Real-time Audio-Visual Media Transport over QUIC

Colin Perkins – University of Glasgow
Jörg Ott – Technische Universität München | callstats.io
Real-time Audio-Visual Media Transport

- Live or prerecorded streaming
- Interactive video conferencing
- Voice-over-IP
- Augmented and virtual reality

Exabytes per Month

33% CAGR 2017–2022

Video Surveillance (2%, 3%)
Live Internet Video (5%, 17%)
Long-Form Internet VoD (61%, 62%)
Short-Form Internet VoD (32%, 18%)

* Figures (n) refer to 2017, 2022 traffic share
Source: Cisco VNI Global IP Traffic Forecast, 2017–2022
Real-time Audio-Visual Media Transport

Performance limited by TCP

Independently decodable, multi-second, media chunks → basis for rate adaptation, media encoding, playout
Reliable and ordered transport with head-of-line blocking
Enforces lower bound on chunk duration → latency
Compression efficiency limitations

Conference'17, July 2017, Washington, DC, USA

Figure 5: Total stall durations for various chunk durations in a simulated MPEG-DASH application, at encoding rates, $R_{\text{encoding}}$ of 560, 1050, 2350, and 4300 kbps, over a lossless 8 Mbps link with (a) 25 ms and (b) 100 ms RTT
carried out in the application. We do not attempt to model these as part of the analysis, but this does not impact our ability to validate that the broad relationships discussed exist.

For (ii), clearly Internet links are lossy. We update our analytical model in Section 3.3 to include loss, showing that the interactions between TCP and loss result in significant latency overhead. Broadly, this greatly increases the minimum values of $T_{\text{chunk}}$ discussed here.

3.2 Simulations

In the previous section, we described the minimum chunk durations required to maintain smooth playback. We define smooth playback to be the case where playback is continuous: each chunk is available in the buffer at the time it is to be played out. If a given chunk is not in the buffer at its playback time, then the application must stall: that is, playback is paused, and resumes with the delayed chunk upon its arrival.

While we will analyse the impact of stalling behaviour in Section 5, we note that, when TCP is used at the transport, stalling delays are cumulative across the entire session: each chunk’s playout is delayed by the sum of all of the previous stall durations.

Given this definition of smooth playback, our analysis identifies a boundary between high levels of stalling (i.e., where chunks are too small – the red hatched area in our diagrams), and low or zero levels of stalling (i.e., chunks are large enough – the green hatched area). For a given RTT, there are chunk durations that are too small to be sustained without continuous stalling, punctuated by playback of the chunks.

In this section, we describe simulations carried out to validate the analysis presented in the previous section. Our simulator’s server and client operate as described. The server and client use the nghttp2 library for HTTP/2 support, and operate over a standard TCP implementation, with the CUBIC congestion control algorithm. As noted earlier, no rate adaptation is applied and all chunks are encoded at the same rate. We note that while the client decodes the received video, the performance of this process is highly dependent on the hardware on which the simulator runs. Therefore, we are interested in the relative trends, rather than the absolute values, that our simulations show.

Figure 5 shows the total stall durations for various chunk durations (from 3 frames to 30 frames, in 3 frame intervals) at 25 ms and 100 ms RTTs. Total stall duration, as described above, is a measure of the smoothness of playback. These plots validate the analysis presented in Section 3.1: increased media encoding rates, $R_{\text{encoding}}$, (with a fixed bottleneck rate, $R_{\text{bottleneck}}$) or higher RTTs result in more stalling at the same chunk durations. The simulations highlight that the boundaries identified by our analysis are not absolute: stalling durations decrease as chunk durations increase.

TCP

Typically gets sufficient throughput; variability hidden through buffering

Deployable

General purpose transport, weakly coupled to application and media; buffer to hide variability and accept latency penalty to avoid complexity

High latency
Real-time Audio-Visual Media Transport

**mpeg-DASH**
Low cost Commodity HTTP CDNs

**QUIC**
Limits stall duration
Each chunk sent on a separate QUIC stream
Avoids HoL blocking between chunks
Limits stall duration

**Deployable**
General purpose transport, weakly coupled to application and media; buffer to hide variability and accept latency penalty to avoid complexity

**High latency**
Latency remains high
Each chunk still delivered on reliable and ordered stream
QUIC has head-of-line blocking within streams
Enforces lower bound on chunk duration → latency
Compression efficiency limitations
Real-time Audio-Visual Media Transport

Unreliable transport trades low latency for loss

Framing becomes important – no more chunked media
Transport and end-points rearchitected for loss tolerance

No longer HTTP based

Unfamiliar to engineers, no CDN support
Harder to deploy

WebRTC
Designed to support real-time & partial reliability

UDP
Layered above UDP since timeliness preferred over reliability for real-time

Complex
Low latency

Applications aware of loss, framing, errors
Close coupling with media and transport
But – can realise very low latency
suitable for interactive use cases
Real-time Audio-Visual Media Transport

WebRTC

- Designed to support real-time & partial reliability
- Open standard; widely implemented in browsers
- Effective low-latency media transport using RTP
- Lacks CDN and infrastructure support

QUIC

- Used by large HTTP adaptive streaming systems
- Implemented in browsers, servers, and CDNs
- Rapidly being adopted as commodity infrastructure
- Open standard via IETF – extensible

Deployable

Incorporate features from WebRTC into QUIC for deployable low-latency real-time transport
Features of WebRTC

- RTP Media Transport
  - Timing
  - Sequencing
  - Framing and packetisation
  - Partial reliability
  - Congestion control
  - Source identification

- RTP Control Protocol
  - Inter-flow synchronisation
  - Congestion feedback
  - QoS/QoE reporting
  - Source description

- Signalling

Essential for real-time performance
Support quality of user experience
Audio-visual media support
Source identification
RTP Media Transport

- Essential real-time support:
  - Timestamp – reconstruct media timing
  - Sequence number – loss detection and re-sequencing
  - Framing and partial reliability

- Audio-visual media support
  - Payload format specification, PT, M
  - Media mixing
  - Source identification

Broadly applicable transport features → real-time extensions for QUIC
RTP Control Protocol

- Sender reports provide a mapping between the media clock and wall clock – inter-flow synchronisation
  - Generally applicable to real-time flows – not audio/video specific

- Reception quality feedback either equivalent to ACK/ECN reports in QUIC, or media specific QoE

- Source description packets highly application specific
RTP Feature Analysis

<table>
<thead>
<tr>
<th>#</th>
<th>RTP Field and Function</th>
<th>QUIC Mapping</th>
<th>Support?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Seq# for loss detection</td>
<td>Seq# plus ACK mechanism</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>ECN marking</td>
<td>ECN marking</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>Media-specific timestamps</td>
<td>Use metadata on top of QUIC</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>(One-sided) wall-clock sync</td>
<td>Extension or metadata on top of QUIC</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>Data retransmission</td>
<td>Adapt retransmission for partial reliability</td>
<td>(Yes)</td>
</tr>
<tr>
<td>6</td>
<td>Generic FEC</td>
<td>Could be added as a generic function (was discussed)</td>
<td>(No)</td>
</tr>
<tr>
<td>7</td>
<td>Media-specific redundancy</td>
<td>Could be realized as payload</td>
<td>N/A</td>
</tr>
<tr>
<td>8</td>
<td>General RX stats from RTCP RR</td>
<td>ACK blocks for losses (abs. and rel.) to be augmented</td>
<td>Partly</td>
</tr>
<tr>
<td>9</td>
<td>Congestion control</td>
<td>Done by QUIC, may need an API</td>
<td>(Yes)</td>
</tr>
<tr>
<td>10</td>
<td>Selective encryption of payloads</td>
<td>Almost full encryption largely including headers</td>
<td>(Yes)</td>
</tr>
<tr>
<td>11</td>
<td>SSRC and CNAME for media bundling</td>
<td>Bundling implicit within a QUIC connection</td>
<td>Implied</td>
</tr>
<tr>
<td>12</td>
<td>Payload type + M bit marking</td>
<td>Use payload framing on top of QUIC</td>
<td>N/A</td>
</tr>
<tr>
<td>13</td>
<td>Source identification (SSRC/CSRC)</td>
<td>Use metadata on top of QUIC</td>
<td>N/A</td>
</tr>
<tr>
<td>14</td>
<td>SDES session metadata</td>
<td>Use metadata on top of QUIC</td>
<td>N/A</td>
</tr>
<tr>
<td>15</td>
<td>External signalling channel</td>
<td>Use an in-band QUIC stream if feasible</td>
<td>(Yes)</td>
</tr>
<tr>
<td>16</td>
<td>IP Multicast support</td>
<td>Not required</td>
<td>No</td>
</tr>
<tr>
<td>17</td>
<td>Mixers and translators</td>
<td>Implement in the application above QUIC</td>
<td>No</td>
</tr>
<tr>
<td>18</td>
<td>Transmission scheduling control</td>
<td>Done by QUIC, needs an API</td>
<td>(Yes)</td>
</tr>
<tr>
<td>19</td>
<td>Header extensions and metadata</td>
<td>Use metadata on top of QUIC</td>
<td>No</td>
</tr>
<tr>
<td>20</td>
<td>Application level framing</td>
<td>Partial, via QUIC streams</td>
<td>Yes</td>
</tr>
<tr>
<td>21</td>
<td>Avoidance of HoL blocking</td>
<td>Partial, via QUIC streams</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 1: Mapping RTP functions to QUIC

Full feature applicability analysis in the paper – categorising RTP features as real-time support, media support, or application support
Bringing Real-time to QUIC: RT-STREAMs (1)

- Incorporate features of RTP that apply to all real-time applications into QUIC
- Timed, framed, RT-STREAMs
  - New stream type to identify real-time data; carries RT_STREAM frames
  - STREAM_ID, offset, and length specify data range being carried
  - Frame sequence number for identification, loss detection, dependencies
  - Timestamp indicates data presentation time – and deadline for retransmission
  - Partial reliability, handled by sender
  - Frame boundaries are preserved, applications fragment into independently useful ADUs

Possible frame format:

```
+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
| RT_STREAM | | | | | | | |
+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
| STREAM_ID       |
+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
| [Offset]        |
+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
| [Length]        |
+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
| ADU Sequence #  |
+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
| [Lowest ADU sequence # to be retransmitted] |
+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
| Timestamp       |
+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
| Payload         |
+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
```
Bringing Real-time to QUIC: RT-STREAMs (2)

- RT_STREAM timestamps give timings within a stream
- ACK_TIMING frames report packet arrival times
- WALLCLOCK frames map timestamps to common clock – allows inter-stream synchronisation
  - E.g., lip-sync of audio and video flows
  - Sent periodically at low rate

- Minimal extensions to add common real-time support
RT-STREAMs vs. datagrams

• Why provide RT_STREAMs rather than datagrams?
  • Raise level of abstraction for real-time traffic – avoid re-inventing common features
    • Timing, sequencing, loss tolerance
  • Enable future transport optimisations for real-time
    • Differential congestion control
    • Scheduling, deadlines, and partial reliability
• Media codec support is application specific and not be part of QUIC

>75% of Internet traffic is real-time data – design the transport to support it

Source: Cisco Visual Networking Index 2018
• QUIC has extensive support for HTTP-based applications

• Easily extended to support low-latency real-time applications

• We’re moving beyond TCP – let’s also move beyond UDP for real-time

Conclusions

This work received support from the UK Engineering and Physical Sciences Research Council (grant EP/R04144X/1)