ABSTRACT

Context. The Web Push API is a popular technology Web developers use to send push notifications to users’ mobile Web apps. Until now, the energy consumption of this technology has not been thoroughly evaluated.

Goal. This research aims to empirically assess the impact of using the Web Push API on the energy consumption of mobile Web apps.

Method. We conduct an empirical experiment in a controlled environment where several push notifications are sent to Telegram running on an Android device, while measuring its energy consumption. Telegram is executed on two different browsers (i.e., Google Chrome and Mozilla Firefox), and push notifications are sent with different frequencies and distributions. The results are analyzed statistically, followed by an effect size estimation.

Results. Telegram tends to consume more energy when receiving push notifications on both browsers. However, it tends to consume generally less energy when loaded on Firefox. The frequency of received push notifications leads to higher energy consumption on Chrome, but we obtained mixed results on Firefox. Push notifications received in bursts lead to lower energy consumption than push notifications received evenly over time.

Conclusions. This study provides evidence about the impact of Web push notifications on the energy consumption of Telegram. Our results provide an initial foundation for Web developers to make better-informed decisions about the frequency and distribution of push notifications in their Web apps.

ACM Reference Format:

1 INTRODUCTION

Since its introduction, push notifications have become a ubiquitous feature in mobile applications, regularly appearing on the interfaces of smartphones [1]. Push notifications, which have evolved to encompass not only plain text but also hyperlinks and images, are delivered (“pushed”) from the Cloud. Before the ascent of mobile Web apps, conventional technologies like long polling and WebSocket served as prevalent solutions to proactively send information to the clients [2]. However, polling and WebSocket are mainly useful in scenarios requiring persistent connections, such as instant messaging. These traditional push technologies like WebSocket have a limitation in that they demand a continuous full-duplex TCP connection throughout the session, while long polling mandates the client’s immediate transmission of a polling request to the server upon receipt of a response, persisting in this cyclical pattern [3]. In mobile Web applications, these push technologies cease to function when the Web page loses focus, or the page is closed [4]. The remedy to enable Web applications to deliver push notifications akin to their native mobile counterparts lives in the realm of Progressive Web Applications (PWAs) and their associated Web Push APIs [5].

PWAs blend the useful attributes of both native applications and Web applications [6]. PWAs represent Web applications that afford users a native-like experience replete with performance enhancements, independent of the device or browser employed. An essential component integral to endowing PWAs with native-like functionality is the service worker which runs in the background. It enables intercepting network requests, caching resources, and handling push events through a bunch of APIs. Among the array of native app-like features that PWAs offer, push notifications stand out as a fundamental element, significantly enhancing user engagement. The utilization of push notifications entails the orchestration of both the Push API and the Notification API which work together, allowing the creation and management of system notifications that persist even after a user has navigated away from the originating site. The Push API also can dispatch messages to applications even when the user is temporarily offline with the help of the push service which queues the messages until the user is online again.

Conversely, the Push API facilitates the management of push messages sent asynchronously from the server to a Web application via an intermediary push service. The Push API’s mechanism entails a complex interplay of five entities: the Web page, the service worker, the user agent, the push service, and the application server [5]. Figure 1 shows the sequence of events, including registration, subscription, push message delivery, and subscription cancellation, involving the above entities. The process begins with the Web page registering a new service worker with the user agent and subscribing the user agent to the push service with the information of worker registration. The resultant push subscription yields a unique push endpoint which is used by the application to subscribe to the server. When the server dispatches a push message, the message traverses the push service and then it gets routed to the active service worker associated with the pertinent push subscription. For the instances when the worker is inactive, the Push API wakes it up to facilitate
One of the key elements that enhance the user experience provided by PWAs is push notifications, which can be used to send instant messages to users. The impact of push notifications on energy consumption is a critical aspect to consider. The main contributions of this study are: (i) an empirical assessment of the impact of push notifications on energy consumption in PWAs across different browsers and with different sending patterns, (ii) a discussion of the obtained results, and (iii) the full replication package of the study containing raw data, source code, and scripts for data analysis.

Our intended audience includes both Web developers and end users. Based on our experience with Telegram, this study offers actionable insights to Web developers for optimizing the strategies that they can apply when managing push notifications in their Web apps. About end users, we provide empirical evidence on the potential adverse effects of activating push notifications on the energy consumption of their Android devices.

2 RELATED WORK

Rammos et al. conducted an empirical experiment to investigate how the frequency and distribution of instant received messages impact the energy consumption of Android devices [11]. The findings revealed that energy consumption increases with the amount of messages. It was also noted that there might be a threshold; beyond this point, as the number of messages increases, the rise in energy consumption becomes gradual. Besides, no evidence emerged to suggest that message distribution has a significant impact on energy consumption. Building on this, we raise two of our research questions and experimented on the impact of energy consumption on message-pushing frequency and distribution. In terms of frequency, the experiment focuses on sending varying amounts of notifications at specific times. The distribution aspect centers on whether the notifications are sent in bursts or evenly spread and pushed to the app.

Malavolta et al. assessed the impact of service workers on the energy efficiency of PWAs [9]. The experiment tested seven PWAs under different conditions. Trepn energy profiler was employed to measure battery usage as the metric for energy efficiency. The results suggest that the usage of service workers does not significantly impact energy consumption. Service workers manage a few APIs, including caching and preloading data, and push notifications. We aim to assess the impact of notifications, providing valuable insights for future developers.

Burgstahler et al. assessed the difference in energy consumption between push- (with Google cloud messaging) and pull-based (with Apache Tomcat) notification services with different payload sizes and notification intervals via Wi-Fi using a Samsung Galaxy S3 (running Android 4.1.2) [7] with an experimental notification app written by them, and concluded that the energy demand is independent of payload size but strongly depends on application scenarios, e.g., amount of messages, temporal distance, and client behavior (determine if needed for information updates). By contrast, our work also studies the energy consumption of push notifications, but based on a different push technology and experimental subjects. Our notifications are managed by the service worker, and we will discuss the impact of message frequency and the difference between even and burst message distribution in terms of energy consumption in terms of energy consumption. Also, we concentrate on how the behaviors of Web push notifications impact the energy consumption of mobile devices, which is the receiving side instead of the sending side.

3 EXPERIMENT DEFINITION

PWA represents an ascending approach to building Web applications that offer a more native app-like experience to users. It allows users to enjoy features of native applications (e.g., cache, push notifications, offline access, etc.) without downloading the application. One of the key elements that enhance the user experience provided...
by PWAs is the Web Push API. This API allows Web applications to send real-time notifications to users’ devices, even when the browser is not activated.

Given the increasing popularity of PWAs and the important role of Web Push API in enhancing user engagement, it becomes crucial to evaluate how the Push API impacts the energy consumption of the end-user devices. For mobile users, the experiment offers them an acquaintance about the energy consumption which could directly affect their device’s battery life [12]. For the developers, the experiment referring to energy consumption could provide valuable information and help them optimize their applications more efficiently and sustainably [13].

Hence the goal of this experiment is to analyze Web push API as a push technology to evaluate with respect to the impact on power consumption and energy efficiency from the point of view of software developers and mobile users in the context of the Telegram mobile Web app.

Based on the goal of our experiment, we define 3 research questions referring to the particular impact of the Web push API on energy consumption, as well as two factors that might influence energy efficiency when enabling the Web push API with different push patterns. The research questions of this study are the following:

**RQ1**: How does the enabling of Push API affect the energy consumption of the Telegram mobile Web app?

- **RQ1-1**: What is the impact of using different browsers on the energy consumption of the Telegram mobile Web app?

**RQ2**: What is the impact in terms of energy consumption on receiving notifications with different sending patterns?

- **RQ2-1**: What is the impact in terms of energy consumption on receiving notifications with different sending frequencies?
- **RQ2-2**: What is the impact in terms of energy consumption on receiving notifications with different sending distribution patterns?

**RQ1-1** aims to assess the influence of enabling or disabling the push API on the energy consumption of the Telegram mobile Web app. By quantifying the energy impact, we can determine whether the utilization of the Push API leads to increased energy consumption. PWAs often rely on push notifications to engage users by providing real-time updates, so it is important to recognize this effect.

**RQ2-1** focuses on the frequency of push notifications and their effect on energy efficiency. We will vary the frequency of push notifications, simulating scenarios with high and low notification rates. For instance, we have one device receiving 10 notifications in 10 minutes, and another receiving 1 notification in the same period. We will be sending the notifications in one message per minute, and for comparison, all 10 messages are sent together within a very short time frame. Understanding the energy implications of different notification rates can help developers find a balance between user engagement and energy conservation.

**RQ2-2** explores various distribution patterns, such as centralized versus decentralized, and analyze how these patterns affect energy consumption. By comparing the results, we can find the impact of distribution strategies on energy efficiency. Costs for energy might vary from distribution strategies, and developers are supposed to know which approach is more energy-efficient. Furthermore, this question reflects the requirement to consider the concern for energy efficiency and notification delivery.

4 EXPERIMENT PLANNING

4.1 Context and Subject Selection

We consider the Android system to conduct our experiment. The Android system holds a higher market share compared to iOS. Also, compared with the closed source iOS system, Android has more open source tools to experiment. We opted to conduct our experiment on the Pixel 5 running the Android 13 system.

To select possible subjects for our experiment, we first referred to the Tranco list [14]. We initially aimed to select a large set of usable Web apps from the Tranco list; however, it turned out that most of the Web apps we sampled were not PWAs (and thus they were not capable of receiving push notifications). So, we turned our attention to a well-known repository of PWAs. However, for our experiment, we also require the ability to control when to receive a push notification. This led us to choose message-oriented applications. After receiving messages, the app can push notifications to the users. By controlling sending messages, we can control the notifications. Besides, data shows that messages create most notifications [15]. Based on the above, we selected Telegram, Facebook, WhatsApp, LinkedIn, and Twitter Lite. During the test, we found that WhatsApp and LinkedIn do not provide a fully functional application in the Android system. They do not provide the ability to push notifications after receiving a message. Facebook imposed restrictions on sending messages - too many messages sent at once lead to an account being blocked. Facebook requires the use of Messenger to send messages, but Messenger does not offer a PWA option on Android. Twitter Lite and Telegram both provide the API for developers to control the message sending. Considering the execution time of the experiment, we selected Telegram as our experiment application. Apart from the functionalities mentioned earlier, it also offers a message-sending API and a Python library that allows us to automate our experiments.

4.2 Experimental Variables

The independent variables of this study are the following:

1. Push API Activation Status: For RQ-1, the activation status of the push API serves as a significant independent variable. It is categorized as either activated or deactivated, allowing us to investigate how the presence or absence of push notifications affects energy consumption.

2. For RQ1-1, we consider three types of browsers: Chrome, Firefox, and Opera. During testing, we discovered that the PWA of Telegram installed from Opera is not able to receive notifications (the problem most likely lies on the Telegram side since Web Push APIs are fully supported by Opera). As a result, we opted to focus our experiments on Chrome and Firefox.


3. Frequency of Messages: For RQ2-1, we manipulated the frequency of incoming messages as an independent variable. This variable includes three levels:

- Idle: No incoming messages.
- Low: Incoming messages at a rate of 5 messages per minute.
- High: Incoming messages at a rate of 30 messages per minute.

4. Distribution Pattern of Messages: For RQ2-2, the distribution pattern of incoming messages was explored as an independent variable. We will consider two distribution patterns:

- Burst: Messages are concentrated within short time intervals, with the same amount of messages incoming within a short period of 0.2 seconds.
- Even: Messages are evenly distributed over time, with incoming messages occurring over a long period of 10 seconds in contrast to burst time intervals.

After giving these variables, we should pay attention to the combinations of browsers and applications. To comprehensively assess the effects of the above variables, we conduct experiments with multiple combinations of browsers and applications. These combinations include the repetition of the same patterns and the number of apps. For browsers, we consider Chrome and Firefox, on which we will perform repetitions for Telegram.

Energy consumption is the dependent variable of this study. It measures the amount of energy consumed by the Telegram Web app while considering the various levels of push API activation and message frequency.

4.3 Experimental Hypotheses

In this experiment, we aim to investigate whether the specified independent variables (i.e., browser, sending frequency, sending distribution pattern) influence the dependent variable (i.e., energy consumption). To establish a sound analytical procedure, we formulate a set of hypotheses for each RQ of the experiment.

RQ1. In the hypotheses for RQ1, the symbol $e$ denotes the activation status of the push API, whereas $d$ represents the dependent variable, specifically energy consumption. $H_{e}^{d}$ shows the influence of enabling the push API ($e$) on the energy consumption ($d$). Additionally, $\mu$ represents the mean of the energy consumption $d$ with enabled Web push APIs $e$. Specifically, $\mu_{on}^{d}$ represents the population mean of energy consumption when push API is on while $\mu_{off}^{d}$ represents the mean of the energy consumption when push API is off.

$$H_{0}^{e,d}: \mu_{off}^{d} = \mu_{on}^{d}$$
$$H_{a}^{e,d}: \mu_{off}^{d} < \mu_{on}^{d}$$

RQ1-1. In the hypotheses of RQ1-1, we investigate the potential influence of different browsers, which could lead to different energy consumption. The symbol $brw$ represents the different browsers that were used to experiment. $d$ shows dependent variables. So $H_{brw}^{d}$ explores the impact different Web browsers (i.e., Chrome and Firefox) may bring to energy consumption. We use $\mu_{Chrome}^{d}$ to represent the mean of the energy consumption when conducting experiments with the Chrome browser, and similarly, we use $\mu_{Firefox}^{d}$ to represent the mean of the energy consumption when using the Firefox browser.

$$H_{0}^{brw,d}: \mu_{Chrome}^{d} = \mu_{Firefox}^{d}$$
$$H_{a}^{brw,d}: \mu_{Chrome}^{d} \neq \mu_{Firefox}^{d}$$

RQ2-1. In the hypotheses for RQ2-1, $f$ represents the frequency of sending messages, which refers to the amounts of messages sent to mobile devices within a specific period. $d$ corresponds to the previously mentioned energy consumption. Hence, $H_{f}^{d}$ is raised here to investigate whether the message-sending frequency will impact the energy consumption. $\mu_{idle}^{d}$, $\mu_{low}^{d}$, and $\mu_{high}^{d}$ calculate the mean of the energy consumption caused by enabling notification with the message sending frequency in idle, low, and high respectively.

$$H_{0}^{f,d}: \mu_{idle}^{d} = \mu_{low}^{d} = \mu_{high}^{d}$$
$$H_{a}^{f,d}: \exists (f_1, f_2)|\mu_{f_1}^{d} \neq \mu_{f_2}^{d}$$
$$\forall f_1, f_2 \in \{Idle, Low, High\}$$

RQ2-2. This RQ aims to assess whether the distribution pattern (i.e., even or burst) of sending messages will impact the energy consumption of mobile. So $dp$ represents the distribution pattern while $d$ signifies the dependent variable, which is the energy consumption in our experiment. So $H_{dp}^{d}$ is to examine the influence of distribution pattern on energy consumption. $\mu_{even}^{d}$ and $\mu_{burst}^{d}$ represent the means of the energy that the mobile device has consumed while enabling notifications with sending messages’ distribution pattern in an even manner and a burst manner, respectively.

$$H_{0}^{dp,d}: \mu_{even}^{d} = \mu_{burst}^{d}$$
$$H_{a}^{dp,d}: \mu_{even}^{d} \neq \mu_{burst}^{d}$$

4.4 Experiment Design

As described in Section 4.2, our experiments have one subject and four factors:

- Notification status: enabled, disabled;
- Browser platform: Chrome, Firefox;
- Notification frequency: idle, low, high;
- Notification distribution: even, burst.

We aimed to assign all combinations of treatments to each factor. When the notification frequency is set to idle, the trials involving the combination of all distribution treatments and idle yield the same results. There are 20 trials in our experiment in total, including 3 different frequencies and 2 different distributions for each notification status under two different browsers. Each trial required 2 minutes to complete, and we repeated each trial 30 times to reduce the deviation. Between each two trials, there will be a 2-second gap. We set this gap to mitigate potential influences because of differences in the startup period of each trial, such as the initialization of the Python script. This process takes approximately 20 × 122 seconds × 30 = 20 hours and 20 minutes to execute.

Each trial will run for two minutes. To guarantee that the application can receive all messages and subsequently push notifications, we implemented a 10-second buffer at the start and end of each trial.
4.5 Data Analysis
Our data analysis process can be divided into four steps: data exploration, normality check, hypothesis testing, and effect size estimation.

The initial phase is data exploration. We calculate descriptive statistics and employ box plots to provide a visual representation. This helps in establishing a basic understanding of the collected energy consumption data.

Following an initial analysis of the data, we aim to dive into the specifics of our research questions and hypotheses. Before conducting hypothesis tests, our initial step will involve checking the normality of the data. This will enable us to determine the most appropriate testing methods for our research. During the phase of Normality check, we shall employ the Shapiro-Wilk test and density plots and Q-Q plots to check data normality. We check the p-value from Shapiro-Wilk whether it is less than 0.05. Besides, the density plots give an initial indication of the data distribution of the energy consumption. Q-Q plots also serve as a metric for normality checks. A straight line in the Q-Q plot diagram signifies the sample follows a normal distribution. The slope of this line corresponds to the standard deviation, while the intercept represents the mean. If the plot deviates from a straight line, it suggests that the distribution may not be normal. By these three means, we can determine whether the data is normally distributed accurately.

The most important phase is Hypothesis testing. For RQ-1, we plan to use a t-test to check whether the notification status impacts energy consumption if the data is normally distributed. In this case, we need to check the normality of the outcome energy consumption with notification on and notification off. If data is not normal, we shall use the Wilcoxon rank sum test, which is a non-parametric alternative of the t-test for two unpaired data groups of notification on and notification off.

We also have a sub-question RQ1-1 about the browser’s impact. We shall conduct two tests for these two different browsers, respectively. RQ1-1 aims to investigate the influence of different browsers on energy consumption when notifications are enabled. To mitigate the variance in baseline energy consumption with notifications turned off because of different browsers, our analysis will focus on the difference in energy consumption between notification on and notification off which is brought by enabling the notification. This comparison will be facilitated through subtraction. The testing strategy for this hypothesis mirrors that of RQ-1, employing a t-test for normally distributed data and resorting to the Wilcoxon rank sum test for non-normally distributed data.

For RQ-2, we shall focus on the two independent factors (i.e., sending frequency and sending distribution pattern) that could impact the final energy consumption. Additionally, we found that Two-Way ANOVA[19] is a statistical method that could be used to analyze the influence of two categorical independent variables on a continuous dependent variable. It allows us to check the main effects of each factor and their interaction effects on the dependent variable, namely the impact of different sending patterns on energy consumption. Hence, we plan to employ the Two-Way ANOVA statistical test in this context. Our specific interest lies in the impact of different sending patterns on energy consumption caused by enabling the notification. Consequently, we will conduct tests for these different sending patterns under the conditions of two distinct Web browsers to eliminate the influence of different browsers, and the energy consumption will be the difference between notification on and notification off under the same sending pattern. If the assumptions of the Two-Way ANOVA test are not satisfied, then we will apply the Wilcoxon rank sum test for these two factors, respectively. All statistical tests are performed using a 95% confidence interval, thus $\alpha = 0.05$.

The final phase is Effect size estimation. After we complete the hypothesis test, we use the Cliff’s Delta effect size statistic[20] to estimate the magnitude of each factor treatment effect on the dependent variable, which in this case is energy consumption. Cliff’s Delta is a non-parametric statistic that is applicable even when our data does not follow a normal distribution. Based on the values of each Cliff’s Delta statistic and the rating scale[20], we provide an intuitive and comprehensive interpretation of the degree of influence that each factor has on energy consumption. Specifically, all the energy consumption mentioned here aligns with the energy consumption used to conduct hypothesis tests as specified above.

5 EXPERIMENT EXECUTION
5.1 Preparation
Several pilot experiments were conducted to evaluate the viability of the experiment’s execution. These included integration tests undertaken to verify the seamless interconnection of all devices, the proper functionality of the message-sending script, and the accurate collection of data regarding energy consumption. Additionally, a series of demo trials, involving 5 repetitions for each trial, was executed to examine the reception and display of push notifications while running the Web application on various browsers, considering potential interferences. As illustrated in Figure 2, the experimental infrastructure is composed of three primary components:

- An Android mobile device serves as the platform for executing the experimental subjects and the measurement tool to collect energy consumption data.
- A local Raspberry Pi 3 Model B (with 1 GB RAM) assumes a dual role in the experiment: it sends messages to trigger push notifications on the mobile device and operates an orchestration tool for executing measurement. For convenience reasons, we use a laptop to connect to the RPIs serving as the display and input device via SSH.
- Cloud-based hosting that contains messaging services and Web Push APIs, as well as push services.

For our experimentation, we employed a Google Pixel 5 device running Android version 13. This device is equipped with 8GB of memory and features a battery with a capacity of 4080mAh. To prepare for the experiments, we set up the BatteryManager profiler on the Raspberry Pi and installed the Google Chrome (version: 117.0.5938.60) and Mozilla Firefox (version: 118.0) browsers on the Android device. Additionally, we separately added the Telegram Web app powered by a different Web browser to the mobile’s home screen. Push notification enabling status is controlled by manually enabling or disabling both the application notification settings and the browser notification settings.

We prepared a Python script[10] that used the Telethon library, which is a third-party Python library that enables easy interaction
with Telegram through Telegram APIs to facilitate user authentication and message sending. The Python script reads the configuration from the `config.json` file to get the information needed for user authentication (i.e., sender’s identification), determine the message destination (i.e., receiver’s username), and set message-transmission patterns. Within the script’s configuration file, parameters such as the number of messages and their transmission intervals are adjustable to align with experiments. In the formal experiment, each trial involved the modification of settings, wherein the number of messages and burst mode values were systematically altered (e.g., `nr_of_message: 50, burst: false`). The subsequent table (Table 1) presents the configuration of parameters applied throughout the entire experiment, with time values provided in seconds.

### Table 1: Example of fixed settings of the Python script

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>sending_window</td>
<td>100</td>
</tr>
<tr>
<td>setup_time</td>
<td>10</td>
</tr>
<tr>
<td>burst_interval</td>
<td>0.2</td>
</tr>
<tr>
<td>receiving_buffer</td>
<td>10</td>
</tr>
</tbody>
</table>

For the orchestration of the execution runs, Android Runner [21], a specialized tool, is employed as the central controlling mechanism. Android Runner which contains Android SDK and Android Debug Bridge (ADB) supports several common profilers that can measure performance and energy for Android devices. In our experiment, the BatteryManager profile from the Android Runner kit is executed on an RPi running Debian GNU/Linux 11.

### 5.2 Setup

The Python script and Android Runner are capable of independent execution on distinct computer devices. However, to minimize potential biases arising from variations in hardware and Internet conditions, our experimental protocol stipulates that all experiments be conducted on a single Raspberry Pi. Moreover, all experimental runs are consistently executed within the same Wi-Fi environment. Before executing the experiment, we established the hardware connections as follows:

1. All essential hardware components, including the laptop, Raspberry Pi, and mobile device, were seamlessly integrated into a shared Wi-Fi environment.

(2) A connection between the Raspberry Pi and the laptop via SSH and Wi-Fi

(3) Energy provisioning for the mobile device was ensured through a direct USB connection to the Raspberry Pi, thus guaranteeing an uninterrupted power supply throughout the experiment.

(4) The Android Debug Bridge (ADB) that runs on the Raspberry Pi, remotely connects to the mobile device via Wi-Fi.

Preceding the initiation of each experiment trial, here is the standardized procedure to follow: (i) All applications on the mobile device are terminated, leaving only the Telegram Web application, which is powered by the specific target browser, actively logged in; (ii) the notification settings on both the mobile application and the browser are precisely configured to match the predetermined target status; (iii) the Python script’s parameters are adjusted to align with the specific trial settings. Once the setup is completed, the experiment trial is formally initiated by executing the BatteryManager profiler via the command-line interface, and the Python script is triggered by the settings in the BatteryManager’s configuration to run parallel with the profiler.

Throughout the experiment, the mobile screen remains locked and deactivated, presenting a black screen. The notification settings on the mobile device were configured to minimize energy consumption from sources other than the browser’s push notification service. Specifically, the notification mode was set to “banner on,” with both the auditory tone and screen illumination deactivated, and vibration functionality turned off to the greatest extent possible. While this configuration may diverge from typical real-world user scenarios, its purpose was to mitigate the impact of external energy consumption factors unrelated to the push service. This approach aimed to standardize conditions and reduce variations arising from differences in energy consumption among hardware components, such as audio components and screens, across diverse mobile devices.

For energy we use `BatteryManager`[22], an Android interface that provides information about the system’s battery charge level and can be used to query battery and charging properties on Android mobile devices. To eliminate the influence of USB charging, before each run of the experiment, the USB charging is automatically disabled and it is enabled at the end of the run.

### 6 RESULTS

#### 6.1 Data exploration

Tables 3 and 4 mentioned in the replication package, display the basic information for Chrome and Firefox with different notification setups. We used this data to create Figure 3, a visual representation. Observing Figure 3, it’s evident that when notifications are enabled, energy consumption is higher compared to when they are disabled, for the same browser. This shows that notifications affect energy usage. From the statistics in Tables 3 and 4, we can find that the experiment data is not symmetric.

In Figure 3, it is evident that when notifications are enabled, but none are pushed to the device, there’s a difference in basic energy consumption. Specifically, Chrome consumes more energy compared to Firefox. To analyze the research question 1-1, the impact of different browsers, we need to calculate the contrast...
between the idle and active states for each browser. This statistical analysis will be addressed in Section 6.3. From the box plot of Figure 3, we can find that in these two different browsers, the energy consumption box plots with notification status “on” shifted upward than notification “off”. Moreover, this shift is more pronounced in Chrome compared to Firefox, indicating that the impact of having notifications enabled is more substantial in Chrome.

Figures 4 and 5 show the energy consumption of Chrome and Firefox across various notification patterns. Examining the impact of notification frequency in Chrome (see Figure 4), we observe that higher frequencies lead to increased energy consumption with the same notification distribution. For Firefox (see Figure 5), a different trend emerges: higher-frequency notifications consume less energy than lower-frequency ones. These findings also align with the hypothesis test results described in Section 6.4. A more in-depth exploration of the reasons behind this interesting phenomenon will be provided in the discussion section. Furthermore, this discovery serves as a potential indicator of the different behaviors of different browsers for what concerns energy consumption.

When looking at the distribution of received notifications, both Chrome and Firefox tend to consume less energy when receiving notifications in bursts. Firefox shows a more pronounced effect compared to Chrome.

Finally, as a part of the normality tests we observed that the overall data is not normally distributed.

6.2 The impact of notification status on energy consumption (RQ1)

To eliminate the impact of browsers on the final report, we will perform hypothesis tests for two different browsers. Specifically, for each of these browsers, we will investigate whether enabling notifications affects energy consumption.

Hypothesis testing. In this case, we only have one factor (i.e. notification on and notification off) with a numerical outcome of energy consumption. As we have conducted the normality check and made the conclusion that these data samples are not normally distributed, hence use the Wilcoxon Signed-Rank Test as the substitute for the t-test to conduct the hypothesis test. For all browsers, we have listed the p-values in Table 2 of which is less than $0.05$. This result indicates that the energy consumption across the notification treatments is significantly different. Hence, we can reject the $H_0$ which states that there is no difference in the energy consumption when comparing the enabling and disabling of notification. This indicates that the status of notification impacts the energy consumption for both browsers. Specifically, there is more energy consumption when notification is enabled.

Table 2: p-value of Wilcoxon signed rank test - RQ1

<table>
<thead>
<tr>
<th>Browsers</th>
<th>Treatments</th>
<th>p-value</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chrome</td>
<td>on-off</td>
<td>$&lt; 2.2 \times 10^{-16}$</td>
<td>✓</td>
</tr>
<tr>
<td>Firefox</td>
<td>on-off</td>
<td>$1.043 \times 10^{-14}$</td>
<td>✓</td>
</tr>
</tbody>
</table>

Effect size estimation. In the last part of hypothesis testing, we concluded that there is an impact on energy consumption with different notification statuses. Here we use Cliff’s Delta to dive into more details about the impact. When analyzing the effect of notification status, we find both Firefox and Chrome show a medium effect size impact on the energy consumption, with the value of $0.51$(Firefox) and $0.75$(Chrome). Given that Chrome’s effect size is greater than that of Firefox, the status of notifications can potentially exert a more substantial influence on energy consumption when using Chrome compared with Firefox.

6.3 The impact of different browsers on energy consumption (RQ1-1)

We have concluded that there is more energy consumption when notification is on in the last section. To further investigate the influence of Web browsers on energy consumption, we will use the
discrepancy between having notifications enabled and disabled in two distinct browsers as our experimental data samples. Detailed explanation for this choice has been provided in Section 4.5.

**Hypothesis testing.** In this case, we aim to find the impact of different browsers on energy consumption. So there is one factor, namely browsers. We have concluded that these two data samples are not normal. Hence, we will use the Wilcoxon Rank-Sum test to conduct the hypothesis test based on RQ1-1. The p-value we got from this test is $9.291 \times 10^{-10}$ which is less than 0.05. Hence, we can reject $H_{0,brw,d}^a$ which states there is no big difference in the energy consumption when enabling the notification under different browsers. It suggests that there is evidence to support the alternative hypothesis ($H_{a,brw,d}^a$), which represents there is a difference in the energy consumption when enabling the notification under different browsers. So different browsers as a factor can influence energy consumption when enabling notifications.

If the P-value is less than 0.05, then reject the null hypothesis. Namely, there is a difference in the energy consumption resulting from the activated push API when using different browsers.

**Effect size estimation.** The value of effect size is medium (0.41) in the browser experiment, hence the factor of browsers can influence the energy consumption. It is also consistent with the previous result, there is a difference in energy consumption between different browsers.

### 6.4 The impact of different sending patterns on energy consumption (RQ2)

**Hypothesis testing.** For RQ-2, we aim to find the impact on the energy consumption brought by enabling notification under different sending frequencies and sending distribution patterns. As we have done the normality check, not all data sets are normally distributed, so we used the Wilcoxon Rank-Sum test to test every factor with two different treatments. Before we give the final p-values, box plots under different sending patterns are offered in Figure 6 and Figure 7. From the box plots, we can find there is no obvious difference when sending messages in the pattern of burst or even under chrome. Additionally, there is an interesting phenomenon of the box plots in Figure 6. When using Firefox, the boxplot of energy consumption in the low frequency shifted upward compared with the high frequency. We propose that this occurrence may be attributed to the mobile phone needing to wake up after a prolonged period with no notifications. Thus, the main energy consumption with the low frequency could be the wake-up overhead. A more comprehensive explanation will be provided in our discussion section.

All the p-values are listed in Table 3. The p-value of different distribution patterns under chrome is 0.9018 > 0.05. Hence, for the browser Chrome, we can accept $H_{0,dp,d}^a$ that there is no difference in the energy consumption of enabling notification when using different sending message distribution patterns under Chrome. Namely, the sending distribution pattern will not impact the energy consumption when using chrome. The other p-values are less than 0.05, so we can reject $H_{0,d}^f$ for both Chrome and Firefox and conclude that there is a difference in energy consumption when using different sending frequencies. We can also reject $H_{0,dp,d}^f$ when using Firefox, namely the sending distribution pattern will bring a difference in energy consumption when using Firefox.

All the results are also aligned with box plots. Additionally, all the energy consumption we have mentioned is calculated by subtraction of the energy consumption with notification on and off so that we focus on the energy consumption brought by enabling the notification rather than baseline energy consumption.

#### Table 3: p-values of Wilcoxon signed rank test - RQ2

<table>
<thead>
<tr>
<th>Browsers</th>
<th>Treatments</th>
<th>p-value</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chrome</td>
<td>idle-low</td>
<td>$4.237 \times 10^{-11}$</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>idle-high</td>
<td>$3.8 \times 10^{-13}$</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>low-high</td>
<td>0.02196</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>even-burst</td>
<td>0.9018</td>
<td>✓</td>
</tr>
<tr>
<td>Firefox</td>
<td>idle-low</td>
<td>0.006084</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>idle-high</td>
<td>$2.916 \times 10^{-13}$</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>low-high</td>
<td>$2.459 \times 10^{-10}$</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>even-burst</td>
<td>0.001169</td>
<td>✓</td>
</tr>
</tbody>
</table>

**Effect size estimation.** Table 4 shows the different effect size measures for different sending patterns. The pair idle-low has a large effect size in both Chrome and Firefox. Hence, we can conclude that the frequency is an important factor regarding the energy consumption for both two browsers. However, we observe a small effect size of 0.24 in the low-high pair for Chrome. This aligns with the previous finding from Figure 6 that there is a small variation in energy consumption between high and low-frequency settings when using Chrome to receive notifications. As for the effect of the distribution pattern, the even-burst pair shows a negligible result.
under Chrome, which is 0.01. It shows that different distributions in Chrome will not influence energy consumption, while this factor has a medium effect on energy consumption when using Firefox.

<table>
<thead>
<tr>
<th>Browser</th>
<th>Treatments</th>
<th>Effect Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chrome</td>
<td>low-high</td>
<td>0.24 (Small)</td>
</tr>
<tr>
<td></td>
<td>idle-low</td>
<td>0.98 (large)</td>
</tr>
<tr>
<td></td>
<td>idle-high</td>
<td>0.94 (large)</td>
</tr>
<tr>
<td></td>
<td>even-burst</td>
<td>0.03 (negligible)</td>
</tr>
<tr>
<td>Firefox</td>
<td>low-high</td>
<td>0.77 (large)</td>
</tr>
<tr>
<td></td>
<td>idle-low</td>
<td>0.82 (large)</td>
</tr>
<tr>
<td></td>
<td>idle-high</td>
<td>0.36 (Medium)</td>
</tr>
<tr>
<td></td>
<td>even-burst</td>
<td>0.34 (Medium)</td>
</tr>
</tbody>
</table>

Table 4: Effect size measures for different sending patterns

7 DISCUSSION

Which notification status? Based on the outcomes derived from our data analysis and statistical testing, we can confidently assert that, in both Chrome and Firefox, the activation of push notifications leads to a statistically significant increase in energy consumption in Telegram, even if no notification is received (i.e., idle). This observation aligns with expectations, considering the additional energy demands incurred by service workers functioning in the background to manage push events. Moreover, our medium-level effect size estimation further underscores that the impact of enabling notifications on energy consumption is substantial, particularly considering our experimental conditions, which involve settings that entail the deactivation of auditory tones, the absence of vibration alerts, and the absence of screen illumination for incoming notifications.

Which browser? While our study solely conducted a two-tail test to assess the difference in energy consumption between notification statuses across Chrome and Firefox, we can conclude that the impact of notification status on the energy consumption of Telegram diverges between these two browsers. However, our data analysis and effect size estimations reveal that, in most cases, the impact on Chrome is just slightly more pronounced than the impact on Firefox. Regardless of notification status, Chrome exhibits higher energy consumption than Firefox. One study conducted by Goncalves et al., in contrast to our findings, identified Chrome as the most energy-efficient browser [23]. However, this discrepancy may be attributed to the observed variations in browser energy consumption across the different types of Web apps, as stated in that study.

Which sending pattern? The distribution of messages, whether evenly or in bursts, did not exert a substantial influence on the energy consumption of Telegram. In our study, our primary focus pertains to discerning the impact of sending patterns in the context of push notifications. To disentangle the energy consumption attributable to message reception, we applied a methodological approach that subtracted comparative trials conducted with notification status deactivated, allowing us to isolate and evaluate the additional energy consumption directly associated with push notifications across different sending patterns.

Our findings for both Chrome and Firefox lead to the conclusion that significant distinctions exist in terms of energy consumption concerning push notifications with varying frequencies. In scenarios characterized by device idleness, the increment in energy consumption was minimal, whereas, for both low and high notification frequencies, the observed impacts were more pronounced in comparison with the idle scenario. However, contrary to what is commonly observed with the energy consumption of standalone instant messages, we noted a higher additional energy consumption for the low-frequency case (i.e., 5 messages per 100 seconds) compared to the high-frequency scenario (i.e., 50 messages per 100 seconds). A plausible rationale for this divergence is the introduction of a wake-up overhead imposed by the browser. In instances where the low-frequency message group prompts the system to awaken more frequently than the high-frequency group, it results in an escalated energy expenditure, attributable to the need to transition from a low-power state to address push events. This observation is also partially confirmed in the Push API W3C specification, which states that "the Push API may have to wake up the Service Worker associated with the service worker registration in order to run the developer-provided event handlers. This can cause resource usage, such as network traffic, that the user agent SHOULD attribute to the web application that created the push subscription" [5]. This observation is particularly useful for Web developers: if the Web app they are working on is not a time-sensitive one, then we suggest to cluster notification messages together in order to save energy.

Notably, within the scope of our experimental settings, the disparities in additional energy consumption associated with varying even and burst message distributions were relatively modest. In the context of Chrome, the distinctions in energy consumption between even and burst distributions were marginal, and with Firefox, the supplementary energy consumption linked to even distribution was slightly higher than that associated with burst distribution. The lower additional energy consumption observed for the latter case in Firefox can be attributed to the notably low notification-receiving rate exhibited during burst distribution.

Table 6: Mean energy consumption with screen illumination

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Chrome</th>
<th>Firefox</th>
</tr>
</thead>
<tbody>
<tr>
<td>low-even</td>
<td>58.437597</td>
<td>55.221537</td>
</tr>
<tr>
<td>low-burst</td>
<td>49.843350</td>
<td>41.659567</td>
</tr>
<tr>
<td>high-even</td>
<td>97.095123</td>
<td>110.748072</td>
</tr>
<tr>
<td>high-burst</td>
<td>56.940672</td>
<td>50.762066</td>
</tr>
</tbody>
</table>

In light of real-world usage scenarios where mobile users rarely disable all types of notification alerts while maintaining notification activation, we conducted additional trials to gain insights into how the illumination of the mobile screen might impact our results. Table 6 presents the average energy consumption in Joules when the mobile screen is illuminated for notifications, contrasting various notification-sending patterns. It is noteworthy that, under the high-even sending pattern, energy consumption exhibited a substantial increase. This result can be attributed to the device employed in our experiment, the Google Pixel 5, which illuminates the screen upon the reception of a notification and subsequently dims it after
approximately 2 seconds. In the high-even pattern, 50 messages are transmitted over a duration of 100 seconds, equating to precisely 2 seconds per notification. In contrast, during burst scenarios, 50 messages are dispatched in just 10 seconds. Consequently, the screen remains illuminated throughout the entire sending window in the case of the high-even pattern. For mobile users who routinely receive a significant volume of daily notifications, particularly in an evenly spaced pattern rather than a burst pattern, our findings suggest that maintaining screen illumination for incoming notifications is likely to result in markedly higher energy consumption.

8 THREATS TO VALIDITY

Internal Validity. To enhance the internal validity of our study, we have prepared a detailed experimental setting that carefully addresses potential extraneous variables that may impact measurement consistency across multiple trials [10]. This design encompasses fixed experimental parameters and variables specific to each trial. The fixed experimental settings are categorized into three dimensions: the measurement tool, the message-sending script, and the experimental environment. To maintain uniformity throughout the experiment, the following measures have been adopted. We use a single energy consumption profiling API, namely BatteryManager, with the assumption that, being it an officially-supported API by Google, it consistently and accurately captures energy consumption on Android devices. A standard configuration is applied to the message-sending script, encompassing fixed parameters for setup time, sending window, burst interval, and receiving buffer for each trial. Precautions are taken to clear caches, reset the subject application, and terminate irrelevant processes before the initiation of each trial. The designated setup time serves the purpose of introducing a cooldown period between successive runs, mitigating the influence of external factors, such as unanticipated background processes within the operating system. To minimize potential external influences, all irrelevant apps and processes are disabled or terminated before executing the experiment. In all runs, the mobile device is configured with consistent notification settings. Specifically, notifications are set to be inaudible (tone off) and imperceptible (vibration off), with the notification banner enabled. Finally, the display settings are set so to keep the display always locked and to prevent any illumination for incoming push notifications.

External Validity. Our research focuses on a single subject, Telegram, and exclusively examines one type of notification, namely push notifications, which addresses potential extraneous variables that may impact the investigation of the relationship between Web push notifications and energy consumption. The experimental setting is designed to eliminate the influence of external factors that are not directly related to the study. By using a consistent and standardized experimental environment, we aim to ensure that the results are generalizable to various contexts.

Construct Validity. Our experiment is carried out in a strictly controlled setting to alleviate the potential influence of external factors. The experiment design with careful considerations, including variable selection, hypothesis formulation, and pilot trial run, mitigates the construct risks such as mono-operation bias. However, one potential threat to the validity of our study resides in our assumption that the activation status of the Push API is contingent upon the notification settings of the application. To ensure that the Push API remains inactive during trials in which the notification status is switched off, we deactivate notification settings not only within the application but also within the browser itself. Moreover, we highlight the fact that in this experiment we are primarily interested in the frequency and distribution of the received notifications and we do not consider the payload of such notifications. Considering various sizes of payload is left for future replications of the study.

Conclusion Validity. In each of our experiment trials, we executed 30 repetitions. This, combined with the consideration of four factors and 20 trials, culminated in a total of 600 individual runs. To mitigate potential sources of bias arising from inappropriate testing methodologies, we conducted normality checks before each hypothesis testing, selecting the appropriate tests accordingly.

9 CONCLUSIONS

In this research, we research the relationship between Web push notifications, Web browsers, and energy consumption, and delve into the energy consumption within Web browsers and the impact of Web Push notifications on this critical aspect. After the experiment design and conduction, a noticeable difference is found in energy consumption when notifications are enabled compared to when they were disabled, for both browsers. Second, the energy consumption disparity between browsers is not to be underestimated, taking Chrome and Firefox as examples. Third, this study also provides objective evidence of an anecdotally-intuitive phenomenon: the more the received notifications, the more the energy consumption. While taking into consideration that the energy consumption of receiving notifications itself also follows this trend, the value we are measuring is additional energy consumption (energy consumption in total, minus energy consumption with notifications off), which reflects the effect of energy consumption of push notification. Under this condition, higher notification frequency is associated with increased energy consumption. Regarding the sending distribution, it has no noticeable effect in Chrome, while it has a significant one in Firefox; in the latter case, an even distribution pattern leads to higher energy consumption.

There are several promising avenues for future studies and research in the domain of energy efficiency in Web notification applications. Given the substantial impact of Web browser choice on energy consumption, further research could delve deeper into browser optimization techniques, which require deeper knowledge about the mechanism of browser behaviors. Besides, future research could be considered based on the results and findings of this paper: it is worth investigating the impact of energy consumption on push notifications between native mobile apps and PWAs.

Acknowledgements. This project is partially supported by the European COST programme under action identifier CA19135 (CERCIRAS).
REFERENCES