Web Performance Automation for the People

Robin Marx*
Hasselt University – tUL
Expertise Centre for Digital Media
Diepenbeek, Belgium
robin.marx@uhasselt.be

ABSTRACT

Web performance is important for the user experience and can heavily influence web page revenues. While there are many established Web Performance Optimization (WPO) methods, our work so far has clearly shown that new network protocols, optimized browsers and cutting-edge web standards can have a significant impact on known best practices. Additionally, there is still low-hanging fruit to be exploited, in the form of personalizing performance based on user context (i.e., current device, network, browser) and user preferences (e.g., text reading vs multimedia experience).

In our PhD project, we strive to integrate this user-specific metadata into dynamic configurations for both existing and new automated WPO techniques. An intermediate server can (pre)generate optimized versions of a web page, which are then selected based on user context and preferences. Additional metadata is also passed along to the browser, enabling improvements on that side, and used to steer new network protocols to speed up the incremental delivery of page resources.

We use the Speeder platform to perform and evaluate full-factorial objective measurements and use subjective user studies across a range of groups to assess the applicability of our methods to end users. Our aim is to provide insights in how WPO can be tweaked for specific users, in the hopes of leading to new web standards that enable this behavior.

CCS CONCEPTS

• **Networks** → **Network protocol design**; **Transport protocols**; **Network performance evaluation**; *Application layer protocols*; *Middle boxes / network appliances*; • **Human-centered computing** → *User studies*; Contextual design; • **Social and professional topics** → Automation; • **Software and its engineering** → Software performance;

KEYWORDS

Web Performance Optimization (WPO); Page Load Time (PLT); User Context; QUIC Protocol; Distributed Systems; Web browsers; Networking; Systems Automation

*Robin Marx is a SB PhD fellow at FWO, Research Foundation - Flanders, project number 1S02717N. This PhD project is supervised by Prof. Peter Quax, PhD and Prof. Wim Lamotte, PhD.

WWW '18 Companion, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5640-4/18/04.
https://doi.org/10.1145/3184558.3186570

ACM Reference Format:
Robin Marx. 2018. Web Performance Automation for the People. In *WWW '18 Companion: The 2018 Web Conference Companion, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3184558.3186570

1 PROBLEM STATEMENT

Modern websites and web-based applications are often slow to load and interact with, especially on mobile devices and cellular networks. This suboptimal performance is annoying to the user; so annoying in fact, that even slight increases in page load time can have a measurable and consistent impact on retention, conversions and revenue in e-commerce scenarios [30]. A study by Microsoft and LinkedIn researchers [17] found that "an engineer that improves server performance by 10msec [...] more than pays for his fully-loaded annual costs. Every millisecond counts.". We cannot solely rely on advances in network and device speeds to ensure happier users over time, as web page and user agent complexity has been incrementing at staggering rates in recent years [14].

Given these challenges, the state of the art on Web Performance Optimization (WPO) is well-established in research (§2) and especially in industry. Many evergreen techniques (e.g., compression, unused code elimination) can be applied automatically at the server side, either in a pre-processing step or at runtime (e.g., modpagespeed). Advanced backend setups can go one step further, also using (limited) user context (e.g., current network connection type, device) to provide further customized optimizations [1] or, in the case of cloud-based browsers, deeply rewrite the web page's representation [8, 36] (though this can negatively impact user experience in other ways). On the user side, modern browsers use advanced heuristics [24] to optimize the common case and provide developers with APIs (e.g., preload directives) to embed additional performance hints. The Accelerated Mobile Pages (AMP) project combines custom HTML/JavaScript APIs with cloud-based hosting to provide improved performance for a subset of use-cases (e.g., news articles). In addition to these automated techniques, developers can also use a wide range of synthetic monitoring and analysis tools (e.g., Webpagetest.org, Sitespeed.io) to manually assess performance bottlenecks. Real User Monitoring (RUM) solutions on the other hand, gather performance information from real page loads at the user's devices, offering more realistic insights. This additional data is often needed to manually tweak individual page performance, as the automated methods make mistakes or use too conservative heuristics.

Even though these automated methods and manual mitigations exist, they might not be enough to guarantee optimal performance in the face of several recent and upcoming changes in the internet landscape and the changing expectations of end users. We want to

focus on three main areas that can have a large impact on existing performance best practices and would require updates to established methods:

- **Area 1 (A1): New network protocols** The recently standardized HTTP/2 and the upcoming QUIC [5] protocols are slated to replace the ageing HTTP/1.1. While they were very much designed with performance in mind, previous work (§2.2, §5) has shown that it can be difficult to fine-tune new protocol features and that differing implementations can lead to large performance impacts. As such, HTTP/2 best practices have yet to emerge and several large sites (e.g., Amazon, Uber, Netflix) have yet to (fully) transfer to the new protocol. Our main research question in this area is which new opportunities and possible pitfalls QUIC brings and how to best put them to use.
- **Area 2 (A2): Rising user context heterogeneity** As devices and networks have become faster, web page complexity has grown accordingly. Mobile devices on slower networks often fail to deliver proper page load performance for these more complex pages. Additionally, user agent implementations are diversifying, with Firefox v57 presenting a complete overhaul and the UC browser dethroning Google Chrome in Asian countries [25]. Existing (automated) methods often do not allow to adjust the optimizations based on this user context and thus many websites continue to serve just two or even one (responsive) version of the website, forcing low-end smartphones to download and process much the same resources as high-end desktops. Our main research question here is how we can tweak new and existing WPO techniques to better fit heterogeneous user contexts.
- **Area 3 (A3): Subjective user preferences** The end user's experience with regards to performance can be highly subjective and individualistic (§2.4). For example, Opera Mini [8] users enjoy faster page loads, making the tradeoff to lose runtime JavaScript execution. Additionally, Progressive JPG images can be annoying and even feel slower to some users [10]. Users are also looking to personalize their web browsing experience in other ways (e.g., the Brave browser provides a different model for serving advertisements). However, almost no existing performance methods take into account (explicit) individual user preferences, instead treating all users the same. For example, modern browsers include a "reader mode", but only enable this after all superfluous content has been downloaded and processed. We mainly aim to research how (explicit) user preferences can be used to guide the (combination of) used WPO techniques in order to generate better individualized experiences.

2 STATE OF THE ART

2.1 Page load bottlenecks

A large body of work has focused on identifying the main bottlenecks in web page loading behaviour. Starting with the WProf study [35], which identified how resource inter-dependencies and load orders define the "critical path", several systems have achieved impressive performance gains by manipulating these load orders and resource fetch priorities (e.g., Klotski [4], Polaris [22], Vroom [28] and Meta-Push [13]). These works all use additional metadata, either on the browser or server side, to further optimize loading performance.

In tandem, other work has focused particularly on mobile performance, and often finds that there the computation capabilities of the device, not the network [37], are often the bottleneck [6], noting that "we need to fundamentally rethink optimizations for mobile browsers"[21]. Yet, many of these studies often only consider a single browser and/or are several years old, warranting a possible re-evaluation, given the rapid browser implementation evolution.

2.2 Network protocols

The new HTTP/2 protocol has been the subject of a large variety of research in recent years. Sadly, our survey shows that many of the studies [7, 9] produce contradictory results (e.g., that Server Push should be used very sparingly [3, 41] versus aggressively [27]). It is our opinion that this is mainly due to limited network emulation variety in individual test setups and researchers testing different versions of the proposed protocols. As the new, even more flexible QUIC protocol is gaining momentum [5], we expect these contradictions to continue, as corroborated by the recent work of Kakhki et al. [15].

Luckily, many of the publications also agree that, depending on certain network parameters (e.g., packet loss), different approaches (e.g., opening additional TCP connections [11]) lead to superior performance, which is also corroborated by our own experiments (§5). As the QUIC protocol allows much more freedom in all levels of the protocol stack, we envision that deep configuration depending on network parameters can be used to tweak performance, as hinted by similar work from Qian et al. [26].

2.3 Proxy and cloud-based browsing

Many of the existing works on WPO are practically implemented and evaluated using an intermediate proxy or cloud-based server that helps provide the speedup. For example, Shandian [36] is similar to Opera Mini [8] in that it does (partial) execution of JavaScript on the server side. Other works bundle resources together [23], manipulate resource loading orders [18, 28] or prefetch and precache website assets [29, 32]. This shows that using such an intermediate server is a valid approach in our own efforts (§3, step 2).

2.4 Subjective performance metrics

WPO techniques are often measured using objective metrics, obtained from the web browser or by observing the visual load progression of the page. However, recent research has clearly shown that these metrics are badly correlated with actual subjective user opinions on perceived page "readiness" [16, 33, 42] and that these opinions are often highly individualized. Related HCI research also indicates that (automatically) personalized user interfaces can lead to shorter task solve times [31]. These findings underscore the potential of the incorporation of (explicit) user preferences into the WPO setup.

3 PROPOSED APPROACH

We envision a traditional setup where an intermediate server (§2.3) optimizes the web page and its resources, received from the origin, either ahead-of-time or upon receiving a request. This setup is akin to the model used by most Content Delivery Networks (CDNs), as we envision our work to be of most interest to that industry.

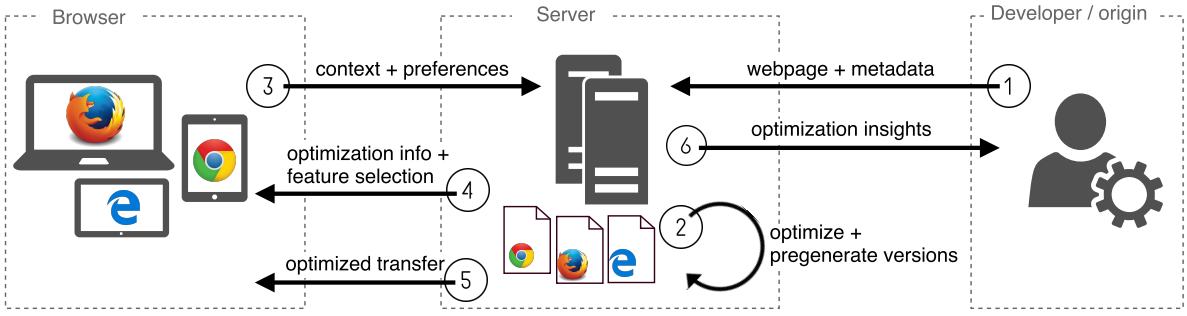

Figure 1: Overview of our proposed overall system. Steps 3, 4 and 5 include our main envisioned contributions.

Our main contributions will be in steps 3, 4 and 5 in Figure 1. The key to our approach is to increase the bi-directional communication of performance-related metadata between user agent and server (steps 3 and 4). This additional information allows us to select much more appropriate, context-related optimizations (step 2) and to tweak the transfer of the actual page data (step 5) using protocol settings appropriate to the current context (e.g., use of multiple connections depending on network quality, §2.2). In turn, the browser can use metadata on the applied optimizations to improve its heuristics and internal processing (steps 4,5). Optimization insights can be communicated back to the developers (step 6) and so lead to more optimized initial web pages (step 1).

Step 3: User context and preferences (A2 and A3)
Until recently, browsers exposed relatively little user context and preferences. This has changed with ongoing work on Client Hints (e.g., indicating screen size), the NetInfo API (providing estimates of current network quality parameters), the Memory Pressure API and Cache Digests. However, several of these options are still very experimental and their optimal usage is yet unknown. Additionally, there are no provisions yet for explicit user preferences, such as to indicate a metered (cellular) connection, to select a type of advertisement (e.g., video vs text-only), to enforce reader mode, to disable animations or to profess a preference to only lazy load images/video.

The main challenges here are to evaluate the usefulness of the data from these new APIs and how to communicate this data to the server as soon as possible. Additionally, we must look at which user preferences are most useful and how they can be integrated into the browser in a usable, secure and privacy-aware way.

Step 4: Page optimization metadata (A2)
Browsers can profit from having additional metadata about the contents of and optimizations applied to the web page. This is currently already partly possible through methods such as Resource Hints or Server Push [13], but more advanced options like explicit Priority Hints are not yet standardized. In all, browsers often do not know what a page will contain ahead of time and have to iteratively discover the subresources and their contents. This forces the browser to use (sub-optimal) heuristics and to be overly conservative about its approach.

If the browser has more detailed metadata up-front (e.g., file sizes for each resource, full list of resources with their respective priority and conceptual contribution to the page) it can dramatically improve its performance, as has already been hinted at in related work (§2.1,§5). Additionally, if the browser is 100% sure that a web page will not use a given JavaScript/CSS API or is structured in a given, well-known way, it might opt for a faster execution path, as slower subsystems can be skipped entirely. While these ideas have been explored in previous work, we aim to further evaluate these possibilities, especially for newer web standards.

Step 5: Optimized network transfer (A1)
Even though the newer network protocols provide clear advantages over the old HTTP/1.1, it is clear that a single configuration will not produce optimal results in all settings. For example, HTTP/2's single connection suffers from packet loss and performance can be improved by using multiple parallel connections [11, 19]. For other new features, such as Server Push and Prioritization, it is yet unclear what the best practices are and how they are influenced by user context [3, 40].

The new QUIC protocol re-implements many of TCP's features in userspace and as such allows even more dynamic configuration (e.g., selecting a different congestion avoidance algorithm based on current network conditions). As QUIC is in the process of being standardized, it is an area of active research and existing studies on older versions of QUIC might have to be nuanced based on newer iterations [15]. We believe QUIC's design allows for many tweakable performance optimizations, more so than other similar protocols.

Step 2: Automated resource rewriting (A1 and A3)
While the network transfer configurations will act mainly on user context, extensive resource rewriting can be used to also adhere to the explicit user preferences. We envision an intermediate server that primarily selects and combines established automated techniques to (pre)generate various different representations of the web page, depending on user context and selected performance preferences (e.g., reader mode version with very simple layout, version that lazy loads images, version with heavy resource compression/bandwidth optimized, version purely for Google Chrome, etc.).

This intermediate server can also generate the optimization-related metadata (e.g., prioritized list of resources) to be communicated to the browser in Step 4 or to be used for aspects such as Server Push in Step 5. While previous work includes several approaches in this regard [22, 36], we expect additional work will be needed to generate additional metadata and look at how developers might provide this metadata themselves.

4 METHODOLOGY

4.1 Planned developments

Custom QUIC and HTTP/2 Server

We are currently in the process of developing a custom NodeJS-based QUIC and HTTP/2 server. This will allow us to gain a deep understanding of the protocol and to more quickly and dynamically test several aspects and (congestion control) algorithms within the QUIC protocol and to compare them to other default implementations.

Intermediate server setup

We will implement an intermediate server which integrates many of the existing automated WPO techniques. These techniques are then combined into several "profiles" that adhere to specific combinations of user context and preferences. These profiles can then be used to either prefetch and pregenerate optimized versions of popular websites or can be applied on-the-fly. The profiles can also further be automatically iteratively refined by testing their output using the Speeder framework (§4.2) and discarding suboptimal permutations.

Custom web browser(s)

It will be necessary to create custom versions of existing web browsers, both to enable fast path processing based on the received metadata and to allow additional user context and especially explicit user preferences to be communicated to the server. To this end, we plan to focus primarily on either/or the open source Chromium and Firefox browsers, starting with an extensive source code review to identify possible improvements and then incrementally creating several custom versions. This should eventually lead to a version to be used in a longer-running subjective user study.

4.2 Evaluation amenities

The Speeder framework for objective measurements

We have already developed an in-house testing framework named Speeder. This automated platform allows for easy test setup and full-factorial evaluations over multiple browsers, physical devices, server implementations, network emulations and protocols. We use tools such as WebPageTest.org, Sitespeed.io and Google Lighthouse to gather objective performance metrics. The platform includes a variety of different visualizations and statistical methods to evaluate the gathered data (§5). In this, Speeder takes a more integrated approach than similar frameworks, such as mahimahi [23], which typically focus on either only the front or backend and rarely integrate visualization or analysis tools. Speeder will allow us full control to test our custom QUIC and browser implementations. Table 1 shows the current Speeder features. More details are available on https://speeder.edm.uhasselt.be.

Subjective user studies

We plan to conduct several user studies and questionnaires. Firstly, we need to obtain additional insight in typical user contexts that can influence performance and into which explicit user preferences would be interesting for a wide range of end-users. Secondly, we need to get a better feeling for why subjective user metrics often mismatch with objective metrics (§2.4) and how to handle this. Lastly, the final version of our custom web browser that integrates advanced user context and preferences should be tested in live scenarios by real users. As we currently have little direct experience in this area, we plan to rely on methods from previous work (§2.4) and the support of our local colleagues active in HCI research. We will use groups consisting of students, volunteering web performance professionals and subjects hired through platforms such as Mechanical Turk.

Table 1: Software, metrics and visualizations supported in the Speeder framework (December 2017).

Protocols	HTTP/1.1 (cleartext), HTTPS/1.1, HTTPS/2
Browsers	Chrome (v51 - v63), Firefox (v45 - v56)
Test drivers	Sitespeed.io (v3), Webpagetest (v3.0)
Servers	Apache (v2.4.20), NGINX (v1.10), NodeJS (v6.2.1), H2O (v2.1)
Network	- `DUMMYNET` (cable and cellular) - fixed `TC NETEM` (cable and cellular) - dynamic `TC NETEM` (cellular) [11]
Metrics	All Navigation Timing values [34], SpeedIndex [39], other Webpagetest metrics [38], Google Lighthouse metrics
Visualizations	Packet timeline (TCP and HTTP/2), HTTP/2 priority dependency trees. Boxplots, linegraphs and CDFs of recorded metrics

5 RESULTS

Our work in the past year has led to three accepted publications and one pending overarching journal publication.

Our first publication validates HTTP/1.1-era best practices in an HTTP/2 setup [19]. We used the Speeder framework to do a broad analysis of different factors and found that HTTP/2's performance is heavily dependent on not only the network quality, but also on the individual browser's internal implementation and handling. Different HTTP/2 configurations (e.g., using multiple parallel connections, more or less resource aggregation) can thus lead to improved performance, depending on these contextual parameters (§3 Steps 3,5). Our results were confirmed by related work [11] and directly contradict earlier expectations [12].

Our second publication, focused on HTTP/2's resource prioritization system [40], shows even more clearly that there are large differences in individual browser implementations and that suboptimal heuristics can have detrimental effects on performance, further demonstrating the need for additional metadata (§3 Steps 3,4,5). We were one of the first to dig deep into HTTP/2's prioritization system and our results confirm the main previous work [2] while providing additional insights into the reasons for the observed behaviors.

Our third publication, dealing with transpiling HTML/CSS pages into WebVR-compatible representations [20], was used as a vehicle to gain more insight in the underlying details of the browser's JavaScript and CSS engines. Our findings show that these engines often have to keep large amounts of internal state to provide esoteric APIs, leaving opportunities for the integration of fast paths (§3 Step 4). Our implementation achieves an order of magnitude performance increase compared to equivalent existing approaches.

6 CONCLUSIONS AND FUTURE WORK

Through a survey of existing work and our own efforts in the past year, we have identified several new opportunities for progressing Web Performance Optimization best practices, based on advances in

network protocols and browser implementations. We aim to employ these opportunities in a personalized manner, by taking into account deep user context and explicit user preferences, improving on the same-size-fits-all approach that is currently still common practice.

Aiming to end our PhD project in three years, we plan to spend year one primarily on preparation for the individual aspects of our setup: the implementation and testing of the QUIC server, browser source code review plus minor custom adjustments and subjective user studies surrounding (image) placeholder format preferences. The second year will center around integrating the different aspects into a single setup, implementing the intermediate proxy server and defining the optimization profiles. The last year will then produce the final custom browser version, to be tested and evaluated by real users. Throughout the whole period we will also update the Speeder framework to be able to test as many parameters as possible.

REFERENCES

[1] Victor Agababov, Michael Buettner, Victor Chudnovsky, Mark Cogan, Ben Greenstein, Shane McDaniel, Michael Piatek, Colin Scott, Matt Welsh, and Bolian Yin. 2015. Flywheel: Google's Data Compression Proxy for the Mobile Web.. In *NSDI*, Vol. 15. 367–380.

[2] Tom Bergan. 2016. Benchmarking HTTP/2 Priorities. Online, https://docs.google.com/document/d/1oLhNg1skaWD4$_D$taoCxdSRN5erEXrH-KnLrMwEpOtFY/. (October 2016).

[3] Tom Bergan, Simon Pelchat, and Michael Buettner. 2016. Rules of Thumb for HTTP/2 Push. Online, https://docs.google.com/document/d/1K0NykTXBbbbTlv60t5MyJvXjqKGsCVNYHyLEXIxYMv0. (2016).

[4] Michael Butkiewicz, Daimeng Wang, Zhe Wu, Harsha V. Madhyastha, and Vyas Sekar. 2015. KLOTSKI: Reprioritizing Web Content to Improve User Experience on Mobile Devices. In *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation (NSDI'15)*. 439–453.

[5] Gaetano Carlucci, Luca De Cicco, and Saverio Mascolo. 2015. HTTP over UDP: an Experimental Investigation of QUIC. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. ACM, 609–614.

[6] Calin Cascaval, Seth Fowler, Pablo Montesinos-Ortego, Wayne Piekarski, Mehrdad Reshadi, Behnam Robatmili, Michael Weber, and Vrajesh Bhavsar. 2013. ZOOMM: a parallel web browser engine for multicore mobile devices. In *ACM SIGPLAN Notices*, Vol. 48. ACM, 271–280.

[7] Hugues de Saxcé, Iuniana Oprescu, and Yiping Chen. 2015. Is HTTP/2 really faster than HTTP/1.1?. In *Proceedings of the IEEE Conference on Computer Communications Workshops*. 293–299.

[8] Dev.Opera. 2012. Opera Mini and JavaScript. Online, http://dev.opera.com/articles/view/opera-mini-and-javascript/. (2012).

[9] Jeffrey Erman, Vijay Gopalakrishnan, Rittwik Jana, and K. K. Ramakrishnan. 2013. Towards a SPDY'ier Mobile Web?. In *Proceedings of the 9th ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT'13)*. 303–314.

[10] Everts, Tammy. 2014. Progressive image rendering: Good or evil? Online, https://blog.radware.com/applicationdelivery/wpo/2014/09/progressive-image-rendering-good-evil/. (2014).

[11] Utkarsh Goel, Moritz Steiner, Mike P Wittie, Stephen Ludin, and Martin Flack. 2017. Domain-Sharding for Faster HTTP/2 in Lossy Cellular Networks. *arXiv preprint arXiv:1707.05836* (2017).

[12] Ilya Grigorik. 2013. *High Performance Browser Networking*. " O'Reilly Media, Inc.".

[13] Bo Han, Shuai Hao, and Feng Qian. 2015. MetaPush: Cellular-Friendly Server Push For HTTP/2. In *Proceedings of the Workshop on All Things Cellular (AllThingsCellular'15)*. 57–62.

[14] Ilya Grigorik and Pat Meenan and Rick Viscomi. 2017. HTTP Archive. Online, http://httparchive.org/. (October 2017).

[15] 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*. ACM, 290–303.

[16] Conor Kelton, Jihoon Ryoo, Aruna Balasubramanian, and Samir R Das. 2017. Improving User Perceived Page Load Times Using Gaze.. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 545–559.

[17] Ron Kohavi, Alex Deng, Roger Longbotham, and Ya Xu. 2014. Seven rules of thumb for web site experimenters. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1857–1866.

[18] Xuanzhe Liu, Yun Ma, Xinyang Wang, Yunxin Liu, Tao Xie, and Gang Huang. 2017. Swarovsky: Optimizing resource loading for mobile web browsing. *IEEE Transactions on Mobile Computing* 16, 10 (2017).

[19] Robin Marx, Peter Quax, Axel Faes, and Wim Lamotte. [n. d.]. Concatenation, Embedding and Sharding: Do HTTP/1 Performance Best Practices Make Sense in HTTP/2?. In *Proceedings of the 13th International Conference on Web Information Systems and Technologies (WEBIST'17)*. Scitepress.

[20] Robin Marx, Sander Vanhove, Wouter Vanmontfort, Peter Quax, and Wim Lamotte. 2017. DOM2AFrame: Putting the Web back in WebVR. In *Proceedings on the Int. Conf. on 3D Immersion (IC3D17)*. IEEE.

[21] Javad Nejati and Aruna Balasubramanian. 2016. An in-depth study of mobile browser performance. In *Proceedings of the 25th International Conference on World Wide Web*. 1305–1315.

[22] Ravi Netravali, Ameesh Goyal, James Mickens, and Hari Balakrishnan. 2016. Polaris: Faster Page Loads Using Fine-grained Dependency Tracking. In *Proceedings of the 13th USENIX Conference on Networked Systems Design and Implementation (NSDI'16)*. 123–136.

[23] Ravi Netravali, Anirudh Sivaraman, Somak Das, Ameesh Goyal, Keith Winstein, James Mickens, and Hari Balakrishnan. 2015. Mahimahi: Accurate Record-and-Replay for HTTP.. In *USENIX Annual Technical Conference*. 417–429.

[24] Osmani, Addy. 2017. Preload, Prefetch And Priorities in Chrome. Online, https://medium.com/reloading/preload-prefetch-and-priorities-in-chrome-776165961bbf. (2017).

[25] Purnell, Newley. 2018. UC Browser vs Google Chrome. Online, https://www.wsj.com/articles/a-browser-youve-never-heard-of-is-dethroning-google-in-asia-1514808002. (2018).

[26] Feng Qian, Vijay Gopalakrishnan, Emir Halepovic, Subhabrata Sen, and Oliver Spatscheck. 2015. TM 3: flexible transport-layer multi-pipe multiplexing middlebox without head-of-line blocking. In *Proc. of the 11th ACM Conf. on Emerging Networking Experiments and Technologies*.

[27] Sanae Rosen, Bo Han, Shuai Hao, Z Morley Mao, and Feng Qian. 2017. Push or Request: An Investigation of HTTP/2 Server Push for Improving Mobile Performance. In *Proceedings of the 26th International Conference on World Wide Web*. 459–468.

[28] Vaspol Ruamviboonsuk, Ravi Netravali, Muhammed Uluyol, and Harsha V Madhyastha. 2017. VROOM: Accelerating the Mobile Web with Server-Aided Dependency Resolution. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*.

[29] Ali Sehati and Majid Ghaderi. 2015. WebPro: A proxy-based approach for low latency web browsing on mobile devices. In *Quality of Service (IWQoS), 2015 IEEE 23rd International Symposium on*. IEEE, 319–328.

[30] Tammy Everts and Tim Kadlec. 2017. WPO Stats. Online, https://wpostats.com/. (October 2017).

[31] Kashyap Todi, Jussi Jokinen, Antti Oulasvirta, and Kris Luyten. 2018. Familiarisation: Restructuring Layouts with Visual Learning Models. In *International Conference on Intelligent User Interfaces 2018 (IUI)*. ACM.

[32] Jeroen van der Hooft, Stefano Petrangeli, Tim Wauters, Rameez Rahman, Nico Verzijp, Rafael Huysegems, Tom Bostoen, and Filip De Turck. 2017. Analysis of a large multimedia-rich web portal for the validation of personal delivery networks. In *Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on*. IEEE, 714–719.

[33] Matteo Varvello, Jeremy Blackburn, David Naylor, and Konstantina Papagiannaki. 2016. EYEORG: A Platform For Crowdsourcing Web Quality Of Experience Measurements. In *Proceedings of the 12th ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT'16)*. 399–412.

[34] W3C Recommendation. 2012. Navigation Timing. Online, https://www.w3.org/TR/navigation-timing. (December 2012).

[35] Xiao Sophia Wang, Aruna Balasubramanian, Arvind Krishnamurthy, and David Wetherall. 2013. Demystifying Page Load Performance with WProf. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation (NSDI'13)*. 473–486.

[36] Xiao Sophia Wang, Arvind Krishnamurthy, and David Wetherall. 2016. Speeding Up Web Page Loads with Shandian. In *Proceedings of the 13th USENIX Conference on Networked Systems Design and Implementation (NSDI'16)*. 109–122.

[37] Zhen Wang, Felix Xiaozhu Lin, Lin Zhong, and Mansoor Chishtie. 2011. Why are web browsers slow on smartphones?. In *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications*. ACM, 91–96.

[38] WebPagetest. 2017. Website Performance and Optimization Test. Online, https://www.webpagetest.org/. (2017).

[39] WebPageTest Documentation. 2012. Speed Index. Online, https://sites.google.com/a/webpagetest.org/docs/using-webpagetest/metrics/speed-index. (2012).

[40] Maarten Wijnants, Robin Marx, Peter Quax, and Wim Lamotte. 2018. HTTP/2 Prioritization and its Impact on Web Performance. In *WWW 2018: The 2018 Web Conference, April 23-27, 2018, Lyon, France*. ACM.

[41] Kyriakos Zarifis, Mark Holland, Manish Jain, Ethan Katz-Bassett, and Ramesh Govindan. 2017. Making Effective Use of HTTP/2 Server Push in Content Delivery Networks. In *Technical report*. University of Southern California.

[42] Torsten Zimmermann, Benedikt Wolters, and Oliver Hohlfeld. 2017. A QoE Perspective on HTTP/2 Server Push. In *Proc. Workshop on QoE-based Analysis and Management of Data Comm. Networks*. ACM, 1–6.