

AVTCORE WG

Virtual Interim

Thursday, January 28, 2021
9:30 - 11 AM Pacific Time

Mailing list: avtcore@ietf.org

Jabber Room: [avtcore@jabber.ietf.org](jabber:avtcore@jabber.ietf.org)

Meeting link: <https://meet.jit.si/avtcore> Password: interim-meeting

Virtual bluesheets and notes: [Audio/Video Transport Core Maintenance \(avtcore\) Working Group - CodiMD \(ietf.org\)](#)

Note Well



This is a reminder of IETF policies in effect on various topics such as patents or code of conduct. It is only meant to point you in the right direction. Exceptions may apply. The IETF's patent policy and the definition of an IETF "contribution" and "participation" are set forth in BCP 79; please read it carefully.

As a reminder:

- By participating in the IETF, you agree to follow IETF processes and policies.
- If you are aware that any IETF contribution is covered by patents or patent applications that are owned or controlled by you or your sponsor, you must disclose that fact, or not participate in the discussion.
- As a participant in or attendee to any IETF activity you acknowledge that written, audio, video, and photographic records of meetings may be made public.
- Personal information that you provide to IETF will be handled in accordance with the IETF Privacy Statement.
- As a participant or attendee, you agree to work respectfully with other participants; please contact the ombudsteam (<https://www.ietf.org/contact/ombudsteam/>) if you have questions or concerns about this.

Definitive information is in the documents listed below and other IETF BCPs. For advice, please talk to WG chairs or ADs:

- [BCP 9](#) (Internet Standards Process)
- [BCP 25](#) (Working Group processes)
- [BCP 25](#) (Anti-Harassment Procedures)
- [BCP 54](#) (Code of Conduct)
- [BCP 78](#) (Copyright)
- [BCP 79](#) (Patents, Participation)
- <https://www.ietf.org/privacy-policy/>(Privacy Policy)

Virtual Meeting Tips

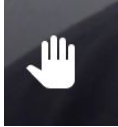
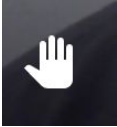

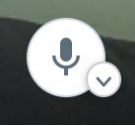
[Upcoming Meetings \(ietf.org\)](https://www.ietf.org)

This session is being recorded

- **No registration required to attend the meeting**
- **Please fill in virtual bluesheets (datatracker login required):**
 - [Audio/Video Transport Core Maintenance \(avtcore\) Working Group - CodiMD \(ietf.org\)](#)
- **Join the session Jabber room via IETF Datatracker Meeting icon:**
[Upcoming Meetings \(ietf.org\)](#)
- **Please use headphones when speaking to avoid echo.**
- **Please state your full name before speaking.**

Virtual Meeting Tips

This session is being recorded

- To enter the queue, type “+q” in chat, leave by typing “-q”
- To answer a hum, raise your hand with , lower it with .
- When you are called on, you need to enable your audio to be heard.
- Audio is enabled by unmuting  and disabled by muting .
- Video is encouraged to help comprehension but not required.

About this meeting



- Agenda:
<https://datatracker.ietf.org/doc/agenda-interim-2021-avtcore-01-avtcore-01/>
- CodiMD (for notes): [Audio/Video Transport Core Maintenance \(avtcore\) Working Group - CodiMD \(ietf.org\)](#)
- Jabber Room: [avtcore@jabber.ietf.org](jabber:avtcore@jabber.ietf.org)
- Secretariat: mtd@jabber.ietf.org
- WG Chairs: Jonathan Lennox & Bernard Aboba
- Jabber Scribe:
- Note takers:

Agenda



1. Note Well, Note Takers, Agenda Bashing, Draft status - (Chairs, 10 min)
2. JPEG XS Payload Format (T. Bruylants, 10 mins)
<https://tools.ietf.org/html/draft-ietf-payload-rtp-jpegxs>
3. Frame Marking WGLC (Chairs, 15 mins)
<https://tools.ietf.org/html/draft-ietf-avtext-framemarking>
4. VP9 Payload Format (Jonathan Lennox, 10 mins)
<https://tools.ietf.org/html/draft-ietf-payload-vp9>
5. SFrame RTP Encapsulation (Youenn Fablet & Sergio Garcia Murillo, 15 min)
6. QRT: QUIC RTP tunneling (Samuel Hurst, 10 mins)
<https://tools.ietf.org/html/draft-hurst-quic-rtp-tunnelling>
7. Wrapup and Next Steps (Chairs, 5 min)

Draft status

- Published
 - RFC 8817: was draft-ietf-payload-tsvcis
 - RFC 8852: was draft-ietf-avtext-rid
 - RFC 8860: was draft-ietf-avtcore-multi-media-rtp-session
 - RFC 8861: was draft-ietf-avtcore-rtp-multi-stream-optimisation
 - RFC 8872: was draft-ietf-avtcore-multiplex-guidelines
 - RFC 8888: was draft-ietf-avtcore-cc-feedback-message

Draft Status (2)

- Completed WGLC
 - draft-ietf-payload-vp9
 - draft-ietf-avtcore-multi-party-rtt-mix
 - draft-ietf-avtext-framemarking (3rd WGLC)
 - draft-ietf-payload-rtp-jpegxs
- Expired
 - draft-ietf-payload-tetra (expired January 27, 2020)
- Adopted
 - draft-ietf-avtcore-rtp-enc (was draft-zhao-avtcore-rtp-enc)
 - draft-ietf-avtcore-rfc7983bis (was draft-aboba-avtcore-rfc7983bis)
 - draft-uberti-avtcore-cryptex (not submitted as WG draft yet)

JPEG XS Payload Format

<https://tools.ietf.org/html/draft-ietf-payload-rtp-jpegxs>

T. Bruylants

JPEG XS Payload Format (1)

- Last WGLC
 - No response was given
 - Was unaware that this would be an issue :(
- Actions taken
 - Asked important stakeholders to join avtcore WG
 - Fraunhofer
 - VSF
 - Other organizations support
 - AIMS
 - JPEG committee (ISO/IEC SC29 WG1 issued a liaison letter at 90th meeting)

JPEG XS Payload Format (2)

- Next todo
 - Request to issue a new WGLC
 - Stakeholders will now follow up on the WGLC and provide a proper response

Frामemarking WGLC

<https://tools.ietf.org/html/draft-ietf-avtext-frामemarking>

Chairs (15 minutes)

Frामemarking WGLC

- **Announced 21 November 2020, concluded 6 December 2020:**

<https://mailarchive.ietf.org/arch/msg/avt/LReN9QCN8tTsYZ0AfaLdanV2BpY/>

- **Responses:**

- **Stephan Wenger:**

https://mailarchive.ietf.org/arch/msg/avt/ipquV2n_YfWpGEanicsYDWGyBPc/

<https://mailarchive.ietf.org/arch/msg/avt/4XDuttNyJSeKI35AB-IN3nVGn6w/>

- **Sergio Garcia Murillo, Dr. Alex: +1**

- **Bernard Aboba:**

<https://mailarchive.ietf.org/arch/msg/avt/SznfLrr7YorwYjPEYXdIH5AU4VA/>

Stephan Wenger: Comment #1

All:

This is not a hard objection, but I want to inquire if there is anyone out there who continues to think that frame marking as originally proposed is still a good idea and will see implementations. The main reason I want to know is that accepting frame marking as an RFC would imply, per previous WG agreement, that future video codec payload formats include a (mandatory?) section on how to map the codepoints in the frame marking draft to the codec in question. That's non-trivial cost and effort, and chances are that effort will grow over time as codecs develop beyond what was mainstream in 2016.

What triggered my thinking about dumping frame marking are a) webrtc's removal of frame marking as a required to implement technology, b) the decreasing relevance, as I perceive it, of SRTP for which frame marking is predominantly designed, and c) the myriad of new ideas in the IETF that skin the secure-SFU cat in different ways (sframe among them, but not the only one).

Stephan

Stephan Wenger: Comment #2

The problem both frame marking and sframe try to solve seems to provide a MANE or SFU sufficient information to do its job—selective forwarding but also hop-by-hop repair and such—without being themselves trusted. Both frame marking and sframe attempt (or appear to attempt in the context of sframe—the design is too new to be sure) to abstract from the syntax of the various codecs. This is sensible from an SFU maker’s viewpoint—they want to reuse the same logic independently from the codec in use. But it’s hard to do. As a historical anecdote, we video coding people in the ITU and MPEG tried the same since ~2000 (starting with H.264v1), and when we finished any of the versions we were quite convinced that the NAL unit header plus the context info from the capability exchange (including things like parameter sets and later the SSEI and such) would be sufficient for informing an SFU. They are not. We got it wrong every time, despite many, many more eyes looking into this over in JVET and its predecessors than avtcore or sframe have typically available today. H.264/SVC/H.265/SHVC SFUs in practice all need to look into the slice header, and we heard during the session from Justin that the VP9/AV1(?) based SFU designs do the same.

Call for Framemarking Implementation Experience

- Issued 22 November 2020:

<https://mailarchive.ietf.org/arch/msg/avt/xx0cKHmoXBmWFBFnfUR2Sy8sdaA/>

- Responses

- Sergio Garcia Murillo (VP8, VP9):

<https://mailarchive.ietf.org/arch/msg/avt/xx0cKHmoXBmWFBFnfUR2Sy8sdaA/>

<https://mailarchive.ietf.org/arch/msg/avt/aOufzIzicXOXpP-1s48IrGUIraU/>

Issue with VP8 PictureID prevents forwarding without payload modification

VP8 receivers need TLOPICIDX to be rewritten if simulcast layers are spliced

Problem *not* specific to framemarking

Issue with VP9 P/U bit (for temporal/spatial upswitch)

Inability to support VP9 K_SVC scalability modes

Frame Marking Implementation Experience (cont'd)

- Jonathan Lennox (H.264):

<https://mailarchive.ietf.org/arch/msg/avt/fcJzawe-A-9cqh0u3pS9Gp0fEs/>

<https://mailarchive.ietf.org/arch/msg/avt/6BxD7yxoEeRUQ9zy4wMXPtvaVJ0/>

Implemented to support temporal scalability (3 layers) in H.264/AVC

Assumed temporal nesting, so every frame is a valid upswitch point

Contributed to webrtc.org code base (subsequently removed)

Questions

- **Given the implementation experience, what is the way forward for the framemarking specification?**
 - Document the issues?
 - Remove requirement for payload specs to support it?
 - Change publication status?
 - Something else?

VP9 Payload Format

<https://tools.ietf.org/html/draft-ietf-payload-vp9>

Jonathan Lennox

VP9 Payload Format

- **One issue raised with VP9's framemarking support: framemarking can't describe all VP9 scalability modes**
- **IMO this is a limitation of framemarking, not of VP9.**
 - Framemarking can't describe K-SVC scalability
 - Limited for flexible mode.
- **Should we:**
 - Document the issue?
 - Drop framemarking support for VP9?
 - Something else?

SFrame RTP Encapsulation

Youenn Fablet & Sergio Garcia Murillo

Goals

Support post-encoding/pre-decoding media transforms

- SFrame
- Insertable Streams

Minimize impact on intermediaries processing

- SFUs
- Browsers

Potential for simplification?

- Adding a new codec to an SFU takes some effort
- Adding a new packetizer to browsers takes some effort

What is needed?

A processing model

- Packetizer can no longer split frames with codec-specific information
 - Application that feeds the packetizer needs to do it

A generic packetization with side-channel information

- Intermediaries need some information about the content

A way to negotiate the use of the generic packetization

- Make use of the generic packetization approach or not

Processing Model

Proposal

- Encoder generates a frame
- Application modifies the frame
 - Application MAY split the frame in individual sub frames with metadata
- Packetizer handles each sub-frame and its metadata as an independent frame to transmit

SFrame example

- H.264 encoder generates a frame, SFrame encrypts it as one frame, packetizer sends it as one frame
- SVC encoder generates a frame with different scalability layers
 - Each layer is encrypted by SFrame as an individual frame
 - Each individual frame is sent as a standalone frame to the packetizer

Generic packetization + side-channel metadata



Proposal

- Frame data is sent as an opaque application payload
 - No data prepended or appended to the application payload by the packetization
 - Packetizer fragments the payload in several RTP packets if too big
- Frame metadata is sent as RTP header extension data
 - Information used by SFUs for their processing
 - Codec, profile, frame type...
 - Information exposed by insertable streams prior the transform
 - <https://w3c.github.io/webrtc-insertable-streams/#rtccodedaudioframe>
 - <https://w3c.github.io/webrtc-insertable-streams/#rtccodedvideoframe>

Generic packetization negotiation

Goals

- Allow to negotiate codecs and formats as done today
- Allow peers to bail out if the other side does not support generic packetization
- Allow peers to identify that a stream payload is to be treated as opaque
- Allow use for both audio and video

Several approaches

- Out-of-band negotiation, SDP negotiation, SDP negotiation & in-stream signalling
- Associated to an already existing codec payload type or independent
- Multiplex several codecs in a payload type or map each payload type to a specific codec/format

Option 1: A generic payload type per defined codec

```
a=rtpmap:96 vp9/90000
a=fmtp:96 profile-id=0
a=rtpmap:97 generic/90000
a=fmtp:97 apt=96
a=rtpmap:98 rtx/90000
a=fmtp:98 apt=96
a=rtpmap:99 rtx/90000
a=fmtp:99 apt=97
a=rtpmap:100 vp8/90000
a=rtpmap:101 generic/90000
a=fmtp:101 apt=100
```

- Generic negotiation relies on negotiation of the standard one
- No in-band cost
- Consumes several payload types

Option 2: An independent generic payload type per codec

```
a=rtpmap:98 generic/90000
a=fmtp:98 codec=vp9; profile-id=0
a=rtpmap:99 rtx/90000
a=fmtp:99 apt=98
a=rtpmap:100 generic/90000
a=fmtp:100 codec=vp8
a=rtpmap:101 rtx/90000
a=fmtp:101 apt=100
```

```
a=rtpmap:98 generic/90000
a=fmtp:98 codec=vp09.00.20
a=rtpmap:99 rtx/90000
a=fmtp:99 apt=98
a=rtpmap:100 generic/90000
a=fmtp:100 codec=vp8
a=rtpmap:101 rtx/90000
a=fmtp:101 apt=100
```

- Generic negotiation is independent on negotiation of the standard one
 - Codec/profiles in [rfc6381](#) format could be used
- No in-band cost
- Best suited to provide offers with either generic or codec specific packetizations, not both

Option 3: A generic payload type for all codecs



```
a=rtpmap:96 vp9/90000
a=rtpmap:97 vp8/90000
a=rtpmap:98 generic/90000
a=rtpmap:99 rtx/90000
a=fmtp:99 apt=96
a=rtpmap:100 rtx/90000
a=fmtp:100 apt=97
a=rtpmap:101 rtx/90000
a=fmtp:101 apt=98
```

- Generic negotiation relies on negotiation of the standard one
- Requires sending actual codec payload type, as a RTP header extension
 - Potential network overhead
 - Requires receiver to be able to change codec on a per frame basis
- Requires negotiating different payload types for each clock rate for audio



Option 4: A RTP header extension to switch on/off

```
a=rtpmap:96 vp9/90000
a=rtpmap:97 vp8/90000
...
a=extmap:10 urn:example.org:rtp-hdext:generic
```

- Works well with existing RTP header extension negotiation
 - Support of the RTP header extension mandates generic packetization support
- Requires sending packetization mode
 - Potential network overhead
 - Well suited for dynamic choice of the packetization mode
 - Requires depacketizer to change mode on a per frame basis

Frame metadata RTP header extension



Identified metadata of interest

- At which packet SFU can switch (SVC and simulcast)
- Resolution and more generally stream 'quality': frame rate, bit rate...
- Codec specific information like profile/levels
- Recovery mechanism required in case of loss (none, RTX, LRR/PLI)
- Opus TOC to know frame length (recording scenarios)

Potential proposals

- Use/extend frame marking
- Use/extend AV1 Dependency Descriptor
- Design a new RTP header extension
 - Complemented with either frame marking or dependency descriptor

Current Draft

- Available on GitHub
 - <https://github.com/murillo128/codec-agnostic-rtp-payload-format/blob/master/codec-agnostic-rtp-payload-format.md>
- Negotiate a single PT for the generic payload type (option 2)
 - Send the associated payload type as a RTP header extension
 - Multiplexing of codec specific payload types over the generic PT
- Use AV1 dependency descriptor for SVC metadata

QRT: QUIC RTP Tunneling

<https://tools.ietf.org/html/draft-hurst-quic-rtp-tunnelling>

Samuel Hurst

QUIC RTP TUNNELLING

Context

- BBC R&D has been looking at **Contribution Transport Protocols** for high-quality, low-latency ingest of media over the public internet
- RTMPS is predominately used for contribution currently, and we have also looked at SRT and RIST
- QRT inspired by the Video Services Forum's RIST Main Profile, but wrapping RTP in QUIC instead of GRE

QUIC RTP TUNNELLING

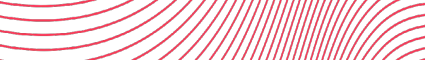
Benefits of QUIC as RTP Transport

- Built on UDP with selective acknowledgements for loss detection
- Strong encryption (replace DTLS)
- 1-RTT and 0-RTT connection setup
- Connection migration

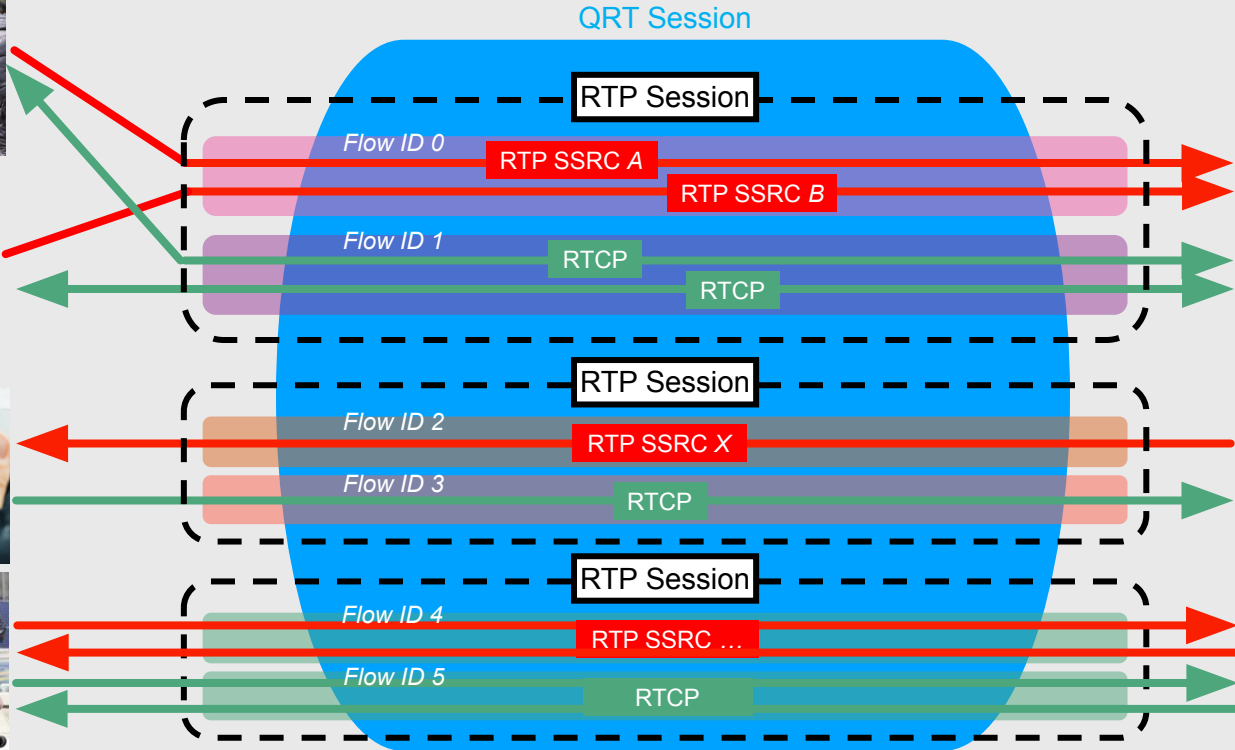
QUIC RTP TUNNELLING

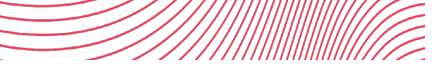
QRT Introduction

- Deliberately simple mapping layer
- Uses the QUIC DATAGRAM extension frame
- Introduces the concept of a QRT Session which can carry multiple RTP Sessions
- UDP port numbers for the RTP/RTCP port pair is replaced with a 62-bit QRT Flow Identifier
- QUIC transport loss detection replaces RTCP Generic NACK

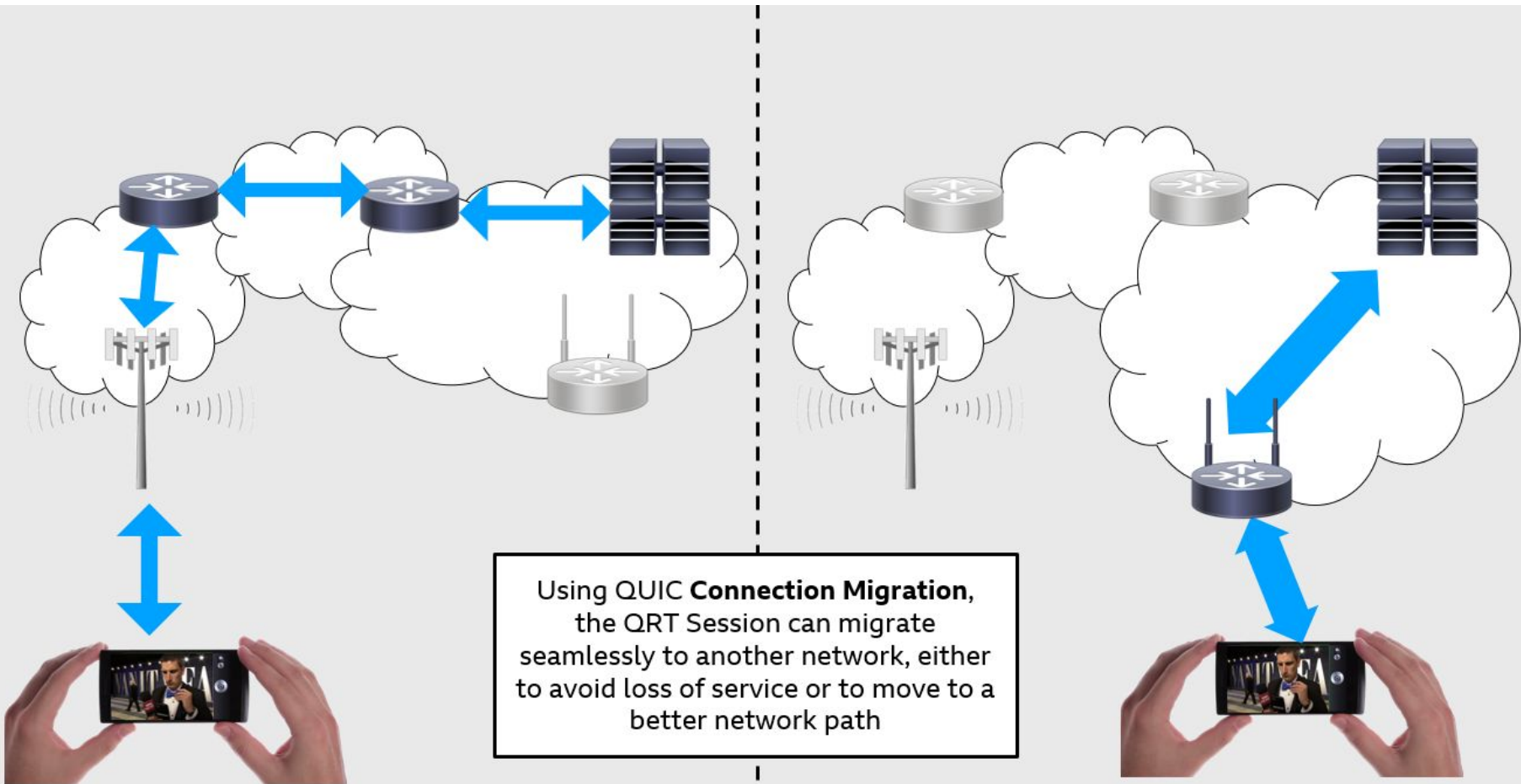


QRT Contribution Use Case





QRT Contribution Use Case



QUIC RTP TUNNELLING

QRT Draft Status

- draft-01 is in progress thanks to previous feedback
- Work underway on an implementation in order to gain experimental experience
- Current road map includes:
 - Carrying SIP in STREAM frames
 - Multipath
 - Session/flow prioritisation
 - Sharing DATAGRAMs with other protocols

QUIC RTP TUNNELLING

Thank you

Sam Hurst

samuelh@rd.bbc.co.uk

<https://www.github.com/bbc/quic-rtp-tunnelling-draft>

<https://tools.ietf.org/html/draft-hurst-quic-rtp-tunnelling>



Thank you

Special thanks to:

The Secretariat, WG Participants & ADs