An empirical study of power consumption of Web-based communications in mobile phones

Inmaculada Ayala, Mercedes Amor, Lidia Fuentes and Daniel Muñoz

Dpto. de Lenguajes y Ciencias de la Computación

Universidad de Málaga, Andalucía Tech

Málaga, Spain

Email:{ayala,pinilla,lff,danimg}@lcc.uma.es

Abstract—Currently, mobile devices are the most popular pervasive computing device, and they are becoming the primer way for Web access. Energy is a critical resource in such pervasive computing devices, being network communication one of the primary energy consuming operations in mobile apps. Indeed, web-based communication is the most used, but also energy demanding. So, mobile web developers should be aware of how much energy consumes the different web-based communication alternatives. The goal of this paper is to measure and compare the energy consumption of three asynchronous Web-based methods in mobile devices. Our experiments consider three different Web applications models that allow a web server to push data to a browser: Polling, Long Polling and WebSockets. The obtained results are analyzed to get more accurate understanding of the impact in energy consumption of a mobile browser for each of these three methods. The utility of these experiments is to show developers what are the factors that influence the energy consumption when different web-based asynchronous communication is used. With this information mobile web developers could reduce the power consumption of web applications on mobile devices, by selecting the most appropriate method for asynchronous server communication.

I. Introduction

Currently, mobile devices are the most popular pervasive computing devices, which are becoming an essential element of our daily activities. Recently, mobile web usage has overtaken access from desktop for first time [1]. However, network communication is one of the primary energy consuming operations in mobile devices. On average, network communications can consume over 40% or more of the total non-idle state energy of an app [2][3]. Considering that handheld devices accounted for 51.3% of Internet usage worldwide by the end of 2016, a reduction of power consumption by mobile applications is of great importance.

According to Statcounter, most of the traffic of mobile devices is based on Hyper Text Transfer Protocol (HTTP) [4].Among all kinds of network operations, those related with HTTP are the most energy consuming, representing almost 80% of the global network related energy consumption [5]. Therefore, reducing the power expenditure of browser data transfer can have a significant impact on the overall energy consumption of the device. In addition, many of the HTTP interactions generated by the browser are derived from asynchronous HTTP-based interactions. There are many web applications that require the server to send (*push*) data to the client asynchronously as the state of a dynamic system changes, but making this not as a response to a user interaction. Since HTTP is a synchronous request/response protocol and the client (i.e. the browser) always has to initiate a request, there are several approaches that emulate asynchronous communication over HTTP, using a continuous client-originated polling. Recently, WebSockets provide Web developers with a real asynchronous method to manage server push with better performance than simply web-based Polling and Long Polling. But, each of these alternatives consume a different amount of energy, contributing in greater or lesser extent to the battery power draining. So, mobile web developers should be aware of how much energy consumes the different alternatives, for different scenarios.

Our goal is to measure, compare and analyze the energy consumption of three asynchronous Web-based methods in mobile devices using a the goal-question-metrics methodology [6]. The study has focused on Android devices, the most popular operating system for mobile devices in recent years. The experiments have been performed in two different devices (Galaxy Nexus and Nexus 5) and with two energy profiling tools (Green Oracle and Trepn Profiler). We have performed our experiments for three different Web applications models that allow a web server to push data to a browser: Polling and Long Polling, based on HTTP requests and responses, and WebSockets, based on server events. The target audience of our findings are web developers interested in optimizing the energy consumption of their web-based applications, for a concrete usage scenario. By knowing the power impact of each of the asynchronous communication methods, and also the concrete factors that most influence the battery consumption, mobile web developers will take more informed decisions, and select the greenest asynchronous mechanism to push data.

This paper is structured as follows: Section II describes our experimental setup, describing the three web-based methods considered. Section III presents the energy profiling tools used to perform our experiments. The experimental results are presented and analyzed in Section IV, and the threats to validity in Section V. Finally, Section VI presents the related work and Section VII the conclusions.

II. Experimental setup

In this section we describe the three Web based communication methods used by mobile browsers that are taken into
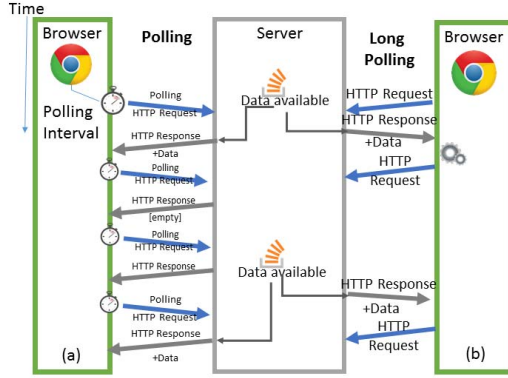
IEEE computer society

Fig. 1: Asynchronous Web Interaction scheme using (a) Polling and (b) Long polling

consideration in our experimental study. As stated in the introduction, we consider polling, long polling and WebSockets.

A. Asynchronous Web-based Communication

Polling and Long polling are based on HTTP, a protocol in which a browser establishes TCP-based connections to a Web server for sending HTTP requests and receiving HTTP responses. Persistent TCP connections are the default behavior of any HTTP/1.1 connection. That is the browser keeps alive the connection for interchanguing one or more request/response, even after HTTP error responses.Persistent HTTP/1.1 connections have several advantages, which can influence positively to energy consumption in mobile devices.

Since the standard HTTP model is synchronous, and a server can not initiate an interaction, it is necessary to implement an asynchronous mechanism between them in order to receive pushed data from the server, .

1) Polling: The polling mechanism is the simplest way to receive asynchronous data. The client polls the server periodically (polling interval) for new content by sending HTTP requests, allowing the server to respond with an HTTP response if new data is available. Each request attempts to *pull* any available data. If no data is available, the server returns an empty response and the client waits for some time (polling interval) before sending another (poll) HTTP request. The polling frequency depends on the latency that the client can tolerate in retrieving updated information from the server. Polling implementation on the client-side relies on features included by default in browsers, such as JavaScript. The basic communication cycle of an application using "HTTP polling" is depicted in the interaction diagram (a) in Fig. 1. However, continuous or short polling can consume significant energy by forcing an HTTP request/response even when no data is available.

2) Long polling: In order to alleviate client continuous polling, there exist different web models in which a long-held HTTP request allows a web server to push data to a browser only when new data is available. One of the most common server push mechanisms is HTTP "Long Polling",

in which the server "holds open" (not immediately reply to) each HTTP request, responding only when there is new data to deliver. Then, there is always a pending request to which the server can reply for the purpose of sending data as it is available, thereby minimizing the latency in message delivery, and the use of processing/network resources.

After receiving an HTTP response, the client sends a new request. The basic communication cycle of "HTTP Long Polling" is shown in Fig. 1 (b).

B. Websocket protocol

WebSocket is a protocol that allows to use the TCP connection between a browser and a Web server as a full-duplex and persistent socket-like channel for interchanging non HTTP messages. It is created on top of TCP and introduces a small overload in comparison to HTTP/1.1 [7]. Based on this connection, the Web server is able to actively send data to the client whenever it is available. Prior to data/message exchange, the WebSocket protocol requires an initial handshake and the message exchange. The initial handshake uses the HTTP-Upgrade-request, which allows to switch from the HTTP to the WebSocket protocol. The message exchange is executed in form of frames, which contain either text or binary data [8]. In WebSockets, TCP connections are persistent. When an update is available, the server sends the new data to the client through the WebSocket. Incoming data is made available to the browser through an event.

III. ENERGY PROFILING TOOL

To measure energy consumption on mobile phones, there are multiple tools based on both hardware and software models [9][10]. Although hardware measurement offers higher precision, selecting and configuring a hardware equipment may represent a complex task, which can introduce additional bias [11]. Some solutions require special equipment, preventing the reproduction of the experiment by third parties with the documentation at disposal [12][9]. Other solutions offer applications that can be easily installed in devices [13] but they are restricted to specific architectures. Other important factor is the validation related to the energy profiling tool. In our case, we found two solutions that suit our requirements: the GreenOracle [14] and the Trepn Profiler [13].

With regard to the equipment, tests have been executed on a Galaxy Nexus, which is the device used to develop the GreenMiner and Nexus 5. The web application is deployed in a Glassfish web server running in a Windows 10 PC connected to a Gigabit ethernet network. On the other hand, mobile phones use WLAN to access the web server.

A. The Green Oracle

GreenOracle is an accurate energy model generated using a big-data approach and hundreds of energy measurements obtained by the GreenMiner [9]. According to authors, GreenOracle has an upper error-bound close to 10% which is similar to other methods like [12]. Additionally, it is easy to apply
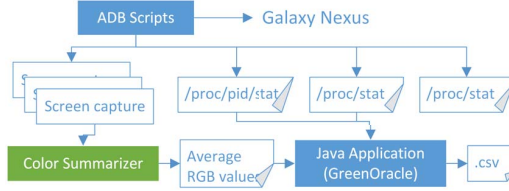
Fig. 2: Framework to apply the GreenOracle Energy Model

as it is based on information that can be extracted from the operating system of the device (i.e. Android).

We have developed several scripts for Android ADB that gather the information of the operating system and interact in an automated way with the mobile phone screen avoiding to introduce additional bias (see Fig. 2). The energy model requires information that is extracted from the operative system files *"/proc/stat"* and *"/proc/pid/stat"*. Other necessary information is the system calls performed as a result of the execution of the application, which is extracted using the *"strace"* command, and average RGB values of colour in the screen during the execution. This information is generated periodically by capturing the screen during application execution and by processing captures using the program Image Color Summarizer [15]. All this information is processed by a Java Application, which contains the GreenOracle model, and generates csv files with the energy lectures.

B. Trepn Profiler

Trepn Profiler is a commercial tool able to profile the work of most of Android devices but specially intended to profile Qualcomm's Snapdrago devices. This tool can be directly installed from Google Play and offers a great variety of information like CPU or energy consumption.

In order to profile the energy consumption, Trepn Profiler requires hardware instrumentation which is only at disposal in some devices. Mainly, the Mobile Development Platform, which collects current readings from different hardware components.According to Qualcomm [16], the accuracy of Trepn Profiler is between 2.1% and 5.5% compared with the Moonsoon Power Monitor [10].

Trepn Profiler supports the automation of tests using external ADB scripts. This functionality allows us to work with Trepn Profiler in a similar way as we work with the Green Oracle (see Fig. 2). So, we have developed an ADB script that setups the tool, launches the application, starts the monitoring process and downloads the csv files that contains the energy consumption information.

IV. EXPERIMENTAL RESULTS

In this section we present the experimental planning and the energy profile results. The measurement unit in our experiments along this work is joules (J).

A. Objectives and research questions

The methodology of this study is defined according to the goal-question-metrics approach [6] as follows: "Analyze

asynchronous HTTP-based communications in Android, from the point of view of web software developers. To achieve this goal we set the following research questions (RQs):

RQ1. What is the factor (polling/pushing interval or data size) that most influences energy consumption in asynchronous communication? This question explores the influence of data size and polling/pushing interval in the energy consumption of the three considered methods. In addition, the data availability in relation with polling interval is also considered, because in polling and Long Polling mechanisms it can cause the reception of empty HTTP responses.

RQ2. Which asynchronous communication method is the most efficient in terms of energy consumption? This question overviews whether the energy expenditure of each asynchronous method is significantly different. Also, we explore if a method is better than others for different scenarios. This information is crucial to advice web developers to make a better decision when selecting the asynchronous method.

B. Data collection

In order to compare the consumption of the three communication mechanisms, we have developed a web application composed of a simple set of JavaScript components. Each JavaScript has a counterpart in the server side for each communication mechanism. The client side allows to configure the experiment with different polling/pushing periods and data size, and has a button to start the experiment. Our experiments focus on data reception, so we have measured the energy consumption of the mobile browser during 1 minute receiving data using the three methods in different scenarios with different data sizes and periods. We have selected 1 minute because according to different works [17] the usual user interaction with mobile phones is for short periods of time of around 1 minute. Regarding the data size, we consider five message sizes (80 bytes, 160 bytes, 512 bytes, 1024 bytes and 3000 bytes) that comes from a usual text message (i.e. 80 bytes) to messages that requires fragmentation at the IP level (i.e. 3000 bytes). The update periods selected for the experiments are (in milliseconds) 1000ms, 2000ms, 5000ms, 15000ms and 30000ms. These update periods represent the polling frequency for the Polling mechanism, and the frequency at which there is new data available for Long Polling and WebSockets. Each test has been repeated 20 times and results have an standard deviation lower than 5J.

According to our experiments (see Fig. 3), results for WebSockets and Long Polling are quite similar. Additionally, it is evident that for larger periods of time (columns 15000 and 30000 milliseconds), the power consumption of the three communication mechanisms seems remarkably similar.

One of the advantages of Long Polling and WebSocket compared to polling is that they are asynchronous in practice, so the exchange of information mainly occurs when there are new information at disposal. This is not the case of polling that requests and receives information continuously with a fixed frequency. In order to measure the added value of Long Polling and WebSocket, we have designed experiments with random
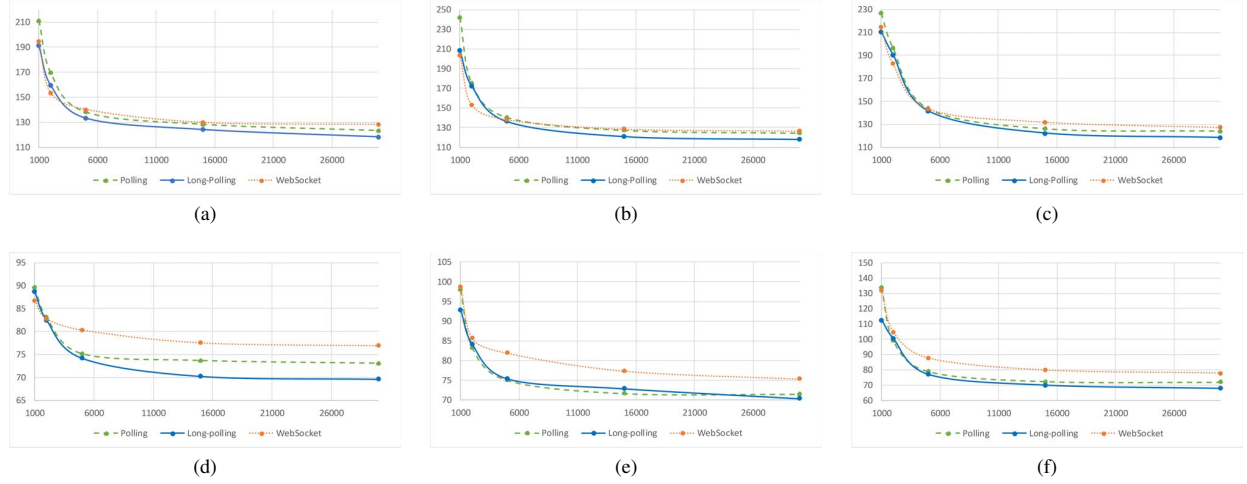
Fig. 3: Energy consumption (in Joules) for different polling periods (in milliseconds) in Galaxy Nexus (top row) and Nexus 5 (bottom row) for messages of 80 bytes ((a) and (d)), 512 bytes ((b) and (e)) and 3000 bytes ((c) and (f)).

traffic. In these experiments, there is new information available of random size (from 80 bytes to 3000 bytes) and in random time (from 500ms to 15000ms). For these experiments, we have measured the energy consumption of the mobile browser during 1 minute and repeated the test 20 times. For Galaxy Nexus, the mean of the results is 119J for Long Polling and 124J for WebSockets. For Nexus 5, means of the energy consumption are 70J for Long Polling and 78J for WebSockets.

C. Answers to research questions

RQ1. What is the factor (polling/pushing interval or data size) that most influences energy consumption in asynchronous communication? In order to answer this question, we apply a classical statistical method, the multiple linear regression. The multiple linear regression tries to model the relationship between two or more explanatory variables and a response variable by fitting a linear equation to the observed data. In our case, having two explanatory variables, we must fit the equation $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$, where y represents the energy consumption, x_1 the polling period, x_2 the data size and β_0, β_1 and β_2 are constants that will be determined by the multiple regression method. The relative importance of the explanatory variables will be given by the values of β_1 and β_2, the one with a higher absolute value determines the stronger explanatory variable. In order to perform this analysis, input and output data must be standardized. This standardization consists in subtracting the mean and dividing by the standard deviation the collected data.

Results of the multiple linear regression analysis are depicted in Table I. The validity of the analysis is supported by the coefficient of determination R^2, which is higher than $0, 4$. The absolute value of β_2 is higher than β_1 in all experiments. Therefore, according to our experiments, the polling interval is the factor that has more influence on energy consumption. We can confirm this fact in charts of Fig. 3. On the one hand, for

TABLE I: Results of multiple linear regression analysis for Galaxy Nexus and Nexus 5.

		β_1	β_2	R^2
Galaxy Nexus	Polling	0.0593	-0.703	0.4978
	Long Polling	0.0983	-0.76139	0.5893
	WebSocket	0.1856	-0.6639	0.4752
Nexus 5	Polling	0.3227	-0.577	0.4378
	Long Polling	0.2648	-0.7006	0.561
	WebSocket	0.4376	-0.5861	0.5351

longer polling periods, the energy consumption is around 110-130J for Galaxy Nexus and 70-80J for Nexus 5, regardless of the size of the message. The same situation arises for shorter polling periods, with a consumption which is between 190-240J for Galaxy Nexus and 85-135J for Nexus 5.

RQ2. Which asynchronous communication method is the most efficient in terms of energy consumption? According to our results (see Fig. 3), there is no single asynchronous mechanism clearly better in all the situations. In general, Polling is the communication mechanism with the highest energy consumption but for longer polling periods (i.e. 15000ms and 30000ms) its consumption is very similar to the other two and lower than WebSocket.

This result is coherent with the active processing that the browser has to perform for small periods. For instance, if 80 bytes of data is available every 2000ms, the Polling client, setting a polling interval of 1500ms, sends: 40 HTTP requests of 462 bytes (an overload of 7760 bytes). For the same scenario, the client using Long Polling sends 30 HTTP requests of 462 bytes. The WebSocket client does not send any request nor data to the server. The number of messages sent explains the higher energy consumption. Significant differences in energy consumption between the three communication mechanisms just happen in shorter polling periods (i.e. 1000ms, 2000ms and 5000ms). For these periods, WebSocket is the greenest

for 80 bytes, but when the message size increases Long Polling consumes less energy. Random tests have confirmed the similarities between Long Polling and WebSockets (see Subsection IV-B). The energy consumption of Long Polling is just slightly lower than the energy consumption of WebSocket (119 vs. 124 for Galaxy Nexus and 70 vs. 78 for Nexus 5).

Taking into account the results, our answer is that the energy consumption of Long Polling and WebSocket is very similar for the experiments performed. However, for small messages, WebSockets has lower energy consumption, while for bigger messages Long Polling shows better results. When the frequency of available data is high but the messages sent are small, Long Polling introduces an overload derived from the processing and sending of HTTP requests, which consumes more energy. Surprisingly, this is not maintained for longer periods. The reason is the WebSocket protocol sends periodically signalling data to keep the connection opened if it is not used, and the energy consumption of this interaction is similar to the energy consumption caused by sending HTTP requests of Long Polling. So, surely many developers may think that WebSockets, being a more recent technology will consume less than a older one, but we have found that the beliefs of these web developers do not correspond to the reality, showing the great utility of this kind of experiments.

V. Threats to validity

In this section, we briefly discuss the internal validity and external validity of our study. The internal validity intends to explore if the energy results are influenced or not by other factors. While, the external validity analyses if the data obtained in the experiments can be generalized or not.

With regard to the internal validity, we should analyse how precise are the obtained results. We have chosen two different software measuring tools, GreenOracle and Trepn Profiler, instead of using hardware solutions, which usually have more precision. As we stated in Section III, the difficulties of reproducing experiments made by hardware solutions by third parties and the precision demonstrated by these tools are the main reasons to select software solutions. Additionally, we are not interested in reporting absolute energy values, but to give recommendations to developers based on comparative results.

We also have analysed if the energy consumption measurements of the communication mechanisms could be influenced by how we have implemented the web applications used in the experiments. In order to mitigate this threat, we have based our implementations in minimal examples provided by tutorials from Netbeans and university courses. Moreover, we have detected that the user interface can be an important source of energy consumption, so we have used the same user interface for the three mechanisms.

Another internal threat can be caused by the set of parameters that we have considered in our experiments for answering the questions is not exhaustive. According to the literature, the main factors that affect energy consumption are the polling period and the amount of information transmitted. The values selected illustrate minimal interactions such as the exchange of a text message in a chat (80 bytes) and more complex cases that even require fragmentation at the IP level (3000 bytes). So, we think we have covered a great variety of situations, but developers must know that the conclusions raised in this paper can be considered valid only for interactions with the mobile phone of 1 minute of duration. Exploring the influence of this parameter is part of our future work.

Regarding the external validity, we should have taken into consideration the influence of using a concrete mobile browser in the experiments. There are only two mobile browsers supporting the three asynchronous mechanisms of this study, Google Chrome and Mozilla Firefox. We are aware that the use of one browser or other can affect to the energy consumption of the device. So, we have opted to use in our experiments the most used one, Google Chrome. In any case, we plan to use Mozilla Firefox in future experiments.

Finally, we consider as an external threat the generalization of the results to all mobile phones and Android versions. Here the limitation is to have reliable energy measurement tools available for enough devices. To mitigate this we have opted for two measuring tools (GreenOracle and Trepn Profiler) for two devices with different versions of Android (4.3 in Galaxy Nexus and 6.0.1 in Nexus 5).

VI. Related Work

As most of the mobile apps transfer data over the Internet, the energy consumption in mobile devices can be studied at different layers. At the network access layer, the work [18] analyze the energy consumption of data of different communication components like Bluetooth, WLAN, 2G, and 3G. This study (for a Nokia N95), concludes that using 3G is more energy consuming than using GSM (2G), when using different application and services requiring the data connection.

The work in [19] provides a comparison between WLAN and 3G with regards to their energy consumption, showing that using WiFi 3G is more energy efficient than 3G. This study also shows how the network activities directly affect the energy consumption and battery life. At the application level, in [20] and [21], the energy consumption of using two data interchange formats (JSON, XML) is compared. The comparison analyzes them considering the processing speed, overhead and energy consumption. Results indicate that JSON format shows better performance in battery management. The work [21] also tests the use of binary protocol buffers, reporting that is recommendable for big data volumes, because it shows a better energy management than protocols for raw data.

Close to our study, different works analyse the energy consumption of data transfer in mobile browsers. In [22], WebSocket and AJAX are measured with regards to their energy consumption and performance for 3D graphic renderings in the browser. In this context the analysis of energy measurements shows that using WebSockets can reduce energy consumption but significantly drop the QoE in devices with slow CPU.

A comparison between WebSockets and Ajax (which uses Long Polling), both using a 3G connection, is done in [23]. In this study it is concluded that battery life can be extended by

falling back to AJAX as long as the interval between AJAX messages is sufficiently large to allow the use of UMTS.If the interval between data messages is smaller, then the Websocket method consumes less battery.

The work in [24] also compares WebSocket and HTTP in the Internet of Things in a WLAN. This work studies the influences of data size and the transfer frequency in the energy consumption, reporting that with WebSocket we can save around 5% energy for a high number of requests per unit of time. Surprisingly, the observed energy savings obtained with WebSocket comparing to HTTP is not so high. The work in [25] analyses REST/HTTP and WebSocket with regards to their energy consumption in a mobile phone using different access network technologies (Edge, 3G and WLAN). REST uses Long Polling with non-persistent connections. It reports that the use of WebSockets consumes less energy than the use of Long Polling/REST, arguing that the reason is not the overhead of the HTTP-protocol, but the use of non-persistent connections, which consumes more energy than using a HTTP/1.1 persistent connection, most used nowadays.

VII. Conclusions

The goal of our study was to compare HTTP-based asynchronous communication and WebSocket based on their energy consumption. The research approach goal-question-metrics was used to analyze the different influencing factors and their effect on the energy consumption.

The answers to the research questions RQ1 and RQ2 in section IV can be summarized as: **RQ1**. Based on the results obtained, the effect of the amount of data transferred is low, which brings us to the assumption that the data overload introduced by HTTP headers size has no effect on the energy consumption. However, in nearly all experiments, data frequency influences negatively on the energy consumption, which shows that, on the basis of using a persistent connection, the difference of energy consumption between the three mechanisms is observable. This difference tends to dismiss as frequency decrease (data is received less frequently). Regarding **RQ2**, Long Polling is more efficient in terms of energy consumption. However, for small messages, WebSockets is the greenest one, while for large messages Long Polling shows better results. While this difference is bigger when polling is made more frequently, it tends to decrease when data is received less frequently.

We have obtained similar results in our experiments using different energy profiling tools (Green Oracle or Trepn profiler), which is the first step to generalize our findings.

References

[1] StatCounter Global Stats. (2016) Mobile and tablet internet usage exceeds desktop for first time worldwide. [Online]. Available: http://gs.statcounter.com/press/2016

[2] D. Li, S. Hao, J. Gui, and W. G. J. Halfond, "An empirical study of the energy consumption of android applications," in *IEEE ICSME*, Sept 2014, pp. 121–130.

[3] M. Tawalbeh, A. Eardley, and L. Tawalbeh, "Studying the energy consumption in mobile devices," *Procedia Computer Science*, vol. 94, pp. 183 – 189, 2016.

[4] R. T. Fielding, J. Gettys, J. C. Mogul, H. F. Nielsen, L. Masinter, P. J. Leach, and T. Berners-Lee, "Hypertext transfer protocol – http/1.1," Internet Requests for Comments, RFC Editor, RFC 2616, June 1999.

[5] D. Li, Y. Lyu, J. Gui, and W. G. J. Halfond, "Automated energy optimization of http requests for mobile applications," in *Proc. of the 38th ICSE*. New York, NY, USA: ACM, 2016, pp. 249–260.

[6] V. R. Basili, "Software modeling and measurement: The goal/question/metric paradigm," University of Maryland at College Park, College Park, MD, USA, Tech. Rep., 1992.

[7] V. Pimentel and B. G. Nickerson, "Communicating and displaying real-time data with websocket," *IEEE Internet Computing*, vol. 16, no. 4, pp. 45–53, July 2012.

[8] I. Fette and A. Melnikov, "The websocket protocol," Internet Requests for Comments, RFC Editor, RFC 6455, December 2011.

[9] A. Hindle, A. Wilson, K. Rasmussen, E. J. Barlow, J. C. Campbell, and S. Romansky, "Greenminer: A hardware based mining software repositories software energy consumption framework," in *Proc. of the 11th Working Conference on Mining Software Repositories*. New York, NY, USA: ACM, 2014, pp. 12–21.

[10] "Power Monitor," Apr. 2017. [Online]. Available: https://www.msoon.com/LabEquipment/PowerMonitor/

[11] A. Hindle, "Green software engineering: The curse of methodology," in *IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, vol. 5, March 2016, pp. 46–55.

[12] D. Li, S. Hao, W. G. J. Halfond, and R. Govindan, "Calculating source line level energy information for android applications," in *Proc. of the 2013 International Symposium on Software Testing and Analysis*. New York, NY, USA: ACM, 2013, pp. 78–89.

[13] "Trepn Power Profiler A product of Qualcomm Technologies, Inc." Apr. 2017. [Online]. Available: https://developer.qualcomm.com/software/trepn-power-profiler

[14] S. A. Chowdhury, S. Gil, S. Romansky, and A. Hindle, "Did i make a mistake? finding the impact of code change on energy regression." PeerJ Preprints, Tech. Rep. 5:e2419v3, 2016.

[15] "Image color summarizer, RGB, HSV, LCH & Lab image color statistics and clusteringsimple and easy," Apr. 2017. [Online]. Available: http://mkweb.bcgsc.ca/color-summarizer/

[16] "Forum Qualcomm," May 2017. [Online]. Available: https://developer.qualcomm.com/forum/qdn-forums/software/trepn-power-profiler/32991

[17] N. van Berkel, C. Luo, T. Anagnostopoulos, D. Ferreira, J. Goncalves, S. Hosio, and V. Kostakos, "A systematic assessment of smartphone usage gaps," in *Proc. of the CHI Conference on Human Factors in Computing Systems*. NY, USA: ACM, 2016, pp. 4711–4721.

[18] G. P. Perrucci, F. H. P. Fitzek, and J. Widmer, "Survey on energy consumption entities on the smartphone platform." in *VTC Spring*. IEEE, 2011, pp. 1–6.

[19] G. Metri, A. Agrawal, R. Peri, and W. Shi, "What is eating up battery life on my smartphone: A case study." in *ICEAC*. IEEE, 2012, pp. 1–6.

[20] N. Nurseitov, M. Paulson, R. Reynolds, and C. Izurieta, "Comparison of JSON and XML Data Interchange Formats: A Case Study," in *Proc. of the 22nd ISCA*, 2009, pp. 157–162.

[21] B. Gil and P. Trezentos, "Impacts of data interchange formats on energy consumption and performance in smartphones," in *Proc. of the 2011 Workshop on Open Source and Design of Communication*. New York, NY, USA: ACM, 2011, pp. 1–6.

[22] K. Kapetanakis and S. Panagiotakis, "Evaluation of techniques for web 3d graphics animation on portable devices," in *International Conference on Telecommunications and Multimedia, 2012, Heraklion, Crete, Greece, July 30 - August 1, 2012*, 2012, pp. 152–157.

[23] B. D. Mandyam and N. Ehsan, "Mobile Systems IV," World Wide Web Consortium, Tech. Rep., 2012.

[24] G. Bovet and H. Jean, "Communicating With Things - An Energy Consumption Analysis," in *Pervasive 2012*, United Kingdom, Jun. 2012.

[25] V. Herwig, R. Fischer, and P. Braun, "Assessment of REST and web-socket in regards to their energy consumption for mobile applications," in *IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, 2015, pp. 342–347.