Shedding (Some) Light on Mobile Browsers Energy Consumption

Matteo Varvello
Nokia, Bell Labs
matteo.varvello@nokia.com

Benjamin Livshits
Brave Software
ben@brave.com

***Abstract*—Following its desktop counterpart, the mobile browsers ecosystem has been growing from few browsers (Chrome, Firefox, and Safari) to a plethora of browsers, each with unique characteristics (battery friendly, privacy preserving, etc.).The main contribution of this work is a thorough benchmark of 15 Android browsers using multiple settings (adblocking, dark mode, power savings), devices, workloads (up to 600 websites and 100 concurrent tabs), and network configurations. We perform a *battery-centric* analysis and show that: 1) 5 out of 8 adblocking browsers produce significant battery savings (up to 30% in the case of Brave) compared to more popular browsers like Chrome or Firefox, 2) Opera is the most efficient browser when dealing with a large number of concurrent tabs, 3) dark mode on AMOLED devices offers an extra 12% of battery savings, granted that the whole webpage is darkened, and 4) up to 30% of energy is wasted loading ads-bloated webpages on a slow mobile network. We exploit the latter observation to build *AttentionDim*, a screen dimming mechanism driven by browser events which we integrated with Brave. Via experiments with 10 volunteers over a month, we conclude that *AttentionDim* offers battery savings of up to 30% with minimal impact on the end-user experience.**

***Index Terms*—Browser, Web, energy, performance.**

I. Introduction

When it comes to mobile apps, users are mostly tied to the official app from the services they access, e.g., YouTube or Facebook. This is not the case for mobile browsers where plenty of options are currently available for both Android and iOS [6]. Such a competitive environment constantly stimulates the development of new browsers as well as new browser functionalities.

In the last years, there has been a growing interest in reducing browsers (and apps in general) power consumption, motivated by the ever-increasing phone usage and app complexity. *Adblocking*—either in the form of an addon [7], [8] or directly integrated in the browser [9]–[11]—is probably the most popular feature which has recently been connected with battery savings [2], [5]. *Dark mode* [12] is another feature which, originally introduced for eye strains, is now credited with high battery savings in presence of AMOLED screens which are capable of turning pixels off when dark. The Yandex browser even offers a power saving mode [13], of which not much is known.

The first goal of this work is to *shed some light* on the Android browser ecosystem. Our approach is clearly battery-centric, but it also covers other metrics which directly impact battery usage, like CPU and bandwidth utilization. Table I shows a direct comparison of the measurement study we perform in our work with similar studies that can be found, to the best of our knowledge, among related works. The main takeaway from the table is that our work completes previous works in three aspects. First, we focus on a larger set of *browsers* (15 versus 1, at most), and browser settings, e.g., also exploring dark and power saving modes. Second, we cover a variety of *workloads*, investigating 7x the number of websites tested in the largest previous study [4] and using several workloads with up to 100 concurrent tabs. Third, we are the first to evaluate browsers in the *wild*.

We have developed a browser testing suite [14] to study the battery consumption (and more) of 15 Android browsers under different configurations (classic, adblocking, dark-mode), workloads (up to 100 concurrent tabs with websites extracted from several lists of popular domains/webpages), devices, and networks (WiFi and 10 locations in a North American mobile network). Our analysis has generated the following findings:

Adblocking saves battery, but not all adblockers are equal – Five out eight adblocking browsers (Brave, DuckDuckGo, Firefox Focus, Opera and Firefox equipped with the Ublock plugin) are more energy efficient than popular browsers like Chrome, Baidu, and Firefox. This result holds both for *ads-heavy* and *ads-free* websites. Three browsers (Kiwi, OperaMini, and Vivaldi) were exceptions, due to either limited adblocking and/or high CPU consumption. Brave's aggressive adblocking and low CPU consumption makes it the most energy efficient browser in presence of ads. This aggressive adblocking introduces a small penalty in absence of ads, where Opera and DuckDuckGo achieve slightly lower battery usage.

Stress testing browsers matters! – Regardless of the websites under test, Opera – despite lower adblocking capabilities – resulted the most energy and CPU efficient browser as soon as the number of concurrently opened tabs grows past 20. Conversely, DuckDuckGo's tab management fails to scale past 50-60 concurrent tabs.

Browsing in the (very) dark saves battery – Dark theme offers an average 12% extra battery savings on AMOLED devices. Such savings are realized when the whole content of the page is darkened. Simple GUI darkening, e.g., current Chrome's dark mode, and Yandex power saving mode resulted in no energy savings in our tests.

978-3-903176-40-9 ©2021 IFIP

	[1]	[2]	[3]	[4]	[5]	This Work
Year	2012	2014	2015	2017	2020	2020
Power Monitor	Agilent 34410A	Monsoon	Monsoon	Monsoon	Software	Monsoon/Battor
Devices	Android Dev. Phone 2	Galaxy Nexus dev. board	Samsung S5	Samsung S4/S5, Nexus	Odroid-XU3	Samsung J7-Duo, Galaxy J3
Android Vrs	4.2	4.2.1	4.4	4.3/5.1.1	4.4	8.0/9.0
Browsers	Built-in Browser	Built-in Browser	Chrome vrs 38	Chrome vrs 31	Brave, Chrome vrs 64	15 browsers
Workloads	25 (indiv.)	5 (indiv.)	10 (indiv.)	80 (indiv.)	23 (indiv.)	News: 10 tabs Ads-Free: 10 tabs Top Sites: 600 (up to 100 tabs)
Features	Default	Default, Adblocking	Default	Default	Default, Adblocking	Default,Adblocking, DarkMode,PowerSave
Connectivity	WiFi/3G (Lab)	WiFi/3G (Lab)	WiFi/3G (Lab)	WiFi/3G (Lab)	WiFi (Lab)	WiFi/4G (Lab, Wild)

TABLE I
COMPARISON WITH RELATED WORK.

Battery is "wasted" while waiting for a webpage to load – When network connectivity deteriorates, e.g., low signal in a mobile network, up to 30% of extra energy is *wasted* showing an empty page. Under these challenging conditions, the energy benefits of adblocking browsers are even more prominent.

The latter observation motivates `AttentionDim`, a generic battery saving mechanism for mobile browsers. Our intuition is to lower the screen brightness when the user attention is low, e.g., during a page load. We build `AttentionDim` as a Chrome patch so that it can be integrated with *any* Chromium-based browser, e.g., 12 out the 15 browsers we tested. Next, we perform a user study involving 10 volunteers who used an AttentionDim-med version of the Brave browser over a month. The analysis of the data collected shows that AttentionDim reduces battery consumption by up to 30%, independently of the device's screen technology, offering high user Quality of Experience (mean opinion score of 4.8/5 from a survey).

II. METHODOLOGY

We start by discussing our browser selection process. First, we target *popular* browsers. Second, we target *adblocking* browsers – either native or enhanced with adblocking addons – because of the potential energy benefits associated with serving smaller and simpler webpages [2], [5]. Finally, we target browser settings associated with *energy saving* capabilities, namely dark mode and Yandex power saving mode. Based on this strategy, we have selected 15 browsers whose name, version, underlying engine (*i.e.,* Blink/Chromium or Gecko/Firefox) and popularity are reported in Table II (see Appendix). The popularity column is derived from data reported by netmarketshare [15] as of October 2020, before being retired due to [16] which breaks their browser detection methodology based on the user-agent. For browsers not included in [15], the table reports the most recent number of downloads from Google Play Store, as of March 2021. As testing devices, we use a Samsung J7 Duo (J7DUO) and a Samsung Galaxy J3 (SMJ337A). The main difference between the two devices is their screen technology: "Active Matrix Organic Light Emitting Diodes" (AMOLED) for the J7DUO and "Liquid Crystal Display" (LCD) for SMJ337. Hardware-wise, the J7DUO is more powerful, with twice as many cores (octa vs quad-core) and RAM (4 vs 2GB).

Browser	Version	Chrome or Firefox Vrs	Popularity
Chrome	86.0.4240.99	86.0.4240.99	89%
QQ	10.3.1.6830	66.0.3359.126	2.8%
Samsung Browser	12.1.4.3	79.0.3945.136	2.4%
Baidu	4.14.5.31	86.0.4240.99	0.79%
Firefox	81.1.5	Gecko/81.0	0.44%
Yandex	20.8.5.97	84.0.4147.135	>100M
Edge	45.09.2.5079	77.0.3865.116	>10M
Opera Mini	51.0.2254 .150807	86.0.4240.99	1.15%
Opera	60.2.3004 .55409	85.0.4183.127	0.64%
Brave	1.15.73	86.0.4240.75	>10M
DuckDuckGo	5.67.0	86.0.4240.99	>10M
Firefox Focus	8.8.3	Gecko/81.0	>5M
Firefox UBlock	81.1.5	Gecko/81.0	>5M
Kiwi	Git201009	86.0.4240.75	>1M
Vivaldi	3.4.2066.74	86.0.4240.99	>100K

TABLE II
ANDROID BROWSERS SELECTED FOR PERFORMANCE EVALUATION. "POPULARITY" REFERS TO EITHER MARKET SHARE (%) OR NUMBER OF DOWNLOADS AS PER THE GOOGLE PLAY STORE. BROWSERS IN THE BOTTOM OF THE TABLE HAVE ADBLOCKING FEATURES.

The battery of these devices are connected to a Monsoon power meter [17] which produces fine-grained battery readings. A second battery for the SMJ337 is also connected to battor [18], a *portable* power meter which enables mobile experiments out of lab settings. More details on this setup can be found in Section III-D.

These devices are also connected to a Rasberry Pi (via WiFi to avoid USB noise on the power readings) which instruments the browsers while monitoring their resource utilization, e.g., CPU usage. Both tasks are realized via the Android Debugging Bridge (ADB), a powerful tool used for debugging and testing Android apps. We wrote a generic browser testing tool which works across 15 browsers, several settings, and our testing devices (and more), which we open source at [14]. Before

Ads-Free	News
https://qwant.com	https://theblaze.com
https://mozilla.org	https://thedailybeast.com
https://gnu.org	https://independent.co.uk
https://i-register.in	https://nypost.com
https://www.wikipedia.org	https://salon.com
https://ipko.pl	https://sfgate.com
https://bankmellat.ir	https://latimes.com
https://jw.org	https://cnn.com
https://sarzamindownload.com	https://mirror.co.uk
https://nginx.org	https://cnet.com

TABLE III
SEQUENCE OF WEBSITES TESTED WITHIN THE ADS-FREE AND NEWS WORKLOADS.

each experiment, the device under test is configured to minimize noise on the measurements. For instance, background processes and app notifications are disabled. Next, the browser is setup with a clean profile, *i.e.,* its cache is emptied, and local data like configuration, cookies, and history are erased.

Our generic browser testing tool takes as input a *workload*, a JSON file describing which websites to test and how, e.g., whether opening each website in a new tab or not. We selected the following workloads:

News* and *Ads-free – Most users keep between 1 and 10 open tabs in their browser [19]. Accordingly, in these two workloads we sequentially open 10 testing pages in a new tab. Each page is requested for 15 seconds, which allows full page load in the testing network conditions (WiFi with ∼50Mbps in download/upload). After the page is loaded, we simulate user interaction for an extra 15 seconds by scrolling the page down four times and then up two times.

With respect to the pages to be tested, we pick the landing pages of 10 news websites popular worldwide (*news* workload) and 10 (hard to find) ads-free websites (*ads-free* workload). The rationale of this selection is that these two workloads are realistic representation of, respectively, the best and worst case scenarios for adblocking browsers – given news websites tend to host lots of advertisement [20]. Table III shows the sequence of websites tested under the news and ads-free workloads. To find ads-free websites, we crawled the tranco top 1,000 websites [21] and matched the content they served against the EasyList and EasyPrivacy filters [22]. We further manually tested each ads-free website to verify functionalities, *i.e.,* response times within 15 seconds and lack of ads.

Top Websites Lists – In these workloads, we *stress test* the browsers by loading 100 websites sequentially, each in a new tab. We choose 100 tabs given Chrome stops counting tabs at this point, suggesting that it should be an upper bound for most users. For these workloads, we used several free lists of popular websites: Tranco [21], Majestic [23], and the recently released Hyspar [24], [25]. The Tranco list is a research effort aiming to replace Alexa [26] due to its paywall. The Majestic list is also free and representative of websites attracting actual Web browsers traffic [27]. Finally the Hyspar list is a recently released list which focuses on internal rather than landing pages, motivated by significant differences, e.g., size and number of ads, when compared with their originating landing page. For each list, we choose the top 100 websites, as well as 100 pages ranked between 1,000 and 1,100. In total, we have constructed a corpus of 600 websites to be tested – of which 539 are unique since 61 websites are in common between tranco and majestic top 100 websites.

III. BROWSERS EVALUATION

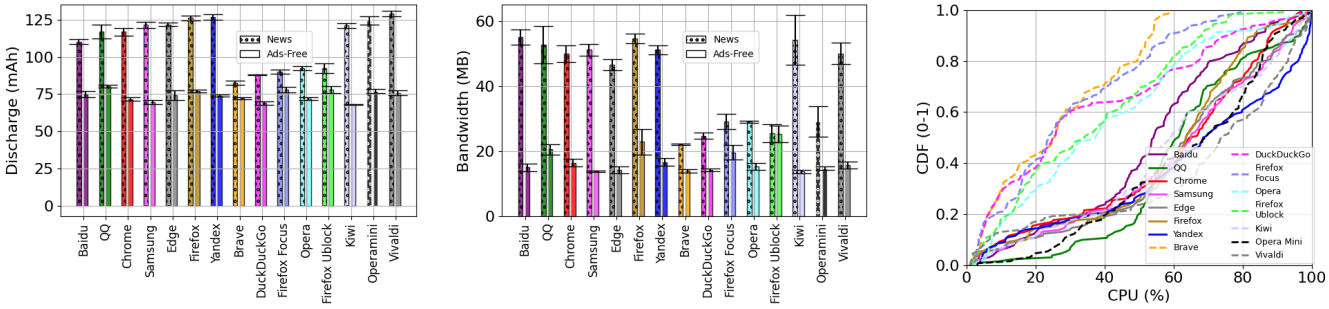
A. Popular or Adblocking?

Figure 1 summarizes the performance evaluation (battery discharge, bandwidth consumption, and CPU utilization) of all browsers under test with default configuration, while considering news and ads-free workloads, and SMJ337A connected over WiFi. Barplots report, for each metric, the mean with errorbars for standard deviation (values computed over 5 repetitions). Given CPU consumption varies during an experiment, Figure 1(c) shows one representative Cumulative Distribution Function (CDF) per browser. We use circle markers (barplots) and dashed lines (CDFs) to highlight adblocking browsers. Browsers are ordered by energy consumption under the news workload, and organized with popular browsers on the left and adblocking browsers on the right.

News workload – Figure 1(a) shows that popular browsers (Chrome, Samsung, QQ, Baidu, Firefox, Yandex, and Edge) have similar energy consumption with values mostly within 5% of each others, e.g., from 125 to about 118mAh. Baidu reduces energy usage by another 5% (absolute value of 110mAh). This reduction is due to Baidu not allowing multiple opened tabs; instead, each new tab replaces the previous one. Most adblocking browsers – with the exceptions of Kiwi, Opera Mini, and Vivaldi – are 20-30% more power efficient than regular browsers consuming between a minimum of 82mAh (Brave) and a maximum of 92mAh (Firefox equipped with the Ublock plugin). Kiwi and Opera Mini are in line with non adblocking browsers (∼120mAh) while Vivaldi consumes the maximum energy measured: 129mAh.

To further explain the latter observation, we investigate the amount of bandwidth saved by the different adblocking browsers. Figure 1(b) shows that Kiwi and Vivaldi fail at capturing most ads, e.g., they consume 2x the bandwidth of actual adblocking browsers. OperaMini's adblocking is instead comparable to others, implying that the extra bandwidth is not the cause of its high energy footprint. Brave is the most aggressive adblocker, saving an extra 5MB when compared to DuckDuckGo, the browser ranked second with respect to bandwidth saving via adblocking.

To explain the high energy consumption of OperaMini we resort to Figure 1(c), which analyzes each browser CPU consumption. Opera Mini (along with Vivaldi) shows very high CPU consumption, with median of 65%, 2.5x times the CPU consumption of the lightest browsers: Brave, Firefox focus, and DuckDuckGo. These 3 browsers have similar CPU utilization for about 65% of the values. Next, Brave is less likely to cause high CPU utilization which is bounded to 60%



(a) Battery discharge (mAh).

(b) Bandwidth consumption (MBytes)

(c) CDF of CPU utilization during a single run ; news workload.

Fig. 1. Perf. evaluation of 15 Android browsers ; News (left barplots) and ads-free (right barplot) workloads; SMJ337A ; WiFi.

while Firefox Focus, for instance, causes higher CPU utilization during 10% of the workload. DuckDuckgo further departs from Brave and Firefox Focus, showing a CPU footprint more similar to Opera and Firefox Ublock for the second 50% of the distribution.

Ads-free workload – We next focus on a workload consisting only of websites without ads. In this case, Figure 1(a) shows a much more similar behavior between browsers, with a minimum and maximum energy usage of 68mAh (Kiwi) and 80mAh (QQ), respectively. The same argument holds for the bandwidth consumption, overall stable around 15MB. QQ and browsers using the Firefox engine are exceptions, consuming up to an extra 10MB (or 66% more). This additional data is *in browser*, meaning that it is not originated by the websites being tested but independently by the browser.

Figure 1(a) also shows that, in absence of ads, DuckDuckGo and Kiwi slightly outperform Brave, *i.e.,* 68mAh vs 72mAh. Given Figure 1(b) (news workload) indicates that Brave's adblocker is the most effective one, this result indicates the Brave's aggressive adblocking causes a (small) performance penalty in absence of ads. While this result is in line with what observed in [5], our tests show that it does not generalize to other adblocking browsers.

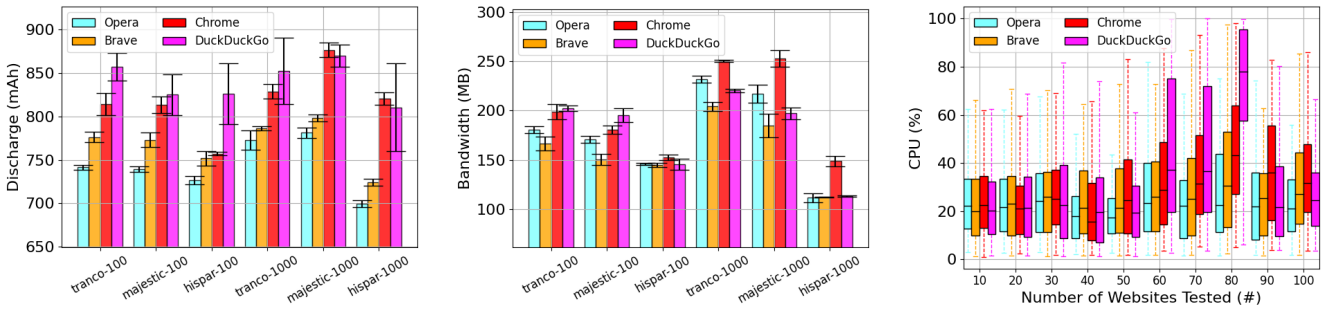
Next, we dive into the energy consumption per browser tab. Figure 3(a) shows the average energy consumption associated with the loading of each website within the news workload, which has shown a more diverse behavior among browsers. We omit the standard deviation for ease of visibility, but it is in line with the previous (cumulative) result. As expected from Figure 1(a), adblocking browsers tend to consume less energy. However, not one browser consumes the least for each page. For example, while Brave consumes the least on 7 out of the 10 pages, it is outperformed by DuckDuckGo on *independent.co.uk*, Firefox equipped with Ublock on *sfgate.com*, and Baidu on *latimes.com*.

Figure 3(a) also shows the cost of *onboarding*. Each browser onboarding procedure is different: some browsers explain their functioning (Brave), some ask for a configuration (Chrome), some do nothing (Kiwi). Post onboarding behavior is also different: some browsers launch an empty page (Kiwi), some load a search engine (Chrome), and some show some news (QQ). Due to the significant different behaviors during onboarding, in our experiments we always synchronize the browsers after one minute, which is long enough to accommodate even the longest onboarding procedure (QQ browser, roughly 30 seconds). Further, we discard the cost of onboarding for each workload since it is not a common user operation. Figure 3(a) shows that onboarding is overall more energy consuming than regular browsing. This cost comes mostly from traffic volumes, either because the onboarding terminates on a news page (like in the case of the QQ browser, causing the download of about 60MB) or to download useful browser components, like the adblocking list used by Brave which account for an extra 5MB.

B. Top Websites Lists

We now *stress test* browsers when loading 100 websites (in 100 tabs) selected from three popular lists: tranco, majestic, and hispar. Figure 2 summarizes the results (energy, bandwidth, and CPU consumption) for Chrome, Opera, Brave and DuckDuckGo, *i.e.,* the top performing browsers from the previous workloads. We also experimented with Baidu and QQ browsers but we had to dismiss them due to: 1) crashing, 2) semi-random notification windows interrupting the automation. From each list, we select the top 100 websites as well as the 100 sites at position 1,000. The same loading procedure as before is used: we let a page load for 15 seconds, then interact with it for the next 15 seconds. In total we tested 600 websites (of which 539 are unique) during about 24 hrs. We repeat this experiment over three consecutive days.

Figure 2(a) shows the energy consumption per browser and workload. The figure confirms – at large scale – the benefits of adblocking with respect to energy savings, given that Opera and Brave outperform Chrome regardless of the workload. The same does not always hold for DuckDuckGo, especially when focusing on the top 100 websites of each list. When comparing Chrome, Brave, and Opera, this test allows us to make two new observations. First, Opera is now slightly more energy efficient than Brave, with 2-5% less energy usage, depending on the websites list. Second, Chrome is a much closer competitor, especially when considering top 100 websites. In fact, while for the top 100 websites adblocking browsers save 4-10% of



(a) Battery discharge (mAh).

(b) Bandwidth consumption (MBytes)

(c) Boxplot of CPU utilization as a function of the number of concurrently opened tabs.

Fig. 2. Stress test (100 concurrent tabs) of the top performing Android browsers ; Tranco, Majestic, Hispar lists ; SMJ337A ; WiFi.

energy compared to Chrome, this percentage grows past 10% (10.8% and 12.5% for Opera and the majestic and hispar lists) when considering less popular websites.

Figure 2(b) offers an insight on *why* there is a reduced gap between Chrome and adblocking browsers. Even the best adblocking browser (Brave) manages to only save 13.8% (tranco), 18.6% (majestic), and 6.6% (hispar) of the bandwidth consumed by Chrome, which is far from the 40% savings measured for the news workload. The bandwidth savings increase as less popular websites are tested – 15.2% (tranco), 26.8% (majestic), and 25.6% (hispar) – which translate in overall more energy savings.

When considering different lists, tranco and majestic offer comparable results (energy and bandwidth) with their top 100 websites. This is not surprising given these lists share 59 websites. The hispar list contains internal pages which are characterized by fewer ads than the landing pages contained in the tranco and majestic list. This lack of ads translates in minimum bandwidth savings from Brave (6% or ∼40MB). The lists are then fully disjoint when focusing on lower rank subsets, which generates new insights with respect to both energy and bandwidth.

A second aspect to investigate is the *high* number of concurrent tabs (up to 100). Figure 2(c) analyzes the evolution of the CPU consumption, per browser, as the number of tabs increases. We focus on the top 100 websites from the tranco list, for which we measure small variance for *all* browsers and metrics. Each boxplot accounts for CPU values sampled every two seconds for a given set of tabs, in 10 tabs intervals. We start by focusing on the boxplots computed when considering 10 opened tabs, which can be compared with Figure 1(c) for the news workload. With respect to Opera, Brave, and DuckDuckGo, the trend is similar, e.g., median CPU consumption of 20-25%. Chrome's median CPU consumption drops significantly (from 60% down to 25%) which is another contributor to the reduced energy usage.

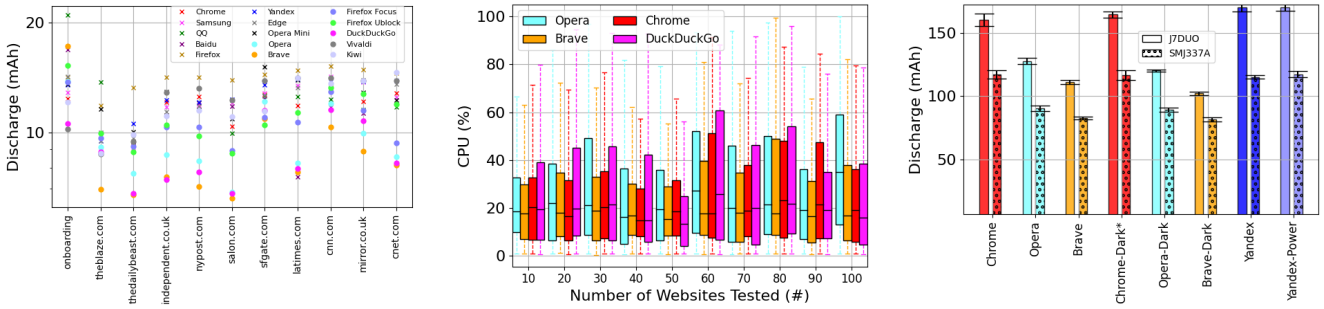
As the number of concurrently opened tabs increases, Brave's CPU consumption becomes consistently higher than Opera – and DuckDuckGo up to 50 tabs – suggesting that Opera is more efficient in handling a higher number of tabs. Even when compared with Chrome, Opera does a better job in keeping the CPU consumption bound. For instance, Opera rarely passes 80% CPU utilization, only once with 60 concurrent tabs. In comparison, both Chrome and Brave reach such value (and higher) multiple times. A split behavior is observable for DuckDuckGo: low CPU consumption up to 50 tabs, rapidly increasing CPU usage between 50 and 80 tabs (median close to 80%), and then significant reduction of CPU usage when the last 20 tabs are added. We have verified a similar behavior across workloads, with the caveat that the increase load might happen earlier – 10 tabs earlier, for example, when testing the majestic-100 list. Although hard to explain *why* we observe this specific behavior without access to the code, this result suggests a pretty evident bug or inefficiency in DuckDuckGo tab management code.

To further explore the latter result, we modified the 100 tabs experiment to ensure a maximum of 10 concurrent tabs, *i.e.,* each 10 tabs we close all previous tabs and then resume a test. Figure 3(b) shows boxplots of the CPU consumption each 10 tabs. In this case Brave consumes consistently less CPU of Opera (and Chrome), as observed in Figure 1(c) for the news workload. Similarly, DuckDuckGo has a more consistent CPU consumption, strengthening the above observations.

C. Power Saving Features

The previous subsections show that *adblocking* results in significant power savings. We here generalize the analysis with respect to other techniques, namely *dark mode* – which can potentially save battery by allowing to turn pixels off on AMOLED screens – and *Yandex power saving mode*, to the best of our knowledge the only explicit power saving feature offered by an Android browser. For this analysis, we introduce the second device (J7DUO) which is equipped with an AMOLED screen, *i.e.,* dark pixels are indeed turned off thus saving energy. Based on the previous results, for this experiment we only focus on Chrome, Opera, and Brave, in addition to Yandex.

Figure 3(c) shows the energy consumption across browsers and devices, assuming the news workload. Regardless of the device, the trend is the same as the one observed before, with the most aggressive adblocking browser (Brave) bringing the



(a) Energy consumption per tab during the execution of the news workload; SMJ337A.

(b) 100 websites test in block of 10 concurrent tabs ; Tranco top100 ; SMJ337A.

(c) Battery discharge of adblocking, dark mode, and Yandex power saving; News workload ; J7DUO and SMJ337A (circle markers).

Fig. 3.

highest battery savings. In addition, dark mode offers about 11-13% extra savings on the J7DUO for Opera and Brave, respectively. For Chrome-Dark we do not measure additional savings; the "*" indicate that for Chrome we selected the basic dark mode settings – *i.e.,* the one available via GUI without accessing *chrome://flags* – which does not darken the page but only the browser's GUI. We purposely selected this mode to measure potential benefits from this feature, which are negligible in our tests. As expected, dark mode only brings benefits for the J7DUO (AMOLED screen).

We did not observe any difference when activating Yandex's power saving mode, despite the 9% battery saving advertised. The figure also shows that the less powerful SMJ337A also consumes ∼20% less than the J7DUO. This could be due to many things, such as the bigger screen and overall more advanced hardware to be powered.

D. Experiments in the Wild

Finally, we compare browsers performance in the *wild*, *i.e.,* when used on the go over a mobile network. We use *battor* [18], a portable power meter powered directly by the device. Given the lack of WiFi in these tests, we leverage Bluetooth to allow the Rasberry Pi to instrument the SMJ337A, e.g., to clean the browser internal state and load webpages. Finally, a power bank (10,000 mAh, 5.0V, 2.0A) allows to power the Pi on the go for up to 20hrs.

We installed Figure 4(a)'s setup in a car and performed experiments at 10 locations within a 25 miles range in North America, while connected to Mint mobile [28]. At each location, while stationary, we run the news workload using Chrome, Brave, and Opera. We perform a single run per location given that: a) previous experiments have shown low standard deviation, b) the variance is provided by the different locations, c) we need to bound the experiment duration (each run currently takes 30 minutes).

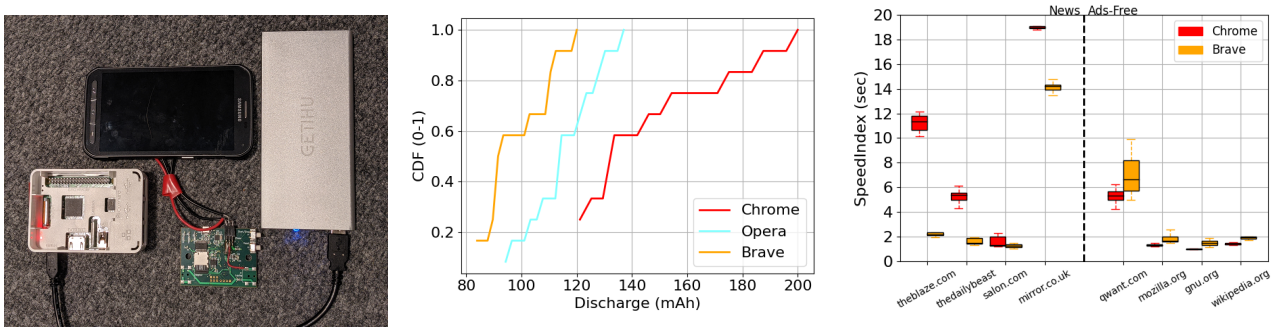
Figure 4(b) shows the CDF of the energy consumption across 10 locations. The figure further confirms the energy benefits of adblocking (Brave and Opera) even on a mobile network with variable signal quality, e.g., the median consumption grows from 92mAh (Brave) up to 131mAh for Chrome. This is a 30% improvement, similar to what measured before. As the network conditions deteriorate – causing an overall increase in energy consumption – Brave and Opera lower bandwidth utilization allows further reduction (up now to 40%, 120 vs 200mAh, for Brave). Note that this result is also a potential lower bound of the saving, given that in our experiments we synchronize browsers every 30 seconds, *i.e.,* faster browsers are forced to wait before loading another page.

To further comment on the latter observation, we perform a second experiment investigating how fast are pages loaded at the 10 locations above. To report on performance metrics, we rely on lighthouse [29]. Specifically, we forward the developer tool port (9222) from the device to the Rasberry Pi where it is used by lighthouse to instrument a browser. Surprisingly, we could not communicate with Opera's developer tool port, and we thus omitted it in this test. With respect to the websites to be tested, we used 4 websites from news and ads-free workload, respectively.

Figure 4(c) shows per website boxplots of the SpeedIndex [30], the average time at which webpages are rendered, distinguishing between browsers, and news (on the left) and ads-free (on the right) websites. With respect to news websites, the figure shows that Brave (and adblocking in general) offers savings of seconds, e.g., median SpeedIndex of 2.5 versus 11 seconds for `theblaze.com`. By discounting the energy needed during this extra time, Brave would offer an extra 30% of battery savings, on average. The result is instead reversed for ads-free websites, where Chrome consistently outperforms Brave by few hundreds ms and up to 1.5 second.

IV. AttentionDim

Under *bad* network conditions a user can spend tens of seconds waiting for a webpage to load (see Figure 4(c)), and maybe even give up with no content displayed. Our intuition is to minimize the screen power consumption during these wasted times, by simply *dimming* the screen brightness – which can save energy on both AMOLED and LCD devices (Table IV) differently from dark mode (Figure 3(c)). We thus propose "AttentionDim", a screen dimming strategy which



(a) Portable test-bed. (b) CDF of energy consumption. (c) Boxplots of SpeedIndex per website.

Fig. 4. Performance evaluation in the wild (10 locations within 25 miles) ; Chrome, Brave, Opera ; SMJ337A ; 4G ; News workload.

leverages the browser state, e.g., loading versus content ready, to control screen brightness.

We motivate this idea by investigating the *potential* savings deriving from screen dimming. Table IV shows, for several increasing brightness values in Android (*i.e.,* 0-250 range) and corresponding scenarios where they apply, the median current (mA) measured on both J7DUO and SMJ337A during one minute displaying a default Android desktop theme. The table further extrapolates the potential battery savings coming from *full* screen dimming, *i.e.,* dropping the screen brightness to zero, as well as a more *conservative* strategy we will detail below. This experiment shows that, even with a conservative strategy, screen dimming offers potential savings between 17 and 40%, on both AMOLED and LCD-equipped devices.

A. Design and Implementation

The idea behind AttentionDim is to use `onLoad()`, a browser event which signals when a page is loaded, as an approximation of user *attention* which requires regular screen brightness. Other "events" are possible, e.g., video buffering, but requires more complex browser modifications and were thus left as future work.

AttentionDim is implemented as a module which controls the screen brightness from the browser. This module currently sits in `ChromeTabbedActivity`, *i.e.,* it can be adopted by all Chromium-based browsers, and it is triggered by the above events to *dim* the screen and then *restore* the brightness when the event completes. When dimming is triggered, this module detects whether the user is using *auto* or *manual* brightness so that it can: 1) manually set brightness to the last value when the event completes, 2) reactivate auto dimming and let the OS decide the brightness value to be used.

We experimented with several dimming strategies and then settle for the following one based on feedback received from our volunteers. When the original screen brightness is *low* (*i.e.,* ¡= 100) we opt for an aggressive strategy, *i.e.,* we lower the screen brightness B to zero. For *mid* brightness values (e.g., 150) we set B to half of the original value (50 and 75). We instead use a fixed $B = 150$ for *high* values, since outdoor and sunny conditions are quite challenging and we need to prevent leaving users in the dark. Last but not least, we implemented a setting option and a GUI to allow users to simply (de)activate AttentionDim as they wish, and to get an estimation of the current battery savings.

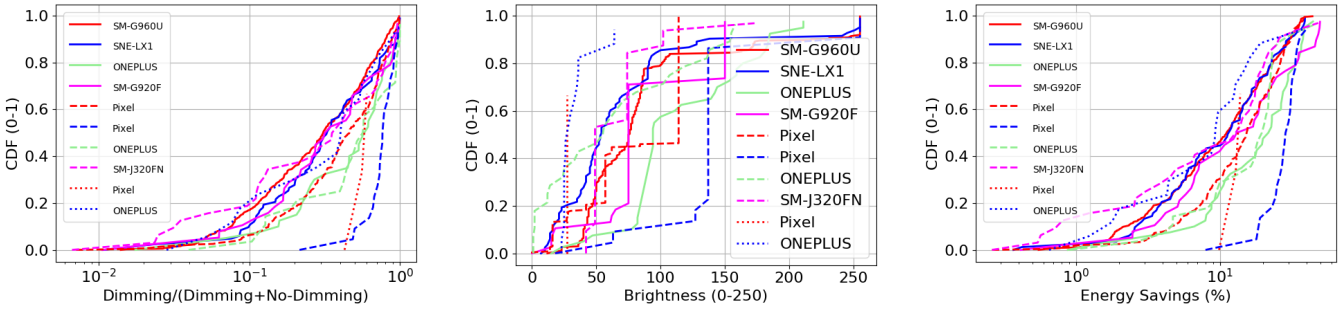
B. Performance Evaluation

We recruited 10 Android users who installed our modified version of Brave for up to 30 consecutive days totaling about 500 hours of browsing – we urged our volunteers to use the browser as normal. Figure 5(a) shows the CDF of the fraction of time spent dimming, per device. Note that some volunteers shared the same device model: 3 Pixel and 2 ONEPLUS. The CDF is calculated using the beginning of a dimming event as both the start time of such event and the end time of the previous non-dimming event. Start/end timers are also triggered whenever the user closes or (re)launch the browser. The amount of dimming is very much user and time-dependent, meaning that some users experience a higher amount of dimming as well as the dimming duration spans a broad distribution. Generally speaking, very short dimming events (e.g., lower than 10%) are rare. Across users, the median dimming event lasts between a minimum of 30 and a maximum of 70% of the time.

Figure 5(b) shows the CDF (per device) of the screen brightness measured during one month. Most brightness values reported are smaller than 100 (indoor usage). One of the Pixel devices is an exception since most values reported were quite high, either because of outdoor or manually set. It has to be

Brightness	Scenario	Current (mA, median)	Savings (Aggr.)	Savings (Cons.)
0	-	145/108	0/0%	0/0%
50	Indoor	189/130	23/17%	23/17%
100	Indoor	239/157	39/31%	39/31%
150	Cloudy Outdoor	299/201	51/46%	28/28%
200	Outdoor	379/243	61/55%	21/17%
250	Sunny Outdoor	417/247	65/56%	28/17%

TABLE IV
POWER SAVINGS FROM SCREEN DIMMING (J7DUO/SMJ337A). AGGR. STANDS FOR *aggressive*, CONS. STANDS FOR *conservative*. BRIGHTNESS IS DEFINED IN THE RANGE 0-250, AS PER ANDROID.



(a) CDF of time spent dimming. (b) CDF of screen brightness. (c) CDF of estimated energy savings.

Fig. 5. Performance evaluation of AttentionDim in the wild ; 10 Android users ; 30 consecutive days and 500 hours of browsing.

noted that this device was also only used for a limited amount of time, as the sharp CDF suggests. Finally, we combine the information from Figure 5(a), 5(b), and Table IV to estimate actual battery savings. Figure 5(c) shows encouraging battery savings of about 20-30%, which means potentially extending the battery life by up to one hour.

C. User Study

The previous subsection shows that AttentionDim is effective in reducing the energy consumption across different users and devices. This was achieved by opportunistically lowering the screen brightness, which can have an impact on the *user experience*. We asked our 10 study participants to fill out a form reporting on their user experience while using AttentionDim. The form asked few simple questions, on which we report in the following.

We started by asking: *from 1 to 5, how would you rate the overall experience with AttentionDim?* The outcome was eight 5s and two 4s (*i.e.,* mean opinion score of 4.8), suggesting that our participants were overall pleased by AttentionDim. Next, we asked: *in which scenarios, e.g., indoor vs a cloudy or sunny outdoor, does AttentionDim perform the best and the worst?* Two volunteers indicated that it performs quite well across all situations they encountered. The remainder eight volunteers indicate that AttentionDim performs the best indoor and the worst outdoor. Finally, we asked for feedback on AttentionDim's *responsiveness*, or how quickly was the screen's brightness resumed when needed. Most participants report on AttentionDim lagging, *i.e.,* taking few extra seconds to resume the screen's brightness. This was perceived as a *feature* rather than a *bug*: the users reported how, over time, they used the screen dimming/resuming as an indication that a page was indeed ready to be browsed.

From this user study, we have identified two areas of improvement and further investigation. The survey shows that AttentionDim currently underperforms in sunny outdoors. This is intuitive given that mobile phone usage in the sun is already problematic. AttentionDim currently leverages Android' screen brightness information (an integer in the range 0-255) to drive its algorithm. This measure is quite coarse especially for high values, and we thus plan to replace it with data from the light sensor. This will offer higher accuracy, hopefully improving AttentionDim's algorithm in sunny outdoors.

The second observation has to do with AttentionDim's slow responsiveness. There are two main causes of the lag experienced by our users. First, `onLoad()` is a conservative page load time metric since it waits for *all* content to be loaded, e.g., even tracking scripts which do not contribute to the appearance or interactivity of a page. Second, Android requires some *delay* (depending on the device) to change the screen brightness. Further, some users might be more impatient than others and just start scrolling even before some or most of the content is retrieved.

To increase responsiveness, we could introduce a new metric, potentially even incorporating the above Android's delay. However, this would reduce the energy savings and introduce other problems, since no metric is perfect [31]. Alternatively, we could abandon page load time metrics and focus on user behavior, e.g., trigger brightness resumption as the user interacts with the page. We could even allow users to explicitly define their own strategy, or offer feedback from which the algorithm can learn over time.

V. Conclusion

This paper has investigated the battery consumption of 15 Android browsers (7 popular and 8 with adblocking features), 3 of the top performing browsers in *dark mode*, Yandex power saving feature, and *AttentionDim*, a novel screen dimming mechanism driven by browser events like `onLoad()` which we have developed. Given the scale of these measurements, we have also built a browser testing suite which is both transparent and extensible. Our results show that *adblocking* offers significant battery savings (up to 30% on WiFi and 40% on 4G) which can be further enhanced via *dark mode* (extra 12%), when applied to whole webpage content and AMOLED devices. Conversely, Yandex power saving feature resulted more a marketing stunt than a batter saving solution. We integrated AttentionDim in Brave – one of the top performing browsers in our tests – and run a study involving 10 users and up to 500 hours of browsing in the wild. Our results show that AttentionDim reduced, on average, the battery consumption by about 20-30%, with minimal impact on the user experience.

References

[1] N. Thiagarajan, G. Aggarwal, A. Nicoara, D. Boneh, and J. P. Singh, "Who killed my battery? analyzing mobile browser energy consumption," in *Proceedings of the 21st international conference on World Wide Web*, 2012, pp. 41–50.

[2] A. Albasir, K. Naik, B. Plourde, and N. Goel, "Experimental study of energy and bandwidth costs of web advertisements on smartphones," in *6th International Conference on Mobile Computing, Applications and Services*. IEEE, 2014, pp. 90–97.

[3] D. H. Bui, Y. Liu, H. Kim, I. Shin, and F. Zhao, "Rethinking energy-performance trade-off in mobile web page loading," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, 2015, pp. 14–26.

[4] Y. Cao, J. Nejati, M. Wajahat, A. Balasubramanian, and A. Gandhi, "Deconstructing the energy consumption of the mobile page load," *Proc. of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 1, pp. 6:1–6:25, Jun. 2017.

[5] N. Heitmann, B. Pirker, S. Park, and S. Chakraborty, "Towards building better mobile web browsers for ad blocking: The energy perspective," in *The 21st ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems*, 2020, pp. 146–150.

[6] Browser Share, "Browser Market Share Worldwide," https://gs.statcounter.com/browser-market-share.

[7] AdBlock, "Block ads. browse better and safer." https://getadblock.com/.

[8] UBlock, "The fastest, most-powerful ad blocker." https://ublock.org/.

[9] Brave, "The browser re-imagined." https://brave.com/.

[10] Kiwi, "Kiwibrowser, fast and quiet." https://kiwibrowser.com/.

[11] Opera, "A browser for the real you." https://www.opera.com/.

[12] Google, "Browse in Dark mode," https://support.google.com/chrome/answer/9275525?co=GENIE.Platform%3DDesktop&hl=en.

[13] Yandex, "Power saving mode," https://yandex.com/support/browser-mobile-android-phone/economy-mode/energy-saving.html.

[14] M. Varvello, "Cappuccino: Third Party Application Testing for Android," https://github.com/svarvel/cappuccino.

[15] Netmarketshare, "Market Share Statistics for Internet Technologies," 2020, https://netmarketshare.com.

[16] WICG, "User Agent Client Hints," https://github.com/WICG/ua-client-hints.

[17] Monsoon Solutions Inc., "High voltage power monitor." https://www.msoon.com.

[18] A. Schulman, T. Schmid, P. Dutta, and N. Spring, "Phone power monitoring with battor," in *Proc. ACM MobiCom*, 2011.

[19] Patrick Dubroy, "How many tabs do people use? (Now with real data!)," https://dubroy.com/blog/how-many-tabs-do-people-use-now-with-real-data/.

[20] E. Zeng, T. Kohno, and F. Roesner, "Bad news: Clickbait and deceptive ads on news and misinformation websites," in *Workshop on Technology and Consumer Protection (ConPro). IEEE, New York, NY*, 2020.

[21] V. L. Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczyński, and W. Joosen, "Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation," https://tranco-list.eu/.

[22] Fanboy, MonztA, Famlam and Khrin, "EasyList - Overview," https://easylist.to/.

[23] Majestic, "Find out who links to your website," https://majestic.com/reports/majestic-million.

[24] 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*, 2020, pp. 680–695.

[25] Hispar, "A top list of web pages; not domains," https://hispar.cs.duke.edu/.

[26] Alexa, "The top 500 sites on the web," https://www.alexa.com/topsites.

[27] Q. Scheitle, O. Hohlfeld, J. Gamba, J. Jelten, T. Zimmermann, S. D. Strowes, and N. Vallina-Rodriguez, "A long way to the top: Significance, structure, and stability of internet top lists," in *Proceedings of the Internet Measurement Conference 2018*, 2018, pp. 478–493.

[28] Mint Mobile, "Wireless that's Easy," https://www.mintmobile.com/.

[29] Google, "Lighthouse, a Tool for Web Developers." https://developers.google.com/web/tools/lighthouse.

[30] C. Arsenault, "Speed Index Explained - Another Way to Measure Web Performance," https://www.keycdn.com/blog/speed-index.

[31] F. Salutari, D. Da Hora, M. Varvello, R. Teixeira, V. Christophides, and D. Rossi, "Implications of the multi-modality of user perceived page load time," in *2020 Mediterranean Communication and Computer Networking Conference (MedComNet)*. IEEE, 2020, pp. 1–8.