Analyzing the Influence of Resource Prioritization on HTTP/3 HOL Blocking and Performance

Constantin Sander, Ike Kunze, Klaus Wehrle
Communication and Distributed Systems
RWTH Aachen University, Aachen, Germany
{sander, kunze, wehrle}@comsys.rwth-aachen.de

Abstract*—HTTP/3 comes with a significant change on the transport layer by switching from TCP to QUIC. Of particular interest is that QUIC features independent streams which removes transport HOL blocking during packet loss: loss-unaffected streams no longer have to wait for full retransmissions of affected streams which was the case for HTTP/2. To leverage this new capability, multiple streams have to be in flight at the same time. This can, e.g., be governed by HTTP resource prioritization which allows a browser to signal its desired scheduling of streams to improve web performance. For HTTP/2, sequential scheduling, as used by Chrome, has proven to achieve good results. This choice, however, prevents QUIC's independent streams from taking effect. In contrast, round-robin scheduling could exploit this specific feature best, but it has shown detrimental effects for HTTP/2. Yet, in that case, it could not benefit from independent streams. Whether round-robin is now beneficial with HTTP/3 is unknown, as the interplay of resource prioritization and HOL blocking on performance for HTTP/3 is unexplored. Since the alleviation of HOL blocking is one of the main features of QUIC, we thus analyze its impact and influencing factors. We find that for bursty loss, e.g., from congestion, sequential scheduling achieves good web performance, but that parallelism can help for increasing random loss rates. Nevertheless, for moderate loss, parallelism taking priorities into account is more helpful than agnostic round-robin.

I. Introduction

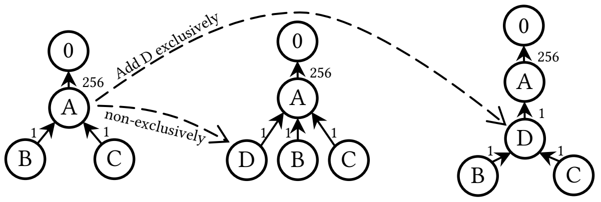
HTTP/3 [1] is the new major version of HTTP. It comes with few feature changes compared to HTTP/2 [2], but introduces a significant shift on the transport layer: from TCP to QUIC [3]. In particular, QUIC features a new stream notion on the transport layer by which streams are independent from each other with respect to loss. While streams do also exist in HTTP/2, they share a single, stream-unaware TCP connection running underneath. In case of packet loss, this overall connection, and with it even resource streams unaffected by loss, stall, and the website load process is unnecessarily interrupted. This phenomenon is widely known as head-of-line blocking (HOL blocking). With QUIC, streams unaffected by loss are no longer hindered by inter-stream HOL blocking and can instead continue sending.

However, for this new capability to take effect, several streams and resources must be multiplexed and in-flight in parallel, which can be governed through HTTP *resource prioritization*. In short, it allows browsers to signal how and in which order website resources should be sent by the servers with the goal of improving a website's loading process. For instance, rendering-important files, such as stylesheets or above-the-fold images, can be prioritized. Today, there exist multiple HTTP/2 prioritization strategies. The engine of Firefox, e.g., relies on a complex dependency tree where resources are sent in parallel with differing weights. In contrast, Chrome uses a sequential approach while using round-robin is also possible. Overall, research has shown that there is no one-size-fits-all solution on how to best schedule resources for HTTP/2 [4], [5] and that flexibility is required.

With HTTP/3, resource prioritization uses a new signaling mechanism: the Extensible Prioritization Scheme (EPS). It is less expressive and no longer represents elaborate dependencies between resources, e.g., making it impossible to directly map Firefox's HTTP/2 strategy to HTTP/3. On the other hand, while representable using EPS, Chrome's sequential strategy stands in contrast to having multiple in-flight resources to exploit QUIC's independent streams. Consequently, none of these established prioritization strategies seem to be capable of leveraging the full potential of QUIC. Instead, switching to round-robin may be superior, although it was shown to perform worst for HTTP/2 [4]. However, while previously subject to HTTP/2's HOL blocking, it might now take advantage of HTTP/3's parallel QUIC streams to avoid unnecessary waiting time during retransmissions. Related work has also already raised this assumption [5], [6], but there has not yet been a systematic study on the various influencing factors.

In this work, we thus set out to explore the previously unknown impact of resource prioritization and HOL blocking on the performance of HTTP/3. Using a controlled testbed study, we compare HTTP/3's web performance when subject to various link characteristics, such as higher latencies and loss, prioritization strategies, and websites, to argue about HOL blocking and when it impacts performance. We further involve HTTP/2 strategies that cannot be signaled via the EPS and can thus assess whether the simplification of the EPS slows down HTTP/3. Specifically, our paper contributes the following:

- We present an exchangeable prioritization testbed for repeatable tests of HTTP/3 and HTTP/2 strategies via HTTP/3.
- With our testbed, we find that parallelism helps for random loss, while this correlation was not found for HTTP/2.
- Yet, for higher bandwidths, lower RTTs, or burst loss, parallelism is detrimental, which has to be taken into account.
- We observe that an EPS-adapted strategy cannot compete



Stream	Urgency	Incremental
A	1	0
B	2	1
C	2	1

Stream	Urgency	Incremental
A	1	0
B	2	1
C	2	1
D	2	1

Stream	Urgency	Incremental
A	1	0
B	2	1
C	2	1
D	2	0

Fig. 1. HTTP/2 Priority Tree and same scheduling as Extensible Prioritization Scheme below. A new node D can be added exclusively (right) or non-exclusively (center) to the priority tree.

with the HTTP/2 original, but the drawbacks are limited.

Structure. Sec. II discusses the evolution of HTTP and HTTP prioritization, while Sec. III discusses related work concerning its performance. Sec. IV describes our testbed study to analyze the prioritization effect for a set of selected websites. Sec. V describes our results, drilling down into the scenarios and effects found. Finally, Sec. VI concludes this paper.

II. Background: HTTP/2, HTTP/3 and Prioritization

HTTP/2 [2] significantly changed HTTP by introducing stream semantics to multiplex several resource streams over a single TCP connection. Compared to the previous HTTP practice of having multiple TCP connections for parallel transfers, it lowers the overhead on the involved hosts as fewer connections have to be maintained and reduces latency by saving handshakes and congestion control slow start [2], [7]. However, building upon TCP, which has no notion of streams, HTTP/2 suffers from inter-stream HOL blocking in case of packet loss as TCP stalls the whole connection and thus all HTTP/2 streams even if only one stream is affected.

QUIC and HTTP/3. This aspect and further issues of HTTP/2 over TCP led Google to develop QUIC [8] (in the following named *gQUIC*) as a new special-purpose transport protocol for HTTP. gQUIC brought major changes, among other things, independent streams on the transport layer such that inter-stream HOL blocking was rectified. Today's standardized QUIC [3] has evolved from a protocol co-designed for HTTP into a general-purpose transport protocol seeing increasing use on the Internet for HTTP [9]. However, the mapping to HTTP semantics is no longer part of QUIC and is instead developed individually in the form of HTTP/3 [1]. It inherits most features of HTTP/2, but some of them required redesigned implementations as they relied on TCP's inherent orderedness. One of these features is resource prioritization.

A. Resource Prioritization

The core idea of resource prioritization is for browsers to explicitly tell web servers which resources they should send first to optimize user experience. For example, they could first focus on *above-the-fold* resources shown first in the viewport of the user. The importance of resources heavily depends on the browser, the webpage, and its resources [10], although research has generally found loading HTML, Javascript (JS), and stylesheets (CSS) first to be beneficial [4], [5], [11] as these critically define the general behavior and layout.

HTTP/2 Prioritization. HTTP/2 [2] defines resource prioritization using a *weighted priority tree* as illustrated on the top-left side of Fig. 1. Starting with the root stream 0, streams transmit data in downwards order where children wait for their parents. Siblings send in parallel according to their weight. In our example, B and C send round-robin after A.

A browser signals the underlying parent-child relation of a stream by sending its weight, its parent and an exclusive flag. With a set exclusive flag (top-right side of Fig. 1), a new stream D will become the parent of all existing children of its own parent and will be sent first, while an unset flag will make it share the bandwidth with the other children (Fig. 1, center).

HTTP/3 Prioritization. QUIC's streams no longer guarantee the required order of the prioritization signals such that the resulting tree can diverge. Further, it was found that web servers often did not follow prioritization correctly, possibly due to the tree's complexity, rendering it ineffective [12], [13]. Prioritization was thus removed from HTTP/3 [5].

Instead, work focused on designing the *Extensible Prioritization Scheme (EPS)* [14] extension, which is illustrated in the bottom row of Fig. 1. Every stream is assigned an urgency and incremental flag, and streams are then sent in increasing order of urgency. If streams have the same urgency, they are either sent in order of appearance, if the incremental flag is not set, or in parallel using round-robin. In contrast to the HTTP/2 tree, the EPS is order-independent as it avoids direct relations such that it is less complex but also less expressive. For example, weighted round-robin cannot be represented as no weights exist.

Resource Prioritization and HOL blocking. As described above, QUIC's support for independent streams has the potential to ameliorate HTTP/2's inter-stream HOL blocking, e.g., during heavy packet loss. For this, multiple streams have to be in-flight in parallel, as can be governed by resource prioritization. However, not every browser uses a suitable strategy as, e.g., Chrome requests resources sequentially. Aggravatingly, related work has found supporting strategies, such as round-robin, to be detrimental to HTTP/2 web performance [4]. Thus, we now take a closer look at related work and the impact of prioritization and QUIC on performance.

III. Related Work: Performance Impact of Prioritization and QUIC

The performance impact of QUIC and web resource prioritization is a frequent subject of study. One branch of research deals with alternatives to HTTP prioritization, such as server push, resource hints, or custom Javascript-based schedulers to prioritize and decide when to transfer resources via HTTP/2 or SPDY [11], [15]–[20]. These works show that carefully structuring resource transfers can significantly improve the webpage loading process but do not discuss specific HTTP/2 prioritization strategies, with respect to sequential or round-robin loading, and rarely discuss the impact of packet loss.

A. *Prioritization over HTTP/2 / TCP*

Work on HTTP/2 resource prioritization mostly confirms the potential for prioritizing web resources. For instance, De Saxcé et al. [21] and Bergan [22] find that HTTP/2 prioritization improves the web performance for websites by more than 5 % in terms of SpeedIndex (SI) [23] and Page Load Time (PLT).

Wijnants et al. [4] provide a comprehensive analysis of the impact of HTTP/2 prioritization strategies on browsers. They observe that browsers either use i) sequential transfers (Chrome), ii) (weighted) round-robin (Safari, Internet Explorer), or iii) combinations of both (Firefox) when transferring web resources. In a controlled testbed study, similar to our work, the authors find that a simple round-robin scheme achieves the worst results, while Firefox's and Chrome's default strategies as well as a FIFO strategy, perform well. However, this effect becomes less significant for higher packet loss rates. Given that QUIC was relatively immature at the time of writing, the authors did not evaluate it.

B. *Prioritization over HTTP/3*

Marx et al. [5] study prioritization schemes for HTTP/3. However, due to missing stack and browser support at the time of writing, they focus on a theoretical, lossless scenario. They replay the download of a website and compute the ByteIndex [24], an approximation of the SI, for above-the-fold resources. Similar to HTTP/2, the authors find that round-robin is the worst strategy. In contrast, a plain sequential loading, as well as simple, website-agnostic sequential strategies, can achieve improvements from 93 % up to 130 %. However, the authors do not test the new HTTP/3 EPS and acknowledge that further work is needed to identify the influence and interplay of prioritization in real-world loss scenarios. They particularly stress that round-robin is worst with respect to web performance for lossless scenarios but potentially best to exploit QUIC's independent, HOL blocking-free streams when subject to loss.

C. *QUIC and Packet Loss*

Other related works omit prioritization and study QUIC's general web performance when subject to loss. Most works observe that gQUIC tested in testbeds [25]–[28] or involving the Internet [8], [28]–[30] improves performance with respect to HTTP/2, especially when subject to high loss. Few works find worse performance [31], [32] attributed to a flaky gQUIC stack [33].

Similarly, HTTP/3 is mostly found to improve performance in testbeds [34] and when using production endpoints [6], [35], [36]. Trevisan et al. [36], e.g., find an improved SI for Chrome when using HTTP/3 in the Internet, although the HTTP/3 performance surprisingly equalizes to HTTP/2's SI when subject to packet loss. Analogously, Yu et al. [6] find that HTTP/3 production endpoints achieve overall improved web performance for small websites compared to HTTP/2. However, the performance improvements of QUIC are seldomly visible in lossy settings, and endpoints using round-robin did not improve web performance in those settings either. These findings raise the question of whether QUIC's HOL blocking improvements are indeed effective.

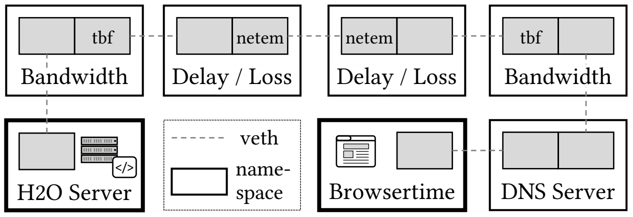


Fig. 2. Namespace Testbed Setup: We move every component of our network emulation in its own network namespace, specifically the server and our browser, to then connect these namespaces with virtual connections (veth) and, finally, apply shaping (tbf) or delay/loss (netem).

D. *Takeaway and Research Gap*

Overall, there is substantial work showing the potential of resource prioritization and highlighting general improvements of QUIC. However, the impact of resource prioritization to leverage QUIC's HOL blocking improvements and the resulting impact on web performance have thus far not been studied. Whether the EPS might interfere is unknown.

We thus recognize a need to shed first light on these interdependencies and devise a controlled testbed to study performance differences. In particular, we adapt the scheduling on server-side and alter link characteristics to evaluate whether HTTP/3's prioritization performance differs due to the differences in HOL blocking, as presented in the next section.

IV. Methodology

The fundamental approach of our study is similar to MahiMahi [37]: we i) download the sources of a diverse set of websites and then ii) replay the websites by rehosting them locally and accessing these local instances with a browser. However, we do not directly use MahiMahi due to multiple reasons. For example, MahiMahi's *webrecord* tool showed problems when downloading websites using HTTP/2. Instead, we use *mitmproxy* and Chromium (with disabled QUIC) as proposed in [11], [25]. A custom framework further allows us to have more fine-grained control over our testbed and increases flexibility to, e.g., use *netem* with burst loss.

In the following, we first describe critical components of our testbed, including how we replay the websites and model the different prioritization strategies before we detail the different parameters that we study.

A. *Flexible Testbed Setup*

We illustrate our testbed setup in Fig. 2: a browser (cf. Sec. IV-A2) fetches a website from an adapted H2O-based web server (cf. Sec. IV-A1), replaying selected websites[1].

[1]Code available at https://github.com/COMSYS/HTTP-3-Prioritization

Both entities are deployed in dedicated network namespaces and interconnected by several additional network namespaces, which allow us to model different network settings using Linux's *tc* tool. We connect the namespaces using veth pairs and deploy *token bucket filters* (*tbfs*) with a burst size of one MTU for bandwidth shaping and *netem* to apply delay and loss on the veth devices. To avoid i) interference between the tbf and netem and ii) backpressure on the sender-side network stacks [38], we dedicate one namespace for each function. Our H2O server uses a dedicated IP that is routed to the browser, and we deploy a DNS server to provide URL translation.

1) Web Server with Adaptable Priorities: Our hosting front-end server is an adapted H2O web server, which we change to follow our inputs for the EPS and where we also integrate its HTTP/2 scheduling scheme implementation into HTTP/3. We specifically use H2O due to its proven implementation shown in research with respect to HTTP/2 prioritization/push [4], [11] and its interoperable quicly QUIC stack [5], [39].

For resource and prioritization retrieval, we identify resources by their URL, following MahiMahi's URL mapping approach [37]. Moreover, we also compare the incoming headers for cache hashes / ETags to replay the corresponding resource as precisely as possible. Further, we stitch together chunks, which caused problems for HTTP/2, and remove headers influencing connections (e.g., "connection", "alt-svc", "link") or expiring resources (e.g., "expires", "date") similar to [11].

Gathering Prioritization Strategies. By default, our browser will only signal its own prioritization strategy, which cannot be leveraged to model all strategies that we want to test. We thus have to precompute the desired strategies to allow a correct prioritization via HTTP/3. For this, we access the websites in our testbed using Firefox and Chromium via HTTP/2 and record the priority weights, and dependencies sent per resource. Moreover, we record the types of Chromium's requests to then post-process the signals into classes as defined in [4] and explained in Sec. IV-B1 to compute an offline hint for every resource. To ensure stable results, we repeat this process 30 times for every website and fuse the classes and weights for every resource by using the corresponding maximum. We then create the corresponding HTTP/2 trees and EPS signals in H2O at runtime, guided by the hints.

2) Webbrowser Measurements with Browsertime: Finally, to measure the actual performance, we fetch websites using a Browsertime docker container[2] with Chromium 95.0.4638.54. Browsertime screen captures the browser during loading a webpage and processes the video to calculate different web metrics, such as the SI. We specifically use the traditional, image-based SI calculation as the resource timing-based real user monitoring (RUM) SI is only an approximation thereof [40]. Note that we host all website resources (also third-party resources) on the same IP and include all domains in the corresponding certificate to enforce connection reuse. We further disable the Fetch credentials flag in Chromium to avoid new connections due to differing connection pools for cross-origin requests [41] and configure Chromium to only accept our custom certificates. This way, we enforce a single connection[3] on which prioritization can govern the scheduling of data without interference from other parallel connections. Thus, while being admittedly unrealistic, we create an ideal scenario to study prioritization.

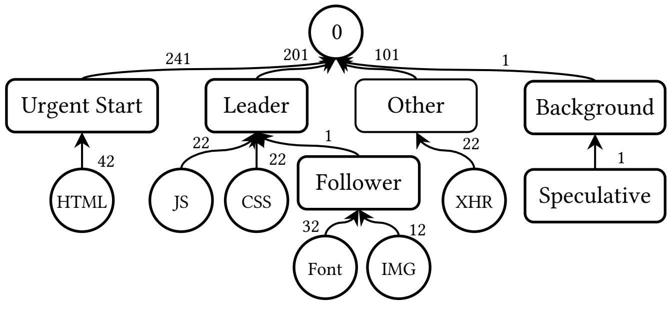


Fig. 3. Firefox HTTP/2 Priority Tree (adapted from [4], [42], updated weights)

3) Physical Measurement Environment: We run our measurements on physical desktop PCs to represent the page load process on a typical consumer-grade machine. The PCs contain an Intel i5-4590 CPU and 16GB of RAM and run Ubuntu Server 18.04.6 with Linux 5.3.0-24. We parallelize our measurements on seven equally-built PCs where each machine only runs one measurement at a time. We further stop and restart all namespaces, running servers, and the browser and use a new profile with an empty cache for every measurement.

B. Measured Parameters

To achieve meaningful results, we investigate different prioritization strategies (Sec. IV-B1), websites of different sizes (Sec. IV-B2), and network scenarios (Sec. IV-B3).

1) Prioritization Strategies: There exist many different strategies and recommendations [5] for prioritization. In our study, we focus on the following strategies based on the findings of Wijnants et al. [4]. In particular, we use Chrome's sequential ordering, weighted round-robin as found in Safari, resource-agnostic round-robin, and Firefox's dependency tree.

Chrome: Chrome uses a sequential ordering with seven priority levels. Any lower level may only send if all higher priority levels have been processed entirely and resources within each level are transmitted in a FIFO fashion. In HTTP/2, this approach is represented as a single dependency chain, while for HTTP/3, the priority levels can be mapped to corresponding urgencies in the EPS with a disabled incremental flag.

Round-Robin (RR): RR is the inverse of Chrome's approach as each stream can alternately send one frame. It easily maps to HTTP/2's tree and the EPS as it ignores weights.

Weighted Round-Robin (WRR): Wijnants et al. [4] find that Safari uses a WRR strategy for prioritizing resources. All resources are added non-exclusively to the root of the

[2]https://github.com/sitespeedio/browsertime

[3]Chromium might still open parallel connections first due to race conditions but closes these redundant connections early on

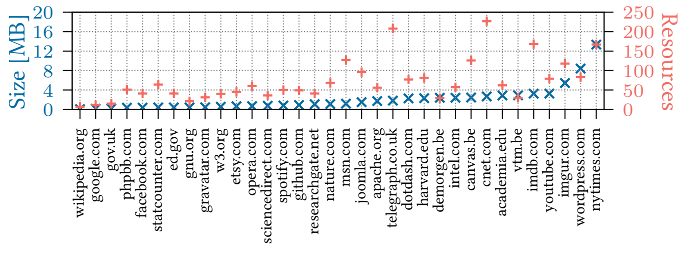


Fig. 4. Size of Downloaded Resources and Resource Count per Website

Scenario (Section)	BW [Mbps]	RTT [ms]	Added Loss
Bandwidth (§V-A)	1, 2, 5, 10	100	0 %
Random Loss (§V-B)	2	100	{0, 1, 2, 5}%, random
RTT (§V-C)	2	10, 50, 100	2 %
Burst Loss (§V-D)	2	100	2 %, burst: {5, 10, 15}

TABLE I
PARAMETERS VARIED DURING OUR MEASUREMENTS

dependency tree as with RR, but their weights differ with respect to to their importance. For example, HTML has a weight of 255, while CSS and Javascript have the next highest weight of 24. This approach does not map to the EPS as it cannot represent individual weights. To still evaluate the influence of such individual weights with HTTP/3, we patched H2O's HTTP/3 implementation to reuse its priority tree code (c.f. Sec. IV-A1) and use it to govern WRR scheduling.

Firefox: Firefox establishes a complex dependency tree that does not map to the EPS (cf. Fig. 3). Resources are divided into four groups which then share the bandwidth unequally. Within these groups, resources further differ in their weights, and some get additional subgroups. Importantly, the individual weights differ slightly depending on Firefox's internal prioritization, so we extract Firefox's weights from real traces to represent them in our patched H2O server correctly.

Firefox (EPS): As a final strategy, we adapt Firefox's dependency tree to fit into the new EPS to assess whether its simplification has negative implications. We adapt it such that we maintain the round-robin scheduling per group but map the groups and subgroups to different urgencies: *urgent start* (urgency 1), *leader* (2), *follower* (3), *other* (4), *background* (5), and *speculative* (6). [4]

2) Websites: We use 34 of the 40 websites selected by Wijnants et al. [4] (with prepended www.) as the starting point for our study. This set includes a wide variety of popular websites with different numbers of resources and of different sizes, as shown in Fig. 4. We freshly downloaded the websites in February 2022 to also get header information which is missing in the original dataset. However, we do not use all 40 websites of the set. First, we exclude the *internal* website of the dataset and bitly.com, because we could not extract the resource types for the latter site. Second, we exclude sciencemag.org, columbia.edu, reddit.com, and pinterest.com, which, for certain scenarios, often exceeded browsertime's default timeout of 5-minute recordings per website. Thus, all of the following results base on the same set of websites.

3) Scenarios: We generally set up H2O and its quicly stack to use CUBIC as congestion control, which is implemented in userspace and follows RFC8312 [43] without hystart.

We then configure different network settings as shown in Tab. I. First, we analyze the influence of bandwidths, followed by the influence of random loss, and, third, the influence of RTTs. While these values are synthetic to argue on the direct impact of specific changes on HOL blocking per parameter, we tried to orient on challenged mobile networks. Moreover, we also distinguish between random loss and burst loss, creating the latter using the simple Gilbert-Eliott model [44]. Moreover, we prepare the tbf queues to always fit 1 BDP of data.

4) Metrics: Our evaluation uses two metrics discussed throughout this paper: web performance and HOL blocking. We use the SI [23] to argue about web performance, opposed to the PLT, as it is known to closely correlate with the user-perceived performance [11], [24], [33], [45]. A lower SI represents a faster page load and vice versa.

To assess HOL blocking, we measure the amount of intra-stream HOL blocked bytes (we name it HOL) on the browser-side, similarly as demonstrated by [5]. A larger HOL means that more bytes were stalled, so in theory, less data could be processed in parallel. Please note, however, that HOL also depends on the number of in-flight bytes / the congestion window (cwnd). As the in-flight bytes represent the number of unacknowledged bytes which are currently sent, they describe the maximum HOL which can occur when all unacknowledged bytes are blocked. Moreover, the cwnd limits the maximum of in-flight bytes and grows over time / shrinks with loss depending on the scenario. I.e., the cwnd and thus the scenario also influence the HOL. Thus, the HOL could be normalized by all bytes sent whenever streams are HOL blocked. However, missing frames might not be retransmitted directly (e.g., due to prioritization as for H2O/quicly [5]), which can then also skew the results arbitrarily. Thus, we do not normalize it but assume similar cwnds and in-flight bytes for *equal* link scenarios (resulting in equal loss) when comparing the impact of the strategies. To measure the HOL, we process the Netlog of Chromium to extract its QUIC session information. Whenever a stream frame is received, we check whether its offset and length are in order or whether out-of-order frames for a stream are received. In the latter case, we save how many unseen bytes cannot be forwarded due to intra-stream HOL blocking up until the missing frames are received. We then sum up the HOL for all streams.

V. RESULTS

In the following, we present our results for the different scenarios. We perform 30 measurements in each setting, for which we then compute our described metrics. To focus on the differences between the sequential scheduling for Chrome and our tested strategies, we use the relative difference (denoted as

[4]Note, that this approach differs from Firefox's current implementation added in November 2021, which maps differently and does not use RR.

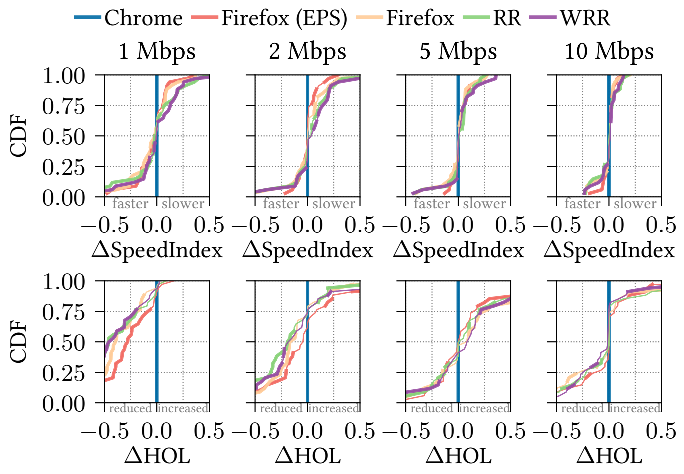


Fig. 5. Impact of Bandwidth on SI and HOL for 100 ms RTT and 0% Loss

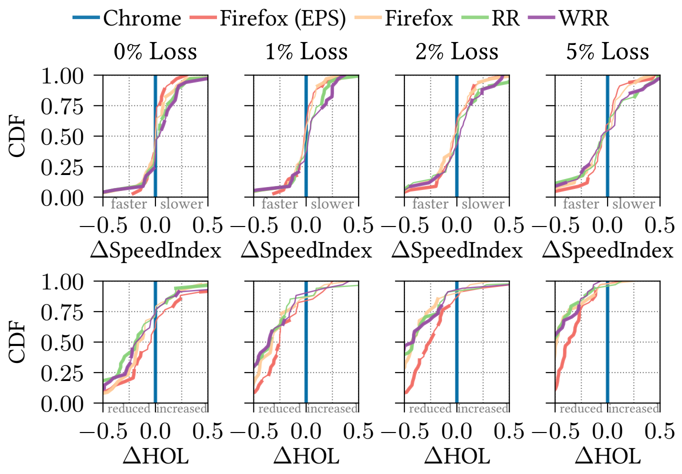


Fig. 6. Impact of Loss on SI and HOL for 2 Mbps Bandwidth and 100 ms RTT

Δ) of their medians. A higher Δ corresponds to an increase with respect to Chrome, e.g., a larger SI, and vice versa. To avoid divisions by zero, we add an $\varepsilon = 1^{-10}$. We further check for statistical significance by adopting [4]'s measurements and present the corresponding results of a Mann-Whitney U test (as our data is not strongly normally distributed) for $p < 0.005$ counted as significant. We mark this significance in our CDF Plots by reducing the line width for insignificant data points.

A. *Influence of Bandwidth*

We first focus on the influence of bandwidth without artificial loss at 100 ms RTT on the prioritization strategies. Fig. 5 shows the relative difference of the SI for the different strategies in comparison to Chrome's sequential strategy (ΔSI) and the relative difference in head-of-line blocked bytes (ΔHOL). We further also tested 50 Mbps (not shown for space reasons).

HOL blocking. We can see that the reduction of HOL blocked bytes vanishes with growing bandwidth and that sequential scheduling can even cause less HOL blocking in many cases as the curve moves to the right, contradicting our initial assumptions. We attribute this effect to larger bandwidths causing larger cwnds and, thus, more in-flight bytes. As the resources transmitted are limited in size, this constellation increases the probability of multiple resources being in-flight – also with sequential sending. The amount of HOL blocked bytes then splits between the resources. Further, as we do not shape any artificial loss, any packet loss occurring in this setting is caused by filled-up queues and, hence, bursty. For example, for a bandwidth of 10 Mbps, we could see burst loss involving up to 20 packets in several Chromium Netlogs and qvis [46]. Such a burst can affect up to 20 RR streams, while for sequential scheduling, only few streams are affected depending on the stream size, which is highly dependent on the website's resource sizes. Thus, depending on the cwnd and stream size, other streams might already follow after that, which can significantly reduce the amount of HOL blocked bytes. We analyze the specific effects of burst loss in more detail in V-D. Further, with higher bandwidths, the link is often not fully saturated (especially for smaller websites) such that no loss / HOL blocking occurred (that is the case for the vertical line from 45% to 75% for ΔHOL at 10 Mbps and for more than 80% of runs at 50 Mbps).

SpeedIndex. The impact of the different strategies on the SI gets smaller and narrower with growing bandwidth. Apart from reduced HOL blocking, we attribute this to increasing bandwidths shifting bottlenecks from *transferring* to *processing* resources. Going from right to left, the parallelized strategies first introduce detriments at 5 Mbps. E.g., for Firefox, 6 websites achieve an SI improvement of more than 5 %, while 9 websites achieve an SI detriment of more than 5 %.

For 2 Mbps, these numbers even out for Firefox and Firefox (EPS), while RR and WRR still suffer. More than one-third of all websites experience a detriment of more than 5 %, while less than one-fifth achieve an improvement in the same order.

For 1 Mbps, WRR and RR catch up, and Firefox and Firefox (EPS) even achieve more significant improvements than detriments by more than 5 %. Given that HOL blocking is reduced for all strategies, its impact is not clearly seen for all SIs.

Takeaway. *Higher bandwidths reduce the effect of resource prioritization strategies with respect to HOL blocking and SI. Counter-intuitively, sequential scheduling can even be beneficial for reducing HOL blocking at higher bandwidths. Lower bandwidths increase the variance of the strategies, i.e., while with high bandwidths the SI changes only marginally, it now improves for certain websites but also decreases significantly. A clear result on which prioritization is in general better for all websites cannot be given, highlighting the need for website-specific approaches at low-bandwidths. A strong influence of HOL blocking on SI cannot be seen in this scenario.*

B. *Influence of Random Loss*

We next analyze the influence of random loss at a rate of 2 Mbps as we could see the largest differences between the strategies at this rate. Fig. 6 shows the results for loss rates of 0 % (for comparison), 1 %, 2 % and 5 %. When looking into

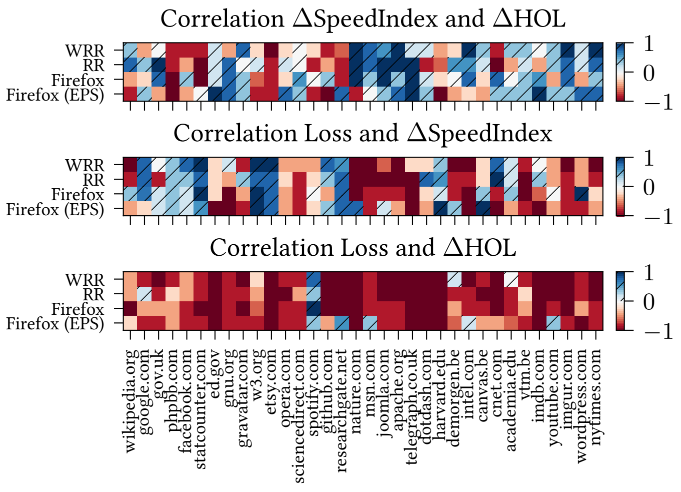


Fig. 7. Spearman's Correlation Coefficient per Website

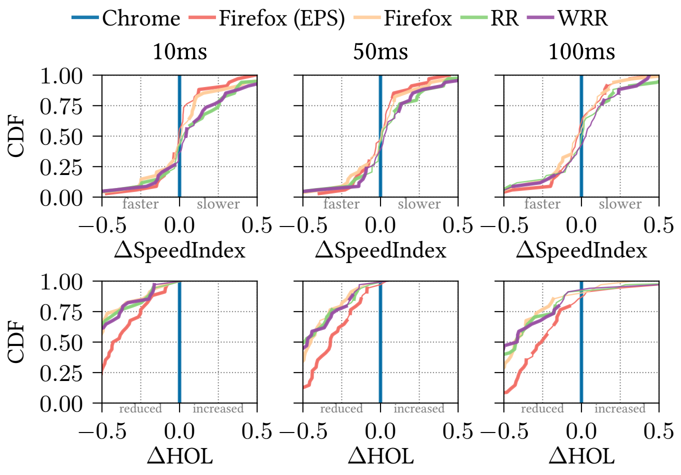


Fig. 8. Impact of RTT on SI and HOL for 2 Mbps Bandwidth and 2% Loss

our prior measurements, we could see similar congestion/burst loss rates from 2% up to 7% in the median. We also repeated this measurement for Reno as congestion control (not shown) and saw the same trend but weaker.

HOL blocking. When increasing the loss rates, we can see that the parallel strategies noticeably reduce the amount of HOL blocked bytes compared to Chrome which we and related work [5] assumed. We deduce this to i) smaller cwnds which, in turn, potentially reduce the number of in-flight streams with sequential scheduling, and ii) single packet random loss affecting sequential scheduling stronger than parallel scheduling. Yet, Firefox (EPS) reduces HOL blocking less than the other strategies. We attribute this difference to our adaptation, which reduces the number of parallel in-flight streams and, thus, potential HOL blocking improvements.

SpeedIndex. The results with respect to the SI are less pronounced. We can see slight improvements in the median for higher loss rates. Firefox achieves the most remarkable improvements, especially at 2 % loss and more. It improves the performance by more than 5 % for 6 more websites than it decreases performance by 5 %. At 5% loss and for 25 % performance change, this difference is 2 websites. Firefox (EPS) instead shows fewer improvements in SI. From 2 % loss on, it achieves its most improvements by 5 % for 2 more websites than it decreases performance. WRR and RR, as before, cannot compete until 5% loss. Generally speaking, we can see that the distribution of benefits and detriments turns at 2 % loss, where the results in total are most balanced.

Correlation. When inspecting individual results, we noticed that larger websites saw a larger influence of loss on HOL blocking and larger improvements for SI. For smaller websites (especially <0.6 MB), we saw less reduction for HOL blocking and mainly SI detriments. Hence, we cannot find a general correlation and show Spearman's correlation coefficient between ΔHOL, ΔSI, and packet loss per website in Fig. 7. The x-axis is sorted with respect to the total size of resources (cf. Fig 4).

Inspecting the correlation between ΔSI and ΔHOL, we can see that it is more often positive for larger websites (right). In comparison, it gets weaker or negative for smaller websites (left). The correlation between ΔSI and loss is more negative for the larger websites, i.e., more loss results in an improved SI compared to Chrome's sequential scheduling. We find the inverse, i.e., a more positive correlation, for smaller websites, i.e., most smaller websites suffer from the parallel scheduling. For loss and ΔHOL, we see strong negative correlations, as could be noticed before, except for spotify.com.

Takeaway. *For random loss, parallel strategies, in general, reduce HOL blocking. However, this reduction does not directly improve the SI, e.g., RR requires high loss to take effect. Also, mainly larger websites see benefits, while smaller websites can even see detriments. Yet, both sides see outliers. As such, packet loss impacts the choice of prioritization, but priorities and website structure are still part of the equation.*

C. Influence of RTT

In this section, we analyze the impact of different RTTs as they have manifold potential influences on performance, e.g., via i) the delay of retransmissions or ii) their impact on the amount of in-flight bytes and in-flight stream frames. We expect that a lower RTT reduces the performance impact of HOL blocking as it reduces wait time. Moreover, the cwnd should be smaller, such that fewer streams are active with sequential scheduling and differences in HOL could grow.

Fig. 8 shows the SI for 10 ms, 50 ms, and 100 ms as RTT when subject to 2 % random loss and 2 Mbps as bandwidth bottleneck. We select 2 % loss, as we could see the most balanced outcome (cf. Section V-B) such that we can see potential changes quickly. We can see that the lower RTTs move our curve downwards, such that more statistically significant detriments and fewer improvements occur. Around 50 % of websites experience a deteriorating performance for RR at 10 ms. HOL blocking-wise, we can see that it is more reduced with parallel strategies for lower RTTs fulfilling our expectations regarding reduced in-flight bytes.

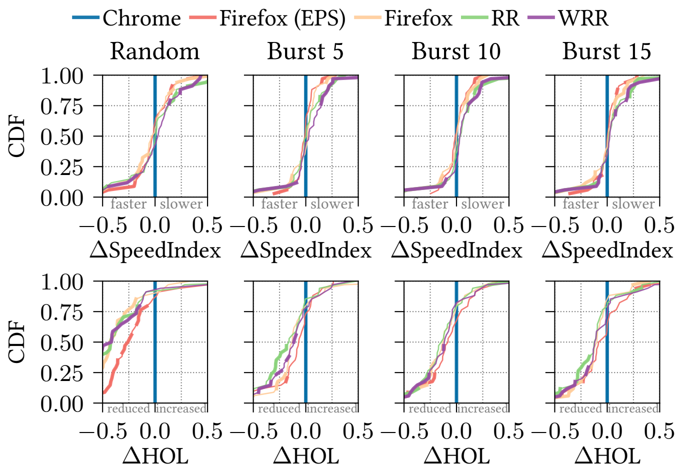


Fig. 9. Impact of Burst Loss on SI and HOL for 2 Mbps Bandwidth and 100 ms RTT for Random Loss and Burst Sizes from 5 to 15 Packets

Takeaway. *Lower RTTs increase the effect of parallel strategies on HOL blocking. However, the effect of HOL blocking on performance vanishes such that the parallel strategies again show a detrimental impact.*

D. Influence of Burst Loss

Until now, we only systematically analyzed the effect of random loss, although we already saw that burst loss changes the behavior in Sec. V-A. Thus, we now measure the influence of burst loss when applying it in a controlled manner.

Fig. 9 shows ΔSI and ΔHOL for an average loss rate of 2 %, but for random loss and burst loss of average length 5, 10, and 15 modeled via the simple Gilbert-Eliott Model [44]. Using this model, very long bursts of loss can occur such that many of our measurements failed as Chromium wrongly detected a network blackhole, and we had to repeat these runs.

We can see that increased bursts of loss while maintaining a steady rate of 2 % artificial loss have a detrimental effect on performance. Also HOL blocking is reduced less strongly. The HOL blocking curve moves right, while the SI curve flattens for improvements. SI detriments stay similar with a widening gap between the Firefox strategies and WRR/RR, where WRR/RR achieves worse performance. We attribute this effect to RR affecting several streams during burst loss, while Firefox distributes stream weight much more diversely, decreasing the probability of loss touching all streams. However, we expected that WRR benefits similarly and that Firefox (EPS) is affected instead, but WRR is less diversely distributed than Firefox. Firefox (EPS) is indeed affected with respect to HOL blocking, which, however, seems to impact performance barely.

A possible alternative might be batched round-robin, which does not alternate after every, but batches of n packets, as suggested by [47] and found for Google's QUIC endpoints [5], [6]. Burst loss thus affects and blocks fewer streams. However, to fully leverage this, an adaption to cwnd and burst size may be required to avoid sequential sending or endangering HOL blocking on many streams, which cannot be signaled today.

Takeaway. *Burst loss is detrimental for per-packet round-robin with respect to HOL blocking and performance, i.e., HOL blocking is reduced less strongly and SI increases outweigh decreases. A more diverse distribution of weights helps, while batched round-robin could also be an alternative.*

VI. Conclusion

HTTP/3 transfers can only benefit from QUIC's HOL blocking free streams during loss if multiple streams are scheduled to be in-flight. HTTP prioritization can govern this scheduling but also impacts web performance when resources are received too late. The newly arising interdependencies between HOL blocking, loss, and performance were unclear.

Hence, we performed systematic measurements to assess the effect of prioritization on HOL blocking and its effect on web performance. We find that parallel strategies are helpful with respect to HOL blocking for high random loss and low BDP scenarios, such that enough streams are active to bridge intra-stream HOL blocking. However, for burst loss, per-packet round-robin strategies suffered contrariwise as now multiple streams were affected, and HOL blocking was not reduced. Further, reduced HOL blocking does not automatically mean better web performance. E.g., smaller RTTs at constant bandwidths mean lower BDPs and cwnds emphasizing sequential issues with respect to loss, but relieve the delay penalty of retransmissions. Yet, a positive effect of reduced HOL blocking on web performance could be seen for higher RTTs and lower bandwidth. However, mostly larger websites benefited, potentially as they have more resources in-flight which could be multiplexed to reduce HOL blocking, while smaller websites often saw detriments. Still, not all larger websites benefited and not all smaller websites suffered. This highlights that a website's structure still also influences web performance and the choice of prioritization. Specifically, we also saw that the mixture of resource priorities and parallelism helps better for moderate loss than pure round-robin, even when using a simplified scheme such as the EPS.

Based on our first insights, we identify several further scenarios that future work can extend on. For instance, we limited our measurements to a single IP to enforce a single connection omitting the influence of third-party resources. Further, we did not visit the influence of non-loss-based congestion control. However, we already saw cases where parallelism deteriorated or improved web performance either depending on a website's structure or the network, which means that HTTP/3 prioritization and even the actual multiplexing has to adapt to both factors. For instance, a mobile network introducing random loss and high RTTs benefits more from per packet round-robin than a burst loss afflicted last mile, which might require that round-robin switches streams after multiples of the burst size to create batches. An AQM deployed on the path could interfere again with the batch size which can also not be signaled today. As such, we believe that a simple one-size-fits-all strategy is improbable but that specifically tailored strategies which not only change per website but also per network are required to get the most out of HTTP/3.

Acknowledgments

This work has been funded by the German Research Foundation DFG under Grant No. WE 2935/20-1 (LEGATO). We thank the anonymous reviewers and our shepherd Anna Brunström for their valuable comments. Furthermore, we would like to thank Konrad Wolsing for providing the sources of [25].

References

[1] M. Bishop, "Hypertext Transfer Protocol Version 3 (HTTP/3)," IETF, Internet-Draft, 2021. https://datatracker.ietf.org/doc/html/draft-ietf-quic-http-34/

[2] M. Belshe, R. Peon, and M. Thomson, "Hypertext Transfer Protocol Version 2 (HTTP/2)," IETF, RFC 7540, 2015. https://datatracker.ietf.org/doc/html/rfc7540

[3] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," IETF, RFC 9000, 2021. https://datatracker.ietf.org/doc/html/rfc9000

[4] M. Wijnants, R. Marx, P. Quax, and W. Lamotte, "HTTP/2 Prioritization and Its Impact on Web Performance," in *Proceedings of the World Wide Web Conference (WWW '18)*, 2018. https://doi.org/10.1145/3178876.3186181

[5] R. Marx, T. De Decker, P. Quax, and W. Lamotte, "Resource Multiplexing and Prioritization in HTTP/2 over TCP Versus HTTP/3 over QUIC," in *Web Information Systems and Technologies (WEBIST '19)*, 2020. https://doi.org/10.1007/978-3-030-61750-9_5

[6] A. Yu and T. A. Benson, "Dissecting Performance of Production QUIC," in *Proceedings of the Web Conference 2021 (WWW '21)*, 2021. https://doi.org/10.1145/3442381.3450103

[7] I. Grigorik, *High Performance Browser Networking: What Every Web Developer Should Know about Networking and Web Performance*. O'Reilly Media, Inc., 2013.

[8] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W.-T. Chang, and Z. Shi, "The QUIC Transport Protocol: Design and Internet-Scale Deployment," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17)*, 2017. https://doi.org/10.1145/3098822.3098842

[9] J. Zirngibl, P. Buschmann, P. Sattler, B. Jaeger, J. Aulbach, and G. Carle, "It's Over 9000: Analyzing Early QUIC Deployments with the Standardization on the Horizon," in *Proceedings of the ACM Internet Measurement Conference (IMC '21)*, 2021. https://doi.org/10.1145/3487552.3487826

[10] X. S. Wang, A. Balasubramanian, A. Krishnamurthy, and D. Wetherall, "Demystifying Page Load Performance with WProf," in *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI '13)*, 2013. https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/wang_xiao

[11] T. Zimmermann, B. Wolters, O. Hohlfeld, and K. Wehrle, "Is the Web Ready for HTTP/2 Server Push?" in *Proceedings of the International Conference on Emerging Networking EXperiments and Technologies (CoNEXT '18)*, 2018. https://doi.org/10.1145/3281411.3281434

[12] A. Davies, "Tracking HTTP/2 Prioritization Issues," 2022. https://github.com/andydavies/http2-prioritization-issues

[13] M. Jiang, X. Luo, T. Miu, S. Hu, and W. Rao, "Are HTTP/2 Servers Ready Yet?" in *IEEE International Conference on Distributed Computing Systems (ICDCS '17)*, 2017. https://doi.org/10.1109/ICDCS.2017.279

[14] K. Oku and L. Pardue, "Extensible Prioritization Scheme for HTTP," IETF, Internet-Draft, 2022. https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-priority-12/

[15] V. Ruamviboonsuk, R. Netravali, M. Uluyol, and H. V. Madhyastha, "Vroom: Accelerating the Mobile Web with Server-Aided Dependency Resolution," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17)*, 2017. https://doi.org/10.1145/3098822.3098851

[16] C. Kelton, J. Ryoo, A. Balasubramanian, and S. R. Das, "Improving User Perceived Page Load Times Using Gaze," in *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI '17)*, 2017. https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/kelton

[17] R. Netravali, A. Goyal, J. Mickens, and H. Balakrishnan, "Polaris: Faster Page Loads Using Fine-grained Dependency Tracking," in *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI '16)*, 2016. https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/netravali

[18] S. Rosen, B. Han, S. Hao, Z. M. Mao, and F. Qian, "Push or Request: An Investigation of HTTP/2 Server Push for Improving Mobile Performance," in *Proceedings of the International Conference on World Wide Web (WWW '17)*, 2017. https://doi.org/10.1145/3038912.3052574

[19] X. S. Wang, A. Balasubramanian, A. Krishnamurthy, and D. Wetherall, "How Speedy Is SPDY?" in *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI '14)*, 2014. https://www.usenix.org/conference/nsdi14/technical-sessions/wang

[20] M. Butkiewicz, D. Wang, Z. Wu, H. V. Madhyastha, and V. Sekar, "Klotski: Reprioritizing Web Content to Improve User Experience on Mobile Devices," in *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI '15)*, 2015. https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/butkiewicz

[21] H. de Saxcé, I. Oprescu, and Y. Chen, "Is HTTP/2 Really Faster Than HTTP/1.1?" in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS '15)*, 2015. https://doi.org/10.1109/INFCOMW.2015.7179400

[22] T. Bergan, "Benchmarking HTTP/2 Priorities," 2016. https://docs.google.com/document/d/1oLhNg1skaWD4_DtaoCxdSRN5erEXrH-KnLrMwEpOtFY/edit

[23] P. Meenan, "How Fast Is Your Website?" *Communications of the ACM*, vol. 56, no. 4, 2013. https://doi.org/10.1145/2436256.2436270

[24] E. Bocchi, L. De Cicco, and D. Rossi, "Measuring the Quality of Experience of Web Users," *ACM SIGCOMM Computer Communication Review*, vol. 46, no. 4, 2016. https://doi.org/10.1145/3027947.3027949

[25] K. Wolsing, J. Rüth, K. Wehrle, and O. Hohlfeld, "A Performance Perspective on Web Optimized Protocol Stacks: TCP+TLS+HTTP/2 vs. QUIC," in *Proceedings of the Applied Networking Research Workshop (ANRW '19)*, 2019. https://doi.org/10.1145/3340301.3341123

[26] P. Biswal and O. Gnawali, "Does QUIC Make the Web Faster?" in *IEEE Global Communications Conference (GLOBECOM '16)*, 2016. https://doi.org/10.1109/GLOCOM.2016.7841749

[27] G. Carlucci, L. De Cicco, and S. Mascolo, "HTTP over UDP: An Experimental Investigation of QUIC," in *Proceedings of the ACM Symposium on Applied Computing (SAC '15)*, 2015. https://doi.org/10.1145/2695664.2695706

[28] S. Cook, B. Mathieu, P. Truong, and I. Hamchaoui, "QUIC: Better For What And For Whom?" in *IEEE International Conference on Communications (ICC '17)*, 2017. https://doi.org/10.1109/ICC.2017.7997281

[29] A. M. Kakhki, S. Jero, D. Choffnes, C. Nita-Rotaru, and A. Mislove, "Taking a Long Look at QUIC An Approach for Rigorous Evaluation of Rapidly Evolving Transport Protocols," in *Proceedings of the ACM Internet Measurement Conference (IMC '17)*, 2017. https://doi.org/10.1145/3131365.3131368

[30] P. Megyesi, Z. Krämer, and S. Molnár, "How Quick Is QUIC?" in *IEEE International Conference on Communications (ICC '16)*, 2016. https://doi.org/10.1109/ICC.2016.7510788

[31] D. Bhat, A. Rizk, and M. Zink, "Not so QUIC: A Performance Study of DASH over QUIC," in *Proceedings of the Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '17)*, 2017. https://doi.org/10.1145/3083165.3083175

[32] K. Nepomuceno, I. N. de Oliveira, R. R. Aschoff, D. Bezerra, M. S. Ito, W. Melo, D. Sadok, and G. Szabó, "QUIC and TCP: A Performance Evaluation," in *IEEE Symposium on Computers and Communications (ISCC '18)*, 2018. https://doi.org/10.1109/ISCC.2018.8538687

[33] J. Rüth, K. Wolsing, K. Wehrle, and O. Hohlfeld, "Perceiving QUIC: Do Users Notice or Even Care?" in *Proceedings of the International Conference on Emerging Networking Experiments And Technologies (CoNEXT '19)*, 2019. https://doi.org/10.1145/3359989.3365416

[34] D. Saif, C.-H. Lung, and A. Matrawy, "An Early Benchmark of Quality of Experience Between HTTP/2 and HTTP/3 Using Lighthouse," in *IEEE International Conference on Communications (ICC '21)*, 2021. https://doi.org/10.1109/ICC42927.2021.9500258

[35] T. Shreedhar, R. Panda, S. Podanev, and V. Bajpai, "Evaluating QUIC Performance over Web, Cloud Storage and Video Workloads," *IEEE Transactions on Network and Service Management (TNSM '21)*, 2021. https://doi.org/10.1109/TNSM.2021.3134562

[36] M. Trevisan, D. Giordano, I. Drago, and A. S. Khatouni, "Measuring HTTP/3: Adoption and Performance," in *Mediterranean Communication and Computer Networking Conference (MedComNet '21)*, 2021. https://doi.org/10.1109/MedComNet52149.2021.9501274

[37] R. Netravali, A. Sivaraman, S. Das, A. Goyal, K. Winstein, J. Mickens, and H. Balakrishnan, "Mahimahi: Accurate Record-and-Replay for HTTP," in *Proceedings of the USENIX Annual Technical Conference (USENIX ATC '15)*, 2015. https://www.usenix.org/conference/atc15/technical-session/presentation/netravali

[38] J. Rüth, I. Kunze, and O. Hohlfeld, "An Empirical View on Content Provider Fairness," in *Network Traffic Measurement and Analysis Conference (TMA '19)*, 2019. https://doi.org/10.23919/TMA.2019.8784684

[39] R. Marx, J. Herbots, W. Lamotte, and P. Quax, "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 (EPIQ '20)*, 2020. https://doi.org/10.1145/3405796.3405828

[40] "RUM-SpeedIndex." https://github.com/WPO-Foundation/RUM-SpeedIndex

[41] C. Sander, L. Blöcher, K. Wehrle, and J. Rüth, "Sharding and HTTP/2 Connection Reuse Revisited: Why Are There Still Redundant Connections?" in *Proceedings of the ACM Internet Measurement Conference (IMC '21)*, 2021. https://doi.org/10.1145/3487552.3487832

[42] R. Marx, T. De Decker, P. Quax, and W. Lamotte, "Of the Utmost Importance: Resource Prioritization in HTTP/3 over QUIC," in *Web Information Systems and Technologies (WEBIST '19)*, 2019. https://doi.org/10.5220/0008191701300143

[43] I. Rhee, L. Xu, S. Ha, A. Zimmermann, L. Eggert, and R. Scheffenegger, "CUBIC for Fast Long-Distance Networks," IETF, RFC 8312, 2018. https://datatracker.ietf.org/doc/html/rfc8312

[44] G. Hasslinger and O. Hohlfeld, "The Gilbert-Elliott Model for Packet Loss in Real Time Services on the Internet," in *GI/ITG Conference - Measurement, Modelling and Evalutation of Computer and Communication Systems*, 2008.

[45] T. Hoßfeld, F. Metzger, and D. Rossi, "Speed Index: Relating the Industrial Standard for User Perceived Web Performance to Web QoE," in *International Conference on Quality of Multimedia Experience (QoMEX '18)*, 2018. https://doi.org/10.1109/QoMEX.2018.8463430

[46] R. Marx, M. Piraux, P. Quax, and W. Lamotte, "Debugging QUIC and HTTP/3 with Qlog and Qvis," in *Proceedings of the Applied Networking Research Workshop (ANRW '20)*, 2020. https://doi.org/10.1145/3404868.3406663

[47] R. Marx, "Head-of-Line Blocking in QUIC and HTTP/3: The Details," 2020. https://github.com/rmarx/holblocking-blogpost