The Impact of Instant Messaging on the Energy Consumption of Android Devices

Stylianos Rammos, Mansi Mundra, Guijing Xu,
Chuyi Tong, Wojciech Ziolkowski, Ivano Malavolta
Vrije Universiteit Amsterdam, The Netherlands
{ s.l.rammos | m.mundra | g.xu | c.tong | w.m.ziolkowski }@student.vu.nl, i.malavolta@vu.nl

***Abstract*—*Context*. One of the primary uses of mobile devices is to send and receive instant messages via messaging apps. However, no evidence is still available about how receiving instant messages impacts the energy consumption of mobile devices.**
***Goal*. With this study we aim to empirically assess to what extent the number and distribution of received instant messages impact the energy consumption of Android devices.**
***Method*. The subjects of our experiment are WhatsApp and Telegram, two of the most known and used messaging apps. Each run of the experiment lasts 5 minutes and is executed on a Nexus 9 Android device. The independent variables of the experiment are: (i) the frequency of the received messages (i.e., 0, 10, 25, 50 per minute) and (ii) the distribution of messages arrival (i.e., evenly or in bursts). The dependent variable of the experiment is the energy consumption of the Android device in Joules.**
***Results*. We confirm that the energy consumption of the Android device tends to be proportional with the number of received messages across both apps. When the number of received messages is fixed, the frequency of their arrival does not significantly impact the energy consumption of the Android device.**
***Conclusions*. This study provides evidence that receiving instant messages can largely reduce the battery life of a user's Android device, even when the number of received messages is relatively low (i.e., 10 messages per minute). Moreover, sending bursts of messages does not lead to significant changes in terms of energy consumption. Developers can use this information to develop new features for their Instant Messaging apps for aggressively bundling messages without the risk of impacting the energy consumption of end users' devices.**

***Index Terms*—Empirical study, Energy consumption, Android, Instant messaging**

I. Introduction

Today, more and more people tend to use mobile devices at a high frequency for social interaction from everywhere and at any time. Therefore, mobile instant message (IM) applications continue to rise in popularity and play an important role in users' social life. For instance, WhatsApp, which is one of the most popular messaging applications worldwide, has two billion active users on a monthly basis as of July 2020 [11].

At the same time, energy consumption has gradually become a major challenge for developers and mobile device vendors since it can have a significant effect on user experience. For instance, users have an Always-on mentality [5] and might need to frequently charge their mobile devices or even plug in mobile power banks if the battery runs low quickly. Not only the hardware (*e.g.,* the display, cameras, network antenna) consumes energy, but also the software can affect the energy consumption of mobile devices [22]. When a mobile application is running, the energy consumption not only depends on the installed code, but also on how the user interacts with the application [27]. For example, the energy consumption of an IM application can be affected by the frequency and the number of instant messages sent and received by the user with their contacts.

When considering IM apps running on mobile devices, instant messages are generally received as *push notifications* [1]. Push notifications are messages generated in the Cloud and asynchronously received by a mobile app. Intuitively, Android push notifications work as follows. On the Android platform, there are a lot of services that provide push notifications functions for devices powered by the Android operating system. Most of those services are basically the Firebase Cloud Messaging (FCM) services. In order to push messages, the FCM needs to be integrated into the application. To push messages, a third-party server (e.g, WhatsApp server) sends a message request to the mobile notification service's (*e.g.,* FCM) server. After that, the mobile notification service's server transmits the request to the user's device. In its turn, the Android operating system running onboard the device checks the received information to associate it to the corresponding application and display it to the user.

For instant messaging applications, push notifications usually include notification icon in the status bar, messages in the notification drawer, heads up notification (from Android 5.0) and lock screen notification display (from Android 5.0). There are a few factors in message reception which can effect the energy consumption of a device, such as the setting of volume and vibration, the frequencies and distribution of messages received, and also, the post limit. To reduce the interruption from the notifications, developers usually tend to group the separate notifications into a group (available on Android 7.0 and higher), which means that the group allows users to collapse multiple notifications into just one post in the notification drawer. The user can then expand the notification to reveal the details for each individual notification [2].

The **goal** of this study is to assess the impact of instant messaging on the energy consumption of Android devices. Specifically, we empirically assess to what extent the *number* and *distribution* of received instant messages impact the energy consumption of Android devices running IM apps.

The **design** of the experiment is based on the following experimental variables. The independent variables of the ex-

periment are: (i) the frequency of received messages (*i.e.,* 0, 10, 25, 50 per minute) and (ii) the distribution of arrival of the messages (*i.e.,* evenly or in bursts). The dependent variable of the experiment is the energy consumption of the Android device in Joules. The subjects of our experiment are WhatsApp[1] and Telegram[2]. With a total number of monthly active users of 2.4 billion in July 2020, WhatsApp and Telegram are two of the most widely-used messaging applications in the Android app store [11]. Each run of the experiment has a fixed duration of 5 minutes and it is executed on a real mobile device (*i.e.,* an HTC Nexus 9). After having collected the measurement data for more than 300 independent runs, we statistically assess how energy consumption varies according to our empirical variables.

The main **results** of this study are that receiving instant messages can largely reduce the battery life of a user's Android device, even when the number of received messages is relatively low (*i.e.,* 10 messages per minute). Moreover, the study provides evidence that receiving instant messages either in bursts or evenly over time does not lead to significant differences in terms of energy consumption.

The **target audience** of this study is composed of both end users and developers. Indeed, the results of our experiment are relevant for both end users and developers alike. Specifically, we discover that (i) receiving instant messages can largely reduce the battery life of a user's Android device, even when the frequency of received messages is relatively low (*i.e.,* 10 messages per minute) and (ii) receiving bursts of messages does not lead to a statistically significant difference in terms of energy consumption.

The remainder of this paper is structured as follows. Section II discusses related work and the study design is reported in Section III. The results of the study are reported in Section V and discussed in Section VI, whereas threats to validity are elaborated in Section VII. Section VIII closes the paper.

II. Related Work

Burgstahler et al. analysed the energy consumption of two types of mobile communication approaches: push notifications and pull-based notifications [7]. The study covers in depth the evaluation of the energy impact on mobile devices of classic push notifications and compares them with a new solution proposed by the authors. In order to obtain precise results, the researchers used an external measurement device. The original battery was separated from the device and connected to special dummy battery in the phone with modified charging cradle in-between to intercept the power connection between the battery and the device. For push notifications, Google Cloud Messaging was used in encrypted and non-encrypted versions. The results of the study revealed that up to 7% improvement in battery consumption can be achieved by switching between the push and pull communication approach since they have different impacts on notification latency and power consumption. Switching between them depending on the context turned out to be more energy efficient. Our study complements the work by Burgstahler et al. since we target the number and distribution of messages receiving via a push-notification communication mechanism, which is the one officially employed by the Android platform.

Chowdhury et al. [10] studied whether maintaining the logs created by the applications to monitor their behaviour impacts battery performance. In order to profile the tested applications, GreenMiner was used. The study found that in most applications logging has little to no impact on the energy consumption, however in about 80% of cases there was at least one version of an application that impacted the battery in a significant manner. Although this paper focuses on a different aspect of mobile application profiling, the goals of measuring the energy usage and empirical methodology are similar.

Acer et al. [3] focused on energy efficient scheduling of push notifications. It argues that due to the sporadic nature of such messages and very small size, sending them immediately to the users device comes with a large cost to both energy consumption and network load. The authors come forward with the network-centric scheduling of push notifications that delays their delivery based on the predictions of users' network activities, allowing for sending multiple push messages at once instead of multiple messages over time. Although the overall goals of the paper differ from ours, it allows for more in-depth analysis of the network cost of push notifications. The energy needed to receive the push notification in a low coverage area where the phone might operate using more broadcasting power for a longer duration to communicate with the remote server through the base station might be one of the factors leading to increased energy usage.

Ding Li et al. analysed the energy consumption of Android applications analysing 405 real world market applications [21]. One of the outcomes of the analysis is that 61% of their energy consumption occurs during their idle states with network being the most energy consuming component. Even though our paper specifically targets push notifications, it is complementary to the one by Ding Li *et al.* since we consider different workloads and workload distributions of message reception during our experiment.

Yongmin Choi et al. [9] argues that new mobile applications cause frequent changes between connected and idle states of the radio equipment in the mobile devices leading to increased power consumption. Although this work focuses on the network aspects of the applications it shows again that the idle activity of mobile apps can bear heavy impact on the battery life of mobile devices.

III. Study Design

This section presents the main points of the study design. We refer the reader to the replication package of the study[3] for more details on the research method, tools, and collected data. The replication package contains all the information for

[1]https://play.google.com/store/apps/details?id=com.whatsapp

[2]https://play.google.com/store/apps/details?id=org.telegram.messenger

[3]https://github.com/S2-group/mobilesoft-2021-replication-package

independent verification and replication of the study, namely: (i) the Python scripts for executing the experiment, (ii) the raw data containing all the measures collected during the execution of the experiment, (iii) the R scripts for analysing the collected data, and (iv) a detailed guide for replicating the experiment.

A. Goal and Research Questions

We define the goal of this study by following the template presented by Wohlin et al. in [32]. Table I shows our goal formulation.

Object of study	IM app's message reception
Purpose	Assessing
Quality of focus	Energy Consumption
Perspective	End users and app developers
Context	Android mobile devices

TABLE I: Goal definition of our study

To achieve the aforementioned goal, this study aims to provide a clear and accurate response to the two research questions depicted below.

RQ1: What is the impact of receiving messages in IM apps at different *frequencies* on the energy consumption of Android devices?

Answering this research question helps app users in making better choices on using IM applications, *e.g.,* by choosing the ones that fit better the typical amount of received messages. App developers and Android maintainers can benefit from our answer to RQ1 because they will obtain evidence about how receiving IM-related push notifications can impact the energy consumption of an Android device under different workloads.

To answer this question, we compute the energy consumption of messages received by two messaging applications. More specifically, those applications are WhatsApp and Telegram. Those applications were selected based on their wide global popularity and their ability to perform automated messaging; a crucial requirement for conducting the experiment. The energy consumption is measured within a specific window of time for a set number of messages, received evenly at fixed time intervals. Those trials will be referred to as *active state trials* and the measurements taken during those trials (for each application) will be normalized using the measurements of *idle state trials*, referred to as the trials during which no messages are received but where the application runs in background.

RQ2: What is the impact of receiving messages in IM apps with different *distributions* on the energy consumption of Android devices?

By answering this research question we offer insights on the potential impact of bulk message reception on energy consumption. It is usual behavior to have messaging applications running in the background and receiving messages as they come, on a continuous basis. In this study, we compare this behaviour to the one where the time between messages is larger, but they arrive in bulk. The insights gained from such comparison can help developers in taking better informed decisions about how to issue push notifications to users (*e.g.,* by aggressively bundling them), specially to the ones who typically receive a large number of messages. The literature already confirmed that bundling HTTP requests reduces energy consumption without imposing significant runtime overhead in Android apps [24]; our answer to RQ2 will fill a similar gap in the context of push notifications.

To provide an answer to this research question, we once again measure the energy consumption of WhatsApp and Telegram. However, this time the messages are received in short bursts, such as the total amount of messages received remains the same but they are received in short periods of time rather than evenly distributed. The idle state trials will remain the same.

B. Subjects Selection

In order to produce results as close to the real-world use case scenario we have decided to use two popular messaging applications – WhatsApp and Telegram [11]. They have been chosen due to (i) their popularity and (ii) the availability of well-documented tools for programmatically sending messages to them (see Section IV); the latter point is particularly important for the viability of the experiment, where we need to reliably send several hundreds of messages during the execution of the experiment.

C. Experimental Variables and Hypotheses

This experiment has two independent variables, one for each research question. For RQ1, the independent variable is the **frequency of the messages** received by the application for the whole duration of the run (i.e., 5 minutes). We defined four treatments for this variable, namely:

- *Idle*: the IM app does not receive any messages during the whole duration of the run, it acts as the baseline for our experiment;
- *Low*: the IM app receives 10 messages per minute;
- *Medium*: the IM app receives 25 messages per minute;
- *High*: the IM app receives 50 messages per minute;

For every treatment, messages are always *randomly distributed* over the 1-minute time window.

For RQ2, the independent variable is the **distribution of arrival of the messages** within a time window of one minute. This variable has two treatments:

- *Even*: in every minute of the run, messages are received with an even distribution over time;
- *Burst*: messages are received all at once at the beginning of every minute of the run.

It is important to note that, in order to be able to observe possible effects of the distribution of arrival of messages, for RQ2 we are always sending 50 messages per minute.

For both RQ1 and RQ2, the dependent variable is the **energy consumption** in Joules of the Android device. We measure energy consumption by using Trepn, a software-based power profiler for Android devices. Trepn is widely used in empirical studies on energy-efficient software [26], [13], [19] and it has been reported as sufficiently accurate with respect to hardware power measurement (*e.g.,* the Monsoon Power

Monitor[4]), with an error margin of 99% [18]. The energy consumption E of each run of the experiment is computed by (i) measuring the average power P consumed by the Android device (in microWatts) and (ii) applying the following formula for obtaining the total amount of consumed energy in Joules during the 5-minutes window of each run:

$$E = (\frac{P}{10^6})W \times 300s \quad (1)$$

The null hypothesis for answering RQ1 states that the average energy consumption of the Android device does not significantly differ across all frequencies of received messages; we formally formulate it as follows.

$$H_0^{RQ1} : \mu_{idle} = \mu_{low} = \mu_{medium} = \mu_{high} \quad (2)$$

where μ is the average energy consumption of the IM apps across the $idle$, low, $medium$, and $high$ treatments of the *frequency* independent variable.

The alternative hypothesis for RQ1 states that the average energy consumption of the device is significantly different for at least one pair of considered frequencies; we formally formulate it as follows.

$$H_a^{RQ1} : \exists i, j \in \{idle, low, medium, high\} : \mu_i \neq \mu_j \wedge i \neq j \quad (3)$$

For RQ2, we are interested in the *distribution* of the arrival of the messages and it states that the average energy consumption of the Android device does not significantly differ when receiving messages either evenly or in bursts; we formally formulate it as follows.

$$H_0^{RQ2} : \mu_{even} = \mu_{burst} \quad (4)$$

The alternative hypothesis for RQ2 states that the average energy consumption of the device is significantly different when messages arrive with different distributions; we formally formulate it as follows.

$$H_1^{RQ2} : \mu_{even} \neq \mu_{burst} \quad (5)$$

D. Experiment Design

We designed this study so to investigate on each research question in isolation. This decision is due to the fact that (i) for each RQ we want to have a complete design (*i.e.,* to cover all combinations of subjects and treatments) and (ii) we want to keep the random distribution of message arrivals when answering RQ1 (this is in contrast with the even/burst distributions we consider in RQ2).

TABLE II: Trials for answering RQ1 (examples)

	idle	low	medium	high
WhatsApp	1st	3rd	2nd	4th
Telegram	2nd	1st	4th	3rd

30 repetitions for each trial

[4]https://www.msoon.com

For answering RQ1 we adopt a randomized complete design [32]. Having a complete design allows us to investigate all possible treatments in the context of all subjects. Table II presents the trials resulting from such design, where each treatment is applied to both the selected subjects (*i.e.,* WhatsApp or Telegram) in a random order for 30 times. We repeated every trial of the experiment 30 times in order to take into account possible fluctuations of the measured energy consumption [4]. The estimated running time for answering RQ1 is $(2 \times 4 \times 30) \times 5m = 1,200m = 20h$ since we are considering 2 IM apps, 4 treatments, 30 repetitions per trial, and 5 minutes per run.

TABLE III: Trials for answering RQ2 (examples)

	even	burst
WhatsApp	1st	2nd
Telegram	2nd	1st

30 repetitions for each trial

We apply a randomized complete design also for answering RQ2 with one factor (*i.e.,* the distribution of the received messages), two treatments (*i.e.,* even and burst), and 2 subjects (*i.e.,* WhatsApp and Telegram). Table III shows the trials used according to the mentioned design. The running time for answering RQ2 is $(2 \times 2 \times 30) \times 5m = 600m = 10h$. Also in this case we repeat every trial 30 times.

E. Data Analysis

The data analysis related to each research question consists of three main phases: data exploration, hypothesis testing, and effect size estimation.

In the data exploration phase we aim at obtaining a preliminary understanding of the obtained energy measures. We compute summary statistics of the obtained measures and visualize them by means of density plots, box plots, and histograms.

In the hypothesis testing phase we aim at answering the research questions of the study by applying statistical tests. For RQ1 we plan to use the One-Way ANOVA statistical test since our independent variable has more than two treatments. If the assumptions of the ANOVA test are not met, then we will apply a non-parametric one, *i.e.,* the Kruskal Wallis statistical test [20]. In order to identify which pairs of treatmets are significantly different, we apply the Dunn Test as post-hoc analysis [15] (with Benjamini-Hochberg correction to reduce the chances of Type-I errors). Differently, when answering RQ2 we plan to use a statistical test fitting a 1-factor-2-treatments study design, *i.e.,* the paired t-test [29]. If the assumptions of the mentioned statistical test are not met by the collected measures, then we apply a non-parametric statistical test, *i.e.,* the Wilcoxon Rank-Sum test [17]. All statistical tests are executed with with α = 0.05.

Finally, we assess the magnitude of the differences among the considered treatments via the Cliff's Delta effect size measure [12]. The Cliff Delta is a non-parametric measure of effect size for ordinal variables and it does make any assumptions about the distributions being compared. The values of the Cliff

Delta measures are interpreted according to the guidelines proposed by Grissom *et al.* [16] and reported according to the following ranges: negligible, small, medium, and large [16].

IV. Experiment Execution

This section provides the technical details of the infrastructure we setup for executing the experiment, as well as the various software tools and hardware devices we used.

As shown in Figure 1, the main components for executing the experiment are three: (i) a laptop for executing the orchestration logic of the experiment and for collecting the measures from the Android device, (ii) an Android device for running the subjects of the experiment, and (iii) the Cloud hosting the APIs for delivering the instant messages.

For orchestrating the execution of all the runs of the experiment we make use of Android Runner [25]. Android Runner is a Python framework for automatically executing experiments involving both native and web applications running on Android-based devices. In accordance to Android Runner, we define our experiment in a descriptive manner as a JSON file, and then the full execution of the experiment is managed by the tool via a combination of Python scripts and Android Debug Bridge (ADB) commands.

In our experiment, Android Runner is executed on a laptop running MacOS Catalina 10.15.6 with a 2.2 GHz 6-Core Intel Core i7-8750H processor and 16Gb of memory.

The technical specifications of the Android device on which the subjects are running are reported below.

- *Manufacturer*: HTC
- *Model*: Nexus 9
- *Android Version*: 7.1.1
- *CPU*: Nvidia Tegra K1
- *Memory*: 2 GB

Both the laptop and the Nexus 9 run under the same Wifi network with a speed of 100 Mbps. To ensure that the Wi-Fi conditions do not alter the results of the experiment, the Nexus 9 and the laptop are the only devices connected to the network and they are always placed at the same distance from the Wi-Fi router. Further, we take special care in keeping the execution environment as clean as possible, specifically: the Nexus 9 is loaded with a clean installation of the Android OS, it has been configured so to do not perform any OS updates, all third-party apps have been uninstalled, and push notifications for apps different from Telegram and WhatsApp have been disabled as well. The specific versions of the subjects of the experiments are reported below:

- Telegram - v7.0.1 (released on 22.08.2020)
- WhatsApp - v2.20.199.14 (released on 15.09.2020)

With the laptop connected to the Nexus 9, the first steps for executing a run of the experiment are: (i) to stop all previously running apps (step 1 in Figure 1) and (ii) to start the Trepn energy profiler (step 2). Trepn is distributed as an Android application and, thanks to the private APIs provided by the Qualcomm CPU mounted on the Nexus 9 device, it is able to accurately measure the energy consumed by the device itself. In this specific phase of the run, Trepn is active, but it is still

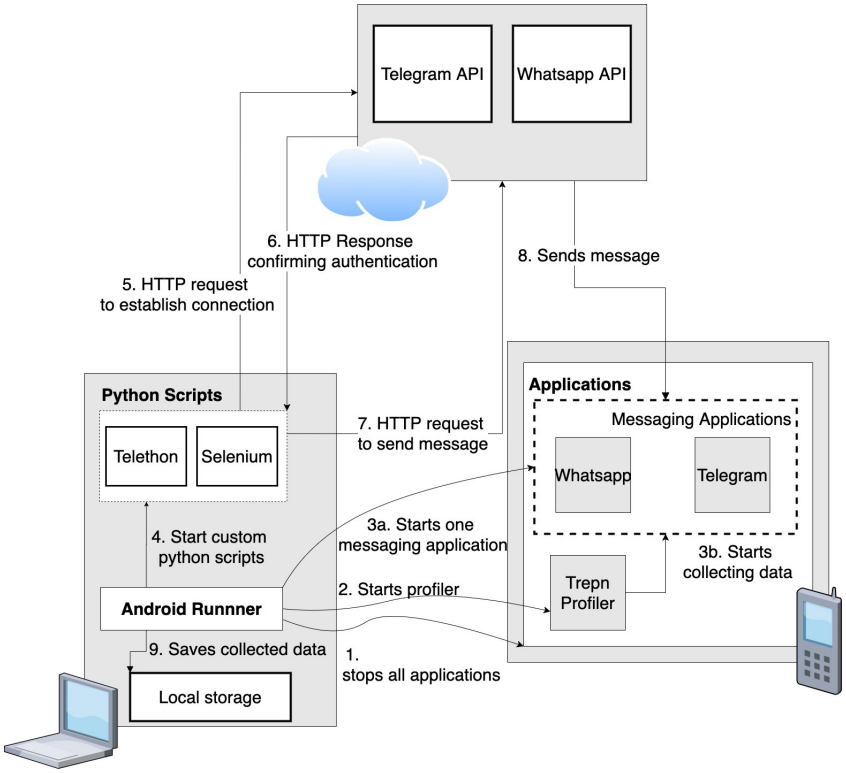

Fig. 1: Overview of the measurement infrastructure and performed steps while executing a run of the experiment

not profiling the energy consumed by the Nexus 9 device.

Then, in step 3 we (i) start the IM app to be measured (*i.e.,* either WhatsApp or Telegram, depending on the plan of the runs created by Android Runner) and (ii) start collecting data about energy consumption via Trepn.

As shown in Figure 1, in addition to Android Runner, we use a series of python scripts to interact with each application's API and send messages programmatically (step 4-8). The messages are sent using custom Python scripts interacting with the application's API, following the experiment planning described previously. The decision to use custom Python scripts was made to have full control over the experiment execution and more flexibility on the time intervals between each message. These scripts use the following Python libraries for each IM app: Telethon for Telegram and Selenium for WhatsApp. Since WhatsApp does not provide a set of APIs to programmatically send instant messages, we use a second Android device to send the WhatsApp messages. The technical specifications of the second device for sending WhatsApp messages are not relevant for the experiment since they do not have an effect on the energy consumption of device receiving the messages.

Finally, after 5 minutes of running-and-profiling time, the current run is stopped and the data collected by Trepn is fetched and saved locally on the laptop (step 9).

In order to take into account the intrinsic variability of energy measurement, we take the following precautions: (i) the order of execution of the experiment runs is randomized, (ii) the measurement of each IM app is repeated 30 times, (iii) between each run the Nexus 9 remains idle for 2 minutes so to take into account tail energy usage, *i.e.,* the phenomenon where where certain hardware components of mobile devices are optimistically kept active by the OS to avoid startup energy costs [23], and (iv) the IM apps are cleared before each run so to reset their cache and persisted data.

V. Results

In this section we report the obtained results according to the research questions of the study.

A. Impact of receiving IMs at different frequencies (RQ1)

Data exploration. Figure 2 and Table IV give an overview of the energy consumed in each run of the experiment. The average energy consumption of the devices in the time slot of 5 minutes is 18.391 Joules, although there are some cases where the energy consumption is comparatively large (*e.g.,* 143.949 joules). The skewness of the energy consumption is 2.812, which implies that the data is not symmetrical.

TABLE IV: Descriptive statistics for RQ1

	Energy (Joules)
Minimum	4.016
1st quartile	7.346
Median	12.779
Mean	18.391
3rd quartile	23.030
Maximum	143.949

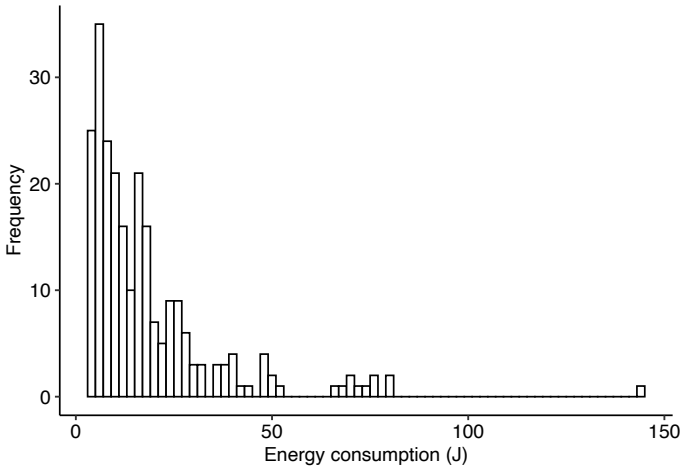

Fig. 2: Frequency of the energy consumption values (RQ1)

Figure 3 and Table V show a visualization and summary statistics of the energy consumption of the Android device across the four treatments of the *distribution* variable. We can observe that the energy consumption of the Android device grows with the increase of the numbers of received messages per minute.

It is also interesting to note that: (i) the energy measures of the idle treatment are very compact, this can be considered as a good indication of the reliability of our measurement infrastructure, and (ii) the variance of the energy measures increases when considering higher frequencies of arrival of instant messages, this phenomenon might be due to the best effort model of Android push notifications or the platform/OS managing push notifications differently (*e.g.,* grouping); a deep investigation of this phenomenon is left for future work.

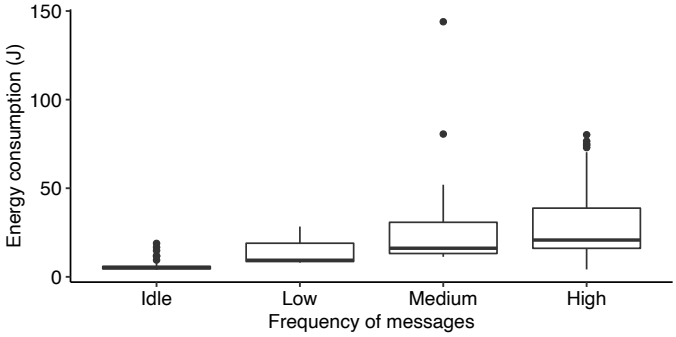

Fig. 3: Energy Consumption for each treatment in RQ1

TABLE V: Energy Consumption for each treatment in RQ1

Treatment	**Energy Consumption (Joules)**			
	Mean	Median	Q1	Q3
Idle	6.064959	5.231174	4.484845	5.907169
Low	13.530337	9.368802	8.648248	18.968561
Medium	25.008157	16.140350	13.169032	30.774704
High	28.960692	20.766082	16.098280	38.774762

Figure 4 and Table VI present the measured energy consumption across the two subjects of the experiment. It can be seen that when the system is idle or with low workload, the two apps consume a similar amount of energy. Differently, when the number of received messages is medium and high, Whatsapp messages tend to consume more energy than Telegram messages. Also, the energy consumption of Whatsapp tends to have a higher variance than Telegram.

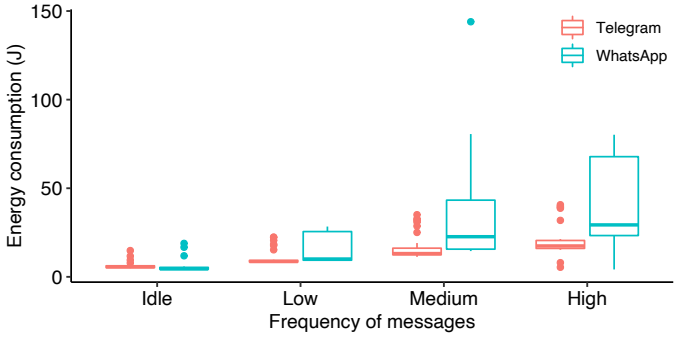

Fig. 4: Energy consumption for different treatments of Telegram and Whatsapp

Treatment	Subject	Energy Consumption (Joules)				
		SD	Mean	Median	Min	Max
Idle	Telegram	2.160	6.355	5.633	4.866	14.835
	WhatsApp	3.576	5.774	4.481	4.016	18.880
Low	Telegram	4.574	10.789	8.648	8.000	22.372
	WhatsApp	8.038	16.271	10.064	8.942	28.380
Medium	Telegram	7.331	17.004	13.135	11.295	35.070
	WhatsApp	26.530	33.013	22.657	14.543	143.949
High	Telegram	8.831	20.199	17.438	5.402	40.696
	WhatsApp	26.258	37.722	29.315	4.167	80.191

TABLE VI: Energy consumption for different treatments of Telegram and Whatsapp

Hypothesis testing. Since our dependent variable(energy consumption) is continuous, and the samples are independent, we need firstly check whether the distribution of the dependent variable in the four treatments is normal. The density plots of the energy consumption for the four treatments are shown in Figure 5. These density plots help us to get an initial indication about the distribution of the energy consumption for each treatment. It can be seen in the figures that all of the distributions in the four groups seem to be not normal. We apply the Shapiro-Wilk normality test to statistically assess the normality of the data related to the four treatments. Table VII shows the p-values of the Shapiro-Wilk test. All of them are lower than the significance threshold, thus we can reject the null hypothesis that the distribution of the data is normal. The skewness of each group is also calculated, and we obtained the following results: idle=2.80, low=0.98, medium=3.45, and high=1.08. In fact, if we refer to Figure 5, we can observe that the density of the energy consumption has a peak at the lower values of the data.

Moreover, the normality check for the distribution of residuals is conducted. The p-value of the Shapiro-Wilk normality test of residuals is 2.2e-16, which means that the distribution of the residuals is not normal, either.

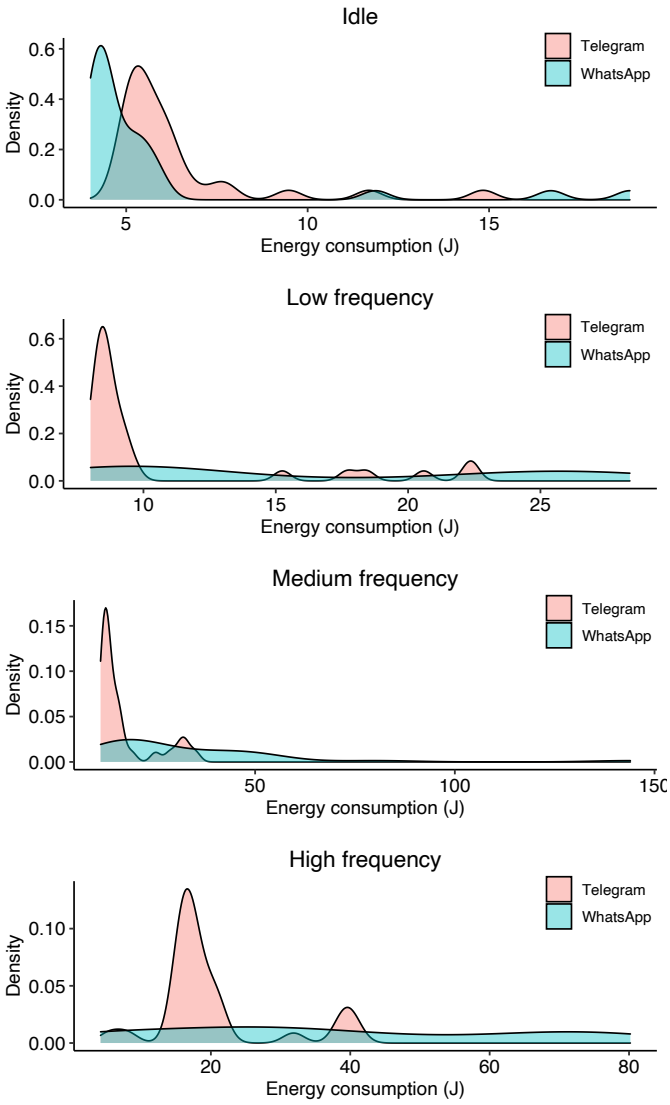

Fig. 5: Density plots of the four treatments of the *frequency* variable

TABLE VII: P-values of the Shapiro-Wilk normality test

Messages frequency	p-value
Idle	1.406842e-11
Low	1.237924e-09
Medium	2.214018e-11
High	1.886828e-06

The Levene's Test for homogeneity of variance is also applied to check the homoscedasticity of data. The p-value of the Levene's test is 6.928e-08, which is less than our significance threshold, so it can be deducted that the variance across the treatments is statistically significantly different, and the assumption of homoscedasticity is not met. In conclusion, given that (i) our data is not normally distributed in all of the

four treatments, (ii) the residuals are not normally distributed either, and (iii) the variance is not homogeneous, we apply the Kruskal-Wallis statistical test for RQ1.

The application of the Kruskal-Wallis test produced a p-value of 2.2e-16, which is lower than our significance threshold (*i.e.,* 0.05). Therefore, we can reject the null hypothesis H_0^{RQ1}, which states that the energy consumption of the Android device is the same for all treatments can be rejected. This result provides evidence that receiving instant messages at different frequencies has an impact on the energy consumption of Android devices.

TABLE VIII: Adjusted p-values of the Dunn's test

Treatments	p-value	Significance
idle - low	9.683172e-09	✓
idle - medium	8.001321e-21	✓
idle - high	9.994922e-22	✓
low - medium	3.501339e-04	✓
low - high	1.396984e-04	✓
medium - high	7.753923e-01	-

In order to identify the frequencies of message arrival with higher impact on energy consumption, we apply the Dunn's test for the non-parametric pairwise multiple comparison between the four different treatments. The p-value adjustment with Benjamini-Hochberg method is also applied to avoid the higher probability of getting statistically significant results when doing multiple comparisons. As shown in Table VIII, almost all pairs of treatments exhibit a statistically significant difference, with the exception of the medium-high pair.

Effect size estimation. Effect size is a quantitative measure of the magnitude of the difference among groups, which can help us to acknowledge the strength of a phenomenon. An effect size can be used to explain how important a difference is: large effect sizes mean the difference is important, while small effect sizes mean the difference is unimportant [30].

TABLE IX: Effect size measures for RQ1

Treatments	Effect size
idle - low	0.866 (large)
idle - medium	0.951 (large)
idle - high	0.834 (large)
low - medium	0.591 (large)
low - high	0.452 (medium)
medium - high	-

As shown in Table IX, we found a *large* effect size between all pairs involving the baseline (*i.e.,* the *idle* treatment). When considering the *low* treatment we found a *large* effect size with respect to the *medium* treatment and a *medium* effect size with respect to the *high* treatment. Finally, as expected, we found a *small* effect size when considering the *medium - high* pair (it was already exhibiting a statistically insignificant difference according to our Dunn's test).

B. Impact of receiving IMs with different distributions (RQ2)

Data exploration. Figure 6 and Table X give an overview of the measurement data we collected across the *even* and *burst* messages distributions. The median values of the two treatments are similar. This could be the result of the same total message amount for each treatment. The mean value of the *even* treatment is higher than the mean value of the *burst* treatment, due to the presence of more outliers and more data points in the first quartile.

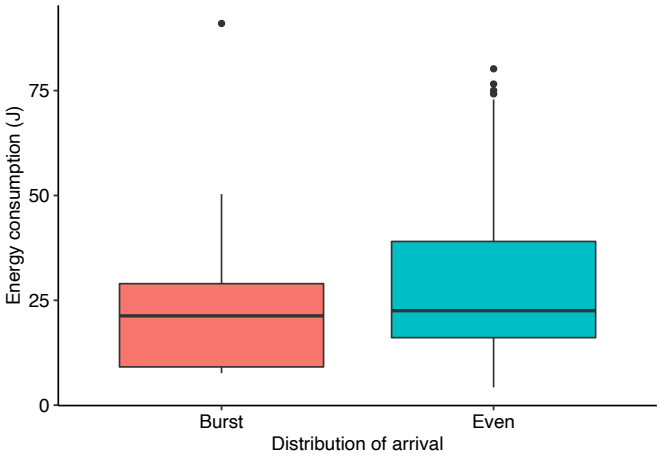

Fig. 6: Energy Consumption for each treatment in RQ2

TABLE X: Energy Consumption for each treatment in RQ2

Treatment	Energy Consumption (Joules)			
	Min	Median	Mean	Max
Even	4.249	22.525	30.058	80.191
Burst	7.641	22.171	21.142	90.999

Hypothesis testing. In order to apply the (parametric) t-test, we need to check if the data meets its assumptions. The first assumption of the t-test is about the normal distribution of the data to be analyzed. Figure 7 presents the density of the energy consumption of the device for each treatment. From a visual inspection, the data belonging to both treatments seems to be not normally distributed.

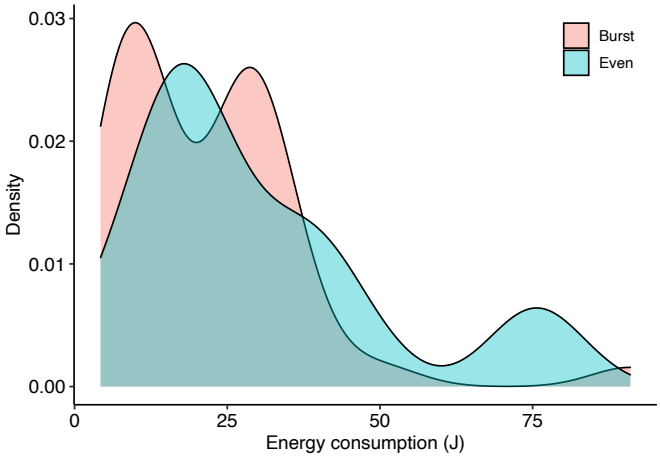

Fig. 7: Density plots of the treatments of the *distribution* variable

We statistically assess the normality of the data via the Shapiro-Wilk normality test. The obtained p-values are: 7.379e-05 for the *even* treatment and 8.303e-07 for the *burst* treatment. This means that we can reject the null hypothesis that our collected data lies within a normal distribution, in turn providing statistical evidence that our collected data is not normally distributed.

Since the assumption of normal distribution is not met, we need to apply a non-parametric statistical test. As discussed in Section III-E, we apply the Wilcoxon Rank-Sum test. We obtained a p-value of 0.061, which is slightly higher than the significance level of 0.05. Thus, we cannot reject the null hypothesis H_0^{RQ2} and we cannot claim that different distributions of the arrival of instant messages impact differently the energy consumption of an Android device.

Effect size estimation. As expected, the value of Cliff's Delta measure is *small* (0.244), meaning that the energy consumption of the Android device is similar across the *even* and *burst* frequencies of arrival of instant messages.

VI. Discussion

With the help of statistical tests and the data analysis, our two research questions can be answered. The conclusion for the first research question is that generally the energy consumption grows with the number of received instant messages; nevertheless, such growth is not obvious when considering the *medium* and *high* treatments. Although it is a natural outcome that energy consumption increases with message load, it is important to validate it through statistical analysis. These results also set the stage for future research which can further investigate the rate of increase in power consumption compared to the message load. As presented in our experiment results, there is no significant difference between the medium and high workload, indicating there might be a cut-off point at which additional messages do not add relevant overhead for Android systems. Further investigations are needed to provide evidence about this specific point. For developers, there might be an opportunity to reduce energy consumption of messages by investigating this cut-off point.

For the second research question, our study does not provide evidence that the distribution of messages received on the Android device has a significant impact on energy consumption. Nevertheless, in Figures 6 and 7 we can observe that evenly receiving messages impacts energy consumption slightly more than receiving messages in bursts. We conjecture that this slight difference might be due to tail energy consumption, *i.e.,*, the phenomenon where components such as the network card remain in high power states after completing a task (*e.g.,* receiving a push notification) [28], [8]. Techniques for mitigating this possible cause of energy drain exist, such as the batch operation pattern presented in [14]. The results we obtained for RQ2 include valuable information for end users and developers alike. On the one hand, end users can trust the Android platform when dealing with sudden bursts of messages, which might not drain the battery of their devices. On the other hand, developers could use this information to develop new features for their IM applications, such as an advanced "do not disturb" mode which does not let the messages go through until it is turned off, without impacting the energy consumption.

Our answers to the two research questions of this study can guide Android users in using their device in a more efficient way, specially when they need to save energy or when their battery is low. For example, users can turn off the notifications of their IM applications when the battery of the device is low. Also, our findings might help the developers acknowledge the energy consumption of devices when the messaging applications are working, and also provide insights about the energy efficient design of messaging applications on the Android platform (*e.g.,* by providing mechanisms for automatically turning off the notifications of IM apps when the battery is low).

VII. Threats To Validity

A careful analysis of the threats to validity of an experiment is essential to assess its scientific validity and to contextualize it to other experiments [32]. Every design choice can bring an array of threats which must be identified and clearly presented alongside the experiment results. Therefore, in this section, we discuss some possible threats to the validity of our study.

A. External Validity

The experiment we conducted considers two subjects – WhatsApp and Telegram. This can be considered too little subjects to generalise our conclusions on all messaging applications. However, those two applications are amongst the most used instant messaging apps in the Google Play Store and our experiment can be replicated on other subjects with relatively low effort. Future experiments may include additional subject such as Facebook Messenger or WeChat, such that the subjects are more representative of the general population of IM applications.

The frequency (*i.e.,* 0, 10, 25, and 50 per minute) and distribution of arrival (even and burst) of IMs have been defined so to facilitate the execution and reporting of the experiment, and its replication by third parties. Clearly, in a realistic scenario, users might not receive messages exactly at those intervals. Although the definition and usage of realistic intervals is out of the scope of this study, this design decision can have an impact on the external validity since the obtained results might not directly generalize to real usages of IM apps.

As hardware device, we used an HTC Nexus 9, which is a relatively modern device with common hardware specifications. With this choice we can be reasonably sure to have a realistic experiment that can directly translate into a real world scenario. Nevertheless, newer devices running newer Android releases may lead to different energy measurements; further replications of the performed experiments can help in mitigating this potential threat to validity.

B. Internal Validity

There is a very small chance that measurements across trials vary due to external factors that change slightly between trials,

such as a background processes running sporadically in the operating system, garbage collection, etc. We applied several strategies to mitigate this effect, such as: we randomized the order of subjects and treatments for each trial, we completely reset the subjects of the experiment at every run, we impose a cool-down period of 2 minutes between each run, etc. Therefore, the risk of a previous trial or a random background process affecting the measurements of a run of the experiment are greatly reduced.

The energy consumption measures were done using a single method, namely Trepn. This means the measurement of this tool could not be corroborated with a second tool. Nevertheless, we are reasonably confident about the reliability of our measures since Trepn is well-accepted and used power profiler in the software engineering community and its measurements accuracy has been shown to be accurate when compared to hardware-based profilers [18].

The reliability of the measures can be affected by various factors, such as the brightness of the display, the network conditions, etc.. In order to mitigate possible biases, we fixed those external factors. For example, the brightness of the screen of the device was always set to the minimum, the distance from the WiFi access point was kept constant, etc.

Finally, being the two experimental subjects commercial (closed-source) apps, we do not have any indication about whether they are implementing some message bundling techniques. The existence of internal message bundling techniques might potentially influence the results of our experiment, specially the ones related to RQ2. Nevertheless, given the long execution time of our runs and the relatively large resolution of considered time slots (*i.e.,* 1 minute), we are reasonably confident that our *even* and *burst* treatments predicate on a more large grain with respect to the (possibly micro-)bundling techniques of Telegram and WhatsApp. We leave the study of the internal bundling mechanisms of IM apps for future work.

C. Conclusion Validity

To mitigate this type of threats, we used a fixed number of treatments for our experiments, with 2 subjects. Overall, 2 messaging applications were executed 30 times per messaging app and per treatment. As a result, we have total sample size of 360, which is relatively large for a measurement-based experiment.

Moreover, before performing the statistical analysis, we made sure to check whether the assumptions of the used statistical tests are met (*e.g.,* data normality). This helped us to ensure that the appropriate tests are performed on the data.

Finally, it is important to note that in our hypothesis testing we combined the data for WhatsApp and Telegram. If on one side this allows us to mitigate the risk of fishing, on the other side it might have masked a potential phenomenon happening only for one of the two apps. We leave a more fine-grained statistical analysis for future work.

D. Construct Validity

We used the Goal-Question-Metric method [6], [31] to define *a priori* the main components of the experiment. The goal of our experiment as well as the questions related to this goal and the metrics that are relevant to answer those questions have been formalized in a GQM-tree (it is part of the replication package). Using the GQM method, we also formulated the hypotheses to address the research questions and identified the independent and dependent variables for our experiment.

When answering RQ2, we are always sending 50 messages per minute. This design decision is due to the need of clearly observe possible effects of the distribution of arrival of messages. A potentially interesting future work of this study would include the investigation of how different frequencies of messages (*i.e.,*, 0, 10, 25, 50) might correlate with the distribution of arrival of messages (*e.g.,* a burst of 50 messages might behave differently from an energy point of view with respect to a burst of 10 messages).

Finally, a complete replication package is publicly available for independent verification and inspection of each step of the performed experiment.

VIII. Conclusions

In this article we measure the energy impact of the messaging applications notifications on a mobile device running the Android operating system. In our experiment we compared the energy consumption of the device while running the WhatsApp and Telegram apps. Our empirical variables are two, namely: the frequency of the received messages (*i.e.,* 0, 10, 25, 50 per minute) and (ii) the distribution of messages arrival (*i.e.,* evenly or in bursts). The response variable of the experiment is the energy consumed by the Android device in Joules. Each run of the experiment lasts 5 minutes and is executed on a Nexus 9 Android device. The main result of the experiment is the statistically significant difference in energy consumption when receiving small/medium/large numbers of instant messages for both WhatsApp and Telegram; however, there is no significant difference when receiving them either evenly per minute or in bursts.

This study provides evidence that receiving instant messages can largely reduce the battery life of a user's Android device, even when the number of received messages is relatively low (i.e., 10 messages per minute). Moreover, receiving messages in bursts does not lead to significant changes in terms of energy consumption.

As future work, we are planning to carry out the following activities: (i) to replicate the study targeting a higher number of subject apps and performing a per-app statistical analysis, (ii) to conduct an experiment targeting additional open-source apps (like Signal[5] and inspecting how they internally manage the reception of instant messages (*e.g.,* micro-bundling messages), and (iii) to investigate how push notifications are managed at the OS level in Android, how the used mechanisms can be linked to the results we obtained in this study, and how such knowledge might help in improving the energy consumption of Android devices.

[5]https://signal.org

REFERENCES

[1] Push notifications mechanism for ios and android. Accessed 21 October 2020. https://idapgroup.com/blog/push-notifications-mechanism-for-ios-android/.

[2] Notifications overview in android platform. Accessed 25 September 2020. https://developer.android.com/guide/topics/ui/notifiers/notifications, 2020.

[3] U. Acer, A. Mashhadi, C. Forlivesi, and F. Kawsar. Energy efficient scheduling for mobile push notifications. In *proceedings of the 12th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services on 12th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 100–109, 2015.

[4] L. Ardito, R. Coppola, M. Morisio, and M. Torchiano. Methodological guidelines for measuring energy consumption of software applications. *Scientific Programming*, 2019, 2019.

[5] L. Baresi, W. G. Griswold, G. A. Lewis, M. Autili, I. Malavolta, and C. Julien. Trends and challenges for software engineering in the mobile domain. *IEEE Software*, 38(1):88–96, 2020.

[6] C. G. R. H. Basili, V.R. *Encyclopedia for Software Engineering*. J. Marciniak (ed.), 1994.

[7] D. Burgstahler, N. Richerzhagen, F. Englert, R. Hans, and R. Steinmetz. Switching push and pull: An energy efficient notification approach. In *2014 IEEE International Conference on Mobile Services*, pages 68–75. IEEE, 2014.

[8] H. Cai, Y. Zhang, Z. Jin, X. Liu, and G. Huang. Delaydroid: reducing tail-time energy by refactoring android apps. In *Proceedings of the 7th Asia-Pacific Symposium on Internetware*, pages 1–10, 2015.

[9] Y. Choi, C.-h. Yoon, Y.-s. Kim, S. W. Heo, and J. A. Silvester. The impact of application signaling traffic on public land mobile networks. *IEEE Communications Magazine*, 52(1):166–172, 2014.

[10] S. Chowdhury, S. Di Nardo, A. Hindle, and Z. M. J. Jiang. An exploratory study on assessing the energy impact of logging on android applications. *Empirical Software Engineering*, 23(3):1422–1456, 2018.

[11] J. Clement. Most popular global mobile messaging apps 2020. Accessed 11 September 2020. https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps/, 2020.

[12] N. Cliff. Dominance statistics: Ordinal analyses to answer ordinal questions. *Psychological bulletin*, 114(3):494, 1993.

[13] M. Couto, J. Saraiva, and J. P. Fernandes. Energy refactorings for android in the large and in the wild. In *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 217–228. IEEE, 2020.

[14] L. Cruz and R. Abreu. Catalog of energy patterns for mobile applications. *Empirical Software Engineering*, 24(4):2209–2235, 2019.

[15] O. J. Dunn. Multiple comparisons among means. *Journal of the American statistical association*, 56(293):52–64, 1961.

[16] R. J. Grissom and J. J. Kim. *Effect sizes for research: A broad practical approach.* Lawrence Erlbaum Associates Publishers, 2005.

[17] M. Hollander, D. A. Wolfe, and E. Chicken. *Nonparametric statistical methods*, volume 751. John Wiley & Sons, 2013.

[18] M. A. Hoque, M. Siekkinen, K. N. Khan, Y. Xiao, and S. Tarkoma. Modeling, profiling, and debugging the energy consumption of mobile devices. *ACM Computing Surveys (CSUR)*, 48(3):1–40, 2015.

[19] Y. Hu, J. Yan, D. Yan, Q. Lu, and J. Yan. Lightweight energy consumption analysis and prediction for android applications. *Science of Computer Programming*, 162:132–147, 2018.

[20] W. H. Kruskal and W. A. Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, 47(260):583–621, 1952.

[21] D. Li, S. Hao, J. Gui, and W. G. Halfond. An empirical study of the energy consumption of android applications. In *2014 IEEE International Conference on Software Maintenance and Evolution*, pages 121–130. IEEE, 2014.

[22] D. Li, S. Hao, J. Gui, and W. G. J. Halfond. An empirical study of the energy consumption of android applications. In *2014 IEEE International Conference on Software Maintenance and Evolution*, pages 121–130, 2014.

[23] D. Li, S. Hao, W. G. Halfond, and R. Govindan. Calculating source line level energy information for android applications. In *Proceedings of the 2013 International Symposium on Software Testing and Analysis*, pages 78–89. ACM, 2013.

[24] D. Li, Y. Lyu, J. Gui, and W. G. Halfond. Automated energy optimization of http requests for mobile applications. In *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, pages 249–260. IEEE, 2016.

[25] 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-ĂŹ20)*, pages 61–66.

[26] I. Malavolta, G. Procaccianti, P. Noorland, and P. Vukmirovic. Assessing the impact of service workers on the energy efficiency of progressive web apps. In *2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, pages 35–45. IEEE, 2017.

[27] A. A. Nacci, M. Mazzucchelli, M. Maggio, A. Bonetto, D. Sciuto, and M. D. Santambrogio. Morphone. os: context-awareness in everyday life. In *2013 Euromicro Conference on Digital System Design*, pages 779–786. IEEE, 2013.

[28] A. Pathak, Y. C. Hu, M. Zhang, P. Bahl, and Y.-M. Wang. Fine-grained power modeling for smartphones using system call tracing. In *Proceedings of the sixth conference on Computer systems*, pages 153–168, 2011.

[29] J. Rosenberg. Statistical methods and measurement. In *Guide to Advanced Empirical Software Engineering*, pages 155–184. Springer, 2008.

[30] T. Schafer and M. A. Schwarz. The meaningfulness of effect sizes in psychological research: Differences between sub-disciplines and the impact of potential biases. *Frontiers in Psychology*, 10:813, 2019.

[31] B. E. van Solingen, R. *A Practical Guide for Quality Improvement and Software Development*. McGraw-Hill International, London/Chicago, 1999.

[32] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012.