The Cost of Ads and Analytics in Mobile Web Apps

Abdul Qadir Ahmed Abbasi
2769474
VU Amsterdam
a.q.a.abbasi@student.vu.nl

Haben Birhane Gebreweld
2692682
VU Amsterdam
h.gebreweld@student.vu.nl

Leroy Velzel
2569971
VU Amsterdam
l.q.velzel@student.vu.nl

Micky Yun Chan
2768212
VU Amsterdam
m.y.chan@student.vu.nl

Ram Prasad Gurung
2770576
VU Amsterdam
r.p.gurung@student.vu.nl

ABSTRACT

***Context*:** Ads are web elements that are designed to promote services or products that are irrelevant to the visited website. In contrast analytics are scripts that run in the backgrounds to track visitors activity on a website. Analytics provides important insight to the website owner while Ads monetize web traffic. Thus it form an important part of a web app.
***Goal*:** This paper aims to assess the impact of ads and analytics on energy consumption and performance of web apps in Android OS.
***Method*:** Empirical assessment was carried out with Android runner targeting 10 different web apps. The experiment is designed as a 1 factor - 2 treatments study, with ads and analytics as the single factors and the presence of ads and analytics as treatments. The response variables of the experiments are (i) the energy consumption of the mobile device and (ii) the performance metrics(CPU and memory usage) of the mobile device. The experiments are executed on a real Android device.
***Result*:** The results of our experiments show that web apps without ads and analytics consume significantly lower energy and memory compared to web apps that has, with a small effect size. But there is negligible difference in CPU usage.
***Conclusions*:** We recommend web app developers to limit the use of ads and analytics. Moreover, based on the empirical evidence of experiments conducted, the study confirms that ads and analytics consume significantly higher energy and memory in a Android mobile, whereas the differences in CPU usage are negligible.

ACM Reference Format:
Abdul Qadir Ahmed Abbasi, Haben Birhane Gebreweld, Leroy Velzel, Micky Yun Chan, and Ram Prasad Gurung. 2022. The Cost of Ads and Analytics in Mobile Web Apps . In *Green Lab 2022/2023 - Vrije Universiteit Amsterdam, September–October, 2022, Amsterdam (The Netherlands).* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Green Lab 2022/2023, September–October, 2022, Amsterdam, The Netherlands

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

It is well known that ads and analytics in mobile web apps increase the latency of response time and energy consumption[1]; this could have a detrimental effect on user experiences as well as the battery health of visiting devices. A study from the Netherlands shows that the additional energy consumption to display web advertisements is 2.5W[2]. Besides the extra energy consumption, ads can lead to battery drainage and high traffic and this, in turn, drops the internet connection speed for the intended task such as loading web pages. Therefore, it is essential for app developers to gain an understanding of the impact of ads and analytics in mobile web apps. Businesses that fail to do so may lose market shares to competitors that do. For the sake of simplicity, ads embedded in web apps can be grouped into two types:

- Banner - as shown in Figure 1 Ads that are embedded in the web page
- Pop up - as shown in Figure 2 Ads that are not embedded in the web page.

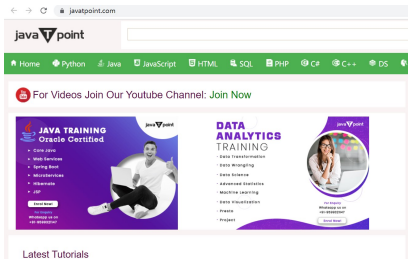

Figure 1: Banner Ads

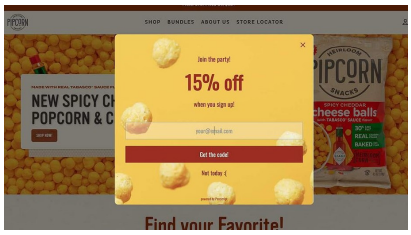

Figure 2: Pop up Ads

It is important to note that each type of ad can be further subdivided into different categories depending on the content such as video, text, etc. The difference in content can make a substantial difference in terms of performance and energy consumption[3]. While both types of ads can be implemented using HTML, CSS, and JS, the majority of ads are created using Meta ads[1] and Google ads [2]. Using Google ads as an example, once an ads campaign is created, the ads will be distributed and displayed on targeted web apps on the Google ad sense network. Web analytics on other hand are javascript codes that are embedded in the header section of webpages, visitors usually do not have visuals on them. These scripts actively monitor and record user activities on the web app including but not limited to mouse clicks, IP address, time spent on a particular page, etc. It is possible to avoid ads and analytics through extensions *e.g.,* Adblock[3], block yourself from Analytic on Google extension. While such an extension is not available on a mobile platform, browsers such as the mobile version of Chrome has built-in ad blocker [4], and additionally, it is possible to manually add rules that block analytics on specialized browsers such as Brave. If such option is not available, the ad script can be remove manually.

The main contribution of this paper will be the following:

- Empirical assessment result on Android mobile web App with, without ads and analytics on two different mobile browsers
- Interpretation of results mentioned above

The experiment is conducted according to the guidelines by Wohlin and colleagues [5]. The result and conclusion of this experiment can provide an insight of energy consumption of ads and web analytics in android browsers.

2 RELATED WORK

In this section, we will summarize some of the relevant research and describe how our research differs from the selected papers. We found out that there has been several research done on the cost of ads in performance and energy consumption, mostly in native mobile apps, whereas there is limited research done regarding the cost of analytics in performance and energy consumption. Moreover, in most of the papers, they have either chosen energy consumption or performance but not both in their studies.

Nagappan et al.[6] studied the hidden cost of mobile advertisement for software developers. They chose 21 real-time apps from Google Playstore to analyze 5 types of hidden costs: performance, energy consumption, network usage, ad-related code maintenance effort, and app reviews. The team discovered that on average ads consume 48% more CPU usage, 16% more energy, 79% more network data, and 23% ads-related changes in app releases. With these results, they suggested developers weigh the tradeoffs of incorporating ads in their mobile apps for better end users' experience. Their experiment treated only ads versions and non-ads versions, but in our experiment, we will examine ads and analytics web apps version in opposite to without ads and analytics web apps version. Apart from this, our experiment will only consider the performance and energy consumption of Android mobile web apps. Additionally, the team only considered the software developer's point of view, whereas we will consider both developer's and researcher's viewpoint.

Halfond et al.[7] proposed and evaluated two lightweight statistical approaches for measuring and predicting ad-related energy consumption. In the first approach, a statistical model that uses information from the ad's unit profile–a piece of configuration information like ad type and refresh rate that a publisher requests from an ad network–is built to get early feedback on the energy consumption of an application even without a working implementation. In the second approach, some key metrics were introduced to provide more accurate information about ad related energy consumption of an app. To evaluate their approach, they used 13 real market apps. In their evaluation, the static model was accurate within 31% of the ground truth to estimate the energy cost of ads, whereas the runtime model was accurate within 14% of the ground truth to estimate the energy cost of ads. Furthermore, this study provides feedback about the energy consumption during the design and early implementation phase, whereas our study is based on web apps that have already been implemented. Another difference is that our research focuses on both energy consumption and performance cost of mobile ads and analytics.

Lyu et al. [8] conducted a study on analyzing user concerns about ad's performance costs. For this, the team proposed RankMiner, an approach to quantify specific app issues of users. CrossMiner, an issue ranking framework, is used as a baseline to verify the effectiveness of RankMiner. They studied the usage traces of 20 popular apps to find the performance cost of ads. In this study, 4 types of performance cost–memory consumption, CPU utilization, network usage, and battery drainage– were measured. As a result, they found out that the performance cost of with-ads version apps are significantly larger than those of non-ads apps version, and users are more concerned with the battery cost of ads. Our research doesn't focus on the users' rankings of the different ad's costs, but from the viewpoint of software developers and researchers, and the inclusion of the cost of analytics on both energy consumption and performance can add a new dimension to the literature.

Abdurhman et al. [9] evaluated the energy and bandwidth cost (over Wi-Fi/3G networks) for downloading, rendering and displaying web ads in 5 well known websites. Energy consumption of the user analytics, which plays a significant role in showing relevant ads based on user's behaviour and data is interesting to be explored. Additionally, their experiments are either limited to Android Browser or a Custom Built Browser ("semi web browser") on Galaxy Nexus mobile. Due to variations of ad content for each network request, it was observed that 4 to 10 seconds of time overhead is taken by each ad. It was also found that 3.5 to 12 Joules over Wi-Fi and 3G networks is the energy utilized due to ads which makes it 6 to 18% of total web browsing consumption. Core motivation of our experimentation revolves around the elimination of fragments of Javascript code in the mobile web apps not only showing advertisements but also responsible for tracking user and then doing analysis on the performance and energy cost of mobile web apps running on the targeted browsers i.e. Google Chrome and Mozilla Firefox.

[1] https://www.facebook.com/business
[2] https://ads.google.com/
[3] https://adblockplus.org/

The content of ads running on the electronic devices i.e. laptop/-mobile, speed of the internet over network, type of browsers and ad blockers used for blocking ads dictates the performance and energy cost. Measurements of Behnam et al. [10] says that Adblock Plus (one of the most popular ad blocker) adds almost 32% of overhead on average for page loading leading to poor user experience and performance. Hence simply using ad blockers even without existence of any ad will still consume sufficient amount of energy. Exploring the impact with the usage of ad blocker on web pages with and without ads is not sufficient for the detailed analysis of actual performance and energy cost. We are interested in cost of both ads and analytics without relying on tracking and ad blockers.

In the past, research done on desktops (PCs) by Simons and Pras [2] for investigating the power consumption of web advertisements during normal continuous browsing left a trail for smaller-sized devices. Experiments were done with and without ad/tracker blocked on browsers (i.e. Mozilla Firefox, Apple Safari, Internet Explorer, and Opera). Browsers had dedicated configurations that were running on 4 different processor chips from AMD and Intel. Results revealed that energy consumption is almost equivalent to the electricity usage of 1891 households in the Netherlands. Blocking strategies used were either black-listing URLs or by filtering the HTML ad content from showing on the screen. In the time span of more than a decade after that experiment, today the ads and analytics have evolved to great extent and the cost has increased significantly. Our motivation is focused on the challenge which comes with mobile devices as they have limited resources on chips without having any continuous power supply. Therefore, the cost of advertisements and analytics for personalized ads running on Android mobile web apps are required to be investigated considering the factors i.e. Android mobile apps with and without ads plus analytics.

3 EXPERIMENT DEFINITION

We used the **GQM** approach[11] to further define the experiment. The goal of this experiment is:

To investigate ads and user tracking/analytics to evaluate their impact on energy consumption and performance from the viewpoint of developers and researchers in the context of Android mobile web apps.

To achieve the goal described above, we identified two main areas it could affect, i.e. performance and energy consumption. Therefore, we refined our goal into the following three research questions:

[RQ1] *What is the overhead imposed by ads and user tracking/-analytics on the energy consumption of mobile web apps?*

To evaluate the energy consumption of the ads and user tracking/analytics, we calculate the extra joules of energy consumed while the ads and user tracking/analytics are running in comparison to them being removed. This value will give us clear metrics on the overhead set by the ads and user tracking/analytics on the Android mobile web apps' energy consumption.

[RQ2] *What is the overhead imposed by ads and user tracking/-analytics on the performance of mobile web apps?*

We have further devided this **[RQ]** in to two.

[RQ2.1] *What is the overhead set by ads and analytics on the Memory usage of mobile web apps?*

[RQ2.2] *What is the overhead set by ads and analytics on the CPU usage of mobile web apps?*

To measure this, we will analyze the CPU and Memory usage of the mobile web apps with the ads and user tracking/analytics intact compared to them removed. The results we get from this experiment should give us an insight into the overhead imposed on mobile web apps by the ads and analytics code blocks.

These two dependent variable, i.e. energy consumption and performance have been verified to be suitable for this experiment. It has been shown in multiple researches that both energy consumption[12] and performance[13] have a impact on the usability of mobile devices and web applications.

It is obvious that different mobile operating systems have different characteristics. These characteristics can have an impact on energy consumption, as well as the performance of web apps. Therefore we have decided to limit our experiment to one operating system namely Android. It is the leading operating system installed on mobile devices[14] and can be used in combination with most well-established web browsers.

We have limited our research to the web browsers Google Chrome and Firefox. The first one being the most installed browser, while the latter one is of the best-rated browsers (with 4.5 reviews) in the application store for Android(Google Play store).

To realize the goal we set out, we chose **metrics** that would answer our research quations quantitatively. For this purpose we have identified the following metrics:

- **Energy Consumption:** we will use **Trepn**[4] to measure the energy consumption of the Android mobile web apps before the ads and user tracking/analytics are removed and after, and is measured in Joule.
- **CPU Usage:** we will collect the cpu usage from **cpuinfo** Android utilities found in ADB's *dumpsys*, and is presented in **percent**.
- **Memory Usage:** we will collect the memory usage from **meminfo** Android utilities found in ADB's *dumpsys*, and measured in **KB**.

4 EXPERIMENT PLANNING

Considering the main objective of this experiment, i.e. to investigate the cost of ads and analytics in the context of Android mobile web apps, we planned the experiment to achieve this goal in a systematic and replicable manner.

[4]https://github.com/S2-group/android-runner/tree/master/AndroidRunner/Plugins/trepn

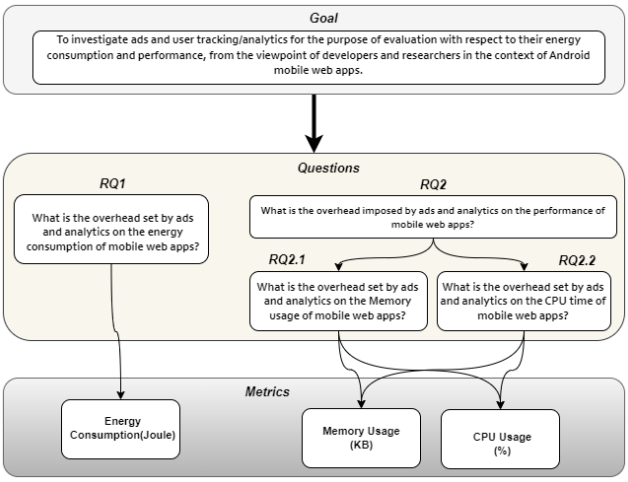

Figure 3: Visual representation of the GQM

4.1 Subjects Selection

For subject selection of this experiment, we consider the selection of android mobile device, web browsers, and web apps. Now, there is a large population of android mobile devices since the release of the first android phone in October 2008[5]. In our experiment, we select 2015 released Huawei Nexus 6P with Android version 6.0.1. This device selection is on account of compatibility with Trepn[6] which is only compatible with Android 6.

Out of all available browsers, we have selected Google Chrome and Mozilla Firefox. In Google play store[7], both of these browsers have ratings above 4.0 with the total downloads of 10B+ for Google Chrome and 150M+ for Mozilla Firefox. Furthermore, Google Chrome and Mozilla Firefox are among the top browsers recommended for Android[8].

At the early stage of mobile web apps selection, we randomly select 50 web apps that are within the top 1000 of the Tranco list[15], a rank listing service that provides a list of the top 1 million popular web apps. The rank of web apps is limited to 1000 so that the selected subjects can be among the top-accessed web apps. By this, the reader can relate to the impact of their visited web apps. Out of 50 web apps, only 10 are selected. The 10 selected web apps are listed in the Table 2. Before selecting the final 10 web apps out of 50 web apps, firstly we check if the web app contains both ads and analytics with the help of *builtwith*[9], an online tool that finds out what a web app is built with. In addition, the web apps are also checked manually by debugging to confirm if they contain both ads and analytics. The criteria for selecting web apps are listed in the Table 1.

The following table contains all of the 10 selected web apps together with their name, URL, and Tranco rank:

[5]https://www.cnet.com/tech/mobile/a-brief-history-of-android-phones/
[6]https://github.com/S2-group/android-runner/tree/master/AndroidRunner/Plugins/trepn
[7]https://play.google.com/
[8]https://www.androidpolice.com/best-android-web-browser/
[9]https://builtwith.com

Table 1: Inclusion Criteria(IC) for web apps selection

Criteria	Description
IC1	Web app ranking should be < 1000
IC2	Web apps should contain both ads and analytics
IC3	Ads should be visible withing pages of web apps

Table 2: Subject of the study

Web apps	URL	Tranco rank
The Mirror	http://mirror.co.uk/	926
IMDB	https://www.imdb.com/	111
Science	https://www.science.org/	975
Cnblogs	https://www.cnblogs.com	294
Speedtest	https://www.speedtest.net/	437
AccuWeather	https://www.accuweather.com/	784
BBC	https://www.bbc.com/	139
LiveJournal	https://www.livejournal.com/	418
Investing	https://www.investing.com/	647
Psychology Today	https://www.psychologytoday.com/	723

4.2 Experimental Variables

To find out the answers for the research questions stated in Section 3, we consider ads and analytics as our main independent variable. Hence, the two treatments deduced for ads and analytics are:

a) *ads and analytics are removed completely*
b) *ads and analytics are not removed at all*

Experiment is either served with the original version (without elimination of ads and analytics related code) of web app or with the modified version where relevant code removal treatment is applied. In the middle, a *proxy server* is used as a common factor for both treatments while serving the web app from server to the client. To achieve the maximum reliability of our experiment, eliminations of ads and analytics is done programmatically using a *Python script*, which enabled us to intercept and change (i.e. removal of ads and analytics related code snippets) the response before it is served. Since the primary focus of our experiment is the cost on the end user due to ads and analytics, the overhead imposed by the *Python script* is ignored because it is on the middle server (i.e. in this case the laptop with proxy), and not on the end client (i.e. mobile).

In our experiment, the type of browser is also considered as a blocking factor because each browser handles the web app functionalities i.e. rendering, visualization, and fetching content over network etc. differently. It is due to the fact that browsers use completely different search engines internally i.e. Blink[10] and Gecko[11] in our case. Therefore, investigating them separately is important for in-depth analysis of performance and energy cost. The two types of browsers under considerations for this research are: *Mozilla Firefox* and *Google Chrome*.

[10]https://www.chromium.org/blink/
[11]https://developer.mozilla.org/en-US/docs/Glossary/Gecko

Energy Consumption, *CPU Usage* and *Memory Usage* are the three opted dependent variables corresponding to the metrics defined earlier using the **GQM** framework[11]. Table 3: shows the names and descriptions of these variables. They are used to evaluate the cost of performance and energy of ads and analytics in the mobile web apps.

Table 3: Dependent variables considered for this research

Name	Description
Energy consumption	Energy consumption in Joule (J) consumed by the mobile web browser for using the web app
CPU usage	CPU percentage (%) used by the mobile web browser for the web app usage
Memory usage	Memory in Kilobyte (KB) occupied by the mobile web browser for the web app usage

4.3 Experimental Hypotheses

As stated in the experiment definition we will calculate the difference for each of the variables. The calculation is stated as μ_{vbr}, which is the mean value of v. v is used to describe the dependent variables $\{e, c, m\}$ as stated in the Table 3, b describes the different browsers that are used, while r represents a subset of settings enabled for a web app $\{true, false\}$.

We assume that for both settings, the same amount of code is being executed at most. It is most likely the case that when the code size is reduced with the application's functionality remaining the same, less code is being executed, which leads to a decline in the energy consumption, CPU, and Memory usage of the mobile device. Due to this reason, the nature of our experiment is a one-tailed hypothesis.

For energy consumption:

$$H_{0,br} : \mu_{ebr} = \mu_{ebr}, \forall b \wedge \forall r$$

$$H_{a,b} : \mu_{ebfalse} \leq \mu_{ebtrue}, \forall b$$

For CPU usage:

$$H_{0,br} : \mu_{cbr} = \mu_{cbr}, \forall b \wedge \forall r$$

$$H_{a,b} : \mu_{cbfalse} \leq \mu_{cbtrue}, \forall b$$

For memory usage:

$$H_{0,br} : \mu_{mbr} = \mu_{mbr}, \forall b \wedge \forall r$$

$$H_{a,b} : \mu_{mbfalse} \leq \mu_{mbtrue}, \forall b$$

4.4 Experiment Design

To analyse the impact of ads and analytics on the performance and energy cost, an experiment of complete 2x2 factorial design is performed for each subject. To measure the dependent variables, each mobile web app is served twice with both the treatments applied on two browsers under consideration. Resultantly, making 40 total trials for a single run covering all the selected subjects i.e. 10 mobile web apps. Measurement of energy consumption, CPU utilization and memory usage can lack consistency due to operating in the actual environment, hence, the results are obtained after repetition of each trial 10 times, eventually making 400 runs in total for the whole experiment. Each web app is executed randomly, so that it can be served with the applied treatment for ads and analytics to avoid any biasness involved in the order of execution during our experiment.

4.5 Data Analysis

We do our quantitative data analysis in four phases: data exploration, normality check, hypothesis testing, and effect size estimation.

Data exploration. Firstly, the collected data is analyzed using descriptive statistics and boxplots to get the first indication about energy consumption, CPU and memory usage.

Normality check. After data exploration, tests are done to check if the data is normally distributed for the dependent variables in both treatments. Normality check of the collected data is investigated by i)visually analyzing the density plots, ii)drawing Q-Q plots–which show the diagonal line for normally distributed data and other structured line for non-normally distributed data, iii) applying the Shapiro-Wilk test with $\alpha = 0.05$, where the test returns p-value greater than 0.05 for normally distributed data and lesser than 0.05 for non-normally distributed data. If the above tests forecast the data isn't normally distributed, then the data will be transformed in order to see the possibility of getting normally distributed data. Data transformation is done by square root, log, and reciprocal methods.

Hypothesis testing. The method of hypothesis testing is selected as per the result of normality check. If the results of normality check conclude that the data is normally distributed, we will apply paired t-test(with $\alpha = 0.05$) as we have two treatments and paired comparison. The paired t-test determines if the mean difference between two sets of treatments,with or without ads and analytics, is zero. In the case of non-normally distributed data, the Wilcoxon Signed-Rank test with $\alpha = 0.05$ as significance level is used to check whether there is difference between two treatments.

Effect size estimation. It evaluates the significant impact of ads and analytics on mobile web apps. The type of effect size estimation depends on which test, either parametric or non-parametric, is chosen in the hypothesis testing. In the case of parametric test, Cohen's d is used, whereas for non-parametric test, Cliff's delta is used.

4.6 Study Replicability

In order to help in verification and replication of our experiment, we share the replication package publicly on Github [12]. The replication package contains (i)Python scripts to remove ads and analytics from the web apps (ii)raw data of the experiment (iii) R scripts used for data analysis (iv)relevant diagrams which aren't included in the paper (iv)*README* file containing guidelines for replicating the experiment.

[12]https://github.com/michiboo/ads-analytics-rep-pkg

5 EXPERIMENT EXECUTION

For ease of reference, we have divided the execution of our experiment into three sub-sections as follows.We discuss the preparation of the subjects used for this experiment in the Preparation sub-section, next briefly explain the infrastructure for executing the experiment in the Setup sub-section, and finally, we execute the experiment and describe the measurement process in the Measurement sub-section.

5.1 Preparation

After collecting a list of suitable web apps from Trancolist, we sent individual HTTP requests to each web app. The HTTP responses of the web app and new incoming request-response pairs were monitored and analyzed using the Mitmproxy[13] tool. First, we manually inspect the HTTP responses and locate both ads andanalytics related code. We matched the lines containing this code by creating strict regex expressions matching only to the related code. Second, we made a new HTTP request to the web app and notice less or no request-response pair related to ads and analytics. The last lines of codes were filtered out by a trial-and-error process. We repeated this process for every web apps until no trace of ads and analytics is left on the website.

5.2 Setup

We managed to create a setup of different tools for the automation and execution of the experiment we set out to achieve in section 3. In order to reliably measure energy consumption and performance metrics like Memory and CPU usage of the 10 web app subjects listed in Table , we used Android Runner(AR)[16] for automating the execution of the experiments. Android Runner(AR) is equipped with an orchestrator, device manager, progress manager, and plugin handler, with Android Debug Bridge(ADB)[14] and MonkeyRunner[15] to automate the usage scenarios. Using the mem-CPU(Performance) and Trepn(Energy) plugins available in Android Runner, we will measure the performance metrics, i.e. CPU and memory usage with and without the ads and analytics and the energy consumption of the web apps.

Table 4: List of Tools used for the experiment

Tools	Specifications
Raspberry Pi 4	Model B, 8GB RAM, 64-bit quad-core Cortex-A72 processor, 16GB SD card
Huawei Nexus 6P	Qualcomm MSM8994 Snapdragon 810 (20 nm), Octa-core (4x1.55 GHz Cortex-A53 & 4x2.0 GHz Cortex-A57), Adreno 430, 128GB 3GB RAM, Android 6.01
Chrome Browser	Version 79.0.3945.136
Firefox Browser	Version 93.2.0
Mitmproxy	Version 8.1.1

[13]https://mitmproxy.org/
[14]https://developer.android.com/studio/command-line/adb
[15]https://developer.android.com/studio/test/monkeyrunner

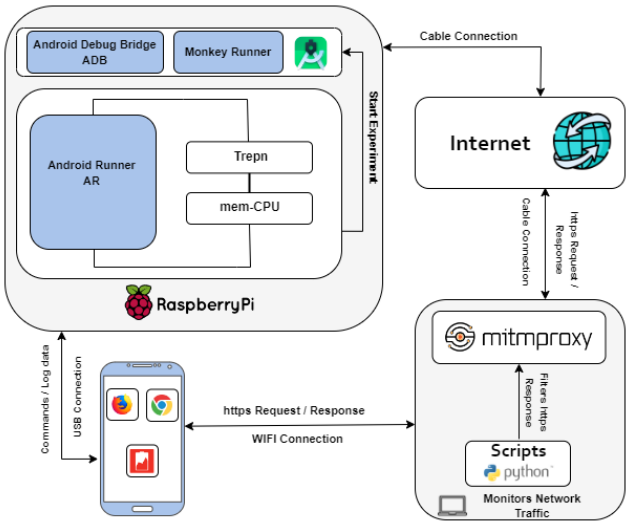

Figure 4: Visual representation of the Experiment Setup

In this experiment, we aim to measure all the dependent variables, and The specifications of all the tools and devices used in this study are shown in Table 4. As shown in Figure 4, We are mainly using three devices to conduct the experiment. On the top left we have a block representing Raspberry PI 4 where the Android Runner(AR), Android Debug Bridge(ADB), and Monkey Runner are all installed. Connected with Raspberry Pi4 through a USB cable on the bottom left of Figure 4 we have Nexus 6P Android mobile device that has both browsers, i.e. Chrome and Firefox, and the Trepn profiler installed on it in order to browse and collect all necessary measures related to the dependent variables of the experiment. We are also connecting the Android mobile device to a laptop that hosts the *mitmproxy*, which is monitoring all the traffics that passes through the Android mobile device via hotspot as shown in the bottom right of the same figure.

In order to handle connection problems both the laptop that hosts our proxy server and the Raspberry Pi 4 are connected to the Router through RJ45[16] data cable, and we made sure that the Android mobile is the only device connected to the laptop's hotspot and sending requests to the *mitmproxy* proxy server. We also took extra precautions by taking actions like configuring the Android mobile device not to perform any OS updates, uninstalling third-party apps, and disabling Google Services and push notifications.

5.3 Measurement

In our experiment, Android Runner[16] is used as an orchestrator. In every run, a web app is automatically launched in both browsers, while energy consumption and performance are calculated by plug-ins installed in Android Runner. The energy consumption of every

[16]https://www.anixter.com/en$_{us}$/*resources*/*literature*/*techbriefs*/*what-is* − *an* − *rj*45 − *connector.html*

run is calculated by Trepn plugin, whereas performance is calculated by mem-CPU plugin, which collects CPU and memory utilization by the cpuinfo and meminfo Android utilities found in ADB's dumpsys[17]. The detailed presentation of dependent variables along with its respective measuring tools are shown in Table 3 of Section 5.2.

Each trial of the experiment is executed by automatically launching each web app in both mobile web browsers. We used the *mitmproxy* as a fixed factor for both treatments. We loaded each treatment separately as it is technically impossible to load and randomize both treatments at the same time. For the first treatment, i.e. "with ads and analytics" case, we run the proxy server without any python script, while for the second treatment, we used a custom-made Python script for each web app that enables us to intercept and edit the response by removing all resources belonging to ads and analytics before we serve them to the Android mobile browsers.

6 RESULTS

In this section, we present our results in three parts: data exploration, normality checks and transformations, and hypothesis testing.

6.1 Data Exploration

We tried to understand the nature of our data by descriptive statistics and boxplots. In the Table 5 , we show the descriptive statistics of all the dependent variable measures across all the 10 subjects and two treatments, with or without ads and analytics. From the Table 5, we can see the increase in the values of CPU usage, memory usage, and energy consumption when the web apps are running with ads and analytics. All standard deviation values are high, which indicate the data not being normally distributed.

The distribution of all the dependent variable data in the case of both the treatments, with or without ads and analytics, is presented by Figure 5. The data is positively skewed as the mean values in all the boxplots are higher than the median. The boxplots also have some outliers. These two reasons indicate that the data is most likely not normally distributed. The detailed study for normality check is done in Section 6.2.

6.2 Normality Checks and Transformations

We performed normality checks to find out which type of test, either parametric or non-parametric, is suitable for hypothesis testing. 3 types of data transformation–log, square root, and reciprocal–were used after the original data failed to show normal distribution. To enhance the efficiency of normality checks, we did normality checks by three methods.

Density plots. First, density plots were drawn from the obtained data. All the plots don't follow a bell curve. Due to this, we transformed all the data by applying all 3 transformation types. But all the density plots drawn from the transformed data didn't show the bell curve of a normal distribution data. For all normal and transformed data, 12 density plots were drawn; only 6 density plots, 3 for normal data and 3 for log-transformed, are displayed in Figure 6.

[17]https://developer.android.com/studio/command-line/dumpsys

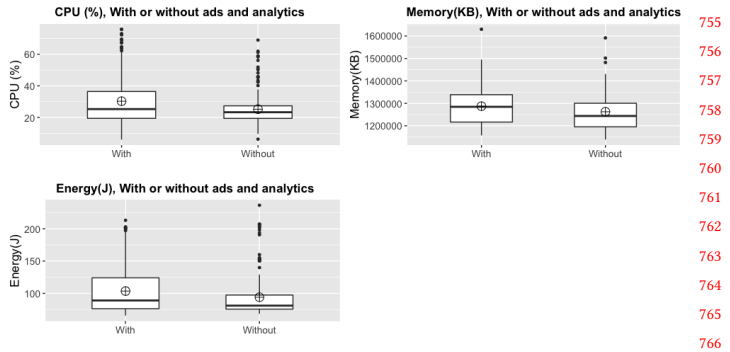

Figure 5: Comparison of all the dependent variable with or without ads and analytics

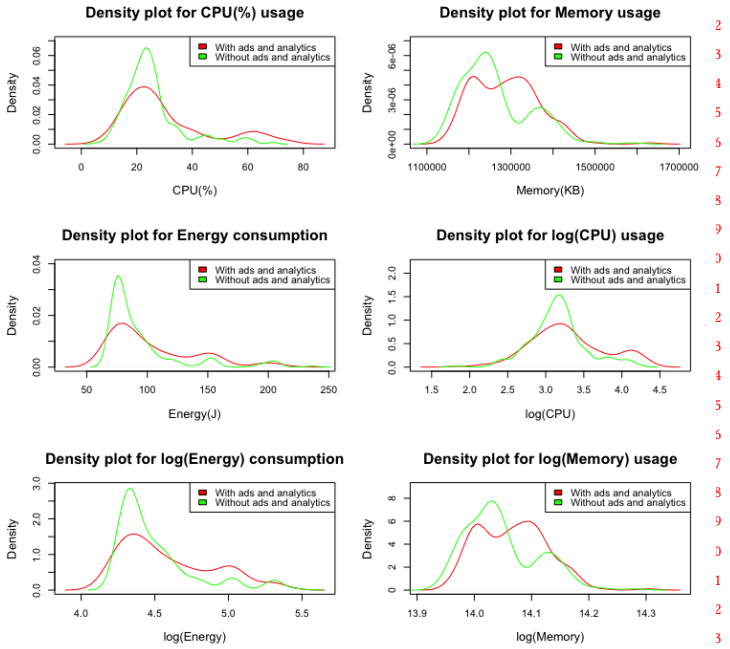

Figure 6: Density plots of normal and log-transformed data of all dependent variables with or without ads and analytics

The remaining diagrams and its data are available in the replication package mentioned in Section 4.6.

Q-Q plot: We drew Q–Q plots of both original and transformed data. In total, 24 Q-Q plots were drawn. All of them have outliers providing the indication of data not being normally distributed. Out of all the plots, only 6 are shown in Figure 7. The remaining diagrams along with the data are available in the replication package mentioned in Section 4.6.

Shapiro-Wilk test: Lastly, the Shapiro-Wilk test was performed in both original and transformed data. P-values of all the Shapiro-Wilk tests are less than $\alpha(0.005)$, which indicate the data not having normal distribution. The detailed results of all the Shapiro-Wilk tests are mentioned in Table 6.

Table 5: Descriptive statistics of CPU usage in percentage(%), memory usage in kilobyte(KB), and energy consumption in joule(J)

Treatments	With ads and analytics			Without ads and analytics		
Variables	**CPU**	**Memory(KB)**	**Energy(J)**	**CPU**	**Memory(KB)**	**Energy(J)**
Minimum	6.007	1157239	65.52	6.285	1138537	68.28
1st Quantile	19.454	1216628	76.19	19.425	1195440	75.55
Median	25.301	1284578	88.96	23.316	1243792	80.96
Mean	30.298	1287301	103.52	25.244	1262512	93.96
3rd Quantile	36.424	1338496	124.04	27.364	1300476	97.35
Maximum	75.820	1630092	213.14	68.964	1591495	236.49
Standard Deviation	16.21182	77255.56	35.82646	10.42061	80429.24	32.49945

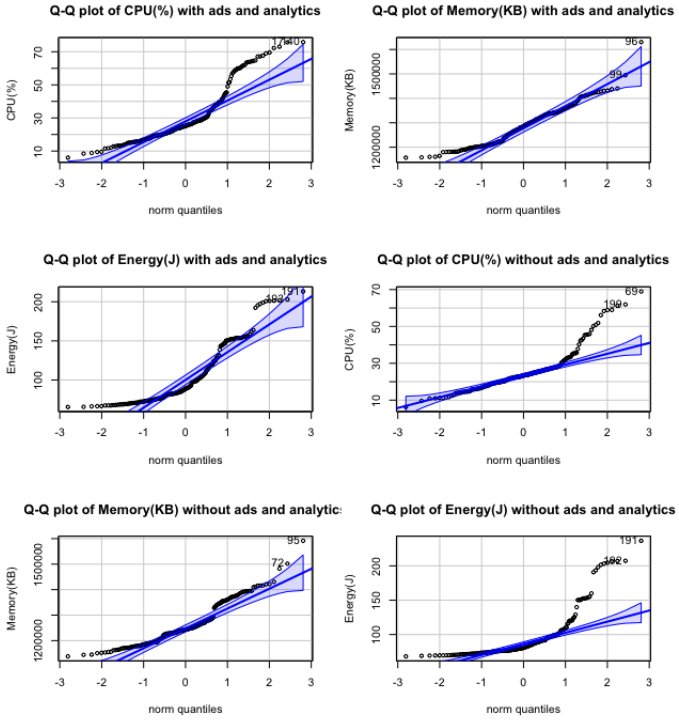

Figure 7: Q-Q plots of all dependent variables with or without ads and analytics

Conclusion: By considering the results of all the above three methods, we can conclude that the obtained data aren't normally distributed, even after applying the various data transformation techniques. As the data isn't normally distributed, we selected Wilcoxon Signed Rank for our hypothesis testing.

6.3 Hypotheses Testings

As it is witnessed in the previous Results' sub sections that the collected data are not normally distributed for all of the dependent variables i.e. *Energy Consumption (e)*, *CPU Usage (c)* and *Memory Usage (m)*. Hence, we tested our all hypotheses using **Wilcoxon Signed Rank** test to find out if there is any noticeable gap between the values of the dependent variables across both treatments (i.e. *with* and *without ads and analytics*). The effect size estimate for each rejected null hypotheses is also calculated using **Cliff's Delta** to analyze the magnitude of impacts. The summary of conducted tests is shown in Table 7.

Table 6: P-values of Shapiro-Wilk tests of each treatment

Treatments	With ads and analytics	Without ads and analytics
CPU	1.412e-12	2.636e-13
Memory	1.072e-05	5.151e-09
Energy	3.149e-13	< 2.2e-16
Log transformed CPU	0.0005406	0.0002012
Square root transformed CPU	2.777e-08	1.608e-08
Reciprocal transformed CPU	4.056e-11	1.275e-11
Log transformed Memory	0.0001095	6.867e-08
Square root transformed Memory	3.75e-05	1.918e-08
Reciprocal transformed Memory	0.0005199	7.625e-07
Log transformed Energy	7.096e-10	1.201e-15
Square root transformed Energy	1.675e-11	< 2.2e-16
Reciprocal transformed Energy	2.095e-07	7.84e-12

Table 7: P-values of Wilcoxon Signed Rank tests and Cliff's Delta effect size estimate

Value	Wilcoxon Signed Rank	Cliff's Delta
Energy Consumption	**0.0009602**	**-0.15505**
CPU Usage	**8.58e-05**	**-0.1455**
Memory Usage	**9.833e-07**	**-0.19745**

RQ1: *What is the overhead imposed by ads and user tracking/analytics on the energy consumption of mobile web apps?*

The test results for *Energy Consumption (e)* confirms the acceptance of alternate hypothesis while rejecting the $H_{0,br} : \mu_{ebr}$ (null hypothesis) with a *p-value* of 0.0009602 which is less than *α-value* (i.e. 0.05). Therefore, backing the claim that mobile web apps running without ads and analytics as compared to those with them do express lower values when it comes to energy consumption on the Android mobile. However to find out the magnitude of such an impact on the consumption of energy, the Cliff's delta effect size estimate[17] is measured as -0.15505, thus showing a *small* effect size ($0.147 \leq |d| \leq 0.330$)[17]. Based on the results obtained, we are able to establish the fact that mobile web apps without ads and analytics running on the browser in our Android mobile consumes less energy with a small effect size as compared to those in which ads and analytics are present.

RQ2: *What is the overhead imposed by ads and user tracking/analytics on the performance of mobile web apps?*

The performance is analyzed together on the basis of CPU and memory usage results in our research. Hypothesis test for both **RQ2.1** and **RQ2.2** are discussed in detail here to check the overhead imposed by ads and analytics on the overall run-time performance of the mobile web apps running on Android mobile.

RQ2.1: Memory *What is the overhead set by ads and analytics on the Memory usage of mobile web apps?*

Hypothesis test results for *Memory Usage (m)* dictates the rejection of null hypothesis $H_{0,br} : \mu_{mbr}$.The obtained *p-value* in this case is 9.833e-07 (which is < *α-value*), hence clearly showcasing the presence of greater memory consumption values in data for running mobile web apps with ads and analytics in contrast to those without ads and analytics. The *Cliff's delta value* obtained is -0.19745, meaning that a *small* effect size estimate is seen in this case[17] .

RQ2.2: CPU *What is the overhead set by ads and analytics on the CPU usage of mobile web apps?*

Here, the *p-value* obtained is 8.58e-05 that is significantly smaller as compared to *α-value*. By this, we can reject the null hypothesis thereby confirming the increase in the percentages of CPU usage when mobile web apps are running with ads and analytics. While checking further the effect size estimate of these findings, we have received a *delta value* of -0.1455 falling in the *negligible* effect size range i.e. ($|d| \leq 0.147$) that is a bit less promising as compared to Memory usage effect size[17].

Based on these findings for both **Memory** and **CPU** usage, we are in a confident position to conclude that running mobile web apps without ads and analytics has a small but positive impact on the overall run-time performance in our Android mobile.

7 DISCUSSION

For **RQ1**, we conclude that mobile web apps running without ads and analytics consume lesser amount of energy in the Android mobile. From a perspective of *web developer*, making mobile web apps without ads and analytics may reduce small amount of energy for a single *end-user* but can potentially reduce energy consumption if we scale it to a larger number of *end-users*; which is infact rapidly increasing at a global level with the easily accessible internet. These findings might be valuable for Web Software Industries, Governments and even common public as well specifically when everyone is aiming for more sustainable future.

For **RQ2**, we reach to a conclusion that mobile web apps running with ads and analytics can affect performance on Android mobile devices to a small but significant margin in terms of memory usage. Moreover, there is a negligible CPU usage difference while comparing web apps without ads and analytics.

Results obtained from our experiment are very different if we consider energy consumption and performance separately for each subject as shown in Figure 8. While analyzing the energy consumption, we found that "*imdb*" web app has surprisingly expressed more energy consumption while running without ads and analytics. This may be due to some possible data corruption. So further in-depth analysis is required to find out the actual reason behind it. Another reason can be the category of mobile web apps in which it falls i.e. "*imdb*" is mostly about video related content.

In case of memory usage for each mobile web apps, a data fluctuation against the normal expected behavior has been witnessed in four of the subjects namely "*imdb*", "*investing*", "*mirror-co-uk*" and a slightly changed value for "*science-org*". As subjects were running on two different browsers for conducting experiments to remove any bias, hence there can be a good possibility to explore how each browser handled set of operations differently for running these web apps. Such information is can be compelling for *browser vendors* as well so that the work could be done for better memory utilization for running mobile web apps with ads and analytics.

CPU usage has shown negligible differences for running mobile web apps with ads and analytics on Android mobile but unexpected values are noticed for some of the subjects i.e. "*cnblogs*", "*imdb*", "*investing*" and "*psychologytoday*". Overall, according to our results, the major part for performance cost is being played by the memory usage and not CPU.

Talking about all the subjects in general, we have seen a strange response values for all of the variables i.e. energy consumption, memory usage and CPU usage in some of the web apps, mainly in "*imdb*". This strange results can be a motivation for researcher to find the reasons behind it.

8 THREATS TO VALIDITY

For evaluating the soundness of our result, we have discussed below four different kinds of threats to validity.

8.1 Internal Validity

We have analyzed issues that arise from the design and execution of our experiment as threats to internal validity and discussed them thoroughly below.

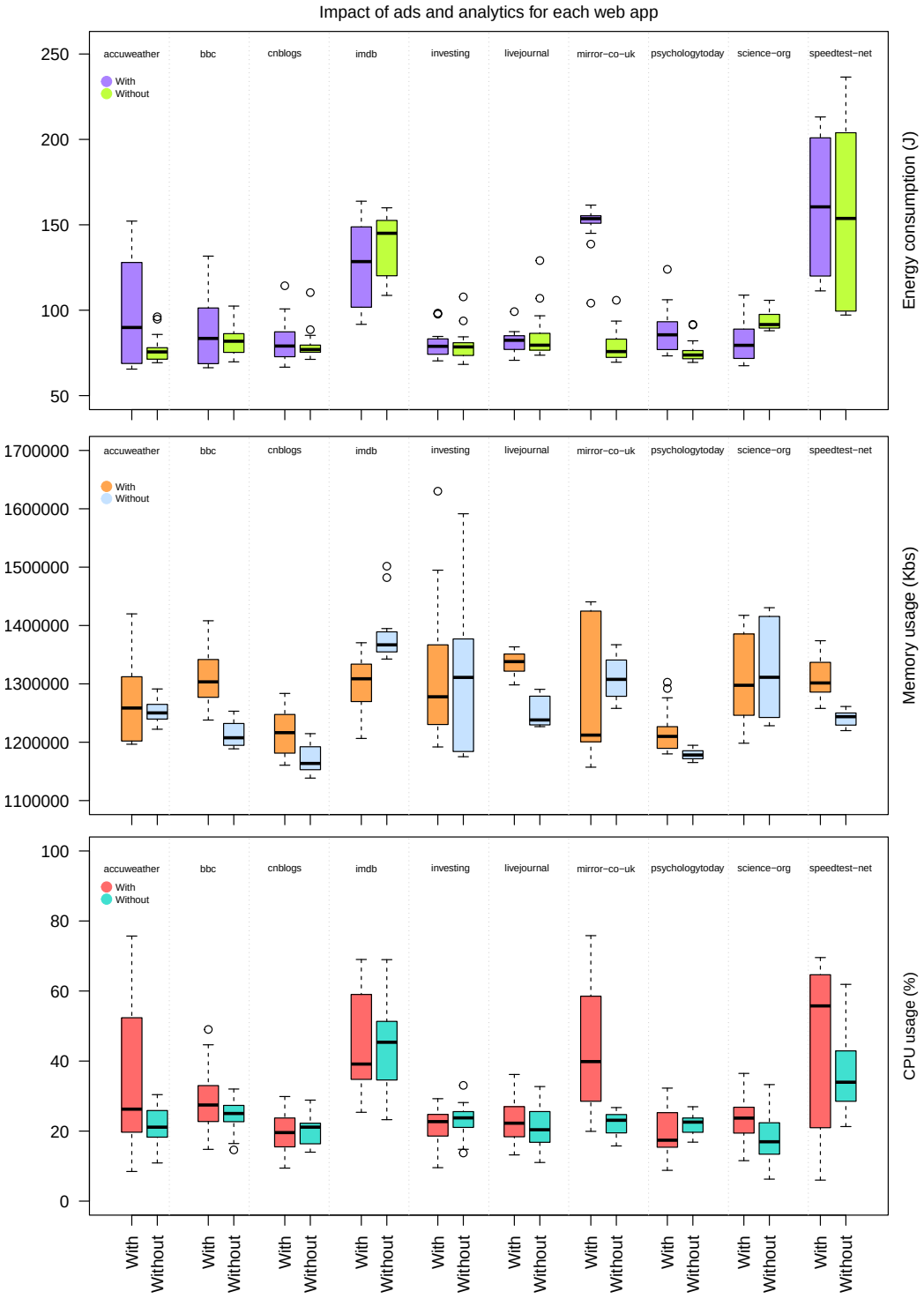

Figure 8: Comparison of all the dependent variable with or without ads and analytics

8.1.1 History: To execute the experiment and collect the data we have considered two options, one is to collect the results in an incremental and modular manner, or the second option is to execute all the repetitions of all subjects for both treatments and collect all the results in one shot. We chose the first option due to two main

reasons. The first reason is we are using a proxy server with a specific script that enables us to block all the resources related to ads and analytics from being downloaded to the Android mobile device, so it becomes practically impossible to run both treatments at the same time. Secondly, all the web apps have a different way of incorporating ads and analytics into their code, and this makes it difficult to write one script that works for all web apps without affecting the experiment results. We ended up conducting 10 repetitions for 10 subjects with both browsers for each treatment individually leading to 400 runs in total. Before executing the actual experiment, we installed the necessary certificate on the Android mobile device and enabled the "Third Party Certificate" option in the Firefox browser settings as Firefox is not a native app in Android OS, and conducted trial runs to ensure the testing environment was reliable and no interruption occurred during the execution phase.

8.1.2 Selection: We selected the final 10 web apps needed as subjects for our experiment through three iterations of selection. First, we randomly selected 50 web apps from the top 1000 web apps in the Tranco list[15], a ranking list that provides the top 1 million popular web apps. In the second iteration, we checked if the 50 subjects contain both ads and analytics with help of an online tool called *builtwith*, then in the third iteration, we manually debugged the source code of each web app and removed the web apps that do load their content by external JS file using *src attribute* in <*script*>tag. Selection criteria might affect this experiment since the selection process is done manually and does not represent the population. In order to mitigate this threat we have followed a thorough selection procedure during the web apps selection process (check section4.1) discarding web apps without ads and analytics, and using external JS files to load their contents, etc.

8.1.3 Reliability of measures: Factors such as screen brightness of the Android mobile device, network connectivity, charging of the device, and other interruptions are mitigated by setting the brightness of the screen to a minimum value, using data cable connection when possible and keeping the Android mobile as the only device connected to the *mitmproxy* proxy server's hotspot to ensure the network connectivity was constant, disabling charging during the experiment and conducting the experiment in a modular, incremental way keeping the environment the same for all the runs in the experiment.

8.2 External Validity

We have analyzed conditions that limit our ability to generalize the results we found in our experiment and to mitigate unrealistic environment settings and resources, we used a relatively modern mobile device and connected it via WiFi, the usual setting users set to browse web apps, and used both Firefox and Chrome browsers on the Android device to load the web apps in our experiment with 10B+ and 150m+ downloads respectively.

8.3 Construct Validity

8.3.1 Inadequate preoperational explication of constructs. By defining our constructs using the GQM method before starting out experiments, we mention to mitigate inadequate preoperational explication of constructs. The goal and question that our experiment is going to tackle as well as the metrics used to measure the goal and question, are derived using the GQM-tree.

8.3.2 Mono-operation bias. Our experiment is based on 3 independent variables. Therefore, our experiment isn't facing the mono-operation bias. Also, each different web app is trailed multiple times in different browsers, reaching a total of 400 trails. In Section 4.4, this is mentioned. This also will help reduce the mono-operation bias in our experiment.

8.4 Conclusion Validity

8.4.1 Low statistical power. As mentioned in Section 4.4, we have a sample size of 400 trails in total, including both with and without ads and analytics. Because of this high number of trail, the statistical power of our experiment increases. Also as shown in our results, significant differences are perceived.

8.4.2 Violated assumptions of statistical tests. To avoid this threat, we have done normality checks on obtained data in three phases– density plots, Q-Q plots, and Shapiro-Wilk test. All these checks gave the results which indicate that the obtained data isn't normally distributed. Furthermore, we did data transformation by three methods: log, square root, and reciprocal. All of these methods gave results which indicate the data being not normally distributed. By considering all of these results, we chose Wilcoxon Signed Rank test for hypothesis testing.

8.4.3 Fishing and error rate. As per the diagrams mentioned in Section 6, we have outliers and few data which shows the values of dependent variables being higher in case of couple of web apps without ads and analytics. Due to this, this threat doesn't apply in our case.

9 CONCLUSIONS

In this paper we presented the design, execution and results of an empirical study assessing the impact of ads and analytics on the energy consumption and performance in the context of real-world web apps. Our results show that there are evidences that web apps without ads and analytics consume significantly lower energy and memory on the Android mobile devices. In case of CPU usage, the difference is negligible. The study provides empirical evidence about the cost of ads and analytics on mobile web apps.

Few possible extensions of this experiment are to investigate (i) the cost on different type of ads. The goal here is to better understand whether or not a certain type of ads consume more energy than others. This can offer an insight for web app owners to choose the suitable type of ads (ii) whether or not the amount of interaction of web app visitors increase the energy consumption and degrade performance of the Android mobile device due the analytics record more activities. (iii) whether or not different Android web browsers with the same web app with ads and analytics has significant difference in energy consumption and performance, the results can provide an insight on which Android browser are better at handling ads and analytics.

The followings are the links for timelog, overleaf link and github link respectively:

Timelog link: https://docs.google.com/spreadsheets/d/1piCNknr4FhWgq68m3GmoFGzK-srZhE0DF5fsNGoGBAM/edit?usp=sharing

overleaf link: https://www.overleaf.com/3191114774gdcmrccjvbss

Github link: https://github.com/michiboo/ads-analytics-rep-pkg

A APPENDIX

REFERENCES

[1] L. Zhang, D. Gupta, and P. Mohapatra, "How expensive are free smartphone apps?" *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 16, no. 3, pp. 21–32, 2012.

[2] R. Simons and A. Pras, "The hidden energy cost of web advertising," in *Proceedings of the 12th Twente Student Conference on Information Technology*. Citeseer, 2010, pp. 1–8.

[3] C. Gao, J. Zeng, F. Sarro, M. R. Lyu, and I. King, "Exploring the effects of ad schemes on the performance cost of mobile phones," in *Proceedings of the 1st international workshop on advances in mobile app analysis*, 2018, pp. 13–18.

[4] "See a site by turning off chrome's ad blocker - android." [Online]. Available: https://support.google.com/chrome/answer/7632919?hl=en&co=GENIE.Platform%3DAndroid&oco=0

[5] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering - An Introduction*. Kluwer Academic Publishers, 2012.

[6] J. Gui, S. Mcilroy, M. Nagappan, and W. G. Halfond, "Truth in advertising: The hidden cost of mobile ads for software developers," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1. IEEE, 2015, pp. 100–110.

[7] J. Gui, D. Li, M. Wan, and W. G. Halfond, "Lightweight measurement and estimation of mobile ad energy consumption," in *2016 IEEE/ACM 5th International Workshop on Green and Sustainable Software (GREENS)*. IEEE, 2016, pp. 1–7.

[8] C. Gao, J. Zeng, F. Sarro, D. Lo, I. King, and M. R. Lyu, "Do users care about ad's performance costs? exploring the effects of the performance costs of in-app ads on user experience," *Information and Software Technology*, vol. 132, p. 106471, 2021.

[9] 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*, 2014, pp. 90–97.

[10] B. Pourghassemi, J. Bonecutter, Z. Li, and A. Chandramowlishwaran, "Adperf: Characterizing the performance of third-party ads," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 5, no. 1, feb 2021. [Online]. Available: https://doi-org.ezproxy.cc.lut.fi/10.1145/3447381

[11] V. R. B. G. Caldiera and H. D. Rombach, "The goal question metric approach," *Encyclopedia of software engineering*, pp. 528–532, 1994.

[12] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *2010 USENIX Annual Technical Conference (USENIX ATC 10)*, 2010.

[13] P. A. Salz, "Monitoring mobile app performance," *Journal of Direct, Data and Digital Marketing Practice*, vol. 15, no. 3, pp. 219–221, 2014.

[14] G. Keizer, "Windows comes up third in os clash two years early," *Online at http://www. computerworld. com/article/3050931/microsoft-windows/windowscomes-up-third-in-os-clash-two-years-early. html*, 2017.

[15] [Online]. Available: https://tranco-list.eu/

[16] I. Malavolta, E. M. Grua, C.-Y. Lam, R. de Vries, F. Tan, E. Zielinski, M. Peters, and L. Kaandorp, "A Framework for the Automatic Execution of Measurement-based Experiments on Android Devices," in *35th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW '20)*. ACM, 2020. [Online]. Available: https://github.com/S2-group/android-runner/blob/master/documentation/A_Mobile_2020.pdf

[17] N. Cliff, "Dominance statistics: Ordinal analyses to answer ordinal questions." *Psychological bulletin*, vol. 114, no. 3, p. 494, 1993.