AVTCORE WG
IETF 111

Virtual Meeting

Monday, July 26, 2021

16:00 - 18:00 Pacific Time

Session III, Room 1

Mailing list: avtcore@ietf.org
Jabber Room: avtcore@jabber.ietf.org
MeetEcho link: https://ws.conf.meetecho.com/conference/?group=avtcore

IETF 111 Meeting Tips

https://www.ietf.org/how/meetings/111
https://datatracker.ietf.org/meeting/agenda

This session is being recorded

- **IETF 111 registration and a datatracker login required to attend**
- **No need to manually fill in blue sheets, it's automatic.**
- **Join the session Jabber room via IETF Datatracker Meeting agenda**
- **Please use headphones when speaking to avoid echo.**
- **Please state your full name before speaking.**

IETF 111 Meeting Tips

https://www.ietf.org/how/meetings/111
https://datatracker.ietf.org/meeting/agenda

This session is being recorded

- **Enter the queue with**  **, leave 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 can also be enabled, but it is separate from audio.**
- **Video is encouraged to help comprehension but not required.**

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)

About this meeting

- Agenda: https://datatracker.ietf.org/doc/agenda-111-avtcore/
- CodiMD (for notes): https://codimd.ietf.org/notes-ietf-111-avtcore
- Jabber Room: avtcore@jabber.ietf.org
- Secretariat: mtd@jabber.ietf.org
- WG Chairs: Jonathan Lennox & Bernard Aboba
- Jabber Scribe:
- Note takers:

Other Meetings This Week

- Video Ingest over QUIC side meeting
- Friday, July 30 at 18:00 UTC (11:00 Pacific)
- For details, see:
  - https://trac.ietf.org/trac/ietf/meeting/wiki/111sidemeetings

Agenda

1. Note Well, Note Takers, Agenda Bashing, Draft status - (Chairs, 10 min)
2. 16:10 - 16:20 Cryptex (Sergio Garcia Murillo, 10 min)
   https://datatracker.ietf.org/doc/html/draft-ietf-avtcore-cryptex
3. 16:20 - 16:35 RTP Payload for VVC (Shuai Zhao, 15 min)
   https://datatracker.ietf.org/doc/html/draft-ietf-avtcore-rtp-vvc
4. 16:35 - 16:55 RTP over QUIC (J. Ott & M. Engelbart, 20 min)
   https://datatracker.ietf.org/doc/html/draft-engelbart-rtp-over-quic
5. 16:55 - 17:45 SPacket (Sergio Garcia Murillo & Youenn Fablet, 50 min)
   https://datatracker.ietf.org/doc/html/draft-murillo-avtcore-multi-codec-payload-format
6. 17:45 - 18:00 Wrapup and Next Steps (Chairs, 15 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
  - RFC 9071: was draft-ietf-avtcore-multi-party-rtt-mix

Draft Status (2)

- RFC Editor Queue
  - draft-ietf-payload-vp9 (MISSREF)
- IESG evaluation: AD followup
  - draft-ietf-payload-rtp-jpegxs
- Waiting for Writeup (all issues resolved)
  - draft-ietf-avtext-framemarking
- Expired and dropped from milestones
  - draft-ietf-payload-tetra (expired January 27, 2020)
- Adopted
  - draft-ietf-avtcore-rtp-evc
  - draft-ietf-avtcore-rtp-vvc (ready for WGLC?)
  - draft-ietf-avtcore-rfc7983bis
  - draft-ietf-avtcore-cryptex (ready for WGLC?)

Completely Encrypting RTP Header Extensions and Contributing Sources (Cryptex)

https://datatracker.ietf.org/doc/html/draft-ietf-avtcore-cryptex

Sergio Garcia Murillo

draft-02

- Added test vectors to the draft
- Two implementations
  - https://github.com/jitsi/jitsi-srtp/pull/29
  - https://github.com/cisco/libsrtp/pull/511
- Only one (editorial) issue pending:
  - Clarify AAD for AEAD modes

Next Steps

- WGLC

RTP Payload Format for VVC

Shuai Zhao, Stephan Wenger (Tencent)

Yago Sanchez (Fraunhofer HHI)

Ye-Kui Wang (Bytedance Inc)

https://datatracker.ietf.org/doc/html/draft-ietf-avtcore-rtp-vvc

Update Summary

- Addressed all remaining editor's notes.
- Added SDP O/A in section 7.2.2
- bug fixes
- What is still missing
- Request WGLC

Editor-note #11, #19, #20, #24

- #11: along with other fixes in the specification regarding using base64[RFC4648] for VVC parameters
- #19: Remove sprop-depack-buf-nalus
  - The justification is that: the mechanism proposed in HEVC for minimizing startup delay by using packetization level buffer has never been implemented in practice. For simplify, we remove the support of this parameter and along with update in Section 6 when talking about de-packetization process
- #20: Section 7.2.1. Mapping of Payload Type Parameters to SDP
  - update the complete list of optional parameters.
  - mostly HEVC languages, with VVC unique parameter (ex. sprop-dci)
- #24: Section 4.3.3
  - We introduced the usage of the 'R' bit in FU header since IETF 110, -08.
    - -"When set to 1, the R bit indicates the last NAL unit of a coded picture, i.e., the last byte of the FU payload is also the last byte of the coded picture. When the FU payload is not the last fragment of a coded picture, the R bit MUST be set to 0".
  - Update in -09, replace R to P(icture)

Editor-note #14, #21

- #14 (page 40): Bug fix. Update the interop-constraints default values.
- #21: Completed the Usage with SDP Offer/Answer Model
  - mostly HEVC language with following update for VVC:
- 7.2.2. Usage with SDP Offer/Answer Model
  - **Non-scalable media format configuration**
  - **Scalable media format configuration**
  - **Payload format configuration**
  - **Multicast**
- 7.2.3. Usage in Declarative Session Descriptions
- 7.2.4. Considerations for Parameter Sets

Still Missing…

Section 11 IANA Considerations:

> “A new media type, as specified in Section 7.1 of this memo, has been registered with IANA”.

Request WGLC on revision 10

- We believe -10 is ready for WGLC
- Small bugs, formatting issues or Typos may exist which can be addressed during WGLC.

RTP over QUIC

J. Ott, M. Engelbart

https://datatracker.ietf.org/doc/html/draft-engelbart-rtp-over-quic

Motivation

- Provide an encapsulation for carrying RTP/RTCP over QUIC
- Leverage information already available in QUIC
- Adjust how to use RTCP

- Discuss how to interact with QUIC congestion control

Related Work

- Similar draft: QRT: QUIC RTP Tunneling (draft-hurst-quic-rtp-tunnelling-01) by Sam Hurst (BBC Research & Development)
  - Leverage QUIC connection state to reduce exchange of RTCP packets
- Both drafts use QUIC's Unreliable Datagram Extension (draft-ietf-quic-datagram-03)
- Currently using Flow IDs in Datagrams to demultiplex RTP Sessions
- Use SDP for signalling

- Recent related work: RUSH: A/V over non-RTP over QUIC
  - draft-kpugin-rush-00.html

RTP/RTCP

- Early background: draft-rtpfolks-quic-rtp-over-quic-01

- RTCP traditionally provides feedback information
  - Round-trip time, jitter, packet losses – at varying granularity in resolution and time
  - RR, Generic NACK, XR, ...
- RFC 8888 defines detailed congestion control feedback
  - Per-packet receipt / loss reports + detailed timing
  - To feed into NADA, SCReAM, GCC, ...
- QUIC naturally gathers partly similar packet statistics
  - Sometimes limited resolution: e.g., only aggregate timing per ACK, though
  - Exact loss information (also for QUIC DATAGRAM frames)
  - RTT measurements every few packets
  - Packet “timestamp” for every packet that triggers an ACK
    - Which may or may not carry a DATAGRAM frame
- Where possible, use QUIC’s information instead of redundant RTCP packets
  - May require interpolation at the sender

Local Interface Requirements

QUIC Implementation

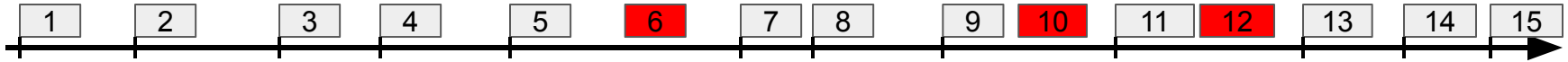
- Unreliable QUIC DATAGRAM frames MUST be supported
- ACKs/Loss of DATAGRAM frames MUST be signalled to the application
- RTT statistics MUST be exposed to application

Congestion Controller

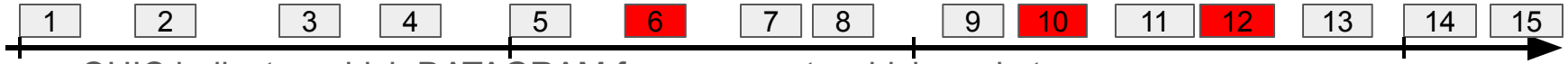
- Assuming one of the algorithms proposed by RMCAT for real-time media
- Input: ACKed packets, delay, RTT estimations, optionally ECN
- Output: bandwidth estimation for media encoder
- QUIC internal vs. external CC: What about QUIC streams and non-RTP DATAGRAM frames?

Some details (see I-D)

RTP + RFC 8888



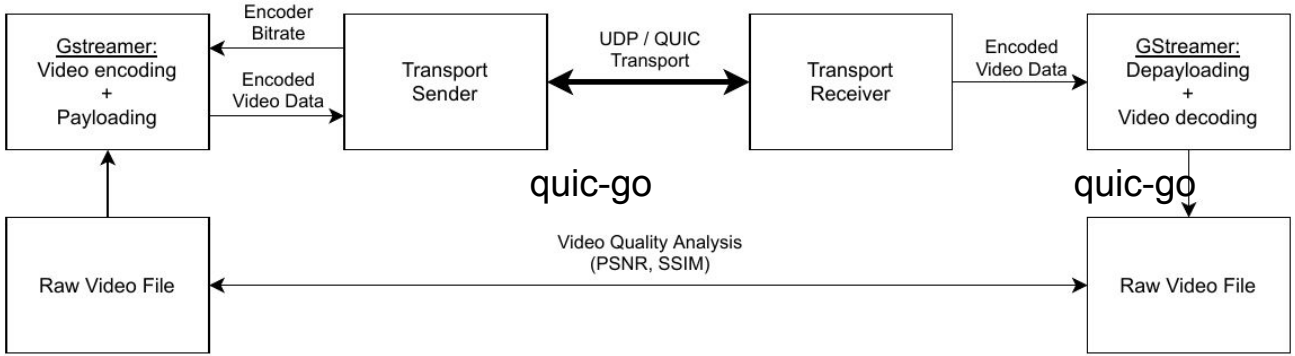
QUIC DATAGRAM frames



- QUIC indicates which DATAGRAM frames map to which packets
  - and when they were actually sent
- QUIC indicates which packets were received / lost
  - along with the reception time of the QUIC packet that triggered the ACK
- QUIC indicates RTT stats (min_rtt, smoothed_rtt, rtt_var)
- Use the above to interpolate the reception times of the other RTP packets
- Option: use QUIC TS extension for one-way delay estimation

Experiments

- Exploring different approaches to interpolation
  - for different CC algorithms, different reporting, ...
- Currently: gstreamer (w/ SCReAM | plain) over (quic-go | plain UDP)
  - quic-go with congestion control disabled for DATAGRAM frames



Quick note on signaling

- Connection-oriented operation [RFC 8122]
- Bundling of RTP sessions into a single transport connection [RFC 8843]
- Multiplexing RTP and RTCP [RFC 5761]
  - Possibly rtcp-mux-only [RFC 8858]

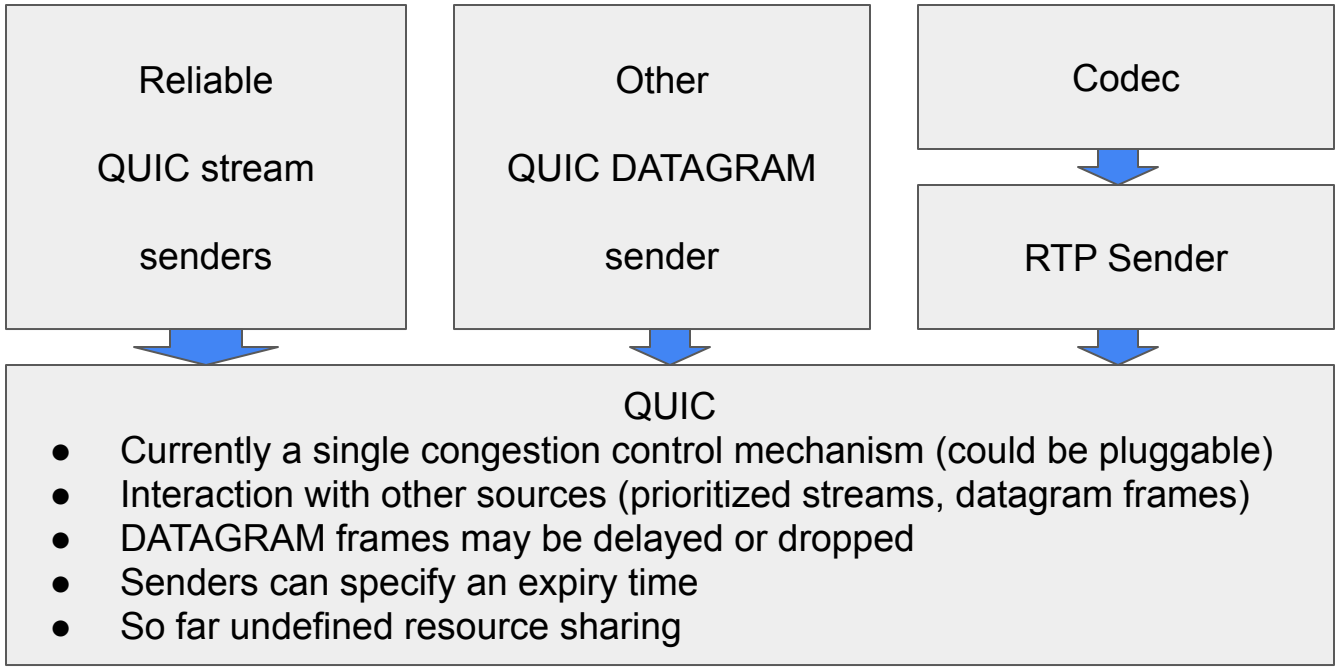
Example

```
a=group:BUNDLE abc xyz

m=audio 10000 QUIC/RTP/AVP 0 8
a=setup:actpass
a=connection:new
a=fingerprint:SHA-256 \
12:DF:3E:5D:49:6B:19:E5:7C:AB: … :DF:3E:5D:49:6B:19:E5:7C:AB:4A:AD
b=AS:200
a=mid:abc
a=rtcp-mux
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=extmap:1 urn:ietf:params:<tbd>

m=video 0 QUIC/RTP/AVP 97
b=AS:1000
a=bundle-only
a=mid:xyz
a=rtcp-mux
a=rtpmap:97 H264/90000
a=extmap:2 urn:ietf:params:<tbd>
```

Meta question: congestion control interaction?



Wrapping up...

Is this useful?

Further questions to be asked or answered?

To be seen in the context of other media-over-QUIC efforts

Multi Codec RTP payload format

Sergio Garcia Murillo, Youenn Fablet

https://www.ietf.org/id/draft-murillo-avtcore-multi-codec-payload-format-01.html

Applying SFrame on a per packet base (SPacket)

- E2E Encryption is applied on a per packet base (SPacket).
- E2E Encryption is done after the RTP packetization, but before RTX/FEC processing and SRTP encryption.
- SPacket does not provide more resilience to packet losses than applying SFrame per IDU.
- SFrame is equal to SFrame for Audio
- SPacket adds some overhead compared to SFrame for video: E2E metadata (key id +IV+counter) + authentication tag is added on each packet instead of each IDU.
- SPacket is not compatible with WebRTC insertable streams.

SPacket & SFrame common needs

SPacket does not require a new packetizer, but the RTP encrypted packet is not a valid RTP packet according to the standard packetizations, so:

- Still requires to be negotiated in the SDP.
- Still requires to be able to differentiate between the encrypted/unencrypted packets at RTP level.
- Still requires a way to signal media metadata so SFU can perform layer switching/forwarding.

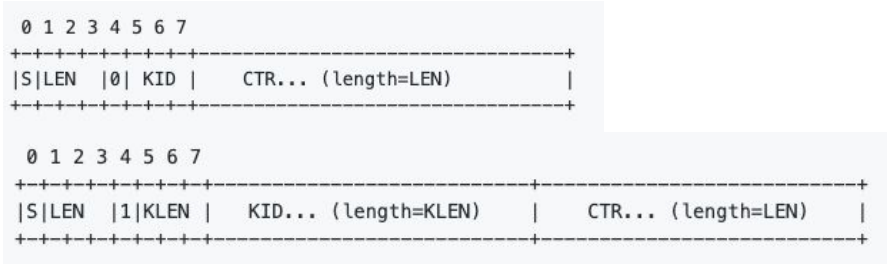
SFrame vs SPacket overhead

Metadata Header		key id<7	kid<255	kid<65535
	1	0	1	2
ctr<255	1	2	3	4
ctr<65535	2	3	4	5
ctr<16777215	3	**4**	**5**	6

Authentication tag len	
AES_GCM_128_SHA256	**16**
AES_CM_128_HMAC_SHA256_64	**8**
AES_CM_128_HMAC_SHA256_32	4

kbps	fps	pps	Overhead
500	30	49	1.633333333
1000	30	97	3.233333333
2500	30	241	8.033333333

CTR 2 bytes len	SFrame	SPacket
500	0:36:24	0:22:17
1000	0:36:24	0:11:15
2500	0:36:24	0:04:31



Overhead in kbps AES_CM_128_HMAC_SHA256_8 kid<7			
Video bitrate	**SFrame**	**SPacket**	**SPacket** (4bytes auth)
500 kbps	3 kbps	5 kbps	3 kbps
1000 kbps	3 kbps	9 kbps	6 kbps
2500 kbps	3 kbps	23 kbps	15 kbps

Overhead in kbps AES_GCM_128_SHA256 kid>7 kid<255		
Video bitrate	**SFrame**	**SPacket**
500 kbps	5 kbps	8 kbps
1000 kbps	5 kbps	16 kbps
2500 kbps	5 kbps	40 kbps

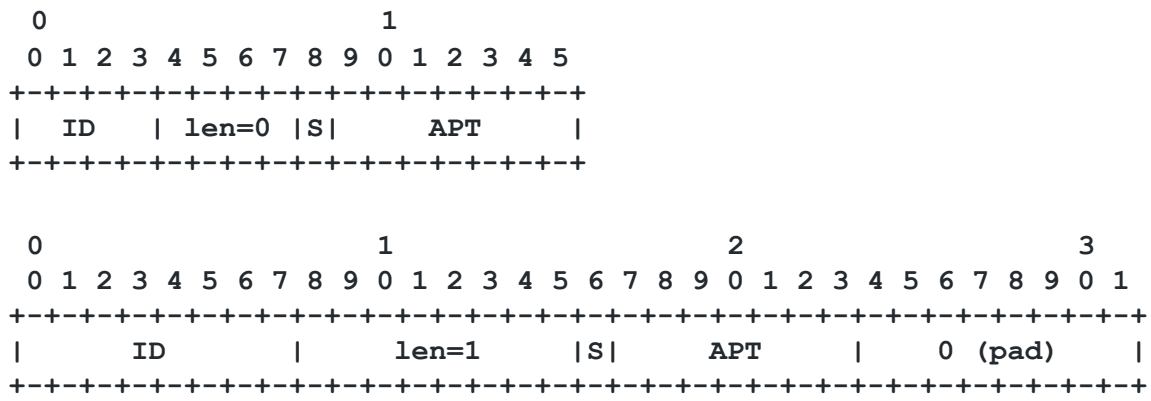
SDP negotiation

```
a=rtpmap:96 vp9/90000
a=rtpmap:97 vp8/90000
a=rtpmap:98 opaque/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
```

- Encrypted negotiation relies on negotiation of the standard packetizer for the each codec.
- Negotiate an opaque payload type for all codecs in order to avoid duplication or triplication the number of payload types in use.
- Requires sending actual codec payload type, as a RTP header extension or as a prefix to the payload, which will cause minor network overhead.
- Requires negotiating different payload types for each clock rate for audio.

Payload Type Multiplexing

- In order to reduce the number of payload type in the SDP exchange, a single payload type is used
- That requires to identify the original payload type of the negotiated media format, called the associated payload type (APT).
- The APT value is sent in a dedicated header extension:



- The APT value is the associated payload type value.
  - The S bit indicates if the media stream can be forwarded safely starting from this RTP packet.
  - Receivers MUST be ready to receive RTP packets with different associated payload types
- The URI for declaring this header extension in an extmap attribute is

`urn:ietf:params:rtp-hdrext:associated-payload-type`

Frame Metadata

Potential 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 solutions

- Use AV1 Dependency Descriptor for all supported video codecs.
- Design a new RTP header extension, potentially complemented with either frame marking or AV1 Dependency Descriptor.

SFrame delta from SPacket

SFrame only requires to define a multi codec packetizer for video on top of what is required by SPacket

- Easy to define an implement: just split the bytes of each spatial frame in several RTP packets payloads up to the MTU without any codec-specific boundary.
- Initial targets: webrtc video codec formats (h264,vp8, vp9, av1).
- Requires to specify what are the “frame bytes” for each supported codec
  - Codec spec
  - mp4/web media container formats
  - WebCodec
  - Media over QUIC

Wrapup and Next Steps

- Action items
- Next steps

Thank you

Special thanks to:

The Secretariat, WG Participants & ADs