Measuring Web Latency and Rendering Performance: Method, Tools & Longitudinal Dataset

Alemnew Sheferaw Asrese*, Steffie Jacob Eravuchira‡, Vaibhav Bajpai†, Pasi Sarolahti*, Jörg Ott†
*Aalto University, Finland †Technische Universität München, Germany ‡SamKnows Limited, UK
(alemnew.asrese | pasi.sarolahti)@aalto.fi,
(bajpaiv | ott)@in.tum.de, steffie@samknows.com

***Abstract*—This paper presents *Webget*, a measurement tool that measures web Quality of Service (QoS) metrics including the DNS lookup time, time to first byte (TTFB) and the download time. *Webget* also captures web complexity metrics such as the number and the size of objects that make up the website. We deploy the *Webget* test to measure the web performance of Google, YouTube, and Facebook from 182 SamKnows probes. Using a 3.5-year-long (Jan 2014 - Jul 2017) dataset, we show that the DNS lookup time of these popular Content Delivery Networks (CDNs) and the download time of Google have improved over time. We also show that the TTFB towards Facebook exhibits worse performance than the Google CDN. Moreover, we show that the number and the size of objects are not the only factors that affect the web download time. We observe that these webpages perform differently across regions and service providers. We also developed a web measurement system, *WePR* (Web Performance and Rendering) that measures the same web QoS and complexity metrics as *Webget*, but it also captures the web Quality of Experience (QoE) metrics such as rendering time. *WePR* has a distributed architecture where the component that measures the web QoS and complexity metrics is deployed on the SamKnows probe, while the rendering time is calculated on a central server. We measured the rendering performance of four websites. We show that in 80% of the cases, the rendering time of the websites is faster than the downloading time. The source code of the *WePR* system and the dataset is made publicly available.**

I. Introduction

The web has evolved from a simple static text and image delivery platform to a complex ecosystem with multiple dynamic and media-rich contents. The delivery of the contents has also changed from a single server to a number of servers often spread across different administrative domains. This complexity increases the download time of the websites [1] and consequently degrades the user experience. Studies show that the user experience has a significant impact on business revenue [2] whereby users with bad web experience tend to abandon the websites early. One of the most frequently used metrics in analyzing web performance is latency; the shorter the waiting time to get the contents, the more the user is satisfied with the service [3].

In order to reduce the latency and improve the web performance, various improvements in content delivery mechanisms, transport and application protocols have been proposed. For instance, TCP improvements such as reordering [4], packet losses [5] and startup performance [6] along with new protocols such as QUIC [7] and HTTP/2 [8] are getting deployed to improve the web performance. Despite these efforts, the web performance is not perfect and the user still expects a lower latency when surfing the web.

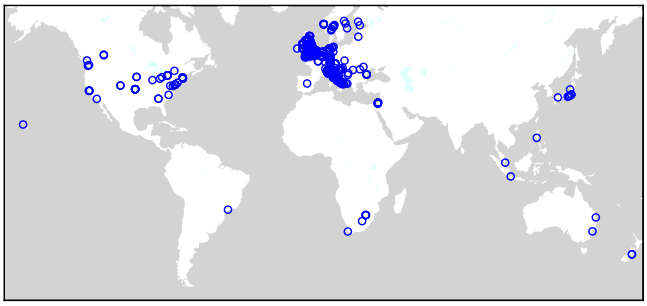


Fig. 1: Geographical distribution of 182 SamKnows probes. More than 85% of them are at residential network.

In this work, we study how the web performance has evolved over time, what factors contribute towards it and how they differ by ISP and by region. We seek also to understand how the download performance of webpages contributes to the visual rendering performance. Towards this end, we designed and developed a web measurement tool, *Webget*, that measures the web QoS metrics for static objects that make up a webpage such as the DNS lookup time, TTFB and the download time of each object. *Webget* also measures web complexity metrics including the number and the size of objects that make up the webpage. Sundaresan *et al.* [9] deployed this test (they call it Mirage) on Bismark probes to measure 9 websites. They deconstructed the page load time (PLT) of static objects into constituent components (DNS lookup time, TCP connection setup and object download time) and showed that latency optimizations can yield a significant improvement in overall PLT. Our work extends their analysis [9] by presenting longitudinal aspects (3.5 years long) of web performance as measured from SamKnows probes [10] using a similar sample size of websites. Given, the SamKnows probes have limited resources and cannot execute JavaScript, we also extended the methodology by developing *WePR*, a measurement system [11] that allows offloading the JavaScript execution to a separate parsing server, thereby providing the possibility to also evaluate the website rendering behavior which is essential to understand the web QoE. *WePR* has a distributed architecture. We deploy the component that measures the web QoS and complexity metrics at the end-user location and calculate the rendering time on a centralized

server. This allows *WePR* to be deployed at scale and measure the web performance without user interaction. We measured the performance of specific webpages of three most popular websites (`www.google.com, www.youtube.com, www.facebook.com`) using *Webget* from 182 SamKnows probes connected at 70 different origin Autonomous Systems (ASes) (see Fig. 1) with more than 85% of these probes connected in residential networks.

Previous studies [1], [12]–[15] that have measured a large sample of destinations either provide a snapshot view or measure from a smaller number of vantage points. In this work, we do not aim to exhaustively cover a large number of destinations. The goal of our work is to evaluate the web performance of Google and Facebook CDNs as seen from a large number of vantage points. We also focus on understanding factors that contribute to web performance bottlenecks over a longitudinal period as the broadband speeds of customers improve over the years. We chose to measure Google and Facebook CDNs because, we observe that a quarter of Alexa top 100 websites are `google.*`, `youtube.com` and `blogspot.*` hosted on Google alone [16], [17], while 3% websites are hosted on Facebook. The traffic distribution of the Alexa top websites also shows a Pareto distribution whereby Google CDN (also hosting YouTube) and Facebook CDN generate the majority [18], [19] of the traffic in ISP networks. In this paper, we analyze the download performance using 3.5 years long dataset collected from the probes. Additionally, we also measured the rendering performance of four websites using our *WePR* system as seen from 65 probes (a subset of the 182 SamKnows probes). We also present the web rendering performance analysis using a nine months long dataset. Overall, we provide two main contributions –

1. Method and Tools – We describe the design and implementation (§ III) of a web QoS measurement tool (*Webget*) and a measurement system (*WePR*) that can measure both web QoS and web rendering performance at scale. The measurement system is validated (§ IV) and the benchmarking performance of the system is presented, along with the deployment (§ V) of the measurement system and the collected dataset that we publicly release to the community.

2. Longitudinal Dataset – The first longitudinal (Jan 2014 - Jul 2017) dataset shows that the DNS lookup time for the three websites (§ VI) and the download time towards Google have improved over time. The webpage complexity (in terms of the number and the size of objects) [1] alone does not affect the webpage download performance. For instance, Google's webpage that we measure has a higher complexity compared to Facebook and YouTube. Yet, Google has a shorter download time than YouTube, and a shorter TTFB than Facebook and YouTube. This is due to content cache deployments within the ISP's network that lower the IP path lengths towards Google. We also witness a small improvement in TTFB of Google over the years. We show that broadband speed improvements does not always yield a better web performance. Our ISP-based analysis reveals that probes connected within the Comcast network in ∼40% of the cases observe longer TTFB towards Facebook. Our region-based analysis shows that probes within ARIN and RIPE region (§ VI) exhibit a better download performance. The second nine-months long dataset of rendering performance towards four websites shows that the download time of the websites is twice longer than the time required to render (§ VII) the visible portion of the websites in half of the measurements.

This paper builds on our earlier work [11]. In this paper, we added substantial background material, including a survey (§ II) of recent web performance testing, monitoring and benchmarking tools and related methods. We also performed a longitudinal analysis of web QoS (§ VI) using a 3.5 years (Jan 2014 – Jul 2017) long collected dataset. In addition, we repeated the analysis in the previous work [11] using a larger (9 months long, Mar 2015 – Dec 2015) dataset. We highlight the implications (§ VIII) of our measurement results towards the management and operations of networks and also discuss limitations (§ IX) and future possibilities of this work

To encourage reproducibility [20], [21], the measurement system [22] is open-sourced. The entire dataset and software used in this study is also made publicly available [23].

II. Related Work

We present related work focusing on the evolution of webpages, impact of latency on web performance [24], mechanisms employed to reduce web latency and improve QoE.

Webpage evolution – Fetterly *et al.* [25] studied how the web has been changing, and how often the changes happen in web content. Others studied where the contents are hosted, how they are replicated and served to the users [26], and also examine the changes in web traffic patterns [27], [28].

Web latency – Several studies have evaluated how web latency affects the overall user satisfaction and experience. Arapakis *et al.* [29] studied the impact of response latency on the user behavior in web search. They showed that users are sensitive to increasing delays in the response. Flach *et al.* [5] analyzed the effects of TCP timeout and loss recovery on webpage latency. They used redundant transmissions to design a new loss recovery mechanisms for TCP so as to reduce the latency caused by TCP's timeout-driven recovery mechanism. Mandalari *et al.* [30] studied the roaming ecosystem and observed web latency penalties due to the home routing policy adopted by mobile operators within Europe.

Web performance – Sundaresan *et al.* [9] used *Webget* to study performance towards nine popular websites from 5K Bismark probes. They show that in situations where the throughput of the access link is more than 16 Mbps, latency becomes the main factor affecting PLT. They show that DNS and TCP connection caching at the edge can yield improvements to the overall PLT. Zaki *et al.* [13] studied web performance in developing regions. They showed that the main causes for poor web performance in these regions are lack of good DNS infrastructure and pervasive content caching. Fanou *et al.* [31] showed that the inter-AS delays and the non-availability of web content infrastructure (as most of the web contents are served by the US and Europe) are also the causes responsible for poor web performance in Africa. Vesuna *et al.* [14] ran a controlled experiment on 400 webpages (a subset of Alexa top 2K websites) to show that caching that

improve PLT by 34% in desktop pages, and 13% in the mobile pages. Liu *et al.* [15] studied the performance of HTTP/2 and HTTPS by cloning 200 Alexa top websites into a local server. The websites were accessed via HTTP/2-enabled proxy using Firefox and Android browsers to show that HTTP/2 could either decrease or increase the PLT under various network conditions and page characteristics.

There have also been work [32], [33] studying the impact of web object interdependencies on the performance of the webpages. Butkiewicz *et al.* [1] studied the impact of webpage complexity on the download performance of websites. They showed that the number and size of objects in the webpage are factors that affect the PLT. Studies [17], [34]–[36] exploring scenarios that affect web performance and QoE have also been conducted. For instance, Naylor *et al.* [34] showed the impact of using the secured version of HTTP on PLT.

Improving web performance – Wang *et al.* [37] introduced a micro-caching technique to improve web performance that caches web content at a finer granularity. They went further and developed *Shandian* [12], a tool that can restructure the webpage load process to consequently half the PLT of webpages. Butkiewicz *et al.* [38] designed a tool that prioritizes the most relevant web content for user's preference to improve the web performance and QoE in mobile devices. Kelton *et al.* [39] proposed a system that uses HTTP/2 push to optimize the user perceived PLT by prioritizing web objects that are visually interesting for majority of the users. Li *et al.* [40] implemented a framework that reorders web objects to reduce the Above the Fold (ATF) time, the time required to show the contents in above-the-fold area of the webpage. They showed that reordering objects using this framework reduced the ATF time especially for complex websites.

In order to reduce web latency, new application and transport protocols have lately been proposed and being adopted. For instance, Zhou *et al.* [41] designed a new naming and transport protocol which reduces latency by shortcutting the DNS request and removing TCP's three way handshake. Google has proposed QUIC [7], a multiplexed, low-latency transport protocol to improve the performance of HTTPS traffic. Biswal *et al.* [42] showed that QUIC improves PLT in poor network conditions, but it does not provide significant improvements when the webpage contains many small objects. Zimmermann *et al.* [43] studied how the HTTP/2 [8] server-push improves the perceived PLT. They showed that server-push does not always yield a better perceived performance, but can also degrades the performance. As such, a proper push configuration is necessary to improve the end user experience.

Web QoE – Different confounding factors [44] that emanate from the human, context, system and the content perspective influences the web QoE. The waiting time to get the content is one of the paramount factor that impacts the browsing QoE. The interaction design and ease-of-use [45] also has an impact on the overall QoE. For instance, Hoßfeld *et al.* [46] show that memory effect (psychological factor of past web browsing experience) is a dominant influencing factor for the user QoE. The user perceived PLT is also different from the PLT measured at the network level since the rendering machine requires certain amount of time to display the content on the browser. Towards this end, Egger *et al.* [47] performed a subjective study, where users were asked to browse a set of webpages and report the task completion time, as their subjective PLT. It was found that the task completion time is also a key influencing factor for web QoE [48]. Sackl *et al.* [49] studied the impact of network outage on web QoE. Their subjective study revealed that a short outage (four seconds or shorter) highly influences the user annoyance level, whereby in web browsing sesssions, users can tolerate outages up to eight seconds. This shows that the impact of network outage in QoE is application dependent. Bocchi *et al.* [2] introduced ByteIndex and ObjectIndex, metrics that can better approximate the actual user QoE. They showed that the proposed metrics are highly correlated with Google's SpeedIndex, and offer advantages in computational complexity. These metrics consider the time taken to download all the objects in the webpages, while our proposed metric (rendering time) considers time taken to show the contents in the above-the-fold area. Albeit, computing the rendering time is computationally expensive, our proposed system also allows measuring the web QoE in the wild.

Tools – W3C web performance working group standardizes the web performance measurements and APIs in web browsers. The group has specified several web performance measurement APIs. Among these, Navigation Timing, Resource Timing and User Timing APIs help to measure the performance of a website on a real world. Page Visibility, Efficient Script Yielding and Display Painting Notification APIs also provide basic information about the rendering state of the webpage and facilitate developers to write resource (CPU and power) efficient web applications. For instance, the Page Visibility API enables a developer to determine the current visibility of the page. These APIs allow measurement of the performance of websites in the browser, but they cannot substitute the visual perception of the end user.

Tools to measure and monitor network and web performance have also been developed. These include browser-based [50], [51], headless web clients [9], [52], host measurements annotated with user perception [53] and HTTP-based application replaying [54] tools. For instance, Hora *et al.* [55] implemented (as a Google Chrome extension) a lightweight tool to approximate the ATF time using the browser heuristics. Albeit, *WePR* is not a lightweight solution to approximate the rendering time, it considers all objects that appear in the above-the-fold area, unlike implementations [55] that consider only images and skip media objects such as Adobe flash. Most of the aforementioned tools are not suitable for large scale deployment. For instance, they require user interaction to run the experiments. They also do not consider the most critical metrics for approximating the user experience. *WePR* closes these gaps by better approximating the user browsing experience and not requiring user interaction for executing the experiment at scale.

III. System Design and Methodology

We present the methodology and the system we developed to measure the web latency and the visual rendering time.

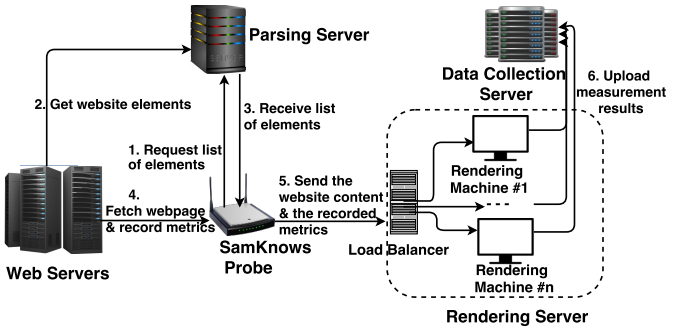


Fig. 2: The distributed architecture of WePR. The SamKnows Probes are located in the customers premises, the parsing and rendering servers are located in data centers.

While different tools are available (see § II) to measure the web browsing performance, most of them are either not scalable or do not cover a wide-range of metrics at different layers. Hence, our goal is to develop a scalable measurement tool for assessing web performance with the network QoS, web complexity and the application level metrics. The metrics include DNS lookup time, TTFB, the number and the size of objects, PLT and the visual rendering time. To better represent and understand the real web user experience, such tools and systems need to collect an actual browsing data from different vantage points located at the gateway of the subscribers' premises. There are measurement platforms (e.g., the SamKnows measurement infrastructure) that enable users to deploy tests and collect measurement results from multiple vantage points. The measurement devices used in these platforms are often lightweight. Due to this constraint, they cannot run a full web browser engine to compute the application-level metrics such as the rendering time. However, it is possible to offload this task and calculate the application level metrics by running a web browser engine on a different machine (e.g., in a data center).

WePR – is a web latency and rendering measurement system composed of different components. Fig. 2 illustrates the overall distributed architecture of *WePR*. We offload each component to different devices and locations. The parsing server parses the home page of a website to extract the URIs of all the objects that make up the webpage. It also executes scripts to get the dynamic objects of a webpage. The SamKnows probes are measurement devices that run the *Webget* and *WebPerf* tests to measure the download performance of a webpage and push the results to the rendering server or to the data collection server. The rendering server computes the rendering time of a website. The data collection server stores the measurement results collected from the rendering server. The flow of operation is as follows. The SamKnows probes run the test and initiate the measurement by requesting the list of URLs for a website from the parsing server (#1 in Fig. 2). The parsing server then fetches the given website, extracts the URLs of the objects (#2) and sends them to the probes (#3). The SamKnows probes then download the objects, measure the performance (#4), and push the results and the downloaded objects to the rendering server (#5), respectively. The load balancer then distributes the incoming rendering request to the available rendering machines. After computing the rendering time, the rendering machines push the results (#6) to the data collection servers.

We choose a distributed architecture for two main reasons. First, the SamKnows probes at the customer premises have limited resources. They cannot run a browser rendering engine that executes scripts and heavy computations like calculating the rendering time. As such, we designed the system that offloads the heavy computations to the parsing server (see § III-B) and the rendering server (see § III-C). Second, assuming the measurement probes are powerful machines with sufficient resources (e.g., a regular PC if crowdsourcing measurements are performed instead) to execute scripts and run resource-intensive computations, one would not have privileges to alter the user's machine settings or install the necessary software (e.g, *ImageMagick*, *FFmpeg*) that are necessary to calculate the rendering time. Therefore, offloading the different functions into different components is essential to calculate the rendering time. We discuss the main components of *WePR*, and describe the metrics that each component measures in the process in more details.

A. SamKnows Probe

The SamKnows probes are OpenWrt based embedded measurement devices running Linux, which execute the web performance test software – *Webget* and *WebPerf*.

***Webget* –** is a software that records the DNS lookup time, TTFB, the download time, the number and the size of each static object that makes up the website. *Webget* runs a maximum of eight concurrent connections, and up to eight parallel threads per domain. We chose this setting to match the behavior of the user-agents that we have used in the past. We have not updated the number of the parallel threads so far because it would cause a significant change in the *Webget* result. Note, the goal is to keep the parameters consistent across the longitudinal (several years) data collection to avoid fragmenting the data into smaller samples by not tweaking the parameter space over time. *Webget* does not execute scripts. It does not download nor take into account the dynamic objects, which are common in modern websites.

***WebPerf* –** In order to capture the detailed TCP and HTTP statistics, and take into account both static and dynamic objects of the webpage, we extended *Webget*. The extension, *WebPerf* [11], downloads each object of the webpage based on the list of URLs it received from the parsing server and pushes them to the rendering server. These objects are necessary for recreating the webpage in order to calculate the rendering time. In the real world, different browsers open up to six concurrent connections per domain and up to tens of parallel connections to optimize the performance. As such, *WebPerf* has an option to configure the maximum number of parallel connections. In our measurement, we set it to open up to 20 simultaneous connections across all domains and up to three parallel threads per domain. *WebPerf* also has a feature of reusing a single TCP connection to send multiple HTTP requests.

WebPerf is written is C and can be cross-compiled and deployed in any Unix-like platform. *WebPerf* measures the

web QoS, web complexity metrics and the CDN used by the website to deliver the contents. It downloads all the objects that make up a webpage including those generated by JavaScript and records a set of metrics for each object. The metrics include the DNS lookup time, the number of messages exchanged during DNS lookup, the time to establish TCP connections, the time to perform the TLS handshake, HTTP header size, the number of HTTP redirects, the time elapsed due to the HTTP redirects, the number and size of the objects and the download time.

The *WebPerf* test takes a URL of a website and sends a request to the parsing server to get the list of URLs of the objects that make up the website. Once it gets the URL of each object of the website, it downloads the objects and in the process measures the aforementioned metrics. We developed our library for handling the HTTP downloads and extract HTTP related information. The library was specifically designed to allow developers to intercept request processing at key points in the HTTP download process. *WebPerf* uses the hooks that the library provides for recording the timestamps, to extract header information, and to save the received data.

WebPerf overrides the default DNS resolver of the host system and utilizes our DNS client for name resolution. Albeit several DNS tools exist for Linux, they do not provide a sufficient level of detail about the DNS resolution process. The DNS resolvers also vary between operating systems (OSs) by adding different optimizations such as client-side caching. Thus, we employ our DNS client to measure the performance without the optimizations available in the OSs. *WePR* uses our own DNS API for the following reason. We calculate the rendering time at a centralized server which runs a different OS than the probes and may use a different DNS optimization techniques than that of the probes. Thus, we want to make sure that both probes and the rendering server use the same DNS optimizations so that the rendering time is calculated in the same settings as probes. Our DNS client measures the delay caused by the DNS resolution when establishing a new connection. It also records details about the resolution process and the final query result. Moreover, the DNS client has a client-side caching capability to optimize the DNS resolution. During a webpage download, DNS resolution is performed once per domain, and only one of the objects per domain will have the metrics of the DNS resolution. All other objects from the same domain will include a reference to the object which triggered a DNS resolution. Therefore, the DNS metrics are tied to the web objects.

B. Parsing Server

The parsing server parses the homepage of a website and lists the URIs of each object (*i.e.,* both static and dynamic) that make up the website. It takes the URL of the website and parses the DOM structure of the homepage and also executes JavaScript codes by using *PhantomJS*. After parsing the DOM and executing the scripts, it records the URIs of the target objects (both static and dynamic) that need to be fetched to render the webpage. Once it extracts the URIs, it sends them to the SamKnows probes so that the probes start a performance measurement and download each object. The parsing server can also be deployed within the probes if they can run a browser rendering engine.

C. Rendering Server

The rendering server runs web rendering test that calculates the rendering time. The rendering time is the time taken by the webpage content in the above-the-fold area (*i.e.,* the portion of the webpage that is visible without scrolling) to reach a final stable state. It is the closest metric to approximate the user-perceived page load time [56]. The PLT (the time to fire the `onLoad` event) is still the commonly used metric to estimate the QoE for web browsing. Nevertheless, recent work proposed other metrics such as ATF time [40], [55] and SpeedIndex to better approximate the end-user browsing experience. Other work [55] call this metric ATF time, but both are equivalent measures. We approximate these metrics using our rendering test.

The rendering server comprises of a load balancer and one or more rendering machines. Our objective is to build a measurement system that can handle multiple measurements simultaneously. Since the rendering time computation is not a lightweight task, it is paramount to use multiple rendering machines. Also, these rendering machines need to work in a synchronized manner so as to effectively process all the requests from the SamKnows probes. Hence, a load balancer is necessary to process the incoming rendering requests and assign them to the available rendering machines. The load balancer receives rendering requests from the SamKnows probes and distributes them to the available rendering machines using a round-robin mechanism. The load balancer is implemented using *HAProxy*, which provides load balancing and proxying for TCP– and HTTP–based applications.

The rendering machines execute two different applications: (a) the playback module, and (b) the rendering manager module. The playback module emulates a DNS server and an HTTP server. It responds to DNS and HTTP requests. The playback module throttles down the response time and transmission rate to mimic the network delay observed at the probes based on the measurements conducted by the *WebPerf* test. Throttling the response according to *WebPerf* measurement results tries to ensure that the rendering performed at the rendering server follows a comparable network QoS as observed by the probes. The playback module gets the inputs for the responses from the locally stored objects. Those objects are downloaded at the probes and pushed to the rendering server together with the *WebPerf* measurement results.

The rendering manager module computes the rendering time of the website as it would have been seen by a user with the same network conditions as the probes. Once the rendering manager module receives a rendering request from the probes, it runs a (virtual) web browser and issues HTTP `GET` request to the given URL. The browser rendering engine renders the website based on the response from the playback module. Thus, the website is recreated and displayed on the screen. Additionally, the rendering manager records a video (10 frames per second) of the browsing session for 15 seconds.

It breaks down the video at every 100 ms into a series of bitmap images. The rendering manager computes the progress of a webpage download by calculating the pixel changes on the browser window in 100 ms intervals (the normal human visual system perceives changes between 150 ms and 200 ms and we believe that a 100 ms interval is adequate). The rendering time is calculated by looking at the pixel changes in the above-the-fold area of the website. If no pixel change has been observed in 30 consecutive screenshots (*i.e.,* no rendering event has happened for 3 seconds), then we declare that the website has stabilized and we take this as the point where the webpage is fully rendered. In some cases, the pixel change continues and the webpage may not stabilize within 15 seconds. This might be due to persistently changing contents such as auto-play enabled video advertisements in the webpage, or because the website takes too long to load or to render the contents.

Nevertheless, determining whether or not a webpage is fully rendered is challenging. The challenge gets worse if the webpage contains frequently changing contents like animating images or auto-play enabled video advertisements. Our current solution is based on the assumption that many of the animating contents in the webpages change less frequently than every three seconds. In our recent study [57] performed in cellular networks, we set three, ten, and fourteen seconds threshold for determining when a website is stabilized and approximated the ATF time. At the same time we used the browser timing API to approximate the ATF time. The result showed that the ATF time computed using the browser timing API is shorter than the rendering time with the three seconds threshold. This shows that three seconds threshold is sufficient to declare when the website has stabilized. As this work focuses on fixed-line networks, we believe that the users expect the page to load within a second and three second threshold is sufficient. Therefore, once a webpage has started loading and no pixel change is observed for three seconds, the webpage is considered to be completely rendered. As such, the rendering time of a webpage is the duration between the time at which the user starts navigating the webpage and the time at which the last rendering event is observed. However, the three second threshold does not consider webpages that have auto-play enabled video contents. This is because videos usually change with tens of frames per second, which makes it harder to set a threshold for declaring when the webpage has stabilized.

IV. Benchmarking and Validation

The rendering server emulates fetching a website content and rendering it on a web browser. For the purpose of parallel analysis of multiple measurements, the rendering server should not require an actual display device. For instance, a user may want to run the rendering server in a virtual cloud infrastructure. As such, we need to start the browsing and capture a video of the browsing session in a virtual screen. We used *Xvfb*, a virtual display server which performs all graphics operations in memory without showing a screen output. We also used *Selenium*, a browser automation tool that can browse websites and record the browsing session in a virtual screen. We accessed the Google Chrome rendering engine using *Chromedriver*. Selenium provides APIs that help to integrate browsing a website in a virtual screen and at the same time recording the virtual screen.

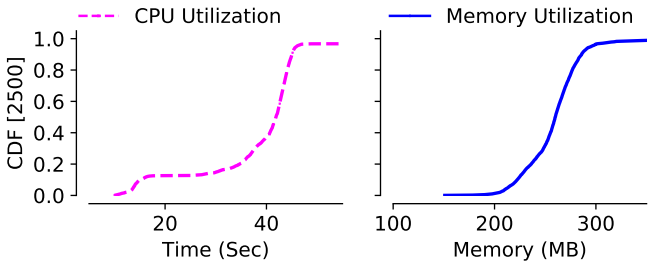


Fig. 3: The CPU and memory utilization by a single rendering process. In the 99^{th} percentile the rendering server consumes up to 45 sec. CPU time and 290MB memory to finish a rendering task.

We measured the CPU and the memory usage of the rendering server in our measurement setup. The specification of the rendering server is – Ubuntu 14.04.2 LTS, quad-core Intel Xeon(R) processor (2.65GHz each), and 4GB RAM. Fig. 3 shows the CPU and the memory consumption by a single rendering process. Approximately 99% of the web rendering processes took 45 seconds of CPU time and 290MB of memory. The most resource consuming part of the rendering server operations are the screen recording and the image processing to calculate the rendering time (see § III-C). Improving the screen recording method would enhance the performance of the rendering server. Investigating efficient screen capturing methods is left for a future work.

We validated our approach and the results by measuring the rendering time of ten non-HTTPS websites in two cases. First, we fetched the contents of the websites using *WebPerf* and rendered using Mozilla Firefox (version 46.0). Then immediately we browse the same website using Mozilla Firefox from the same laptop. We calculate the rendering time for both cases. The laptop has 8GB RAM, quad-core Intel processor (2.30GHz each), and Ubuntu 16.04.2 LTS operating system. The laptop is connected to a university WiFi network. The browser cache is cleared before fetching the web contents. To retrieve similar contents of the websites, the same User-Agent string has been used for both *WebPerf* and Firefox. Moreover, we set Firefox and *WebPerf* to use the same number of maximum parallel connections to hosts and concurrent threads per server. We ran this experiment 1500 times.

Fig. 4 shows the similar rendering time of the websites using both tools. The rendering time of the websites, when *WebPerf* and Firefox fetch the content, shows a positive correlation (Pearson correlation coefficient of 0.4). We calculated the delta of the rendering time of the websites (when the content is fetched either by Firefox and *WebPerf*). The difference in the rendering time of the website is negligible in 50% of the cases, implying the rendering time of the websites is equal irrespective of the tool used to fetch the contents. In 25% of the times, the rendering time is faster (by 400 ms) when Firefox fetches the content. In 25% of the cases, the rendering time is faster (by 500 ms) when *WebPerf* is used to download the content. The reasons for this difference could be the load on the web server and the latency difference

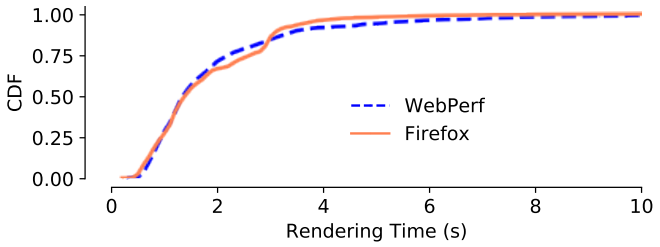


Fig. 4: The rendering time of ten websites when the contents are fetched by the Firefox browser and by WebPerf. The websites show similar rendering behavior regardless of the tool (WebPerf or Firefox) used to fetch the contents.

due to the variation on the paths while fetching the objects using Firefox and *WebPerf*. That is, at different times the web server could have different load and the delays in the response vary depending on the server load. Furthermore, due to network routing changes at different times, Firefox and *WebPerf* software may traverse through different paths while fetching the contents of the websites, which may add variation in the latency. Note, we observed that the rendering time is shorter when the website contains encrypted contents and downloaded by *WebPerf*. This is due to the fact that the current implementation of the playback server does not support secure connections (see § IX) and the browser does not get response for secure object requests. As a result, pixel changes in the above the fold area are small and the rendering time becomes short. Consequently, for our validation, we have only chosen websites that have non-secured contents.

We also manually inspected the webpage download behavior for cases that have a short rendering time. We witnessed the website appearance looks normal even if the rendering time is ∼500 ms. We observed this behavior in both cases when the websites are fetched using either Firefox or *WebPerf*. That is, we did not see any missing content in the above-the-fold area of the website. In fact, while browsing the web in a real world, the whole content of the webpage may instantly appear, a broken page may display in the browser, or the browser window may remain blank until further reloading is performed. We verified that all these browsing behaviors exist in our measurement system *WePR* as well.

V. Deployment and Data Collection

We cross-compiled the *Webget* test and deployed it on 182 SamKnows probes distributed globally. These probes are located in more than 70 origin ASes, covering 28 different countries including South Africa and South Asian countries (see Fig. 1). The test measures the performance of specific webpage of three most popular websites:

YouTube – `www.youtube.com`
Facebook – `www.facebook.com/policies`
Google – `www.google.com/mobile`

These webpages are measured every hour. Going forward, we refer to these webpages as YouTube, Facebook, and Google, respectively. We considered popularity, content consistency and the size of the webpages as a criteria of choice. We ensure these webpages do not require user interaction

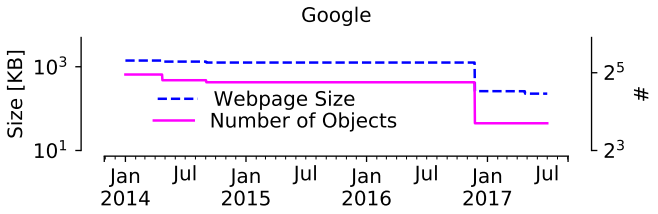


Fig. 5: Time series of the daily median of the size and the number of the objects of Google.

to show meaningful content. Since the goal of the study is to understand longitudinal aspect of web latency, we picked webpages that exhibit relatively consistent static content and change infrequently (Fig. 6) over the duration of the measurement study. Moreover, as our measurements run from volunteers home and repeat every hour, we also took the size of the webpage into account to ensure that our measurement traffic does not overwhelm the users' home network. The primary objective of this study is not to compare download performance of webpages. Instead, the aim is to understand what factors contribute to web performance across ISPs and geographical regions. In § VI, we present key observations from the analysis of 3.5 years long (Jan 2014 to Jul 2017) dataset collected from this measurement study.

The rendering server runs on a Virtual Machine (VM) which is part of a cluster of servers at Aalto University. In our deployment we used three VMs as a rendering machine and one VM as a load balancer. Each VM runs Linux (Ubuntu 14.04.2 LTS, Intel Xeon(R) 2.65GHz quad-core CPUs, and 4 GB RAM). We used the Google Chrome browser (Version 46.0) and Chromedriver (version 2.30) for rendering the websites. We deployed the *WebPerf* test for nine months (Mar 2015 – Dec 2015) on 65 SamKnows probes located in different origin ASes mostly in Europe. Each probe measures four websites:

- `www.bbc.com`
- `www.ebay.com`
- `www.sina.com.cn`
- `www.reddit.com`

These websites are measured every four hours. We chose these websites to cover a range of popular categories and also take the deployment location of the probes into consideration. In § VII, we present the analysis of the rendering performance of these websites.

VI. A Longitudinal View of Web Latency

We present the analysis of the 3.5 years long dataset collected using the *Webget* test focusing on web latency. We begin by presenting the web latency of the three popular web services, and then perform a temporal analysis followed dissecting web latency across regions and service providers.

A. Web latency

We begin by presenting the evolution of complexity of these webpages. Fig. 5 shows the daily median of the total size and the number of objects of Google. Both the total size and number of objects of Google has reduced since Jan 2017.

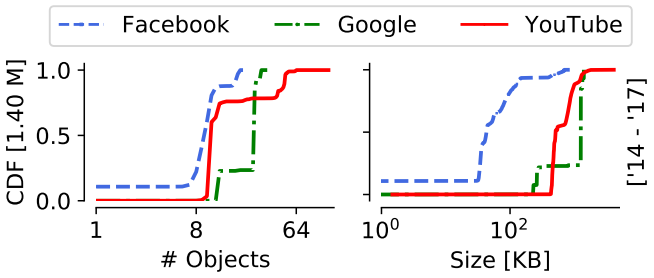


Fig. 6: The CDF of the size and the number of web objects.

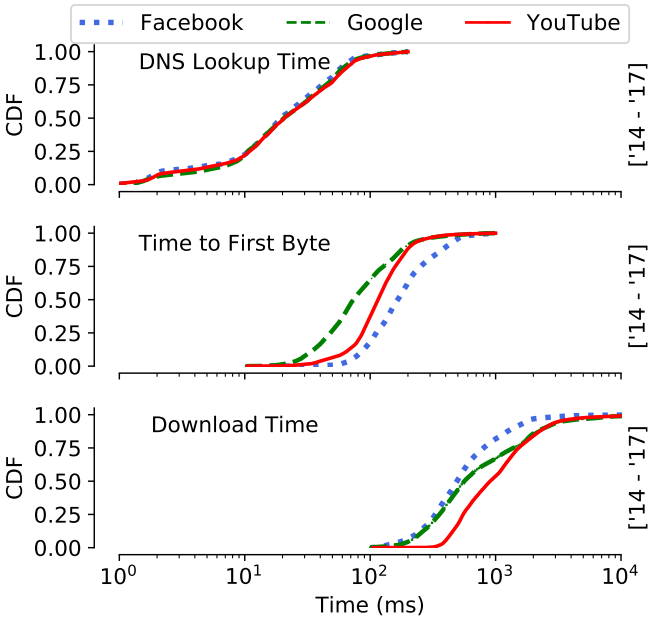


Fig. 7: CDF of the performance of the three popular websites in terms of different metrics.

Fig. 6 shows the distribution of the size and the number of objects of all the three webpages. As it can be seen, Facebook has the smallest size and a lower number of objects throughout the measurement period. Moreover, in 80% of the cases, Google has the highest number of objects and the largest size of objects. YouTube showed different behavior over time. For instance, since mid of 2015 the number of objects of YouTube shrunk by about 70% compared to the previous year.

The webpages that we measure have multiple objects possibly hosted at different server locations. We measured the DNS lookup time required to resolve the URL of each object and also the TTFB of each object. We consider the webpage's DNS lookup time and TTFB as the average of the DNS lookup time and the average of TTFB of the objects within the webpage, respectively. Fig. 7 shows the CDF of the average DNS lookup time, the average TTFB, and the download time of the three webpages. The results show that in 55% of the measurements, the three websites have relatively similar behavior in terms of DNS lookup time. However, 45% of the cases, the DNS lookup time of Facebook is 10% shorter than Google and YouTube. The reason for this difference is that the clients need to make a maximum of two domain name lookups to get the Facebook's

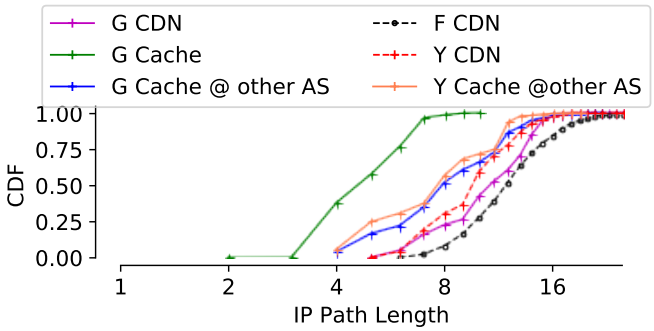


Fig. 8: The IP path length from 100 SamKnows probes towards Facebook (F), Google (G) and YouTube (Y). In 41% of the cases, the probes get Google caches from the ISP.

policy page. Instead, in the case of Google and YouTube, the clients need to resolve two to four and one to three DNS names, respectively.

We also observed the average TTFB of the webpages. The results show that there is difference in the average TTFB of the webpages, even within Google and YouTube which are supposed to share the same CDN infrastructure. For instance, 50% of the measurements show that Google has the shortest average TTFB than the other two (i.e., it has about 55% and 117% improvement compared to YouTube and Facebook, respectively). The reasons for this variation could be the presence of caches, and difference in the IP path length between the clients (probes) and the server (content replica).

Another observation is that Facebook takes longer to get the first byte of the objects although the domain name resolution takes the same amount of time as the other two webpages. We further investigated the possible causes for this longer average TTFB of Facebook, including the number of servers that the clients contact to fetch the objects of the webpage [1] and the path length to Facebook's CDN [58]. We observed that the objects embedded in the Facebook policy page are located in a maximum of two web servers (in the median case, in a single server). As such, the longer average TTFB of Facebook (compared to the other two) is not due to the number of servers that the clients need to contact to fetch the web objects. Chui *et al.* [58] have previously shown using RIPE Atlas probes [59] that the path length to Facebook's CDN is longer compared to other popular content providers such as Google. For instance, 60% of the Facebook contents are reachable in at least 2 AS hops, while more than 55% and 80% of Google's CDN can be reached with just 1 AS hop and 2 AS hops, respectively.

We performed `traceroute` to measure the IP path lengths from 100 SamKnows probes (a subset of 182 probes used in the *Webget* test) towards Facebook, Google and YouTube services. Fig. 8 shows that Facebook has a longer IP path length compared to Google and YouTube. In our `traceroute` measurement, when the probes' AS number is the same as the destination AS number, it is identified as a cache at the ISP. When the destination AS number is not the same as that of the probes or Google, we consider them as a cache at other AS. Applying this heuristic, we did not find any Facebook cache at Internet Service Providers (ISPs). In 25% and 24% of the cases, we witnessed Google and YouTube caches at

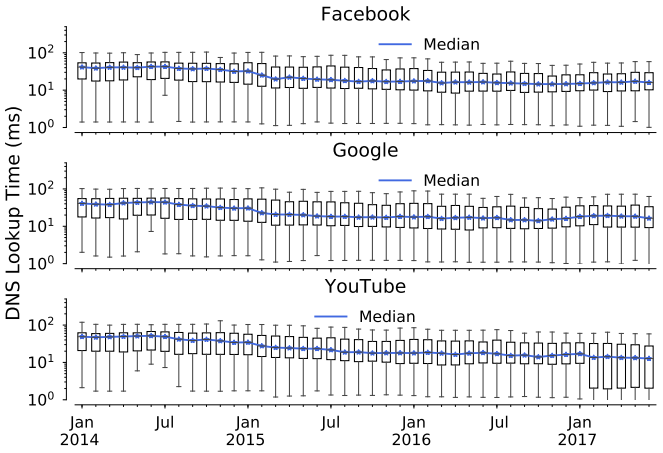


Fig. 9: The monthly median of the DNS lookup time of webpages over time. The DNS lookup time of webpages has improved over time.

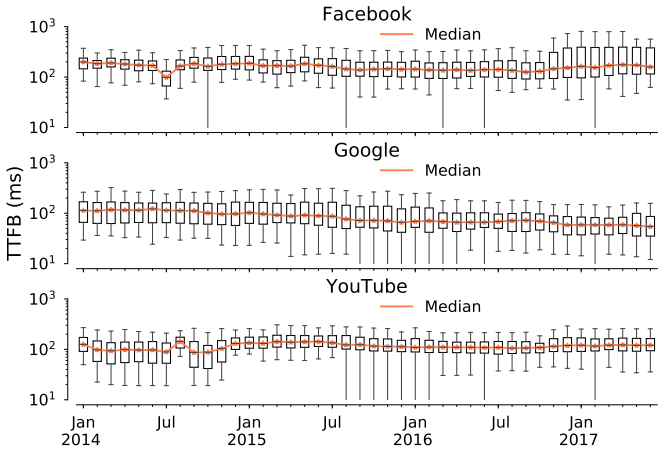


Fig. 10: The monthly median of the TTFB of webpages over time.

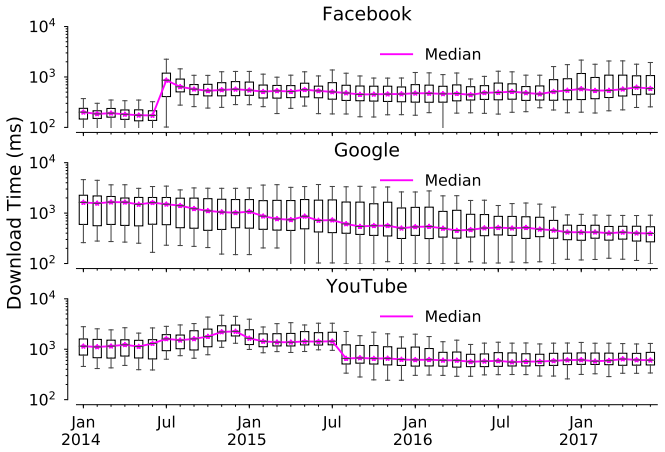


Fig. 11: The monthly median of the download time of the webpages over time.

the ISPs, respectively. Moreover, in 16% and 10% of the times, the probes reach Google and YouTube caches at other ASes, respectively. Therefore, we can conclude that the longer average TTFB of Facebook is due to a longer path length of the Facebook CDN and absence of caches. The reason for the spike in the average TTFB of YouTube (compared to Google) could be the due to the absence of caches at the ISPs network. That is, in 66% of the times, the probes need to contact the YouTube/Google CDN to get YouTube's landing page. Nonetheless, in the case of Google, the probes go to the Google CDN only 59% of the times.

In terms of the webpage download times, Facebook loads faster than the other two webpages. While, at the 90^{th} percentile, the download time of Google is faster (62% at the median) than YouTube. Note, the goal of this paper is not to compare download times of different websites due to the different static and dynamic nature of the websites. However, we seek to understand the impact of the number and the size of the objects on the download times. Fig. 6 shows that Facebook has fewer and smaller of objects, and indeed Facebook shows better performance in terms of the download time. Butkiewicz *et al.* [1] have previously studied the impact of number and the size of the objects on the download performance of a webpage. They highlight that such complexity metrics do not directly affect the download times of the websites. Our measurements confirm this assertion that the number and the size of objects are not the only determinant factors affecting the download times. For instance, we observe that even though YouTube has the smaller number and size of objects than Google, the download time of Google is shorter than that of YouTube.

We also went further to understand the relationship between the TTFB and download time for the webpages. We computed the Pearson correlation coefficient between these two metrics for each webpage. Google shows a stronger positive correlation (0.64). Instead, Facebook shows a weaker correlation (0.34) and YouTube shows a correlation of 0.46. The strength of the relation between the TTFB and the download time of the webpage could be attributed to the presence of caches. That is, the average TTFB is a function of distance and the distribution of objects into different locations. While, the download time is a function of distance, the number and size of objects, and the geographical distribution of objects into different locations. As such, in situations where caches are available, both the TTFB and the download time decreases symmetrically. While, in situations where there is no cache available, the TTFB increases, but the download time changes depending on the number and size of objects. In our measurement, Google has higher number of caches in the ISP's network than YouTube and Facebook. As such, the average TTFB and download time of Google show a stronger correlation. On the other hand, Facebook does not have caches inside the ISP's network and the correlation between the average TTFB and download time of Facebook is weak.

B. Temporal view of web latency

Fig. 9 shows the evolution of the monthly median of the DNS lookup time for the three websites. It can be seen that the DNS lookup time for all websites has improved over time. For instance, from 2014 to 2016, in the median case the average DNS lookup time of Facebook, Google and YouTube have improved by 59%, 56% and 61%, respectively. This improved DNS performance could largely be a function of

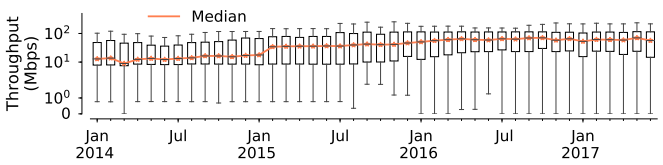


Fig. 12: The monthly median of the achieved throughput as witnessed by the probes over time.

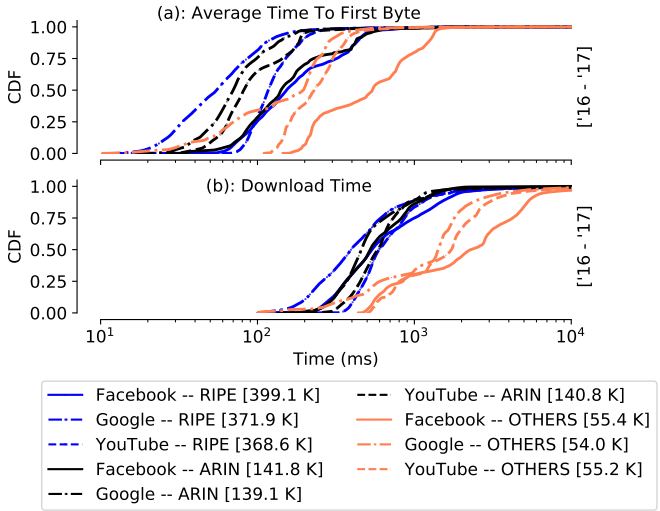


Fig. 13: The CDF of the average TTFB and download time of the webpages across different origin ASes grouped by region. The average TTFB of Facebook is longer in the rest of the world than Europe and North America.

the reduced last-mile latency [9], [60] due to improved ISP infrastructure [61] over the years. Fig. 10 and Fig. 11 show the evolution of the performance of the webpages in terms of TTFB and the download time, respectively. The results show that, in the median case, the average TTFB and the download time of Google has improved over time. Meanwhile, the other two websites do not show a clear change over time. Nonetheless, YouTube shows different behaviors over time both in terms of TTFB and download time. That is, until mid of 2015, YouTube shows both an increase and decrease, and around July 2015 there was a dramatic improvement in the download time. Since then, there is not much change in the download time of YouTube. In the middle of 2015, YouTube was known to have performed refactoring of its homepage which led to reduced number of static objects by a factor of four. As such, the total size of the static objects in the landing page also decreased significantly. We speculate this to be the reason for the download time improvement of YouTube over time. Meanwhile, we observe that the download time of Facebook also significantly increased in the middle of 2014. A reason for this change could be the introduction of Facebook privacy basics that updated the terms and policies. As a result, the Facebook policy page added more number and a larger size of objects compared to the situation until July 2014, whereby the Facebook policy page only had a single object and could be the reason for the different in download times.

The web browsing performance can also be affected by

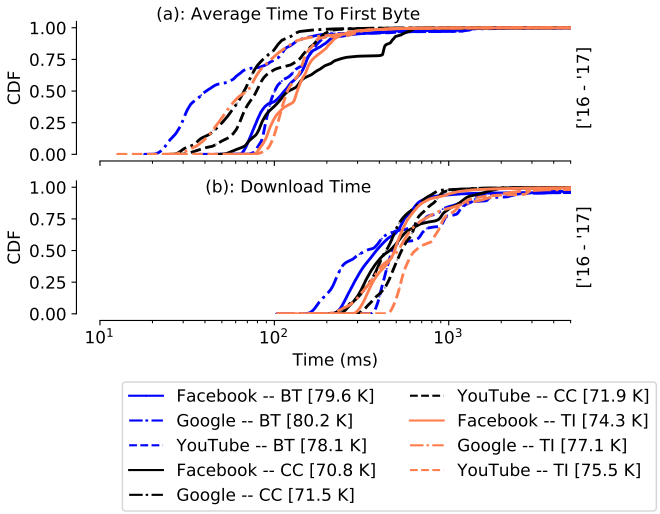


Fig. 14: The CDF of the average TTFB and download time of the three webpages over selected origin ASes.

the broadband speed of the client [62]. In order to study the evolution of (if any) broadband speeds within our dataset and to verify any correlation between the achieved throughput and the web latency, we also performed throughput measurements on the SamKnows probes. Fig. 12 shows the time series of the achieved throughput improvement as seen by the probes over the measurement period. We have seen in Fig. 10 and Fig. 11 that the download times of Google has improved over time, while the other two websites (i.e., Facebook and YouTube) do not exhibit improvements in latency over time, even though the achieved throughput has improved over the years. As such, improvements in broadband speeds do not necessarily lead to a better browsing experience.

C. Web latency by region and service provider

We dissected the web latency distributions of an year-long subset of the dataset (08/2016 - 07/2017) by region and origin AS of the probes. For instance, Fig. 13 shows the distribution of the average TTFB and the download time of the webpages from different origin ASes grouped by region (*i.e.,* Europe, Middle East and parts of central Asia (RIPE), North America (ARIN), and the rest of the world (OTHERS)). We observe that all the webpages have different download performance across regions. The probes hosted in the regions other than RIPE and ARIN show worse performance for all webpages. For instance, the longer average TTFB and download time of Facebook in these regions is most likely attributed to the longer AS path length from the probes to Facebook's CDN and absence of caches as shown in Fig. 8.

Similarly, we also analysed the web latency towards webpages as observed from selected ISPs that host more than 10 SamKnows probes. Since, our goal was larger varied distribution of probes, most of the origin ASes host up to five probes. While, three origin ASes, i.e., Telecom Italia (TI), British Telecom (BT) and Comcast (CC) host 14, 13, and 13 probes, respectively which we focus in this analysis. Fig. 14 shows the CDF of the average TTFB and the download time of the webpages on these selected ISPs. As can be seen,

TABLE I: Average TTFB and download time [in millisecond] of the webpages across different regions and ISPs.

WEBPAGE [Metrics]		RIPE		ARIN		Others		BT		CC		TI	
		Med.	Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.	Mean
Facebook	TTFB	153	222	141	209	540	626	114	152	121	202	135	149
	Download Time	536	962	523	654	2286	3244	385	1182	459	641	489	563
Google	TTFB	55	78	70	93	191	177	41	108	67	75	70	89
	Download Time	394	808	460	645	1407	1610	335	1633	438	500	511	786
YouTube	TTFB	117	142	82	118	230	253	101	158	79	101	119	140
	Download Time	595	958	585	743	1722	2068	475	1571	537	608	642	916

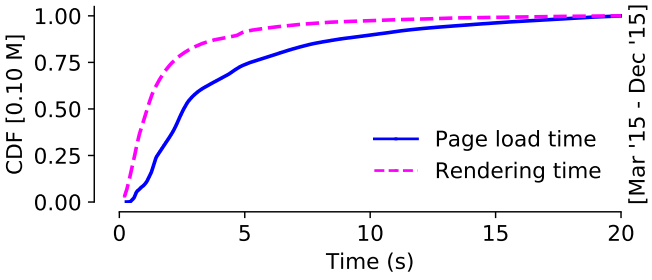


Fig. 15: The PLT and rendering time of the websites.

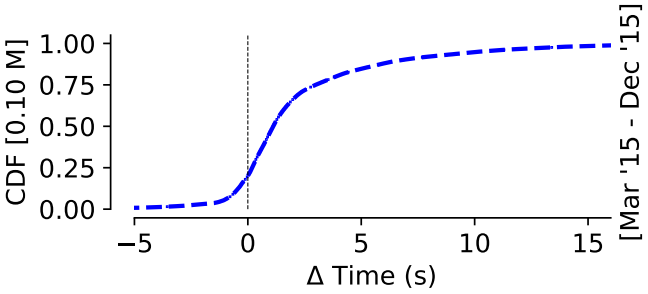
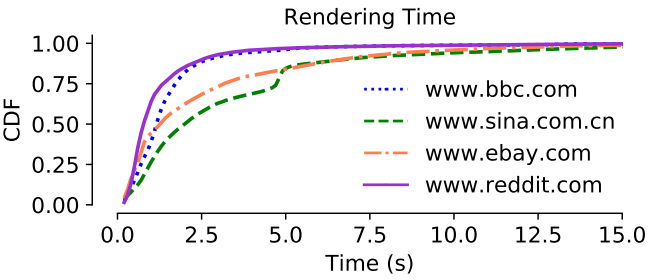


Fig. 16: The CDF of difference of PLT and rendering time of websites.



	Med.	Mean
www.ebay.com	1.4	2.7
www.reddit.com	0.8	1.4
www.sina.com.cn	2.1	3.4
www.bbc.com	1.2	1.6

Fig. 17: The rendering time of the websites [in seconds].

Facebook has longer TTFB in ∼40% of the probes hosted in the CC network. The longer TTFB from some probes hosted in the CC likely is due to longer AS path lengths towards Facebook CDN. On the other hand, YouTube has slower download performance in 70% of the probes hosted in the TI network. Christian [63] in 2015 has shown that Google has less (27) cache instances in Italy compared to UK (67) and the USA (296) which could be one of the reasons for the degraded download performance of YouTube in the TI network. Table I illustrates the raw statistics of the average TTFB and the download time of the webpages across the studied regions and selected ISPs.

VII. Web Rendering Performance

We now present the rendering performance of four websites (§ III) that we measured using our *WePR* system. The websites have a median object count of 62, 176, 131, and 32, respectively. The rendering behavior in most of the websites is that after an initial download period most of the visible content is shown almost instantly. Additional web objects are also downloaded later, but they do not have a significant effect to change the visible part of the webpage. On the contrary, we observed that there are some cases in which the websites take longer to stabilize. That is, the pixels between consecutive screenshots continue to change. We reason this could be due to the page having animating images or auto-played advertisements, which change frequently. Fig. 15 shows that the PLT is longer than the rendering time for the measured websites. The PLT and rendering time show a positive correlation (*i.e.,* a Pearson correlation of 0.41). In order to verify whether the PLT is always longer than the rendering time, we calculated the difference between PLT and the rendering time of the websites. Fig. 16 shows that in about 20% of the cases the rendering time takes longer than the PLT. This shows that even though the web objects are downloaded, it may take longer to be displayed to the user. We speculate that such a response occurs in situations where the browser's rendering engine takes longer to process and to render the contents. Fig. 15 also exhibits a heavy tail in the download time, which we attribute to third party content that may not be optimized. The clients attempt to download these objects although the content in the above the fold area may not change much after the download.

We also witness that the download time of these websites varies substantially. The variation in the download time is due to factors such as the number and size of objects, and the distance to the content. Additionally, HTTP redirects during

the download process add to the download delay. For instance, we observe that `www.ebay.com` involves six HTTP redirects in the median case and contribute to large download times, whereas other websites involve at most a single HTTP redirect. As such, HTTP redirects increase the webpage download and rendering time and consequently degrade the user QoE.

Yet another observation is that the distance from the probes towards the web server affects the rendering time. For instance, `www.bbc.com` is hosted at Tadworth in the UK while `www.sina.com.cn` is hosted in Beijing, China. Fig. 17 shows that the rendering performance is affected by the content location relative to the client's geographical location. Given, majority of the probes used in the rendering analysis are located in Europe, the effect of the server location on the download and rendering time is visible, for instance, in the the heavy tail distribution for `www.sina.com.cn` curve, since this website is likely not optimized for Europe.

We also investigated the impact of throughput on the rendering performance of websites. We used a `speedtest` that runs on each probe. The web rendering test and the speed test run independently. We took the hourly averages of the results from both tests for each probe and investigated how the results correlate. The results shows that there is no direct correlation (*i.e.,* a Pearson correlation coefficient of -0.17) between the throughput and the rendering time of a website. Similarly, using a `ping` test from each probe (the tests run independently), we investigated the correlation between latency (*i.e.,* the round trip latency of a single packet) and the rendering time of a website. The result shows that the latency and the rendering time have a weak correlation (*i.e.,* a Pearson correlation coefficient of 0.11).

VIII. Implications for Service Management

Our results can help ISPs and content providers to better design and manage their infrastructure to improve the end-user QoE. For instance, our results show that cache deployments (see Fig. 8) inside an ISP's network help reduce latency and path length towards popular content. This quantification motivates the ISPs to provision caches within their network to improve their end-users QoE.

ISPs can use our *WePR* measurement system for actively and continuously monitoring their customers web browsing experience, in place of traditional tools such as `wget` that cannot measure dynamic objects. In order to monitor their end-users web QoE and understand the possible bottlenecks, an ISP can deploy our rendering server at its back-end infrastructure and deploy *WebPerf* at the customer premises of their subscriber base. Vantage point distribution within the network helps the ISP identify whether a quality degradation for a user is due to a problem in the shared part of the network, or unique to a single user line, or in the home network or a problem with the over-the-top service. Moreover, ISPs can complement the measurements that they already collect using our tools with a passive traffic measurement that they can capture at their backbone network not only to understand bottlenecks, but also to better manage web traffic towards popular CDNs. The ISP can also use our measurements for better capacity planning and network design.

IX. Limitations and Future Work

The SamKnows probes have been changing locations and it is hard to maintain the metadata for all the probes. As a result, we performed the ISP-based performance analysis only on a subset of the timeline for which we knew the location of the probes with certainity. Given the dataset does not have information about which content is served by a primary domain (or a third party domain), we are not able to analyse the impact of content delivered by third party domains on the download time. The content of a website may change depending on the location from where the request is made. In our measurement setup, the parsing server is located at one vantage point only, which skews our view of content delivery. We plan to explore the possiblity of distributing the parsing server geographically on Amazon instances. The current implementation of the playback module does not currently support HTTPS and WebSocket requests. The three seconds threshold, when calculating the rendering time, to observe whether the webpage has stabilized or not does not consider webpages that have auto-play enabled video contents. Such cases need special but not trivial consideration to set the threshold, and we plan to explore this case in future work. We also plan to study the relationship between the rendering time with different website stabilizing threshold and the other relevant QoE metrics such as ByteIndex [55]. Evaluating the rendering time for different screen sizes is yet another potential future work item. The rendering time computation consumes substantial computing resources to do the web rendering test on a large scale and the scalability aspects of *WePR* need more thorough investigation. Protocols such as HTTP/2 and QUIC have recently got standardized and the deployment is steadily increasing. The setup can be adapted to also initiate measurements over HTTP/2 [64] or QUIC.

X. Conclusion

We presented *WePR*, a web performance measurement system, composed of a set of tools that can be deployed at scale and can be used to capture different web performance metrics. Using *WePR*, we measured the web latency and the rendering time of selected websites. Together with describing the tools, system design and the associated metrics, we presented the analysis of a 3.5 years long dataset collected using *Webget*, one of the tools of our measurement system. Using this dataset we showed that the DNS resolution time for the three popular webpages has improved over time. The TTFB and the download time instead show a significant difference across webpages, that is, only Google shows improvement over the period of our measurement study. The webpages exhibit different download performance across geographical regions and ISP networks. We also measured the rendering time for selected webpages in addition to the web latency. The web rendering results show that in 80% of the cases the rendering time of the websites is faster than the download time. While, achieved throughput does not appear to have a direct correlation with the download and rendering time.

Reproducibility Considerations – To encourage reproducibility, the measurement system is open-sourced [22]. The

collected dataset and the Jupyter notebooks used for analysis are made publicly available [23]. Guidance on how to reproduce these results is provided and reproducers can contact the authors for further questions.

Acknowledgements – We thank Magnus Boye, Sam Crawford, Jamie Mason, Ermias Walelgne and Sosina Gashaw for providing valuable contributions to this manuscript. This work was partially funded by the EU Marie curie ITN program METRICS (grant no: 607728) and EU FP7 program Leone (grant no: 317647), and the EU Horizon 2020 research and innovation programme SSICLOPS (grant no: 644866).

REFERENCES

[1] M. Butkiewicz, H. V. Madhyastha, and V. Sekar, "Characterizing Web Page Complexity and Its Impact," ser. IEEE/ACM TON , vol. 22, no. 3, 2014. [Online]. Available: https://doi.org/10.1109/TNET.2013.2269999

[2] E. Bocchi, L. D. Cicco, and D. Rossi, "Measuring the Quality of Experience of Web users," ser. Internet-QoE Workshop, 2016. [Online]. Available: http://doi.acm.org/10.1145/3027947.3027949

[3] S. Egger, T. Hoßfeld, R. Schatz, and M. Fiedler, "Waiting times in Quality of Experience for Web based services," ser. IEEE QoMEX, 2012. [Online]. Available: https://doi.org/10.1109/QoMEX.2012.6263888

[4] E. Blanton and M. Allman, "On Making TCP more Robust to Packet Reordering," ser. ACM CCR, vol. 32, no. 1, 2002. [Online]. Available: http://doi.acm.org/10.1145/510726.510728

[5] T. Flach, N. Dukkipati, A. Terzis, B. Raghavan, N. Cardwell, Y. Cheng, A. Jain, S. Hao, E. Katz-Bassett, and R. Govindan, "Reducing Web Latency: the Virtue of Gentle Aggression," ser. ACM SIGCOMM, 2013. [Online]. Available: http://doi.acm.org/10.1145/2486001.2486014

[6] N. Dukkipati, T. Refice, Y. Cheng, J. Chu, T. Herbert, A. Agarwal, A. Jain, and N. Sutin, "An Argument for Increasing TCP's Initial Congestion Window," ser. ACM CCR, vol. 40, no. 3, 2010. [Online]. Available: http://doi.acm.org/10.1145/1823844.1823848

[7] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. R. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W. Chang, and Z. Shi, "The QUIC Transport Protocol: Design and Internet-Scale Deployment," ser. ACM SIGCOMM, 2017. [Online]. Available: http://doi.acm.org/10.1145/3098822.3098842

[8] M. Belshe, R. Peon, and M. Thomson, "Hypertext Transfer Protocol Version 2 (HTTP/2)," RFC 7540 (Proposed Standard), May 2015. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7540.txt

[9] S. Sundaresan, N. Feamster, R. Teixeira, and N. Magharei, "Measuring and Mitigating Web Performance Bottlenecks in Broadband Access Networks," ser. ACM IMC, 2013. [Online]. Available: http://doi.acm.org/10.1145/2504730.2504741

[10] V. Bajpai and J. Schönwälder, "A Survey on Internet Performance Measurement Platforms and Related Standardization Efforts," ser. IEEE Communications Surveys and Tutorials, vol. 17, no. 3, 2015. [Online]. Available: https://doi.org/10.1109/COMST.2015.2418435

[11] A. S. Asrese, P. Sarolahti, M. Boye, and J. Ott, "WePR: A Tool for Automated Web Performance Measurement," ser. IEEE GLOBECOM QoEMC Workshop, 2016. [Online]. Available: https://doi.org/10.1109/GLOCOMW.2016.7849082

[12] X. S. Wang, A. Krishnamurthy, and D. Wetherall, "Speeding up Web Page Loads with Shandian," ser. USENIX NSDI, 2016. [Online]. Available: https://goo.gl/Nycfpz

[13] Y. Zaki, J. Chen, T. Pötsch, T. Ahmad, and L. Subramanian, "Dissecting Web Latency in Ghana," ser. ACM IMC, 2014. [Online]. Available: http://doi.acm.org/10.1145/2663716.2663748

[14] J. Vesuna, C. Scott, M. Buettner, M. Piatek, A. Krishnamurthy, and S. Shenker, "Caching Doesn't Improve Mobile Web Performance (Much)," ser. USENIX ATC, 2016. [Online]. Available: https://goo.gl/n9k7NZ

[15] Y. Liu, Y. Ma, X. Liu, and G. Huang, "Can HTTP/2 Really Help Web Performance on Smartphones?" ser. IEEE SCC, 2016. [Online]. Available: https://doi.org/10.1109/SCC.2016.36

[16] V. Bajpai and J. Schönwälder, "A Longitudinal View of Dual-Stacked Websites - Failures, Latency and Happy Eyeballs," ser. IEEE/ACM Transactions on Networking, 2019, to appear.

[17] V. Bajpai and J. Schönwälder, "IPv4 versus IPv6 - Who Connects Faster?" ser. IFIP Networking, 2015. [Online]. Available: https://doi.org/10.1109/IFIPNetworking.2015.7145323

[18] C. Marquez, M. Gramaglia, M. Fiore, A. Banchs, C. Ziemlicki, and Z. Smoreda, "Not All Apps Are Created Equal: Analysis of Spatiotemporal Heterogeneity in Nationwide Mobile Service Usage," ser. ACM CoNEXT, 2017. [Online]. Available: https://doi.org/10.1145/3143361.3143369

[19] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian, "Internet Inter-domain Traffic," ser. ACM SIGCOMM, 2010. [Online]. Available: http://doi.acm.org/10.1145/1851182.1851194

[20] V. Bajpai, A. Brunstrom, A. Feldmann, W. Kellerer, A. Pras, H. Schulzrinne, G. Smaragdakis, M. Wählisch, and K. Wehrle, "The Dagstuhl Beginners Guide to Reproducibility for Experimental Networking Research," ser. ACM CCR, 2019 (to appear).

[21] V. Bajpai, M. Kühlewind, J. Ott, J. Schönwälder, A. Sperotto, and B. Trammell, "Challenges with Reproducibility," ser. ACM SIGCOMM, Reproducibility Workshop, 2017. [Online]. Available: http://doi.acm.org/10.1145/3097766.3097767

[22] A. S. Asrese, M. Boye, and P. Sarolahti, "WePR: Web Performance and Rendering ," https://github.com/alemnew/wepr, 2019.

[23] A. S. Asrese, S. J. Eravuchira, V. Bajpai, and J. Ott, "Measuring Web Latency and Rendering Performance: Method, Tools & Longitudinal Dataset (Dataset)," https://github.com/alemnew/2019-tnsm-webperf-analysis, 2019.

[24] B. Krishnamurthy and C. E. Wills, "Analyzing Factors that Influence End-to-End Web Performance," ser. Computer Networks, 2000. [Online]. Available: https://doi.org/10.1016/S1389-1286(00)00067-0

[25] D. Fetterly, M. S. Manasse, M. Najork, and J. L. Wiener, "A Large-scale Study of the Evolution of Web Pages," ser. Wiley SPE, 2004. [Online]. Available: https://doi.org/10.1002/spe.577

[26] B. Ager, W. Mühlbauer, G. Smaragdakis, and S. Uhlig, "Web Content Cartography," ser. ACM IMC, 2011. [Online]. Available: http://doi.acm.org/10.1145/2068816.2068870

[27] T. Callahan, M. Allman, and V. Paxson, "A Longitudinal View of HTTP Traffic," ser. PAM, 2010. [Online]. Available: https://doi.org/10.1007/978-3-642-12334-4_23

[28] S. Ihm and V. S. Pai, "Towards Understanding Modern Web Traffic," ser. ACM IMC, 2011. [Online]. Available: http://doi.acm.org/10.1145/2068816.2068845

[29] I. Arapakis, X. Bai, and B. B. Cambazoglu, "Impact of Response Latency on User Behavior in Web Search," ser. ACM SIGIR, 2014. [Online]. Available: http://doi.acm.org/10.1145/2600428.2609627

[30] A. M. Mandalari, A. Lutu, A. Custura, A. S. Khatouni, Ö. Alay, M. Bagnulo, V. Bajpai, A. Brunström, J. Ott, M. Mellia, and G. Fairhurst, "Experience: Implications of Roaming in Europe," ser. MobiCom, 2018. [Online]. Available: https://doi.org/10.1145/3241539.3241577

[31] R. Fanou, G. Tyson, P. François, and A. Sathiaseelan, "Pushing the Frontier: Exploring the African Web Ecosystem," ser. WWW, 2016. [Online]. Available: http://doi.acm.org/10.1145/2872427.2882997

[32] Z. Li, M. Zhang, Z. Zhu, Y. Chen, A. G. Greenberg, and Y. Wang, "WebProphet: Automating Performance Prediction for Web Services," ser. USENIX NSDI, 2010. [Online]. Available: https://goo.gl/mRY7ta

[33] X. S. Wang, A. Balasubramanian, A. Krishnamurthy, and D. Wetherall, "Demystifying Page Load Performance with WProf," ser. USENIX NSDI, 2013. [Online]. Available: https://goo.gl/WJ1wZZ

[34] D. Naylor, A. Finamore, I. Leontiadis, Y. Grunenberger, M. Mellia, M. M. Munafò, K. Papagiannaki, and P. Steenkiste, "The Cost of the "S" in HTTPS," ser. ACM CoNEXT, 2014. [Online]. Available: http://doi.acm.org/10.1145/2674005.2674991

[35] A. Balachandran, V. Aggarwal, E. Halepovic, J. Pang, S. Seshan, S. Venkataraman, and H. Yan, "Modeling Web Quality-of-Experience on Cellular Networks," ser. ACM MobiCom, 2014. [Online]. Available: http://doi.acm.org/10.1145/2639108.2639137

[36] Y. El-khatib, G. Tyson, and M. Welzl, "Can SPDY Really Make the Web Faster?" ser. IFIP Networking, 2014. [Online]. Available: https://doi.org/10.1109/IFIPNetworking.2014.6857089

[37] X. S. Wang, A. Krishnamurthy, and D. Wetherall, "How Much Can We Micro-Cache Web Pages?" ser. ACM IMC, 2014. [Online]. Available: http://doi.acm.org/10.1145/2663716.2663739

[38] M. Butkiewicz, D. Wang, Z. Wu, H. V. Madhyastha, and V. Sekar, "Klotski: Reprioritizing Web Content to Improve User Experience on Mobile Devices," ser. USENIX NSDI, 2015. [Online]. Available: https://goo.gl/rCggN4

[39] C. Kelton, J. Ryoo, A. Balasubramanian, and S. R. Das, "Improving User Perceived Page Load Times Using Gaze," ser. USENIX NSDI, 2017. [Online]. Available: https://goo.gl/eXTLAH

[40] W. Li, Z. Zhao, G. Min, H. Duan, Q. Ni, and Z. Zhao, "Reordering Webpage Objects for Optimizing Quality-of-Experience," ser. IEEE

Access, 2017. [Online]. Available: https://doi.org/10.1109/ACCESS.2017.2689002
[41] W. Zhou, Q. Li, M. Caesar, and B. Godfrey, "ASAP: a Low-Latency Transport Layer," ser. ACM CoNEXT, 2011. [Online]. Available: http://doi.acm.org/10.1145/2079296.2079316
[42] P. Biswal and O. Gnawali, "Does QUIC Make the Web Faster?" ser. IEEE GLOBECOM, 2016. [Online]. Available: https://doi.org/10.1109/GLOCOM.2016.7841749
[43] T. Zimmermann, B. Wolters, and O. Hohlfeld, "A QoE Perspective on HTTP/2 Server Push," ser. SIGCOMM Internet-QoE Workshop, 2017. [Online]. Available: http://doi.acm.org/10.1145/3098603.3098604
[44] S. Baraković and L. Skorin-Kapov, "Survey of Research on Quality of Experience Modelling for Web Browsing," ser. Springer QUX, 2017. [Online]. Available: https://doi.org/10.1007/s41233-017-0009-2
[45] M. Varela, L. Skorin-Kapov, T. Mäki, and T. Hoßfeld, "QoE in the Web: A Dance of Design and Performance," ser. IEEE QoMEX, 2015. [Online]. Available: https://doi.org/10.1109/QoMEX.2015.7148084
[46] T. Hoßfeld, S. Biedermann, R. Schatz, A. Platzer, S. Egger, and M. Fiedler, "The Memory Effect and its Implications on Web QoE Modeling," ser. IEEE ITC, 2011.
[47] S. Egger, P. Reichl, T. Hoßfeld, and R. Schatz, ""Time is Bandwidth"? Narrowing the gap between Subjective Time Perception and Quality of Experience," ser. IEEE ICC, 2012. [Online]. Available: https://doi.org/10.1109/ICC.2012.6363769
[48] D. Strohmeier, M. Mikkola, and A. Raake, "The Importance of Task Completion Times for Modeling Web-QoE of Consecutive Web Page Requests," ser. IEEE QoMEX, 2013. [Online]. Available: https://doi.org/10.1109/QoMEX.2013.6603203
[49] A. Sackl, P. Casas, R. Schatz, L. Janowski, and R. Irmer, "Quantifying the Impact of Network Bandwidth Fluctuations and Outages on Web QoE," ser. IEEE QoMEX, 2015. [Online]. Available: https://doi.org/10.1109/QoMEX.2015.7148078
[50] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson, "Netalyzr: Illuminating the Edge Network," ser. ACM IMC, 2010. [Online]. Available: http://doi.acm.org/10.1145/1879141.1879173
[51] M. Dhawan, J. Samuel, R. Teixeira, C. Kreibich, M. Allman, N. Weaver, and V. Paxson, "Fathom: a browser-based network measurement platform," ser. ACM IMC, 2012. [Online]. Available: http://doi.acm.org/10.1145/2398776.2398786
[52] S. J. Eravuchira, V. Bajpai, J. Schönwälder, and S. Crawford, "Measuring Web Similarity from Dual-Stacked Hosts," ser. CNSM, 2016. [Online]. Available: https://doi.org/10.1109/CNSM.2016.7818415
[53] D. Joumblatt, R. Teixeira, J. Chandrashekar, and N. Taft, "HostView: Annotating End-host Performance Measurements with User Feedback," ser. ACM SIGMETRICS PER, 2010. [Online]. Available: http://doi.acm.org/10.1145/1925019.1925028
[54] R. Netravali, A. Sivaraman, S. Das, A. Goyal, K. Winstein, J. Mickens, and H. Balakrishnan, "Mahimahi: Accurate Record-and-Replay for HTTP," ser. USENIX ATC, 2015. [Online]. Available: https://doi.org/10.1145/2619239.2631455
[55] D. N. da Hora, A. S. Asrese, V. Christophides, R. Teixeira, and D. Rossi, "Narrowing the Gap Between QoS Metrics and Web QoE Using Above-the-fold Metrics," ser. PAM, 2018. [Online]. Available: https://doi.org/10.1007/978-3-319-76481-8_3
[56] Q. Gao, P. Dey, and P. Ahammad, "Perceived Performance of Top Retail Webpages In the Wild: Insights from Large-scale Crowdsourcing of Above-the-Fold QoE," ser. ACM SIGCOMM, Internet-QoE, 2017. [Online]. Available: http://doi.acm.org/10.1145/3098603.3098606
[57] A. S. Asrese, E. A. Walelgne, V. Bajpai, A. Lutu, O. Alay, and J. Ott, "Measuring Web Quality of Experience in Cellular Networks," ser. PAM, 2019, to appear.
[58] Y. Chiu, B. Schlinker, A. B. Radhakrishnan, E. Katz-Bassett, and R. Govindan, "Are We One Hop Away from a Better Internet?" ser. ACM IMC, 2015. [Online]. Available: http://doi.acm.org/10.1145/2815675.2815719
[59] V. Bajpai, S. J. Eravuchira, and J. Schönwälder, "Lessons Learned From Using the RIPE Atlas Platform for Measurement Research," ser. CCR, 2015. [Online]. Available: http://doi.acm.org/10.1145/2805789.2805796
[60] V. Bajpai, S. J. Eravuchira, and J. Schönwälder, "Dissecting Last-mile Latency Characteristics," ser. ACM CCR, vol. 47, no. 5, 2017. [Online]. Available: http://doi.acm.org/10.1145/3155055.3155059
[61] E. A. Walelgne, S. Kim, V. Bajpai, S. Neumeier, J. Manner, and J. Ott, "Factors Affecting Performance of Web Flows in Cellular Networks," ser. IFIP Networking, 2018.
[62] J. Shaikh, M. Fiedler, and D. Collange, "Quality of Experience from User and Network Perspectives," ser. Springer ANTE, vol. 65, no. 1-2, 2010. [Online]. Available: https://doi.org/10.1007/s12243-009-0142-x
[63] Varas, Cristian, "Demystifying Google Global Cache," https://goo.gl/wdMQwp, 2015, Retrieved on Mar 15, 2018.
[64] E. Bocchi, L. D. Cicco, M. Mellia, and D. Rossi, "The Web, the Users, and the MOS: Influence of HTTP/2 on User Experience," ser. PAM, 2017. [Online]. Available: https://doi.org/10.1007/978-3-319-54328-4_4

Alemnew Sheferaw Asrese is a doctoral student at Aalto University, Finland. He is advised by Jörg Ott and Pasi Sarolahti. He received Master's (2014) degree in Computer Science from University of Trento, Italy and Bachelor's (2010) degree in Information Science from Adama University, Ethiopia. He was a visiting PhD student at TU Munich (2017), INRIA (2017), Simula Research Labs (2016). His research focus is on network measurement and QoE modelling. He has been involved in METRICS ITN, FP7 Leone and VENUS-C European research projects.

Steffie Jacob Eravuchira is an Embedded Communications Engineer at SamKnows Limited, London, UK. She received her Masters degree in Computer Science from Jacobs University Bremen, Germany in August 2015. During her Masters she specialized in Internet measurements using large-scale measurement platforms such as SamKnows and RIPE Atlas. She is interested in latency and IPv6 measurements in broadband networks.

Vaibhav Bajpai is postdoctoral fellow at TUM, Germany. He received his PhD (2016) and Masters (2012) degrees in Computer Science from Jacobs University Bremen, Germany. He is the recipient of the ACM SIGCOMM (2018) best paper award and IEEE COMSOC award (2017) for the best dissertation in network and service management. He is interested in future Internet protocols, content delivery, network operations and management and reproducibility of scientific research.

Pasi Sarolahti received his Master's degree from University of Helsinki (2001), and PhD degree on Computer Science from University of Helsinki in 2007. He is currently a Lecturer in Aalto University where he joined on 2010, after working 8 years in Nokia and as a visiting researcher in International Computer Science Institute in Berkeley, CA, USA. His interests are in TCP/IP networking both as a researcher and as an educator. He also served as IETF working group chair for 6 years in DCCP and TCPM working groups until 2016.

Jörg Ott holds the Chair for Connected Mobility at Technical University of Munich since Aug 2015. He is also an adjunct professor at Aalto University, where he was a professor from 2005. He is interested in understanding, designing, and building Internet-based communication systems and services. His research focus is on network and system architectures, protocols, and applications for mobile systems. This includes measuring, modeling, analyzing, and predicting network characteristics and application performance as well as preserving user privacy.