Deploying WebRTC in a Low-Latency Streaming Service

Dr. Alex Gouaillard
Millicast.com
a.k.a.
2 years of WebRTC “Fake News”
or “how to make Brexit look easy in comparison”

Dr. Alex Gouaillard
Millicast.com
“WebRTC was **NOT** designed for one-way 1-to-many”
“WebRTC was NOT designed for one-way 1-to-many”
WebRTC was **NOT** designed for one-way 1-to-many

Permission prompt for viewers!

- For security reasons, you cannot access the camera or microphone without user consent,
- By default the design assumes streams are flowing both ways, and ask for permission in any case.
- Asking people to give permission to access their cam and their microphone to watch a stream is NOT good UX :-)

Dr. Alex Gouaillard - @agouaillard - webrtcbydralex.com
“WebRTC was NOT designed for multiple hops” ?
“WebRTC was **NOT** designed for multiple hops”
WebRTC was **NOT** designed for multiple hops (1)

- **Quality:** RTP/RTCP is a single hop tech  
  - noisy neighbor

- **Security:** SRTP is a hop-by-hop encryption scheme  
  - no end-to-end encryption

- **Scalability:** one media server common is ok, but not two in a row  
  - does not scale
WebRTC was **NOT** designed for multiple hops (2)

- Quality: Noisy Neighbor illustration of RTP/RTCP limits

One viewer (among millions) has a bad network access with high packet loss
WebRTC was **NOT** designed for multiple hops (2)

- Quality: Noisy Neighbor illustration of RTP/RTCP limits

---

Packets loss generates requests for retransmission. The usual way to deal with that is to cache packets in the media server. Solved for 20 years already.
WebRTC was **NOT** designed for multiple hops (2)

- Quality: Noisy Neighbor illustration of RTP/RTCP limits

If too many packets get lost, or if it takes too long for the decoder to wait any longer, a request for a full frame is sent. Only the encoder can answer this request.
WebRTC was **NOT** designed for multiple hops (2)

- Cracked by e.g. CoSMo, integrated e.g. in MilliCast.

A full frame is then created by the encoder and sent to ALL viewers, increasing bandwidth consumption for all. You have a "noisy neighbor".
WebRTC was **NOT** designed for multiple hops (3)

- Security: SRTP is a hop-by-hop encryption scheme

**SECURE BUT NOT SCALABLE**

**Using Peer-to-Peer**

WebRTC encryption is hop-by-hop by design, and only end-to-end encrypted in p2p connections.
WebRTC was **NOT** designed for multiple hops (3)

- Security: SRTP is a hop-by-hop encryption scheme
WebRTC was **NOT** designed for multiple hops (3)

- Solution: end-to-end encryption with user/customer keys a-la telegram.
WebRTC was **NOT** designed for multiple hops (3)

- What can we use for end-to-end encryption?
  - **PERC**: IETF standard for Privacy Enhanced RTP Conferencing (SIP/webRTC)
    - RTP only, Untrusted server, untrusted web-app
  - **WebRTC NV - E2EME**
    - Transport agnostic,
    - Video frame-based
      - Less bandwidth overhead,
      - Codec-Agnostic

Dr. Alex Gouaillard - @agouaillard - webrtcbydralex.com
WebRTC was **NOT** designed for multiple hops (4)

- Bandwidth Evaluation (BWE) and Congestion Control (CC/rmcat)
  - P2p: each can probe the other
  - 1 media server: the media server can probe both peers
  - N media servers …… ?
WebRTC does **NOT** have the tools to Test
WebRTC does **NOT** have the tools to Test
WebRTC does **NOT** have the tools to Test

- **Interoperability: KITE**

KITE Interop SE Grid - Browser configs
(without saucelab, without Mobile, Without Electron [comm])

By 2018, 23 browsers configurations are tested on a daily basis.
https://webrtc.org/testing/kite/

2016: W3C recognize the gap. Google and CoSMo create KITE.
WebRTC does **NOT** have the tools to Test

- **Interoperability: KITE**
- Google uses it (webrtc.org),
- Microsoft uses it (UWP)
- Apple uses it,
- And so on and so forth ….

(*) https://webkit.org/blog/8672/on-the-road-to-webrtc-1-0-including-vp8/

---

**Video Simulcast**

To further improve WebRTC support for multi-party video conferencing, simulcast is now supported for both H.264 and VP8. Kudos to the **libwebrtc** community, including **Cosmo Software**, for making great progress in that important area. Simulcast is a technique that encodes the same video content with different encoding parameters, typically different frame sizes and bit rates. This is particularly useful when the same content is sent to several clients through a central server, called an **SFU**. As

(*) https://webrtc.org/blog/2017/on-the-road-to-webRTC-1-0-including-vp8/
WebRTC does NOT have the tools to Test

- Load testing: KITE

![Graph showing bit rate in bps vs number of participants for different tools: mediasoup, Janus, OpenVidu, Jitsi, and Medooze.](image)
WebRTC does NOT have the tools to Test

- Load testing: KITE

![Graph showing round-trip time in ms. (logarithmic scale) vs. number of participants, comparing OpenVidu, Jitsi, Mediasoup, and Janus.]
WebRTC does **NOT** have the tools to Test

- **Real Time**
  - Image quality Assessment

---

**Video Quality According to Bitrate**

- **NARVAL**
WebRTC always ON encryption is Heavy (CenturyLink)
WebRTC always ON encryption is **Heavy** (CenturyLink)
WebRTC always ON encryption is Heavy (CenturyLink)

- Not since chrome 47 (december 2015)

"These are as strong as 3072-bit RSA keys, but several thousand times faster: call setup overhead with ECDSA is just a few milliseconds."

(*) https://developers.google.com/web/updates/2016/06/webrtc-ecdsa
WebRTC always ON encryption is **Heavy** (CenturyLink)

- for PaaS you can just remove encryption intern.
  - RTP forwarding without transcoding example
    - mediasoup v3
    - Janus SOLEIL
  - RTSP or other non-encrypted protocol

=> *ever seen those hacked American drones feeds?*

- Snowden showed us it was a bad idea though. =>
- Facebook is now forcing RTMP only.
  - Medooze (CoSMo/MilliCast),
    - encryption always ON
    - double encryption
Safari does **NOT** support WebRTC!

The initiative is initially supported by Ericsson, Igalia, Centricular, and Dr Alex Gouaillard. This website and its blog is maintained by Ericsson Research and Dr. Alex Gouaillard.

To further improve WebRTC support for multi-party video conferencing, simulcast is now supported for both H.264 and VP8. Kudos to the libwebrtc community, including Cosmo Software, for making great progress in that important area. Simulcast is a technique that encodes the same video content with different encoding parameters, typically different frame sizes and bit rates. This is particularly useful when the same content is sent to several clients through a central server, called an SFU. As
Edge does **NOT** support WebRTC!

- Well .... Microsoft dealt with that!
  - Edgium.
Encoders in WebRTC are SLOW
Encoders in WebRTC are SLOW
Encoders in WebRTC are SLOW

- Pre-recorded content streaming does not care => decoder opt.
- Reference encoding time: H264 on GPU:
  - Fast (throughput) vs fast (delay)
- Software encoding VP9
- Software encoding AV1
  - libaom
  - SVT-AV1 (and INTEL/NETFLIX announcement @NAB!?)
Encoders in WebRTC are SLOW

Performance Differences On H.264 Hardware Encoding By GPU

Median Encoding Latency Measured in Milliseconds:

- Intel: 11.00
- Nvidia: 5.81
- AMD: 15.06

Dr. Alex Gouaillard - @agouaillard - webrtcbydralex.com
Encoders in WebRTC are **SLOW**

Is 30/6-FPS Real-Time?

Dr. Alex Gouaillard - @agouaillard - webrtcbydralex.com
Encoders in WebRTC are **SLOW**

Throughput is not to be confused with latency

40+fps .......
With 6~7 seconds latency!

Wait .... what?

Heavily parallel
=> high throughput

72 frames deep frame buffer
=> capturer to encoder: 7s

Notion of Real-Time Codec
Encoders in WebRTC are **SLOW**
Real Time group @ AOmedia, anyone?

Focussing on making AV1 a good real-time codec.

- real-time encoding mode
- Bitstream optimization for better bandwidth usage
- RTP payload spec
- RTP-level optimizations for faster encoding decoding in real-time.
Encoders in WebRTC are **SLOW**

Benefits Of Shifted Temporal Prediction Structures

- More balanced bit distribution on the wire (reduced congestion/delay)
  - Rate distribution with concurrent TLO's: 54%, 15%, 16%, 15%
  - Rate distribution with shifted TLO's: 36%, 21%, 28%, 15% (variance ratio 4.5:1)

- More balanced CPU usage at encoder

![Bit Rate Allocation For Non-shifted And Shifted Cases](image-url)
Encoders in WebRTC are **SLOW**

Who wants to break free ~~

Example K-SVC mode: L4T7_KEY_SHIFT
Encoders in WebRTC are **SLOW**
Decoder complexity decrease

To the rescue: Decode Target Information

<table>
<thead>
<tr>
<th></th>
<th>DT0</th>
<th></th>
<th>DT1</th>
<th></th>
<th>DT2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="image1" alt="Frame 1" /></td>
<td><img src="image2" alt="Frame 2" /></td>
<td><img src="image3" alt="Frame 3" /></td>
<td><img src="image4" alt="Frame 4" /></td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image5" alt="Frame 5" /></td>
<td><img src="image6" alt="Frame 6" /></td>
<td><img src="image7" alt="Frame 7" /></td>
<td><img src="image8" alt="Frame 8" /></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Decode target**: The set of frames needed to decode a coded video sequence at a given spatial and temporal fidelity.
- **Decode Target Information (DTI)**: Describes the relationship of a frame to a Decode target.
WebRTC does **NOT** have *name-a-back-end-feature*

- WebRTC is standardized JS API in browsers, and protocols on the wire. There is not standard for signaling or server-side features.

- Recording (e.g.) is not a flash feature, nor an RTMP feature. It’s a feature that was added by some servers that were otherwise also supporting the flash/rtmp protocols.
WebRTC does **NOT** have ABR, that’s why we did it server side (phenixRTS, red5)
WebRTC does NOT have ABR, that’s why we did it server side (phenixRTS, red5)
WebRTC does **NOT** have ABR (phenixRTS, red5)

- To their defense: good choice, at the time they made it, and for people not involved in the standards.
- For previous generation codecs (H.264, VP8)
  - Simulcast is here (chrome 72) thanks to CoSMo contribution.
  - Spatial Scalability was merged on may 2nd.
- For current generation codecs (H.265, VP9)
  - Full scalability (SVC)
- For AV: see previous slides

(*) **Why should we care?** For every standard protocol, you should stick to the standard or risk not interoperate with browsers / web apps. The age of proprietary apps or device is not behind, but corresponding non-OTT market is shrinking.
WebRTC does **NOT** have AAC 5.1 or eq. (MS)

- OPUS can do up to 255 channels by design,
- OPUS 6.1 has now been integrated in chrome (29th April 2019)
What did we learn in the past 2 years in short (1)

The first "maturity" phase where hard problems and false assumptions needed to be addressed is now over.

● 2019 The (IETF) protocols and (W3C) APIs are final.
● Quality, Scalability, Reach, the base is now here, and services exists that show it works well enough.

(next slide)

● It’s a standard, all browsers implement it natively (unlike e.g. SRT).
● WebRTC stack packages are available => client-side SDK are easy.
What did we learn in the past 2 years in short (1)

WebRTC Developer Tools Landscape

- Browsers
  - chrome
  - Firefox
  - Opera
  - Edge

- Media Servers
  - Open source
  - jitsi.org
  - Janvs
  - Mediasoup
  - ant Media
  - REDS PRO
  - Wowza

- Commercial
  - Medooze
  - Ivideon
  - Flashmed
  - Dialogic

- VoIP
  - SIP.js
  - jsSIP
  - Asterisk
  - Freeswitch
  - Kamailio

- Signaling
  - Open source
    - simpleWebRTC
    - Firebase
    - PubNub
    - Pusher
  - Hosted
    - matrix
    - easyRTC
    - EasyRTC

- NAT traversal
  - Open source
    - coturn
    - Restund
    - Pion
  - Hosted
    - XIRSys
    - SignalWire

- CPaaS
  - agor.io
  - api2
  - apigee
  - CAFÉ
  - Kandy
  - Onsip
  - Temasys

- Testing & Monitoring
  - callstats.io
  - testRTC
  - KITE

Dr. Alex Gouaillard - @agouaillard - webrtcbydralex.com
What did we learn in the past 2 years in short (2)

- You’re likely doing it **wrong** if:
  - you have had a WebRTC implementation for many years (FB 2012!),
  - you don’t have an end-to-end layer, and are just chaining servers,
  - you mix with other protocols in a way that requires transcoding in all cases (*),
  - you do **not** have support for all 4 major browsers, desktop and mobile, and native SDKs for mobile (missing desktop and IoT is ok).

(*) Works needs to be done with sources and composition software / hardware to get there. OBS is easy: OpenSource. Some in some cases transcoding for ingest is mandatory, but it should be the exception and not the rule.
For most WebRTC vendors who have stabilized their WebRTC core tech, and address the needs of businesses which could not compromise on latency, the goal is now: “**feature parity**”

- increase the value for **existing customers**, like e.g.
  - (Encrypted) Recording,
  - Monitoring, Analytics, Compliance,
  - End-to-end Media Encryption,

- enable **additional revenue models**, like e.g.
  - Ad insertion (SCTE35, VAST, ….)

- enable yet other revenue models to run on webrtc by **protecting high-value content**
  - Watermarking and DRM,
Hi Alex,

April was a very exciting month for Limelight Networks, and I want to make sure you are up to date on all the news!

FIRST, Limelight announced new features to our Video Delivery Services, including streamlining the process of protecting premium online video content with forensic watermarking at the edge, and the availability of server-side ad insertion capabilities through the Adapex Dynamic Ad Insertion platform.

---

Viewers are rapidly transitioning to live streams, and taking brands’ ad dollars with them. To stay relevant, ad-supported live event streaming must be part of your engagement model. Video encoding in the cloud to stream large-scale live events, gives you details on encoding video with minimal latency and multi-cloud resiliency to deliver a better user experience that drives loyalty and increases monetization.

Dr. Alex Gouaillard - @agouaillard - webrtcbydralex.com
WebRTC Tech Innovations to come 2020

- According to Google:
  - WebRTC Group: AV1, QUIC, WASM
- According to Facebook:
  - QUIC (Open Sourced stack on April 30th, 2019)
- According to AOMedia, INTEL and Netflix:
  - AV1 (*) (NAB Press Release)
How to keep in touch with this rapid innovating field?

1. Join the standard committees: W3C, IETF, AOMedia, ITU, MPEG, …..

2. Read my tweets :-)  

@agouaillard,

Warning: tech-only, tainted with some french accent and arrogance.