

The Hidden Energy Cost of Web Advertising

R.J.G. Simons

Faculty of Electrical Engineering, Mathematics and Computer Science
University of Twente, Enschede, The Netherlands

ut@randysimons.nl

ABSTRACT

Advertising is an important source of income for many websites. To get the attention of the unsuspecting (and probably uninterested) visitors these ads tend to use elaborate animations and graphics. Displaying such ads on the visitor's screen sometimes requires a vast amount of CPU-power. Present day desktop-CPU's can consume in excess of 100W when in full use. Thus, an advertisement that consumes a lot of the computational capacity of the computer, also consumes a lot of energy. Manufacturers strive to reduce energy consumption for environmental benefits and to increase battery life of mobile devices. But using a lot of energy for displaying advertisements can be considered counterproductive.

We've investigated the power consumption of advertising on web pages while browsing the web. To do so, we used an energy meter to measure the difference in power consumption of PCs while surfing the web normally with ads enabled, and while surfing with ads blocked.

To consistently simulate normal web browsing, we've created a browser-based tool called AutoBrowse, which periodically opens an URL from a predefined list. For blocking advertisements, we used the Adblock Plus extension for Mozilla Firefox. We also used Apache HTTP server and its mod_proxy module to act as an ad-blocking proxy server, which can be used with all browsers.

The measurements on several PCs and browsers show that on average the energy consumption caused by advertisement on websites is 2.5W. This is 3.4% of the total energy consumption of the PCs while browsing the web.

Keywords

Energy consumption, Adobe Flash, web browsing, advertising

1. INTRODUCTION

Many laptop users know that the amount of noise the cooling fan of their laptop makes closely corresponds to the actual workload of the CPU. And so did we. But we were surprised that sometimes simply viewing websites also seemed to cause the fan to start cooling. Closer inspection soon learned us that this mostly seems to happen on web pages with “flashy” advertisements.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

12th Twente Student Conference on IT, January 22, 2010, Enschede, The Netherlands.

Copyright 2010, University of Twente, Faculty of Electrical Engineering, Mathematics and Computer Science.

The business model of present day websites is often based on revenue of advertising. These advertisements often take form as “banners”, wide or tall graphics above or next to the content of the web pages.

To maximize income by increasing exposure of the ads, there usually are several banners on a single page. The result is that ads have to compete with each other for attention of the unsuspecting, and often uninterested, visitor. To do so, the ad-designers resort to animated images, and sometimes even large animated overlays and videos.

There are currently two de-facto standard techniques for animation: animated GIFs, and Adobe Flash “Movies”. The latter is popular for its ability to animate vector graphics, scripting possibilities and ability to include normal video footage.

Driven by ecological, financial and laptop-battery-preserving motives, CPU and operating system manufacturers try to minimize the power consumption of CPUs, and computers in general. They are fairly successful to minimize the energy consumption of present day CPUs when the computer is idle, but a desktop-CPU under full load can consume from about 45W to about 95W, depending on the model [1].

1.1. Specific problem

Because putting a CPU under computational stress requires power, and thus harms the environment, costs money, and decreases battery live of laptops, it is best to keep the CPU-load as low as possible, and only give it something “useful” to do.

However, web advertisements often seem to be quite CPU-intensive when they are active, even though they occupy relatively little display space. While the user might ignore these ads displayed by his web browser, the CPU of his computer certainly doesn't. Depending on the CPU model and the specific design of the ad, the power consumed by the CPU for simply displaying the advertisements might be significant. An example of this is shown in the figures below.

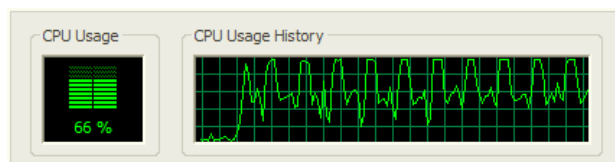


Figure 1: CPU usage of displaying a website (www.telegraaf.nl) with ads enabled.

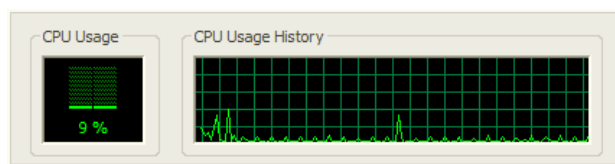


Figure 2: CPU usage of displaying the same website, but with ads blocked.

The figures 1 and 2 are screenshots of the Windows XP task manager, showing the total CPU-usage over a few minutes time, while rendering and displaying the website www.telegraaf.nl in a web browser. In both cases, the site is opened just once, after which no further actions were taken. The first screenshot shows the CPU-usage of rendering and displaying the website with advertisements enabled, while the second screenshot shows the CPU-usage of displaying the exact same site, but with advertisements blocked.

The screenshots indicate that 50% to 100% of the CPU is used only for displaying the Flash-advertisements on web page, which amounts to about 15W of energy consumption for the CPU used. For comparison, a contemporary laptop requires around 15W when operational. However, this is just a single example; not all advertisements we've seen put so much load on the CPU. Others hardly use any CPU-power at all.

Apart from Flash, Javascript is often used on websites for marketing purposes. We recognize two forms: keyword advertising [18] and web tracking software.

An example of keyword advertising is IntelliTXT [16]. When a visitor opens a page containing the IntelliTXT-script, the script scans the text on the web page, and converts some keyword or sentences into clickable links to the advertiser's website. Since the scanning is done on the client using Javascript, it requires computational effort of the web browser.

Web tracking software, such as Google Analytics [8], is used on websites to gather information on demographic, sociographic and behavioural aspects of the visitors of websites. It can use Javascript to enhance its capabilities, which now can include the ability to monitor the exact links the visitors click, how often and far the visitor scrolls his browser screen and how long the visitor stays on the web pages. We concede that web trackers are not advertisements, but the two are closely related: the information gained by web trackers allows for better targeted advertising. And just like advertising, web tracking can be considered as an "unsolicited feature" on websites.

While we believe these Javascript-based advertising and tracking software is relatively "energy efficient" compared to some other (Flash)-based advertising, we did notice it requires some CPU-effort when loading certain websites.

Since there usually is no indication on the power consumption of the computer apart from the CPU cooling fan on laptops, the user might be oblivious to the costs (monetary, ecological, battery life) of these ads on websites. If the visual extravagance doesn't bother him, there would be little reason to try to avoid those ads.

All in all, the total power consumption caused by web advertising and tracking could be quite substantial. If so, it might be useful to find out what can be done to reduce the energy consumption. And perhaps websites can change their advertisers-policy, and only allow "green ads" to be published on their website. Given the current climate-crisis, this could very well be an extra marketing-strategy for some sites.

1.2. Related work

The website AnandTech made a comparison between web browsers on laptops [2], to determine which browser was the most energy efficient. As a side-project, they measured the effect of web advertisements on the battery life of the tested laptops, by blocking the ads using the Adblock Plus extension [10] on Firefox.

They found that using Firefox the battery of the laptops lasted longer with Adblock enabled, than with Adblock disabled. For the AMD Athlon CPU the battery lasted 4.3% longer, on the Intel Core 2 Duo CPU even 5.7% longer. Using the Intel Atom CPU, the difference was negligible. However, they do note that "Safari seems to want more CPU power than the Atom can provide, with the result being the processor is often near 100% utilization for significant amounts of time on Flash-heavy sites". This would indicate that the browser can also be of influence on the energy consumption of the ads.

1.3. Research questions

We try to answer the following questions:

1. Does web advertising and tracking consume a significant amount of energy?
2. How much energy is used for displaying advertisements and banners compared to the total energy usage of a computer, while browsing the web?
3. Is there a difference between platforms? (CPU architecture, browser)

1.4. Approach and structure

Directly measuring the direct energy consumption of ads by a web browser is difficult. We've chosen for a sequential approach: first we measured the energy consumption of the entire PC while browsing with ads enabled, then we measured the consumption while browsing with ads disabled. The difference, normalized to a period of one hour, is the amount of energy used for displaying ads and tracking systems. This result can then be compared to the energy consumption of the entire PC.

Each test session (ads enabled and ads disabled) was run on the same PC on the same web browser. We've conducted several tests sessions, ranging from older, low end PCs to medium-end PCs. Several browsers were tested as well. All in all typical set-ups for generic home and office use.

For this approach we used and made some specific tools to aid the measurements, which are detailed in chapter 2. Chapter 3 explains the set-up of the experiment and the use of the tools. The results of the measurements are shown in chapter 4, and are discussed in chapter 5. In chapter 6 we draw the conclusions, and answer the research questions.

2. MEASUREMENT TOOLS

Our approach mentioned in section 1.4, hinges on three methods: realistic and repeatable web browsing, consistent blocking of ads and trackers, and accurate measurements of the PCs used while browsing the web.

2.1. Automatic consistent web browsing

For accurate and consistent results we set out to automate the web browsing for the measurements. The following requirements were considered:

1. The pages visited should be realistic, containing a realistic amount of realistic ads.
2. The web browsing should be plausible "human-like":

1. Visit relatively popular websites.
2. Periodically navigate from one page to another.
3. Open up more than one browser tab or window.
3. Repeatable; if the exact same test was run twice, it should produce the same results.
4. Cross-platform; it should work on several platforms, on several contemporary web browsers.

We believe that if these criteria are met, the results are close enough to real life web browsing behaviour to be able to answer the research questions in section 1.3.

2.1.1. AutoBrowse

To run consistent tests in compliance with the requirements stipulated in the beginning of section 2.1, we developed a JavaScript-tool called “AutoBrowse”, which can be run directly in any present-day web browser [13].

The basic function of AutoBrowse is to periodically open a link from a predefined list of web links in a new browser tab or window. When a certain maximum number of tabs or windows have been opened, the oldest is closed. At the end of the list, AutoBrowse simply restarts from the top of the list.

When opening the AutoBrowse page, the user can alter the list of links to be opened, the interval between every opened link, and the maximum number of opened tabs or windows. After changing the default values, the “reset”-button must be pushed to apply the changes. If the user is satisfied with the parameters, he can start the procedure by pressing the “start/pause”-button. AutoBrowse will now open each link in a new tab or window. The progress is shown in the title bar or tab of AutoBrowse.

It should be noted that it can make a difference whether the links are opened in new tabs or in new windows. Tabs always overlap each other, thus only one tab can be active at any given time. It might be possible that Flash and browsers suspend rendering or updating invisible tabs, thereby reducing the energy consumption. Windows, on the other hand, might only partially overlap each other or even not at all. Browsers are thereby forced to render the screen, resulting in more energy consumption.

Due to limitations in Javascript, AutoBrowse has no influence in whether the links are opened in new tabs, or in new windows. Mozilla Firefox and Opera default to open new pop-ups as tabs, whereas Apple Safari, Google Chrome, Microsoft Internet Explorer 7 and 8 open pop-ups as new windows. In Firefox, Opera and Internet Explorer, this behaviour can be altered. Safari on Windows and Chrome currently lack this option; pop-ups are always opened in a new window.

2.2. Blocking ads and trackers

Several methods for blocking advertisements and trackers have been investigated. They can be divided into two groups:

1. **Blocking of URLs.** Whenever a web page -and thus the browser- requests a document from an URL, this URL is matched against a black-list. If the URL matches with one or more URLs in the black-list, the request is aborted or denied.
2. **Content filtering.** The (HTML) content of retrieved web pages is checked against a rule-based filter. The

parts of the content that match one or more rules are scrapped or replaced, thereby preventing that the filtered content is shown on screen.

Both methods can be implemented either inside or outside the browser. Also, both methods can be combined: filters can employ both URL blocking and content filtering for better results.

Implementations inside the browsers are always browser-specific, and not all contemporary browsers support this option.

External implementations are usually designed as an HTTP-proxy. This method works with every browser, as long as the browser can route the HTTP-requests through an HTTP-proxy. In practise, all browsers support this option.

2.2.1. False positives and negatives

Whichever option is chosen, care must be taken to avoid incorrect classification of the URLs or content. But unlike anti-spam solutions for e-mail where false positives should be avoided at all costs, a few false positives are of little concern, as long as it's insignificant and/or are “compensated” by false negatives.

However, filtering shouldn't be too strict. For example, simply disallow all Flash-based content would also block many videos on popular websites as YouTube.

2.2.2. Investigated filtering options

The following browsers and their options have been investigated:

Opera, with built-in URL-filter

This browser has built-in support for URL-based filtering, using a black-list. Items on this list can contain wildcards. Whenever the browser retrieves a document, the document's URL is checked against the blocked content list. If a match is found, the browser aborts the request.

Mozilla Firefox, with Adblock Plus

By default Firefox has no filtering capabilities. However, the free add-on Adblock Plus [10] adds both URL and content filtering. Ready-made configuration files can be downloaded for both ad-blocking and tracker-blocking. The syntax for filtering-rules is more powerful than Opera's, and includes regular expressions. Content filtering is based mainly on CSS-selectors for popular sites, which targets known ads very precisely.

Easylist is a popular configuration file for Adblock. It can be supplemented by Easylist Privacy, which adds tracker-blocking.

Proximodo

Proximodo is a proxy server with content and URL filtering capabilities [5]. It targets much more than just ad-blocking, but it still is one of its main features. The filtering syntax and options are very flexible, and include regular expressions.

By default it does not employ URL-filtering, but it relies primarily on content filtering. Proximodo's content-filtering techniques seem to rely on heuristics to recognize ads.

Privoxy

The Privoxy proxy server aims to improve privacy while browsing the web [12]. Like Proximodo it has a wide range of options and features. But unlike Proximodo, manual configuration is required in order to get ads and tracker-blocking, without modifying the content and/or requests.

Squid

Squid is specifically designed as an optimizing and caching proxy server [15]. This nature can interfere too much with the tests, and therefore this option was quickly abandoned.

Apache webserver + mod_proxy

Despite the fact that the Apache web server is primarily an HTTP server for websites [3], it can easily be converted in a non-caching, full pass-through proxy server, by enabling the mod_proxy module [4].

The normal installation and configuration procedure of Apache is less convenient than the other options listed here. However, an all-in-one package such as XAMPP makes it easier. Also, we happen to have prior experience with Apache and the mod_proxy-module.

Mod_proxy doesn't support content filtering, but it is possible to use regular expression based rules [13].

2.2.3. Decision for filtering options

The Adblock extension for Firefox, in combination with the Easylist configuration files, does an excellent job filtering both advertisements and web trackers. During the years of Opera-use, our Opera blocked content lists have proven to be very successful as well, even though the list is much shorter.

However, while both options are good solutions, they are browser-specific. A proxy-based solution is browser-independent.

While the specialised solutions Proximodo and Privoxy undoubtedly can eventually perform excellently, the standard configurations do more than simply blocking, but perform other “clean-up tasks” as well, which is not what we want. Moreover, Proximodo's default lists performed less than Adblock and our Opera's blocked content list. During the investigation, Proximodo v0.2.5 crashed on occasion, which rendered it too impractical for the tests.

It turned out to be easy to convert our wildcard-based Opera's blocked content list to a regex-based-rule for Apache with mod_proxy. In effect, this set-up provides the exact same result as Opera's blocked content list, but now implemented in a cross-browser way.

We already had Apache with mod_proxy installed on a laptop. When the laptop is placed inside the LAN of the test-PCs, we could simply configure all the web browsers to use the laptop with Apache/mod_proxy as a proxy server. Thus, this solution is a browser- and platform-agnostic advertisement and tracker filtering solution, which can be used with little effort for all browsers and platforms.

All on all, our conclusion is that Opera's blocked content list and Adblock for Firefox are good options, when testing using those browsers. The solution using a laptop with Apache, mod_proxy and the adaptation of our Opera's blocked content list is very practical for general use with all browsers and platforms.

2.3. Measuring energy consumption

For the actual measurement of the energy consumption, we used a Voltcraft Energy Monitor 3000 [6]. This is a consumer-grade measuring device, amongst others capable of measuring the current power use in W with a resolution of 0.1W, and cumulative energy consumption, displayed as kWh, with a resolution of 1Wh.

The device has a built-in clock, with a resolution of 0.01 hour. In combination with the kWh-measurement, this provides an averaged power usage of the measured device in W.

However, due to the fact the resolution of the energy consumption is just 1Wh, a measurement of one full hour has to be made in order to get averaged results with a resolution of 1W. A two hour measurement results in a resolution of 0.5W, etc.

3. MEASUREMENT ENVIRONMENT

3.1. Obtaining the browse list

The AutoBrowse tool uses a list of URLs, which it will open in order. The next task is to devise such a list. The URLs should meet requirement 1 and 2 of the list at the beginning of section 2.1.

At first, we've obtained a list of websites nominated for the Dutch “website of the year”-award. This list was reasonably diverse, and the listed websites are reasonably popular. But on closer inspection, this list didn't satisfy requirement 1: all the URLs in the list pointed to the homepage of the websites. However, homepages are often little more than portals for the rest of the site. In many of these cases, the homepages contained less advertisements than the actual content pages of the sites.

We then looked for a list of deeplinks to popular sites. For this, we looked at the website Digg, which is “[...] a place for people to discover and share content from anywhere on the web. From the biggest online destinations to the most obscure blog, Digg surfaces the best stuff as voted on by our users” [7]. The “top stories”, such as posted on the main page, are good candidates: popular, deep links to wide variety of websites.

Digg provides a REST-based API to retrieve at most 100 links at once. The standard result is an XML-document which contains the links with other details. We used this API to get two documents with the 200 most popular stories, containing URLs and other meta-data. Using an XPath-query we then extracted the URLs from the list [13]. This resulting list of URLs is used for the test sessions using AutoBrowse.

3.2. Interval between visits

AutoBrowse opens a new URL from the browse list every 30 seconds by default. While this might or might not be a good value for the average time spend by a human on a web page is over little importance; the browse list is considered a “random” list of representative web sites, thus on average, over a longer period, the time a single page is visible is mostly irrelevant.

However, there are factors which should be considered:

- Smaller intervals causes more pages to be loaded and rendered within the same period, increasing the CPU-

time spend for rendering the page, and thus the total power consumption.

- It often takes a couple of seconds before the (potentially CPU-intensive) ads are shown when the URL is opened, thus lowering the total power consumption when browsing with ads enabled.
- Some advertisements appear to have a limited number of animation-cycles. After the animation sequence has played for a certain amount of times, the animation stops, which can significantly reduce the CPU-load.

We maintained the default 30 seconds interval for most of the tests.

3.3. Number of open tabs/windows

Almost all current-day popular web browsers support tabs. Instead of spawning a new window when the user clicks on a link with `target="_blank"`-attribute, the browser opens the link in a new tab. Users can also manually open new tabs.

This option, and particularly due to the fact that many websites use the `target="_blank"`-attribute on links to external pages, users often have several websites open at any given time. Weinreich et al. determined in 2006 that users had 2.1 open tabs on average [17].

Because the number of open tabs and windows, and thus the number of open web pages with “active” advertisements can be of influence on the energy consumption, AutoBrowse opens each link in a new tab, while closing the oldest tab. This behaviour mimics the human web browsing behaviour.

The maximum number for this experiment of tabs or windows that AutoBrowse opens is set to 3. While this is more than the 2.1 open tabs as determined by Weinreich et al., we believe that due to the increased number of browsers supporting tabs since 2006 -especially Internet Explorer-, this trend is rising.

3.4. Ad blocking

As was determined in section 2.2.3, the Adblock-extension for Firefox in combination with the Easylist configuration files, and the laptop with Apache + `mod_proxy` with our own configuration [13] are the best candidates for the blocking of advertisements and web trackers.

For all tests, except some sessions with Firefox, the laptop/Apache-proxy server was used.

3.5. PC configurations

For this experiment we used a variety of PCs, ranging from low-end to medium/high-end. We believe these can be considered fairly average, and can provide valid results. The details of the PCs are listed in table 1 below.

Before the test sessions the screens savers and power management functionality of the display were disabled. Both options could interfere with correct measurements: a screen saver might cause the browsers to stop updating the screen, as it is invisible anyway. Power management might suspend large portions of the GPU, reducing the power consumption. And when a person is browsing the web, obviously neither the screen saver, nor power management is active.

Also, no other activities were done on the PC while measuring power consumption.

Table 1: PCs used for the experiment

PC	CPU	GPU	Operating system	Flash version
1	Intel Q6300 Quad core @ 2,5GHz	Nvidia GF 7600 GS v8.16.11.9107	Windows Vista SP2	v10.0
2	AMD AthlonXP 1700+ Single core @ 1,47GHz	ATi Radeon 5500 v8.33.0.0	Windows XP SP3	v10.0
3	Intel 6600 Dual core @ 2,4GHz	Nvidia GF 7600 GS v6.14.11.8585	Windows XP SP3	v10.0
4	Intel Pentium M 735 Single core @ 1,8GHz	ATi Radeon 9700M	Windows XP SP2	v10.0

3.6. Browser configurations

Various browsers have been used for the test. Some browsers are available as a “portable” version [11], [9]. Were possible, the portable version was used; it provides a clean and typical installation, can operate independent of other browsers installed on the PC, and does not interfere with the rest of the PC.

The used browsers and their configurations for the experiment were:

Apple Safari

This browser, version 4.0.4 for Windows, had to be installed normally on the PC. It was a clean, typical installation, and no special settings were made.

Safari opens new pop-ups in a new window, instead of new tab. AutoBrowse has no influence in this behaviour, but because most people will open new tabs rather than new windows, we looked for a way to change this behaviour of Safari. Unfortunately, while the Mac OS X-version of Safari can be instructed to open pop-ups as new tabs using the `TargetedClicksCreateTabs = true` configuration option, this option has no effect on the Windows version used.

Another problem was that Safari had only a global setting for blocking or allowing pop-ups. Because AutoBrowse uses pop-ups, all pop-ups had to be allowed. Fortunately only a few websites use (unwanted) pop-ups these days. During testing, those pop-ups were closed manually.

Microsoft Windows Internet Explorer

The current version 8 of Internet Explorer was used. It was installed normally on the PC – a true portable version is impossible. However, because this browser wasn't the default browser used on the systems, it had been in a clean and unmodified, typical state.

Internet Explorer 8 supports tabbed browsing. By default, pop-ups are blocked, but can be allowed per domain. New pop-ups are opened in a new windows, but this setting can be altered, so they are opened as new tabs instead.

On occasion, websites caused a (javascript)-error, in which case Internet Explorer showed a message box on screen. When this happens, AutoBrowse cannot automatically close the corresponding tab. On such occasions, we closed the tab manually.

Mozilla Firefox

In all cases, the portable version of Firefox 3.5.5 was used. This version has disk caching disabled. As per typical installation, unwanted pop-ups are disabled, but this can be overridden on a per-domain basis. New pop-ups (those opened by AutoBrowse) are opened in a new tab.

Opera

We used the portable version of Opera v10.10. Apart from the disabled disk cache this provides a very typical installation. Just like Firefox, unwanted pop-ups are disabled by default, but can be enabled on a per-domain basis. For the test, we've enabled pop-ups for the AutoBrowse domain.

Adobe Flash, however, is a system-wide installation. The portable versions of the browsers use the Flash already installed on the system. In all situations, this was the current version 10.0.

3.7. Measurement procedure

After a PC and browser have been chosen, a test session can be prepared.

1. The mains of the PC – and only the PC – are connected to the mains through the energy meter.
2. Non-essential USB devices, such as printers, photo cameras, mobile phones, are disconnected, as they can draw up to 2.5W each from the PC [14], which might distort the measurement.
3. The PC is started. At the desktop, the screen saver and power management for the display are disabled.
4. The chosen web browser is started, and configured to open pop-ups as new tabs, if possible. Also, pop-ups from the domain hosting AutoBrowse is added to the allow-list for pop-ups. The browser cache is cleared.
5. The AutoBrowse web page is requested. Settings are changed, if necessary, after which the “reset” button is pressed.

6. The time tracker of the power meter is reset, and immediately the “start/pause” button in AutoBrowse is pressed.
7. After at least two hours, the exact elapsed time and energy consumption as is recorded by the energy meter is read, and written down in a log.
8. The reset button in AutoBrowse is pressed.
9. The browser is prepared for filtering, e.g. Adblock is installed and configured, or the proxy settings are made to use the laptop with Apache/mod_proxy as proxy server. The browser cache is cleared, and the browser is restarted.
10. AutoBrowse is opened again, and the same parameters as in step 5 are set.
11. The time tracker of the power meter is reset, after which the “start/pause” button in AutoBrowse is pressed.
12. Again, after about two hours, the exact elapsed time and energy consumption is written down in a log.

This procedure is repeated on several PCs, with varying browsers.

4. RESULTS

Table 2 shows the results of the measurements. The average power is the measured consumption divided by the time for the test. The difference in the average power consumption between the normal browsing and the ads/tracker blocked browsing is the averaged energy in watts used to display the advertisement on the websites.

On average, the power consumption used by our test-PCs and browsers while browsing the web was 76.5W with ads/trackers enabled, and 74.0W with ads/trackers blocked. Thus, 2.5W is used solely for web trackers and displaying ads. In other terms: given the baseline of 74.0W, web advertisements increase the energy consumption by 3.4%.

5. DISCUSSION

The increase in energy consumption of 3.4% due to advertisement is comparable to the decrease of a laptop's battery life of

Table 2: Result of measurements. Power consumption of PCs, while browsing the web.

PC	Browser	Open tabs	Filter method	seconds/page	Normal browsing			Ads/trackers blocked			Difference (Power used for ads) (W)
					Time (hour)	Measured consumption (Wh)	Average power (W)	Time (hour)	Measured consumption (Wh)	Average power (W)	
1	Firefox 3.5.5	3	Adblock Plus / EasyList + EasyPrivacy	20	2.03	138	68.0	2.03	135	66.5	1.5
1	Safari 4.0.4	3	Apache filter proxy	30	2.09	162	77.5	2.09	154	73.7	3.8
1	Firefox 3.5.5	10	Adblock Plus / EasyList + EasyPrivacy	30	2.11	148	70.1	2.11	143	67.8	2.4
1	Firefox 3.5.5	3	Apache filter proxy	30	2.09	142	67.9	2.09	138	66.0	1.9
2	IE8	3	Apache filter proxy	30	2.08	221	106.3	2.08	214	102.9	3.4
2	Firefox 3.5.5	3	Apache filter proxy	30	2.04	214	104.9	2.04	209	102.5	2.5
3	Firefox 3.5.5	3	Apache filter proxy	30	2.29	231	100.9	3.1	304	98.1	2.8
4	Opera 10.10	3	Apache filter proxy	30	2.11	34	16.1	2.15	31	14.4	1.7
Average (W):							76.5			74.0	2.5

about 5% as measured by AnandTech on the laptops with desktop-like CPUs [2]. However, AnandTech used only a very few websites in their test.

Still, the figure of 5% is remarkable: a laptop has a built-in display, which requires about the same amount of energy regardless whether the image it shows is static, or animated. Thus, if the display would be taken out of the equation, as is the case in our test, the loss of battery life would be even more than 5%. This is supported by the measurement of our own laptop, PC 4, which had its internal display disabled in favour of an external display. Table 2 indicates an increase of 12% in energy consumption due to web advertisement for this PC.

Some other interesting facts might be derived from the figures in table 2: Firefox with 10 open tabs used more energy than when having 3 open tabs. This might be due to the fact that some websites had videos playing, which were not necessarily ads, but which kept playing even if their tab was invisible. More open tabs meant the videos played longer before the tab was closed by AutoBrowse. Because video decoding is rather CPU-intensive, it increases the energy consumption.

Another observation is that both Apple Safari and Microsoft Internet Explorer 8 seem to require more energy than Firefox 3.5. However, accurate measurement of Safari and Internet Explorer using AutoBrowse was difficult, as Safari couldn't be persuaded to open new tabs instead of new windows; Internet Explorer occasionally showed a Javascript message box, and thereby prevented AutoBrowse to close the tab. We had to close those tabs manually.

The final result of 2.5W difference due to ads remains an approximation, due to several circumstances:

- The energy meter had a resolution of 1Wh.
- Ad blocking wasn't always successful; some ads slipped through the filters.
- The number of test sessions conducted is limited.

6. CONCLUSIONS

We measured the difference in energy consumption of PCs while browsing the web with ads and trackers enabled, and with ads and trackers blocked. While the difference is relatively small, it is clear that advertisements on web pages do require a significant amount of energy.

The power required for rendering and displaying these ads in our experiment was 2.5W. Without ads, the PCs used an average of 74.0W. Thus, while browsing the web on the tested PCs and browsers, web advertisement increases the total energy consumption of the PCs by 3.4%.

The energy consumption depends on the browser used. On one occasion Internet Explorer 8 used 38% more energy than Firefox 3.5.5, just for rendering ads. Another measurement showed that Safari on Windows required 100% more energy for ads than Firefox. However, this might be in part or in whole due to the fact that we couldn't get AutoBrowse to open new tabs instead of new windows in Safari.

Because of these differences between the browsers, the global average power consumption is probably more than the 2.5W measured here. Especially Internet Explorer is still more popular than Firefox and Opera, which use considerably less power.

On the 4 platforms measured, ranging from older laptop to a quad core contemporary PC, the energy consumption by ads was mostly less than $\pm 1W$ of the average of 2.5W. This indicates the platform is of little effect on the energy consumption.

6.1. Further work

While this study makes clear that web advertisement and tracking does require a measurable amount of energy, the number of tests conducted was limited and insufficient to answer the following questions:

- Why do some ads require more energy than others? Is there a "quick fix" for designers and developers to reduce the required CPU-usage?
- As more and more web browsers run each browser tab or window in a separate process, is a multi-core CPU an advantage or disadvantage when it comes to energy consumption for ads?
- Apart from the energy consumption by the local PC, what is the energy costs for transportation the data of all those ads across the internet?
- As monitor sizes and resolutions increase, does it have additional effect on the energy consumption, by allowing more and larger ads to be displayed at once?
- How much energy is actually spent globally by ads displayed by browsers?
- Do alternatives for Adobe Flash, such as SVG and Canvas, fare better or worse?

7. REFERENCES

- [1] AMD: *AMD Athlon™ II Processor Competitive Comparison*, <http://www.amd.com/us/products/desktop/processors/athlon-ii-x2/Pages/AMD-athlon-ii-x2-processor-competitive-comparison.aspx>
- [2] AnandTech: *Browser Face-Off: Battery Life Explored*, <http://anandtech.com/mobile/showdoc.aspx?i=3636>
- [3] Apache foundation: *Welcome! - The Apache HTTP Server Project*, <http://httpd.apache.org/>
- [4] Apache foundation: *Apache Module mod_proxy*, http://httpd.apache.org/docs/2.2/mod/mod_proxy.html
- [5] Boucher, Antony: *Proximodo*, <http://proximodo.sourceforge.net/>
- [6] Conrad Electronic Benelux: *Energy Monitor 3000 - Energiekosten meter (Dutch)*, <http://shop.conrad.nl/elektronica-meetapparatuur/meetinstrumenten/energiekosten-meter/125331.html>
- [7] Digg: *What is Digg?*, <http://about.digg.com/>
- [8] Google: *Google Analytics | Official Website*, <http://www.google.com/analytics>
- [9] Obermair, Markus: *Opera@USB mobile Opera - portable Opera for USB*, <http://www.opera-usb.com/operausb.htm>

- [10] Palant, Wladimir: *Adblock Plus: Save your time and traffic*, <http://adblockplus.org/>
- [11] PortableApps: *Mozilla Firefox, Portable Edition*, http://portableapps.com/apps/internet/firefox_portable
- [12] Privoxy Developers: *Privoxy - Home Page*, <http://www.privoxy.org/>
- [13] Simons, R.J.G.: *The Hidden Energy Cost of Web Advertising*, <http://randysimons.nl/125,english/149,energy-consumption-of-web-ads/>
- [14] USB Implementers Forum: *USB Frequently Asked Questions*, http://www.usb.org/developers/usbfaq/#pow_dis
- [15] Various developers: *squid : Optimising Web Delivery*, <http://www.squid-cache.org/>
- [16] Vibrant: *What is IntelliTXT™?*, http://www.vibrantmedia.com/advertisers/intext_advertising/faq.asp#4
- [17] Weinreich, Harald et al., 2008: *Not Quite the Average: An Empirical Study of Web Use*. In: ACM Transactions on the Web, vol 2, no 1, page 5:15
- [18] Wikipedia contributors: *Keyword advertising*, http://en.wikipedia.org/wiki/Keyword_advertising