AMP up your Mobile Web Experience: Characterizing the Impact of Google's Accelerated Mobile Project

Byungjin Jun[†] Fabián E. Bustamante[†] Sung Yoon Whang[†] Zachary S. Bischof[†⋆]
[†]Northwestern University [⋆]IIJ Research

ABSTRACT

The rapid growth in the number of mobile devices, subscriptions and their associated traffic, has served as motivation for several projects focused on improving mobile users' quality of experience (QoE). Few have been as contentious as the Google-initiated Accelerated Mobile Project (AMP), both praised for its seemingly instant mobile web experience and criticized based on concerns about the enforcement of its formats. This paper presents the first characterization of AMP's impact on users' QoE. We do this using a corpus of over 2,100 AMP webpages, and their corresponding non-AMP counterparts, based on trendy-keyword-based searches. We characterized AMP's impact looking at common web QoE metrics, including Page Load Time, Time to First Byte and SpeedIndex (SI). Our results show that AMP significantly improves SI, yielding on average a 60% lower SI than non-AMP pages without accounting for prefetching. Prefetching of AMP pages pushes this advantage even further, with prefetched pages loading over 2,000ms faster than non-prefetched AMP pages. This clear boost may come, however, at a non-negligible cost for users with limited data plans as it incurs an average of over 1.4 MB of additional data downloaded, unbeknownst to users.

CCS CONCEPTS

• **Networks → Network measurement**; **Mobile networks**.

KEYWORDS

Google AMP; Accelerated Mobile Pages: Network measurement; Mobile Networks

ACM Reference Format:

Byungjin Jun, Fabián E. Bustamante, Sung Yoon Whang, Zachary S. Bischof. 2019. AMP up your Mobile Web Experience: Characterizing the Impact of Google's Accelerated Mobile Project. In The 25th Annual International Conference on Mobile Computing and Networking (MobiCom'19), October 21–25, 2019, Los Cabos, Mexico. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3300061.3300137

MobiCom '19, October 21–25, 2019, Los Cabos, Mexico

ACM ISBN 978-1-4503-6169-9/19/10…$15.00
https://doi.org/10.1145/3300061.3300137

1 INTRODUCTION

Over the last decade, the number of mobile subscriptions has grown rapidly, surpassing 7.7 billion by late 2017 [25]. Since October 2016, more websites have been loaded on smartphones and mobile devices than on desktop computers as users spend up to three times more hours browsing on their smartphones than on any other devices [43, 46].

Despite the shift toward mobile browsing, much of the web has been designed for desktop machines on wired connections. This "disconnect" typically translates into complex websites that offer poor quality of experience to mobile users [20], something that has recently received a significant amount of attention from researchers and industry alike [21, 32, 36, 39, 49]. The impact of low quality of experience (QoE) on user retention (e.g., [19]) has led to a number of efforts studying mobile performance and exploring ways to reduce page load time (PLT) and related proxies of QoE. Projects such as Facebook Instant Articles [10] rely on formats and infrastructure to speed up browsing, while Amazon Silk [13] and Opera Mini[18] do it through specialized web browsers. Accelerated Mobile Project (AMP) is a recent effort started by Google with a similar goal of improving the mobile browsing experience. Announced in October 2015 [16], AMP provides content creators with what is essentially a stripped-down and optimized version of standard web development tools. While there has been anecdotal evidence of its benefits [22, 24], we are not aware of prior, independent efforts to quantify its impact on users' browsing QoE.

This paper presents the first characterization of the performance impact of AMP on user experience. We use a set of common metrics to characterized AMP's impact on QoE – Page Load Time (PLT), Time to First Byte and SpeedIndex (SI) – with a corpus of over 2,100 AMP webpages and their corresponding non-AMP counterparts, collected using trending keyword-based search results.

We show that AMP significantly improves SI, yielding a 60% lower SI on average, not accounting for pre-rendering, compared with non-AMP pages. The performance improvement comes primarily from strict restrictions for simpler page, lazy loading, caching in the AMP CDN and pre-rendering of Google search results.

We find that the pre-rendering of AMP pages pushes this advantage even further, with prefetched pages loading over 2,000ms faster than non-prerendered AMP pages. This clear boost may come, however, at a non-negligible cost for users with limited data plans as it results in over 1.4 MB of additional data downloaded on average, unbeknownst to users.[1]

Our contributions can be summarized as follows:

- We develop and apply a methodology to evaluate the performance benefits of AMP. We use a corpus of AMP and non-AMP page pairs gathered using trending keyword-based searches (§3).
- We present the first characterization of AMP's impact on web QoE, analyzing the contribution of the different aspects of AMP's design (§4).
- We quantify the performance benefits and potential hidden cost of AMP's use of prefetching, in terms of the average number and the cost of additional bytes downloaded on a search (§5).
- We make the collected data, test frameworks, corpus of AMP pages and measurement results available to the community.[2]

2 AMP OVERVIEW

The next paragraphs provide an overview of the AMP project, the sometime confusing set of URLs associated with it, and Google's approach to improve mobile web performance through AMP.

2.1 AMP and its URLs

Web pages have become increasingly complex over the years, from early sites hosting text and images to current ones with tens or hundreds of files, including several HTML, JavaScript, CSS and images, that fetch content from a range of third-party providers such as ad agencies, analytics and content distribution networks (CDNs) [20]. Although most networks

[1]The actual cost for a single user depends on the rate at which prerendering is use and the frequency with which the user visits the prerendered site.
[2]http://www.aqualab.cs.northwestern.edu/projects/AMPUp.html

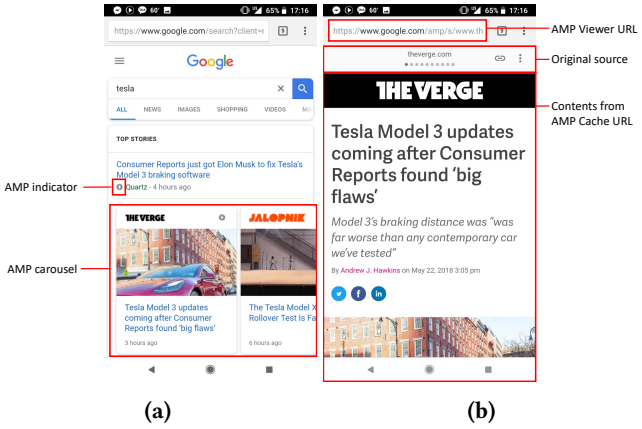


Figure 1: An example of a Google search result on an AMP-enabled device (left) and an AMP page displayed on the same device (right).

have managed to "keep up" with this increased complexity, mobile web browsing can be excruciatingly slow, even when doing it from high-end devices connected through state-of-the-art networks. Given the potential impact that web experience on user retention, several efforts have focused on improving mobile users' quality of experience, from Amazon's Silk to Facebook's Instant Articles and Google AMP.

AMP aims to improve mobile web performance by reducing website complexity and leveraging Google's large infrastructure and presence throughout the web browsing pipeline. Since its first release in 2016, AMP continues to gain adoption by content providers and CDNs [12], as we briefly discuss in Section 7.

When a user submits a search in Google through an AMP-enabled browser, the search results will typically include AMP-enabled pages. Figure 1a shows an example of a search result on Chrome, an AMP-enabled browser. When the user clicks on an AMP link, perhaps from the search result page, an AMP page such as the one in Figure 1b is quickly rendered.

The AMP URLs. There are three different types of URLs associated with an AMP-enabled webpage: the *AMP original URL*, the *AMP cache URL*, and the *AMP viewer URL* [27].

The *AMP original URL* is an AMP version of the page hosted by the original content provider (potentially in the provider's CDN). The *AMP original URL* is not exposed to users from Google search. This URL is validated and cached on Google's own dedicated CDN for AMP pages, the *AMP CDN*, as the *AMP cache URL* [15].

Users do not typically encounter the *AMP cache URL*, but see instead the *AMP viewer URLs*. When a user does a web search that returns AMP-enabled results, one of the results is pre-rendered behind the scene in a hidden iframe. If the user clicks on this result, the AMP viewer nearly instantaneously

Non-AMP URL:	http://www.example.com/doc.html
Original AMP URL:	http://www.example.com/amp/doc.html
AMP Cache URL:	https://www-example-com.cdn.ampproject.org/c/www.example.com/amp/doc.html
AMP Viewer URL:	https://www.google.com/amp/www.example.com/amp.doc.html

Figure 2: An example of four different types of URLs that can correspond to a single AMP-enabled article.

switches to the already rendered iframe. At this point, the user has not moved to a different page, but the viewer URL will be updated to reflect the document shown. To update the browser URL, the viewer uses the `History API`[3] which restricts the new URL to be in the same domain as the original URL (i.e., Google's search result URL) [27]. This restriction means that the actual AMP page revealed to the user must be served from the Google CDN, which is different from the AMP CDN where the *AMP cache URL* resides. As a result of this, the URL displayed at the top of the browser will make the page look like it belongs to Google (i.e., the URL starts with `www.google.com/amp/`).[4] To provide the appropriate attribution for the content, every AMP viewer adds a header bar that displays the page's actual origin.

In addition to these three AMP URLs, there is also the non-AMP URL, that of the page hosted by the original content provider. Note that not all publishers maintain this version of the page.

For illustration, Figure 2 shows examples of the types of URLs described for http://www.example.com/doc.html, including the non-AMP URL and the three AMP-related URLs.

2.2 How AMP works

AMP's potential performance benefits come mainly from three sources: simpler, better performing pages and lazy loading, caching in the AMP CDN and pre-rendering of page search results.

First, AMP largely simplifies the original webpage and replaces some of the standard HTML tags by its own HTML tags, marked with the `<HTML ⚡>` (or `<HTML AMP>`) tag, in a performant way. For example, object tags for images (`<image>`), audio (`<audio>`), and video (`<video>`) are replaced, respectively, by the AMP tags `<amp-image>`, `<amp-audio>`, and `<amp-video>`. These objects must now be declared with static sizes and are loaded by the AMP system. Also, AMP restricts the use of JavaScript and CSS. AMP pages are not allowed to include any author-written JavaScript besides the AMP-provided custom AMP elements which have JavaScript under the hood. The use of CSS is discouraged for the same reason, and all styles must be written inline. AMP also limits the total size of CSS code in order to minimize file size and improve load time. The AMP team particularly put efforts on preventing JavaScript and CSS from blocking the DOM rendering during the initial page load. To clean the critical path, all JavaScript on AMP pages are executed asynchronously, and all third-party JavaScript and other extensions like Instagram embedding need to be in an `<amp-iframe>` to prevent blocking the rendering of the base page.

Second, AMP adopts lazy loading of certain objects and leverages Google's AMP CDN for caching. It uses lazy loading to avoid loading unnecessary objects. For instance, images and ads currently not in the first viewport are not loaded until the user scrolls down. Most of these restrictions so far are adapted on AMP original URLs. Beside caching, AMP cache validates the owner's original AMP page so that any AMP page encountered by users is always a complete one that has all benefit of AMP, and does additional optimization of images, transforming images to have no invisible data, and scaling down in size and quality for mobile devices careful not to impact users' visual experience.

Lastly, some of AMP pages in the Google's search results that users may potentially click (e.g., top result) are pre-rendered at Google's search results page in an effort to further reduce loading times. This particular technique results in instant-like access to the target page.

Beyond this, the combination of AMP and the Chrome browser allows Google to configure both ends of a browsing connection – the client (Google Chrome) and the server (Google CDN), making it easy to deploy performance-enhancing transport layer protocols such as QUIC [6, 31].

3 METHODOLOGY

In this section we describe the experimental methodology we devised to understand the impact of AMP on mobile web performance, including our approach to collecting AMP web page URLs for analysis, our baseline for comparison, and the testbed configuration and experimental design.

3.1 Collecting a set of AMP pages

To obtain a representative set of AMP pages for analysis, we built a crawler and scraper with Selenium [1] that collects AMP page URLs from Google search results. We search for

[3] https://developer.mozilla.org/en-US/docs/Web/API/History

[4] We briefly discuss the concerns about attribution in Sec. 7.

Category	Num. of pages
News/Weather/Information	1,001
Movies	79
Law/Government/Politics	79
Television/Video	78
Financial News	76
World Football/Soccer	65
American Football	64
Automotive	61
Sports	59
Basketball	56

Table 1: Top ten most popular Alexa categories in our collection. News organizations are the primary adopters of AMP.

popular keywords using Google Trends for 2017,[5] both globally and for the United States (US). We include keywords for the US considering the dominance of the English language in the web [42]. Google Trends provides a list of the top 10 search keywords for a number of popular categories (e.g., actors, TV shows, global news) in a given year. In total, we collected 670 keywords, 150 globally and 520 from US, and were left with 578 after removing duplicates. Nearly 90% (522) of all keywords in the set were in English and 9% (52) in Spanish. The remaining 1% of words were in Chinese and Turkish.

We issued searches with these keywords (e.g., "meghan markle", "french election") and collected distinct AMP viewer URLs from the search results. Our 578 keywords yielded a set of 3,401 unique AMP pages after removing duplicates.

3.2 Defining a baseline for comparison

A challenge we faced in characterizing the impact of AMP is determining the most appropriate baseline for comparison. Rather than crafting a set of synthetic pages with and without AMP, we rely on the fact that most AMP pages have an associated non-AMP version hosted by the content provider.

In many cases, the non-AMP version can be found easily through the `<link>` tag in the page header (e.g., `<link rel="canonical" href="...">`). For some of the pages that do not include this tag, it is possible to find their non-AMP version following some basic naming conventions (e.g., removing `/amp/` from the original AMP URL) [27].

After expanding our dataset by identifying the corresponding original AMP URLs as described, we were left with 2,132 pages for which we can find their non-AMP counterparts. Table 1 shows the number of pages in each of the top 10 most popular categories of the web pages in our collection. This reveals that news organizations are currently the primary adopters of AMP.

[5]The latest set at the time we began this iteration of our analysis.

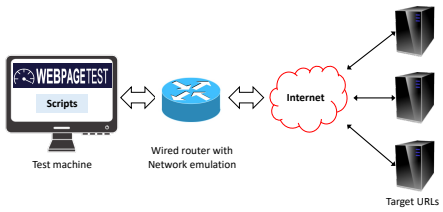


Figure 3: Testbed setup. The test machine runs our scripts on top of WebPageTest.

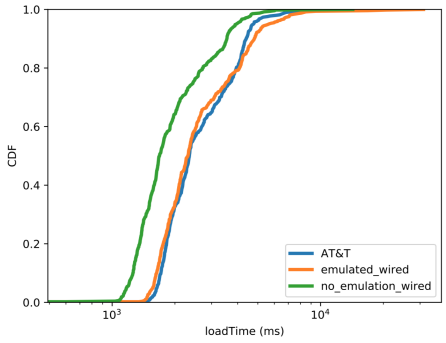


Figure 4: CDF of the PLT for each of the AMP pages across three different network configurations (x-axis in log scale.

3.3 Testbed

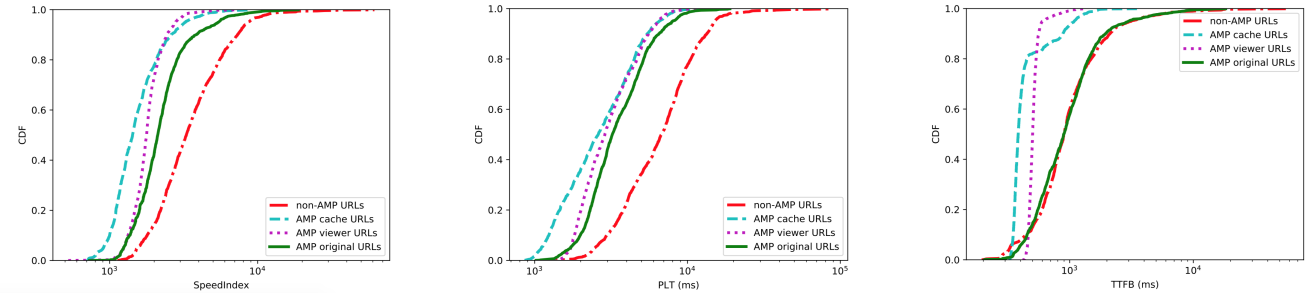
For our evaluation, we use a testbed that consists of a machine using WebPageTest v17.12 [4], connected to the Internet through a router under our control (Figure 3).

WebPageTest (WPT) is an open source tool for testing websites' speed under different network conditions. We run a private instance of WPT and use WPT's *wpt_batch.py+* API to automate testing. While WPT itself introduces a small overhead when testing webpages, this is consistent across sites and thus does not impact our comparative analysis.

Our client runs on a MacBook Pro with 16 GB of 1600MHz memory and an Intel Core i7 processor. We use a TP-Link N750 Wireless Wi-Fi Dual Band router (TL-WDR4300) which we configured to run OpenWRT (Chaos Calmer 15.05.1, r48532) and Linux's Traffic Control and Network Emulation tools, which we use to emulate mobile network conditions. We use a separate router, as in Kakhki et al.'s analysis of QUIC [31], to minimize interference.

3.4 Web QoE Metrics

To study the impact of AMP on mobile web quality of experience, we rely on three commonly used metrics of QoE – Page Load Time, Time to First Byte, and SpeedIndex [17].



(a) SI of AMP pages. **(b) PLT of AMP pages.** **(c) TTFB of AMP pages.**

Figure 5: Overall performance of AMP. The x-axis is shown in log scale for each figure.

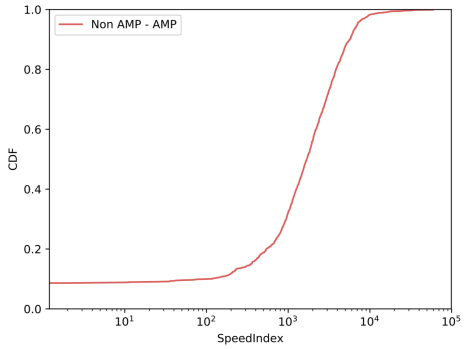


Figure 6: The relative difference in SI between AMP and non-AMP URLs (x-axis is log scale).

Page Load Time (PLT) is the time until all objects on a page have been loaded, while Time to First Byte (TTFB) is the time at which the first byte of the payload is received at the client. Speed Index (SI), a more recent "time to interactive" metric, captures the visual progress as the page loads – potentially a more accurate approximation of a user's experience (UX)– and computes an overall score for how quickly the content is painted [8]. Pages with faster performance have a lower SI score. In our anlaysis, we focus on SI rather than PLT, considering that AMP's main goal is to improve the user-perceiving speed, and SI nicely reflects it.

3.5 Network Environment

Due to the high data usage and cost of loading tens of thousands of pages repeatedly, we configured the network connection in our testbed environment to simulate the behavior of a large US mobile provider. The settings for our testbed were based on a characterization analysis of this network using several tools, including SpeedTest [2] and NDT [3]. The connection offered nearly 45 Mbps of bandwidth with 40 ms of latency and almost 0% loss. To validate our emulated connection, we loaded a set of AMP pages using the actual and the emulated mobile connection, and compared PLT.

Figure 4 shows the distribution of PLT of AMP pages over both the emulated and actual mobile connection. For context we include the same distribution over a wired connection. Overall, the distribution of PLT over the emulated connection closely follows to that of the actual mobile service. Using the Kolmogorov-Smirnov test to quantify the similarity of these distributions, we find a K-S value of 0.097 with a p-value of 0.99, suggesting that is highly unlikely they are different distributions.

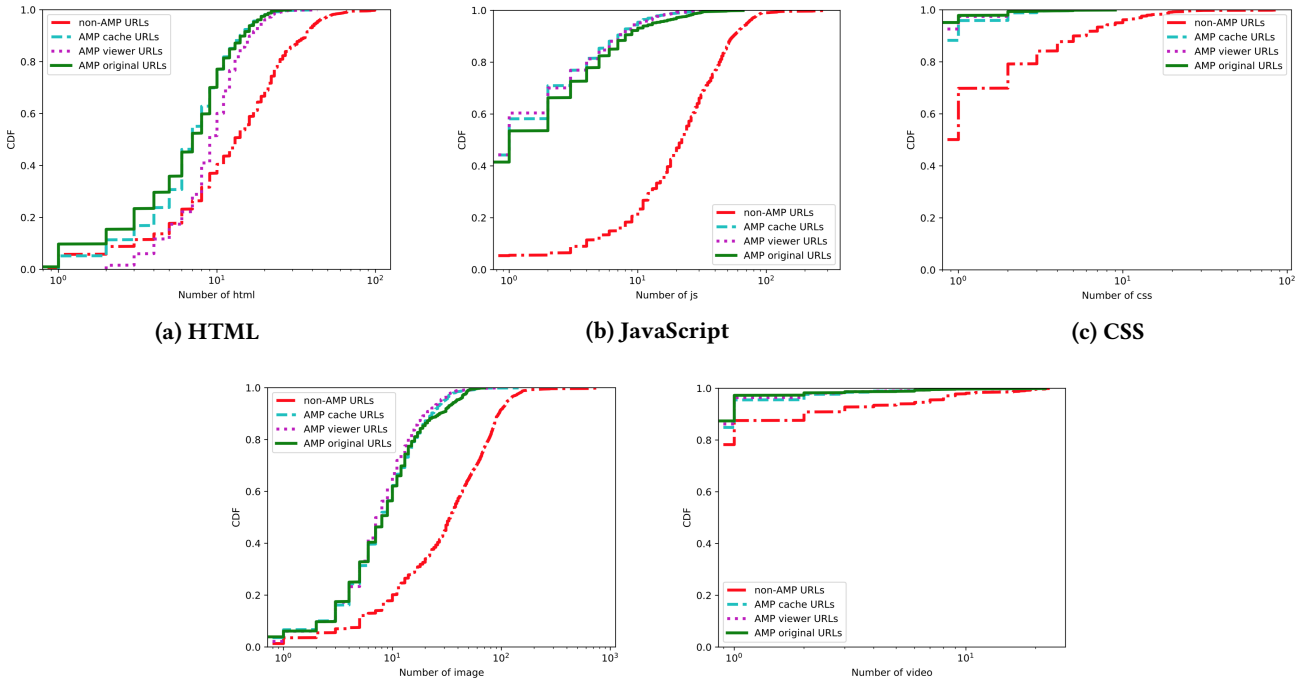
4 PERFORMANCE OF AMP

In the following paragraphs, we analyze the performance benefits of AMP. We begin by discussing its overall impact, before looking at the different aspects of AMP and their contributions.

4.1 Overall performance

To evaluate the overall performance impact of AMP, we compare the four different versions of each page: the non-AMP, the AMP original URL (hosted by the original content provider), AMP cache URL, and the AMP viewer URL version. We loaded each page three times and measured SI, PLT, and TTFB. Figure 5 presents these distributions for each version.

Overall, AMP cache URLs show the best performance and AMP viewer URLs have the most stable performance among all versions. Focusing on AMP, these two versions should show similar performance given that both are already optimized and are served from close-by servers via CDN (AMP CDN for AMP cache URLs and Google CDN for AMP viewer URLs). However AMP viewer URLs' slightly worse performance may result from the additional work done for pre-rendering, whose benefits are not visible in this test. The performance gaps between these version, as measured



(a) HTML (b) JavaScript (c) CSS

(d) Image (e) Video

Figure 7: The number of different object types loaded in AMP and non-AMP pages.

by SI and PLT, get smaller toward the right of the graph (e.g., higher PLT). We reason that the presence of the AMP viewer header bar in AMP viewer URLs, placed over the top of the original page, result in less content from the original page being drawn in the first viewport in the AMP viewer, reducing the effect of heavier page. The higher variation we observed on the performance of AMP cache URLs appears to be the result of pages' revalidation of AMP original versions. AMP cache treats every page as if it came with stale-while-revalidate=(3 months). We regard AMP cache URLs with the worst 20% TTFB as revalidated pages while test.

Figure 5a shows the distribution of SIs for all types of URLs. All AMP pages (AMP cache URLs, AMP viewer URLs and AMP original URLs) show significantly lower SI (1,666 ms, 1,859 ms, and 2,488 ms respectively) and smaller standard deviations (858 ms, 542 ms, and 1,574 ms respectively) than the non-AMP version (with a mean SI of 4,142 ms and a standard deviation of 3,370 ms). AMP cache URLs particularly shows 60% lower SI than non-AMP version. We can see the the largest SI gap is between AMP original URLs and non-AMP URLs, suggesting that much of *the advantage of AMP comes from a reduction in page complexity rather than its reliance on the caching infrastructure.*

Figure 5b plots the PLT results for the same four versions of a page. Again, all AMP versions perform significantly better in terms of PLT than the non-AMP counterparts, with an average PLT dropping from 7,719 ms to 3,058 ms, a 2.5x improvement on PLT. This large benefit comes from AMP's lazy loading (§ 2.2) that by delaying the loading of some objects excludes them from the PLT calculations. Focusing on the AMP versions, AMP cache URLs show the best PLT (3,058 ms, compare with 3,342 ms for AMP viewer URLs and 3,882 ms for AMP original URLs).

TTFB also shows the performance benefits of using AMP. As can be seen in Figure 5c, AMP cache and viewer pages have a shorter TTFB than their non-AMP counterparts. For the non-AMP URLs, we found an average TTFB of 1,333 ms (standard deviation, $\sigma = 2,420$ ms). Both AMP URLs had much lower latency, with an average TTFB of 484 ms ($\sigma =$ 267 ms) for the AMP cache URLs and 516 ms ($\sigma = 83$ ms) for the AMP viewer URLs. Interestingly, the AMP original URLs shows TTFB closer to those of the non-AMP versions, 1,225 ms. We believe this is due to the greater diversity in terms of physical proximity of the different content providers' servers and CDNs compared to either the AMP CDN or the Google CDN (for AMP cache and viewer URLs, respectively). The higher, top 20% TTFB of AMP cache URLs are probably caused by revalidation of the cache, as explained before.

Objects	non-AMP	Original	Cache	Viewer
HTML	16.8 (13.7)	7.7 (5.0)	7.9 (4.8)	10.0 (4.8)
JS	27.4 (24.5)	3.2 (6.0)	2.5 (4.2)	2.5 (4.0)
CSS	2.0 (4.8)	0.1 (0.5)	0.2 (0.6)	0.1 (0.4)
Image	45.8 (53.0)	11.6 (11.2)	10.9 (10.0)	10.0 (8.9)
Video	0.9 (2.8)	0.2 (1.1)	0.3 (1.0)	0.3 (1.2)

Table 2: Average (std) number of objects per URL type.

Through our evaluation, we found that while the distribution of SIs for the AMP versions of most sites were significantly faster than ther non-AMP original counterparts, about 9% of non-AMP URLs have lower SI than their corresponding AMP version. Upon further inspection we learned that all these cases are well-optimized non-AMP pages for which the AMP's additional optimizing task become overhead.

4.2 Website Complexity

We focus now on the differences in content between AMP and non-AMP versions of a page. We find that the performance improvement of AMP pages are significantly correlated with a reduction of objects downloaded – JavaScript objects in particular – for each page.

Figure 7 compares the total number of objects loaded per page for five different filetypes (HTML, JavaScript, CSS, image, and video) via the four types of URLs, while Table 2 lists the averages and standard deviations (between parentheses) for each comparison. Overall, the three types of AMP pages show similar number of objects across file types. Compared to the non-AMP version, on the other hand, the average numbers of HTML, JavaScript, and image objects are reduced by nearly an order of magnitude. This trend, again, reflects the impact of lazy loading as PLTs for entire pages do.

Looking across pages, there is a significant difference in the shapes of the distributions of the number of objects for AMP and non-AMP pages. For instance, AMP pages tend to be much more uniform in terms of the number of objects per page than their non-AMP counterparts (for all object types, the standard distribution for the number of objects on AMP pages is always smaller than for the non-AMP version). These large differences translate into much less time spent downloading the different types of objects. For example, the total time spent loading HTML objects for non-AMP pages is reduced from 5,370 ms ($\sigma = 4,473$ ms) to 1,979 ms ($\sigma = 996$ ms) for AMP cache URLs. On the other hand, the time improvement is smaller for CSS objects, as the reduction in the number of CSS objects is not as dramatic. For CSS objects, the total time spent loading for non-AMP pages is 300 ms ($\sigma = 479$ ms) and 40.5 ms ($\sigma = 136$ ms) for AMP cache URLs.

Our analysis of AMP versions show some interesting handling of CSS and HTML objects. While AMP requires all CSS to be in-lined, Fig 7 suggests there are some CSS files for a few AMP pages. Most of these CSS objects in AMP pages are either a font or CSS initiated by a third-party application. Other than CSS files from third-party applications, there are also a few AMP pages with actual CSS objects loaded from another URL of the same organization of the original page. It seems those providers has adopted this technique for uniformity of their pages. As for HTML objects, unlike other objects, AMP viewer URLs have prominently more files than other AMP versions. The more HTML objects are typically HTML pages downloaded from AMP cache URLs. This is predictable in consideration of that AMP viewer URLs' fetch the actual AMP page from the AMP cache URLs.

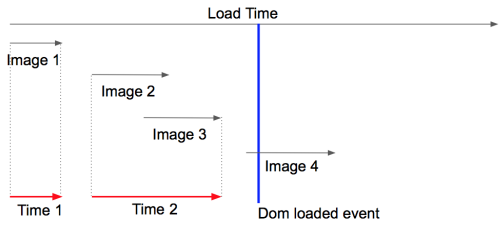


Figure 8: We use Time 1 + Time 2 to compute load time of image objects that block DOM, removing overlapping times.

Objects	non-AMP	Original	Cache	Viewer
HTML	1386 (2433)	1028 (1282)	312 (265)	335 (105)
JS	667 (739)	22.1 (118)	2.5 (17.5)	0.4 (10.6)
CSS	125 (220)	2.9 (26.4)	0.3 (5.5)	0 (0)
Image	518 (544)	30.6 (129)	6.3 (42.6)	0.2 (5.7)
DOM	3179 (3599)	1658 (1459)	860 (532)	907 (115)

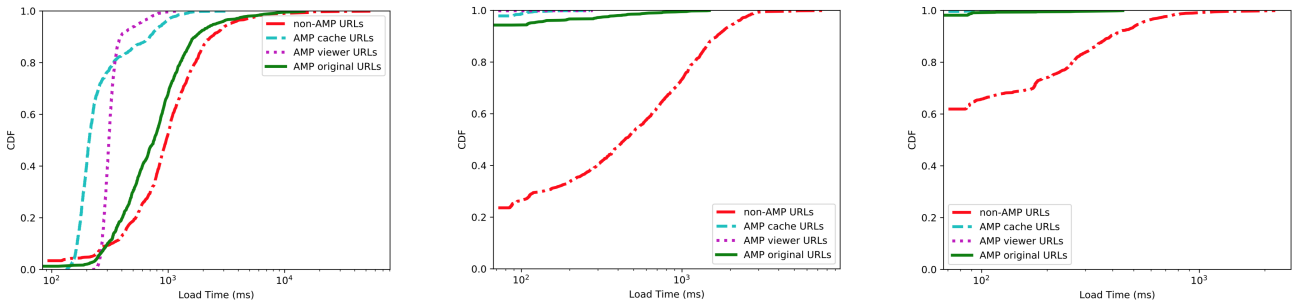
Table 3: Average (std) load time (ms) of objects before DOM event for each URL.

4.3 Removal of objects blocking the DOM

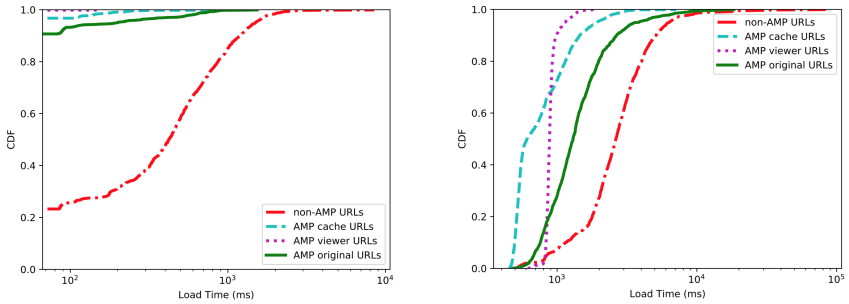
Although AMP pages typically have significantly fewer objects and a much lower PLT than their non-AMP versions, the main goal of AMP is improving user-perceived initial load time, to provide a "nearly instantaneous load".

One aspect that impacts initial loading time is the loading of the Document Object Model, or DOM, that provides the tree structure representation of a page. The loading of the DOM can be blocked or delayed by the fetching of additional objects such as HTML or JavaScript. AMP adopts additional techniques to prevent objects from blocking the DOM including the use of asynchronous JavaScript, in-line CSS, as well as minimizing style recalculation and removing unimportant objects from the critical DOM construction path.

While we are not able to analyze the individual contributions of each of these features, we study the impact of them,



(a) HTML (b) JavaScript (c) CSS



(d) Images (e) DOM load event

Figure 9: DOMContentLoadedEventEnd timing and load time of objects loaded before it (blocking the DOM). All x-axes are log scale.

as a whole, on the time to the triggering of the DOM content loaded event. We focus on analyzing the timing of `DOMContentLoadedEventEnd` (`DOMLoaded`), logging the objects loaded before `DOMLoaded` timing events for each version of the URLs (Figure 8). Figure 9 shows, for each object type, the total time spent loading each type of object before the DOMLoaded timing event for their page.

Overall, we find that the AMP techniques indeed result in a faster `DOMLoaded` event for the AMP versions of a page over their non-AMP counterparts as Table 3 shows. Regardless of version, AMP pages generally take very short time in loading most kind of objects except HTML, which is as expected since every webpage must load the base page. Except HTML objects, the loadtime of objects is consistently smaller for AMP original URLs than for non-AMP URLs, and for AMP cache and viewer URLs than for AMP original URL. The different optimizations discussed in the previous paragraphs explained this.

Looking at HTML objects, both AMP cache URLs and AMP viewer URLs shows smaller variance than the other two. As in our analysis of TTFB (Figure 5c), this can be explained by the pages being served from a CDN, with more consistent response times. Since loadtimes of the other objects with AMP URLs show negligible values comparing to HTML objects, HTML mainly decide the shape of entire `DOMLoaded` event graph although the gap between AMP versions and non-AMP pages enlarged.

Zero loadtime of CSS objects in AMP viewer URLs is particularly interesting. We already discussed tiny CSS loadtime for all AMP versions on CSS objects in AMP pages (Section 4.2). Furthermore, AMP viewer URLs have even fewer CSS files (0 here) than the other AMP URLs as most of the CSS files loaded before `DOMLoaded` events in other AMP URLs are fonts, which AMP prevents from loading before the `DOMLoaded` event.

In summary, by getting rid of objects blocking the DOM, `DOMLoaded` timing can be greatly improved over 4x, from 3,179 ms ($\sigma = 3,599$) for non-AMP URLs to 860 ms ($\sigma = 532$) for AMP cache URLs and 907 ms for AMP viewer URLs.

4.4 Fewer DNS Resolutions

The significant reduction in the number of objects in an AMP page should result, naturally, in fewer DNS resolutions being made. In addition, as AMP caches many of a page associated objects on the AMP CDN, we should see even fewer DNS requests. To verify this, we analyze the HAR files generated

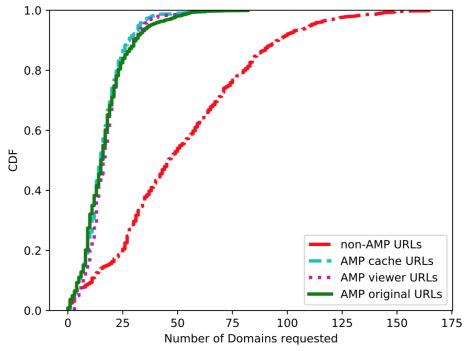


Figure 10: The number of DNS resolutions made while loading each URL.

from visiting the AMP and non-AMP pages and count the number of DNS resolutions made during a page load.

Figure 10 shows the CDF of these results, which are consistent with our analysis. We find large differences in the number of DNS resolutions between AMP and non-AMP pages. Concretely, we see a mean of 16 - 18 DNS ($\sigma \approx 10$) resolutions made for each AMP URLs, compared to a mean of 52 DNS ($\sigma = 33$) resolutions made for a non-AMP URLs.

4.5 QUIC and AMP

Beyond the factors already discussed, the combination of AMP CDN servers and the Chrome browser allows Google to potentially take advantage of a performance-enhancing, user-level transport protocol such as QUIC [6, 31]. While QUIC is not strictly a feature of AMP, considering the variability in network conditions to which mobile users are commonly exposed, it can clearly contribute to its performance.

To find out the extent of QUIC protocol's contribution to AMP's performance, we make use of the QUIC flag in Chrome (*chrome://flags/#enable-quic*) enabling/disabling it as needed. We visit the same URLs with QUIC-enabled and QUIC-disable Chrome, and repeat this under different network conditions, with different latencies and loss ratios. Figure 11 shows PLT of pages with both configurations and two network settings – without added latency or loss, and with 40 ms of additional latency and 1% of additional loss. With no additional latency or loss (two left-most lines in the plot), using QUIC (QUIC-on) yields a 1,983 ms of mean PLT ($\sigma = 1,046$), a minor improvement over the non-QUIC option with 2,082 ms ($\sigma = 1,073$). Under more stressed network conditions (two right-most lines in the plot), however, the use of QUIC shows clear advantages over the non-QUIC setting (QUIC-off), with a 1,000 ms gap in mean PLT (2,386 ms ($\sigma = 1,218$) with QUIC-on, 3,334 ms ($\sigma = 1,881$) with QUIC-off), consistent with the findings of Kakhki et al. [31].

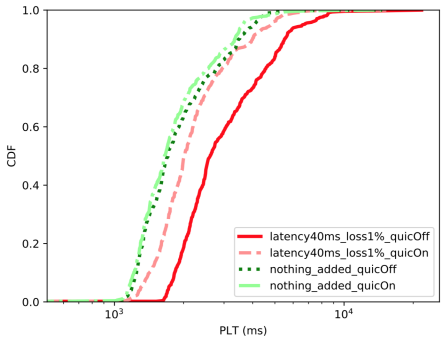


Figure 11: PLT in multiple configurations of QUIC and network conditions. The x-axis is log scale.

5 PRE-RENDERING WITH AMP CACHE

So far, we have analyzed AMP's performance benefits by comparing non-AMP pages with their AMP counterparts using the AMP's direct URL. Most users, however, would probably access AMP pages using the AMP viewer URLs embedded on web search results.

How users access AMP pages is key to their web experience given that pre-rendering of AMP pages is linked to search results. Concretely, the resources on the first viewport of AMP pages found on a search result are prefetched and pre-rendered while the user is looking at the search result page. If the user clicks on one of the pre-rendered pages, the results show up in nearly instant-like loading.

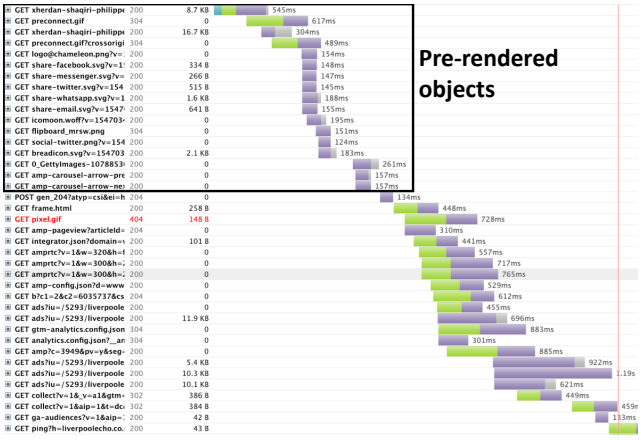
The following paragraphs discuss our approach to capturing the benefits and potential costs of pre-rendering.

5.1 Capturing the advantage of pre-rendering

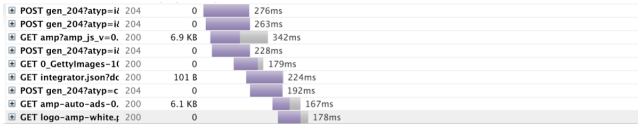
Pre-rendering in AMP is restricted to the first AMP page on the search result, or one of the AMP pages in the AMP carousel (as in Figure 1a) when the carousel is at the top of the search results. If there are no AMP pages above the fold, pre-rendering is done on the first AMP result appearing if/when the user scrolls down. We verify this by inspection of the HAR files, searching for any responses originated from the endpoint *cdn.ampproject.org*, AMP CDN.

To understand the impact of AMP pre-rendering of search results, we must correctly identify the top ranked AMP result – the one that will be pre-rendered – to immediately load it automatically, simulating a user selection, and capture the full benefits of pre-rendering. For every keywords in our dataset, then, we issue a search, identify the top AMP URL in the result page, and load the associated AMP page capturing the difference QoE metrics.

WPT supports tests with sequences of webpage requests using its scripting functionality. We run a two-step script per



(a) Har timeline of direct accessing to AMP viewer URLs.



(b) Har timeline of pre-rendered AMP viewer URLs.

Figure 12: HAR timeline of accessing to AMP viewer URLs with and without pre-rendering.

keyword that only records the second step (the subsequent page load) after the first step (the initial Google search).

Figure 12 uses an example AMP viewer URL[6] to illustrates the difference between the cascade of objects loaded with and without pre-rendering. In the figure, each row represents a request for an object, the request status code, the size of the response (size 0 means an asynchronous request), and the timing for processing the request. The vertical red line shows the page loaded event (PLT) and we presents objects loaded before this line[7]. This line does not show up in the bottom figure since the base page has been loaded from the pre-rendering step. A quick comparison of the top and bottom figures shows that a large number of objects, included the base page and image files, are not loaded when a user access the first AMP viewer URL as they have already been prefetched in the pre-rendering process. These prefetched objects, are downloaded before PLT (Figure 12a).

5.2 Pre-rendering and QoE

We now look into timing of prefetching at search result pages (Figure 13). We calculate each timing in the plot from the beginning of the page loading and see that prefetching always begins after PLT of the search result page (i.e., once the search results are presented to the user). Since the amount

[6]https://www.google.com/amp/s/www.liverpoolecho.co.uk/sport/football/football-news/xherdan-shaqiri-philippe-coutinho-surprise-15645729.amp

[7]PLT of directly accessed page is 4,306 ms, and pre-rendered page is 476 ms

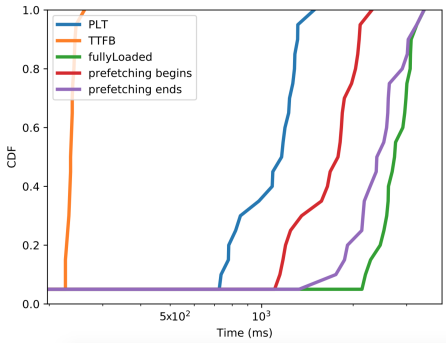


Figure 13: Timings of Google' search result page. The x-axis is shown in log scale.

of prefetched data is not trivial, the length of prefetching time is significant as well. Even though there are a few other objects loaded between the beginning and ending of prefetching, most of objects loaded within that interval come from the AMP CDN. Additionally, we find that the moment the connection to the AMP CDN is established (by requesting *preconnect.gif*) can vary from the beginning of the page loading to after the PLT.

To understand the impact that this may have on mobile users' QoE, we estimate the difference in PLT [8] when accessing AMP viewer URLs with and without pre-rendering. We estimate the PLT for 50 prefetched/pre-rendered sites from Google search result using our keyword sets.

Figure 14 compares this with the PLT time "direct". As expected, there is large difference between directly accessed pages and pre-rendered pages. Mean PLT of pre-rendered AMP viewer URLs is 833 ms (σ = 494), 2,057 ms faster than direct's PLT at 2,890 ms (σ = 1,610). This difference explains the nearly instantaneous rendering of the first AMP page which, because of pre-rendering, translates simply in the showing of a hidden, pre-rendered iframe.

5.3 Cost of AMP Prefetching

As one would expect, prefetching is not without a cost. To estimate the potential overhead of prefetching, we re-visit the search result pages of the selected 578 keywords, and look at amount of AMP data prefetched and pre-rendered.

We measure the total bytes downloaded by AMP prefetching from web search result page with the keyword set. Figure 15 shows this as a CDF over all keyword searches. As can be seen, there is a large jump on the top 10pct of web search results. This means that for around 90% of search result there are no AMP pages in the first viewport and, thus, no data being prefetched from AMP CDN (without scrolling).

[8]SI is apparently better than PLT to show the impact of pre-rendering, but we choose PLT since WPT's multi step script does not correctly record SI.

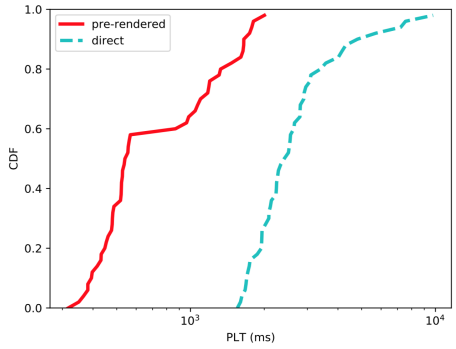


Figure 14: PLT of 50 AMP pages for 'direct' and 'pre-rendered'.

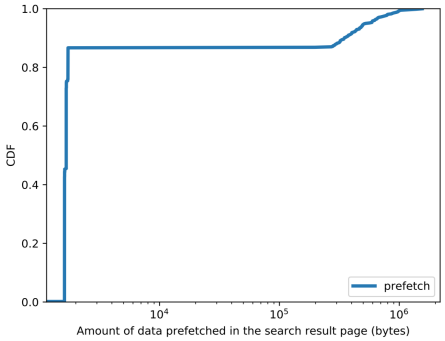


Figure 15: Prefetched data Google's search result page. The x-axis is shown in log scale.

For the other 10% of search result, however, where AMP results appear in the first viewport, pre-rendering incurs a not negligible overhead. In average, looking at those web search results, AMP prefetching consumes 1.4MB on data (2.3MB with 95 percentile, and 6.1MB for the maximum).

While the fraction of search results with AMP entries in the first viewport is a function of the particular set of keywords use, we posit that the collection of keywords from Google Trends provides a good estimate of the expected overhead of pre-rendering for end users. We consider the estimated overhead to be a lower-bound, as it does not include AMP resources in the next viewport although AMP will prefetch those upon scrolling. A report from ITU [30] let us eastimate the potential cost of this overehead to users. For users in the top-20 countries in terms of price of mobile-broadband, a 1.4MB extra data means $0.14 extra, on average (Figure 16). This overhead may not be an issue if users commonly follow the first, prefeteched, link they found in Google search. Based on average Click-Through-Rate for last one year in [37], however, only 22.64% of the first result in the

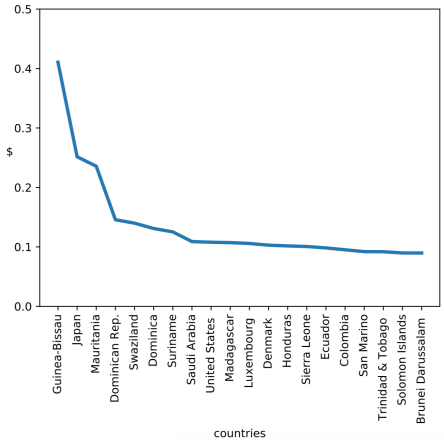


Figure 16: Cost per Google search with AMP in the first viewport

search result page is directly accessed, rendering the other 78.36% of prefetched pages mere overhead.

6 IMPACT OF AMP'S FEATURES

AMP is a collection of various design features which contributes to improving the mobile web experience. We focus here on the differences between the three AMP URLs, partially because many features in the AMP original URLs are difficult to isolate (e.g., the number of domain requests and total JavaScript object).

Figure 17 shows the performance gaps in terms of PLT for 50 pages from the collected set used in Sec. 5. A horizontal line across the graph shows, *for a particular page* (e.g., www.crowfishboxes.com), the PLT improvements respect to the non-AMP version of the AMP original, AMP cache and AMP pre-rendered versions. The differences between each of these points captures the contributions of each version's set of features. The largest performance benefit comes with AMP original page, with an average improvement of PLT of 9,657 ms compare to the baseline, non-AMP URLs. This improvement is the result of AMP's numerous static methods and its adoption of lazy loading.

AMP cache provide less impressive benefit, with 714 ms of average PLT improvement over the AMP original URLs. AMP cache's reduction of image size and quality, a key functionality of AMP cache, does not significantly improves PLT since AMP only records the first viewport and there are likely a few images.

As expected, pre-rendered AMP significantly enhances PLT (with a 1,879 ms improvement of pre-rendered AMP viewer URLs over the AMP cache URLs) since objects in the first viewport of the AMP viewer URLs has already been loaded when it was pre-rendered.

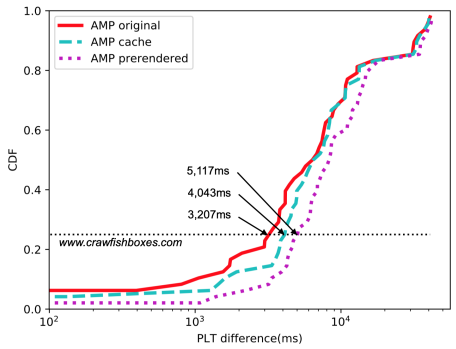


Figure 17: Impact of AMP's features on PLT. The horizontal line shows one particular page and the improvement to PLT of each of its AMP version over its non-AMP version.

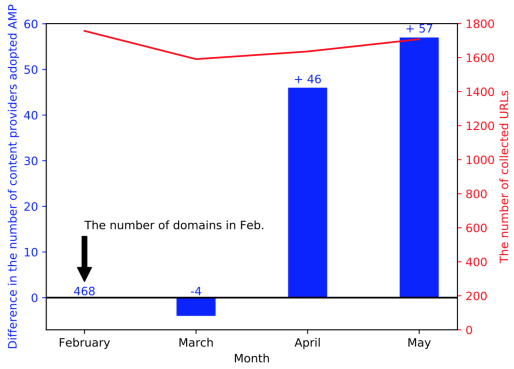


Figure 18: AMP adopting trend for 4 months.

7 DISCUSSION AND FUTURE WORK

Google has been promoting the adoption of AMP in a number of ways. The company has introduced plans for new application domains for AMP, such as Gmail and AMP Stories, and has proposed its standardization [28, 40, 45]. Beyond this, Google has been providing a number of incentives for content providers through their search engine and their popular browser. For instance, the AMP carousels for top stories, some of which can be seen in Figure 1a, place AMP pages at the top of the search response page.[9] Our preliminary analysis of AMP adoption rate suggests that the incentives are working.

Between February and May of 2018, we applied the same data collection approach we used to gathered our corpus of AMP pages. Using the same set of 578 Google trend keywords 2017, the most recent year available, we searched for keywords and count the number of AMP-adopted content providers revealed by our crawler.

Figure 18 plots the changes in the number of results per month using the number found in February, 468, as baseline. Other than a small dip in March (464), the figure shows a growing adoption trend with 514 and 525 content providers for April and May, respectively. While this is only an approximation, based on a subset of popular keywords that changes over time, the growing adoption trend seems clear.

7.1 Criticism around AMP

AMP has also attracted a significant degree of controversy [14, 23, 29], mainly driven by concerns about Google's influence, complications with attribution of content and the overhead of pre-fetching. A common concern is Google's prioritization of AMP results. AMP carousels, the visible prioritization, is already an important thing to consider for content providers, as we mentioned. Beyond this, there are also some concerns about the relationship between loading speed and the ranking of search results. Google has announced Google Speed Update [48], saying that Google will use page speed in mobile search ranking. There is no direct evidence of the relationship between this and AMP, but it would seem "natural" that AMP's improved performance would result in higher page rankings.

Another issue that has been raised is the fact that, by serving AMP pages to the users through AMP viewer URLs, user traffic stays within the Google's ecosystem. This, combined with the resulting "clean look" of AMP pages from the restricted format it enforces, also creates some confusion with respect to content attribution. Google has tried to help attribute content to the original sources [15] by adding the source URL in the AMP viewer page and by eventually matching URLs to the original source [44].

Finally, as we show in Sec. 5, AMP prefetching and prerendering results in some additional data (and power) use with each search. The average 1.4 MB of additional data per search that is used for pre-rendering an AMP page that the user may not visit is not trivial overhead for certain users with limited data plans. Most mobile users search a variety of things through search engines unaware of these hidden costs. Leaving pre-rendering as an option for users would still provide fast-enough user experience as we have shown, even if at the cost of an 'instant' web browsing experience.

7.2 Future work

There are a number of possible directions for future work. For starters, we conducted our tests for AMP with a single configuration (i.e., one location, using one type of connection, on a single device, etc.). Different network conditions may yield different results on either the performance impact of AMP and/or the relative contributions of its features. For

[9] Other browsers beyond Chrome have adopted the AMP carousel, including Firefox (https://addons.mozilla.org/en-US/firefox/addon/amp-browser/).

instance, a more challenged access link may amplify the benefits of AMP (e.g., fewer DNS resolutions) and QUIC.

In addition, our analysis relies on proxies of actual Quality of Experience, such as PLT, TTFB, and SI. More qualitative research on AMP user experience may help improve our understanding of the benefits of the different features of AMP. Also, expanding our definition of "users" to include adopters and developers could help us understand some the controversy around AMP.

Last, it may be interesting to also compare AMP to other techniques aimed at improving web browsing QoE for mobile users to draw more general lessons on how to improve the mobile web browsing experience.

8 RELATED WORK

There is a large body of prior work on measuring and improving mobile web performance. We review some of these efforts in the following paragraphs.

Client-side optimization. Content prefetching, pre-rendering and speculative loading systems can reduce the impact of high latency in networks on web QoE [26, 35, 50]. Predicting user browsing behavior is, however, challenging and can lead to wasted energy and data usage [38]. Google adopted prefetching and pre-rendering as a feature in Google Chrome to improve user-perceived web performance [5]. Webpages with *pre-render* link tags triggers pre-rendering, where the browser prepare a (hidden) page ahead of time so that the content can be presented to users as soon as they click the link. This feature, adopted by Chrome since 2011 [10], has an important role in making AMP's seemingly instantaneous page loading possible. From our analysis, AMP appears to prefetch only one – the top – AMP page on the assumption that users most commonly follow on the first item listed in a search result page.

Proxy-based acceleration. A number of research proposals and products have aimed to improve mobile web performance by dividing the load process between the client and a remote proxy server [21, 33, 34, 41, 47]. For instance, Amazon Silk and Opera Mini are web browsers for mobile devices with server-aided web speed boosting [13, 18]. While these approaches raise concerns on security and privacy, by resolving dependencies at a wired-connected proxy, they are able to reduce clients' PLT. AMP performance benefits from its CDNs in a similar manner.

Platform solutions. Beyond AMP, other projects and proposals try to alleviate the impact of network issues by altering how pages are written and served. For instance, companies with influential platforms have put significant effort into boosting performance of mobile web browsing in their platforms, particularly news, to capture user traffic. Apple has provided *Apple News* [9] for their mobile devices, and Facebook made *Instant Articles* [10] available through the Facebook apps since 2012. While these options can greatly improve the mobile experience, their benefits are confined to users of their particular apps.

Network optimizations. HTTP/2 [11] and SPDY [7] reduce PLT by allowing client browsers to multiplex all requests to an origin on a single TCP connection. HTTP/2 also allows servers to speculatively "push" objects before they are requested in order to improve loading time. QUIC, a protocol succeeding SPDY, is widely deployed and used by Google. While not strictly AMP, AMP leverages QUIC when possible.

9 CONCLUSION

We characterized Accelerated Mobile Pages (AMP), one of Google's most active and controversial efforts to enhance the mobile browsing experience. We found that deploying AMP greatly reduces the complexity of web pages, in large part thanks to restrictions set forth by the AMP standards, and that this reduction does translate in significant performance improvement on mobile users' quality of experience. We showed that the pre-rendering of AMP pages in Google search results is a key contributor to AMP's responsiveness, but that this may come with a significant overhead on mobile data consumed, depending on the rate at which prerendering is use and the frequency with which users' visit the prerendered sites. The controversy surrounding AMP and its impact on the health of the Web is ongoing. Our work contributes an independent analysis of AMP's performance benefit which, we acknowledge, is just one of the many factors determining mobile users' web experience.

ACKNOWLEDGMENTS

We thank our shepherd Aruna Balasubramanian and the anonymous reviewers for their thoughtful feedback, and the AquaLab group members for their support. We are particularly grateful to Malte Ubl, creator and tech lead of the AMP project at Google, whose detailed comments have significantly improved this work. This research was partially supported by NSF grant CNS-1619317.

REFERENCES

[1] 2004. Selenium. https://www.seleniumhq.org.
[2] 2006. Speed Test. http://www.speedtest.net/.
[3] 2008. Network Diagnostic Tool. http://www.measurementlab.net/run-ndt/.
[4] 2008. WepPageTest. https://www.webpagetest.org.
[5] 2011. Chrome Prerendering. https://www.chromium.org/developers/design-documents/prerender.
[6] 2012. QUIC, a multiplexed stream transport over UDP. https://www.chromium.org/quic.
[7] 2012. SPDY Protocol - Draft 3. https://www.chromium.org/spdy/spdy-protocol/spdy-protocol-draft3.

[10] This has been disabled by 2017.

[8] 2012. SpeedIndex. https://tinyurl.com/c77w9j4.

[9] 2015. Apple News. https://www.apple.com/news/.

[10] 2015. Facebook Instant Articles. https://instantarticles.fb.com/.

[11] 2015. Hypertext Transfer Protocol Version 2 (HTTP/2). https://tools.ietf.org/html/rfc7540.

[12] 2017. Cloudflare Ampersand. https://www.cloudflare.com/website-optimization/ampersand/.

[13] Amazon. 2017. What is Amazon Silk? https://tinyurl.com/jhjgnxp.

[14] Amelia Andersdotter, Daniel Appelquist, Alice Bartlett, Andrew Betts, Ada Rose Cannon, Kaelig Deloumeau-Prigent, Terence Eden, Alberto Elias, Patrick Hamann, Jeremy Keith, Zach Leatherman, Ethan Marcotte, Mark McDonnell, Rogier Mulhuijzen, Mark Nottingham, Natasha Rooney, Yuya Saito, Jed Schmidt, Steve Souders, Léonie Watson, and Estelle Weyl. 2018. A letter about Google AMP. http://ampletter.org.

[15] Paul Bakaus. 2017. Why AMP Caches exist. https://tinyurl.com/yaxe7a5l.

[16] David Besbris. 2015. Introducing the Accelerated Mobile Pages Project, for a faster, open mobile web. (October 2015). Google: Official Blog.

[17] Enrico Bocchi, Luc de Cicco, and Dario Rossi. 2016. Measuring the Quality of Experience of Web users. *SIGCOMM Comput. Commun. Rev.* 46, 4 (2016). October.

[18] Andreas Bovens. 2015. Opera Browsers, Modes & Engines. https://dev.opera.com/articles/browsers-modes-engines/.

[19] Jake Brutlag. 2009. Speed matters for Google web search.

[20] Michael Butkiewicz, Harsha V. Madhyastha, and Vyas Sekar. 2011. Understanding Website Complexity: Measurements, Metrics, and Implications. In *Proc. of IMC*.

[21] Michael Butkiewicz, Daimeng Wang, Zhe Wu, Harsha V. Madhyastha, and Vyas Sekar. 2017. Klotski: Reprioritizing Web Content to Improve User Experience on Mobile Devices. In *Proc. of ACM SIGCOMM*.

[22] Christine Chun. 2018. Google AMP yields 600% increase on mobile site page load speed. https://tinyurl.com/y7kw3am5.

[23] Corbin Davenport. 2018. Opinion: Google AMP is still confusing, and it's not getting any better. https://tinyurl.com/yc9aax7j.

[24] Eric Enge. 2018. AMP-lify Your Digital Marketing in 2018. https://tinyurl.com/yad5e7oq.

[25] Ericsson. 2016. Ericsson mobility report. https://tinyurl.com/gmnezg6.

[26] Li Fan, Pei Cao, Wei Lin, , and Quinn Jacobson. [n. d.]. Web Prefetching Between Low-Bandwidth Clients and Proxies: Potential and Performance.

[27] Alex Fischer. 2017. What's in an AMP URL. https://developers.googleblog.com/2017/02/whats-in-amp-url.html.

[28] Rudy Galfi. 2018. AMP stories: Bringing visual storytelling to the open web. https://tinyurl.com/yckkdouw.

[29] Scott Gilbertson. 2017. Kill Google AMP before it kills the web. https://tinyurl.com/kv4djy7.

[30] ITU. 2016. *Measuring the Information Society Report 2016*. Technical Report. International Telecommunication Union.

[31] Arash Molavi Kakhki, Samuel Jero, David Choffnes, Cristina Nita-Rotaru, and Alan Mislove. 2017. Taking a Long Look at QUIC. In *Proc. of IMC*.

[32] Javad Nejati and Aruna Balasubramanian. 2016. An In-Depth Study of Mobile Browser Performance. In *Proc. WWW*.

[33] Ravi Netravali, Ameesh Goyal, James Mickens, and Hari Balakrishnan. 2016. Polaris: Faster Page Loads Using Fine-grained Dependency Tracking. In *Proc. of USENIX NSDI*.

[34] Ravi Netravali, ANirudh Sivaraman, Somak Das, Ameesh Goyal, Keith Winstein, James Mickens, and Hari Balakrishnan. 2015. Mahimahi: Accurate Record-and-Replay for HTTP. In *Proc. USENIX ATC*.

[35] Venkata N. Padmanabhan and Jeffrey C. Mogul. 1996. Using Predictive Prefetching to Improve World Wide Web Latency. In *Proc. of ACM SIGCOMM*.

[36] Feng Qian, Subhabrata Sen, and Olivier Spatscheck. 2014. Characterizing resource usage for mobile web browsing. In *Proc.. of ACM MobiSys*.

[37] Advanced Web Ranking. 2018. CTR study. https://www.advancedwebranking.com/ctrstudy/.

[38] Lenin Ravindranath, Sharad Agarwal, Jitendra Padhye, and Christopher Riederer. 2013. Give in to Procrastination and Stop Prefetching. In *Proc. of HotNets*.

[39] Vaspol Ruamviboonsuk, Ravi Netravali, Muhammed Uluyol, and Harsha V. Madhyastha. 2017. VROOM: Accelerating the Mobile Web with Server-Aided Dependency Resolution. In *Proc. of ACM SIGCOMM*.

[40] Aakash Sahney. 2018. Bringing the power of AMP to Gmail. https://tinyurl.com/y9yv2trk.

[41] Ashiwan Sivakumar, Shankaranarayanan Puzhavakath Narayanan, Vijay Gopalakrishnan, Seungjoon Lee, Sanjay Rao, and Subhabrata Sen. 2014. PARCEL: Proxy Assisted BRowsing in Cellular Networks for Energy and Latency Reduction. In *Proc. ACM CoNEXT*.

[42] Joel Sommers. 2018. On the Characteristics of Language Tags on the Web. In *Proc. of PAM*.

[43] James Titcomb. 2016. Mobile web usage overtakes desktop for first time. https://tinyurl.com/zsuasva.

[44] Malte Ubl. 2018. Improving URLs for AMP pages. https://tinyurl.com/yagguvs4.

[45] Malte Ubl. 2018. Standardizing lessons learned from AMP. https://tinyurl.com/y8qda2o4.

[46] UK Office of Communication (Ofcom). 2016. *The International Communication Market Report*. Technical Report. Ofcom.

[47] Xiao Sophia Wang, Arvind Krishnamurthy, and David Wetherall. 2016. Speeding Up Web Page Loads with Shandian. In *Proc. of USENIX NSDI*.

[48] Zhiheng Wang. 2018. Using page speed in mobile search ranking. https://tinyurl.com/y8klhhg4.

[49] Zhen Wang, Felix Xiaozhu Lin, Lin Zhong, and Mansoor Chishtie. 2011. Why are Web Browsers Slow on Smartphones?. In *Proc. of HotMobile*.

[50] Zhen Wang, Felix Xiazhou Lin, Lin Zhong, and Mansoor Chishtie. 2012. How Far Can Client-Only Solutions Go for Mobile Browser Speed?. In *Proc. WWW*.