Demystifying Content-blockers: A Large-scale Study of Actual Performance Gains

Ismael Castell-Uroz
Universitat Politècnica de Catalunya
Barcelona, Spain
icastell@ac.upc.edu

Josep Solé-Pareta
Universitat Politècnica de Catalunya
Barcelona, Spain
pareta@ac.upc.edu

Pere Barlet-Ros
Universitat Politècnica de Catalunya
Barcelona, Spain
pbarlet@ac.upc.edu

***Abstract*—With the evolution of the online advertisement and tracking ecosystem, content-filtering has 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 with the 100K most popular websites on the actual performance improvements of using content-blockers. We focus our study on two relevant metrics for measuring the browsing performance; page size and loading time. Our results show that using such tools results in small improvements in terms of page size but, contrary to popular belief, it has a negligible impact in terms of loading time. We also find that, in the case of small and lightweight websites, the use of content-blockers can even result in increased loading times.**

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

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. As recently 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 is believed to significantly reduce the bandwidth used, improving the website loading time and, thus, the overall browsing experience.

In this paper, we present a comprehensive study of the advertisement and tracking ecosystem with special emphasis on the performance gains 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 improvements of different content-blockers in terms of latency and bandwidth 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 compare three different blocking approaches; *advertisement blocking*, *tracker blocking* and *generic content blocking*. For this purpose, we developed an open-source highly parallel network measurement system [10] that loads every website using one of the most relevant content-blockers of each category and compares their 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. In some cases, there could even be an overhead to be paid. This is the case for two of the studied plugins, especially in small and fast loading websites. 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.

The rest of the paper is organized as follows. Section II provides the necessary background as well as the related work. Section III describes the methodology used to collect the data and perform the experiments. Section IV presents and analyzes the results in terms of page size and loading time. Lastly, Section V concludes the paper and discusses the future work.

II. Background and Related Work

A. Background

Advertisements have been used as a way to get benefits from online resources since the foundation of the Internet. At first, owners tried to add some advertisements to monetize each site. Soon, the difficulty to attract user attention became relevant [11]. To be able to stand out in such an environment, companies started to use pop-ups and other intrusive methods to reach their users. Lohtia et al. [12] studied the impact of those methods on advertisement performance and user perception. Such an 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 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, super-cookies), 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. In [13]

Bujlow et al. show a summary of the different mechanisms that can be used to track users and correlate their information.

Online tracking has raised serious privacy and surveillance concerns. Consequently, a new method of content-filtering appeared; **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 [14] and EasyPrivacy [15], 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* [16], 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).

B. *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. 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.

Pujol et al. [17] studied the usage of adblockers in a European ISP and found that 22% of the most active users used an adblocker. Malloy et al. [18] measured the percentage of adblockers users in the U.S. and inferred that about 18% was using it. The constantly increasing adoption rate and lose of revenues made online publishers to start using different techniques to try to bypass adblockers. In [19] Iqbal et al. studied the anti-adblocking ecosystem and found it to be continuously increasing.

In [20] Lerner et al. studied the prevalence of tracking methods over the Internet. 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.

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 [8] the protection and performance improvement using an adblocker with a population of only 100 websites (most of them belonging to the same country). In [7], Newman et al. used Amazon Mechanical Turk [21] 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, we show (Section IV-B) that the actual impact of using a content-blocker on page loading times is negligible in most cases with a sample of 100K websites.

III. Methodology and evaluation

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

A **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.

B **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.

C **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).

D **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.

E **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.

A. *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], [22]). Note that we only examine the homepages of each one of the websites included in the list but none of the links and external resources available within them.

B. *Browsing Experience*

We developed our own highly parallelized system making use of *Selenium* [23] 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. [22], [24]).

The experiments are done browsing the 100K websites using one plugin from each group presented in Section II-A; **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*. All of them have versions available for the most common internet browsers like *Google Chrome* or *Mozilla Firefox*.

AdBlock Plus is by far the most popular adblocker in the market. It uses a combination of the *EasyList* [14] and *EasyPrivacy* [15] pattern lists to block both, advertisements and tracking systems (EasyPrivacy is disabled by default). AdBlock Plus is a supporter of the *acceptable ads program* [16], allowing advertisements if they follow specific guidelines.

Ghostery was initially developed as a tracker blocker, but evolved into a combination of adblocker and tracker blocker. Unlike AdBlock Plus, Ghostery uses its own maintained database 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* is one of the most used generic content-blockers in the web. Its functionality is very similar to AdBlock Plus, using the same EasyList and EasyPrivacy lists as the main method to discard content from the web. Unlike AdBlock Plus, uBlock Origin does not permit any kind of advertisement and blocks also other nuisances and privacy threats like Cross Site Scripting (XSS) loaded resources. To this end, it uses a combination of several different additional pattern lists. A summary of the differences between them is shown in Table I.

We decided to use *Chromium* (version 76.0.3809.100), the open-source version of *Google Chrome*, as the base live to compare all the observed information. *Firefox* was also tested executing some experiments using a small data set. we 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 permits us to easily get information about all the network communications being performed.

TABLE I
CONTENT-FILTERING PLUGINS COMPARISON

	AdBlock Plus	**Ghostery**	**uBlock Origin**
Ads blocking	○	○	○
Allow some ads	○	○	–
Track blocking	△	○	○
EasyList	○	–	○
EasyPrivacy	Available	–	○
Others	–	Private Database	Additional lists

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 [17], 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%.

C. *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.

D. *Experiments repetition*

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.

TABLE II
PERFORMANCE TIMING API

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

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 using 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 [25].

E. 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 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.

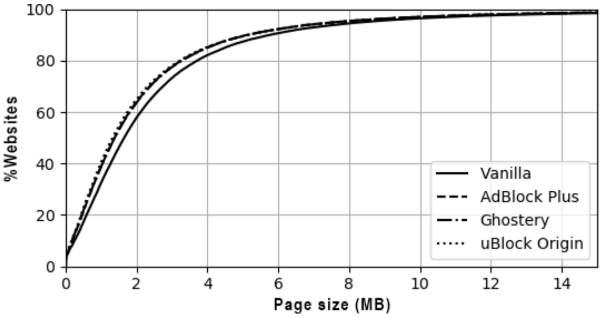

Fig. 1. Page size distribution (CDF)

As for the loading time metrics, we use the *Performance Timing* API [26], 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 content-filtering plugin, we will be focusing our experiments computing the request time, between the *requestStart* event and the *domComplete* event. The request time gives us the total loading time from the point when the browser makes the actual request until it loads all the needed resources. In this way, we avoid accounting undesired variability like the time spent unloading the previous website, process mainly dependent on the computer CPU speed and memory.

IV. RESULTS

A. 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 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%).

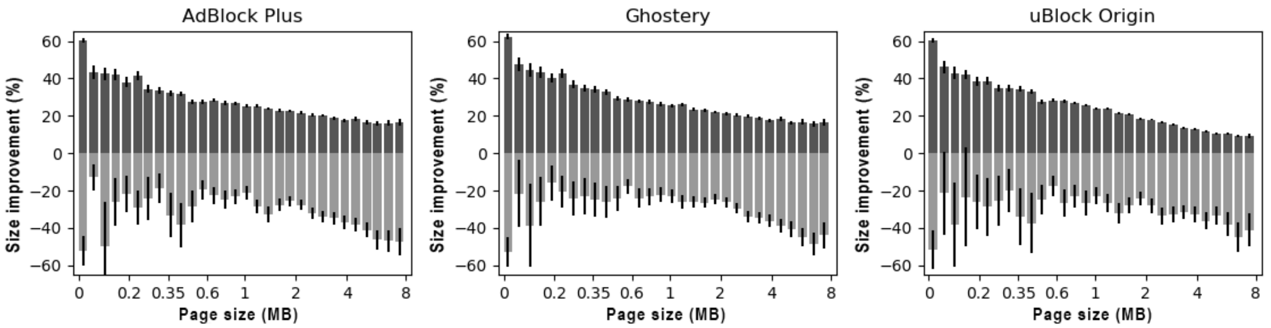

Fig. 2. Size improvement/Page size

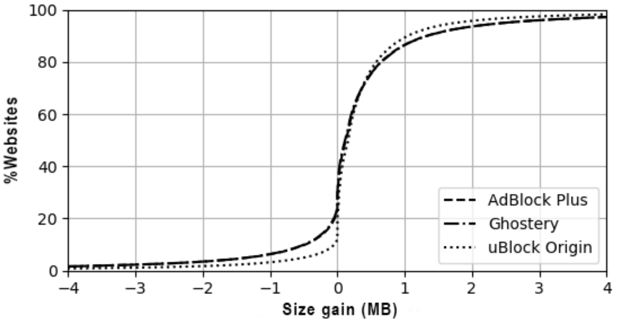

Fig. 3. Page size gain

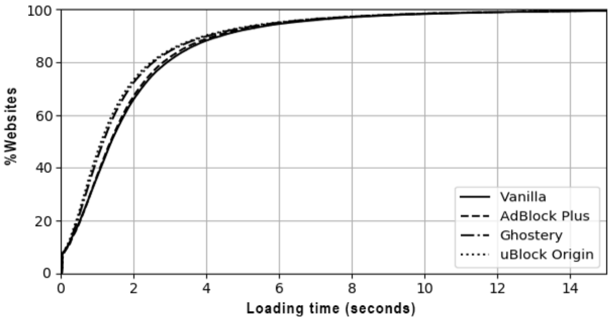

Fig. 4. Request time

Fig. 3 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. 2. 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. 3, 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. 2 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. Consequently, 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.

B. Loading time comparison

Another metric that has an important impact 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. 4. Approximately 90% of the websites have a loading time lower than 5 seconds, being also this range the one that benefits more from using 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. 6 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,

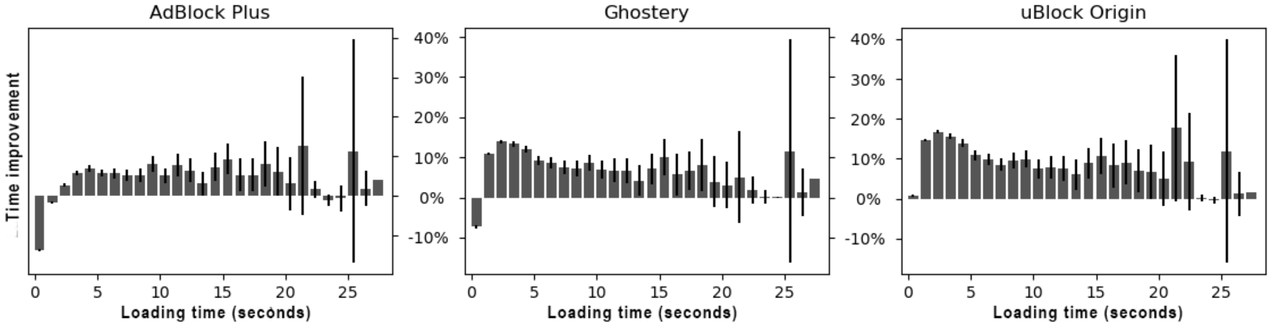

Fig. 5. Improvement distribution

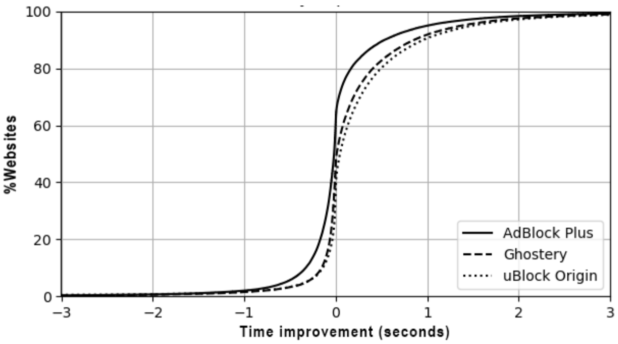

Fig. 6. Loading time improvement

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. 5 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. 4), 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.

Content-filtering plugins 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. The main difference between the three explored plugins lies on their advertisement management engine. While uBlock Origin automatically blocks any type of content that matches its pattern lists, Ghostery and AdBlock Plus allow some advertisements to be shown to the user.

V. Conclusions and Future Work

In this work, we presented the results of a comprehensive measurement study that analyzes the actual performance gains of using content-filtering plugins in terms of page size and loading time. Contrary to common belief, our results show that the improvements in terms of effective page size are small, while speed ups in page loading times are almost negligible.

Regarding the relative performance among the analyzed tools, we observed slight gain margins in both, page size and loading time, for *uBlock Origin* against *AdBlock Plus* and *Ghostery*. Moreover, we discovered interesting differences in terms of loading time, where *Ghostery* and specially *AdBlock Plus* introduced an overhead for fast-loading websites.

Based on our results, we can conclude that the use of content-filtering plugins for performance reasons can still be useful 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 an average loading time improvement between 50s and 250 milliseconds does not represent a fundamental performance improvement.

As a future work, we plan to characterize the loading and access times depending on the source and destination of the connections, looking for differences between countries and website content type. We also plan to investigate the content-blocker performance in the mobile environment, where loading time, and especially the page size can have a big impact.

The data set collected for this study has been made publicly available at [25].

VI. Acknowledgments

This work was supported by the Spanish MINECO under contract TEC2017-90034-C2-1-R (ALLIANCE).

References

[1] AdBlock Plus, "Adblock Plus," vol. 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," 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] 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.

[8] 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.

[9] K. Cooper, "Alexa: Most popular website list," *https://www.alexa.com/*.

[10] I. Castell-Uroz, J. Solé-Pareta, and P. Barlet-Ros, "Network measurements for web tracking analysis and detection: A tutorial," *IEEE Instrumentation & Measurement Magazine*, vol. Special Issue "Measurements for Advanced Networking & Networks for Advanced Measurements", In press.

[11] S. Krishnamurthy, "Deciphering the Internet Advertising Puzzle," SSRN Scholarly Paper ID 651842, Social Science Research Network, Rochester, NY, Jan. 2005.

[12] 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.

[13] 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.

[14] "EasyList," *https://easylist.to/*.

[15] "EasyPrivacy," vol. https://easylist.to/easylist/easyprivacy.txt.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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," vol. 25th USENIX Security Symposium (USENIX Security 16), USENIX Association, 2016.

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

[22] 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.

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

[24] 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.

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

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