Green Mining: Energy Consumption of Advertisement Blocking Methods

Kent Rasmussen
Department of Computing Science
University of Alberta
Edmonton, Canada
krasmuss@ualberta.ca

Alex Wilson
Department of Computing Science
University of Alberta
Edmonton, Canada
aewilson@ualberta.ca

Abram Hindle
Department of Computing Science
University of Alberta
Edmonton, Canada
hindle1@ualberta.ca

ABSTRACT

Extending battery life on mobile devices has become an important topic recently due to the increasing frequency of smartphone adoption. A primary component of smart phone energy consumption is the apps that run on these devices. Many apps have embedded advertising and web browser apps will show ads that are embedded on webpages. Other researchers have found that advertising libraries and advertisements tend to increase power usage. But is the converse true? If we use advertisement blocking software will we consume less energy, or will the overhead of ad-blocking consume more energy?

This study seeks to determine the effects of advertisements on energy consumption, and the effects of attempts to block the advertisements. We compared different methods of blocking advertisements on an Android mobile phone platform and compared the power efficiency of these methods. We found many cases where ad-blocking software or methods resulted in increased power use.

1. MOTIVATION

Recently, smartphones and other mobile devices have seen a huge spike in popularity. Between May 2011 and May 2013, an additional 20% of U.S. adults acquired a smartphone, resulting in more than half of American adults now owning a cellphone [13]. This means many consumers are bound to mobile devices that are limited by the energy capacity of their batteries.

There are a variety of android applications that attempt to save battery life, some of which have been downloaded over 10 million times [5]! Within online forums, one can find many tips and suggestions for improving battery life, but most of these anecdotes are not empirically validated. Thus we do not know if these suggestions are useful until we measure the effect of following a particular suggestion.

One such claim, which this paper will address, is that advertising has an effect on battery life and installing ad-blocking software will save battery life. Under a heading "General Knowledge" on BSDgeek_Jake's post on the xda-developers web forum, it is stated that "this mod will save you CPU processing used in advertisement contents, in-app ads, hidden app call-back home routines, malware/spyware sites, in-app user data collection, ... and precious bandwidth resulting [in] extended battery life..." [2]. Such claims are verifiable and quantifiable, but there is no empirical evidence supporting these claims. This paper aims to provide empirical evidence to the claims of advertisement blocking increasing power efficiency on mobile devices.

2. BACKGROUND & RELATED WORK

While there is a lot of work on measuring and estimating software energy consumption [8, 16, 7, 11, 6], we will focus primarily on work that deals with advertising on mobile devices.

Some very relevant work has been done by R.J.G. Simons and A. Pras on determining the energy consumption of advertisements on web browsers using desktop computers running a Windows operating system [12]. They determined that there was a 3.4% reduction in energy consumption overall when ad-blocking methods were applied.

Jarred Walton from Anandtech compared laptop run time using different browsers, but also included one test where the AdBlock Plus plugin was installed in conjunction with Firefox. From the tests, it was discovered that on two laptops battery life was extended by 7 minutes (4.3%) and 10 minutes (4.7%) using AdBlock Plus while on another laptop the battery life went down by a minute ($< 0.1\%$) [15].

Prashanth Mohan, Suman Nath and Oriana Riva used Windows phones to investigate whether or not an advertiser could monetarily afford prefetching mobile advertisements and serving them locally at a later time in order to save battery life [9]. While doing so, they discovered that advertisements used 23%, on average, of the applications' total energy consumption when simulated on a cell-phone network.

Vallina-Rodriguez et al. [14] determined that a major cause of advertisement energy consumption was the complex network requests being made. They made recommendations to improve advertisement's network use in a more power-aware manner.

3. METHODOLOGY

Our experiment relied on accurately measuring the energy consumption of our device and for the tests to be consistent

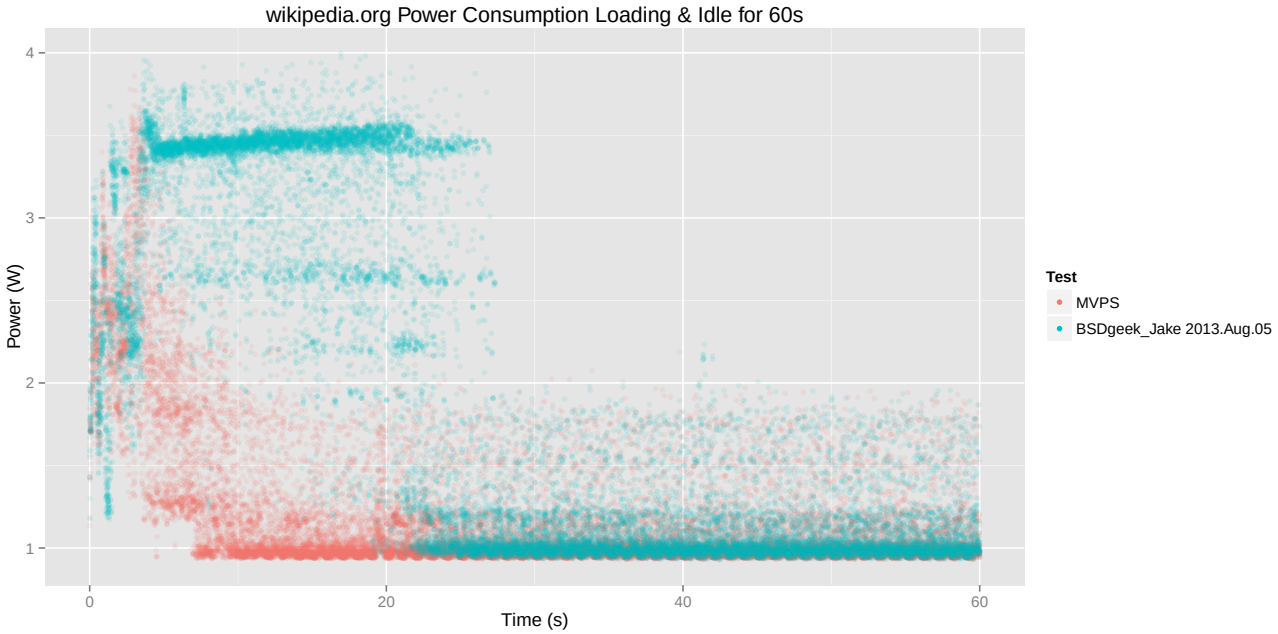

Figure 1: Power usage for all runs of MVPS (non-modified) and BSD_Geek Jake's August 5th, 2013 on wikipedia.org

and replicable across runs. The major components of our experiment were all off-the-shelf hardware, and include a Samsung Galaxy Nexus phone, a YiHua YH-305D power supply, a Raspberry Pi model B computer, an Arduino Uno and an Adafruit INA219 breakout board.

We did not test the claims of hosts files reducing security or privacy.

3.1 Power Monitoring

In this section we describe our power monitoring setup that measured the power use of an android phone while it ran tests.

A YiHua YH-305D power supply was set up to output a constant voltage of 4.1V. The voltage was routed through an Adafruit INA219 breakout board, which then went to the Samsung Galaxy Nexus phone, which had wire soldered to the pins where the battery would have been attached.

The INA219 reported the voltage and amperage that the phone was receiving to an Arduino Uno over I2C, which then passed the readings along over a serial USB connection to a Raspberry Pi that recorded the data. The INA219 was set up to internally average 32 readings, which resulted in 50 data points per second consisting of a voltage with a resolution of 0.01V and an amperage with a resolution of 0.1 mA. The INA219 was also set up to read voltages between 0 and 16V at an amperage range of approximately 1.3 A.

3.2 Software Setup

3.2.1 Raspberry Pi

The Raspberry Pi was set up with Raspbian and a version of Android Debug Bridge that was compiled for the ARM CPU architechture, and Python 3.3 with pySerial. The Raspberry Pi was responsible for launching the testing scripts on the phone through ADB shell, as well as controlling the USB communication power, through a transistor switch on the Arduino to toggle USB connectivity and power on and off between the Raspberry Pi and the Galaxy Nexus.

3.2.2 Android & Samsung Galaxy Nexus

The Samsung Galaxy Nexus "maguro" phone we used was flashed with an Android 4.2.2 (JDQ39) factory ROM image from Google. The software on the phone was then modified to enable root access and all applications that could be disabled through the in-phone settings were disabled. The phone was set up in airplane mode with wi-fi re-enabled afterward. This disabled the cellular radio, bluetooth and NFC. The phone was connected to a WPA secured wireless N network broadcasting from the same room to grant the device internet access for our tests.

Some custom software was pushed to the phone including busybox, sqlite3 and three of our own custom scripts. When USB connectivity was turned off between the Raspberry Pi and the phone, busybox's 'nohup' was used in order to allow the testing script to continue. Before and after each test, to ensure settings were changed and reverted, sqlite3 was used to retrieve all the phone settings as this was much faster than using Android's built in 'settings' command.

Initially Android's built-in 'monkey' application exerciser was used to run tests on the phone; however, it proved unreliable and introduced variability and errors into our results. We switched to running a shell script on the phone to automate our tests instead, which improved reliability for the test runs.

The three pieces of software that we developed and installed on the phone were to improve reliability and ease of use. The first piece of software we installed was called 'microtime', which simply returned the system time of the Galaxy Nexus with microseconds. This was needed as An-

droid's built in 'date' command only returned seconds, and the alternative time syncing method, using '/proc/uptime' only has resolution up to 10 milliseconds. With our change we could reliably obtain the time of the Android device to 1 millisecond. Our second piece of software was 'microsleep', which simply made the script sleep for that amount of microseconds. This was designed to give more control for timing as Android 4.2.2's built-in 'sleep' only has a resolution of 1 second (Android 4.3's 'sleep' commmand now allows for greater resolution than 1 second).

The last piece of software we installed was a small program called 'tapnswipe' which allowed us to write directly to the /dev/input/event# device file in order to inject touchscreen taps and swipes for navigating the UI of applications. We wrote our own, as Android's built in 'input' command takes approximately 0.7 seconds to launch and perform a tap, and ours brought that down to approximately 0.07 seconds, which helped speed up tests.

3.3 Adblock Methods

For our tests we only used methods that could be deployed directly on the phone. In addition, we chose the hosts file and AdBlock plus plugin methods as these are already deployed on a variety of mobile devices by a variety of users. There are applications on Android to automatically download and apply hosts files from a variety of sources. There is also an Android application AdBlock Plus that is separate from Firefox; however, we did not use this application in our testing.

3.3.1 Hosts Files

Host files provide hostname look-up shortcuts for a computer. One can provide hostname to IP addresses bindings that alleviate the need to execute a DNS name look-up with an actual DNS server. Hosts files work by a user defining a list of IP addresses and aliases for each IP address in a file located at `/etc/hosts` in many linux systems, including our Android 4.2.2 phone. Adding an entry, for example `127.0.0.1 google.com` would route any TCP/UDP requests to the host named google.com to the localhost internally on the phone. Since there is no web-server set up on the phone, the website simply appears unreachable. Our tests replace the phone's /etc/hosts file with the different hosts files depending on which test is being run.

For our hosts file tests we chose a popular hosts file MVPS [10] which has around 15,000 entries and another hosts file from a popular post on the xda-developers forum by user BSDgeek_Jake [2]. The MVPS host file used was the July 8th version. For the MVPS hosts file we modified the hosts file and made two additional tests, one where the hosts file was stripped of all newlines and comments (MVPS (Trimmed)), and another where multiple websites were put into one line at a maximum of 1024 characters (MVPS (Multiple)). For BSDgeek_Jake's hosts file, we used two different versions. The August 5, 2013 version has approximately 185,000 entries, while the July 8, 2013 version has approximately 283,000 entries.

In addition to the hosts files, entries in the hosts files for all tests, including our control, were added for some Mozilla and Firefox domains in order to prevent Firefox from sending data or checking for updates.

3.3.2 AdBlock Plus Plugin

AdBlock Plus blocks advertisements by using custom filters that are then transformed to regular expressions. It can download and apply filter lists such as EasyList [4] to perform advertisement blocking on a variety of webpages. We install AdBlock and give it ample time to update and apply the filter lists.

Firefox 22.0 for Android supports the AdBlock Plus plugin directly in the browser and the plugin method is recommended over using the AdBlock Plus application as per AdBlock Plus' developers [1].

We used AdBlock Plus 2.3 with the EasyList, Malware Domains, EasyPrivacy and Fanboy's Social Blocking List filters [1]. In addition, we also disabled the default setting 'allow some non-intrusive advertising'. We controlled for the same filters and settings across versions although we did not control for the change of subscription filters across our testing.

3.4 Testing Procedure

The 100 websites in our tests were determined from a list of the estimated top 1 million U.S. websites from Quantcast, accessed on August 14th, 2013 [3]. Quantcast has some websites that have hidden their domain from the list, so the top 100 non-hidden websites were chosen. The URLs were loaded in order of descending popularity, as determined by Quantcast.

Before and after each test run, the correct hosts file and Firefox application was installed and removed from the phone. Power measurement samples were taken from the moment USB power was disconnected between the computer and the phone until they were reconnected. At the very start of the test, the device time is recorded. During the test, our script recorded the phone's current time to a file inside the phone's storage. The difference in these times was used to determine the start of various partitions in our logged power measurements. We recorded the time right after tapping 'enter' to start loading the webpage, right before the script executed its command to sleep for 60s. After 60s elapsed, a command then recorded another time and the next URL was entered, and the process repeated for each website in the test. The same browser tab in Firefox was used for all 100 websites.

4. RESULTS & DISCUSSION

Test runs were aggregated, averaged and compared against each other using 2-sided paired t-tests paired by website. Each test (all 100 webpages) was run a total of 8 times over a span of 1 week (August 16th, 2013 to August 23, 2013). We isolated the energy consumption to only that of loading & idling the webpage, ignoring the energy consumption used when loading the Firefox application, setting up ad-blocking, entering the URLs and waiting for the test to end.

This testing resulted in approximately 19.2 million data points being aggregated into groups based on webpages and tests. We averaged the energy consumption of all of the runs of each unique test and webpage, then performed a 2-sided paired t-test, paired by website, between the different tests as shown in Figure 3.

Figure 1 shows an example of MVPS host files power use measured over time, with BSDgeek_Jake's host file overlayed on top in blue. One can clearly see that for the first 20 seconds of the test that BSDgeek_Jake's host file runs consume far more power than the MVPS host files. After 20 seconds both tests look similar.

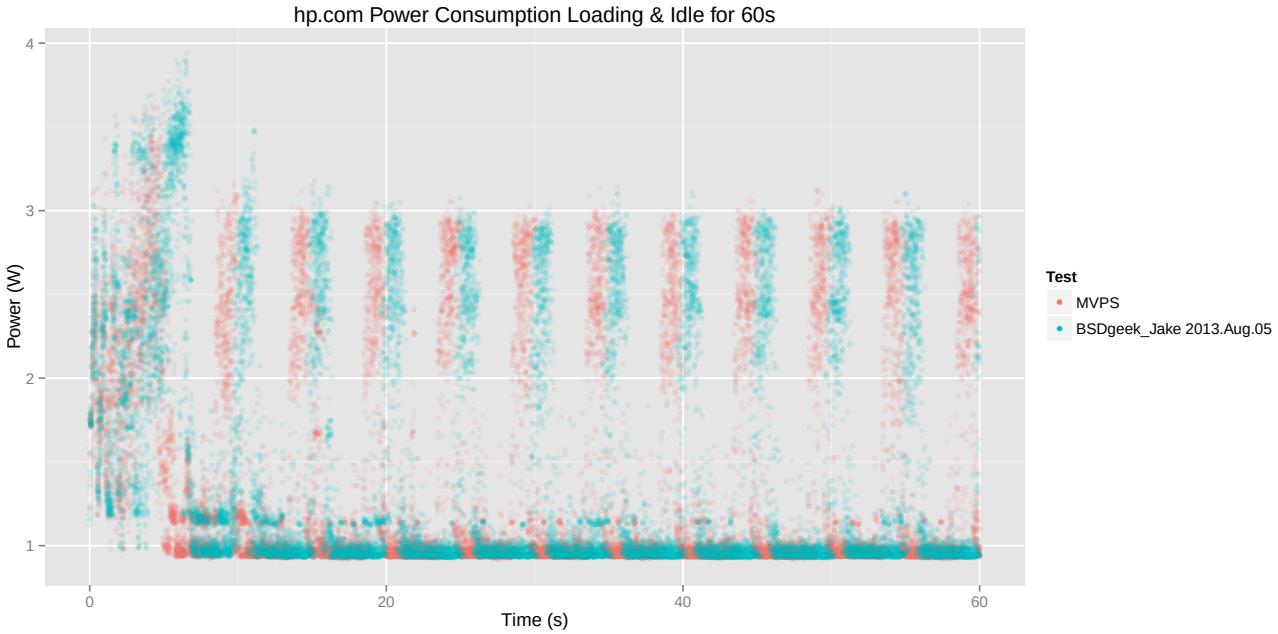

Figure 2: In this run from hp.com, the offset between load times between MVPS and BSDgeek_Jake's hosts files is noticeable.

Compared to our control (None) where no ad-blocking method was installed, 5 out of 7 tests showed an improvement in energy consumption. The MVPS hosts file and our 2 modifications both showed improvements in energy consumption. The original unmodified MVPS hosts file and the MVPS (Trimmed) version showed no significant difference in energy consumption compared to each other, while the multiple-per-line hosts file, MVPS (Multiple), was significantly less efficient than the original and trimmed versions. The two AdBlock plus plugin tests were also significantly better than the non-ad-blocking control. The two AdBlock plus plugin tests showed no significant different in energy consumption compared to each other.

The remaining 2 tests that showed a decrease in power efficiency were the two different versions of BSDgeek_Jake's host files. The August 5, 2013 was significantly better than the July 8, 2013 version.

The least efficient hosts file, BSDgeek_Jake's July 8th, 2013 hosts file, took an additional 0.124 watts of power compared to the most efficient hosts file, the MVPS hosts file when comparing all 100 webpages. This 0.124 watt difference equates to a 10.8% difference in energy consumption. Unfortunately, BSDgeek_Jake's host file also blocked 5 out of the 100 websites tested. Removing these five websites and comparing the other 95 non-blocked websites the power difference becomes 0.143 watts, or 12.5%.

Figure 3 shows that these differences between the control and the ad-blocking software and hosts files were usually different from not-ad-blocking, except for the case of BSDGeek_Jake's August 05 2013 host file, which was not significantly different statistically after correction for multiple hypotheses.

Figure 4 shows the confidence intervals of the differences between test runs. By following the None row (the third row) we can see that MVPS and AdBlock are more efficient than not ad-blocking, while the BSDGeek host files are less efficient than ad-blocking.

The method which showed the most improvement was the MVPS unmodified and trimmed hosts files, which had a energy consumption improvement of 0.046 watts, or 3.8% over no ad-blocking method.

4.1 Security

We did not test any claims of hosts files or plugins improving society and privacy, however it is important to note that the hosts file can reroute domain names to any IP and could be used to hijack websites. As such, it is important to consider the trustworthiness of the websites and the sources for which these lists are obtained from. AdBlock plus is less affected by this as it does not have any control over system-wide domain name resolution and no longer does typo correction.

4.2 Observations

During testing we came across a few specific observations.

4.2.1 AdBlock Plus Plugin

AdBlock Plus uses a more advanced method of blocking elements on webpages, which can substantially alter the look of webapges. AdBlock Plus can also block textual advertisements on webpages such as the ones found on `google.com`.

Our tests were run on a Galaxy Nexus phone and the AMOLED display it contains will consume different power depending on what is being displayed [8]. The AdBlock plus plugin could remove elements from the page, increasing the total amount of white or black which could in turn increase or decrease the power usage of the webpage being displayed. Host files might produce this issue as well since `iframes` might be displayed bright or dark on a webpage.

4.2.2 BSDgeek_Jake's Hosts File

BSDgeek_Jake's hosts file blocked 5 out of the 100 top websites we tested. These websites were `bizrate.com`, `bleacherreport.com`, `break.com`, `goodreads.com` and `yelp.com`.

Additionally, BSDgeek_Jake's hosts file increased page loading time on some webpages, and the increase was consistent across runs. An example is depicted in Figure 2. This shows how the tests for reading and idling on the hp.com webpage were delayed by the use of this host file. The delay is evident in the figure because the fulfillment of the requests (blue dots) using BSDGeek_Jake's show up later in time.

We argue the performance of this hosts file indicates that modifications could be made to the Linux Kernel that Android uses that could improve the hosts file look-up via better caching and hashing. This caching might reduce CPU usage and thus power use on the phone when looking up domain names.

5. THREATS TO VALIDITY

This test only measured energy consumption of loading and idling websites for 60s at a time in Firefox 22.0. We also only measured energy consumption of 100 websites [3], although we did our best to choose a sample that many users might consider visiting.

We did not control for the AdBlock plus filter changes, which could have possibly changed power usage over the testing period.

6. CONCLUSIONS

We tested ad-blocking browser plugins and ad-blocking hosts files on physical Android devices. We found most of these ad-blocking solutions reduce power use, but some are overzealous and do not.

The best improvement in energy consumption for the ad-blocking methods tested was 3.8%, when the MVPS hosts file was applied. All ad-blocking methods do not improve energy consumption however, as indicated by BSDgeek_Jake's July 8th and August 5th host files consuming 6.5% and 2.2% more power, respectively. The host files' performance expose possible Linux kernel domain name look-up performance issues that could be optimized.

Our recommendations for improving mobile power use via ad-blocking include:

- Optimizing the hosts file further by removing entries that are not likely to be shown and sorting the most used websites early on in the list.
- Optimizing the kernel to resolve domains more efficiently in the presence of large host files.
- Purchasing ad-free versions of apps such that ad-blocking hosts files are not necessary.
- Measuring energy consumption before optimization in order to validate if any optimization has been achieved.

In the future we plan to measure the effect of the ad-blocking and hosts file ad-blocking on apps that use advertising network libraries.

Acknowledgments

Abram Hindle is supported by an NSERC Discovery Grant.

7. REFERENCES

[1] Adblock Plus. Adblock Plus for Android FAQ, August 22 2013. https://adblockplus.org/en/android-faq.

[2] BSDgeek_Jake. Mother of all ad-blocking (12 august 2013) blocks malware spyware bloatware, August 22 2013. http://forum.xda-developers.com/showthread.php?t=1916098.

[3] Q. Corporation. Quantcast - top ranking international websites, August 14 2013. https://www.quantcast.com/top-sites.

[4] fanboy, MonztA, Famlam, and Khrin. The official easylist website, August 28 2013. https://easylist.adblockplus.org/en/.

[5] Google. Juice defender - android apps on google play, August 22 2013. https://play.google.com/store/apps/details?id=com.latedroid.juicedefender&hl=en.

[6] S. Hao, D. Li, W. G. J. Halfond, and R. Govindan. Estimating Mobile Application Energy Consumption using Program Analysis. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 92–101, Piscataway, NJ, USA, 2013. IEEE Press.

[7] A. Hindle. Green Mining: A Methodology of Relating Software Change to Power Consumption. In *MSR*, pages 78–87, 2012.

[8] R. Mittal, A. Kansal, and R. Chandra. Empowering developers to estimate app energy consumption. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, pages 317–328. ACM, August 2012.

[9] P. Mohan, S. Nath, and O. Riva. Prefetching mobile ads: Can advertising systems afford it? In *Eurosys '13*, April 2013.

[10] MVPS. Blocking unwanted connections with a hosts file, August 22 2013. http://winhelp2002.mvps.org/hosts.htm.

[11] A. Pathak, Y. C. Hu, and M. Zhang. Where is the Energy Spent inside My App?: Fine Grained Energy Accounting on Smartphones with Eprof. In *Proceedings of the 7th ACM european conference on Computer Systems*, EuroSys '12, pages 29–42, New York, NY, USA, 2012. ACM.

[12] R. Simons and A. Pras. The hidden energy cost of web advertising. June 2010. http://eprints.eemcs.utwente.nl/18066/01/Hidden-Energy-Costs.pdf.

[13] A. Smith. Smartphone ownership 2013. June 5 2013. http://pewinternet.org/Reports/2013/Smartphone-Ownership-2013/Findings.aspx.

[14] N. Vallina-Rodriguez, J. Shah, A. Finamore, Y. Grunenberger, H. Haddadi, K. Papagiannaki, and J. Cowcroft. Breaking for Commercials: Characterizing Mobile Advertising. In *IMC'12*, November 2012.

[15] J. Walton. Browser face-off: Battery life explored, September 11 2009. http://anandtech.com/show/2834.

[16] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang. Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones. In *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, CODES/ISSS '10, pages 105–114, New York, NY, USA, 2010. ACM.

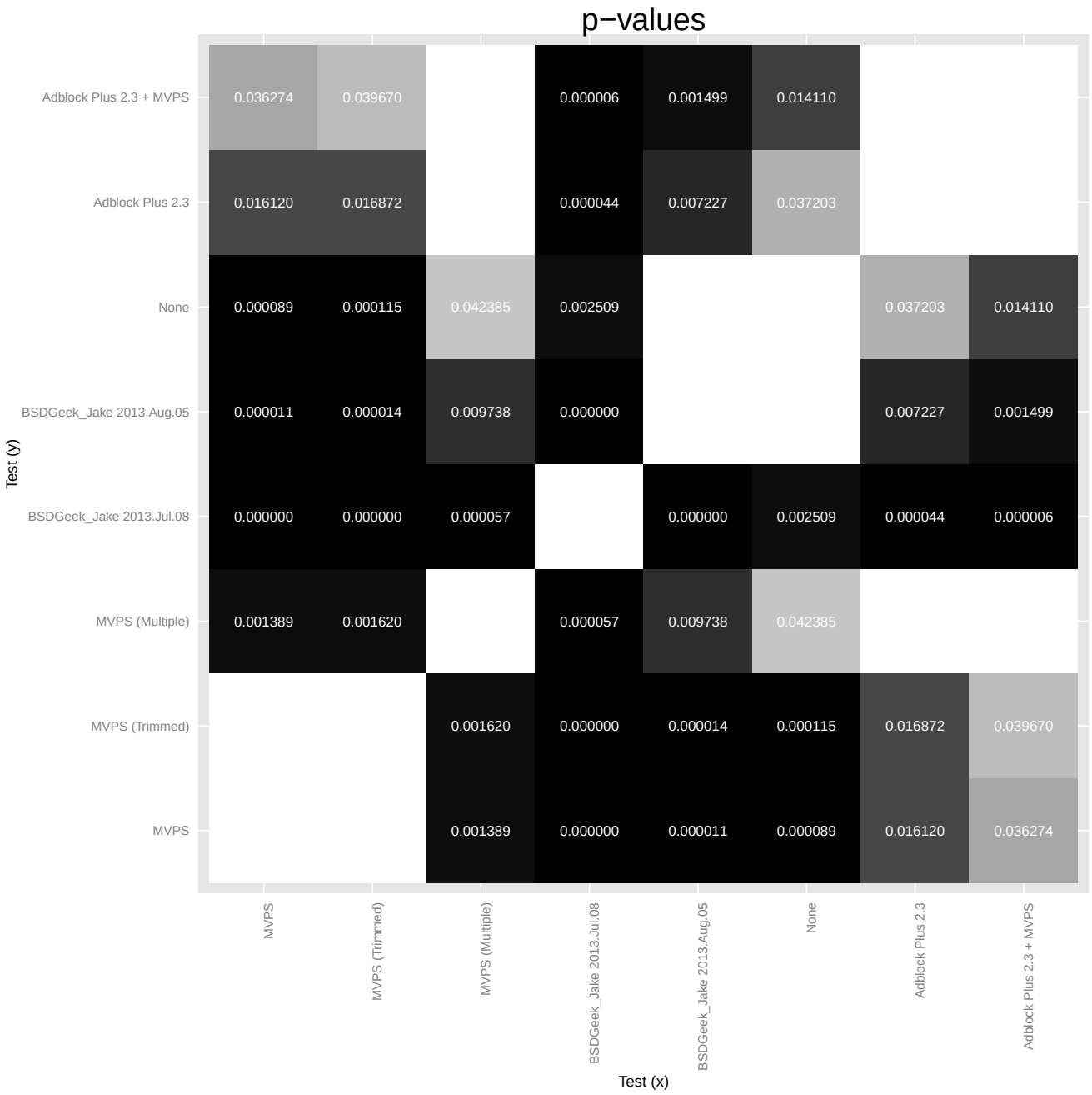

Figure 3: p-values for the paired t-tests. p-values > 0.05 are coloured in white, and the values from 0.00 to 0.05 95% to 100% confidence are shaded from black to white. A p-value less than 0.05 indicates that the power measurements between the tests are different. White tests indicate that the test results are not statistically significant in their difference.

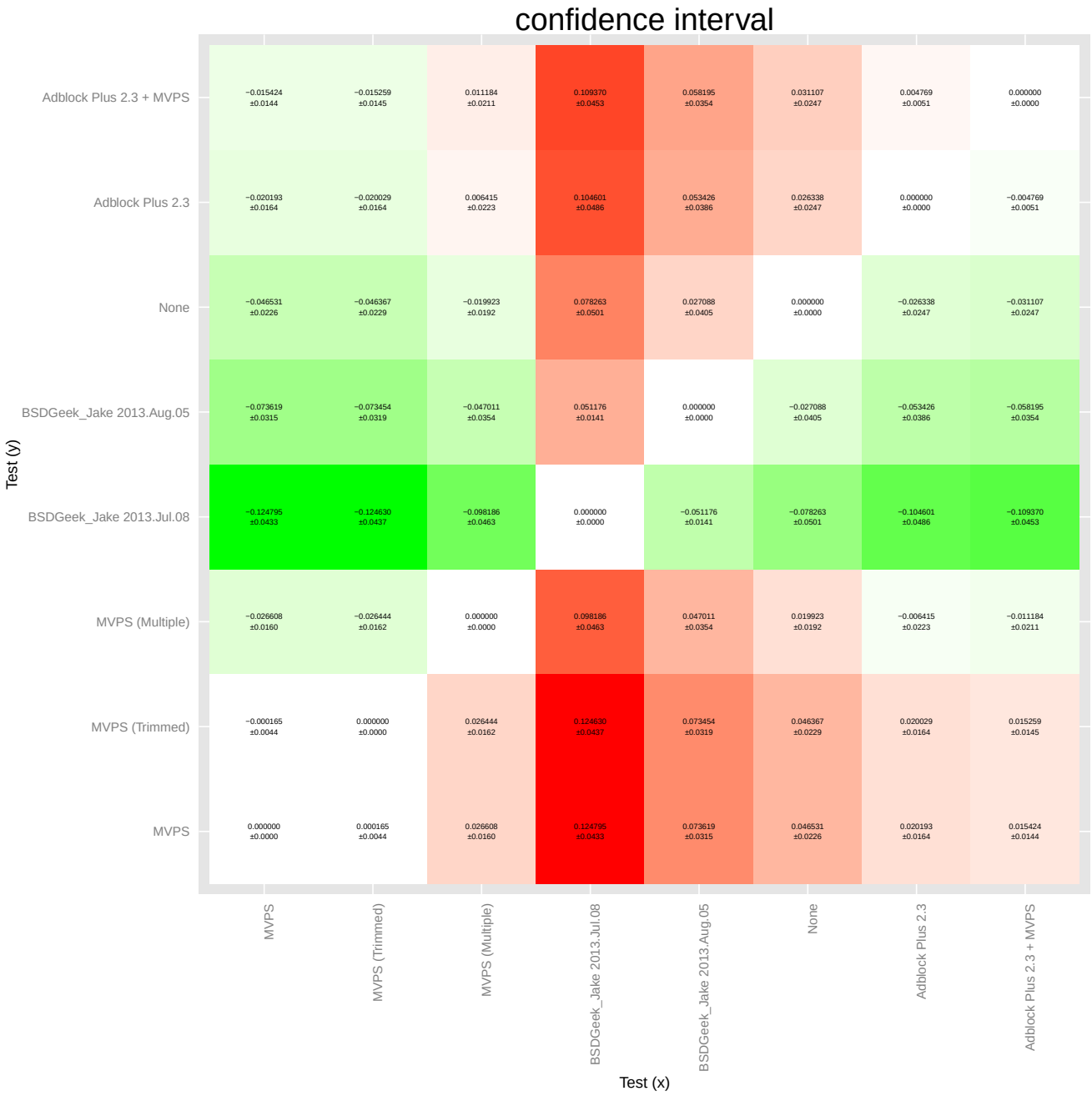

Figure 4: 95% Confidence intervals for the comparison between tests, in watts. A green square means that x-axis test, Test(x), is more power efficient than the y-axis test, Test(y), and a red square is the reverse. For example, AdBlock Plus 2.3 consumed 0.026 watts less than the control (None), and is colored a light shade of green.