Responsiveness, Energy Efficiency, and Data Usage: The Future of Mobile Web Applications

Alen Luković

June 2, 2017

1 Introduction

During the last decade, smartphones have become many people's favorite choice of electronic device allowing their users to connect to and participate in the world wide web. Checking news, posting and updating one's social media accounts, even writing messages and calling each other are only very few of many possible examples manifesting today's mobile web's importance and our dependence on it. However, with our dependence growing, the mobile web and its applications providing the most possible easy interfaces guaranteeing access to it naturally has to face growing challenges. With mobile web traffic increasing and, at least in some places in the world, exceeding desktop web traffic[BYRZ16], several concerns regarding its responsiveness, energy efficiency, and data usage (often shortened as "RED")[BYRZ16] have to be raised. This particular work aims at answering the question which challenges the mobile web regarding RED currently faces and is going to face by briefly explaining these concepts, examining their contemporary importance, and elaborating shortly on possible solutions.

2 Responsiveness

Whenever a web browser – regardless if it is a desktop or mobile browser – tries to display a web page with all its features, it has to load, fetch, and evaluate objects, such as HTML, CSS, JavaScript, and images. These objects may consist of other objects the web browser has to load, fetch, and evaluate, and so on – evidently, loading a web page is, put simply, resolving a dependency graph. However, most web pages do not give away their dependency graphs to the web browser completely. Therefore, in order to not violate any dependency constraint the web browser may not know about, conservative assumptions regarding these dependencies are made, resulting in web pages being displayed more slowly[NGMB16]. Furthermore, in many places in the world, connecting to the world wide web with a smartphone and its mobile version of desktop web browsers results in bigger delays, since, in addition to resolving a web page's dependency graph, the data transfer manifests itself to be slower. Conclusively,

if slow connections to the world wide web should not contribute additionally to the time a web page needs in order to be displayed, elaborating on these dependency graphs and their resolution seems reasonable.

In order to understand any possible improvements regarding responsiveness, understanding fundamentally how a web page is loaded is essential. First of all, the web browser parses the HTML tags and constructs a so called "Document Object Model" (DOM) tree, with each node representing an HTML tag, and the tree representing the hierarchical structure of the HTML tags. Then, the web browser constructs a so called "render tree", consisting of all DOM nodes which are going to be displayed. A so called "layout tree" for which the previously constructed render tree functions as its foundation is additionally constructed in order to determine the web page's layout. Finally, the layout tree is traversed and the web page is displayed.[NGMB16]

Difficulties may occur whenever CSS and JavaScript objects in particular have to be loaded, fetched, and evaluated. While CSS may only block the construction of render and layout trees just for a moment, JavaScript is capable of blocking the construction of DOM trees, since JavaScript may execute writing operations upon the current HTML file. Therefore, because the web browser knows that the future HTML file may differ from the current HTML file, it precautionally blocks the parsing of the HTML file, consequently resulting in bigger delays. In general, blocking is assumed to be necessary whenever a certain state of uncertainty may be reached. However, these uncertainties turn out to be less constrictive than the web browser believes they are.[NGMB16] Since introducing the ideas how to take advantage of these circumstances in detail goes beyond this work's scope, I am going to introduce existing solutions and present their results shortly.

"Scout", a measurement infrastructure, is capable of automatically tracking fine–grained and less obvious data dependencies. It rewrites HTML files and JavaScript codes in order to create more accurate dependency graphs. Therefore, when the web browser tries to display the according web page, it will succeed faster because of the new dependency graph. When tested, the amount of fine–grained dependencies, represented by additional edges in a graph, increased by 29.8% at the median, while 86.6% of all web pages had less so called "slack nodes" which are irrelevant and solely contribute to bigger delays. Another contribution, "Polaris", uses Scout's new dependency graphs and performs dynamic observations of the network's condition in order to determine when to load certain objects in order to optimize web page loading time. Results show that Polaris is capable of reducing web page loading time by 34% at the median.[NGMB16]

3 Energy Efficiency

As obvious as before, energy efficiency's importance regarding mobile web applications must not be underestimated, for a smartphone's battery ultimately determines whether a mobile web application is actually viable or not. However, energy efficiency and its concerns have mostly been delegated onto hardware and operating system manufacturers. Therefore, introducing solutions contributing to energy efficiency on programming language level seems to represent a direction filled with lots of potential.[ZR16]

One of the fundamental ideas with which energy efficiency on programming language level is possible is the introduction of so called "user Quality-of-Service (QoS) expectations".[ZR16] User QoS expectations mean that certain expectations users have regarding the quality of whatever mobile web application they currently use should be regarded and somehow integrated into the development of software. However, similar to responsiveness, elaborating on this idea in detail goes beyond this work's scope. Therefore, I am only going to hint on certain aspects of this particular approach, and, finally, present some research results.

Put simply, user QoS expectations enable the software developer to influence his or her mobile web application's energy consumption depending on whether the application's user, for example, taps his or her touchscreen, moves his or her finger across it, or does nothing at all. Depending on the action of the application's user, certain expectations regarding energy efficiency may be fueled and applied, and in addition to particular programming language extensions, software developers are enabled to contribute to their software's most possibly efficient energy management. When tested, some researchers even managed to save approximately 29.2% to 66.0% energy in comparison to Android's "Interactive" governor.[ZR16]

4 Data Usage

Even though it may not seem evident at first, data usage already has a certain importance, and is going to have an even much bigger one in the future. In emerging markets, where mobile web traffic has already exceeded desktop web traffic[BYRZ16], but access to the world wide web is not as spread as it is, for example, in most of Europe and North America, seemingly insurmountable obstacles, such as high costs of data and desktop versions of web pages becoming inefficient when displayed on smartphones and other mobile devices, prevents the population from unrestricted access to the world wide web. Therefore, compressing data efficiently in order to simplify data transfer and reduce its costs is obviously going to result in more and more people being able to participate in the world wide web.[ABC+15]

In order to elaborate on the fundamental ideas within this work's scope, I briefly

present "Flywheel", an HTTP proxy server which aims at compressing data as efficient as possible in order to decrease the amount of data transfered noticeably. Developed by Google, Flywheel is embedded within Google Chrome auomatically. On average, world wide web content passing through the proxy server is compressed by 58%, and 50% by the median. Images behave extraordinarily, since they are compressed to approximately ⅓ of their original size (66%) on average, being also responsible for 85% of the data size benefit. Other significant data size reductions are observable regarding CSS (52%), JavaScript (41%), HTML (38%), and plain text (20%), on overage.[ABC+15]

5 Conclusion

RED, its concepts, and its current results seem very promising regarding the future of mobile web applications. Responsiveness is going to make web pages easier accessible, energy efficiency is going to make mobile web applications more viable, and data usage is not only going to contribute further to responsiveness, but also make the world wide web accessible to and, most important, more affordable for people in emerging markets who have not yet participated in the world wide web at all. Last, but not least, introducing solutions regarding RED's concerns is definitely going to affect the world's economy heavily. For example, an one second delay when loading a web page costs Amazon $1.6 billion in annual sales lost[BYRZ16], and energy inefficiency is one of the main reasons for negative application reviews, with 55% of mobile web application users claiming to uninstall an application which uses too much energy[ZR16]. Additionally burdened with economic aspects, introducing and developing RED should be one of the main concerns of software developers, for it is going to affect our use of the world wide web sustainably.

References

[ABC+15] Victor Agababov, Michael Buettner, Victor Chudnovsky, Mark Cogan, Ben Greenstein, Shane McDaniel, Michael Piatek, Colin Scott, Matt Welsh, and Bolian Yin. Flywheel: Google's data compression proxy for the mobile web. In *NSDI*, volume 15, pages 367–380, 2015.

[BYRZ16] Peter Bailis, Jean Yang, Vijay Janapa Reddi, and Yuhao Zhu. Research for practice: web security and mobile web computing. *Communications of the ACM*, 60(1):50–53, 2016.

[NGMB16] Ravi Netravali, Ameesh Goyal, James Mickens, and Hari Balakrishnan. Polaris: Faster page loads using fine-grained dependency tracking. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*. USENIX Association, 2016.

[ZR16] Yuhao Zhu and Vijay Janapa Reddi. Greenweb: language extensions for energy-efficient mobile web computing. In *Proceedings of the 37th*

ACM SIGPLAN Conference on Programming Language Design and Implementation, pages 145–160. ACM, 2016.