Analysis of the Energy Consumption of JavaScript Based Mobile Web Applications

Antti P. Miettinen[1] and Jukka K. Nurminen[1,2]

[1] Nokia Research Center, Helsinki, Finland
{antti.p.miettinen,jukka.k.nurminen}@nokia.com
[2] Aalto University School of Science and Technology, Finland

Abstract. JavaScript is the dominant language of modern web applications. In this research, we have investigated the battery-consumption of JavaScript applications running on mobile phones. In our empirical study, we developed and analysed eight implementations of the same application using different JavaScript libraries. The results show that there are significant differences between different implementations. There is no single factor explaining the performance differences. Furthermore, the performance of different libraries is strongly affected by the communication technology (3G or WLAN). The long latencies that 3G introduces suggest that increasing the parallel execution of server queries has potential for energy and speed improvements.

Keywords: Energy-efficiency, mobile services, JavaScript, web-applications.

1 Introduction

Recently the emphasis of software development has shifted from applications to services. Instead of installing specific software, the end user devices access services residing on the Internet servers. The shift that started on the PC side is gradually also happening with mobile phones and can extend to other lightweight wireless devices.

In many respects the mobile devices, such as modern smart phones, resemble PC devices. They are more limited in resources but the difference is not very big: modern smart phone has roughly the same CPU power, amount of RAM memory, and file storage capacity as PC computers less than ten years ago. However, a fundamental difference between PC computers and mobile devices is sensitivity to energy consumption. Because a mobile device has only intermittent access to electricity grid, it is essential that it is able to run hours, preferably days, without recharging. The progress on battery technology is slow; the annual growth of battery capacity is around 5% [1]. This is much less than the rapid growth of CPU power, memory capacity, and communication bandwidth. Therefore, energy-efficiency is of primary concern for the design of wireless devices and applications, and is likely to remain so also in the future.

JavaScript is an important building block for modern web applications and a key component of the AJAX paradigm [2], which enables the development of interactive and dynamic web services. Initially, JavaScript was developed to add dynamic effects to web pages with small code snippets. Since JavaScript was originally not intended for large-scale programming, its use to implement complex web applications of today is not trivial for the developers. The developers work is further complicated by the fact that different browsers require slightly different JavaScript code sequences for the same tasks.

One way to make JavaScript programming easier is to use JavaScript libraries which package widely used functionality into reusable modules and, at the same time, isolate the developer from the differences of the underlying platforms. As there are a number of libraries available, it is not easy for the developer to choose which one is most applicable for each task. The problem is even more difficult for the mobile web service developer since there is hardly any information about how the different libraries work on mobile devices.

In this research, we investigate the JavaScript libraries and their use on mobile devices. In particular, we focus on the energy consumption of different libraries. Does the JavaScript library selection influence the energy consumption of the mobile web service? Which libraries enable energy-efficient application development? What are the essential reasons for the differences?

These issues have immediate importance for current mobile phones and applications targeted for them. However, similar issues are likely to be important also for other lightweight wireless devices, many of which could be even more sensitive to battery consumption.

It would be nice if the same languages and tools could be used to develop applications for highly heterogeneous devices. This would save the training costs of the developers, reduce the need of different implementations for different devices, and allow the progress made on web development to be applied to a variety of devices in a short time span. Finding out if this is possible and, especially, if battery consumption is a problem is one of the contributions of this research.

Although the battery consumption is often mentioned as a problem for mobile AJAX applications [3-5], there has been little prior research on JavaScript energy-efficiency. To our knowledge, our research is the first one to analyze systematically the energy-consumption of an AJAX application.

The rest of the paper is structured as follows. Section 2 reviews related research. Section 3 describes our empirical experiment where eight developers implemented the same web application using different libraries. Section 4 presents the measurement results of the performance of these applications. Section 5 discusses and analyses the results. Finally, section 6 concludes the paper and presents ideas for further research.

2 Related Research

Hansen [6] and Walton [7] have investigated the browser power consumption on PC hardware. Hansen's work is more driven by the "green thinking" that is how much electricity is needed to run web applications. On mobile computing, this approach is also relevant but of lesser importance than the inconvenience that frequent recharging causes to the user. Hansen looked at the effect of different elements of a web page to energy consumption. His finding was that Flash banner ad was the most harmful for energy consumption but all client side scripting technologies, JavaScript, Java, VBScript and Silverlight, tend to increase the energy consumption. Walton compares different browsers on three different processors. His results show that there are no major differences in energy consumption between popular browsers (with the exception of Safari 4, which is consistently worse than others on all platforms).

Wei [8] has investigated the performance of JavaScript in different browsers on PC environment. His results show that different browsers have different performance bottlenecks although there are some common problems.

Other studies focus on JavaScript execution performance without explicit analysis on energy consumption. Sounder [9] has investigated the JavaScript more from the developer perspective. He comes up with a set of rules that make JavaScript more effective (Frontend performance best practices). Kiciman and Livshits [10] have analyzed JavaScript behavior and developed an end-user tool to give visibility to a web applications performance.

Pervilä's research [5] is one the few studies that explore the use of AJAX applications on mobile devices. The analysis is mainly focused on how the AJAX application work on mobile environment as well as on the execution speeds. He notes the relevance of battery consumption and analyzes it briefly. He does not measure energy on a detail level but uses the frequency of battery recharging as the unit for energy consumption.

Some studies compare different ways to implement the same web application functionality. Smullen and Smullen [11] compare two implementations of a university student information system; one based on HTML and one based on AJAX. AJAX provides significant performance improvements (40-70%) but only when appropriate caching is in use. In a worst-case scenario it performed worse (10-15%) than the traditional HTML application. Xie and Parsons [12] compare the use of an AJAX application and an Active Server Pages (ASP) application with identical functionality in GPRS networks. There results are mobile Ajax over GPRS can reduce the bandwidth requirement by about 70% and cut the server's response time in half.

3 Research Methodology

In order to understand how the performance of JavaScript libraries influences the battery consumption of a web application we implemented the same task with different libraries. Different implementations were done by computer science students at Aalto University who were either at the final year of their master studies or at post-graduate level. The test group thus presented technically skilled developers who had experience in programming with various languages. However, most of the persons did not have prior experience with JavaScript development.

3.1 Task Specification

The students were first taught how to develop web applications and write JavaScript code. Then they were given the following task.

Develop an application that allows a person to see if friends are nearby. Your application should have the following display. Center person, Location, and Distance are editable fields. Distance to friends should be populated by those friends who are within the specified distance from the Center person.

Center person: Peter
Location 12345 6789
Distance: 5 km

Distance to friends
John 3 km
Paul 4 km
Mary 5 km

The application should support three tasks

1. *Select the center person from a listbox*
2. *Change the distance*
3. *Change the location of friend (with another browser) to see that the friend list is able to detect the change and update itself. Note that this step requires advanced AJAX concept "long poll".*

When the above changes are made, the rest of the page should be updated automatically using the AJAX techniques.

Each student was free to choose the JavaScript library to suit his taste. The following libraries were used:

- MooTools http://mootools.net/
- Dojo http://www.dojotoolkit.org/
- ExtCore http://www.extjs.com/products/extcore/
- Prototype http://www.prototypejs.org/
- YUI (two implementations) http://developer.yahoo.com/yui/
- jQuery http://jquery.com/
- GWT http://code.google.com/webtoolkit/

Note that GWT, Google Web Toolkit, differs from the other libraries in the respect than instead of writing JavaScript code and calling the library functions from the code, the GWT developer writes Java code, which the system compiles into JavaScript.

The same server side implementation hosted at a university server was used by all applications. The server side was based on Apache and python code and was used via a REST interface (http://maps.cs.hut.fi/cloud/index.php/Server_API).

3.2 Measurement Setup

For the measurements, we used Nokia N900 mobile phone with the built-in browser. The measurements were done with Nokia Energy Profiler, which is a software-only analysis tool for Nokia mobile phones available at Forum Nokia (www.forum.nokia.com). In addition to power consumption, it is able to measure downlink and uplink bitrates, CPU load, memory usage, and some other attributes.

In our measurements, we performed the following steps:

- Initial page load (no center person specified)
- Select a person e.g. Peter
- Change distance from 5 (default) to 2
- Move John from x to y with another browser

For each step we used the Nokia Energy Profiler to record the execution time, CPU load, average power consumption, and the number of bytes uploaded and downloaded.

We used Elisa cellular operator for the 3G measurements. The measurements were done in residential suburban area, where HSDPA (3.5G) functionality was available. For WLAN measurements, we used a 802.11g access point, which was connected to the public Internet via ADSL.

We repeated each measurement five times to compensate the inherent performance variability of live networks that arise e.g. from varying signal quality, activities of other users, and other similar external causes that are beyond our control.

4 Results

Figure 1 shows the cumulative energy consumption, amount of downloaded and uploaded data, CPU usage, and execution time for the complete use case. To get these values we measured each step separately five times and summed up the averages of each step. Note that if in a real case the steps were executed tightly one after another the results might differ a bit from our measurements because the "hot start" for the later steps could influence the results. The data in Figure 1a is from the case when the mobile phone was using 3G and the data in figure 1b is from using wireless LAN (802.11g) and ADSL to communicate with the server. The unit on the y-axis presents the energy (Joules) while the heights of the bars that represent other attributes are relative values that allow comparisons between different libraries and carrier technologies.

It is clearly visible from Figure 1 that there are differences between different libraries. Cumulative energy, amount of data, CPU usage, and execution time vary considerably from one implementation to the other. As expected WLAN was clearly more energy efficient and had a faster execution time than 3G. However, the other dependencies are more complicated. There is no clear correlation between energy and execution time, energy and amount of data, or energy and CPU usage. It seems that the energy consumption is dependent on these attributes in a non-trivial fashion.

Some observations are easy to derive. First, for GWT the additional complexity arising from the automatically generated JavaScript code clearly increases the amount of data transferred. However, the larger amount of data does not directly translate into increased energy consumption. GWT is one of the most energy hungry implementations but the difference in the amount of transferred data is far bigger than the difference in energy consumption in comparison with the other implementations.

Figure 2 shows the energy consumption for each step of the test case. It can be seen that there is a lot of variability in the charts. For MooTools the intial step was dominant in 3G while in WLAN it is on par with most of the libraries. This cannot be explained only by large data size because in Prototype much more data transfer is happening.

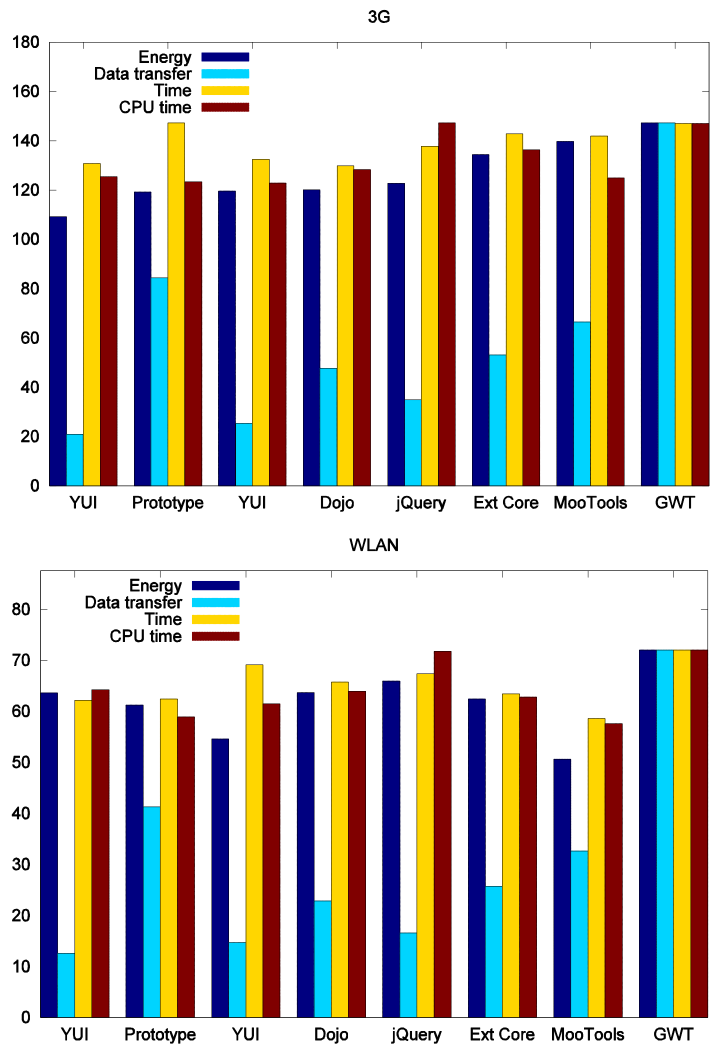

Fig. 1. The cumulative energy consumption, amount of downloaded and uploaded data, CPU usage, and execution time for the complete use with a) 3G and b) WLAN

The relative energy consumption of different steps seems to vary between the libraries. For many libraries, the differences are so small that they can be attributed to measurement error.

With 3G for many libraries the “Change distance” step was the dominant energy consumer while with WLAN the role of “Move friend” and “Select” are with many libraries the most significant ones.

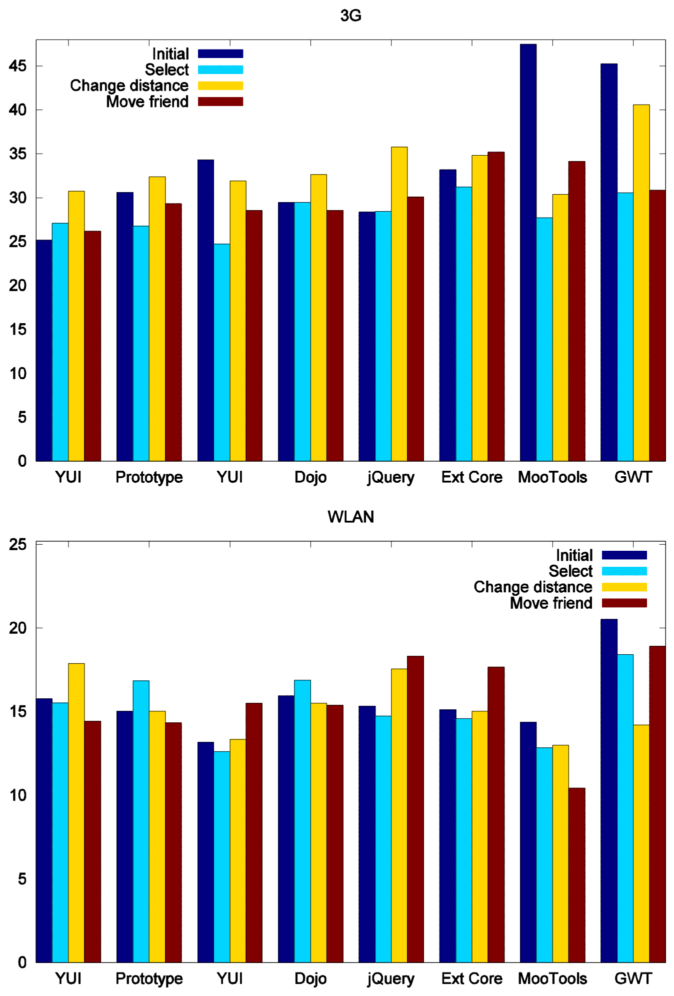

Fig. 2. Energy consumption split to the different tasks with a) 3G and b) WLAN

To understand the reasons for the performance differences we plotted the data transfer as a function of execution time in Figure 3. This example analyzes the MooTools implementation but the behavior of the other implementations is similar with minor variations. The dotted (green) curves show the downlink bitrate (in kilobytes per second) while the solid (red) curve shows the same for the uplink.

An obvious difference is the higher bitrate that WLAN is able to reach, which is about double the 3G data transfer. However, since the amount of data to be transferred is quite small, there is no significant difference in the execution time of the active data transfer. However, the long latency of 3G is a far more important component. The arrow in Figure 3 a) shows the time when the first GET request is issued. After that, it takes almost two seconds before the download starts. In WLAN case Figure 3 c) the download starts almost immediately after the GET request has been issued.

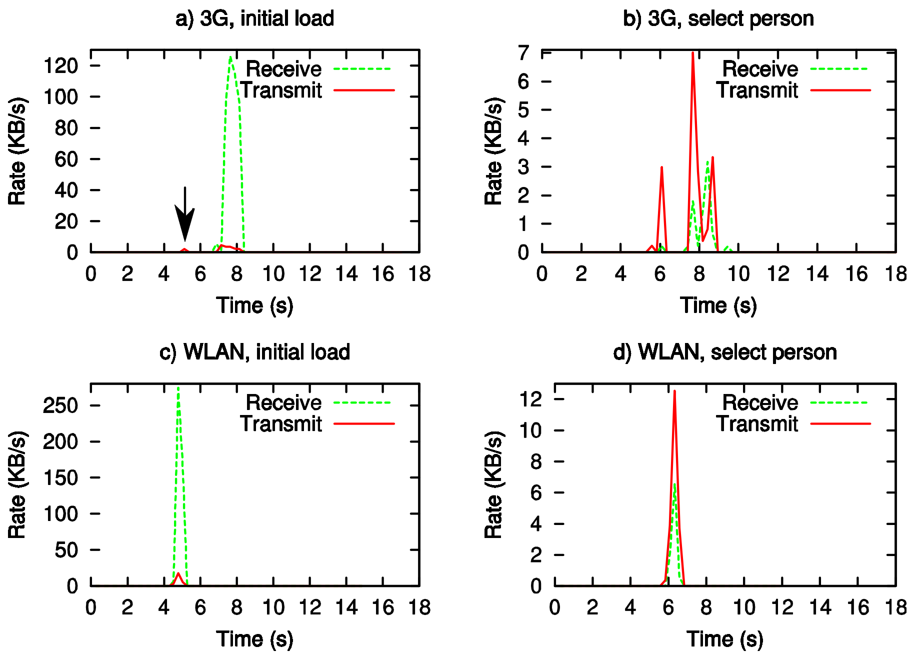

Fig. 3. MooTools data transfer charts for the first two steps, initial load (a&c) and select person (b&d)

This effect of latency becomes even clearer when the use case is more complicated. The “Select person” task plotted in Figure 3 b) and d) consists of five GET requests. These requests are executed in three phases so that the execution does not progress before each GET request of a phase has been completed and the server has returned their results. Because of the long latency, these requests are clearly separate in Figure 3 b) while in the WLAN case (Figure 3 d) they seem to merge to a single data transfer. The key lesson from this is that the more parallel execution of http requests the code has, the better the performance especially with high latency networks like 3G.

To analyze the sensitivity of our results, we performed measurements with an additional device, the Nokia E71 phone. Our primary test device Nokia N900 is a high-performance multimedia oriented mobile computer with a touch display while E71 is a traditional smart phone with a smaller display and more limited resources. Another difference between N900 and E71 is that the former is based on the Maemo Linux operating system while the latter uses Symbian. Figure 4 plots the correlation of the execution time and energy consumption with WLAN communication. It shows the measurements (averages ± error margins) with the two devices. The red measurements are with Nokia N900 and the green measurements are with Nokia E71. It is clear from the figure that there is strong dependency between execution time and energy spending.

This dependency is the clearest explaining factor for most of the energy differences in this study. During the execution, energy is spent on different hardware modules of the mobile phone. For instance, if the display is on during the execution then a

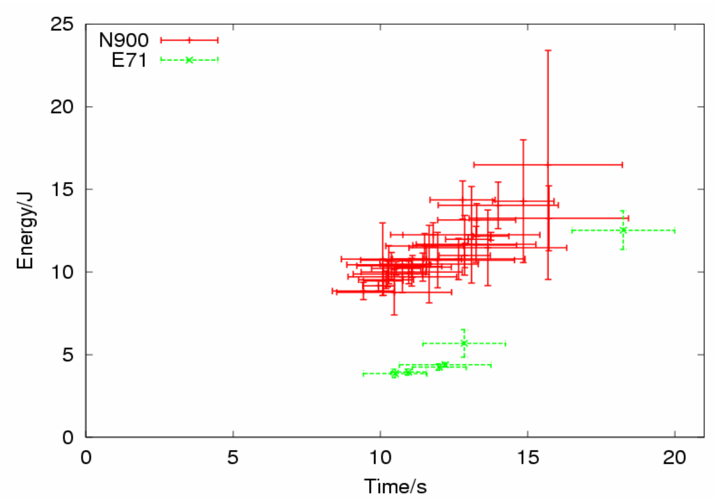

Fig. 4. Correlation between execution time and energy consumption

device with a larger display (N900) consumes more energy than a device with a smaller display. Memory and CPU are other components that are constantly drawing power during the execution.

5 Discussion

The results indicate that the library selection is relevant for web application energy-efficiency. With 3G, the difference between the most power hungry (GWT) and the least power hungry implementations was around 20%. With WLAN, the difference was 30%. Interestingly the least power hungry implementation was different for 3G and WLAN; with 3G it was YUI and with WLAN MooTools.

However, the energy difference is small in comparison to the difference in the amount of transferred data. The biggest implementation (GWT) required about five times more data transfer than the most compact one (YUI).

Although YUI was both the most compact and most energy-efficient implementation for 3G, it is wrong to draw the conclusion that the energy consumption differences can be explained by the transferred amounts of data alone. E.g. the data transferred by jQuery was about half in comparison to Prototype while the energy consumptions of the two implementations were about the same.

GWT is a special case compared to other libraries. While with other libraries the developers write JavaScript code, a GWT application is developed with Java and the JavaScript code used by the application is the result of a compilation process. The generality of this approach seems to result into large JavaScript code size. Especially in case of 3G the larger amount of data transfer is also reflected in energy consumption.

Execution time seems to be the attribute that best explains the differences in energy consumption between different implementations. Execution speed is not only good for energy consumption but also for user experience in the form of better response time.

However, the results do not rule out the effect of other factors although their influence is smaller. The simple conclusion is to use libraries with fast execution speeds. The good news is that the library developer when optimizing the speed of a library is at the same time optimizing the energy-efficiency as well.

Long latency is a major factor differentiating 3G from WLAN cases. Therefore finding ways to live with the long latency would benefit both the execution time and the energy consumption. The developers should be familiar with the issue and consider how to maximize the parallelism of server requests when developing web applications. However, it can be hard to realize because the typical development environment for mobile web applications is a PC, which normally has a high-speed, low-latency connectivity. Therefore, the issue only becomes visible when the web application is tested on a mobile device. A simple alternative is that development tools would allow the developers to experiement how their applications work when the latency increases. Another alternative would be to build automatic support for maximal parallel execution into the library or middleware code. The system would need to buffer the separate GET or POST requests and send them as a single entity to the server. Similar mechanism could also be useful at the server side. However, implementing such a mechanism could be difficult because detecting the dependencies between GET and POST calls automatically can be complicated.

When drawing conclusions based on this study it is important to keep in mind two limitations of our approach. First, it is impossible to filter out the individual differences in development style between different developers. However, as the task was relatively simple, there are no major algorithmic differences between the different implementations. Moreover, the two implementations with the YUI library have very similar performance, which gives some limited evidence that the differences between developers do not have major effect on the result.

Secondly, the application is very simple compared to many AJAX applications in use today. Do the results of this small-scale study scale when the application size increases? It is difficult to give a definite answer to this question. Typically, the larger applications consist of smaller modules. If the results of this study apply to a typical module then it is to be expected that they also apply for aggregates of such modules.

A further observation of this study is that all of the libraries work on mobile devices. This indicates that the mobile browsers have matured and that the resources of mobile devices are adequate for relatively complex tasks. It is important to note that none of the tested libraries has been targeted for mobile devices. In this respect, the difference between mobile handheld devices and PC computers is getting blurrier. However, the number of library functions needed for the test application is quite limited so it can be that for applications that are more complicated some of the libraries would experience problems.

Although the functional capability of the browsers on mobile devices is almost equal to browsers running on PC environment, important differences still exist. The usability of the application with a small display without mouse still needs improvement. Data transfer and rendering speeds could be further enhanced to make the response times faster. However, in this research we did not focus on these aspects as our target was to understand what the web application developer can do to create efficient applications.

6 Conclusions

The energy-efficiency of JavaScript based web applications is important for mobile phones and other lightweight devices. Battery-consumption is not only a device or hardware R&D topic but it can be influenced by application developers. In particular, in this research we have investigated how the web application developer can influence the battery consumption of the target devices of his application.

Although battery consumption is important for applications for mobile phones and other lightweight devices, the amount of prior research on the area is limited. In particular, as far as we are able to find out, this is the first empirical study about how JavaScript developer can influence the battery consumption of the target device.

Our research shows that there are relevant differences between different ways to implement JavaScript based application from the energy-efficiency point of view. By choosing a proper library, the user is able to save up to 30% of energy in the implementation of the same functionality.

In this experimental study, we have compared eight implementations of the same task with different JavaScript libraries. We have shown that the size of the code alone does not explain the difference in energy consumption. However, further research would be useful to verify and to extend our results via more controlled experiements. Moreover, better understanding of the reasons for the performance differences could be used to develop new libraries that would also consider the energy-efficiency dimension.

Experimentation with larger scale solutions would also be useful. Naturally, the bigger the software size the harder it is to create multiple implementation for comparison purposes. Moreover, in case of more complicated software the differences between design decisions that different developers make become more dominant making the comparison harder.

Acknowledgements. We are grateful to the participants of the Seminar on Distributed Systems at Aalto University in fall 2009 for the implementation of the test applications.

References

1. Robinson, S.: Cellphone Energy Gap: Desperately Seeking Solutions. Strategy Analytics (2009)
2. Garrett, J.: Ajax: A New Approach to Web Applications (2005), `http://www.adaptivepath.com/publications/essays/archives/000385.php`
3. Kunito, G.: Issues for mobile Ajax for cellular users (2007), `http://www.w3.org/2007/06/mobile-ajax/papers/docom`
4. Van De Walle, D., Goeminne, N., Gielen, F., Van De Walle, R.: Challenges for Mobile Gaming based on AJAX (2007), `http://www.w3.org/2007/06/mobile-ajax/papers/mobix`
5. Pervilä, M.: Performance of AJAX applications on mobile devices. MSc thesis, University of Helsinki (2008)

6. Hansen, R.: Browser Power Consumptions (2008), `http://www.sectheory.com/browser-power-consumption.htm`
7. Walton, J.: Browser Face-Off: Battery Life Explored (2009), `http://anandtech.com/mobile/showdoc.aspx?i=3636`
8. Wei, C.: A Study of Ajax Performance Issues (2008), `http://www.coachwei.com/blog/_archives/2008/1/22/3480119.html`
9. Souders, S.: High-performance web sites. ACM Commun. 51(12), 36–41 (2008)
10. Kiciman, E., Livshits, B.: AjaxScope: a platform for remotely monitoring the client-side behavior of web 2.0 applications. SIGOPS Oper. Syst. Rev. 41(6), 17–30 (2007)
11. Smullen III, C.W., Smullen, S.A.: Modeling AJAX Application Performance. In: 2nd IASTED International Conference on Web Technologies, Applications, and Services (2006)
12. Xie, F., Parsons, D.: Measuring Ajax Performance on a GPRS Mobile Platform (2008), `http://www.massey.ac.nz/~dpparson/Xie_Parsons_APIS7.pdf`