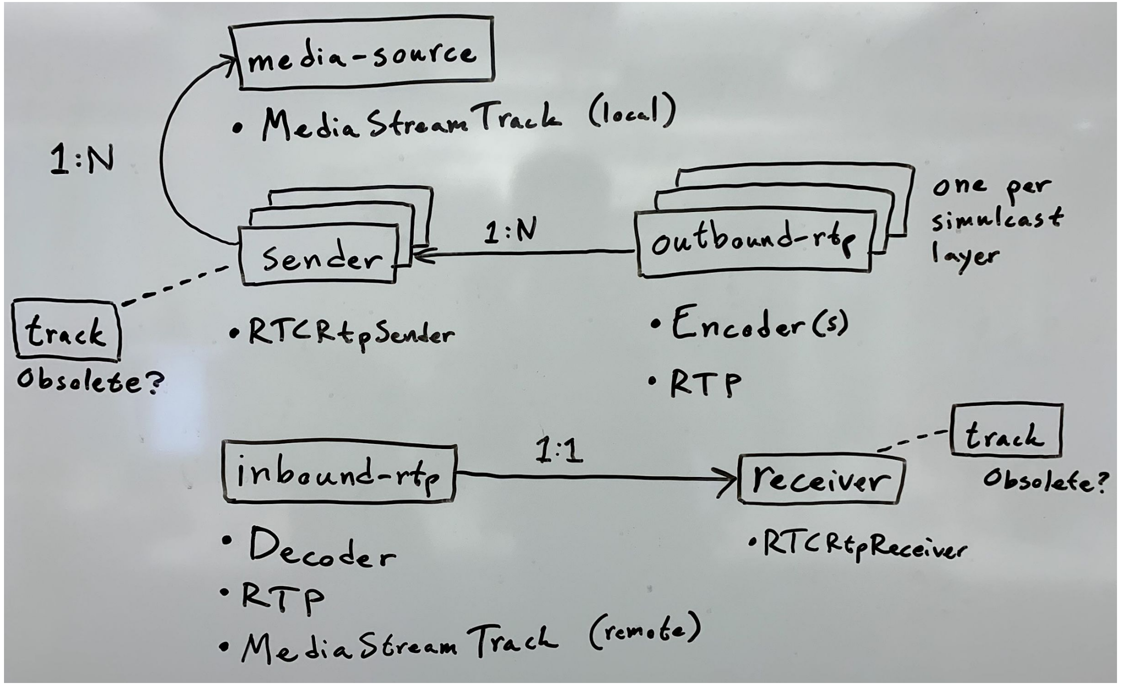
… but “sender” and “receiver” has not yet been implemented.

And they’re still a mix of stats: track, sender, encoder all in the same place

And what about simulcast?!

“State of the Stats” Report (Henrik) - 5/5

In [#463] and [#466] “track”, “sender” and “receiver” stats are moved to “outbound-rtp”, “inbound-rtp” and “media-source”.



media-source
- Captured resolution, etc.

sender/receiver
- Identifies the sender/receiver

outbound-rtp
- Per-simulcast metrics
  - Encoded resolution, etc.
  - RTP

inbound-rtp
- Decoded resolution, etc.
- RTP
- “MediaStreamTrack (remote)” is basically decoder output.

Issue 361: Should track also contain RTCRtpReceiver stats? / Issue 452: Add receiving "track" stats to the obsolete section (Henrik)

The spec neglects receiving “track” stats. These are currently shipped in Chrome. Note:

- Receiving “track” stats are equivalent to “receiver” stats, since the track attachment of a receiver can’t change, but “receiver” stats have not been shipped.

Proposal:

- Define receiving “track” stats as receiving track attachments that are equivalent to “receiver” stats.
- But place this definition in the Obsolete section, reflecting the fact that they are not needed if you have implemented “receiver” stats.

Issue 478: Should we have transceiver stats? / Issue 396: RTC[Audio/Video]SenderStats should have mid (Henrik/Varun)

Proposal 1: Add “transceiver” stats:

```
dictionary RTCRtpTransceiverStats : RTCStats {
    DOMString senderId;
    DOMString receiverId;
    DOMString mid;
    DOMString direction;
    DOMString currentDirection;
    sequence<DOMString> codecIds;
}
```

Proposal 2: Just add the “mid” to the “sender” and “receiver” stats.

Issue 480: What should we do about onstatsended? / Issue 472: Issues with replaceTrack, will statsended fire or give me what I want (Henrik)

Problem: It is useful to know if outbound-rtp metrics changed because the track was replaced or because of other conditions (network etc). replaceTrack() would trigger “track” stats to change, causing onstatsended. If “track” is deprecated, how would you know that “something significant happened”?

Proposals:

1. Remove “onstatsended”:
   - The application knows if it called replaceTrack() and could trigger getStats() again.
   - Each getStats() report tells you what the mediaSourceId is; you can tell if it changed.
     Warning: May encourage more frequent getStats() polling, like 1s instead of 10s.
2. **Replace it with “RTCPeerConnection.onreplacetrack” which fires when replaceTrack() resolves.**
   - **(This is similar to “ontrack” firing when SLD/SRD resolves!)**
3. **Replace “onstatsended” by “onstatsevent”, fired with an enum describing what “significant stats event” occurred, such as stats ending or replaceTrack happening.**

Issue 479: Should we move track/sender/receiver to the obsolete section? (Henrik)

After PR 463 and PR 466 moved all track/sender/receiver stats to outbound-rtp and inbound-rtp, the only remaining “track” members not obsolete are:

- ~~`mediaSourceId`~~
  - Outbound: **Already present in outbound-rtp**
  - Inbound: **Not applicable to inbound-rtp**
- `trackIdentifier`
  - Outbound: **Already present as “outbound-rtp.mediaSourceId → media-source.trackIdentifier”**
  - Inbound: **[Proposal] Add inbound-rtp.trackIdentifier**
- ~~`remoteSource`~~
  - **Not needed; can tell direction by type ("outbound-rtp" or "inbound-rtp")**
- `ended`
  - Outbound: **[Proposal] Add media-source.ended. MediaStreamTrack stats belong here!**
  - Inbound: **[Proposal] If transceiver.currentDirection stats exists this is not needed. Otherwise, add inbound-rtp.ended. Or maybe not needed if the RTP stream is not referenced by a receiver anymore?**
- ~~`kind`~~
  - **Already exists in outbound-rtp and inbound-rtp.**
- `priority`
  - **This refers to a Feature at risk in webrtc-pc. [Proposal] If we still want it, solve as part of encoding parameters (#398) or add it to outbound-rtp. N/A to inbound-rtp**

Issue 479: Should we move track/sender/receiver to the obsolete section? (Henrik) (Hidden surprise slide!)

Note that…

- **"track" stats are not needed.**
  - **"track" does not contain any accumulative counters anymore.**
  - **The only stat that changes on replaceTrack() is mediaSourceId.**
  - **It's only function seems to be to get onstatsended to fire… (#480)**
- **"sender" stats are needed to group outbound-rtp of simulcast layers (same senderId), however if we have "transceiver" stats (#478) then we don't even need "sender" stats.**

Opinion: "sender" and "receiver" stats still make sense to match the APIs, and a sensible place to put mediaStreamIds stats (#470), but does not contain any useful information at the moment.

Proposal A:

- **Previous slide's proposals + Move "track" stats to the Obsolete section. Keep "sender" and "receiver" stats around.**

Proposal B:

- **Previous slide's proposals + Move "track", "sender" and "receiver" stats to the Obsolete section. Rely on "transceiver" stats for correlating simulcast streams.**

Content-Hints (Harald, 30 minutes)

Content-Hints API (Harald)

- **Issues**
  - **[Issue 2248](): degradationPreference is under-specified (bernard)**
  - **[Issue 28](): Redundancy and lack of normative clarity around interaction with constraints (Jan-Ivar)**
  - **[Issue 30](): Permanence Issues (Jan-Ivar)**

[Issue 2248](#): degradationPreference is under-specified (bernard)

- Effect not easily tested
  - WPT only tests the ability to set and get the value of the degradationPreference attribute.
  - Effect not easily determined in a loopback test
- Disparate implementations
  - Chrome: degradationPreference has no effect, currently.
    - C/C++ level: only used to decide whether to reduce resolution or framerate in the event of congestion
  - Current Edge: treats degradationPreference similarly to a content-hint
    - “Prefer-resolution” equivalent to “detail” content-hint
    - “Prefer-framerate” equivalent to “motion” content-hint
- Recommendation
  - Include degradationPreference as a “feature at risk”
  - Add clarification that degradationPreference is purely about the resolution/framerate tradeoff

Issue 28: Redundancy & lack of normative steps; interaction with constraints (jib)

“Hints” with unspecified behavior favor dominant browser implementation. Everyone else must reverse engineer. A missed opportunity to make things normatively testable.

Mentions "Echo-cancellation", "noise suppression", "boost intelligibility of the incoming signal" (autoGainControl?)

Hints seem somewhat useful as a simpler API to constraints:

```
// Assuming unconstrained audio track
track.contentHint = "music";
console.log(track.getSettings().echoCancellation); // false?
console.log(track.getSettings().noiseSuppression); // false?
console.log(track.getSettings().autoGainControl);  // false?

await track.applyConstraints({echoCancellation: true});

console.log(track.getSettings().echoCancellation); // true
console.log(track.getSettings().noiseSuppression); // false?
console.log(track.getSettings().autoGainControl);  // false?
```

Bonus: more directly controls “default” values Today latter ones tend to track echoCancellation when absent.

Q: Which spec should specifies this?

[Issue 30]: Permanence Issues (Jan-Ivar)

Hints are ***not*** inherent properties of a track e.g. a `"music"` track; a `"motion"` video; invariant and ever-present:

1. **They're a control surface**, a runtime knob. JavaScript can modify them at any time, expecting results, yet results are not specified anywhere, nor when observable effects may be expected, if any.

2. **They don't follow the media** i.e. track replication through sink → source pipes like peer connection, element.captureStream(), web audio, MediaRecorder, or track.clone() (or do they)? Spec doesn't say.

3. **They may be wrong.** User agents may (someday) detect speech vs. music at run-time. Are they allowed to ignore bad hints? If they're ignored, what happens to any observable (testable) effects/settings we define? If user agents *can't* ignore them, are they a footgun API? Misnomer? Option: allow ignoring "in the future"?

How should browsers behave if the JS twiddles the bit live over time?

How does it work with track.clone()? Can each clone have its own value?
Is JS expected to tack these hints back on like post-it notes that keep falling off?

Should normative language live in the contentHint spec, or be pushed to individual specs using contentHints?
Spec should probably give guidance to other (future) specs at least, to ensure consistency.

Disposing of content-hint

- Currently implemented in Chrome, Edge Preview
- Persistent usage, but very low
  - Content-hints API enabled in Skype screen sharing this week.
- Drop, advance or icebox?

Other Current Specifications (Harald & Peter, 30 minutes)

DSCP API Status Report (Harald)

- Field trials of DSCP have given strictly worse performance than DSCP=0
  - Controlled deployments may be different
- No current implementations of DSCP API
- Also no current implementations of Priority API, which DSCP augments

Proposal 1: Move Priority to DSCP document, harmonize, keep experimenting

Proposal 2: Drop DSCP API

WebRTC-ICE: Reminder of Purposes

Enable RTCIceTransport (free from PC):

- Needed by everything else NV-style (SctpTransport, DtlsTransport, RtpXer)

Of FlexICE:

- wifi/cell control
- check activity/frequency control
- "relay first" checking
- continual gathering and
- network switching control
- forking

Reminder of status at TPAC 2018

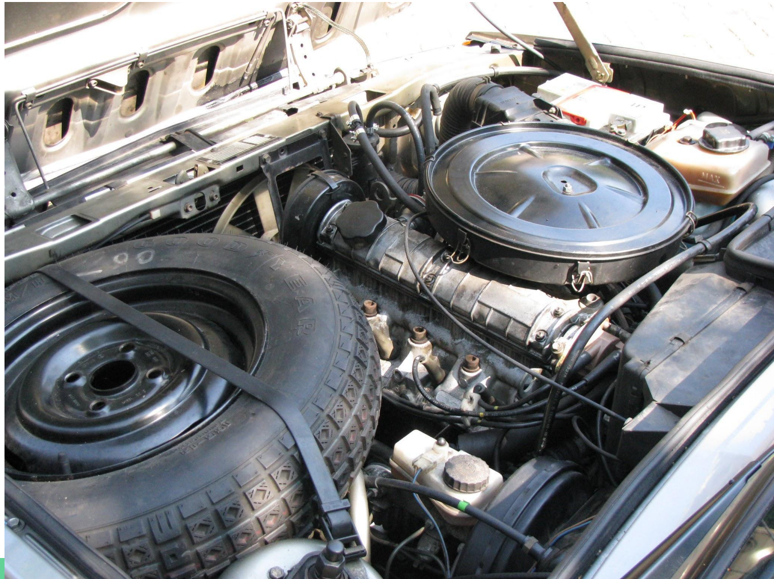
- Consensus on doing NV-style IceTransport (w/o PC)
- Consensus on doing FlexICE (generally)
  - With observation that it could apply to PC-style IceTransport

WebRTC-ICE Status Report (Peter)

- Editors draft available here: https://w3c.github.io/webrtc-ice/
- Open issues: 16
- Implemented in Chrome, Edge Beta
- Call for consensus to publish a WG draft?
- Functionality
  - Stand-alone IceTransport object with no SDP dependency
  - No forking support
  - Building block for Data Channel in workers (would need RTCDtlsTransport + RTCDataChannel in addition)
  - Does not currently meet requirements for p2p mesh use case (slides to follow)

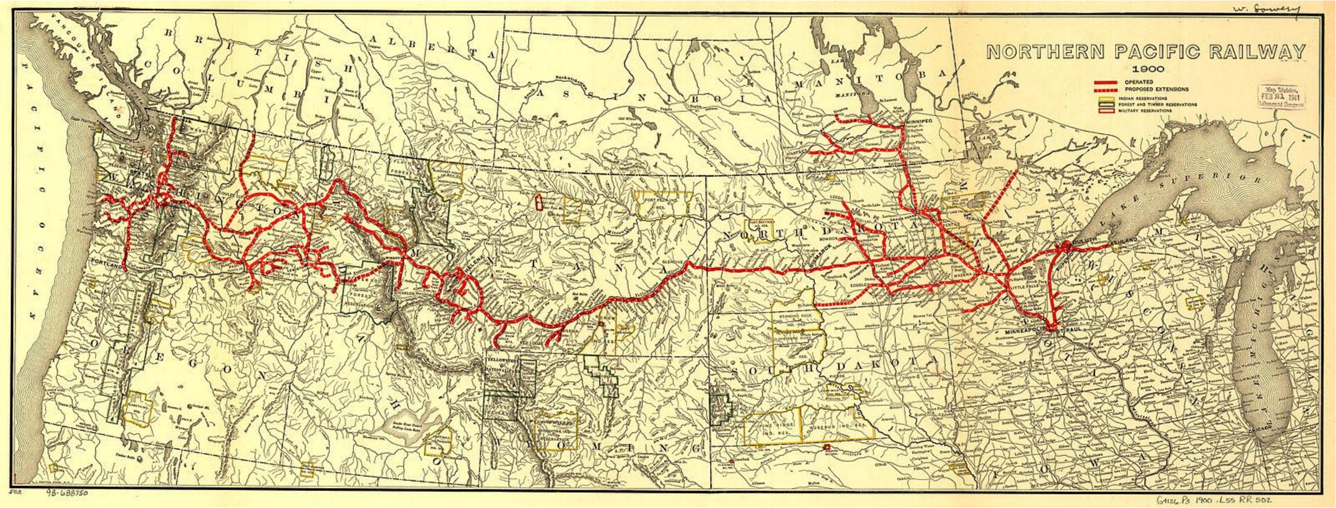
Implementation Progress

- Implementation of NV-style IceTransport in Chrome (no forking support, no FlexICE)
- ORTC-style implementation in current Edge (no forking support, no FlexICE)



Next Steps

- Implement NV-style data channels on top (is there still interest in this?)
- Understand potential use cases motivating FlexICE (e.g. P2P mesh)
- Implement FlexICE



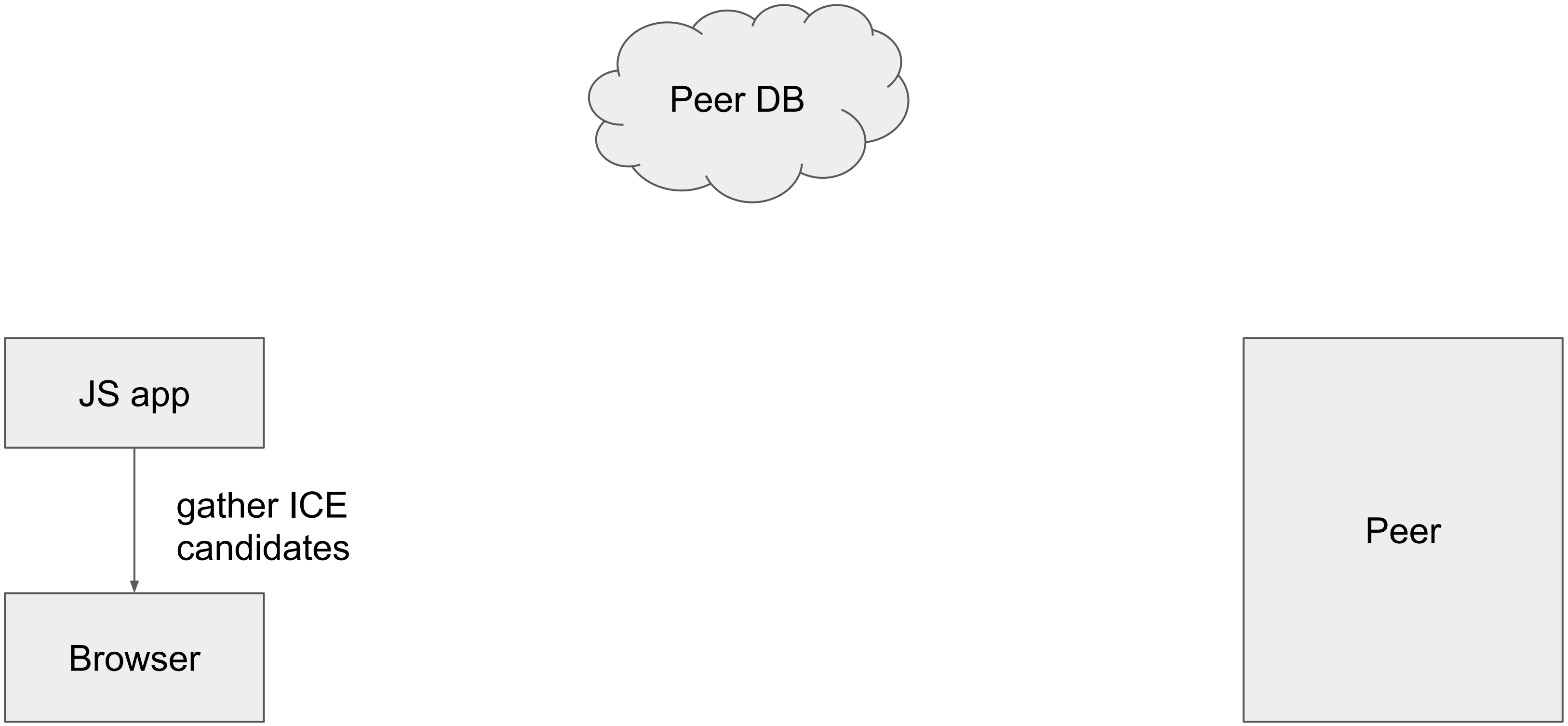
What do the p2p mesh folks need?

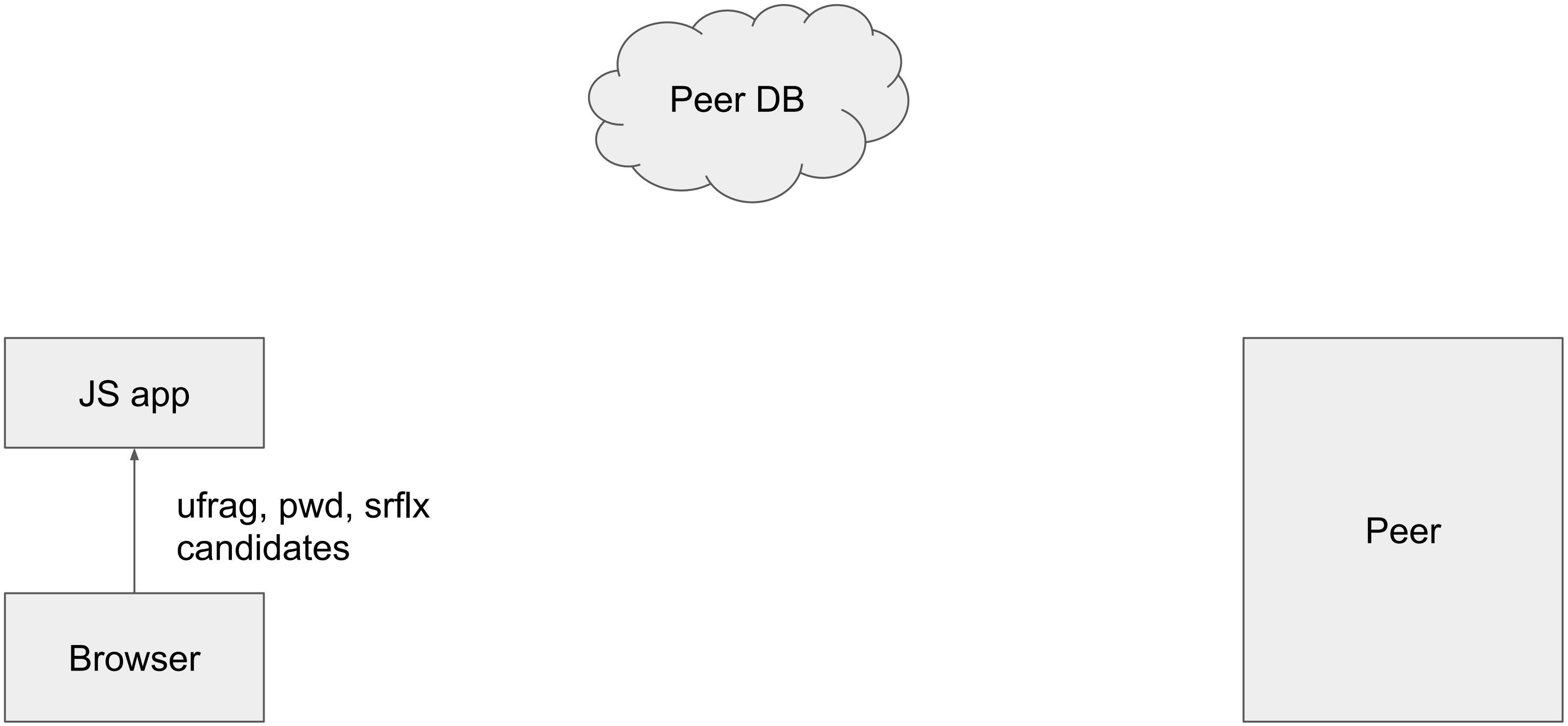
To act like a server

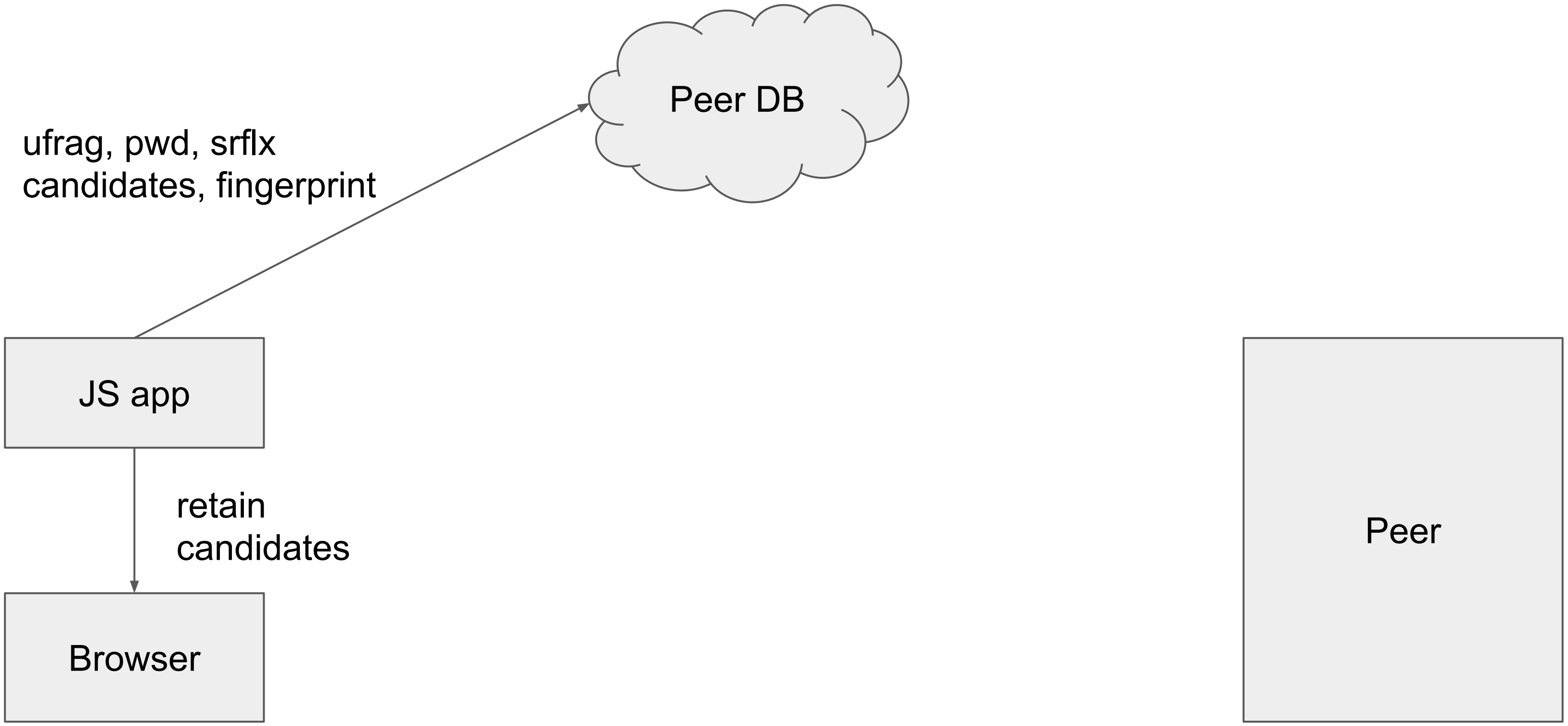
Post local candidates

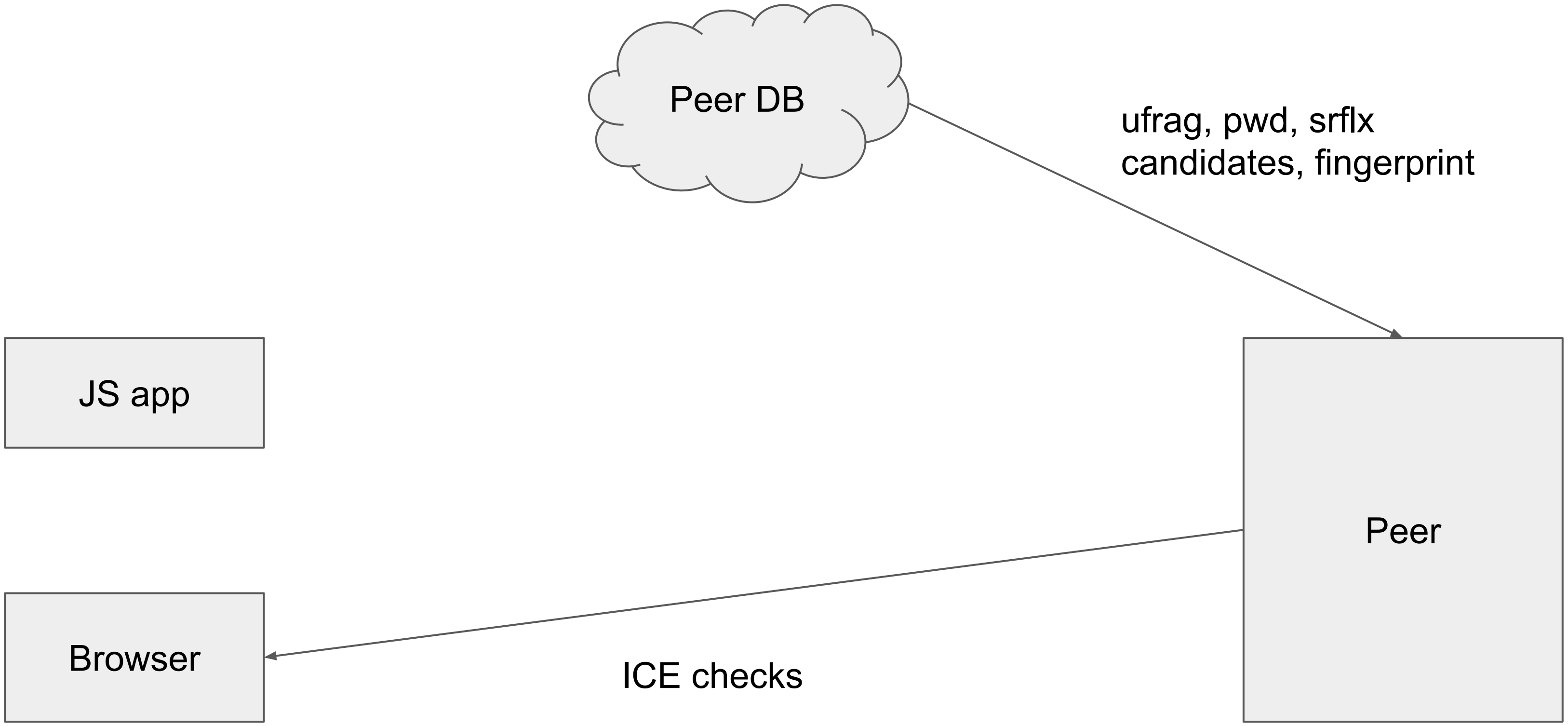
Listen for incoming connections

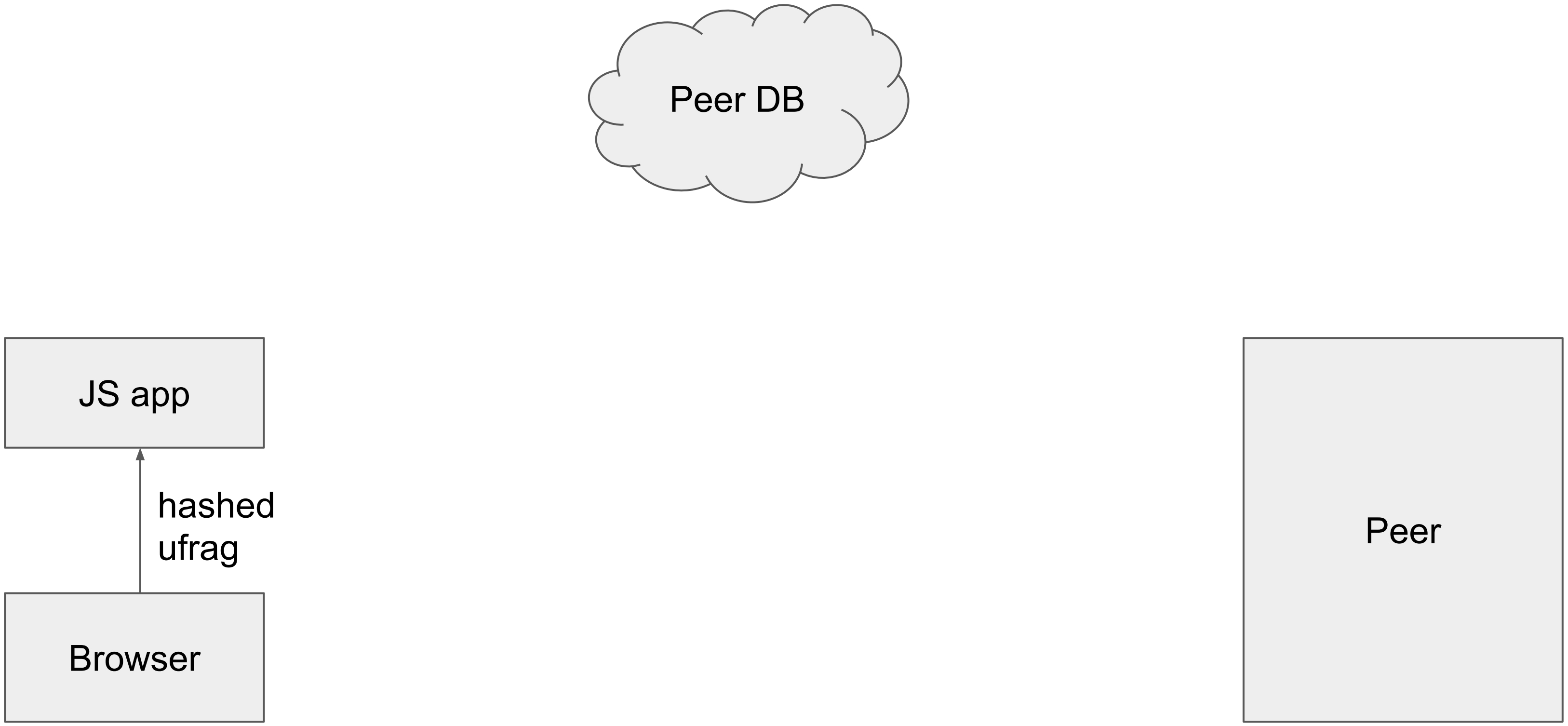
Scale to many connections

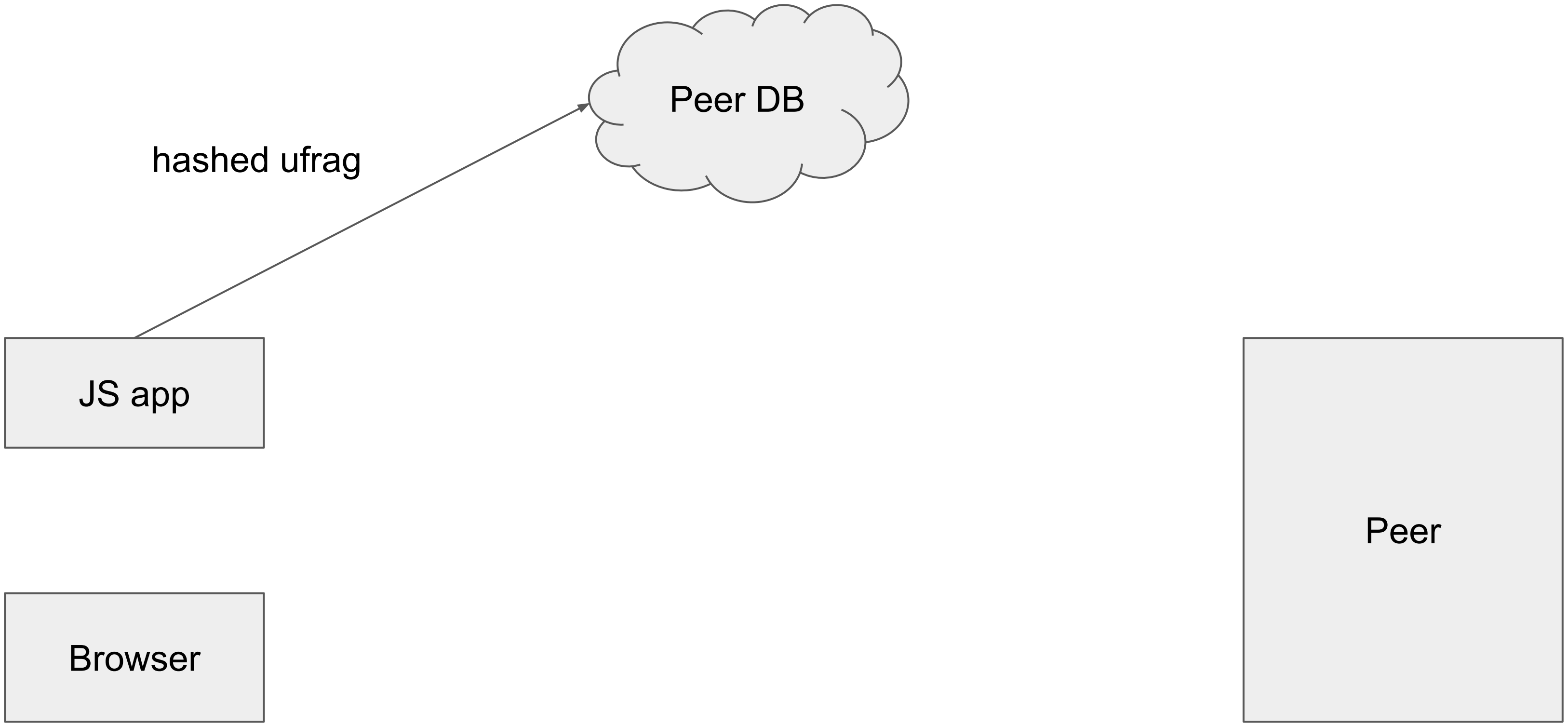

Peer DB
JS app
gather ICE
candidates
Browser
Peer

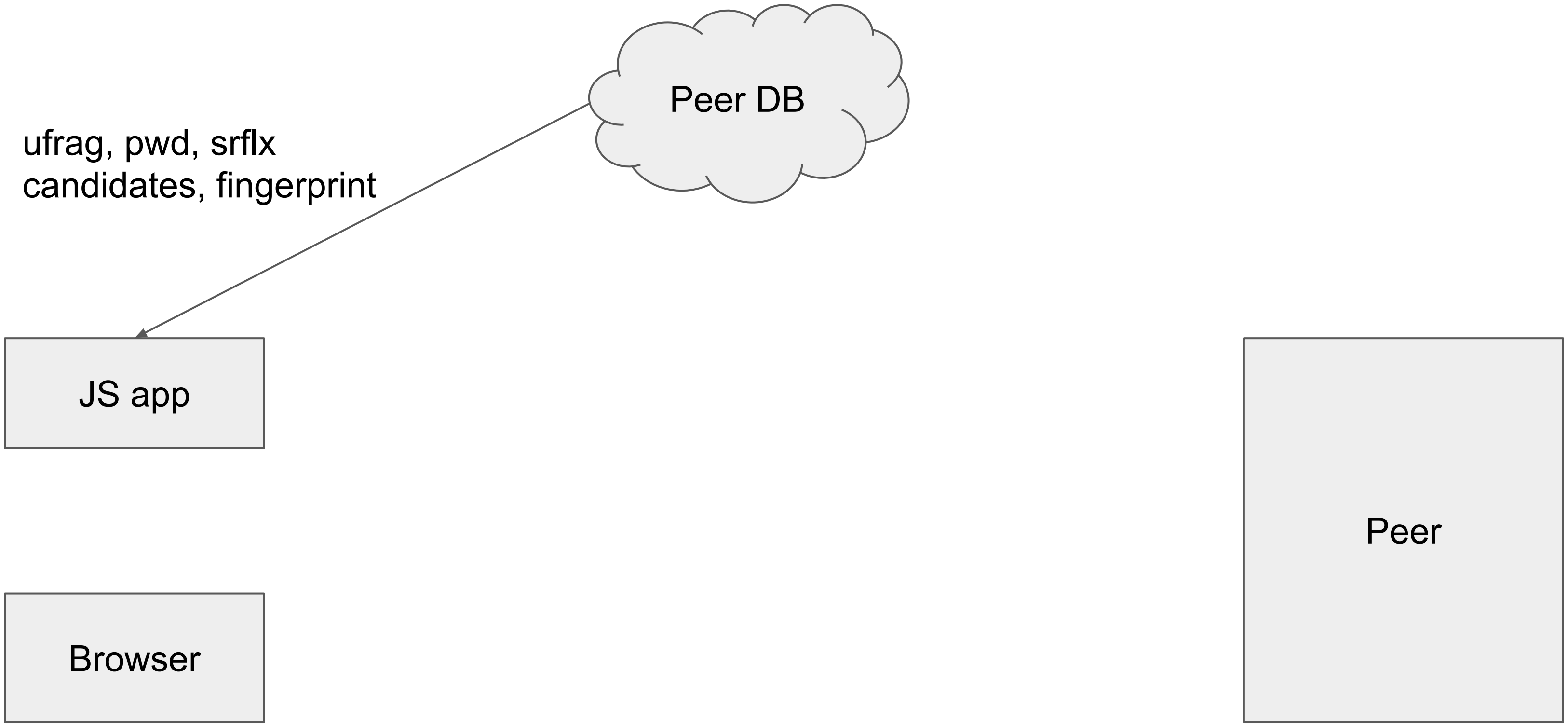

Peer DB
JS app
ufrag, pwd, srflx
candidates
Browser
Peer

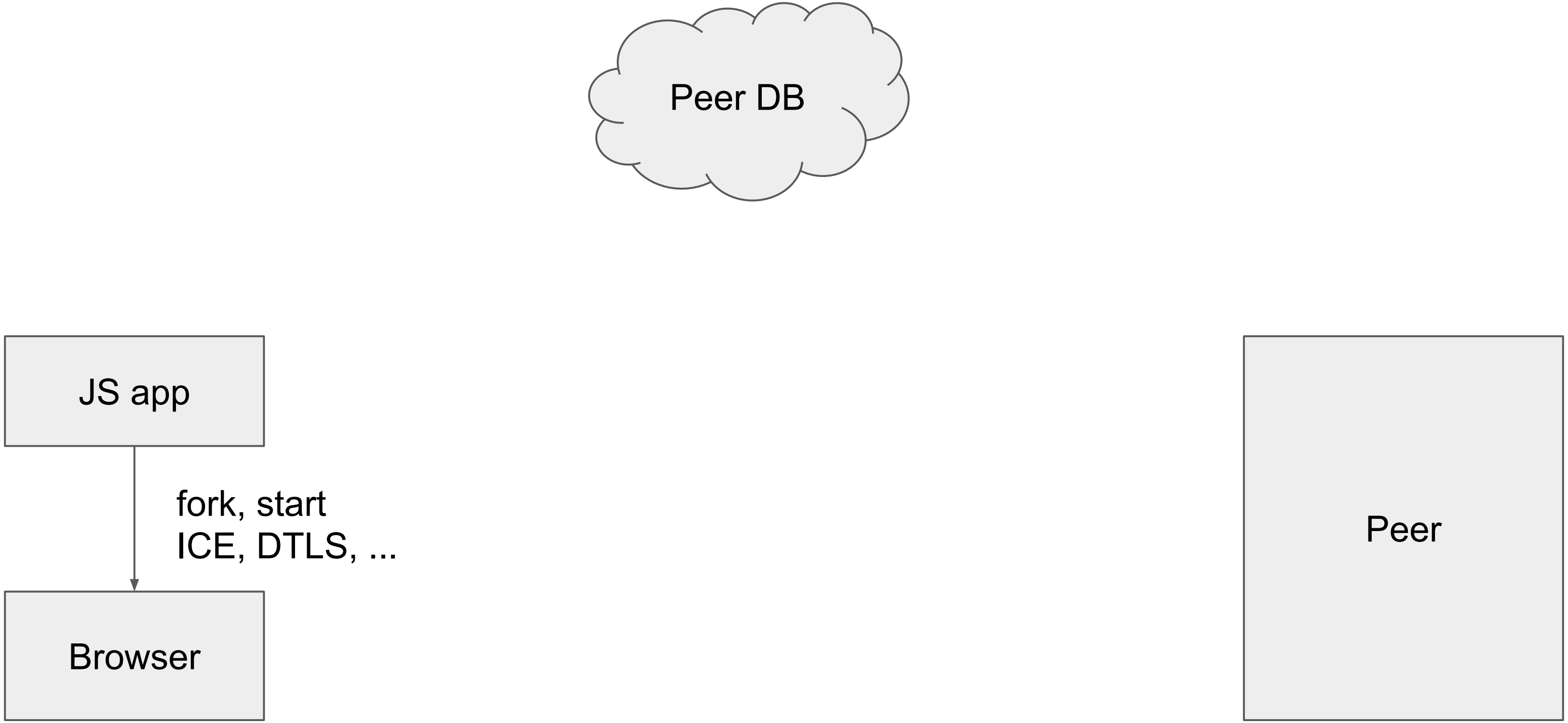

Peer DB
ufrag, pwd, srflx
candidates, fingerprint
JS app
retain
candidates
Browser
Peer

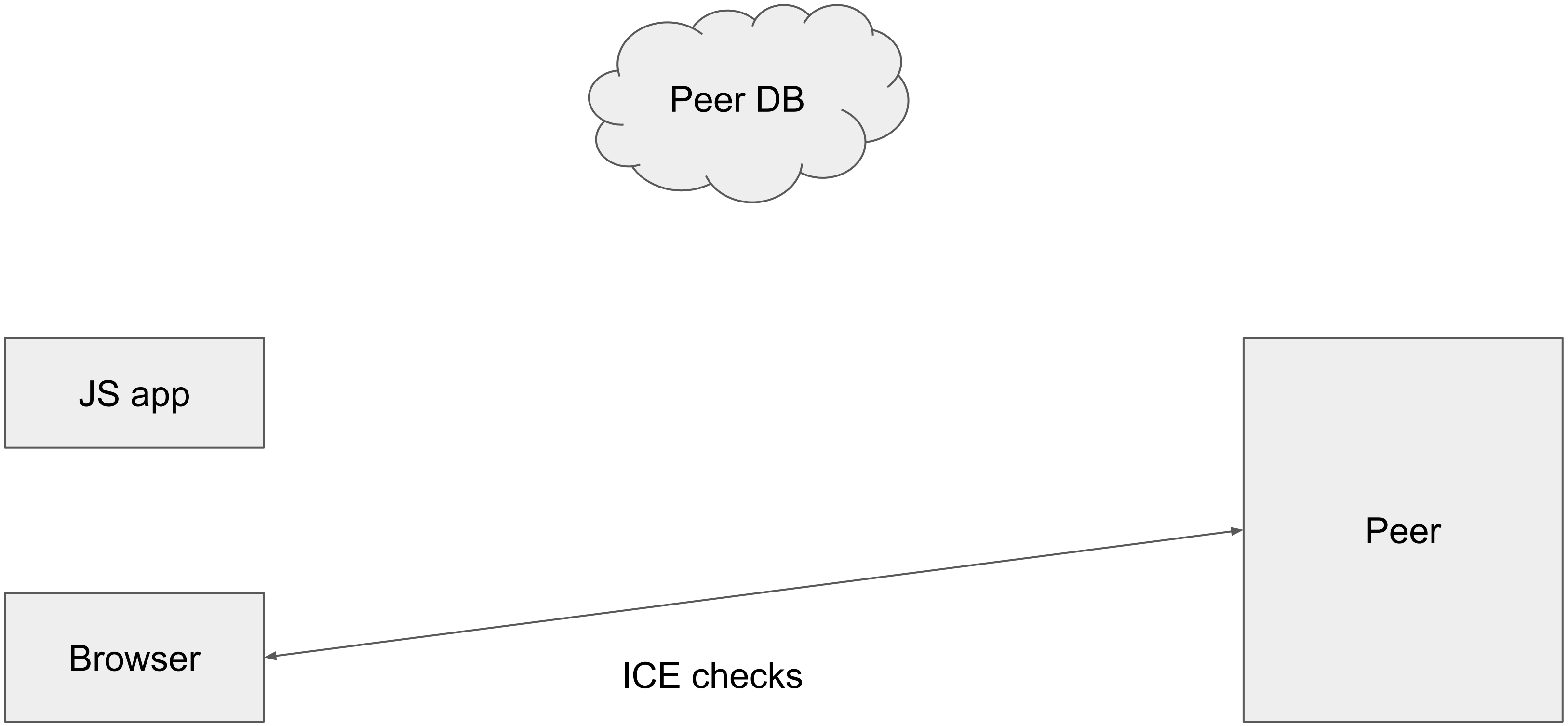

Peer DB
ufrag, pwd, srflx
candidates, fingerprint
JS app
Peer
Browser
ICE checks

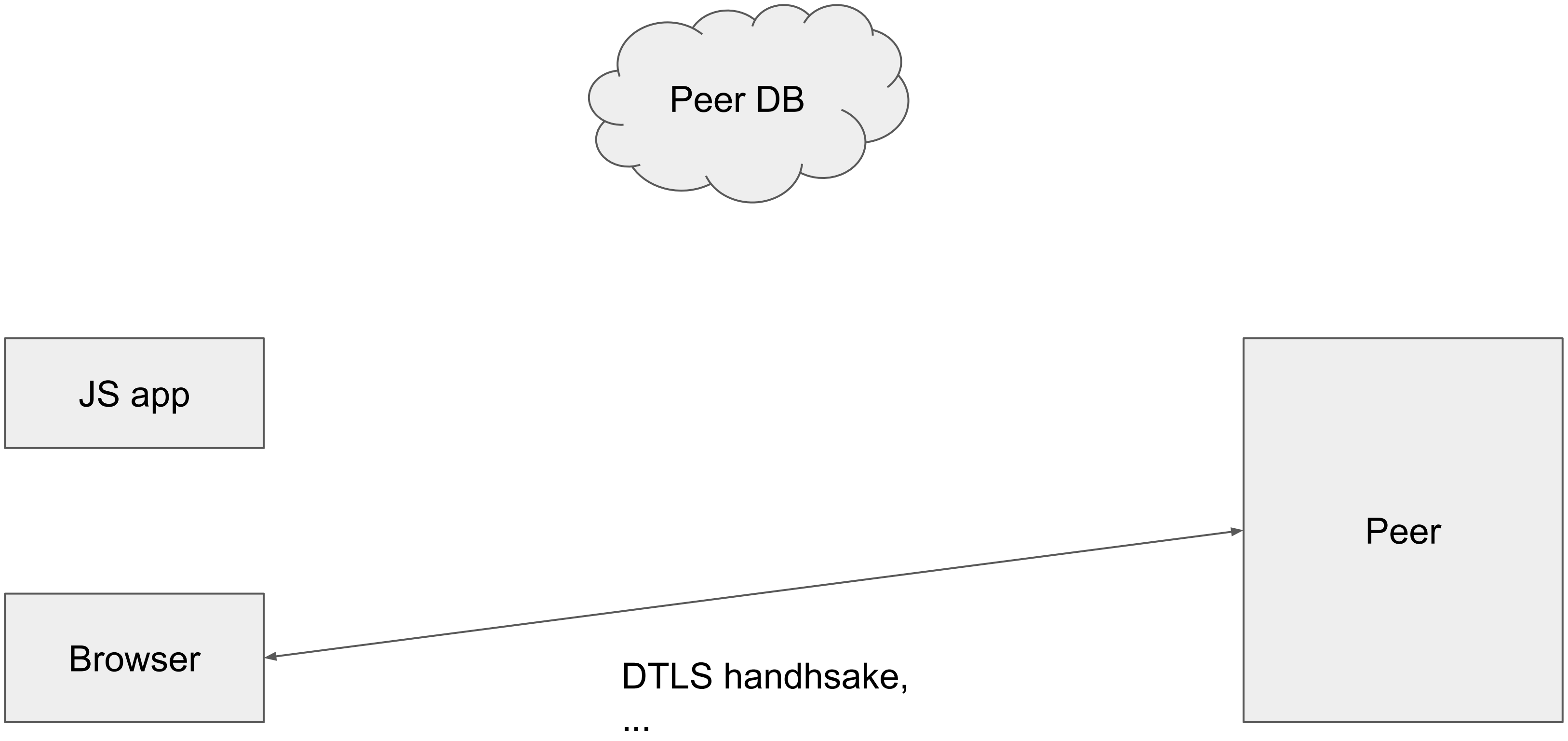

Peer DB
JS app
hashed
ufrag
Browser
Peer


Peer DB
hashed ufrag
JS app
Browser
Peer


Peer DB
ufrag, pwd, srflx
candidates, fingerprint
JS app
Browser
Peer


Peer DB
JS app
fork, start
ICE, DTLS, ...
Browser
Peer


Peer DB
JS app
Peer
Browser
ICE checks


Peer DB
JS app
Browser
Peer
DTLS handhsake,
...

Problems

local candidates only usable for one connection

local candidates only useful temporarily

Must know remote ufrag/pwd

Can't create very many PeerConnections

Solutions

ICE forking

retainCandidate()

.onreceivedcheck

Free-standing IceTransport, SctpTransport

ICE forking

```
let iceServer = new IceTransport();
iceServer.gather({iceServers: ...});
// ORTC-style IceGatherer might be better
let iceClient1 = iceServer.fork();
iceClient1.start({...});
let iceClient2 = iceServer.fork();
iceClient2.start({...});
```

retainLocalCandidate()

```
let ice = new IceTransport();
ice.gather({iceServers: ...});
ice.onicecandidate = (evt) => {
 if (evt.candidate.type = "srflx") {
   iceServer.retainLocalCandidate(evt.candidate);
   post(evt.candidate, ice.localParameters);
 }
}
```

.onreceivedcheck

```
let iceServer = new IceTransport();
iceServer.gather({iceServers: ...]});
iceServer.onreceivedcheck = (evt) => {
  let peer = await lookup(
    evt.hashedRemoteUsernameFragment)
  let iceClient = iceServer.fork();
  iceClient.start(peer.iceParams);
}
```

Free-standing objects

```
let ice = new IceTransport();
// Option A
let dtls = new DtlsTransport(ice, cert);
let sctp = new SctpTransport(dtls);
// Option B
let quic = new QuicTranspot(ice, cert);
```

Full example

```
let iceServer = new IceTransport();
iceServer.gather({iceServers: ...});
let localCertificate = ...;
ice.onicecandidate = (evt) => {
 if (evt.candidate.type = "srflx") {
   iceServer.retainLocalCandidate(evt.candidate);
   post(evt.candidate, ice.localParameters, localCertificate);
 }
}
iceServer.onreceivedcheck = (evt) => {
  let peer = await lookup(evt.hashedRemoteUsernameFragment) // <-- HERE BE THE MAGIC
  let iceClient = iceServer.fork();
  iceClient.start(peer.iceParams);
  let quic = new QuicTransport(iceClient, localCertificate);
}
```

Will this work with NATs?

A full cone NAT will let the ICE checks through

Apparently that's good enough for p2p meshes

Why is the ufrag hashed?

To avoid abuse.
Otherwise, one could use the ufrag field to send arbitrary data from a server to a client (without encryption and congestion control)

Kind of like this, only easier

Is this hard to implement?

ORTC-style IceTransport: easy

.onreceivedcheck: probably easy

long-lived candidates: probably easy

Free-standing DtlsTransport/SctpTransport: moderate

ICE forking: probably hard (supported in Ortc lib)

TL;DR

1. Are we willing to implement ICE forking?
2. Are we willing to implement free-standing objects?
3. Is .onreceivedcheck safe?

👍 ? 😄 : 😢

Break
3:00 PM - 3:30 PM

“Features at Risk”
(Jan-Ivar, 60 minutes)

“Feature at risk” Overview

1. **Features at risk Options**
2. **Spec gap analysis: Gray area. Where implementations are.**
3. **Features at risk Options (revisit)**
4. **Features at risk in WebRTC 1.0**

“Feature at risk” Options

- **If there are no implementations and no developer interest:**
  - **Remove the “feature at risk”**
- **If there are no implementations but developer interest:**
  - **Leave the “feature at risk” in the spec (if implementation is imminent)**
  - **Move the “feature at risk” to an extension specification**
- **If there is at least one implementation**
  - **Leave the “feature at risk” in the spec (if another implementation is imminent)**
  - **Move the “feature at risk” to an extension specification**

Spec gap analysis Features implemented	Chrome	Edge	Firefox	Safari	**2+**	comment
rollback	❌💪	❌💪	✔️	❌	💪	Have commitment
RTCIceTransport	✔️	✔️	❌		✔️	Missing members?
RTCDtlsTransport	✔️	✔️	❌		✔️	Missing members?
RTCSctpTransport (aka `pc.sctp`, not really a "transport" per se)	✔️	❌	❌💪	❌	💪	Low-hanging fruit
setParameters	✔️	✔️	❌		✔️	Missing members?
VoiceActivityDetection	✔️ buggy	❌	❌	❌	❌	At risk
iceCandidatePoolSize	✔️	✔️	❌	✔️	✔️	
RTCOAuthCredential	❌	❌	❌	❌	❌	At risk (Extension?)
RTCRtcpMuxPolicy's "negotiate" value	❌	❌	❌	❌	✔️	At risk
RTCCertificate.getSupportedAlgorithms()	❌	❌	❌	❌	❌	At risk
PRAnswer	✔️	❌	❌	✔️	✔️	fiddle ✔️
QoS dc.priority & setParameters({priority})	❌	❌	❌	❌	❌	At risk
RTCPeerConnection's onicecandidateerror	✔️	✔️	❌	❌	✔️	
RTCError/RTCErrorEvent	constructor	❌	❌	❌	❌	Hard to remove. Error wiggle room?

Only features marked with ❌ in **2+** column are missing two implementations (red name) ***and*** have no near-term commitments

Legend:

❌ = not implemented
✔️ = implemented
💪 = Working on it or have an actual developer assigned to work on it near-term (2019).

From looking at wpt.fyi/interop/webrtc?label=master

And missing from https://dontcallmedom.github.io/webrtc-impl-tracker/?webrtc

Spec gap analysis Features implemented	Chrome	Edge	Firefox	Safari	**2+**	comment
Simulcast-aware stats	❌	❌💪	❌💪	❌	💪	Have commitment
Sending of blobs in data channels	❌	❌	✔️	✔️	✔️	fiddle
statsended	❌	❌	❌💪	❌	❌	Remove
sender.setStreams	✔️	❌	❌💪	❌💪	💪	Have commitment
RTCPeerConnectionIceEvent's url	❌💪	❌💪	❌	✔️	❌	Unscheduled
Identity	❌	❌	✔️	❌	❌	Complete move to extension spec
getDefaultIceServers	❌	❌	❌	❌	❌	At risk
setCodecPreferences	✔️	✔️	❌	❌	✔️	
maxFramerate	❌💪	❌💪	❌	❌	💪	JIB: inserted Friday

“Feature at risk” Options

28 [mandatory stats] show up in WPT as not implemented, but this is a result of stats members having moved in the spec, so they have proof-of-concept implementations, and are therefore considerered NOT at risk:

- **RTCReceivedRtpStreamStats's packetsDiscarded**
- **RTCInboundRtpStreamStats's receiverId**
- **RTCInboundRtpStreamStats's remoteId**
- **RTCOutboundRtpStreamStats's senderId**
- **RTCOutboundRtpStreamStats's remoteId**
- **RTCRemoteOutboundRtpStreamStats's localId**
- **RTCRemoteOutboundRtpStreamStats's remoteTimestamp**
- **RTCDataChannelStats's dataChannelIdentifier**
- RTCMediaStreamStats's streamIdentifer
- RTCMediaHandlerStats's trackIdentifier
- RTCMediaHandlerStats's remoteSource
- RTCMediaHandlerStats's ended
- RTCAudioHandlerStats's audioLevel
- RTCVideoHandlerStats's frameWidth
- RTCVideoHandlerStats's frameHeight
- RTCVideoHandlerStats's framesPerSecond
- RTCVideoSenderStats's framesSent
- RTCVideoReceiverStats's framesReceived
- RTCVideoReceiverStats's framesDecoded
- RTCVideoReceiverStats's framesDropped
- **RTCVideoReceiverStats's partialFramesLost**
- **RTCCodecStats's codecType**
- **RTCCodecStats's channels**
- **RTCCodecStats's sdpFmtpLine**
- **RTCTransportStats's rtcpTransportStatsId**
- RTCIceCandidateStats's address ← Called id in Chrome
- **RTCIceCandidateStats's url**
- RTCCertificateStats's issuerCertificateId ← N/A

“Feature at risk” Options

- **If there are no implementations and no developer interest:**
  - **Remove the “feature at risk”**
- **If there are no implementations but developer interest:**
  - **Leave the “feature at risk” in the spec (if implementation is imminent)**
  - **Move the “feature at risk” to an extension specification**
- **If there is at least one implementation**
  - **Leave the “feature at risk” in the spec (if another implementation is imminent)**
  - **Move the “feature at risk” to an extension specification**

Webrtc-PC "Features at risk"

- **Issue 1:** `Oauth` **value of** `RTCIceCredentialType (extension)`
- **Issue 2:** `RTCOauthCredential` **dictionary** `(extension)`
- **Issue 3:** ~~**non-multiplexed RTP/RTCP:** `negotiate` **&** `rtcpTransport`~~ `(Remove)`
- **Issue 4:** ~~`voiceActivityDetection` **attr of** `RTCOfferAnswerOptions`~~ `(Rem)`
- **Issue 5:** `getDefaultIceServers()` **method of** `RTCPeerConnection (ext)`
- **Issue 6:** ~~`RTCPriorityType` **enum**~~ `(remove)`
- **Issue 7:** ~~`getSupportedAlgorithms` **method**~~ `(remove)`
- **Issue 8:** ~~`RTCRtpSendParameters.priority`~~ `(remove)`
- **Issue 9:** ~~`RTCRtpReceiveParameters.encodings`~~ `(remove)`
- **Issue 10:** ~~`RTCRtpEncodingParameters.dtx`~~ `(remove) &`
- ~~`RTCRtpEncodingParameters.ptime & payloadType`~~ `(remove)`
- **Issue 11: (JIB: maxFrameRate was not discussed Thursday due to slide typo)**
- **Issue 12:** ~~`RTCDataChannel.priority`~~ `(remove) &` ~~`RTCDataChannelInit.priority`~~ `(remove)`

Developer Feedback Session (Bernard, 60 minutes)

Presenter Feedback

- **Sean DuBois (Pion)**
- **Mészáros Mihály (GÉANT/GITDA, coTURN)**
- **8x8 Team**
- **Silvia Pfeiffer**

Sean’s Developer Feedback

- ICETransport
  - Restrictive networks only allow port ranges/some interfaces (users want more control)
  - addIceCandidate before SetRemoteDescription is a common bug
- DataChannel
  - Closing (Message+Code) can be done by sending final message
  - Ability to ‘deny’ a DataChannel. Currently accept/close quickly
  - Media via Datachannels is being chosen more and more (can be HW accelerated)
- RTPTransports
  - More control over latency/loss tradeoffs. Security camera always wants sub-second, others demand zero loss
  - setCodecPreference not available yet (Users want to force H264, SDP munging painful)
- DtlsTransport
  - Lack of CipherSuite choice, people want to explicitly choose AES-GCM (SFU/embedded)
- PeerConnection(?)
  - addStream vs addTrack vs addTransceiver (Users unsure which to use)
  - Provisional Offers/Answers come up constantly. Any value?
  - API is being targeted in other languages. Possible to keep API portable/simple?
  - Tor Snowflake has issues with fingerprinting (Can we set Ciphers, other details)

Mészáros Mihály's Developer Feedback

- Use Case: TURN for NRENs (Education & Research) community
  - TURN is the best if it is distributed around the globe to keep latency low
  - NRENs have vm-s, high bandwidth network around the globe, and we trust each each other
    - keep media traffic in our network
  - Multi-tenant TURN is needed (multiple auth database, single service and operation)
- TURN Auth
  - TURN (RFC5766) default auth Long Term Credential (not designed for web, can't hide credential)
    - No co located TURN support (origin draft rejected in ietf)
  - Time Limited LTC (REST) draft-uberti-behave-turn-rest-00
    - It is an expired draft, not supports co-located TURN (it is canceled to move on to OAuth)
  - TURN + OAuth (RFC7635)
    - Supports co-Located TURN == multiple Auth Servers
      - AS: OAuth and PoP key distribution is ready to implement (+in coTURN there is a tool to create token)
      - TURN server: in coTURN there is an implementation validating the token
      - Firefox implementation started: https://bugzilla.mozilla.org/show_bug.cgi?id=1247616
- We should still consider to keep OAuth in Spec until we have alternative for TURN Auth that 1. IETF standard, 2. suitable for Web 3. supports coLocated TURN

Silvia’s Developer Feedback

- Recording API on Safari & iOS
- CPU requirements by videos are often too extensive
- Statistics since the reorg are quite limited and really not compatible: despite Chrome, Safari and Firefox all now implementing the standard stats, the actual content of the stats reports is still different and the stats report types returned by each browser is also a different set
- we have a whole pile of compatibility code that we need to have setup to get datachannels setup correctly

- as a business focused on delivering quality video calls to customers, we have so many different versions of the standards, and browsers, and devices to support
- part of that legacy is due to the constant, changing nature of the spec and vendors choosing to implement different parts - lack of interoperability is really challenging
- it'd be nice if mobile versions didn't have the occassional little gotcha when compared to desktop versions of the same browser

8x8 Developer Feedback

- The “Plan” mess. This huge change that needs to happen that will not bring us value. We’d therefore appreciate it if it can be implemented iteratively
  - SSRC’s being missing in Unified Plan is a pain. Means that we need to change clients and SFU simultaneously and can’t do it in two steps. Bigger project: harder to just squeeze it in.
- Chrome (and Edge Beta) can’t share certain Windows windows (Metro?)
- As an SFU provider, want to assist the call quality.
  - We cannot help audio quality much. For video we could add FEC, but there is no audio FEC support. Would like to add our own FEC to peer connections with the appropriate hooks.
  - PERC efforts in IETF do not work for us. Would need WebRTC hooks to be able to implement our own. Something like this would be great: https://github.com/alvestrand/webrtc-media-streams/blob/master/explainer.md

For extra credit

Name that reptile!

Thank you

Special thanks to:

WG Participants, Editors & Chairs

The reptile

Friday, September 20

W3C WG IPR Policy

- This group abides by the W3C Patent Policy https://www.w3.org/Consortium/Patent-Policy/
- Only people and companies listed at https://www.w3.org/2004/01/pp-impl/47318/status are allowed to make substantive contributions to the WebRTC specs

Welcome!

- Welcome to the Friday meeting of the W3C WebRTC WG at TPAC!
- During this meeting, we hope to make progress on the future of the WebRTC WG (WebRTC-NV use cases, extensions and WG Re-Charter).

Agenda for Friday, September 20

8:30 AM - 9:15 AM Scalable Video Coding Extension for WebRTC (Bernard & Florent)
Reference: https://w3c.github.io/webrtc-svc/

9:15 AM - 10 AM Privacy Issues (Youennf)
MediaCapture & Streams: https://w3c.github.io/mediacapture-main/
Audio Output Devices API: https://w3c.github.io/mediacapture-output/
Media Capture from DOM Elements: https://w3c.github.io/mediacapture-fromelement/
WebRTC-PC: https://w3c.github.io/webrtc-pc/

10:00 AM - 10:30 AM Break
10:30 AM - 11:00 AM WebRTC NV use cases (Bernard)
Reference: https://w3c.github.io/webrtc-nv-use-cases/

11:00 AM - 12:00 PM WEBRTC WG Re-Charter (Dom)

12:00 PM - 1 PM Lunch
1 PM - 2 PM Joint meeting with Accessibility Platform Architectures WG (Bernard)
Reference: https://www.w3.org/WAI/APA/wiki/Accessible_RTC_Use_Cases

2 PM - 3 PM WebRTC-Stats (Varun & Henrik)
Reference: https://w3c.github.io/webrtc-stats/
3 PM - 3:30 PM Break
3:30 PM - 4:30 PM TBD
4:30 PM - 5:30 PM Wrapup and Next Steps (Harald)

Scalable Video Coding Extension for WebRTC
(Bernard & Florent, 45 minutes)

WebRTC-SVC Issues

- PRs
  - **Issue 12/PR 13: Maintenance of scalabilityMode table and diagrams**
- Issues
  - **Issue 3: Custom scalability structures (bernard)**
  - **Issue 14: Encoding parameters for spatial layers (orphis)**
    - **Issue 4: Layer drop/add (bernard)**

Issue 12/PR 13: Maintenance of scalabilityMode table and diagrams (bernard)

- Currently, WebRTC-SVC includes a scalability mode table (Section 6) as well as prediction structure diagrams (Appendix B).
  - Scalability mode table was previously a subset of AV1 ("S" modes were missing), but now includes all modes other than "SS"
  - Diagrams based on (and in some cases, copied from) the AV1 bitstream specification Section 6.7.5.
- Concerns
  - Requires WebRTC-SVC specification to be updated whenever AV1 specification adds new modes or diagrams
  - W3C editorial standards require diagrams in .svg rather than .png format
- Proposed solution (PR 13)
  - W3C: Remove Appendix B.
  - AoMedia: reformat prediction structure diagrams in .svg format?

Issue 3: Custom scalability structures (bernard)

- Currently the WebRTC-SVC specification support discovery and configuration of:
  a. Pre-canned SVC scalability modes defined in AV1 Section 6.7.5
  b. “S” modes (simulcast on a single SSRC without RIDs)
- WebRTC-SVC does not currently support configuration of custom scalability structures.
  a. AV1 defines a “Scalability Structure” scalability mode that enables support for custom scalability structures, but WebRTC-SVC does not include this mode.
- Recommendation: Just say no!
  a. The (complex) AV1 Scalability Metadata syntax makes the same assumption as ORTC did (that SVC structures are hierarchical). So adding support for custom scalability structures cannot enable support for a wider range of AV1 SVC modes than is already supported by pre-canned modes.
  b. If there is a need for better support for scaling ratios, this can be achieved by adding new pre-canned scalability modes (see next slide).

Preconfigured Modes (from AV1 bitstream specification)

scalability_mode_idc	Name of scalability_mode_idc
0	SCALABILITY_L1T2
1	SCALABILITY_L1T3
2	SCALABILITY_L2T1
3	SCALABILITY_L2T2
4	SCALABILITY_L2T3
5	SCALABILITY_S2T1
6	SCALABILITY_S2T2
7	SCALABILITY_S2T3
8	SCALABILITY_L2T1h
9	SCALABILITY_L2T2h
10	SCALABILITY_L2T3h
11	SCALABILITY_S2T1h
12	SCALABILITY_S2T2h
13	SCALABILITY_S2T3h
14	SCALABILITY_SS
15	SCALABILITY_L3T1
16	SCALABILITY_L3T2
17	SCALABILITY_L3T3
18	SCALABILITY_S3T1
19	SCALABILITY_S3T2
20	SCALABILITY_S3T3
21	SCALABILITY_L3T2_KEY
22	SCALABILITY_L3T3_KEY
23	SCALABILITY_L4T5_KEY
24	SCALABILITY_L4T7_KEY
25	SCALABILITY_L3T2_KEY_SHIFT
26	SCALABILITY_L3T3_KEY_SHIFT
27	SCALABILITY_L4T5_KEY_SHIFT
28	SCALABILITY_L4T7_KEY_SHIFT
29-255	reserved

Common Scaling Ratios

- 16:9
  - 1920 x 1080
  - 1280 x 720 (1.5:1)
  - 640 x 360 (2:1)
- 4:3
  - 1920 x 1440
  - 1280 x 960 (1.5:1)
  - 640 x 480 (2:1)
  - 320 x 240 (2:1)

AV1 Scalability Metadata (Section 6.7.5)

The scalability metadata provides two mechanisms for describing the underlying picture prediction structure of the bitstream:

1. Selection among a set of preconfigured structures, or modes, covering a number of cases that have found wide use in applications.
2. A facility for specifying picture prediction structures to accommodate a variety of special cases.

The preconfigured modes are described below. The mechanism for describing alternative structures is described in scalability_structure() below.

All predefined modes follow a dyadic, hierarchical picture prediction structure. They support up to three temporal layers, in combinations with one or two spatial layers. The second spatial layer may have twice or one and a half times the resolution of the base layer in each dimension, depending on the mode. There is also support for a spatial layer that uses no inter-layer prediction (i.e., the second spatial layer does not use its corresponding base layer as a reference) and a spatial layer that uses inter-layer prediction only at key frames. The following table lists the predefined scalability structures.

AV1 Scalability Metadata

5.8.5. Metadata scalability syntax

metadata_scalability( ) {
scalability_mode_idc
if (scalability_mode_idc == SCALABILITY_SS)
scalability_structure( )
}

5.8.6. Scalability structure syntax

scalability_structure( ) {	Type
spatial_layers_cnt_minus_1	f(2)
spatial_layer_dimensions_present_flag	f(1)
spatial_layer_description_present_flag	f(1)
temporal_group_description_present_flag	f(1)
scalability_structure_reserved_3bits	f(3)
if (spatial_layer_dimensions_present_flag) {	
for (i = 0; i <= spatial_layers_cnt_minus_1 ; i++) {	
spatial_layer_max_width[i]	f(16)
spatial_layer_max_height[i]	f(16)
}	
}	
if (spatial_layer_description_present_flag) {	
for (i = 0; i <= spatial_layers_cnt_minus_1; i++)	
spatial_layer_ref_id[i]	f(8)
}	
if (temporal_group_description_present_flag) {	
temporal_group_size	f(8)
for (i = 0; i < temporal_group_size; i++) {	
temporal_group_temporal_id[i]	f(3)
temporal_group_temporal_switching_up_point_flag[i]	f(1)
temporal_group_spatial_switching_up_point_flag[i]	f(1)
temporal_group_ref_cnt[i]	f(3)
for (j = 0; j < temporal_group_ref_cnt[i]; j++) {	
temporal_group_ref_pic_diff[i][j]	f(8)
}	
}	
}	
}	

Issue 14: Encoding parameters for spatial layers (orphis)

It would be interesting to be able to control the encoding parameters for the spatial layers in a similar way we can control simulcast layers. I was thinking of an array of spatial coding parameters nested in the RTCRtpEncodingParameters dictionary, one entry per layer.

I believe we should be able to have parameters like maxBitrate, maxFramerate, scaleResolutionDownBy and the active flag.

- Bitrate and framerate are quite self explanatory.
- scaleResolutionDownBy should be decreasing for each spatial layer (aka layers have to be bigger). There will be some interaction with the same encoding parameter for simulcast to define.
- active is a bit tricky when there are layer dependencies and should disable that layer and all layers having dependencies on disabled layers. This might not apply in "S" modes.

Issue 14: SVC Features vs. WebRTC Simulcast

- What features of WebRTC simulcast do we need to support with SVC?
  - **Spatial layer activation/deactivation (active)**
  - **Bitrate limitation (maxBitrate)**
  - **Temporal layer activation/deactivation**
- Do we care about feature parity between “S” mode simulcast (single stream) and WebRTC simulcast?
      - Active, maxBitrate, maxFramerate?
- Non-goals
  - Arbitrary scaling ratios: can be handled via new scalabilityMode values
  - Support for maxFramerate for temporal scalability
  - Support for custom prediction structures (see previous slides)

Issue 14: Encoding parameters for spatial layers (cont'd)

- Proposal 1: Array of spatial coding parameters nested in the RTCRtpEncodingParameters dictionary, one entry per layer.
- Proposal 2: Augment proposal 1 with temporal weights
- Proposal 3: Support for spatial scalability within RTCRtpEncodingParameters (no nesting)

Issue 14: Proposal 1 (cont'd)

```
partial dictionary RTCRtpEncodingParameters : RTCRtpCodingParameters {
   DOMString scalabilityMode;
   sequence <RTCRtpLayerParameters> spatialLayers;
};

dictionary RTCRtpLayerParameters {
    unsigned long   maxBitrate;
    boolean         active = true;
};
```

- Pros:
  - Reuses existing attributes: `maxBitrate, active`
- Cons:
  - Applies only to spatial layers
  - Cannot activate/deactivate temporal layers
  - Does not achieve parity with multi-SSRC simulcast
    - Example: simulcast with full res/framerate, half res/half maxFramerate (thumbnail)
  - Cannot configure all AV1 prediction structures

Issue 14: Proposal 2 (cont'd)

```
partial dictionary RTCRtpEncodingParameters : RTCRtpCodingParameters {
   DOMString scalabilityMode;
   sequence <RTCRtpLayerParameters> spatialLayers;
};

dictionary RTCRtpLayerParameters {
    unsigned long   maxBitrate;
    boolean         active = true;
    Sequence <float> temporalLayerWeights;
};
```

- Pros:
  - Enables allocation of bandwidth between temporal layers for each resolution
  - Enables deactivation of temporal layers (weight = 0)
- Cons:
  - Cannot configure all AV1 prediction structures

Issue 14: Proposal 2 Example (cont'd)

```
var encodings = [
  {
    scalabilityMode: 'L3T3',
    maxBitrate: 600000,
    maxFramerate: 30,
    scaleResolutionDownBy: 2,
    active: true,
    spatialLayers: [
       {active: true, maxBitrate: 50000, temporalLayerWeights:
[0.6, 0.2, 0.2 ] },
       {active: true, maxBitrate: 150000, temporalLayerWeights:
[0.5, 0.4, 0.2 ] },
       {active: false, temporalLayerWeights: [0.6, 0.3, 0.3 ] },
    ]
   }
];
```

Issue 14: Proposal 3 (cont'd)

```
partial dictionary RTCRtpEncodingParameters : RTCRtpCodingParameters {
    DOMString              encodingId;
    sequence<DOMString>  dependencyEncodingIds;
    double                 scaleFramerateDownBy;  // For temporal scalability
};
```

- Based on RFC 5583 "Signaling Media Decoding Dependency in SDP"
- Pros:
  - Compatible with existing attributes: `maxBitrate`, `scaleResolutionDownBy`, `active`
    - `scaleFramerateDownBy` needed for temporal modes
  - Can describe hierarchical picture structures (LxTy)
- Cons:
  - Error-prone
    - `scaleFramerateDownBy` cannot take arbitrary values
  - Cannot configure all AV1 prediction structures

Privacy Issues
(Youenn, 45 minutes)

Privacy Issues

- Media Capture and Streams
  - Issue 612: Move enumerateDevices behind permission (Youenn)
  - Issue 607: Fixed, per-origin, device ID creates tracking risk (Youenn)
- Audio Output Devices
  - Issue 83: Selecting audio output in case device info permission is not granted (Youenn)
- WebRTC-Stats
  - Issue 374: Exposing RTCIceCandidateStats.networkType might trigger fingerprinting (Youenn)
- Media Capture from DOM Elements
  - Issue 68: Investigate and document the fingerprintability of user media rendering (Youenn)
- WebRTC-PC

Issue 612: Move enumerateDevices behind permission (Youenn)

- Problem: enumerateDevices is providing fingerprinting information
  - The number of devices and persistent device ID values
- Solution: Stop exposing this information if 'device-info' permission is not granted
- Proposal 1
  - Expose at most one device of each type, device ID values are left empty
  - Web compatible: Safari ships this approach
- Proposal 2
  - Expose exactly one device of each type, device ID values are left empty
  - Expose proposal 1 list after one getUserMedia call
    - Promise rejected with *NotFoundError* if there is no such device
- Alternate solution: put enumerateDevices behind a prompt
  - Difficult to implement, difficult to ship

Issue 607: Fixed, per-origin, device ID creates tracking risk (Youenn 1/2)

- **Problem**
  - **Device IDs are persistent**
    - **Can potentially be used to do cross-site tracking**
- **Current mitigations**
  - **Device IDs cleared whenever any other persistent data is being cleared**
  - **Device IDs not exposed to cross-origin iframes by default**
    - **Opt-in using feature policy**
  - **Device IDs not exposed if iframe is not granted 'device-info' permission (?)**
- **Solution**
  - **Partition device IDs as done for other persistent data**

Issue 607: Fixed, per-origin, device ID creates tracking risk (Youenn 2/2)

- Proposal
  - Implementation MUST partition device IDs if other data is partitioned
  - Add a note that future spec versions will require device ID partitioning
    - And/or implementation SHOULD partition device IDs
- Why not mandating partitioning to all user agents?
  - Difficult to ship
    - It might confuse web applications storing device IDs for later reuse
  - Not effective
    - Web applications may use localStorage anyway
- Why not using some global deviceIds?
  - Proposal of using 1,2,3,4... is not 100% solving the issue
    - A '1,4,5' list would be user-specific
    - Cannot clear the global list if clearing data from a specific origin only

Issue 374: Exposing RTCIceCandidateStats.networkType might trigger fingerprinting (Youenn)

networkType reveals whether a candidate is “wifi”, “ethernet”, “vpn”, etc.

- **Main use case is to do bad connection analytics and identify root causes of network issues**

Problem

- **This increases the fingerprinting surface**
- **This could be misused to try optimizing the service based on this information**
- **The user does not need to provide this information for the website to provide the desired service**

Proposal A

- **Move this stat in an extension spec**
  - **Note :This does not require any change to existing implementations**
    - **Implementations shipping this stat can still expose this information and be compliant**
- **Add guidance about when this stat field should be generated in the extension spec**
  - **For instance, only expose this stat for the selected candidate pair**

Proposal B

- **Close this issue: a fingerprinting note has already been added to the network type**

Proposal C

- **Add guidance about when “unknown” should be used.**
  - **… such as only when getUserMedia() permissions have been granted**

Issue 83: Selecting audio output in case device info permission is not granted (Youenn 1/2)

- Selecting device output is currently tied to a getUserMedia call
  - This is an important limitation we should try to remove
- Some devices could be exposed without too much issues
  - Expose output devices already known from user agent strings
    - Phones earpiece and loudspeaker
- Some applications want a specific output behavior not a specific device
  - Phone apps may want to use loudspeaker for ringtones and whatever most convenient output device for the actual phone call
- Solution
  - Allow applications to select audio output for these known cases
- Proposal
  - Make setSinkId understand predefined values: 'earpiece', 'loudspeaker', 'phone-like'...

Issue 83: Selecting audio output in case device info permission is not granted (Youenn 2/2)

- Some applications allow user selection of specific output devices
- Currently implemented on top of enumerateDevices, so tied to getUserMedia
  - Label-based UI works but could be improved (device type UI e.g.)
- Solution
  - Add a new method to ask user to select an output device
- Proposal
  - Promise<MediaDeviceInfo> requestAudioOutput(optional constraints)
    - User Agent is responsible to do the prompt
- Similar APIs
  - getDisplayMedia to select a screen source
  - webkitShowPlaybackTargetPicker to select an AirPlay device

Issue 68: Investigate and document the fingerprintability of user media rendering (Youenn)

- HTMLMediaElement.captureElement()
  - Exposes internals of video rendering
    - Error concealment for instance (WebRTC pipeline or not)
  - Already exposed by rendering video in a canvas
    - But can increase the fingerprinting accuracy as each frame is potentially captured
  - MediaStream is created synchronously but the video frames can be pushed asynchronously
    - Can prompt the user before exposing data, ending a track is also possible at any point
- MediaRecorder API
  - Exposes internals of video encoding
    - Hardware encoder/software encoder determination and differences
    - Encoder configuration might be different than encoders exposed by WebRTC
      - MediaRecorder might also expose more encoders than WebRTC
  - Exposes internals of video packaging
    - Should be OS generic (?)
  - Ability for a user agent to fire events asynchronously
    - Can prompt the user before firing events

Break
(10:15 AM - 10:45 AM)

WebRTC-NV Use Cases (Bernard, 30 minutes)

Open Issues

- Total: 14
- Breakdown by topic:
  - TPAC 2019: 2
  - Ready for PR: 2
  - PR exists: 1

WebRTC-NV Use Cases Issues

- Issues
  - **Issue 53: Advanced Codec Capabilities API (Henrik)**
  - **Issue 37: Requirements for Secure Web Conferencing (bernard)**

Issue 53: Advanced Codec Capabilities API (Henrik) - 1/3

Encoding/decoding is expensive. More information about encoder/decoder implementation capabilities would better support low-end devices or high-end use cases such as low latency requirements (e.g. gaming).

Use Case A: “Hard” capabilities:

- Is the encoder hardware accelerated?
- What are the min and max supported resolution? (HW limits)
- Are there limits to the number of simulcast layers? (Limit on instantiating certain number of encoders)
- Does it support SVC? Which modes?

Issue 53: Advanced Codec Capabilities API (Henrik) - 2/3

Use Case B: Identifying implementation:

- Which implementation was used?
  - Better debugging.
  - Certain implementations are “bad” for certain use cases, identifying implementations would allow applications to learn about them and avoid them in a way that a User Agent could not.

Use Case C: Expected Performance ← *How would you implement this?*

- Expected frame rate?
- Expected latency?
- Expected bitrate?

Issue 53: Advanced Codec Capabilities API (Henrik) - 3/3

Example API for Use Case A and partially B:

```
interface CodecCapabilities {
    static Promise<sequence<VideoEncoderImplementation>>
        getVideoEncoderImplementations();
}

interface VideoEncoderImplementation {
    DOMString codec;
    DOMString profile;
    DOMString implementation;
    boolean isHardwareAccelerated;
    FrozenArray<VideoCodecMode> videoModes;
}

dictionary VideoCodecMode {
    DOMString scalabilityMode;
    unsigned long minWidth, maxWidth;
    unsigned long minHeight, maxHeight;
}
```

Issue 37/PR 49: Requirements for Secure Web Conferencing (bernard)

- Second attempt to develop requirements for Secure Web Conferencing.
- PR 49: submission by Cullen Jennings
  - One use case where Javascript is trusted
  - Another use case where Javascript is not trusted
  - In both use cases, the conference server is not trusted to have access to cleartext media.

Issue 37/PR 49: Requirements (cont'd)

- Requirements for both use cases (based on MLS Security Architecture):
  - N25: Only current group members can receive media or text sent to the group.
  - N26: A group member cannot send media or text that appears to be from another group member.
  - N27: The conference server must not have access to cleartext media or text or to the identity of group members.
  - N28: Perfect Forward Secrecy (PFS): access to encrypted traffic as well as all current keying material does not compromise the secrecy of media or text older than the oldest key of a compromised client.
  - N29 : Post Compromise Security (PCS). Protection against past or future device compromise.

WEBRTC WG Re-Charter
(Dom, 60 minutes)

Timeline

- Current charter ends 2020-03-31
- Needs ~3 months to approve an updated charter
- ⇒ Needs WG-happy draft by EoY

Obvious charter changes

- WebRTC 1.0 & getUserMedia: done! (? (!))
- Update deliverable timelines
- Update dependencies to other groups
    - (e.g. + Media WG)

Obvious charter changes

- WebRTC 1.0 & getUserMedia: done! (? (!))
- Update deliverable timelines
- Update dependencies to other groups
  - (e.g. + Media WG)

Open questions

- What to do with existing deliverables?
  - Both finished ones and unfinished ones
- Which new deliverables do we want to take up?
- How long this updated charter should be?

Existing deliverables: Recs

- WebRTC 1.0
- Media Capture and Streams

    => 1.1 versions?

Existing Deliverables: ~CR (1/2)

- WebRTC-specific:
  - WebRTC Stats
  - Identity
- Media capture:
  - Audio Device Output
  - Screen Capture
  - Media Recorder
  - MediaStream from DOM Element

Options for ~CR deliverables

- Options:
    - Finish them all
        - Need clear Editors commitment
    - Finish WebRTC-specific ones and move Media Capture ones to someone else (e.g. Media?)
    - Give up some if low momentum
- Special question about WebRTC-stats as registry-like

Existing deliverables: early ones

- Image Capture (advanced camera control)
- Depth cameras
- Content Hints
- WebRTC-dscp
- WebRTC-SVC

Options for early specs

- Options:
  - Finish them all
    - Need clear Editors commitment, implementor interest
  - Finish WebRTC-specific ones and move Media Capture ones to someone else (e.g. Media?)
  - Give up some if low momentum

Future deliverables: WebRTC 1.0 extracts

- OauthCredential
- Streaming data channels
- gUM-less Mode-1 P2P
- (more from features at risk discussion?)

Future deliverables: WebRTC NV use cases

- More granular object model (ORTC-like)
- WebRTC-insertable
- End-to-end encryption
- FlexICE / peer to peer Mesh
- WebRTC-in-worker
- (WebCodecs)
- (WebTransport)
- More ?

Options for future deliverables

- Options:
  - Adopt some
    - Need clear Editors commitment, implementor interest
  - Push to incubation
    - Dedicated WebRTC incubation CG?
  - Find another group

Lunch
12:00 PM - 1:00 PM

Joint Meeting with Accessibility Platform Architectures (APA) WG (60 minutes)

https://www.w3.org/WAI/APA/wiki/Accessible_RTC_Use_Cases

Relationship Between W3C WEBRTC WG and IETF

- W3C WEBRTC WG: develops APIs
- IETF
  - Develops protocols in WGs such as RTCWEB, MMUSIC, AVTCORE, ICE, SLIM, RUM, etc.

Accessibility Work within the IETF ART Area

- Language negotiation (SLIM): RFC 8373, draft-ietf-slim-use-cases
    - Enables SDP negotiation of spoken, written and signed languages between parties.
- T.140 over WebRTC Data Channel (MMUSIC): draft-holmberg-mmusic-t140-usage-data-channel
    - Enables Realtime Text to be sent over the WebRTC data channel.
- Interoperability profile of the Video Relay Service (RUM): https://tools.ietf.org/html/draft-rosen-rue
    - References RTCWEB documents, including JSEP, Overview, RTP Usage, Security Architecture, Transports, RFC 7742 (Video requirements) and RFC 7874 (Audio requirements)
    - History & Background: https://datatracker.ietf.org/meeting/105/materials/slides-105-rum-rum-history-background-00

WebRTC-Stats
(Varun & Henrik, 60 minutes)

webrtc-stats (Varun & Henrik)

- Issue 365: Remove track/stream stats in favor of RTCMediaHandlerStats.mid /
  Issue 470: RTCMediaStreamStats can be made obsolete (Henrik)
- Issue 437: RTCStats.id should not be "predictable" (Henrik)
- Issue 395/PR 482: Add RTCOutboundRtpStreamStats.rid (Varun/Henrik)
- Issue 398: Add RTCEncodingParameterStats (Henrik)
- Issue 401: Add SVC stats in RTC[In/Out]boundRtpStreamStats (Henrik)
- Issue 443: Detecting Video Playback glitches (Henrik)
- Issue 440: Clarify qualityLimitationReason when limited by multiple reasons (Henrik)
- Issue 391: audioLevel can be removed from "track"|"receiver"|"sender" stats (Henrik)
- Issue 448: Audio samples and channels (Henrik)
- Issue 358: identifying the ice generation of a candidate pair (Henrik)
- Issue 376: [DataChannels] Use RTT from sctp in the stats (Henrik)
- Issue 377: [DataChannels] Expose bandwidth or congestion window from the SCTP lib (Henrik)

Issue 365: Remove track/stream stats in favor of RTCMediaHandlerStats.mid / Issue 470: RTCMediaStreamStats can be made obsolete (Henrik)

RTCMediaStreamStats { streamIdentifier, trackIds } no longer make sense. Because:

- **The track-stream relationship is defined by senders/receivers, not by membership of a MediaStream.**
- **With "mid" (#396) you know the relationship between stats objects and senders/receivers.**
- **"track" stats are Obsolete?**

Proposal A:

- **Add streamIdentifiers to "sender" and "receiver" stats.**
    - **Would sender.streamIdentifiers refer to streams set locally or not what has been successfully negotiated? Do we need currentStreamIdentifiers?**
- **Remove RTCMediaStreamStats.**

Proposal B:

- **Just remote RTCMediaStreamStats, who cares about streams?**

Issue 437: RTCStats.id should not be "predictable" (Henrik)

The spec does not say how IDs are generated, it only requires that IDs are the same for the same stats object between calls to getStats().

Problem: Implementations (e.g. Chrome, Firefox) have predictable IDs (e.g. first track attached always has ID “RTCMediaStreamTrack_sender_1”) that are not standardized. Applications might start depending on implementation-specific behavior, causing interoperability problems with other browsers.

Harald: “base64(sha1(predictable-string)) is, to all intents and purposes, unpredictable.”

- I think we need to avoid the same random-looking string between sessions, but this can be avoided with a salt randomly generated for each peer connection.

Proposal:

- RTCStats.ids MUST NOT be predictable by the application. An application MUST NOT be able to guess what ID a particular stats object will have without having looked it up in a prior getStats() call of the same RTCPeerConnection.

Issue 395/PR 482: Add RTCOutboundRtpStreamStats.rid

Problem: There is no way to know which “outbound-rtp” stats correspond to which simulcast layer.

Proposal: Expose the RTCRtpCodingParameters.rid in stats.

DOMString RTCOutboundRtpStreamStats.rid:
Exposes the rid encoding parameter of this RTP stream if it has been set, otherwise it is undefined. If set this value will be present regardless if the RID RTP header extension has been negotiated.

<Update> If we add “encoding-parameters” we would get rid for free (#398), see next slide.

Issue 398/PR 495: Add RTCEncodingParameterStats (Henrik)

Problem: Simulcast stats are incomplete without knowing what parameters are set.

Proposal A: Add RTCRtpEncodingParametersStats as "encoding-parameters".
This is the same as RTCRtpEncodingParameters, including the webrtc-svc extension of it.

```
dictionary RTCRtpEncodingParametersStats : RTCStats {
    DOMString outboundRtpId;    // One outbound-rtp : one encoding, if simulcast is used.
    DOMString rid;
    DOMString codecId;          // Stats reference instead of "octet codecPayloadType;"
    RTCDtxStatus dtx;
    boolean active;
    unsigned long ptime;
    unsigned long maxBitrate;
    double maxFramerate;
    double scaleResolutionDownBy;
    DOMString scalabilityMode;  // From webrtc-svc
}
```

Proposal B: Same members, but add them directly to the "outbound-rtp" object.

Issue 401: Add SVC stats in RTC[In/Out]boundRtpStreamStats (Henrik)

Problem: There are no SVC-specific stats. It would be useful to be able to tell…

- **Resolution**
- **Frame rate**
- **Bitrate**

Proposal: Add “svc-layer” stats objects; multiple ones per “outbound-rtp” or “inbound-rtp”.

```
dictionary RTCScalableVideoCodecLayerStats : RTCStats {
    DOMString           outboundRtpId or inboundRtpId;  // or rtpId ?
    unsigned long       layer;
    unsigned long       width;
    unsigned long       height;
    unsigned long       framesSent or framesReceived;  // or framesTransmitted ?
    unsinged long long bytesSent or bytesReceived;    // or bytesTransmitted ?
}
```

The metrics would represent the latest information sent or received.

Issue 443: Detecting Video Playback glitches (Henrik)

Problem:

- With the current metrics it's hard to distinguish glitches from low frame rates.
- What is perceived as a "glitch" is different for different frame rates (200 ms inter-frame delay of 5 fps video is great, but for 30 fps it is terrible); we cannot define a "glitch" as X ms inter-frame delay.

Proposal A:

- Add RTC[In/Out]boundRtpStreamStats.interFrameDelay:
  The max inter-frame delay that was recorded during the last second. An inter-frame delay is measured by taking the time between two consecutive frames, or the time between the last frame and the current time. In other words, if a frame has not been received in half a second or more, interFrameDelay would be the same as the time since the last frame was received.

Proposal B:

- totalGlitches and totalGlitchesDurations, where a "glitch" is defined to be strong deviation from the average frame rate.
  - E.g. 100% greater delay than expected, based on avg FPS, where avg FPS = last second if at least 10 frames arrived, or based on last 10 frames otherwise?

Issue 440: Clarify qualityLimitationReason when limited by multiple reasons (Henrik)

Problem: RTCQualityLimitationReason is an enum: “cpu”, “bandwidth”, “other” or “none”.

- **What do you report if you are both CPU and Bandwidth limited?**
- **Reasons are not independent: lowering resolution affects both CPU and Bandwidth usage.**
- **The implementation probably only knows you are limited once you exceed the limit, so the limit is just an estimate. It probably does not reliably know what the limit of each reason is.**

Proposal A:

- **Chrome’s current implementation: Prefer to report “bandwidth” over “cpu” if both are detected.**

Proposal B:

- **Say that an implementation SHOULD report the most limiting factor.**
  This makes it a best-effort and different implementations would be more likely to report different values under the same circumstances.

Issue 391: audioLevel can be removed from "track"|"receiver"|"sender" stats (Henrik)

The case against audioLevel: We have getSynchronizationSources() which is efficient, and don't want to encourage people from polling getStats() frequently. However…

- **getSynchronizationSources() is only for the sender, not the receiver.**
- **totalAudioEnergy (and friends) allows calculating average audio levels over intervals; polling is not needed. audioLevel is based on audio energy so it's easy to implement if you have energy.**
- **audioLevel existing in legacy goog-stats but not in standard stats was encouraging people to continue to use the legacy API, so this has now been implemented in Chrome's standard API.**

Proposal A:

- **Keep audioLevel - close this issue.**
- **Follow-up issue: Clarify it to make it interoperable? It's an average level based on an "implementation dependent" interval!**

Proposal B:

- **Move audioLevel to the Obsolete section in favor of totalAudioEnergy.**

Issue 448: Audio samples and channels (Henrik)

Problem: The spec neglects the notion of channels; our sample counters and related metrics like totalAudioEnergy and totalSamplesDuration does not say what to do if you have >1 channels.

- **Current implementations neglect channels too! totalSamplesDuration increase one second per second! Even if you have samples in both Left and Right channels.**
- **Correctly measuring things on a per-channel basis requires per-channel metrics.**
- **Modifying the existing metrics *might* cause regressions (e.g. all counters twice as high for 2 channels).**

Solution:

- **Samples counters and sample duration: When processing a frame, increment counters based on the samples of any one of the channels. Multiple channels does not make any counter increase faster.**
- **totalAudioEnergy: When processing a frame, increment by the energy of the channel with the highest energy.**
- **echoReturnLoss: When processing a frame, increment by the ERL of the channel with the lowest ERL value.**

If we want per-channel metrics in the future those should be added as new metrics (separate issues).

Issue 358: identifying the ice generation of a candidate pair (Henrik)

Problem: The same candidates can be generated after an ICE restart, and there is no way to tell which ICE generation a candidate belongs to.

Proposal A:

- **Add iceUfrag to RTCIceTransport (which is referenced by candidates and pairs).**

Proposal B (not mutually exclusive with Proposal A):

- **Clarify that new candidate objects are to be generated after ICE restart, even if they represent the same address. This would imply that there is a bug in Chrome if an ice candidate has the same ID before and after an ICE restart?**

Issue 376: [DataChannels] Use RTT from sctp in the stats (Henrik)

We don’t have any SCTP stats. RTT measurements are available through RTCP (requires RTP) and ICE, but Firefox does not support RTT with ICE, and ICE pings might not be as accurate for SCTP.

- **Exposing SCTP-based RTT may allow more accurate RTT measurements.**

Proposal: Add “sctp-transport” stats!

```
dictionary RTCSctpTransportStats : RTCStats {
    double roundTripTime;
}
```

roundTripTime is the current smoothed round-trip time, in seconds, based on spinfo_srtt in RFC6458.

- **Alternatively: Is there a total RTT measurements and number of measurements we can reference? The above RFC reference does not say how this is calculated.**

Issue 377: [DataChannels] Expose bandwidth or congestion window from the SCTP lib (Henrik)

… do we want other SCTP stats?

spinfo_cwnd: This contains the peer address' current congestion window.

Discussion: Do we want to expose this?

- **@shacharz: "It can be very helpful for applications to get the bandwidth/c_wnd from the SCTP layer, so it can adjust business logic accordingly." (comment)**
  - **Is this needed given a (non-buggy) bufferedAmount / onbufferedamountlow?**
  - **We don't want to encourage frequent getStats() polling.**

Content-Hints (Harald, 30 minutes)

Content-Hints API (Harald)

- Issues
    - Issue 2248: degradationPreference is under-specified (bernard)
    - Issue 28: Redundancy and lack of normative clarity around interaction with constraints (Jan-Ivar)
    - Issue 30: Permanence Issues (Jan-Ivar)

Issue 2248: degradationPreference is under-specified (bernard)

- Effect not easily tested
  - WPT only tests the ability to set and get the value of the degradationPreference attribute.
  - Effect not easily determined in a loopback test
- Disparate implementations
  - Chrome: degradationPreference has no effect, currently.
    - C/C++ level: only used to decide whether to reduce resolution or framerate in the event of congestion
  - Current Edge: treats degradationPreference similarly to a content-hint
    - “Prefer-resolution” equivalent to “detail” content-hint
    - “Prefer-framerate” equivalent to “motion” content-hint
- Recommendation
  - Include degradationPreference as a “feature at risk”
  - Add clarification that degradationPreference is purely about the resolution/framerate tradeoff

Issue 28: Redundancy & lack of normative steps; interaction with constraints (jib)

“Hints” with unspecified behavior favor dominant browser implementation. Everyone else must reverse engineer. A missed opportunity to make things normatively testable.

Mentions "Echo-cancellation", "noise suppression", "boost intelligibility of the incoming signal" (autoGainControl?)

Hints seem somewhat useful as a simpler API to constraints:

```
// Assuming unconstrained audio track
track.contentHint = "music";
console.log(track.getSettings().echoCancellation); // false?
console.log(track.getSettings().noiseSuppression); // false?
console.log(track.getSettings().autoGainControl);  // false?

await track.applyConstraints({echoCancellation: true});

console.log(track.getSettings().echoCancellation); // true
console.log(track.getSettings().noiseSuppression); // false?
console.log(track.getSettings().autoGainControl);  // false?
```

Bonus: more directly controls “default” values Today latter ones tend to track echoCancellation when absent.

Q: Which spec should specifies this?

Issue 30: Permanence Issues (Jan-Ivar)

Hints are ***not*** inherent properties of a track e.g. a `"music"` track; a `"motion"` video; invariant and ever-present:

1. **They're a control surface**, a runtime knob. JavaScript can modify them at any time, expecting results, yet results are not specified anywhere, nor when observable effects may be expected, if any.

2. **They don't follow the media** i.e. track replication through sink → source pipes like peer connection, element.captureStream(), web audio, MediaRecorder, or track.clone() (or do they)? Spec doesn't say.

3. **They may be wrong.** User agents may (someday) detect speech vs. music at run-time. Are they allowed to ignore bad hints? If they're ignored, what happens to any observable (testable) effects/settings we define? If user agents *can't* ignore them, are they a footgun API? Misnomer? Option: allow ignoring "in the future"?

How should browsers behave if the JS twiddles the bit live over time?

How does it work with track.clone()? Can each clone have its own value?
Is JS expected to tack these hints back on like post-it notes that keep falling off?

Should normative language live in the contentHint spec, or be pushed to individual specs using contentHints?
Spec should probably give guidance to other (future) specs at least, to ensure consistency.

Wrapup and Next Steps (Harald, 60 minutes)

Conclusions

We have minutes of the meeting.

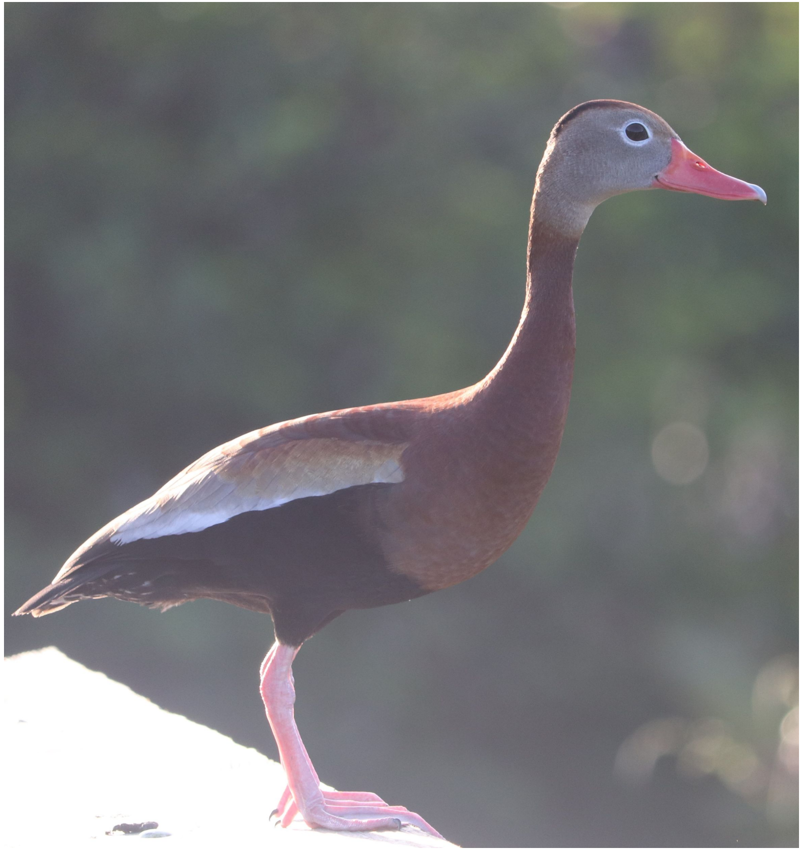
We are slogging forward.

We have documents that need to go to wide review, TAG review and advancement (e.g. CR).

We have gone through “features at risk”.   Current consensus is to delete most of the features at risk except maxFramerate and RTCError (needs more discussion).

Identity. Need to move isolation properties out of Identity into Media Capture.

For extra credit



Name that bird!

Thank you

Special thanks to:

WG Participants, Editors & Chairs

The bird