

W3C WebRTC WG Meeting

Stockholm, Sweden
June 19, 2018

Chairs: Stefan Hakansson

Bernard Aboba

Harald Alvestrand

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 face-to-face meeting of the W3C WebRTC WG!
- During this meeting, we hope to:
 - Make progress on WebRTC NV
 - Discuss issues relating to WebRTC 1.0 such as testing, identity and statistics.

TPAC (Lyon, France)

We will meet Monday October 22 and Tuesday the 23rd of TPAC week.

About this Meeting

Information on the meeting:

- Meeting info:
 - https://www.w3.org/2011/04/webrtc/wiki/June_19-20_2018
- Link to latest drafts:
 - <https://w3c.github.io/mediacapture-main/>
 - <https://w3c.github.io/webrtc-pc/>
 - <https://w3c.github.io/mediacapture-screen-share/>
 - <https://w3c.github.io/webrtc-stats/>
- Link to Slides has been published on [WG wiki](#)
- Scribe? IRC <http://irc.w3.org/> Channel: [#webrtc](#)
- The meeting is being recorded.
- WebEx info [here](#)

Proposed Agenda for June 19

9 AM - 10:00 AM: WebRTC NV use cases and requirements (Peter Thatcher)

10:00 - 11:00 AM General direction (Peter Thatcher)

- a. Low-level versus high-level APIs
- b. Backward compatibility with WebRTC 1.0 and transition to NV

11 AM - 11:30 AM Break

11:30 AM - noon: WebRTC in Workers (Tim Panton)

noon PM - 12:30 PM Encoding/Decoding and RTP (Peter Thatcher)

12:30 PM - 1 PM Access to Raw Media (Harald Alvestrand)

1:00 PM - 1:30 PM Lunch

1:30 PM - 2:30 PM ICE

- a. [ORTC ICE API](#) (Bernard Aboba)
- b. [WebRTC-ICE](#) proposal (Peter Thatcher)
- c. FlexICE/SLICE transport proposals (Peter Thatcher)

2:30 PM - 3 PM Identity (Harald)

3 PM - 3:30 PM Permissions and WebDriver (Dr. Alex)

3:30 PM - 5:00 PM WebRTC 1.0 Testing (Dr. Alex and team)

Proposed Agenda for June 20

9:00 AM - 9:30AM: Transports (Peter Thatcher)

- a. Potential data channel transports: SCTP, QUIC, RTP
- b. Potential media transports: RTP, QUIC

9:30 AM - 10 AM Use Case Reprise - includes API level discussion (Peter Thatcher)

10:00 AM - 11 AM QUIC (Peter Thatcher)

Reference: <https://w3c.github.io/webrtc-quick/>

11 AM - noon: SCTP, Data Channel and Streams (Lennart Grahl)

Noon - 12:30 PM: Scalable Video Coding (Sergio and Bernard)

12:30 PM - 1 PM: E2E Security Drill-down (Goran, Youenn & Dr. Alex)

1PM - 1:30 PM: Lunch break

1:30 PM - 2 PM: Protocol dependencies (Stefan)

2 PM - 2:30 PM: WebRTC-Stats (Varun)

2:30 PM - 3:00 PM: Worker reprise (Peter Thatcher)

3:00 PM - 4:00 PM: Open slots

4 PM - 5 PM Wrapup and next steps

Use Cases and Requirements (Peter Thatcher)

Things already happening (WebRTC has had wild success)

- Mobile
- Video conferencing that wants to be higher quality and secure
- File sharing (WebTorrent, IPFS, ShareDrop)
- Browser<->server web games (using data channel)
- VR communications
- Remote control (Screenhero)
- Video (live) server to/from browser
- Browser<->devices (casting, IoT)
- SFU in JS (<https://mediasoup.org/>)
- Lenses/Filters/Fake Bokeh
- "Ultra low" latency audio (playing music together)

Existing use cases, but better

- Browser <-> Browser 1:1 call
- Browser <-> Mobile app 1:1 call
- Browser <-> Browser mesh call
- Browser <-> Browser mesh game
- Browser <-> Server-based multiway audio/video call

Existing, improved use case: Server-based multiway audio/video call

Example: WebEx, Hangouts

New requirements:

- App can enable and control SVC and temporal scalability
- App can send without negotiating codecs
- App can receive without setting up senders
- App can send and receive data (for example, live text or files) without clobbering audio/video congestion control

New use case: Browser<->devices

Example: Cast, W3C Second Screen WG, Lego robot

New requirements:

- App can minimize ICE connectivity checks or eliminate them entirely (battery issue)
- App can control ICE networks used (only use wifi, not cell)
- App can make ICE work with local broadcast bootstrap (ICE forking?)
- App has low-level control of media transport (low-level RTP or QUIC)
- Media protocol is easy to terminate on a device
- Congestion control shared between audio/video/data

New use case: File sharing, CDN, DHT

Example: WebTorrent, IPFS

New requirements:

- (DHT) App can send/recv data in a way that does not compete aggressively with audio/video
- (DHT) App can "pause" and "resume" ICE connections (control ICE checking interval)
- (DHT) App can listen for incoming ICE connections (ICE forking; control port lifetime)
- App can easily send large files without buffering too much and without having the throughput drop badly

New use case: Client <-> server games

Example: Agar.io, Diep.io

New requirements:

- Data channel uses minimal/zero buffering when sending at a low, fixed-frequency rate.
- Data channel is easy to terminate on a server (ideally with protocol implementations that are open source and proven in the wild)
- Data channel has congestion control that works well with audio/video (ideally the same congestion control and with the same protocol)
- Data can be synchronized with audio and video

New use case: VR communications

Example: Facebook VR Chat

New requirements:

- Data channel is easy to terminate on a server
- Data channel has congestion control that works well with audio/video (ideally the same congestion control and with the same protocol)
- Data can be synchronized with audio and video
- Advanced encoding control: for example, depth@15fps, color@90fps

New use case: Remote control

Example: Screenhero,

New requirements:

- Data channel has congestion control that works well with audio/video (ideally the same congestion control and with the same protocol)

New use case: Video (live) server to/from browser

Example: Facebook Live, millicast.com

New requirements:

- Media protocol is easy to terminate on a server. Ideally, allow app to control the wire format and transport to match existing server infrastructure
- App has advanced control over retransmissions and FEC
- App has advanced control over jitter buffer
- Share congestion control context with HTTP?

New use case: Lenses/Filters/Fake Bokeh/Funny Hats

Example: Snapchat

New requirements:

- App can modify raw media before encode
- App can modify decoded media before render/playout

New use case: end-to-end encrypted video conference

Example: WebEx, Hangouts, but the server doesn't see unencrypted media

New requirements:

- App can either
 - A. add end-to-end encryption on top of hop-by-hop encryption
 - B. change the encryption key to something other than from DTLS-SRTP with server endpoint
 - (B seems like the only realistic option)
- App can allow server to see some media metadata unencrypted while keeping media encrypted

New use case: SFU in JS

Example: mediasoup.org

New requirements:

- App can cause take media from one endpoint and send it to another without reencoding (for example, a media transport that can send and receive encoded media)
- App can control over what encoded media is forwarded and which is dropped
- App can control back-propagation or generation of key frame requests

New use case: "ultra low" audio

Example: ultragrid.cz (playing music together)

New requirements:

- Disable or severely shrink audio jitter buffer
- Disable audio codec

Potential new use cases

- ? end-to-end-encrypted video conference
- File sharing, CDN, DHT
- Client <-> server games
- Server -> browser media
- Browser -> server media
- Browser <-> device media/data
- X SFU in browser
- VR
- X? Funny hats
- XX Ultra-low-latency audio
- Added at f2f: ML analysis, synchronized overlay, text with audio and data

Potential new requirements for media/data

- Spatial and temporal scalability
- e2ee mechanism
- Separate encoder/decoder from transport
- Combined audio/video/data under friendly or the same CC
- SCTP transport w/ more controls
- QUIC transport
- SSRC selection
- Control over jitter buffer
- BYO RTX, FEC, packetization, codec, jitter buffer
- Work in web workers
- Data channel back pressure

New requirements for ICE

- Candidate local gathering control (wifi/cell)
- Control over local candidate lifetime (retaining it, closing it)
- Control over checking frequency (or pausing it)
- reselection/nomination
- forking

Questions for the WG

Which existing use cases do we want to improve on?

Which new use cases do we want to add?

Which new requirements do we want to take on?

WebRTC NV API Level

6/2018 f2f

Big picture

1.0 was

one big class

does everything

sometimes controlled by the app

NV is

many components

each with a purpose

coordinated by the app

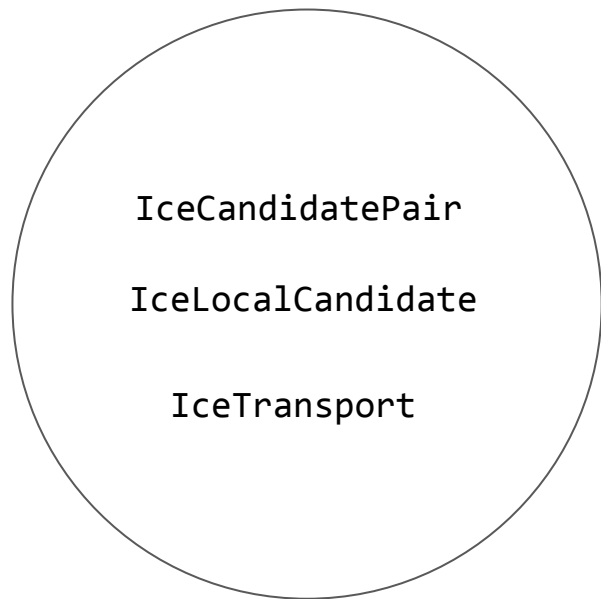
Big picture

1.0 was
one big class
does everything
sometimes controlled by the app

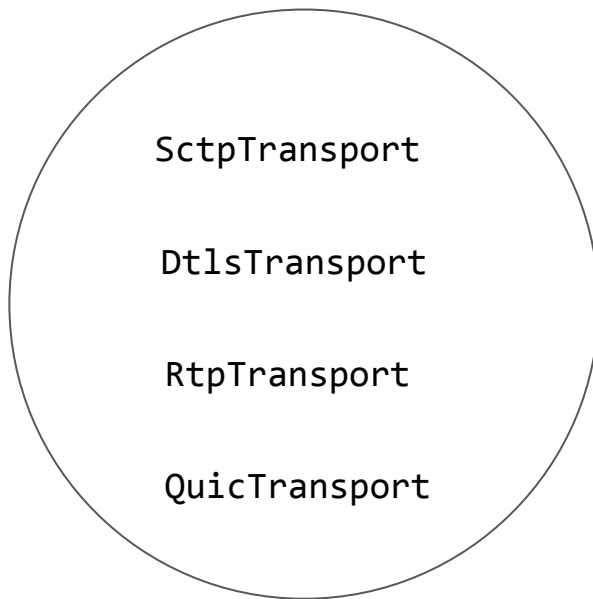
NV is
many components
each with a purpose
coordinated by the app

But how low-level of components?

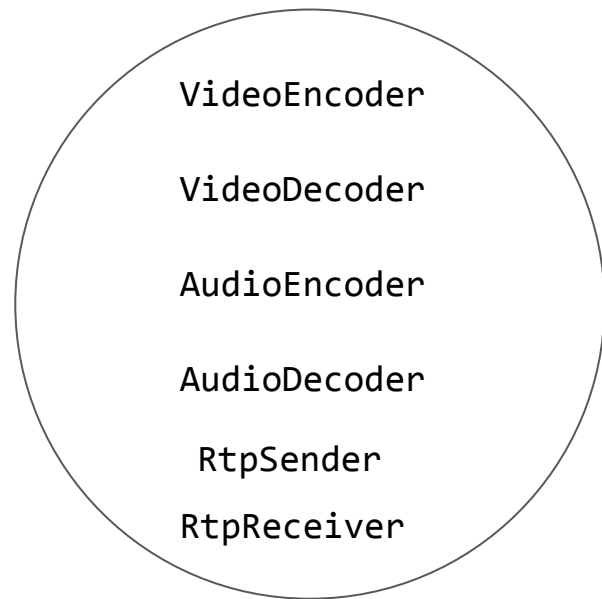
3 types of components



Network



Transport



Media

Background

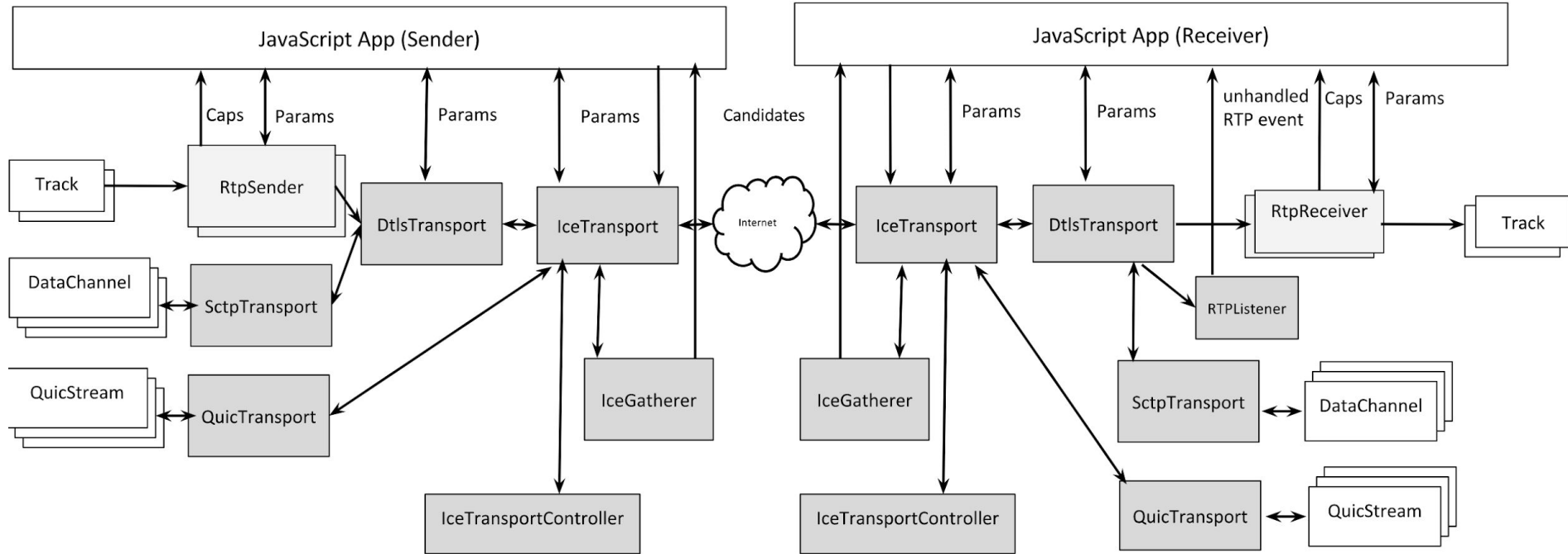
We know we need to go lower than 1.0.

But how low?

ORTC is lower than PeerConnection.

Is it low enough?

ORTC Object Model



Source: <http://draft.ortc.org/>

How low is too low?

Having the app do SRTP or congestion control is too low-level.

We can't trust the app to get those right.

What options are in between?

Let's see what we have to choose from, going from high-level to low-level.

RTP

Simple things simple
Less control

More control
More complex



RTP

A. PeerConnection: Mix of direct control and SDP (👁️)



Simple things simple
Less control

More control
More complex



RTP

A. PeerConnection: Mix of direct control and SDP (👤)



B. ORTC: all direct control; no SDP (👨‍🔧)



Simple things simple
Less control

More control
More complex



RTP

A. PeerConnection: Mix of direct control and SDP (👤)



Simple things simple
Less control

B. ORTC: all direct control; no SDP (👋)



C. Split out encoder/decoder from RtpSender/RtpReceiver (pluggable transport, encoder)



More control
More complex



RTP

A. PeerConnection: Mix of direct control and SDP (👤)



Simple things simple
Less control

B. ORTC: all direct control; no SDP (👋)



C. Split out encoder/decoder from RtpSender/RtpReceiver (pluggable transport)



D. Low-level RTP control (bring your own RTP packetization, FEC, RTX, e2e crypto, ...)



More control
More complex

RTP

A. PeerConnection: Mix of direct control and SDP (👤)



Simple things simple
Less control

B. ORTC: all direct control; no SDP (👋)



C. Split out encoder/decoder from RtpSender/RtpReceiver (pluggable transport)



D. Low-level RTP control (bring your own RTP packetization, FEC, RTX, e2e crypto, ...)



E. Max wasm (bring your own codec, jitter buffer): need raw media access



Pros and cons

ORTC allows for SVC, control of temporal scalability and avoidance of SDP. But it requires app to handle signaling and negotiation.

Spilt codecs/transport additionally allows for media over QUIC or SCTP, client SFU, some forms of e2ee, and potentially BYO codecs or jitter buffer. Those things require the app to be in the media path, but the app doesn't have to be in the media path if it doesn't want to (opt-in).

A low-level RTP transport additionally allows e2ee, BYO FEC, BYO RTX, BYO packetization, and RTP data. Those things require the app to do more, but the app doesn't have to if it doesn't want to (opt-in).

If we can allow opt-in replacement, Option D seems like the sweet spot

ICE

Simple things simple
Less control

More control
More complex



ICE

A. PeerConnection: Not standalone, uses SDP (☹)



Simple things simple
Less control

More control
More complex



ICE

A. PeerConnection: Not standalone, uses SDP (👹)



B. WebRTC-ICE extension spec: standalone, no SDP (👹)



Simple things simple
Less control

More control
More complex



ICE

A. PeerConnection: Not standalone, uses SDP (🤖)



B. WebRTC-ICE extension spec: standalone, no SDP (👉)



C. ORTC: split gatherer/transport (supports forking)



Simple things simple
Less control

More control
More complex



ICE

A. PeerConnection: Not standalone, uses SDP (🤖)



B. WebRTC-ICE extension spec: standalone, no SDP (👉)



C. ORTC: split gatherer/transport (supports forking)



D. FlexICE



Simple things simple
Less control

More control
More complex



ICE

A. PeerConnection: Not standalone, uses SDP (🤖)



B. WebRTC-ICE extension spec: standalone, no SDP (👉)



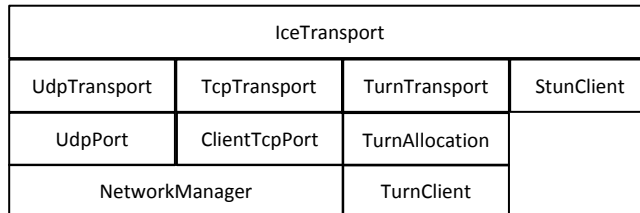
C. ORTC: split gatherer/transport (supports forking)



D. FlexICE



E. SLICE



Simple things simple
Less control

More control
More complex



Pros/cons

WebRTC-ICE allows for constructing an ICE agent separate from a PeerConnection, without SDP, suitable for use with other transports (QuicTransport).

ORTC also allows parallel forking. But it also requires more objects to be managed by the app (IceGatherer)

FlexICE also allows for control over ICE checking frequency/pausing, local candidate lifetimes, continual gathering, and timeouts for disconnectivity. But it requires the app to do more to get the advanced uses (you just get normal behavior otherwise). New controls can be added incrementally.

SLICE allows the app to implement a custom ICE stack with the lowest-level primitives possible. But it requires the app use a library, because no one will be implementing their own ICE stack. Apps that don't need advance things can just use a high-level IceTransport. Those that do are capable of tweaking everything.

Questions for the WG

We'll get into RTP and ICE specifics later, but right now, in general, how low do we want to go?

Backwards compatibility with 1.0

On the wire, ICE, DTLS, SCTP, and RTP should all be compatible, with the exceptions of:

- RtpTransport would allow apps to implement incompatible packetization formats
- SLICE and FlexICE would allow apps to implement an incompatible ICE stack if it tried hard enough
- Codecs implemented in JS/wasm might not be compatible with existing endpoints

Backwards compatibility with 1.0

In the API:

- PeerConnection as-is will remain (for now, at least)
- We have a lot of objects with the same name but different controls (IceTransport, RtpSender). It would make the most sense to just define new interfaces. But then what do we name them? And can you pass an IceTransport from PeerConnection into an NV DtlsTransport/RtpTransport?
- Big question: do we want to require everything PeerConnection can do to be implementable using new NV structure (I hope not :):
 - Non-muxed RTCP
 - ICE agent with multiple components/streams

WebRTC in Workers (Tim Panton)

- <content goes here>

WebRTC NV

Encoders/Decoders

6/2018 f2f

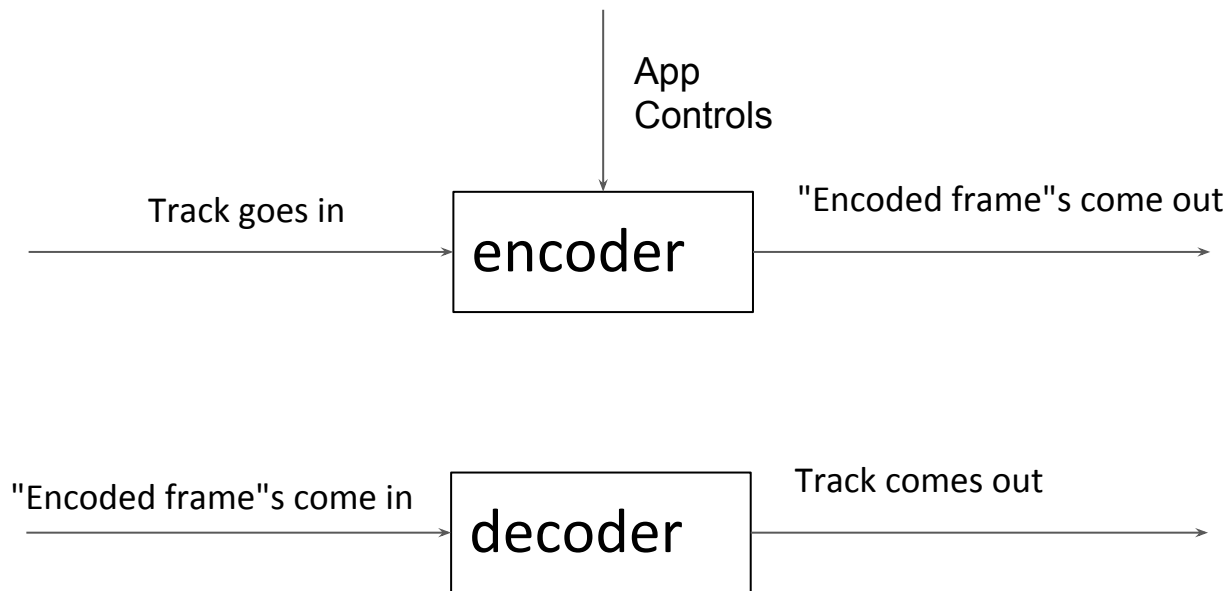
Big Picture



Uses cases for splitting codec and transport

- e2ee: app can add encryption between encode and transport
- browser SFU: app can pull encoded media from transport and put in another
- browser to/from server: can send media over QUIC
- gaming, VR, file share during video conference: can put audio, video and data over the same transport (QUIC or SCTP)
- improved quality: BYO FEC, RTX, jitter buffer, or codec

What are encoders and decoders?



What is an encoded frame?

Video: the encoded pixels at a given point in time, plus metadata (height, width, timestamp, codec type, ...)

Audio: the encoded samples over a range of time, plus metadata (time range, sample count, number of channels, codec type, ...)

What can the app control?

Audio Encoder:

- ptime
- bitrate
- DTX
- in-band FEC

VideoEncoder:

- bitrate
- scaleResolutionDownBy
- maxFramerate

How does media get transported?

Two options:

A. Be in the media path:

1. Pull encoded frame from encoder
2. Do something with it (serialize, packetize, FEC, e2ee)
3. Hand it over to transport/RtpSender

B. Not be in the media path

1. Connect the encoder to the RtpSender. Perhaps `RtpSender.setEncoder(encoder)`, perhaps WHATWG streams.

AudioEncoder

```
interface AudioEncoder {  
    // Can be called any time to change parameters  
    void start(MediaStreamTrack rawAudio, AudioEncodeParameters encodeParameters);  
    void stop();  
    attribute EventHandler onEncodedAudio; // of EncodedAudioFrame  
    // Alternative, WHATWG-stream-style:  
    readonly attribute ReadableStream encodedAudio; // of EncodedVideoFrame  
}
```

```
dictionary AudioEncodeParameters {  
    unsigned long frameLength; // aka ptime, in ms  
    unsigned long bitrate;  
    // ...  
}
```

EncodedAudioFrame

```
dictionary EncodedAudioFrame {  
    // Start timestamp in the samplerate clock  
    unsigned long startSampleIndex;  
    unsigned int sampleCount;  
    unsigned int channelCount;  
    CodecType codecType;  
    ByteArray encodedData;  
}
```

VideoEncoder

```
interface VideoEncoder {
    // Can be called any time to change parameters
    void start(MediaStreamTrack rawVideo, VideoEncodeParameters encodeParameters);
    void stop();
    attribute EventHandler onencodedvideo; // of EncodedVideoFrame
    // Alternative, WHATWG-stream-style:
    readonly attribute ReadableStream encodedVideo; // of EncodedVideoFrame
}
```

```
dictionary VideoEncodeParameters {
    unsigned long bitrate;
    boolean generateKeyFrame;
    // TODO: SVS/simulcast, resolutionScale, framerateScale, ...
    // ...
}
```

EncodedVideoFrame

```
dictionary EncodedVideoFrame {  
    unsigned short width;  
    unsigned short height;  
    unsigned short rotationDegrees;  
    unsigned long timestampMs;  
    CodecType codecType;  
    ByteArray encodedData;  
}
```

AudioDecoder

```
interface AudioDecoder {  
    void decodeAudio(EncodedAudioFrame frame);  
    readonly attribute MediaStreamTrack decodedAudio;  
  
    // Alternative, WHATWG-stream-style:  
    readonly attribute WritableStream encodedAudio; // of EncodedAudioFrame  
}
```

VideoDecoder

```
interface VideoDecoder {  
    void decodeAudio(EncodedVideoFrame frame);  
    readonly attribute MediaStreamTrack decodedVideo;  
  
    // Alternative, WHATWG-stream-style:  
    readonly attribute WritableStream encodedVideo; // of EncodedVideoFrame  
}
```


Questions for the WG

- Do we want to split out the encoders and decoders?
- Do we want to split out the RtpTransport?
- Do we want the app to be able to opt in to being in the media path?
- Do we want this general approach to encoders/decoders?
- Do we want this general approach to RtpTransport?
- How do we want to model simulcast/SVC?

Where is the jitter buffer?

Well... which kind of buffer do you mean? There are 3. You probably mean #2.

The **packet buffer** buffers packets before assembling into an encoded frame. This is outside of the decoder, such as in the RtpReceiver, which interacts with the encoder at the encoded frame level.

The **encoded frame buffer** buffers encoded frames before decode. For simplicity, we put this in the "decoder". Thus a "decoder" is also an encoded frame buffer and can accept frames out of order. This could theoretically be split up, but the benefit is unclear.

A **smoothing buffer** buffers decoded audio and smooths it before playout. For simplicity, this is also in the "decoder". This could also be split up, but the benefit is unclear.

Raw Data Access

WebRTC WG interim, June 19-20

How we can access raw media today - issues

- This is already possible
 - MediaStream -> Canvas + capture canvas data for incoming video
 - Canvas to MediaStream for outgoing video
 - WebAudio for audio
- Interfaces are neither particularly elegant, convenient or fast
- The Javascript processing model is not very suitable for real-time work
 - (that's putting it mildly)

Javascript model issues

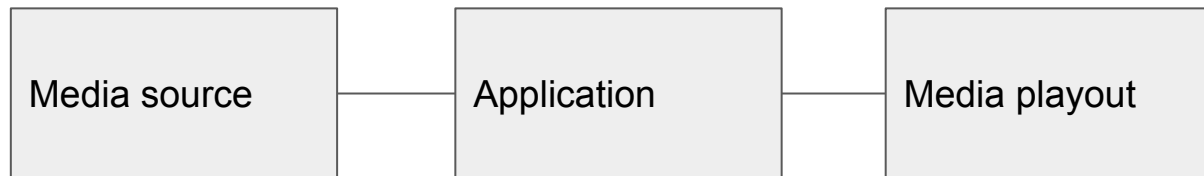
- Single-process model
 - Random stuff happens on the main thread
 - Anything that's in a hurry has to happen in a worker
- Garbage collection
 - Implicit in create/release model
 - WebAssembly doesn't by itself cause create/release (memory static within object)
 - Interacting with Web APIs does cause create/release (for the time being)
- Raw video is very big - ms per frame copy - but can tolerate some hiccups
- Raw audio is much smaller - but the ear is sensitive to glitches

In-pipeline and from-pipeline use cases

Some apps require in-pipeline work, and thus real-time operation

- Funny hats (the canonical use case)
- Voice distortion (male -> female pitch, voice clarification)
- Augmented reality applications
- Echo cancellers

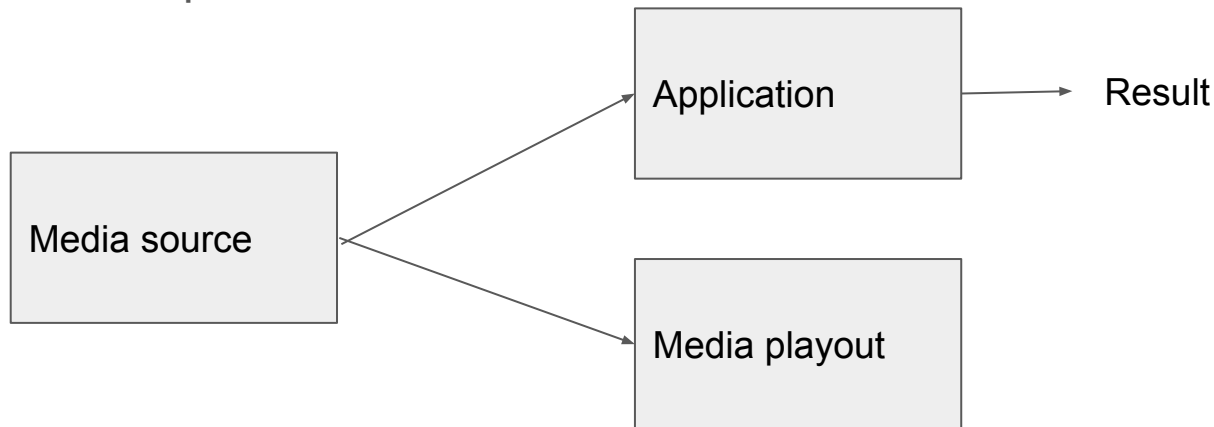
Milliseconds count. Consistency counts even more.



In-pipeline and from-pipeline use cases (2)

Some aren't in that much of a hurry

- Volume meters
- Typing detectors
- “What bird” - object recognition
- Automatic transcription



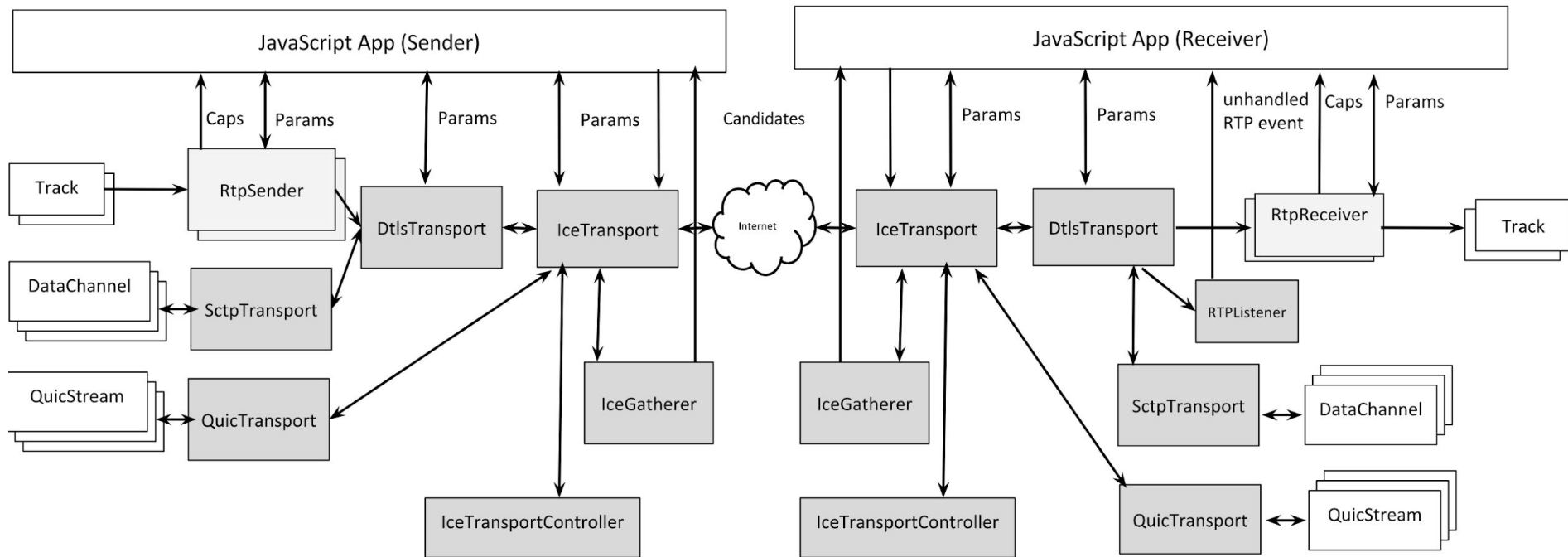
Requirements for real time raw data APIs

- Accessible from workers
- Must not create “too many” required copies of media data
- For “in-pipeline” applications:
 - Must have control over synchronization
 - Must have predictable scheduling
 - Buffering isn’t an option.
- For “from-pipeline” applications
 - Must have timing information
 - Must be able to guarantee that they will “keep up” with data flow
 - Buffering helps. GC is the enemy.

ICE

6/2018 f2f

ORTC Object Model



Source: <http://draft.ortc.org/>

ORTC IceGatherer

- Features:
 - Forking (IceGatherer can map to multiple IceTransports)
 - ICE candidate gathering control (extended in ORTC lib)
 - Support for RTP/RTCP mux or non-mux
 - Support for renomination (extended in ORTC lib to support candidate removal)
- Existing implementations: Edge (no forking), Ortc lib (forking, candidate filtering, candidate removal/renomination).
- Usage
 - ORTC Lib now very popular in games utilizing gaming consoles (XBOX) and mobile platforms (iOS, Android)

ORTC IceGatherer

WebIDL



```
[Constructor(RTCIceGatherOptions options),  
 Exposed=Window]  
interface RTCIceGatherer : RTCStatsProvider {  
    readonly attribute RTCIceComponent component;  
    readonly attribute RTCIceGathererState state;  
    static sequence<RTCIceServer> getDefaultIceServers();  
    void close();  
    void gather(optional RTCIceGatherOptions options);  
    RTCIceParameters getLocalParameters();  
    sequence<RTCIceCandidate> getLocalCandidates();  
    RTCIceGatherer createAssociatedGatherer();  
    attribute EventHandler onstatechange;  
    attribute EventHandler onerror;  
    attribute EventHandler onlocalcandidate;  
};
```

ORTC IceTransport

WebIDL



ReSpec

```
[Constructor(optional RTCIceGatherer gatherer),  
Exposed=Window]  
interface RTCIceTransport : RTCStatsProvider {  
    readonly attribute RTCIceGatherer? iceGatherer;  
    readonly attribute RTCIceRole role;  
    readonly attribute RTCIceComponent component;  
    readonly attribute RTCIceTransportState state;  
    sequence<RTCIceCandidate> getRemoteCandidates();  
    RTCIceCandidatePair? getSelectedCandidatePair();  
    void start(RTCIceGatherer gatherer,  
              RTCIceParameters remoteParameters,  
              optional RTCIceRole role = "controlled");  
  
    void stop();  
    RTCIceParameters? getRemoteParameters();  
    RTCIceTransport createAssociatedTransport();  
    void addRemoteCandidate(RTCIceGatherCandidate  
remoteCandidate);  
    void setRemoteCandidates(sequence<RTCIceCandidate>  
remoteCandidates);  
    attribute EventHandler onstatechange;  
    attribute EventHandler oncandidatepairchange;  
};
```

WebRTC-ICE

- A directly constructable/controllable version of WebRTC 1.0's IceTransport
It doesn't require SDP and it works with DtlsTransport and QuicTransport .
- A simplified version of the ORTC IceTransport + IceGatherer.
It doesn't support forking, but it's simpler to use, and ***is compatible with the WebRTC 1.0 IceTransport.***

This is much better.

For some new use cases, **this is not enough.**

Why?

- New requirements
 - Candidate local gathering control (wifi/cell)
 - Control over local candidate lifetime (retaining it, closing it)
 - Control over checking frequency (or pausing it)
 - reselection/renomination
 - forking
- And other improvements
 - More debugging info

FlexICE

Take the IceTransport from WebRTC-ICE and incrementally add controls:

- **IceTransport.gather()** and **removeLocalCandidate()** to add/remove local candidates
- **IceTransport.retainLocalCandidate()** to not free local candidates
- **IceTransport.removeCandidatePair()**
- **IceTransport.retainCandidatePair()**
- **IceTransport.fork()**
- **RTCIceGatherOptions.networkInterfaceIds** to control network usage
- **IceLocalCandidate.networkInterfaceType** + **networkInterfaceId** to know networks
- **IceCandidatePair.setMinCheckInterval()**, **setFrozen()** to control check activity/frequency
- **IceCandidatePair.setCheckPriority()** to control check order
- **IceCandidatePair.select()** and **nominate()** to select and nominate a candidate pair
- **IceCandidatePair.onchecksnt**, **.oncheckresponsereceived**, **.setReceiveTimeout()**, **.onreceivetimeout** to know when things stop working and control what "disconnected" means.

Non-standard behavior

Some of what this would allow the app to do is not compliant with RFC 5245(bis).

In particular:

- Renomination (there is an expired draft)
- Non-standard freezing (probably doesn't matter; usually not implemented anyway)

SLICE

The "brains" of the ICE agent isn't implemented by browser, but by the app, using low-level pieces needed by an ICE agent:

1. Enumerate networks using NetworkManager
2. Gather candidates using UdpPort, ClientTcpPort, StunClient, TurnClient
3. Pair ports with remote candidates to create UdpTransport, TcpTransport, TurnTransport.
4. Send checks and receive check responses using IceNetworkRoute.
5. Select/nominate a candidate pair

Which one?

If you're interested in really tight control over how ICE works, you want SLICE.

If you just want to tweak the existing behavior a little bit, you want FlexICE.

If you just want a normal ICE agent, WebRTC-ICE is good enough

FlexICE is probably the sweet spot, especially since it can start simple and be extended over time.

Questions for the WG

Which path for ICE do we want to pursue?

Identity - what next?

WebRTC WG, June 19-20 2018

Status of this preso

This preso is based on the message that the chairs sent to the mailing list on May 23, 2018. No further discussion on the mailing list has occurred.

The message contained a proposal for further action.

No decision has been made.

Identity Implementation Status

- Firefox: Implemented (but per march 2018 not up to date)
- Chrome: No plans announced
- Edge: No plans announced
- Safari: No plans announced
- Web developers: No interest expressed
 - Exception: Cisco
- Identity providers: No interest expressed

Identity - Process Relationships - IETF

IETF:

- draft-ietf-rtcweb-security-arch 5.6: Identity is required
 - In status “Waiting for WG Chair go-ahead” since April 18
- RFC 7478 (requirements): “cryptographically binding media ... to the user identity”
- These are normative references from rtcweb-overview

Conclusion: An identity protocol is a normative requirement.

Identity - Specification status

- No production service
 - We don't know if it's deployable, or if it can be attacked - nobody's tried
 - Discussion after hackathon led to suggestions for protocol and API changes
- 23 “identity related”-tagged open bugs in the issue tracker
 - Oldest from Dec 2016, newest from Oct 2017
 - No closed bugs with this tag
 - No submitted PRs addressing these bugs
 - No current editors with Identity expertise
 - WPT tests “deep red” on all browsers
 - 2 messages posted on these bugs since WebRTC 1.0 first went to CR in November 2017 (by Harald and Varun, related to WebRTC-stats)

Identity - W3C formalisms

- Charter (unchanged since our first charter) says:
 - “To advance to Proposed Recommendation, each specification is expected to have two independent implementations of each feature defined in the specification.
 - To advance to Proposed Recommendation, interoperability between the independent implementations (that is, bidirectional audio and video communication as well as data transfer between the implementations) should be demonstrated.”
- W3C gives “stable references” for any document in WD, CR or PR state
- W3C routinely publishes PRs with references to non-PR documents
 - the rules require justification for the downref, just like the IETF
- The WG has been encouraged to strive for PR status
 - whether that’s a valuable thing is also debated. Let’s not have that debate here.

Short summary of status

- Identity needs work.
 - Specification cleanup
 - Implementations
 - Experience
 - Possibly redesigns based on experience
- A webrtc-pc specification with identity in it can't advance to PR
- A webrtc-pc specification with a normative reference to identity can advance

Proposal

Move Identity to its own document. Let those who most believe that identity is important contribute resources to editing that document - and ONLY that document.

Discuss.

Questions for the group

- Split section 9 out into a separate document? Yes or no?
- Who volunteers to be its editors? (goes for both section and document)

Webdriver extensions for webrtc testing (proposal)

Dr Alex G,
With inputs from Youenn F, Brian B

Browser configs and delayed gratification



Some [...] have been considering extensions [...] via WebDriver commands [...].
This will delay each API's availability on the web and presence in WPT.

Specific WebRTC Testing Issues

1. Permission prompt

- Those prompts are not part of the DOM (UA), cannot be access by JS in purpose.
- Those prompts are not modal in nature, no existing webdriver API to manipulate them
- All browsers have a by-pass mechanism (except edge), which all differ
 - Registry entry (egde)
 - Persistent choice (edge, manual once)
 - Profile (mz)
 - Command line argument (cr)
 - Dev Menu / command line (safari)

2. Media

Existing solutions

1. Permission prompt: proposal

- Permission spec, automation chapter (11).
- Additional implementation in safaridriver:
<https://gist.github.com/burg/17a6f362fedfe6ee91354d197b415736>
- ***Small Details***
 - Good for ON/OFF choices,
 - Might require more work for enumerating devices. Could be left out at this stage, in the absence of strong testing use case.

Specific WebRTC Testing Issues

2. Media creation: How to

- To test specific video/audio capture HW on specific devices [\[do not forget!\]](#)
- To test peer connection or other apis with programmatically generated media
 - Front end / back end, audio/video sync, degradation, resolutions,
- To test peer connection or other apis on VMs or devices without capture HW
 - CL arg (cr)
 - Mock capturer automatically made default capturer under automation (safari)
 - Read from file (cr)
 - Other JS API to generate media (webAudio, from Canvas, ...)
 - Virtual device (registering as OS device driver) ([bocoup proposal](#))

Specific WebRTC Testing Issues

2. Media creation: Proposal

- Virtual devices are not possible on all OS we want to support.
- Use JS APIs to generate media, not there yet, even though it's doable for many wpt tests ([bocoup's PR](#)).
- Most of the browser have a fake media / mock capturer implemented
 - **Why not reusing it, and just make a simple webdriver API to switch it On/Off that would be the same across browser and OSes.** implementations remains the same, "trigger" is unified.
- **Small details:**
 - Should not be the unique and default choice when on (safari)
 - Better that what we have now, but having more control on the media would be nice down the line.

WPT Progress

Soares C

2018 GitHub Stats (Jan - June)

- 53 Merged PRs
 - 44 Chromium Exports
 - 3 Gecko Exports
 - 6 Others
- 14 Open PRs
- 8 Open Issues
- File Changes
 - 13 Files Added
 - 109 Files Modified
 - 8 Files Deleted

Merged PRs (Chromium export)

- [#8814](#) - RTCRtpSender.replaceTrack added behind flag
- [#8968](#) - Add WPT test coverage for getStats() returning stats for tracks/streams
- [#8994](#) - getStats() WPT tests for inbound and outbound RTP stream stats
- [#8999](#) - WPT tests added for getStats() in combination with replaceTrack()
- [#9134](#) - Fix issues in RTCStats-helper.js for WPT
- [#9388](#) - Support MediaStreamTrack.getCapabilities() for echoCancellation and deviceId
- [#9479](#) - Improve RTCRtpSender.replaceTrack tests' Firefox compatibility
- [#9516](#) - Make WPT webrtc/simplecall.html pass
- [#9570](#) - Add support for video properties in MediaStreamTrack.getCapabilities()
- [#9583](#) - Add the "dtmf" attribute on RTCRTPSender
- [#9688](#) - getCapabilities() should not have range properties without valid values
- [#9733](#) - Introduce InputDeviceInfo interface
- [#9927](#) - Implement InputDeviceInfo.getCapabilities() for audio devices
- [#9941](#) - Improve Firefox compability of RTCDTMFHelper
- [#9971](#) - Implement InputDeviceInfo.getCapabilities() for video devices
- [#10075](#) - Add memory of last SDP offer/answer created
- [#10109](#) - Test that DTMFSender rejects properly after close
- [#10142](#) - RTCRtpSender.getStats() in blink added behind flag
- [#10154](#) - Add support for autoGainControl and noiseSuppression to getCapabilities()
- [#10219](#) - RTCRtpReceiver.getStats() in blink added behind flag

- [#10247](#) - RTCPeerConnection.getStats(MediaStreamTrack? selector = null) added
- [#10278](#) - Don't enforce name rule for RTCDTMFToneChangeEvent
- [#10342](#) - Fix RTCPeerConnection-track-stats.https.html flake.
- [#10458](#) - Fix race in track-stats.https.html test
- [#10545](#) - Support the groupId property in MediaStreamTrack.getSettings()
- [#10553](#) - Support groupId in MediaStreamTrack.getCapabilities() for video tracks
- [#10554](#) - Support groupId constrainable properties in MediaDevices.getUserMedia()
- [#10884](#) - Pass test in promise_test and async_test
- [#10885](#) - Adds a test for basic WebRTC video codec conformance
- [#10893](#) - Pass test function in more tests
- [#10997](#) - Add test to verify a particular PeerConnection setup does not deadlock
- [#11138](#) - Support groupId in MediaStreamTrack.getCapabilities() for audio tracks
- [#11177](#) - Remove test_state_change_event
- [#11208](#) - Do not allow the empty string as a facingMode constraint value for MediaStreams
- [#11209](#) - More video protocol tests
- [#11245](#) - Support groupId in MediaDevices.getUserMedia() for audio tracks
- [#11267](#) - check signalingState before addIceCandidate
- [#11301](#) - Improve RTCPeerConnection-setRemoteDescription-tracks.https.html tests
- [#11455](#) - Add cleanup to close peerconnections
- [#11459](#) - Fix errors external/wpt/RTCPeerConnection-removeTrack.https.html
- [#11460](#) - Use assert_array_equals, not assert_equals
- [#11471](#) - Fix missing argument in RTCPeerConnection-getStats
- [#11472](#) - Fix errors in external/wpt/RTCDTMFSender-insertDTMF.https.html
- [#11501](#) - Fix/improve external/wpt/webrtc/RTCPeerConnection-addTrack.https.html

Merged PRs (Gecko export)

- [#10999](#) - Fix typo in RTCPeerConnection-setDescription-transceiver.html
- [#11031](#) - Update RTCPeerConnection-setRemoteDescription.html to spec, and avoid hang-prone test_state_change_event()
- [#11054](#) - Add t.step_func() to callback and remove closed state test

Merged PRs (Others)

- [#8996](#) - Fix RTCScpTransport.maxMessageSize default value
- [#9181](#) - Add missing untested IDL for MediaStream tests
- [#9342](#) - Add a negative test for the DataChannel interface (Gecko only)
- [#9853](#) - Update the webrtc-pc IDL file
- [#10515](#) - URL.createObjectURL should not work with MediaStream
- [#11211](#) - Simply test and use standard API

Open PRs

- [#11528](#) - Rename external/wpt/webrtc/RTCRtpParameters* tests to *.https.html
- [#11526](#) - Update external/wpt/webrtc/RTCRtpParameters* tests
- [#11524](#) - Add tooling to allow using jscodeshift codemods
- [#11506](#) - Support all constrainable properties for audio tracks in MediaStreamTrack.getSettings()
- [#10746](#) - Use procedurally-generated media streams
- [#10610](#) - [webrtc-stats] add META.yml
- [#10577](#) - Seed directory for WebRTC Statistic spec
- [#10566](#) - addTrack: split up tests and reduce dependencies
- [#10468](#) - More Tests for WebRTC Data Channels
- [#10282](#) - adds tests for the missing members of RTCRtpHeaderExtensionParameters
- [#10271](#) - Bring RTCCertificate interface up to date with Candidate Recommendation
- [#9708](#) - getCapabilities() should not have range properties without valid values
- [#9434](#) - Rename RTCIceCandidate ufrag field to usernameFragment
- [#9424](#) - Rewrite RTC ICE and DTLS transport tests with alternative dependencies

Open Issues

- [#10981](#) - Firefox test doesn't load H.264 codec
- [#10253](#) - RTCRtpHeaderExtensionParameters missing members are not tested
- [#10140](#) - Identity provider mock should succeed on Firefox
- [#10126](#) - RTCDataChannel-bufferedAmount.html are flawed
- [#9213](#) - Parts of WebRTC require generating RTP to test
- [#9108](#) - RTCRtpContributingSource's audioLevel member should be double (asserted to be byte)
- [#8301](#) - WebRTC tests should close RTCPeerConnection's at end of test
- [#6533](#) - Add test for event instanceof RTCTrackEvent in ontrack handler

WebRTC tests are leaking heavy resources (Chromium [836871](#))

- Too many unclosed `RTCPeerConnections` cause browsers crashing when running WPT
- Solution: add `t.add_cleanup(() => pc.close())` to every test. (Merged in [#11455](#))
- Toolings added to automatically add missing cleanups. ([#11524](#))

Use procedurally-generated media streams ([#10746](#))

- `getUserMedia()` requires fake media devices for testing
- `addTransceiver().receiver.track` not implemented in all browsers
- Solution:
 - Video - Use ``canvas.getContext('2d').captureStream()`` from [mediacapture-fromelement](#)
 - Audio - Use ``new AudioContext().createMediaStreamDestination()`` from WebAudio
- Test helper
 - Use `getNoiseStream()` instead of `getUserMedia()`
- Cons: `captureStream()` not implemented in all browsers
 - Workaround: Use audio tracks whenever possible

Adds a test for basic WebRTC video codec conformance ([#10885](#))

- [RFC 7742](#) - WebRTC Video Processing and Codec Requirements
- Test that H.264 and VP8 should be supported in initial offer
- TODO: more tests on H.264 and VP8 requirements

wpt/webrtc-stats Directory (#10577)

- Move stats tests to new top-level webrtc-stats directory
- Overlapping issues with stats APIs in webrtc? ([#10610](#))

New WPT Contributors

- @lgrahl - More Tests for WebRTC Data Channels ([#10468](#))
- @kritisingh1 - Adds tests for the missing members of RTCRtpHeaderExtensionParameters ([#10282](#))
- @snuggs - various comments and reviews
- PRs not reviewed since April. (Sorry)
 - Better ways to build up volunteer contributors in WPT?

KITE for webrtc testing 06-2018 Update

W3C WebRTC interim f2f meeting

KITE Goals

- **Goal:** testing p2p communication between two webrtc-capable [*browsers*]
- **W3C Goal:** test webrtc APIs to proceed to standard (compliance)
- **Browser vendors:** fasten implementation, reference test (compliance)
Regression testing, early warning (production)
- **All:** interoperability between any webrtc capable [*clients*], and any [*back-ends*], using any signaling protocol.

Mostly Open-Source code with currently three W3C Members contributing with codes or concepts: [Google](#), [CallStats.io](#) and [CoSMo](#). Others members have publicly expressed interest: [Cisco](#), [MS](#), [Apple](#), ..

KITE Interop SE Grid - Browser configs

(without saucelab, without Mobile, Without Electron [comm])



21 MARCH
2018



23 MARCH
2018



21 MARCH
2018



21 MARCH
2018



23 MARCH
2018



23 MARCH
2018



23 MARCH
2018



23 MARCH
2018



21 MARCH
2018



21 MARCH
2018



21 MARCH
2018



21 MARCH
2018



21 MARCH
2018



23 MARCH
2018



23 MARCH
2018



23 MARCH
2018

KITE Modular Design



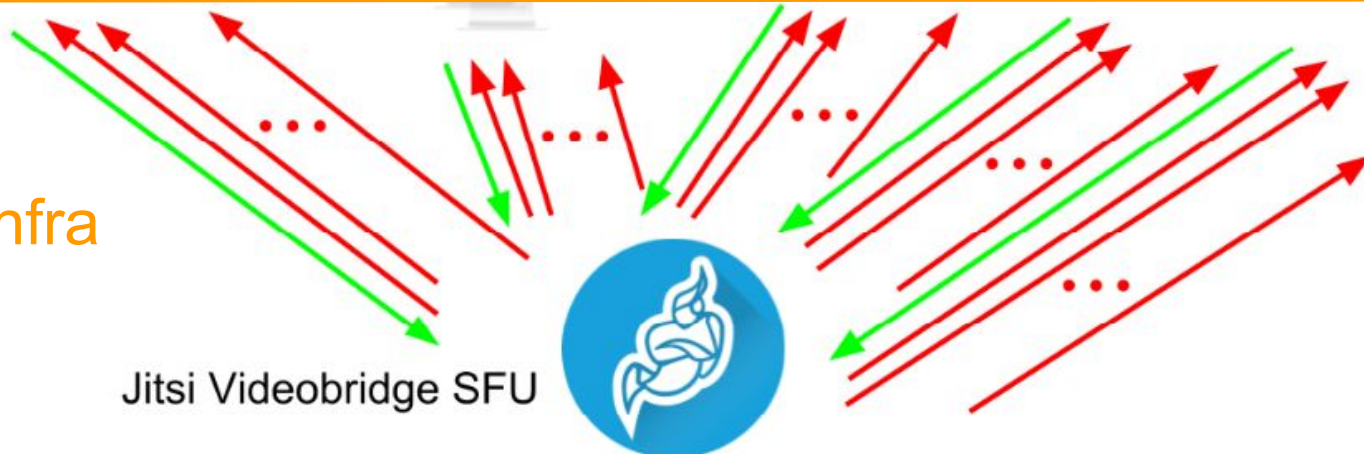
SE Grid

Running test



Test own Infra

Jitsi Videobridge SFU



KITE 1-browser : auto WPT runs

(2d of work for all available configs)

WPT / WEBRTC	23 MARCH 2018	23 MARCH 2018	27 MARCH 2018	28 MARCH 2018	27 MARCH 2018	27 MARCH 2018	28 MARCH 2018	28 MARCH 2018	27 MARCH 2018	27 MARCH 2018	27 MARCH 2018	27 MARCH 2018	27 MARCH 2018	27 MARCH 2018	27 MARCH 2018	27 MARCH 2018
iceConnectorState	0/2	0/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2
RTCPeerConnection-iceGatheringState	0/3	0/3	2/3	2/3	2/3	2/3	2/3	2/3	2/3	2/3	2/3	2/3	2/3	2/3	1/3	1/3
RTCPeerConnection-onDataChannel	0/3	0/3	2/3	2/3	2/3	2/3	2/3	2/3	1/3	1/3	1/3	2/3	2/3	2/3	1/3	1/3
RTCPeerConnection-onnegotiationneeded	0/7	0/7	3/7	3/7	3/7	3/7	3/7	3/7	7/7	7/7	7/7	7/7	7/7	7/7	5/6	5/6
RTCPeerConnection-ontrack.https	0/5	0/5	1/5	1/5	1/5	1/5	1/5	1/5	5/5	5/5	5/5	4/5	5/5	5/5	2/5	2/5
RTCPeerConnection-peerIdentity	0/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6
RTCPeerConnection-removeTrack.https	0/12	0/12	0/12	0/12	0/12	0/12	0/12	0/12	9/12	9/12	9/12	4/6	9/12	9/12	2/12	2/12
RTCPeerConnection-setDescription-transceiver	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	4/5	4/5	4/5	4/5	4/5	4/5	0/5	0/5
RTCPeerConnection-setLocalDescription-answer	0/5	0/5	1/6	1/6	1/6	1/6	1/6	1/6	2/6	2/6	2/6	2/6	2/6	2/6	4/6	4/6
RTCPeerConnection-setLocalDescription-offer	0/5	0/5	2/7	2/7	2/7	2/7	2/7	2/7	3/7	3/7	3/7	3/7	3/7	3/7	3/7	3/7
RTCPeerConnection-setLocalDescription-pranswer	0/4	0/4	4/7	4/7	4/7	4/7	4/7	4/7	0/7	0/7	0/7	0/7	0/7	0/7	6/7	6/7

web-platform-tests dashboard

Latest Run Recent Runs About GitHub Source





Search test files, like cors/allow-headers.htm

WPT /

Showing results for run 4a0df34066.

[View latest run](#)

Data below are intended for web platform implementers and do not contain useful metrics for evaluation or comparison of web platform features. Also note that tested Edge and Safari are not pre-release versions (#109, #110).

Spec	 chrome 66.0.3359.117 linux 4.4 @4a0df34066 Apr 18 2018	 edge 15 windows 10 @4a0df34066 Apr 20 2018	 firefox 59.0.2 linux 4.4 @4a0df34066 Apr 18 2018	 safari 11.0 macos 10.12 @4a0df34066 Apr 18 2018
/2dcontext	1924 / 1999	1232 / 1503	1861 / 1999	1654 / 1880
/BackgroundSync	35 / 35	8 / 18	17 / 35	17 / 35
/FileAPI	827 / 842	504 / 708	752 / 827	650 / 707
/IndexedDB	1828 / 1833	464 / 881	1809 / 1833	1531 / 1613
/WebCryptoAPI	40606 / 43207	791 / 17251	14541 / 27366	36328 / 41493
/WebIDL	455 / 470	165 / 254	461 / 470	422 / 470
/accelerometer	23 / 103	21 / 101	23 / 103	23 / 103
/acid	101 / 103	0 / 3	101 / 103	101 / 103
/ambient-light	19 / 57	17 / 55	19 / 57	19 / 57
/apng	2 / 3	1 / 3	3 / 3	2 / 3
/audio-output	18 / 20	3 / 12	4 / 15	4 / 15
/background-fetch	59 / 181	59 / 180	59 / 181	60 / 181
/battery-status	44 / 46	3 / 9	4 / 10	4 / 10
/beacon	51 / 117	71 / 85	100 / 107	22 / 107

KITE 2-browsers: appRTC runs and Stats integration

Daily runs on webrtc.org !

Analysis and bug reports against Browsers, webdrivers impl, ...

ICE stats:

[chrome 65.0 ANDROID] [MicrosoftEdge 16.16299 WINDOWS 10]

LOCAL CANDIDATE	REMOTE CANDIDATE	ICE STATE	NOMINATED	PRIORITY	BYTES SENT	BYTES RECEIVED
RTCIceCandidate_YMvfhdZ	RTCIceCandidate_gWp4gO4C	succeeded	true	9079290933605827000	831542	123116

SDP

Offer Answer

Offer

```
v=0
o=- 48895174166018984 0 IN IP4 127.0.0.1
s=-
t=0 0
a=group:BUNDLE audio video
a=msid-semantic: WMS 5882B0EC-B490-44E4-A310-9244428F4357
m=audio 50434 UDP/TLS/RTP/SAVPF 111 0 8 126
c=IN IP4 192.168.1.201
a=rtcp:9 IN IP4 0.0.0.0
a=candidate:1 1 udp 2130706431 192.168.1.201 50434 typ host generation 0
a=candidate:2 1 udp 33553919 2001:0:9d38:6abd:2098:14ae:bd9f:336c 61862 typ host generation 0
a=ice-ufrag:UWNR
a=fingerprint:sha-256 66:28:E0:08:AC:93:8E:91:3C:A2:40:CD:04:28:EC:5D:07:EB:B3:97:88:93:A0:D8:EB:7D:D0:08:B1:DC:5D:62
a=setup:active
a=mid:audio
```

Audio (Packets lost: 4 | Avg jitter: 0.00522)

Bytes (avg S/R: 3/5 kB/s) Packets (avg S/R: 61/61 /s)

Video (Packets lost: 0 | Avg jitter: 0.00000)

Bytes (avg S/R: 40/0 kB/s) Packets (avg S/R: 47/1 /s)

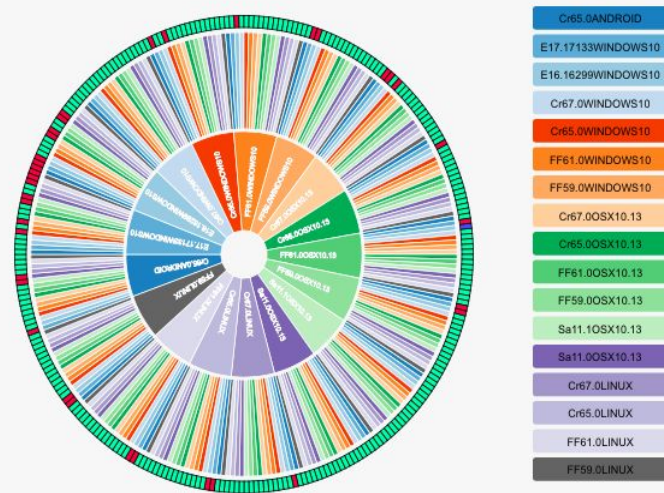
IceConnectionTest (total: 288)

Started at: 11 Apr, 2018 10:23:43 AM

Filter: [Success] [Failed] [Error] [Pending] All browsers

STATS	RESULT	DURATION	BROWSER(S)
0	Failed	30s	65.0ANDROID 17.17133WINDOWS 10
1	Success	39s	65.0ANDROID 16.16299WINDOWS 10
2	Success	38s	65.0ANDROID 67.0WINDOWS 10
3	Success	35s	65.0ANDROID 65.0WINDOWS 10
4	Success	34s	65.0ANDROID 61.0WINDOWS 10

SUCCESSFUL



KITE 3-browsers: Jitsi runs

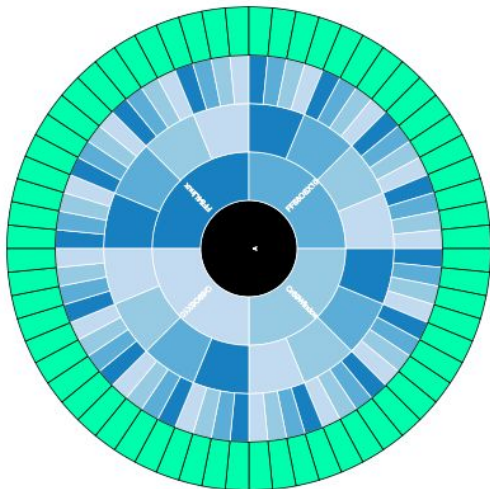
Result(s) for: Jitsi ICEConnection Test

Started at: 20 Jun, 2017 9:56:07 AM

including 64 test case(s)



- FF54Linux
- FF53OSX10.12
- Cr59Windows8.1
- Cr59OSX10.11



STATUS	DURATION	BROWSER(S)
✓	15s	54Linux 54Linux 54Linux
✓	25s	54Linux 54Linux 53OS X 10.12
✓	12s	54Linux 54Linux 59Windows 8.1
✓	19s	54Linux 54Linux 59OS X 10.11
✓	13s	54Linux 53OS X 10.12 54Linux
✓	27s	54Linux 53OS X 10.12 53OS X 10.12
✓	31s	54Linux 53OS X 10.12 59Windows 8.1
✓	26s	54Linux 53OS X 10.12 59OS X 10.11
✓	15s	54Linux 59Windows 8.1 54Linux
✓	28s	54Linux 59Windows 8.1 53OS X 10.12
✓	32s	54Linux 59Windows 8.1 59Windows 8.1
✓	22s	54Linux 59Windows 8.1 59OS X 10.11
✓	26s	54Linux 59OS X 10.11 54Linux

KITE Update 04/2018: new tests for [multiparty](#), [multistream](#), [simulcast](#) ...

Multiparty -> jitsi test. No limit in the number of parties ... in KITE :-)

Multistream -> testing against Unified Plan, helping chrome delivering faster
(see next slide)

Simulcast -> Requires SFU. Only sender->SFU defined in webrtc 1.0, logic for layer switching not defined. Lots of arbitrary decisions for now. See next slides.

KITE: multistream and Unified Plan

Multi-stream (Unified plan):

- It runs a local signaling server (node), currently with and without adapter.js
- Using streams from html video elements for peer connection.
- The test verifies :
 - that the generated SDPs are compliant with the unified plan format (sender).
 - the received stream ids against ids announced in the SDP offer (receiver).
 - that media is flowing (with canvas sum as usual).

Results (4/12):

- Firefox: it works as expected, even without adapter.js
- Chrome: there are some errors, and the flags provided do not seem to work yet. In Progress.

KITE: Simulcast

Simulcast

- the dedicated app runs over https and is available at <https://simulcast-test.dev.cosmosoftware.io/>
- The test verifies the following (not in order):
 - echoed stream is displayed from loopback peerconnection.
 - stream is sent back from SFU (validates it received it, format was correct, and it could extract the right layer).
 - access to SDP offer/answer from peerconnection object.
 - SDP offer/answer format.

Results (4/12):

- it works as expected on Firefox only, but using the SDP format from the older draft. However, that should be considered a failure for strict compliance testing, since the SDP is not using the latest specs format.

KITE Update 06/2018: Some recent results

- The majority of the failed cases are Edge's. There are 2 reasons for this:
 - webdriver mismatch for Edge insider:
 - Edge's webRTC implementation is slightly different from the others browsers. ([here](#))
- Firefox crashed when tested against safari. ([here](#))
- Chrome bug related to multi-stream and unified plan ([here](#))
- Electron webdriver hanging, fixed.
- Edge does not enumerate virtual capture devices (manycams, [vlc2vcam](#) with [Magic Camera](#),)

Alas, no regression found (regressions did happen. No existing tool caught them).

KITE Update 06/2018: Network Instrumentation

Goal: Evaluate the behavior of the following algorithm types:

- Bitrate adaptation and degradation preferences (Q, spatial, temporal)
- Simulcast / SVC layer control
- Bandwidth estimation
- Congestion control
- Error correction
- Jitter correction

Mean: Instrumentation and programmatic control the following

(1) independently for each client (2) or server (3) on each OS a browser is available:

- Network Bandwidth and corresponding variations across time
- Network Quality and corresponding variations across time
 - Jitter
 - Packet loss
- NAT settings,
- Firewall settings,

Original collaboration proposal by CallStats.io, which had a solution without (3).

KITE Update 06/2018: Network Instrumentation

NW Instrumentation Test Bed - Two NATs

Emmanuel André | Cosmo | June 1, 2018

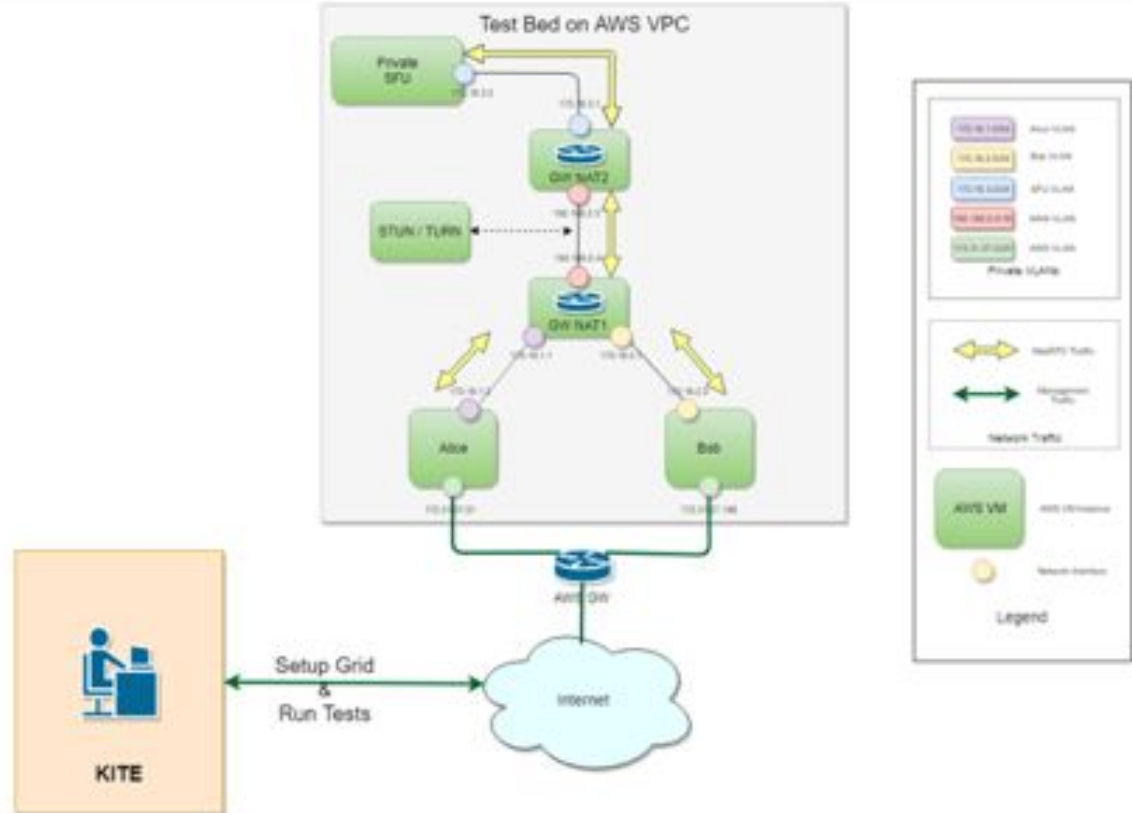
NTT => VPC

Elastest => Docker

Problem of scope

Problem of OS

Problem of links



KITE Update 06/2018: New Use Cases, New Tool

- Load testing: a missing tool for Video Conference use case:
 - Jitsi hammer, Jattack, multiplier, “bees with machine guns”, ...
- VC is a relatively simple use case, 30 to a few 100s of browsers is enough.
- Broadcast/streaming is a more demanding use case.
 - Publication on the subject (“*WebRTC Testing: Challenges and Practical Solutions*” in *IEEE Communications Standard Magazine*, june 2017) by the Kurento / ElasTest team
 - “One of the most demanded service in the WebRTC arena: Broadcasting”
 - Could only test up to 200 viewers, and by using native clients in the mix, introducing a bias.

We needed a new tool:

KITE LOAD TESTING

KITE Update 06/2018: New Use Cases, New Tool

- Publication on the subject
“WebRTC Testing: Challenges and Practical Solutions” in IEEE Communications Standard Magazine, june 2017
by the Kurento / ElasTest team
- “One of the most demanded service in the WebRTC arena: Broadcasting”
- Could only test up to 200 viewers, and by using native clients in the mix, introducing a bias.

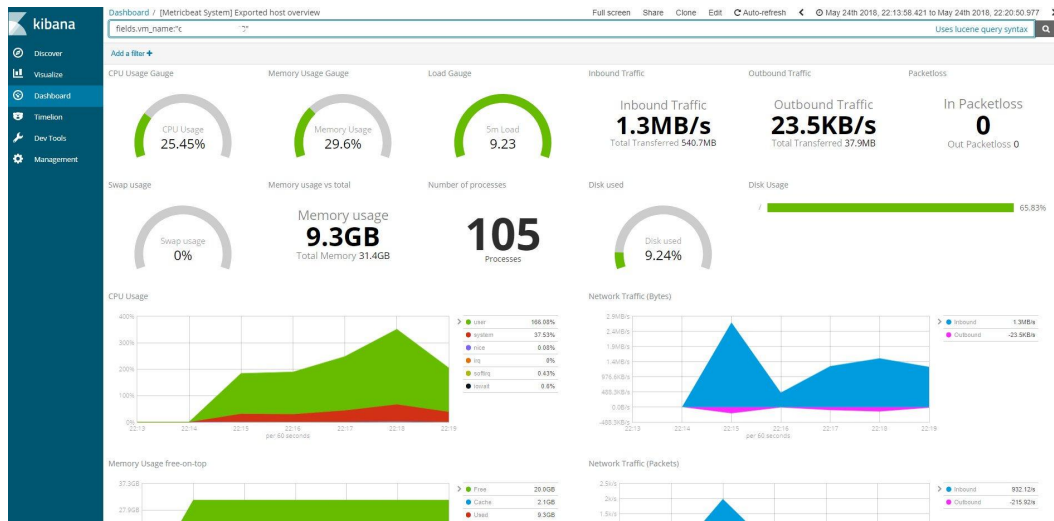
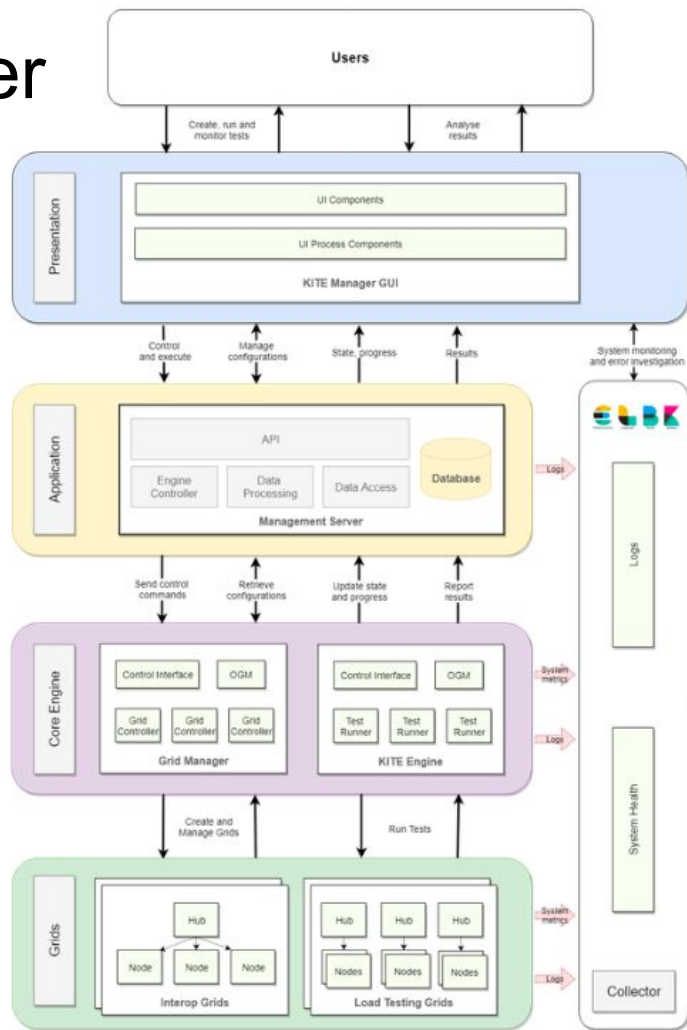


Dr. Alex. Gouaillard
@agouaillard

Test of [#webrtc](#) Video Confs? fun, but not hard. at most 1k users. Test of WebRTC Streaming? Superbowl viewers = 100M, Game of Throne = 10M, Youtube = [youtube.googleblog.com/2012/10/missio....](https://youtube.googleblog.com/2012/10/missio...) KITE load test mode? close to 1M. We had to develop a Grid Mngr and Op. Dashboards to handle that scale!

KITE: Load Testing - Grid Manager

- Never seen before Selenium Grid Scalability
 - Presentation proposed at Selenium World - pending
- Tested against most open-source webtrc SFU,
 - results being proposed for publication at IPTComm
- Tested in production with SpankChain and millicast.com (a Xirsys collaboration)
 - up to 50,000 concurrent viewers per source



For extra credit



Name that bird!

Thank you

Special thanks to:

W3C/MIT for WebEx

WG Participants, Editors & Chairs

The bird