# Poster: Cross-layer metrics sharing for QUICker video streaming

Joris Herbots
Hasselt University – tUL – EDM
Diepenbeek, Belgium
joris.herbots@uhasselt.be

Maarten Wijnants
Hasselt University – tUL – Flanders
Make – EDM
Diepenbeek, Belgium
maarten.wijnants@uhasselt.be

Wim Lamotte
Peter Quax*
Hasselt University – tUL – Flanders
Make* - EDM
Diepenbeek, Belgium
{first.last}@uhasselt.be

## ABSTRACT

QUIC is marketed to hold many advantages over TCP. However, preliminary experimentation has shown that simply running contemporary HTTP Adaptive Streaming (HAS) implementations over QUIC does not improve but actually hurts streaming performance compared to a traditional TCP deployment. We argue that this behavior can be attributed to the amount of TCP specialization that HAS Adaptive BitRate (ABR) algorithms have received over the years. In contrast to TCP (which can be regarded as a "black box"), QUIC actually encompasses all the necessary tools to empower streaming performance optimization (e.g., definition of custom congestion control algorithms, access to transport-layer metrics in the application layer). This however comes at the expense of added complexity which in turn could lead to misinterpretations of the root causes of suboptimal streaming performance. To facilitate research on HTTP adaptive bitrate streaming over QUIC, in this paper we propose a solution towards jointly visualizing transport- and application-layer metrics to allow for a better understanding of HAS streaming performance over various types of transport protocols (i.e., TCP versus QUIC). We see the work presented in this paper as an important stepping stone towards cross-layer optimization of HAS ABR performance to achieve a better overall Quality of Experience (QoE) for streaming users.

## CCS CONCEPTS

• **Networks** → **Transport protocols**; **Application layer protocols**.

## KEYWORDS

QUIC, HTTP/3, MPEG-DASH, HLS, Adaptive Streaming, Transport Protocol, QoE

**ACM Reference Format:**
Joris Herbots, Maarten Wijnants, Wim Lamotte, and Peter Quax. 2020. Poster: Cross-layer metrics sharing for QUICker video streaming. In *The 16th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '20), December 1–4, 2020, Barcelona, Spain.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3386367.3431901


*CoNEXT '20, December 1–4, 2020, Barcelona, Spain*

ACM ISBN 978-1-4503-7948-9/20/12…$15.00
https://doi.org/10.1145/3386367.3431901

## 1 INTRODUCTION

Over-the-top (OTT) media streaming forms the bulk of contemporary Internet traffic. It owes its popularity to the widespread availability of HTTP infrastructure which is used by present-day HTTP adaptive bitrate streaming (HAS) techniques such as MPEG-DASH and HLS. Any device with an active Internet connection and support for the TCP/IP stack has inherent support for HAS. TCP, however, is an ossified transport protocol plagued by long-standing issues (e.g., head-of-line blocking, slow connection setup times) which since its inception has been used in varying ways not intended by its original creators in 1974. The, almost finalized, IETF QUIC (henceforth QUIC) transport protocol is designed to solve these issues by combining all the lessons learned from past TCP, SCTCP and HTTP/2 protocols.

With QUIC rapidly gaining interest, many [1, 2, 5] in the field of HAS are looking at the possibilities it brings to the table. Arguably QUIC's most disruptive feature (compared to TCP) is that it runs in userspace. This implies that elements such as flow control and congestion control can easily be customized and tuned to achieve better performance, which in terms of HAS could lead to a better QoE. A second benefit of having QUIC in userspace is the ability to directly use transport-layer metrics in the application layer. One of the biggest problems with HAS nowadays is the dual control loop [2], i.e., an HAS client has to *estimate* transport-layer behavior in order to stay adaptive because of the lack of APIs in TCP. With QUIC, such guesstimates become redundant because we have direct access to these parameters. All QUIC's new possibilities however come with the downside of having added complexity and even though QUIC implementations are quickly maturing, there is a large heterogeneity among them [4]. This makes it challenging to compare HAS performance over QUIC versus TCP (and also among individual QUIC implementations).

At the time of writing, there has not yet been a lot of research into HAS performance over QUIC and the work that exists is prone to miscalibrated QUIC implementations which in turn lead to misleading conclusions [3]. To mitigate this hiatus, we propose charting out the performance of HAS over QUIC. We argue that it is important to visualize the relationship between transport- and application-layer metrics to facilitate root cause analysis of performance issues. The resulting insights will pave the road towards our goal of cross-layer HAS optimization (when deployed over QUIC) to achieve a better overall QoE for streaming users.

## 2 METHODOLOGY

The work by Marx et al. [4] introduces a common logging format for QUIC named `qlog` which enables us to perform rapid analysis and

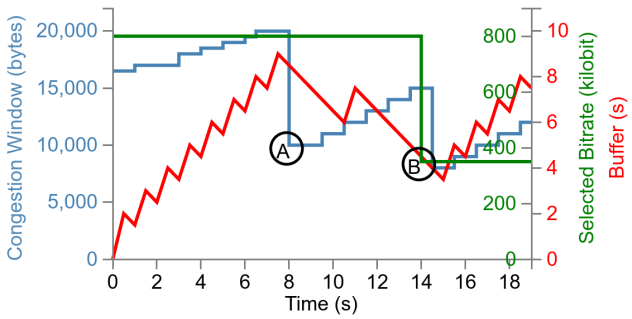


**Figure 1: A hypothetical visualization of an HTTP adaptive bitrate streaming trace. The blue line represents the congestion window size in bytes as set by the transport layer. The red line represents the actual buffer contents of the video-player in seconds. The green line represents the selected bitrate by the client.**

create visualisations. `qlog` has proven to be an indispensable tool in understanding and debugging the QUIC protocol. It is currently deployed by 70% of all QUIC implementations and even used in production environments by large companies such as Facebook.

The `qlog` format is protocol agnostic, it defines logging schemas for QUIC[1] and HTTP/3[2] which we intend to expand by adding our own logging schema for HAS. The intention is to log internal HAS states such as, but not limited to, player events (e.g., play, pause, manual quality changes), playback buffer level, rebuffer occurrences, viewport changes (e.g., landscape vs portrait orientation), decoding performance (e.g., dropped frames), startup delay, network requests (e.g., requests for media segments made to the transport layer) and the ABR estimations for network conditions (e.g., estimated RTT and throughput). One should be able to replay captured HAS client behavior by using aforementioned logs.

For the visualization part, we intend to extend `qvis`[3], a toolsuite which allows for charting QUIC metrics in a variety of graphs and diagrams. Figure 1 shows a hypothetical visualization in which the HAS client encounters congestion, evidenced by a drop in the congestion window (cwnd) size (A). Realistically, some time later, a throughput reduction ensues if we assume the round-trip-time (RTT) stays the same (not plotted in Figure 1). Present TCP-specialized HAS clients do not have knowledge of intricate transport-layer metrics like cwnd size but instead typically only estimate the throughput (e.g., by averaging the download speeds of requested segments). As such, this change in network conditions is only witnessed by the ABR logic long after the transport layer has noticed it (B). Because QUIC runs in userspace, this scenario can be improved by allowing cross-layer information sharing, where the ABR logic is informed of network changes directly by the transport layer such that the gap between (A) and (B) could be shortened. The aforementioned example shows the usefulness of jointly visualizing both layers, as it allows for more tangible root cause analysis and for uncovering protocol deficiencies which might go unnoticed without proper graphical support.

Jointly visualizing transport- and application-layer metrics is just a stepping stone towards our broader goal of optimizing HAS by exploiting cross-layer information sharing. Not only will it allow us to correctly compare TCP versus QUIC configurations, but also to verify that the results we achieve by taking advantage of the new capabilities offered by QUIC are indeed causing performance improvements. The following are some of the optimizations we have in mind: adjusting initial flow control parameters (i.e., flow control limits for QUIC streams) and exploiting 0-RTT to improve startup delays, and a new ABR logic which makes quality decisions based on network information retrieved from the transport layer (i.e., congestion window size changes, flow control limits, RTT measurements).

In order for our changes to have any meaning, it is imperative to gather traces from different deployment scenarios. We will therefore perform tests in lab setups as well as in-the-wild. The results gathered from this process will act as a guide during our efforts of holistically tuning QUIC and ABR performance parameters.

## 3 CURRENT STATUS AND OUTLOOK

At the time of writing, we are working on implementing the aforementioned extensions on top of `qlog` and `qvis` to achieve results similar to the hypothetical scenario in Figure 1. We are using the MPEG-DASH dash.js player as a base for our HAS statistics as these can be extracted from internal state objects. Both Chrome and Firefox provide builds with working QUIC clients which we have already successfully used for adaptive bitrate streaming using MPEG-DASH. In order to create baseline measurements of HTTP adaptive streaming over TCP, we are using extended Berkeley Packet Filter (eBPF) programs to extract the necessary TCP statistics from the kernel such that a comparison between HAS over TCP and QUIC becomes possible.

Short-term future actions include the tuning of HAS for QUIC by incorporating cross-layer information sharing as explained in previous section. In the long term, we are convinced that the proposed logging and visualization framework will be of great importance when exploring more exotic use cases utilizing upcoming QUIC features such as multipath QUIC[4] and unreliable streams[5].

## REFERENCES

[1] Sevket Arisu and Ali C Begen. 2018. Quickly starting media streams using QUIC. In *Proceedings of the 23rd Packet Video Workshop*. 1–6.
[2] Divyashri Bhat, Amr Rizk, and Michael Zink. 2017. Not so QUIC: A performance study of DASH over QUIC. In *Proceedings of the 27th workshop on network and operating systems support for digital audio and video*. 13–18.
[3] Arash Molavi Kakhki, Samuel Jero, David Choffnes, Cristina Nita-Rotaru, and Alan Mislove. 2017. Taking a long look at QUIC: an approach for rigorous evaluation of rapidly evolving transport protocols. In *Proceedings of the 2017 Internet Measurement Conference*. 290–303.
[4] Robin Marx, Joris Herbots, Wim Lamotte, and Peter Quax. 2020. Same Standards, Different Decisions: A Study of QUIC and HTTP/3 Implementation Diversity. In *Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC*. 14–20.
[5] Jan Rüth, Konrad Wolsing, Martin Serror, Klaus Wehrle, and Oliver Hohlfeld. 2019. Blitz-starting QUIC Connections. *arXiv preprint arXiv:1905.03144* (2019).

[1] https://tools.ietf.org/html/draft-marx-qlog-main-schema
[2] https://tools.ietf.org/html/draft-marx-qlog-event-definitions-quic-h3
[3] https://qvis.edm.uhasselt.be
[4] https://tools.ietf.org/html/draft-deconinck-quic-multipath
[5] https://tools.ietf.org/html/draft-ietf-quic-datagram