Demystifying Content-blockers: Measuring their Impact on Performance and Quality of Experience

Ismael Castell-Uroz, Rubén Sanz-García, Josep Solé-Pareta, and Pere Barlet-Ros

***Abstract*—With the evolution of the online advertisement and tracking ecosystem, content-blockers have become the reference tool for improving the security, privacy and browsing experience when surfing the Internet. It is also commonly believed that using content-blockers to stop unsolicited content decreases the time needed for loading websites. In this work, we perform a large-scale study on the actual improvements of using content-blockers in terms of performance and quality of experience. For measuring it, we analyze the page size and loading times of the 100K most popular websites, as well as the most relevant QoE metrics, such as the Speed Index, Time to Interactive or the Cumulative Layout Shift, for the subset of the top 10K of them. Our experiments show that using content-blockers results in small improvements in terms of performance. However, contrary to popular belief, this has a negligible impact in terms of loading time and quality of experience. Moreover, in the case of small and lightweight websites, the overhead introduced by content-blockers can even result in decreased performance. Finally, we evaluate the improvement in terms of QoE based on the Mean Opinion Score (MOS) and find that two of the three studied content-blockers present an overall decrease between 3% and 5% instead of the expected improvement.**

***Index Terms*—Content-filtering, adblock, advertisement, web tracking, performance, page size, loading time, QoE, Speed Index**

I. Introduction

THE use of content-filtering tools has seen an exponential increase since the initial development of AdBlock Plus [1] in 2005; one of the most popular and widely used adblockers. It was one of the first attempts to improve user's privacy and browsing performance against the most prevalent problem surfing the web at the time; the increasing amount of intrusive and malicious advertisements spreading over the Internet.

With time, the Internet has grown and some alternatives often more specific (e.g. JavaScript, Flash or third-party blockers) appeared to address the same problem. Unfortunately, such tools usually affect the usability of websites, making some of them even inaccessible. For that reason, general content-blockers have become the *de facto* solution to deal with advertising and user tracking practices.

As shown in [2], one of the main motivations to use content-filtering systems is to improve the browsing experience. Almost one third of the users that install a blocking system do it to increase the browsing performance. Blocking advertisements and tracking systems are believed to significantly reduce the bandwidth used, improving the website loading time and, thus, the overall Quality of Experience (QoE).

I. Castell-Uroz, R. Sanz-García, J. Solé-Pareta and P. Barlet-Ros are with the Broadband Communications Research Center of the Universitat Politècnica de Catalunya, Barcelona, Spain (e-mail: ismael.castell@upc.edu).

In this paper, we present a comprehensive study of the advertisement and tracking ecosystem with special emphasis on the performance and QoE improvements resulting from using content-blockers when surfing the Internet. Some previous works have analyzed and compared the effectiveness of existing content-blockers in terms of blocking accuracy [3]–[6]. However, very few of them [7], [8] have tried to analyze their impact in terms of browsing performance. To the best of our knowledge, this is the first work to evaluate the actual performance and QoE improvements of different content-blockers with a large and diverse set of websites.

In particular, we measure the loading time and page size when visiting the top 100K sites according to the Alexa list [9]. We also revise the most relevant QoE metrics, such as the *Speed Index*, *Cumulative Layout Shift* or *Time to Interactive* for the subset of 10K top most popular websites. We compare three different blocking approaches; *advertisement blocking*, *tracker blocking* and *generic content blocking*. For this purpose, we developed ORM [10], an open-source highly parallel network measurement system that loads every website using one of the most relevant content-blockers and compares their performance.

Regarding performance, we found that, although we can observe some improvements in terms of effective page size, the results do not directly translate to gains in loading time. Moreover, in some cases, there could even be a decrease in performance, especially in small and lightweight websites. Similarly, our results show that using content-blockers can slightly increase the overall browsing quality of experience, but for fast loading websites the extra layer introduced to preprocess and clean the website can introduce some delay to visualize the website content. The measurement system and methodology proposed in this paper can also be useful for network and service administrators to evaluate the web performance observed by their users.

This paper is an extension of the work presented at [11]. The new contributions of this paper include:

- An extensive study of the impact of content-blockers on the perceived Quality of Experience (QoE) according to multiple indicators, such as the *Speed Index* and the *Mean Opinion Score*.
- A detailed analysis of the actual page size reduction with respect to the number of resources blocked by each content-blocker.
- Extended evidence that confirms that improvements on the overall performance when using content-blockers do not always translate to increased Quality of Experience.

- The implementation of a new content-blocker plugin (Deep Tracking Blocker) inspired by the findings of this work, and its comparison with the other studied plugins.[1]

The rest of the paper is organized as follows. Section II provides the necessary background while Section III presents the related work. Sections IV and V present and analyze the results in terms of performance and quality of experience respectively. Section VI discusses the reasons behind the performance variations found in this work and proposes alternatives to improve the results. Lastly, Section VII concludes the paper.

II. Background

Advertisements have been used as a way to pay for online expenses as well as for getting benefits from online resources since the foundation of the Internet. At first, the advertisement ecosystem was very simple and owners tried to add some advertisements to monetize each site. Soon, the difficulty to attract user attention became relevant within advertisers [13]. To be able to stand out in such an environment, companies started to modify their advertisements with sounds and vivid colors, using pop-ups and other intrusive methods to reach their users. Lohtia et al. [14] studied the impact of those methods on advertisement performance and user perception. Such an advertisement environment became a nuisance for some of the users, being the main motivation for the creation of the first **adblockers**, a browser content-filtering plugin that tries to block all the advertisement being shown in a website.

With the evolution of the Internet and the languages used to develop it, companies started to use profiling mechanisms (e.g. HTTP cookies) over their users to perform targeted advertisement campaigns and to support their services. These primal web tracking techniques were more or less harmless and very simple to avoid. Nowadays companies use much more complex techniques (e.g. fingerprints, supercookies), collecting data not only from their own sites but from other apparently non-related websites (third-party trackers). This permits to extremely refine the information collected about the opinions and preferences of their users, and raises serious privacy concerns. Consequently, in the past few years many countries have created new regulations in order to improve the privacy of their citizens. Some of those regulations, such as the European GDPR [15], force websites to obtain the explicit user content to collect their personal data. However, it is extremely difficult to ensure online privacy as many collection systems are hidden and transparent to the user. In [16] Bujlow et al. show a summary of the different mechanisms that can be used to track users and correlate their information.

To improve their privacy users started to use the so-called **tracking blockers.** These plugins try to do the same than adblockers do with advertisements but with the tracking systems included in websites. Usually both of them, *adblockers* and *tracking blockers*, use custom databases or filter lists to distinguish between safe and non-safe content. Two of the most important filter lists are EasyList [17] and EasyPrivacy [18], each one of them dedicated to block advertisements and tracking methods respectively. These lists are maintained by the community and used by some of the most popular content-filtering extensions on the market.

Since advertisement blocking and tracking blocking are very similar, adblockers started to introduce at some level the possibility to also block web tracking mechanisms. Note that many online publishers have become part of the *acceptable ads program* [19], which allows them to avoid being blocked by some of the adblockers if their advertisements follow specific guidelines that ensure them to be less intrusive.

Recently, a new type of content-filtering is gaining traction; the generic **content blocker**. This blocker category tries to intercept not only advertisements and trackers, but other unneeded resources or security threats that could be detected in the website loading process, like for instance Cross-Site Scripting (XSS).

III. Related Work

Krishnamurthy et al. [3] was one of the first to study the impact of adblockers and other privacy protection mechanisms in web browsing. The study included measurements of the impact on page functionality and quality when using such protection mechanisms. In [5] Wills et al. makes a comparison between a set of adblockers and tracker-blockers, studying the success rate of their methods to block the content of different third-party trackers. Traverso et al. [7] study the behave of seven different content-filtering plugins within a set of 100 specific websites classified in different popular categories. Studies both, the tracker blocking success rate and the performance gains. Mazel et al. [4] compares 14 different tracking protection measures, including content-filtering plugins as well as other approaches like javascript blockers or machine-learning based blockers. The comparison is not only done for blocked content but on the differences between the discovered content and their impact in website usability.

In [8], Newman et al. explore the quantity of resources blocked by AdBlock Plus and relate them with an improvement in site loading latency, but at the expense of an increase of the Above-the-Fold (AtF) latency (time to load only the visible elements of the website). To evaluate the consequences of their findings they conduct an experiment on Amazon Turk. They find that users prefer a short AtF instead shorter total loading latency. On the other hand, Zach et al. in [20] demonstrate how the presence of advertisements could not only impact the quality of experience of the browsing session itself, but also the quality of experience of the browsed services such as for instance on-demand video services.

Pujol et al. [21] studied the usage of adblockers in a European ISP and found that 22% of the most active users used an adblocker. Malloy et al. [22] measured the percentage of adblockers users in the U.S. and inferred that about 18% was using it. The constantly increasing adoption rate and loss of revenues made online publishers to start using different techniques to try to bypass adblockers. In [23] Iqbal et al. studied the anti-adblocking ecosystem and found it to be continuously increasing.

In [24] Lerner et al. studied the prevalence of tracking methods used for both, targeted advertising as well as other

[1]The new plugin can be freely downloaded from the Chrome Web Store. The source code is publicly available at [12].

purposes, over the Internet. They also studied it's increase over time using the Wayback Machine. They found that almost 70% of websites use some type of tracker, enforcing the use of privacy protections like trackers blockers.

Shiller et al. [25] tried to measure the impact of advertisement blocking on the quality of content created, and revenues of online publishers.

Butkiewicz et al. performs in [26] a series of experiments to understand the complexity of the current website ecosystem. Although the research does not have direct relation to content-blockers, they explore several terms very related to this research like resources loaded, number of different origins between them, the page sizes or the website loading times.

From the user's perspective there are different reasons to use a content-filtering plugin. Mathur et al. [2] studied the different content-filtering group adoption (*adblocker*, *tracker blocker* or *general content blocker*) as well as the different and common motivations to use all of them. They found *adblockers* to be the most prevalent blocker system (51.2%), with *general content blocker* next (20.5%) and *tracker blocker* in the last position (8.4%). The primary reason to use an *adblocker* or a *general content blocker* (the two most adopted solutions) was to improve the user experience (85-89%), and between the most common given motivations was **speedup loading times** with a 33.1% of acceptance. Other reasons were to improve privacy (mainly by means of *tracker blockers*) and only a small percentage used blockers to improve security.

In this paper, we analyze whether there is an actual improvement in browsing performance when using content-filtering tools, as one third of the plugin users seem to think. To the best of our knowledge, there are only two previous works that tried to address similar questions [7], [8]. Traverso et al. explored in [7] the protection and performance improvement by means of the Time-to-First-Paint (TTFP) using *AdBlock Plus* with a population of only 100 websites (most of them belonging to the same country). In [8], Newman et al. used Amazon Mechanical Turk [27] to study the relationship between loading time and quality of experience with a set of 1000 users, who analyzed a sample of 10 pages each from an overall population of 965 web pages.

In contrast, our analysis is based on measurements of more than 20 different indicators within a population of 100K websites, using a diverse set of content-blockers focused on different purposes. Moreover, we measure both, the real network performance improvement as well as the quality of experience associated with this improvement.

IV. Browsing Performance

A. Methodology and evaluation

To explore the real browsing performance gain introduced by content-filtering plugins we will explore the principal parameters that can represent an increase in performance; effective page size and page loading time. To be able to generalize our results we have to accomplish several conditions.

1) **Test population**: The number of websites to browse has to be big enough to be able to extrapolate the results to a bigger population, and the selected websites should be representative enough of an usual browsing session.
2) **Browsing experience**: The website has to be loaded with each content-filtering plugin exactly as it would be done if accessed by a real user. This will avoid websites to detect automation scripts that would possibly make them not to load all the resources.
3) **Loading interval restrictions**: The same website loading process using different content-filtering plugins should be performed in a small time window to avoid whenever possible different temporal conditions (e.g. rush hours, periodic maintenance, etc).
4) **Experiments repetition**: The number of repetitions should allow us to discard possible non-reliable values, especially in loading time experiments, where external conditions can change from one execution to another.
5) **Browsing completion**: We have to measure all the resources being loaded, even third-party ones or resources loaded dynamically and not included in the website code.

1) Test population: For the test population we decided to use the top 100K most popular websites according to the Alexa list [9]. Scraping the most popular websites give us a good approximation of what a real user would access. Alexa's list is part of the Amazon ecosystem and has been used in several publications in the past (e.g. [3], [28]). Note that we only examine the homepages of each one of the websites included in the list but none of the links available within them.

2) Browsing Experience: We developed ORM [10], our own highly parallelized system making use of *Selenium* [29] for the automation process. This allows us to run real browsers such as *Mozilla Firefox* or *Google Chrome* that will present identical results to those a user would normally get. This also permits us to customize all the experiments using browser settings and network headers. Selenium has been used in the past for automation of research experiments in several different topics (e.g. [28], [30]).

The experiments are done crawling the 100K websites using one plugin from each group presented in Section II; **AdBlock Plus** (version 3.5.2) as an *adblocker*, **Ghostery** (version 8.4) as a *tracker blocker* and **uBlock Origin** (version 1.19.6) as a generic *content blocker*. Each one of them uses their own filtering engine based on JavaScript regular expressions, caching as well as serialization of their filter internal representation to match filters with URLs. Moreover, the three of them use the same filter list format, which is compatible with the most popular filter lists *EasyList* [17] and *EasyPrivacy* [18]. All of them have versions available for the most common Internet browsers like *Google Chrome* or *Mozilla Firefox*. A summary of the differences between the plugins is shown in Table I.

AdBlock Plus [1] is by far the most popular adblocker in the market. It can block both, advertisements and tracking systems, using *EasyList* and *EasyPrivacy*, although EasyPrivacy is disabled by default. AdBlock Plus is a supporter of the *acceptable ads program* [19], allowing advertisements if they follow specific guidelines.

Ghostery [31] was initially developed as a tracker blocker, but evolved into a combination of adblocker and tracker blocker. Unlike AdBlock Plus, Ghostery uses its own main-

TABLE I
CONTENT-FILTERING PLUGINS COMPARISON

	AdBlock Plus	Ghostery	uBlock Origin
Ads blocking	Yes	Yes	Yes
Allow some ads	Yes	Yes	No
Track blocking	Not by default	Yes	Yes
EasyList	Yes	Unknown	Yes
EasyPrivacy	Available	Unknown	Yes
Others	No	Private Database	Additional lists

tained filter list, not publicly available, to decide whether to block a content or not. It also has its own monetizing program different from the acceptable ads program. Instead permitting some advertisements to pass through, Ghostery blocks all the original advertisements but introduces some of their own non-intrusive advertisements.

Lastly, *uBlock Origin* [32] is one of the most used generic content-blockers in the web. Its functionality is very similar to AdBlock Plus, and can block advertisements (EasyList) as well as tracking systems (EasyPrivacy). It includes by default many other additional filter lists (e.g: regional or uBlock's own filter lists) to block other nuisances, such as *resource abuse* or unneeded cosmetic elements. Unlike AdBlock Plus, uBlock Origin does not permit any kind of advertisement and blocks also other privacy threats like Cross Site Scripting (XSS) loaded resources.

We decided to use *Chromium* (version 76.0.3809.100), the open-source version of *Google Chrome*, as the baseline to compare all the observed information. We tested as well other alternatives also based on the *Chromium* font code such as *Microsoft Edge* to discover possible performance differences. However, as they use the same internal systems (e.g. *Blink* or *V8 JavaScript* engines) our tests presented only minimal differences (0.36% variance in performance and less than 90ms for QoE parameters in average). *Firefox*, which uses a different engine, was also tested although we also found performance to be almost equivalent. However, Firefox does not provide an easy way to parse all the resources being loaded by a website, forcing us to develop our own DOM parser to search for embedded resources. Thus, we opted for Chromium as it is a good representation of most available browsers and it permits us to easily get information about all the network communications being performed as well as to load the needed plugins in an straightforward way.

Regarding the plugins, we decided to use the default settings for all of them, as usually users do not change them after installation. Note that in AdBlock Plus the acceptable ads program list is enabled by default and we leave it this way. The only modification introduced is adding EasyPrivacy subscription to AdBlock Plus. We have two main reasons for those decisions. First, we want to test the three plugins in equal conditions, blocking both advertisements and web tracking mechanisms. Secondly, as observed in [21], AdBlock Plus users usually activate EasyPrivacy list even if it is disabled by default, but do not disable the acceptable ads program.

All the experiments where executed in a server operating an Ubuntu 16.04 LTS over an Intel Xeon E5-2697 (18 cores, 36 threads) and 32GB of memory. The network connection used pertains to a high speed academic network from Spain. We found memory to be the limiting factor as the average CPU and network usage was consistently below 50%.

3) Loading interval restrictions: Specially for loading time experiments we have to assure that the measures are taken for all the plugins in a short period of time, to avoid changing network conditions. Our developed system opens the same website with 4 different browsers in parallel, three of them loading the corresponding plugin and the last one with a vanilla browser.

4) Experiments repetition: Due to the dynamic nature of the Internet, where many content is selected in real-time to customize the browsing experience, we can not force websites to load always the same content. Consequently, there would always be websites presenting differences between the Vanilla and the plugin-enabled versions. To minimize the effect that this could have on the results, we focus on the statistical differences working with the highest possible volume of measures, and their average values. This allows us to find trends as well as to isolate special cases.

Our page size experiments compute the *effective page size*, including the size of all the files loaded by the website. To this end, it is enough to scrap the resources being loaded by the browser once for each of the content-filtering plugins, and once more using a vanilla. This gives us enough information to compare the size improvement using each of the plugins.

For the loading time experiments the same website is opened in parallel by the four browsers 5 times consecutively, having a timeout setting of 30 seconds, usually bigger enough to load all the resources given the network characteristics. Between each repetition the browsing cache as well as the cookies are deleted to obtain a clear browsing experience.

On top of that, if one of the browsers is unable to get 5 measures of the same website (e.g. network issues, hardware issues, server miss-configurations, server performance issues), all the measures are discarded, as it is not possible to compare them reliably. To avoid interference from external network circumstances we have discarded noisy samples where the 5 taken measures differed significantly in their request time. To this end we used the *interquartile range* measure. This permitted us to discard non-reliable samples (0.66%) due to excessive changes in network conditions during the measurements. This methodology makes the system robust against the dynamic nature of the website. If the included dynamic content always slows down the loading process it is already considered in our measurements. On the contrary, if there are only punctual issues, it will be discarded by the interquartile range deviation. We have computed the average for the rest of the 5 observations to obtain an approximation of the loading latency of each website. With this system, from the initial population of 100K websites we ended up scrapping successfully a total of 80.974 websites. All the measures were taken in the period May-July of 2019. The resulting dataset is publicly available at [33].

5) Browsing completion: Nowadays most websites include script files that load resources dynamically. Moreover, almost all of them are obfuscated or minifyed; a process that tries to reduce the size of the script files and improve the loading

TABLE II
PERFORMANCE TIMING API

Grouping		Event	Stage
Total loading time		startTime unloadEventStart unloadEventEnd	Prompt for unload
		redirectStart redirectEnd	Redirect
		fetchStart	AppCache
		domainLookupStart domainLookupEnd	DNS
		connectStart secureConnectionStart connectEnd	TCP
	Request time	requestStart responseStart	Request
		responseEnd	Response
		domInteractive domContentLoadedEventStart domContentLoadedEventEnd domComplete	Processing
		loadEventStart loadEventEnd	Load

time by removing white-spaces, break-lines and shortening the names of the variables. This prevents to get a list of the resources being downloaded exploring the code in the traditional way. To solve it we make use of Google's *DevTools Protocol* to get access to all the resources included inside the network communications executed by the browser. We do it by enabling the logging capabilities of the browser and extracting the information directly from the on-memory network logs in real time.

On the other hand, privacy policies commented in Section II force websites to present banners to obtain the consent of the user prior to collecting personal information. This can have an impact in our measurements as tracking content should be avoided (according to privacy regulations) until obtaining the user consent. However, a recent study [34] concludes that about 70% to 80% of websites do not modify their tracking behavior based on those privacy banners. Thus, the impact in our experiments should be limited to only a small subset of websites. On the other hand, another study has shown [35] that websites content greatly differs between the homepage and the internal sites. However, the same research [35] found that homepages are prone to contain more web tracking than internal websites. Thus, we expect the highest performance gain obtained by using a content-blocker to be obtained in the website homepage.

As for the loading time metrics, we use the *Performance Timing* API [36], defined by the W3C and supported by the majority of the current browsers, to extract the information of all the loading events each website produces. In Table II it is shown the events produced by the *Performance Timing* API in execution order when a website loads. Using the time difference between those events we can compute different measures. The total loading time collects all the process including the time to unload the current website as well as the time spent in redirections. As we only want to account the time difference between loading the website with and without

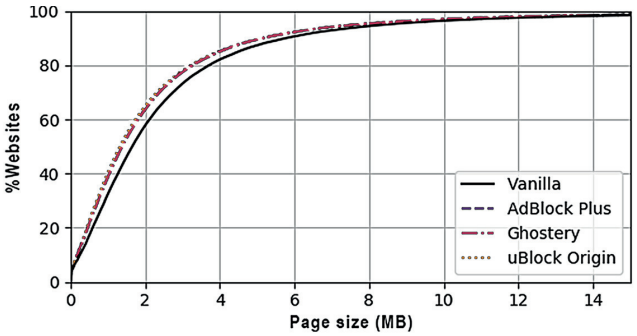

Fig. 1. **Page size distribution (CDF)**: All the content-blockers present slight improvements in page size due to the resource blocked prior to being downloaded.

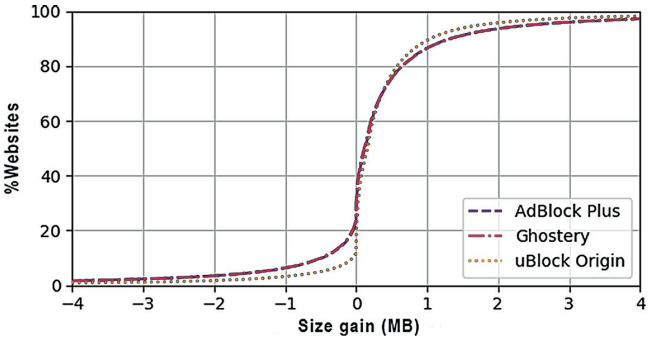

Fig. 2. **Page size gain**: More than 60% of websites present similar size gains thanks to content-blockers. AdBlock Plus and Ghostery present overhead for almost 20% of them while in uBlock Origin is only present in 10%.

a content-filtering plugin, we will be focusing our experiments on the request time, between the *requestStart* event and the *domComplete* event. The former is executed just before starting any resource request, while the latter is generated when the *Document.readyState* becomes *complete*. This is equivalent to actively inspecting the JavaScript *window.onLoad* call, and ensures that all the content, including scripts, have finished loading. We purposely avoid accounting undesired variability like the time spent unloading the previous website, process mainly dependent on the computer CPU speed and memory.

B. Page size comparison

In this section, we analyze the website loading performance, when using different content-filtering tools, in terms of the *effective page size* once loaded all its resources. This includes not only the resources loaded by the website itself, but also the dynamic content loaded from third-party domains called inside it. Intuitively, a vanilla browser will load all the resources included in the website, while a browser with a content-blocker will only load those that are not being blocked. Note that many websites include dynamic content that can differ each time the website is loaded. Moreover, some content embedded in the website can only be filtered once already downloaded, not incurring in size savings.

Fig. 1 shows the cumulative distribution function (CDF) comparing the effective page size needed for loading the top 100K most popular websites. At first sight, we can see the

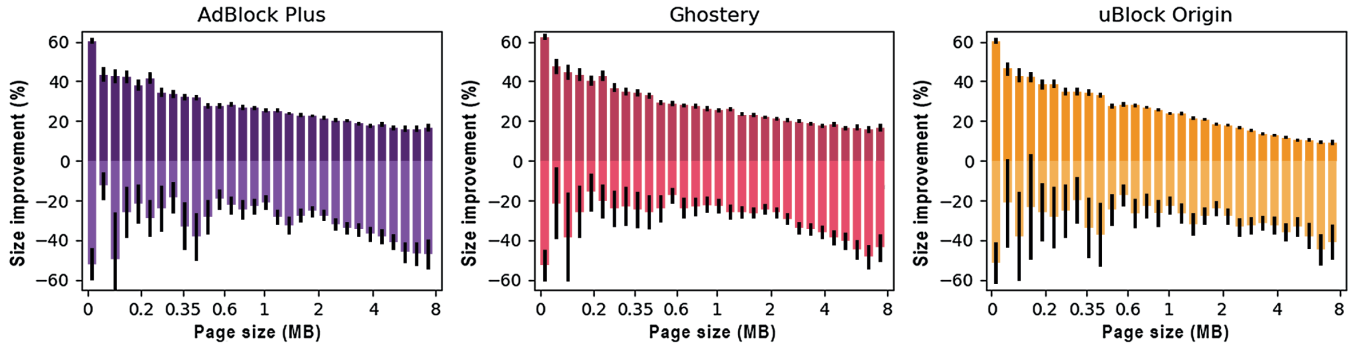

Fig. 3. **Size improvement/Page size**: Positive Y axis presents the average improvement for the subset of websites inside the given interval that actually presents an improvement. On the contrary, Negative Y axis presents the average overhead for the subset of websites presenting an overhead. Positive values include almost 85% of the collected samples.

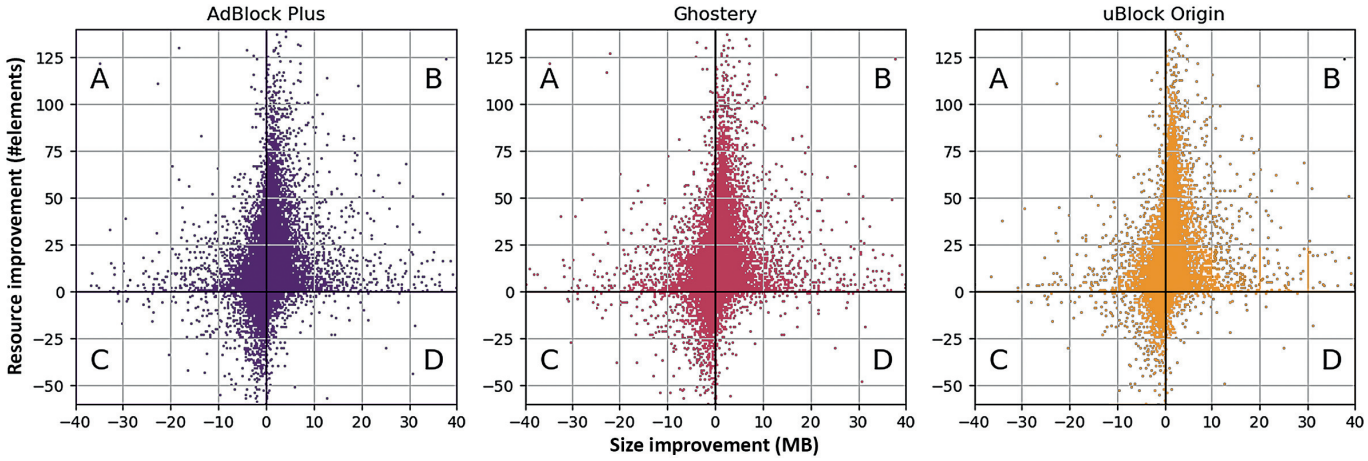

Fig. 4. **Resource improvement / size improvement**: Depicts the resource improvement between the vanilla browser and each plugin over the website size improvement. Quadrants B and C contain most of the samples (90%) and are the expected results. Both negative values or both positive values demonstrate that reducing the number of elements improves the page total size. Quadrants A and D are counter-intuitive as they break that relation. Carefully examining the data we found them to be the result of the inherent dynamic nature of the Internet, where some elements vary even in consecutive runs.

minimal difference between the three content-blockers. All of them present almost the same distribution, resulting in a small size improvement over the vanilla browser (10% to 20%).

Fig. 2 plots the absolute page size improvement, computed as the difference in megabytes with the original size of the website and all its resources. AdBlock Plus and Ghostery follow almost the same distribution, while uBlock Origin shows a slight improvement compared to the other tools. Note that not all websites benefit from using a content-blocker. There are between 10% and 20% of websites where the performance decreases when using it.

To explore these results further, we analyzed the relationship between the effective size gains and the total size in Fig. 3. As websites smaller than 4 megabytes represent more than 80% of the total population (Fig. 1), the page size is depicted in logarithmic scale. Positive values correspond to the average improvement obtained by the subset of websites where the use of a content-blocker results in size savings. For instance, the first interval corresponding to websites of only a few kilobytes, shows an average improvement of 60% for the subset of websites that have a page size improvement. On the contrary, negative values represent the average overhead for those websites where the use of a content-blocker results in an increase of the total size of the website. Note that, as shown in Fig. 2, the percentage of websites that benefit from using a content-blocker (80%-90%) is much larger than those where it introduces an overhead (10%-20%). Consequently, the number of instances with negative values is much smaller than the instances with positive values. This is also reflected as bigger interval errors present within the negative averages.

Fig. 3 shows a clear relationship between total page size and size improvement. This is not unexpected because every major advertisement engine imposes some policies defining not only the type of advertisements, but also limiting the placement and number of advertisements visible per site. Thus, for usual websites using only one or two of those engines there is an upper bound to the total size to be used for advertisements. This upper bound is comparatively larger for small websites than for larger websites, where the size improvement obtained blocking the ads will result in lower size improvements.

Fig. 4 analyzes the resource improvement (number of elements present in the vanilla that are not present in the plugin-enabled browser), and relates them to the website size improvement. Negative Y values indicate websites with a higher number of elements using the plugin than using the vanilla. The results are divided using the coordinate axes

TABLE III
BLOCKED RESOURCES / SIZE IMPROVEMENT DISTRIBUTION

Quadrant	AdBlock Plus	Ghostery	uBlock Origin
A	3.81%	6.21%	3.45%
B	84.73	81.62%	86.27%
C	3.52%	4.4%	2.63%
D	0.17%	0.15%	0.08%
Origin (0, 0)	7.76%	7.6%	7.56%

TABLE IV
ANOVA: SIZE DIFFERENCE CORRELATED WITH RESOURCE TYPE

Source value	Sum of squares	Degree of freedom	F-ratio (%)	P-value(>F)
C(resource type)	$2.44x10^5$	11.0	748.118931	**0.0**
Residual	$1.82x10^6$	61284		

in four quadrants named from **A** to **D** in the graph. Table III summarizes the percentage of websites included in each quadrant for all the plugins. The results are very similar for the three content-blockers compared.

Most observations (≈84%) are situated in quadrant **B**, with positive values for both axes. This is the expected behavior, as it indicates a positive relationship between the number of blocked resources and website size improvements. Surprisingly, this relationship is not linear, and blocking more resources does not necessarily result in more size savings.

The figure also shows a small percentage of websites (≈3%) in quadrant **C**, the opposite case. The page size overhead is small and exploring a sample we found it to be mostly related to the dynamic loading process of the website. As commented before each time a website loads, the content varies slightly including new information depending on the third-party trackers being loaded. It is not rare that the same website loaded by the vanilla browser and a browser including a plugin presents some minor differences.

For the remaining two quadrants, **D** with only few observations (≈0.13%) is very intuitive. It would be very uncommon to have effective size gains if the number of resources increases. Unexpectedly, the number of observations in quadrant

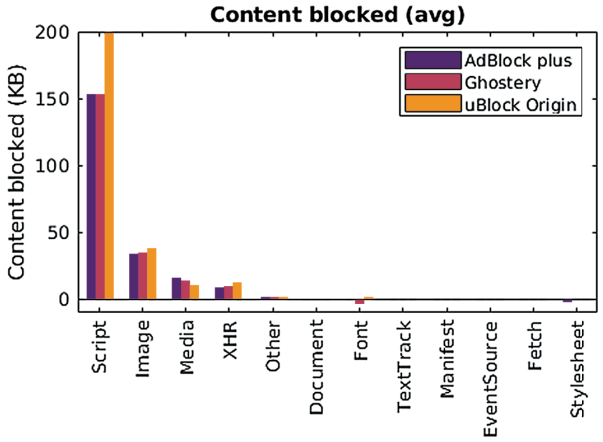

Fig. 5. **Average content blocked**: Almost 80% of the website size gain comes from blocking JavaScript resources. Size reduction thanks to image blocking is limited, as most blocked images are very small pictures used for browser fingerprinting.

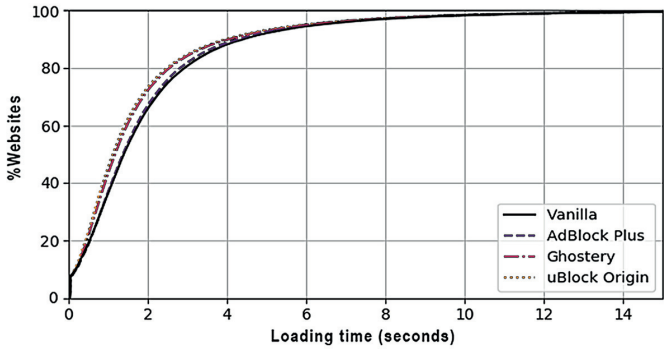

Fig. 6. **Request time (CDF)**: AdBlock Plus presents a loading time almost equal to the vanilla browser. On the other hand, Ghostery and uBlock Origin show some gain margins on the total Page Loading Time.

A is not negligible in some cases. Quadrant **A** depicts websites that load less resources but are bigger in the plugin-enabled browser than in the vanilla. To understand this counter-intuitive behavior, an ANOVA test was applied to determine the significance of the resource type and the difference in size within the websites pertaining to Quadrant **A**. The ANOVA results are provided in Table IV. Since the P-value is less than 0.05, the file type has a statistically significant effect on the size difference at the 95% confidence level. To discover the most relevant resource type between the 11 differentiated categories we also performed a *Tukey Test* to make pairwise comparisons between the means of each group while controlling for the total error rate. The results (Appendix A) show that image files are the main responsible for the increase in website size. We manually inspected a randomized sample of those image files to discover possible differences in the treatment given by the website once discovered a content-blocker. We did not find any evidence of the content being customized for plugin-enabled browsers. Once more, the difference in size was due to the dynamic content of the website, presenting slightly bigger advertisement images in comparison to the vanilla browser.

Fig. 5 presents the average size of blocked resources classified by type. *uBlock Origin* presents a slightly better blocking performance than *Ghostery* and *AdBlock Plus*. This difference is aligned with previous results, and it is explained by the extra contents being blocked by a generic content blocker. The size savings due to image files is small because most of them are tiny non-visible images dedicated to fingerprinting. This result is in par with the results found at [26].

C. Loading time comparison

Another metric that has an important role on the browsing performance is the time needed to load a website. If introducing a content-filtering plugin represents an important decrease in loading time, users can benefit not only from the privacy and security benefits brought by such plugin, but also from increased browsing speeds.

The loading time distribution resulting from our population of 100K websites is shown in Fig. 6. Approximately 90% of the websites have a loading time lower than 5 seconds, being also this range the one that benefits more from using

TABLE V
LIGHTHOUSE MOST IMPORTANT QUALITY OF EXPERIENCE METRICS

Metric	Description
Performance	Overall QoE performance score: (0.25 x (LCP + TBT) + 0.15 x (SI + FCP + TTI) + 0.5 x CLS)
SI (Speed Index)	Measures how quickly content is visually displayed during page load.
FCP (First Contentful Paint)	Measures how long it takes the browser to render the first piece of DOM content after a user navigates to the page.
LCP (Largest Contentful Paint)	Reports the render time of the largest image or text block visible within the viewport, relative to when the page first started loading.
FMP (First Meaningful Paint)	Measures when the primary content of a page is visible to the user. The raw score is the time in seconds between the user initiating the page load and the page rendering the primary above-the-fold content.
CLS (Cumulative Layout Shift)	Measures the sum total of all individual layout shift scores for every unexpected layout shift that occurs during the entire lifespan of the page.
FCP (First CPU Idle)	Measures how long it takes a page to become minimally interactive.
TTI (Time to Interactive)	Measures how long it takes a page to become fully interactive.
TBT (Total Blocking Time)	Measures the total amount of time that a page is blocked from responding to user input, such as mouse clicks, screen taps, or keyboard presses.

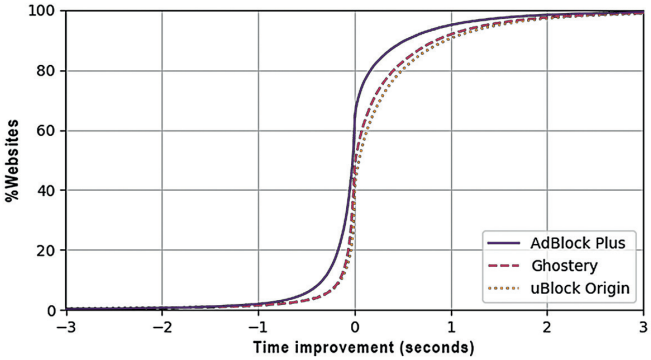

Fig. 7. **Loading time improvement**: Ghostery and uBlock Origin present similar results, improving the loading time in more than half of the websites and decreasing it on about 20% of them. AdBlock Plus only improves about 30% and on the contrary it introduces an overhead for almost half of them.

a content-blocker. Nevertheless, as in the case of page size, the difference is very small, with only 174 milliseconds of average gain. Note that the curve presented by AdBlock Plus is almost equal to the one of the vanilla browser.

Fig. 7 plots the CDF with the loading time improvement in seconds for each of the three content-filtering plugins. Unlike in the page size experiments, we can clearly see a difference between the three of them. Ghostery and uBlock Origin present an improvement in almost 50% of the websites, and a loading time increase in a small percentage. In contrast, AdBlock Plus improves the loading time in only about 30% of the websites, but degrades it in almost 50% of them.

To study this performance penalty, we computed the average improvement as a function of the total loading time of the website. Fig. 8 shows the results for each content-blocker. In the first column (less than 1 second), we can see that both Ghostery (-7%) and AdBlock Plus (-13%) introduce a decrease of performance. In the case of AdBlock Plus, this overhead is still present in the range between 1 and 2 seconds. Even uBlock Origin, while not suffering this performance penalty, experiments only a small improvement in the 1 second interval.

Considering the percentage of websites that load faster than 2 seconds (Fig. 6), we can observe that using one of the two firsts content-filtering plugins, increases loading time in more than half of the explored websites. This fact initially contradicts the usual assumption that using a content-filtering plugin improves the performance of the browsing session. Overall, thanks to slow loading websites, AdBlock Plus presents an average improvement of 53 milliseconds, while Ghostery and uBlock Origin have an average loading time improvement of 207 and 263 milliseconds, respectively.

Most content-blockers work comparing each URL loaded by the website to a list of patterns included in the black lists used to discard them. These lists can contain thousands or even hundred of thousands of different rules. Even very optimized algorithms inherently introduce an overhead to check in real time the resources being acquired by the browser. Thus, their usage will only represent an improvement in loading time if the time reduction gained by blocking some resources is higher than the time spent by the plugin checking all of them. Our results show that for very fast loading websites the overhead introduced by the plugin itself is not negligible.

V. QUALITY OF EXPERIENCE

To verify if our findings about browsing performance directly translate to the actual quality of experience perceived by the user, in this section we look at the most important QoE parameters available when using content-blockers.

A. Methodology

The methodology we followed to perform the QoE experiments is mostly the same as presented in Section IV-A. Browsing experience, loading interval restrictions, experiments repetition and browsing completion requirements remains exactly the same, except the loading timeout that we increased from 30 to 60 seconds. We also repeat the same set of plugins. However, taking multiple QoE measures for a website may take a few minutes, and doing so for multiple plugins and 5 times for each website greatly increases the time needed to

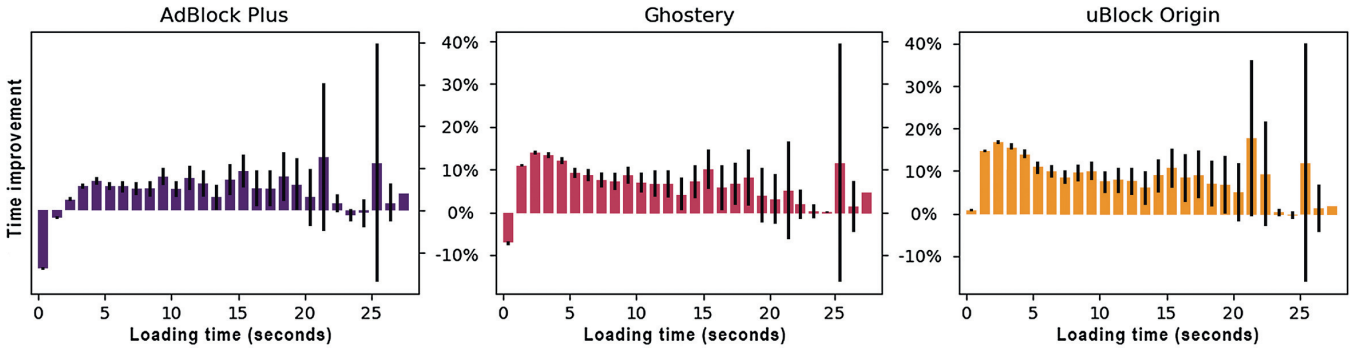

Fig. 8. **Improvement distribution**: Each interval represents the average page loading time improvement for the subset of websites with a total loading time inside the interval. For very fast loading websites the improvement is almost negligible or even negative.

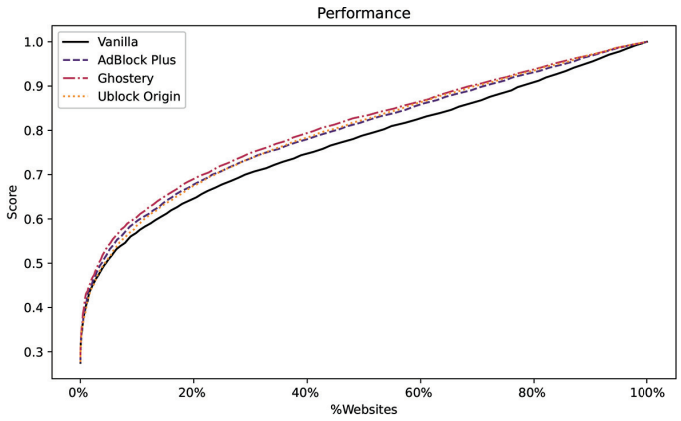

Fig. 9. **Lighthouse performance score**: Overall QoE score computed by Lighthouse. The best performer plugin is Ghostery, followed by AdBlock Plus and in the last position uBlock Origin, both of them very close.

TABLE VI
LIGHTHOUSE QUALITY OF EXPERIENCE METRICS AVERAGE

	Vanilla	AdBlock Plus	Ghostery	uBlock Origin
Performance	0.7728	0.7982	0.8076	0.7989
Speed Index	0.5842	0.6414	0.6614	0.6443
FCP	0.6437	0.6798	0.7065	0.6718
LCP	0.5163	0.5693	0.5868	0.5777
FMP	0.5967	0.6357	0.6513	0.6318
CLS	0.8739	0.8856	0.8774	0.8844
TTI	0.9444	0.9534	0.9572	0.9491
FCI (ms)	321,00	314,30	298,25	331,09
DOM Size	1512,47	1492,16	1485,69	1482,26
Redirects	592,61	553,80	523,48	558,64
RTT	27,90	39,4858	33,2413	44,6951
Max. RTT	167,04	115,6420	103,6021	121,4503
Throughput (Mb)	27,357	30,178	25,276	33,651
Tasks over 500ms	0,2924	0,2829	0,2680	0,2691
Resource Average	142,3857	97,3174	87,7849	86,4070

perform the measurements. Thus, in this section we limit the test population to the top 10K most popular websites. To collect the time events instead of using W3C defined *Performance Timing API* we make use of *Google's Lighthouse* [37].

Lighthouse is a module designed by Google to compute QoE metrics over a given website. It is available as a plugin for the most popular browsers, but for Google's Chromium, our selected browser, it is already available as part of the *Developer Tools*. Table V presents a subset of the most important metrics computed by Lighthouse with a description of each of them. In order to run Lighthouse inside ORM, our own developed collection tool, we modified it to open the browser with a debug port, load the corresponding content-blocker plugin, and pass the debug port to Lighthouse, which is in charge of loading the required website inside a new tab and computing the metrics.

B. Results

Fig. 9 shows the overall quality of experience score computed by Lighthouse. In version 6, the Lighthouse version used in this work, this performance responds to the formula presented at Table V. As shown in the figure, the plugin that results in the greatest improvement in terms of quality of experience is Ghostery, with AdBlock Plus and uBlock Origin in second and third place, respectively. The difference between these last two plugins in most points is very small. However, the results seem to indicate that there is a performance improvement equally present in all the revised websites. This initially contradicts our findings in Section IV where we found a subset of websites that present an overhead in the page loading time that can impact the QoE. To focus the lens on fast websites we decided to repeat our performance measurements study present in Fig. 8 using the Speed Index instead of the page loading time. Results are depicted in Fig. 10.

All the websites are split in subsets of 1 second depending on the time needed to show all the visual elements inside the viewport. Then, for each subset of websites the average improvement is computed and assigned as the interval value. As in Section IV, the first interval, containing websites with a total Speed Index of less than 1 second, shows a negligible or even negative improvement, representing an overhead for very fast loading websites. As expected, for the rest of the intervals there is an improvement over the vanilla browser, increasing proportionally to the time needed to load the portion of the website loaded inside the viewport. Nevertheless, more than 80% of the websites are included inside the first 5-second intervals, presenting an overall improvement lower than 20%.

Table VI presents the average of all the collected metrics classified per content-blocker for the 10K inspected websites.

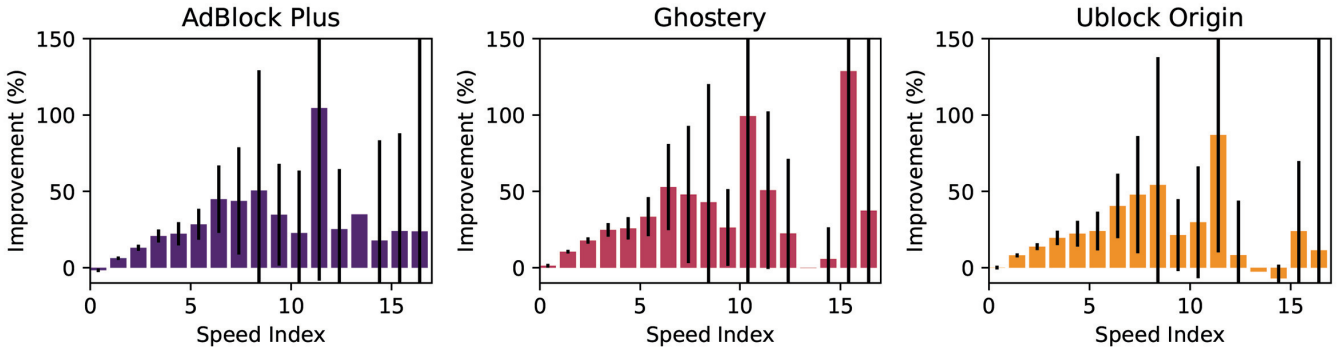

Fig. 10. **Lighthouse Speed Index improvement**: Each interval value corresponds to the average improvement of the Speed Index for the subset of websites on the vanilla browser with an initial Speed Index inside that interval. The firsts interval presents negligible or negative improvements for all the content-blockers.

The best result for each category is highlighted in grey color. Note that all the metrics are collected by the browser and not directly from the network link. Thus, the browser will only account for the traffic being loaded by itself, and not the real link status.

In average, all the performance related metrics show an overall improvement for all the plugins in comparison to the vanilla browser. Moreover, as already seen, *Ghostery* stays on top on almost all of them. On the other hand, all the content-blockers decrease the number of files loaded by the website, working as expected. Between all of them the tool blocking the highest number of resources is *uBlock Origin*. This is not unexpected because among all the analyzed plugins it is the most strict one, using numerous black lists in order to detect not only web trackers and advertisements, but also other nuisances between the files (e.g. Cross Site Scripting).

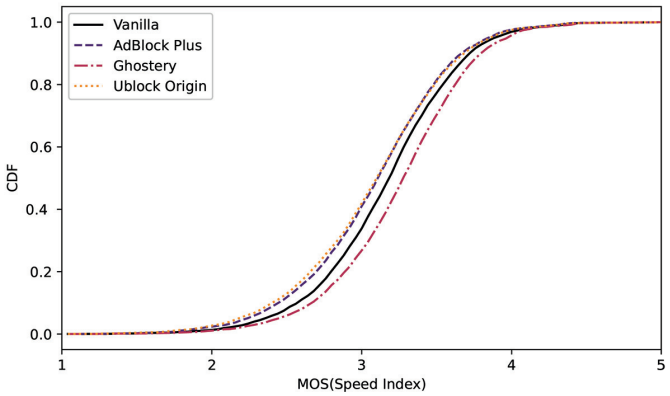

Fig. 11. **QoE Mean Opinion Score**: QoE modeling defined at ITU-T G.1030 [39] applied over the Speed Index. Ghostery presents an slight QoE increase over the vanilla. AdBlock Plus as well as uBlock Origin present both an overall decrease of QoE.

C. Quantifying the improvement

Our results show that content-blockers can help to increase the performance and quality of experience perceived by the user in many cases, but this improvement is almost always very small and even negative in some cases. However, determining if those small differences are perceptible to the user is very difficult. For very fast loading websites introducing an overhead can be irrelevant if the user does not perceive it. Similarly, slow loading websites may be too slow, even using content-blockers, to avoid user frustration.

To better understand the *noticeable* improvement we computed the Mean Opinion Score (MOS) model over the Speed Index obtained results, following the proposal at [38]. MOS is a standard defined by the ITU (ITU-T G.1030 [39]) to estimate end-to-end performance in IP networks. It defines the quality of experience as a value between 1 and 5, being 1 the worst possible value and 5 the best experience. It is based on the WQL hypothesis (the relation between the waiting time and the QoE follows a logarithmic linear scale), and it was originally applied to the page loading time (PLT). However, recent advances in the area of QoE research have found that correlating it with QoE measures is more appropriate than PLT. In particular, the study performed by Hoßfeld et al. in [38] found that applying it to the Speed Index (SI) leads to a very good approximation of the actual MOS values. The SI-driven QoE MOS is formulated as:

$$SQ_{WQL} = -a \cdot ln(SI) + b \tag{1}$$

Fig. 11 shows the CDF distribution for the 10K collected websites. Surprisingly, the MOS values for AdBlock Plus and uBlock Origin present worse results than for the vanilla, letting Ghostery as the only plugin with a noticeable performance increase. This difference between the QoE obtained metrics and the MOS is not directly observable in the previously obtained figures, where the global impression is an overall QoE improvement. Nevertheless, as expected, the difference is small (between 3% to 5% of websites). These results confirm our previous findings that using plugins to improve the browsing performance has only a relatively small impact on the overall experience, and depending on the selected plugin it can even deteriorate the perceived QoE on some websites.

VI. Discussion and possible improvements

In this work we found that, under certain circumstances, the overhead introduced by a content-blocker can decrease the overall quality of experience perceived by the user. The main reason of this overhead relies in the number of resources

TABLE VII
DEEP TRACKING BLOCKER QOE RESULTS

Speed Index	FCP	LCP	FMP	CLS	TTI
0.6383	0.7200	0.5963	0.6738	0.8515	0.9638

loaded by the website. Content-blockers check in real time every resource URL in order to block them if they are present inside a black list, and this extra computation adds a small delay. In fact, paradoxically for completely privacy-friendly websites, where no resource has to be blocked, we will always experience a reduction in performance if we use a content-blocker to protect us against the other websites that use tracking resources.

Given that the problem is inherent to the use of pattern matching algorithms, a possible improvement would consist of using machine learning to reduce this overhead. Following this idea, we implemented a new browser plugin called *Deep Tracking Blocker* (DTB) that is based on a Deep Learning model proposed in [40]. This approach was shown to obtain a detection accuracy similar to traditional blacklists (97% precision), while moving the bulk of the pattern matching cost to an offline training phase. DTB is an open-source project [41] and it is already available in the Firefox [42] and Chrome [43] plugin markets. In order to evaluate the performance of the new plugin, we repeated the QoE measurements in Section V but using now DTB. Table VII shows a summary of the results obtained with the same set of 10K websites. Our results show that DTB results in an increase of performance between 2% and 15% for most of the QoE parameters analyzed in Table VI. We attribute the small performance improvement to the limited support current browsers provide to run neural networks within a browser extension. Nevertheless, even if the improvement is modest, the use of Deep Learning brings also other advantages compared to the use of traditional pattern matching, such as the generalization to other tracking URLs not included in the blacklists. Improving the implementation of DTB as a browser extension is part of our future work.

VII. CONCLUSIONS

In this work, we presented the results of a comprehensive measurement study that analyzes the actual performance gains of using content-blockers in terms of *Page Size*, *Page Loading Time*, and other Quality of Experience metrics, such as *Speed Index* or *Time to Interactive*. Contrary to common belief, our results show that the actual improvements in terms of effective page size are small, while speed ups in page loading times and Speed Index are almost negligible.

Regarding the relative performance among the analyzed tools, we observed slightly better results in both, page size and loading time, for *uBlock Origin* compared to *AdBlock Plus* and *Ghostery*. However, those results do not directly translate to the QoE parameters, where *Ghostery* performs slightly better than the other content-blockers.

We also quantified the total fraction of websites where a user could expect a noticeable difference in terms of performance and QoE to be only around 3-5% when using a content-blocker. Only Ghostery presented an increase on the QoE perceived by the user, while AdBlock Plus and uBlock Origin introduced a small penalty in a subset of websites. Based on these results, we can conclude that the use of content-blockers for performance reasons can only be beneficial for slow connections with limited bandwidth, where page size gains of about 10% can make a difference. This could be of special importance on itinerant devices where the volume of data downloaded can incur additional charges. On the contrary, for normal connections where the throughput is not a barrier, content-blockers present at best a very small performance and quality of experience improvement, but in many cases may result in a performance overhead.

The data set collected for this study has been made publicly available at [10].

VIII. ACKNOWLEDGMENTS

This publication is part of the Spanish I+D+i project TRAINER-A (ref. PID2020-118011GB-C21), funded by MCIN/ AEI/10.13039/501100011033.

REFERENCES

[1] AdBlock Plus, "Adblock Plus." https://adblockplus.org/en/.
[2] A. Mathur, J. Vitak, A. Narayanan, and M. Chetty, "Characterizing the Use of Browser-Based Blocking Extensions To Prevent Online Tracking," in *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*, vol. Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018), pp. 103–116, USENIX Association, 2018.
[3] B. Krishnamurthy, D. Malandrino, and C. E. Wills, "Measuring privacy loss and the impact of privacy protection in web browsing," in *Proceedings of the 3rd symposium on Usable privacy and security*, SOUPS '07, (Pittsburgh, Pennsylvania, USA), pp. 52–63, Association for Computing Machinery, July 2007.
[4] J. Mazel, R. Garnier, and K. Fukuda, "A comparison of web privacy protection techniques," *Computer Communications*, vol. 144, pp. 162–174, Aug. 2019.
[5] C. E. Wills and D. C. Uzunoglu, "What Ad Blockers Are (and Are Not) Doing," in *2016 Fourth IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, pp. 72–77, Oct. 2016.
[6] A. Gervais, A. Filios, V. Lenders, and S. Capkun, "Quantifying Web Adblocker Privacy," in *Computer Security – ESORICS 2017* (S. N. Foley, D. Gollmann, and E. Snekkenes, eds.), Lecture Notes in Computer Science, (Cham), pp. 21–42, Springer International Publishing, 2017.
[7] S. Traverso, M. Trevisan, L. Giannantoni, M. Mellia, and H. Metwalley, "Benchmark and comparison of tracker-blockers: Should you trust them?," in *2017 Network Traffic Measurement and Analysis Conference (TMA)*, pp. 1–9, June 2017.
[8] J. Newman and F. E. Bustamante, "The Value of First Impressions," in *Passive and Active Measurement* (D. Choffnes and M. Barcellos, eds.), Lecture Notes in Computer Science, (Cham), pp. 273–285, Springer International Publishing, 2019.
[9] K. Cooper, "Alexa: Most popular website list." https://www.alexa.com/.
[10] "CBA-UPC/ORM," May 2021. https://github.com/CBA-UPC/ORM.
[11] I. Castell-Uroz, J. Solé-Pareta, and P. Barlet-Ros, "Demystifying Content-blockers: A Large-scale Study of Actual Performance Gains," in *2020 16th International Conference on Network and Service Management (CNSM)*, pp. 1–7, Nov. 2020.
[12] R. S.-G. Ismael Castell-Uroz and P. Barlet-Ros, "Deep tracking blocker github," May 2022. https://github.com/CBA-UPC/DTB.
[13] S. Krishnamurthy, "Deciphering the Internet Advertising Puzzle," SSRN Scholarly Paper ID 651842, Social Science Research Network, Rochester, NY, Jan. 2005.
[14] R. Lothia, N. Donthu, and E. K. Hershberger, "The Impact Of Content And Design Elements On Banner Advertising Click-Through Rates," *Journal of Advertising Research*, vol. 43, pp. 410–418, Dec. 2003.
[15] "General data protection regulation," Dec. 2021. https://gdpr.eu/.

[16] T. Bujlow, V. Carela-Español, J. Solé-Pareta, and P. Barlet-Ros, "A Survey on Web Tracking: Mechanisms, Implications, and Defenses," *Proceedings of the IEEE*, vol. 105, pp. 1476–1510, Aug. 2017.

[17] "EasyList." https://easylist.to/.

[18] "EasyPrivacy." https://easylist.to/easylist/easyprivacy.txt.

[19] Acceptable Ads, "Acceptable Ads." https://acceptableads.com/.

[20] O. Zach, M. Slanina, and M. Seufert, "Investigating the impact of advertisement banners and clips on video qoe," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1618–1623, 2018.

[21] E. Pujol, O. Hohlfeld, and A. Feldmann, "Annoyed Users: Ads and Ad-Block Usage in the Wild," in *Proceedings of the 2015 Internet Measurement Conference*, IMC '15, (Tokyo, Japan), pp. 93–106, Association for Computing Machinery, Oct. 2015.

[22] M. Malloy, M. McNamara, A. Cahn, and P. Barford, "Ad Blockers: Global Prevalence and Impact," in *Proceedings of the 2016 Internet Measurement Conference*, IMC '16, (Santa Monica, California, USA), pp. 119–125, Association for Computing Machinery, Nov. 2016.

[23] U. Iqbal, Z. Shafiq, and Z. Qian, "The ad wars: retrospective measurement and analysis of anti-adblock filter lists," in *Proceedings of the 2017 Internet Measurement Conference*, IMC '17, (London, United Kingdom), pp. 171–183, Association for Computing Machinery, Nov. 2017.

[24] A. Lerner, A. K. Simpson, T. Kohno, and F. Roesner, "Internet Jones and the Raiders of the Lost Trackers: An Archaeological Study of Web Tracking from 1996 to 2016," in *25th USENIX Security Symposium (USENIX Security 16)*, vol. 25th USENIX Security Symposium (USENIX Security 16), USENIX Association, 2016.

[25] B. Shiller, J. Waldfogel, and J. Ryan, "The effect of ad blocking on website traffic and quality," *The RAND Journal of Economics*, vol. 49, no. 1, pp. 43–63, 2018.

[26] M. Butkiewicz, H. V. Madhyastha, and V. Sekar, "Understanding website complexity: measurements, metrics, and implications," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, IMC '11, (Berlin, Germany), pp. 313–328, Association for Computing Machinery, Nov. 2011.

[27] Amazon Turk, "Amazon Mechanical Turk." https://www.mturk.com/.

[28] S. Englehardt and A. Narayanan, "Online Tracking: A 1-million-site Measurement and Analysis," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, (Vienna, Austria), pp. 1388–1401, Association for Computing Machinery, Oct. 2016.

[29] Jason Huggins, "SeleniumHQ Browser Automation." https://www.selenium.dev/.

[30] M. Ikram, H. J. Asghar, M. A. Kaafar, A. Mahanti, and B. Krishnamurthy, "Towards Seamless Tracking-Free Web: Improved Detection of Trackers via One-class Learning," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, pp. 79–99, Jan. 2017.

[31] Ghostery, "Ghostery Makes the Web Cleaner, Faster and Safer!." https://www.ghostery.com/.

[32] R. Hill, "uBlock Origin," Feb. 2020. https://github.com/gorhill/uBlock.

[33] ORM, "Online resource mapper," Feb. 2020. https://github.com/CBA-UPC/ORM.

[34] M. Basart, "Online privacy: Analyzing the use of cookies in web pages," Dec. 2021. https://github.com/CBA-UPC/ORM/blob/codesets/publications/Online privacy - Analyzing the use of cookies in web pages.pdf.

[35] W. Aqeel, B. Chandrasekaran, A. Feldmann, and B. M. Maggs, "On landing and internal web pages: The strange case of jekyll and hyde in web performance measurement," in *Proceedings of the ACM Internet Measurement Conference*, IMC '20, (New York, NY, USA), p. 680–695, Association for Computing Machinery, 2020.

[36] W3C, "Navigation timing." https://www.w3.org/TR/navigation-timing-2/.

[37] Google Inc., "Lighthouse | Tools for Web Developers." https://developers.google.com/web/tools/lighthouse?hl=es.

[38] T. Hoßfeld, F. Metzger, and D. Rossi, "Speed index: Relating the industrial standard for user perceived web performance to web qoe," in *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX)*, pp. 1–6, 2018.

[39] "Itu-t g.1030," Dec. 2021. https://www.itu.int/rec/T-REC-G.1030-201402-I/en.

[40] I. Castell-Uroz, T. Poissonnier, P. Manneback, and P. Barlet-Ros, "Url-based web tracking detection using deep learning," in *2020 16th International Conference on Network and Service Management (CNSM)*, pp. 1–5, 2020.

[41] "CBA-UPC/DTB," Apr. 2021. https://github.com/CBA-UPC/DTB.

[42] "Deep Tracking Blocker (Firefox)." https://addons.mozilla.org/en-US/firefox/addon/deep-tracking-blocker/.

[43] "Deep Tracking Blocker (Chrome)." https://chrome.google.com/webstore/detail/deep-tracking-blocker/jjhinlfobgkmbbplbaaegglolichbfol.

Ismael Castell-Uroz (ismael.castell@upc.edu) is a Ph.D. student at the Computer Architecture Department of the Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, where he received the B.Sc. degree in Computer Science in 2008 and the M.Sc. degree in Computer Architecture, Networks and Systems in 2010. He has several years of experience in network and system administration and currently holds a Projects Scholarship at UPC. His expertise and research interest are in computer networks, especially in the field of network monitoring, anomaly detection, internet privacy and web tracking.

Rubén Sanz-García (ruben.sanz.garcia @estudiantat.upc.edu) received the B.Sc. degree in Computer Science in 2021 at the Universitat Politècnica de Catalunya (UPC), Barcelona, Spain. He did and Erasmus during 2020 at the Faculty of Science and Engineering in the University of Groningen(UG), Groningen, Netherlands. He is currently working as a back-end developer at Floorfy and his research and interest are in web tracking, internet privacy and cybersecurity. He was the main developer of the new content-blocker plugin called *Deep Tracking Blocker* created by the UPC-Advanced Broadband Communication Center.

Josep Solé Pareta (pareta@ac.upc.edu) joined the Computer Architecture Department of Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, where he is currently Full Professor. He received the M.Sc. degree in telecommunications engineering and the Ph.D. degree in computer science from UPC in 1984 and 1991, respectively. He did Postdoctoral studies at Georgia Institute of Technology, Atlanta, GA, in 1993 and 1994. He is co-founder of the UPC-Advanced Broadband Communication Center, and his research interests are in nanonetworking communications, traffic monitoring and analysis and high-speed and optical networking, with emphasis on traffic engineering, traffic characterization, MAC protocols, and QoS provisioning.

Pere Barlet-Ros (pere.barlet@upc.edu) is currently an Associate Professor with the Computer Architecture Department of the Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, and Scientific Director at the Barcelona Neural Networking Center (BNN-UPC). He received the M.Sc. and Ph.D. degrees in Computer Science from UPC, in 2003 and 2008, respectively. From 2013 to 2018, he was Co-founder and Chairman of the machine learning startup Talaia Networks. His research has focused on the development of novel machine learning technologies for network management and optimization, traffic classification and network security, which have been integrated in several open-source and commercial products, including Talaia, Auvik TrafficInsights, Intel CoMo and SMARTxAC.

APPENDIX A
MULTIPLE COMPARISON OF MEANS - TUKEY HSD, FWER=0.05

group1	group2	meandiff	p-adj	lower	upper	reject
Document	EventSource	0.1086	0.9	-0.2441	0.4613	False
Document	Fetch	0.0526	0.9	-0.3001	0.4053	False
Document	Font	-0.2769	0.2996	-0.6296	0.0758	False
Document	**Image**	**-7.3018**	**0.001**	**-7.6545**	**-6.9491**	**True**
Document	Manifest	0.1077	0.9	-0.245	0.4604	False
Document	Media	-0.4103	0.0079	-0.7631	-0.0576	True
Document	Other	0.0622	0.9	-0.2905	0.4149	False
Document	Script	-1.2581	0.001	-1.6108	-0.9054	True
Document	Stylesheet	-0.1589	0.9	-0.5116	0.1938	False
Document	TextTrack	0.1086	0.9	-0.2441	0.4613	False
Document	XHR	-0.6577	0.001	-1.0105	-0.305	True
EventSource	Fetch	-0.056	0.9	-0.4087	0.2967	False
EventSource	Font	-0.3855	0.0184	-0.7382	-0.0328	True
EventSource	**Image**	**-7.4104**	**0.001**	**-7.7631**	**-7.0577**	**True**
EventSource	Manifest	-0.0009	0.9	-0.3536	0.3519	False
EventSource	Media	-0.5189	0.001	-0.8716	-0.1662	True
EventSource	Other	-0.0464	0.9	-0.3991	0.3063	False
EventSource	Script	-1.3666	0.001	-1.7194	-1.0139	True
EventSource	Stylesheet	-0.2675	0.355	-0.6202	0.0853	False
EventSource	TextTrack	0.0	0.9	-0.3527	0.3527	False
EventSource	XHR	-0.7663	0.001	-1.119	-0.4136	True
Fetch	Font	-0.3295	0.0936	-0.6822	0.0232	False
Fetch	**Image**	**-7.3544**	**0.001**	**-7.7071**	**-7.0017**	**True**
Fetch	Manifest	0.0551	0.9	-0.2976	0.4079	False
Fetch	Media	-0.4629	0.0011	-0.8156	-0.1102	True
Fetch	Other	0.0096	0.9	-0.3431	0.3623	False
Fetch	Script	-1.3106	0.001	-1.6634	-0.9579	True
Fetch	Stylesheet	-0.2115	0.6937	-0.5642	0.1413	False
Fetch	TextTrack	0.056	0.9	-0.2967	0.4087	False
Fetch	XHR	-0.7103	0.001	-1.063	-0.3576	True
Font	**Image**	**-7.0249**	**0.001**	**-7.3776**	**-6.6722**	**True**
Font	Manifest	0.3846	0.0189	0.0319	0.7374	True
Font	Media	-0.1334	0.9	-0.4861	0.2193	False
Font	Other	0.3391	0.0734	-0.0136	0.6918	False
Font	Script	-0.9811	0.001	-1.3339	-0.6284	True
Font	Stylesheet	0.118	0.9	-0.2347	0.4708	False
Font	TextTrack	0.3855	0.0184	0.0328	0.7382	True
Font	XHR	-0.3808	0.0214	-0.7335	-0.0281	True
Image	**Manifest**	**7.4095**	**0.001**	**7.0568**	**7.7622**	**True**
Image	**Media**	**6.8915**	**0.001**	**6.5388**	**7.2442**	**True**
Image	**Other**	**7.364**	**0.001**	**7.0113**	**7.7167**	**True**
Image	**Script**	**6.0437**	**0.001**	**5.691**	**6.3965**	**True**
Image	**Stylesheet**	**7.1429**	**0.001**	**6.7902**	**7.4956**	**True**
Image	**TextTrack**	**7.4104**	**0.001**	**7.0577**	**7.7631**	**True**
Image	**XHR**	**6.6441**	**0.001**	**6.2913**	**6.9968**	**True**
Manifest	Media	-0.5181	0.001	-0.8708	-0.1653	True
Manifest	Other	-0.0455	0.9	-0.3983	0.3072	False
Manifest	Script	-1.3658	0.001	-1.7185	-1.0131	True
Manifest	Stylesheet	-0.2666	0.3604	-0.6193	0.0861	False
Manifest	TextTrack	0.0009	0.9	-0.3519	0.3536	False
Manifest	XHR	-0.7655	0.001	-1.1182	-0.4127	True
Media	Other	0.4725	0.001	0.1198	0.8252	True
Media	Script	-0.8477	0.001	-1.2004	-0.495	True
Media	Stylesheet	0.2515	0.4585	-0.1013	0.6042	False
Media	TextTrack	0.5189	0.001	0.1662	0.8716	True
Media	XHR	-0.2474	0.4838	-0.6001	0.1053	False
Other	Script	-1.3202	0.001	-1.673	-0.9675	True
Other	Stylesheet	-0.2211	0.6379	-0.5738	0.1317	False
Other	TextTrack	0.0464	0.9	-0.3063	0.3991	False
Other	XHR	-0.7199	0.001	-1.0726	-0.3672	True
Script	Stylesheet	1.0992	0.001	0.7465	1.4519	True
Script	TextTrack	1.3666	0.001	1.0139	1.7194	True
Script	XHR	0.6003	0.001	0.2476	0.953	True
Stylesheet	TextTrack	0.2675	0.355	-0.0853	0.6202	False
Stylesheet	XHR	-0.4989	0.001	-0.8516	-0.1461	True
TextTrack	XHR	-0.7663	0.001	-1.119	-0.4136	True