Energy efficiency of HTTP2 over HTTP1.1 on the mobile web

CSE-E5440: Energy-efficient Mobile Computing
Group C

Raatikka, Marko
`marko.raatikka@aalto.fi`

Lucas, Triefenbach
`lucas.triefenbach@aalto.fi`

April 30, 2016

1 Part I

1.1 Introduction

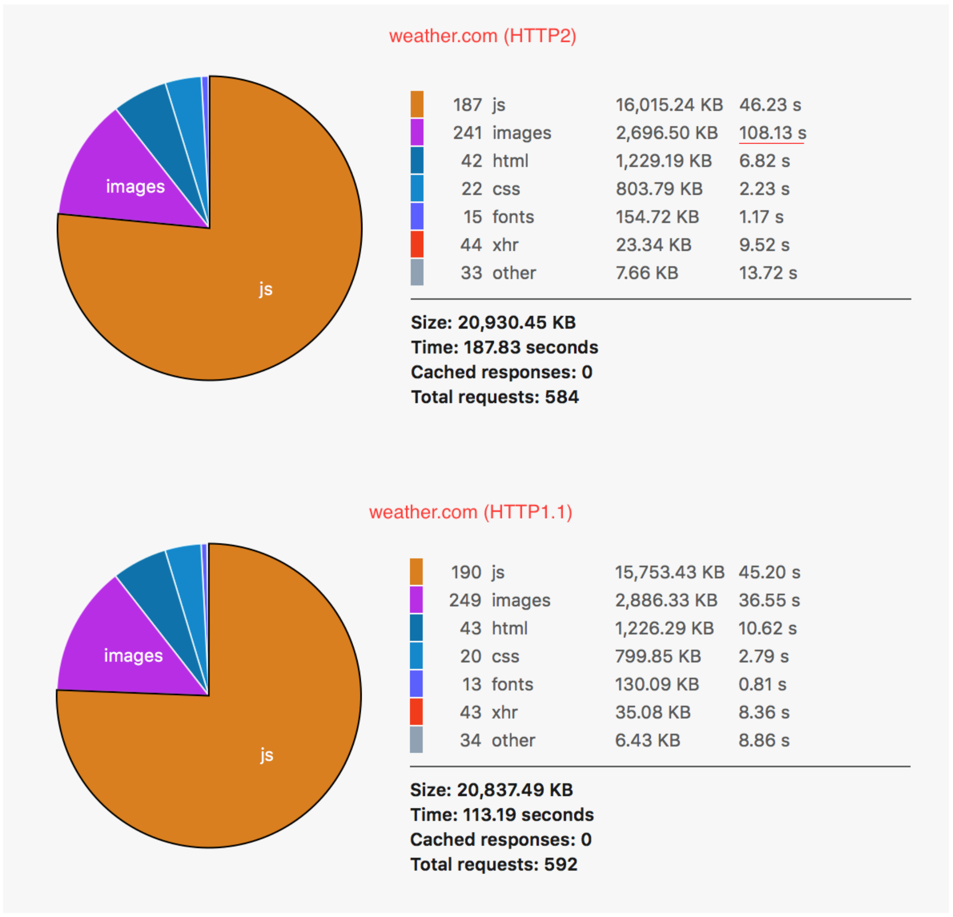
With the increasing computation power and connection speed of mobile phones in recent years, the amount of data flowing through the Internet to the consumer's end device continues to grow. While web applications have become ever more complex and the amount of HTML, JavaScript and CSS required to render a web page has grown, HTTP1.1 – the prevailing transfer protocol of today's web – has failed to keep up with the pace. One of the culprits of HTTP1.1 is that it can only handle one request per a TCP connection and that it can only have a certain maximum number of connections per host (5-8 connections depending on the browser). To work around these shortcomings developers have used techniques such as domain sharding (hosting assets across multiple domains), image spriting and inlining CSS & JavaScript. However, a transport layer problem can not be efficiently solved at the application level.

The work for a new version of HTTP protocol was set forth by Google in 2012 in the form an open networking protocol called SPDY. Later, in 2015 , building on the work done for SPDY, HTTP2 was proposed as the future standard protocol. The biggest improvements HTTP2 introduces over HTTP1.1 are multiplexing of requests and responses to avoid a so called *head-of-line blocking* inherent in HTTP1, header compression, prioritization of requests and server push mechanism. Web services are slowly migrating into adopting HTTP2, but as of today, only about 7% of the websites have fully migrated to the new protocol. [1][3][5]

This work sets to study the impact of HTTP2 on mobile power consumption. Previous work on the topic has been done by [4]. However, this work focuses more on generating high traffic and mimicking common navigation patterns. Due to time constraints testing is limited to a single browser as opposed to testing across various versions. The content is organized as follows: chapter 1.2 describes the method used, the testbed environment and the steps of a test run. Chapter 1.3 continues by presenting the experiment results, which are later reviewed in Chapter 1.4. Last, an outline for the part II of the study is given.

1.2 Design and Implementation

The energy efficiency of the HTTP1.1 and HTTP2 protocols were measured by generating large amounts of network traffic. Six different web services were chosen as the targets, including Flickr, Yahoo, Facebook, Twitter, Weather.com and Instagram. These web services were selected based on their popularity and support for HTTP2 clients (browsers). Facebook, Twitter and Weather.com were eventually omitted from the final experiment corpus: the network traffic generated by Facebook was highly optimized, and therefore, it was assumed that it would not introduce any interesting differences in the payload sizes. Twitter on the other hand was deemed too lightweight (consisting of mostly text-based content), while Weather.com suffered from peculiar latency issues when transmitting images over HTTP2 as shown by Figure 4.



weather.com (HTTP2)

🟧	187	js	16,015.24 KB	46.23 s
🟪	241	images	2,696.50 KB	108.13 s
🟦	42	html	1,229.19 KB	6.82 s
🟦	22	css	803.79 KB	2.23 s
🟦	15	fonts	154.72 KB	1.17 s
🟥	44	xhr	23.34 KB	9.52 s
⬜	33	other	7.66 KB	13.72 s

Size: 20,930.45 KB
Time: 187.83 seconds
Cached responses: 0
Total requests: 584

weather.com (HTTP1.1)

🟧	190	js	15,753.43 KB	45.20 s
🟪	249	images	2,886.33 KB	36.55 s
🟦	43	html	1,226.29 KB	10.62 s
🟦	20	css	799.85 KB	2.79 s
🟦	13	fonts	130.09 KB	0.81 s
🟥	43	xhr	35.08 KB	8.36 s
⬜	34	other	6.43 KB	8.86 s

Size: 20,837.49 KB
Time: 113.19 seconds
Cached responses: 0
Total requests: 592

Figure 1: Weather.com was excluded from the final experiment corpus because it exhibited abnormally long response times when image assets were fetched over HTTP2. This was done the prevent introducing bias to the energy consumption measurements.

The experiment was conducted using Firefox (version 42.0) as no other publicly available browsers support manually disabling HTTP2 support, as of this writing. The implementation of a robust and reproducible experiment environment required automation of the navigation logic. Browser automation frameworks Appium[1] and Mozilla's Robocop[2] were originally considered for this task. However, Appium lacked support for the Firefox browser, and Robocop – Mozilla's internal tool for testing Firefox on mobile – missed proper docu-

[1] Appium - `appium.io`
[2] Robocop - `https://wiki.mozilla.org/Auto-tools/Projects/Robocop`

mentation. Thus, an alternative approach – including the use of Android's low level input API – was taken. The utilization of the input API was inspired by an open-source energy consumption measurement framework called GreenMiner [2].

The Android input API allows for the emulation of device gestures, and consequently, automation of browser navigation. The trade-off of this approach is that it requires root access to the device. Therefore the Samsung S4 used for the experiment was rooted using ClockworkMod recovery (version 6.0.4.7) and CyanogenMod (version 12), which ships with Android 5.1.1. The upside of installing a custom ROM – a customized OS image installed on the device's ROM memory – like CyanogenMod is that any pre-installed Google services or bundled Samsung software were not installed in the phone. This helped in minimizing any unwanted power consumption generated by services running in the background unknowingly.

Monsoon Solutions' Power Monitor[3] was used for collecting the power consumption data. The negative and positive terminals of the smartphone battery were connected to the power monitor using copper wire. The two reading terminals of the Samsung S4 battery were isolated to allow the smartphone to be started using external power (the power monitor). Output voltage set to 3.7V as per Samsung S4 battery information.

For the test setup most smartphone features were disabled. This ,includes automatic brightness, notifications, sounds, vibrations, locking of the screen and even battery information. The only main features enabled were wifi, airplane mode and developer mode. Further the screen was set to maximum brightness. So basically only the debugging via adb and networking via wifi was activated. Wifi is the only type of networking used in the setup for the following reason. Different types of connections were tested and in the testing location 2G resulted in a high paket loss, which made 2G incomparable to the other types. While 3G and 4G did not display any difference in latency and package loss, than the wifi connected to the aalto-open network. This may be due to the setup station being located near the mobile providers antennas, which are on the roof of the CS building. For this reason only wifi was used in the testing setup. Another factor that might have an influence on the latency is that all readings were taken in the evening, when the building was nearly empty. This results in low latency, since all networks have low loads in general. On the other hand testig during the day might lead to biased results because of higher load fluctuations in the networks.

test setup (testbed environment) disable some smartphone features disable browser features (caching) network environment (why WiFi and not 2G, 3G, latency table in Google Drive) sampling frequency 5000Hz

test run procedure steps duration

list relevant test variable values

[3]`https://www.msoon.com/LabEquipment/PowerMonitor/`

1.3 Results

RESULTS baseline power OS: 1 W
app baseline power with Browser: 1.8 W
mean power consumption +- stdev smoothed graph (3 runs for both) ”aggregated” graph (average of every 1s segment (5000samples)) annotation network request pie chart

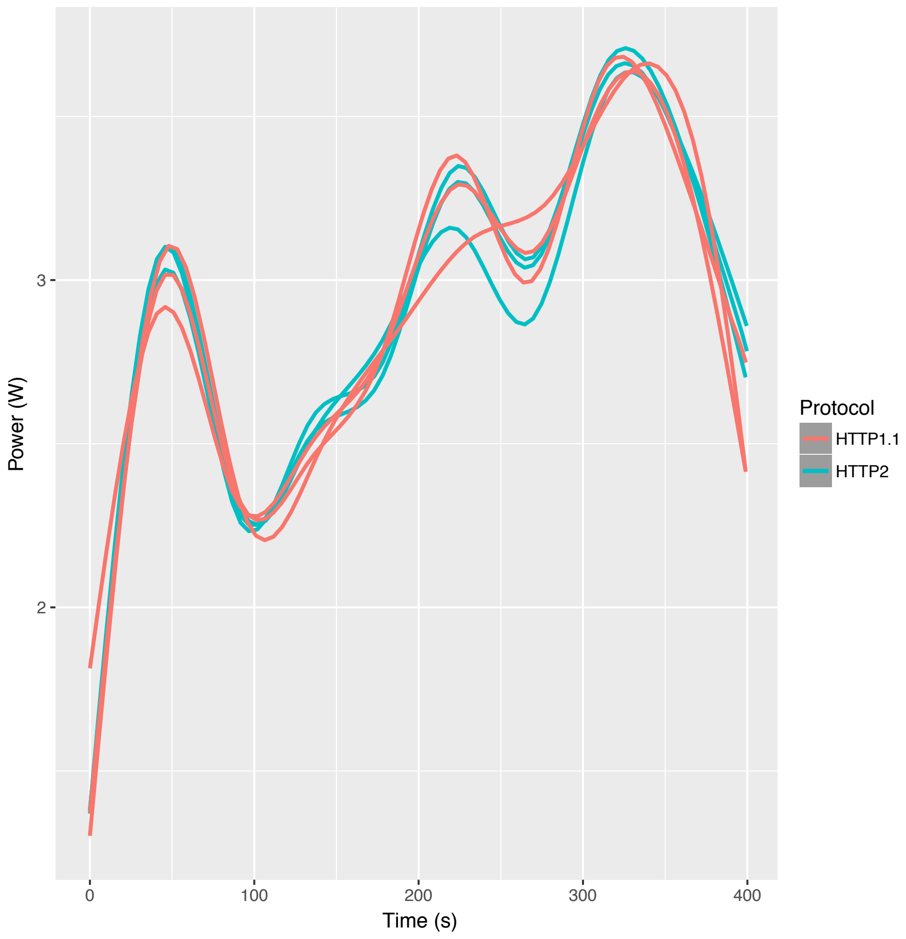


Figure 2: smoothed out power levels of all test runs in HTTP2 and HTTP1

1.4 Discussion

DISCUSSION talk about ENERGISE (`https://github.com/ds4se/chapters/blob/master/abramhindle/energymining.md#Hindle2014`) interpretation of results (explain A-E) outline for part II

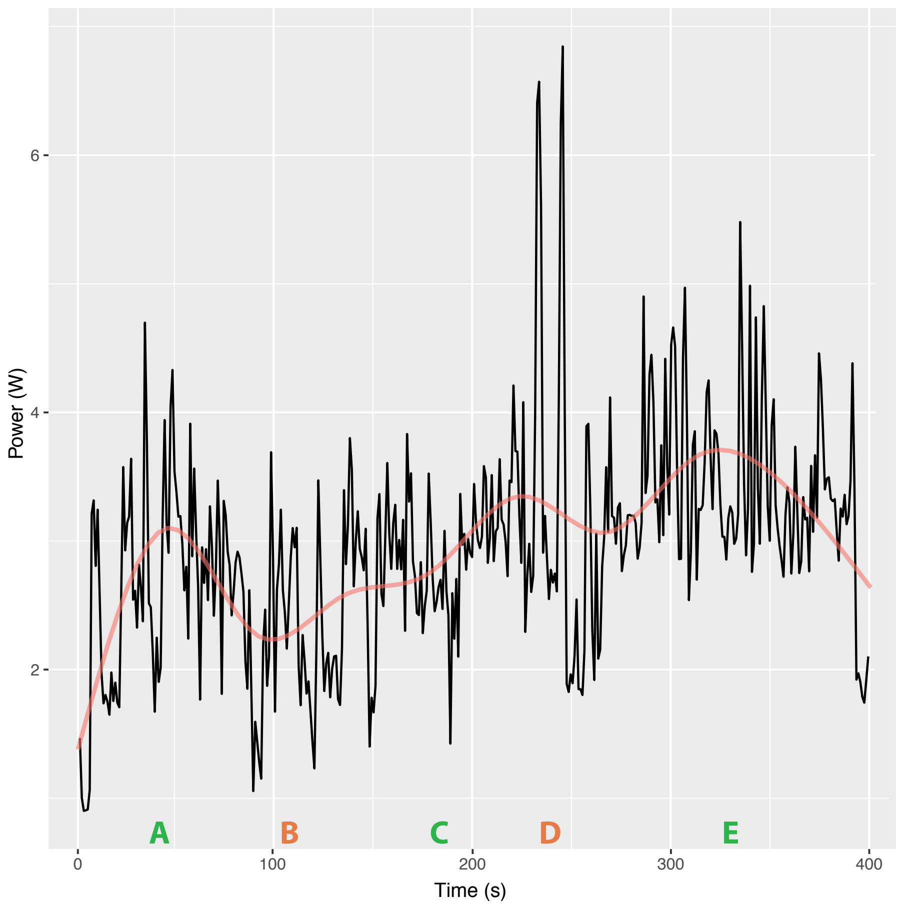


Figure 3: Reading of power levels compared to smoothed out readings in a HTTP/1.1 example

2 Part II

2.1 Proposal for Improvements

2.2 Conclusions

References

[1] Google Inc. SPDY Performance on Mobile Networks, 2016. `http://developers.google.com/speed/articles/spdy-for-mobile`. Accessed 29.04.2016.

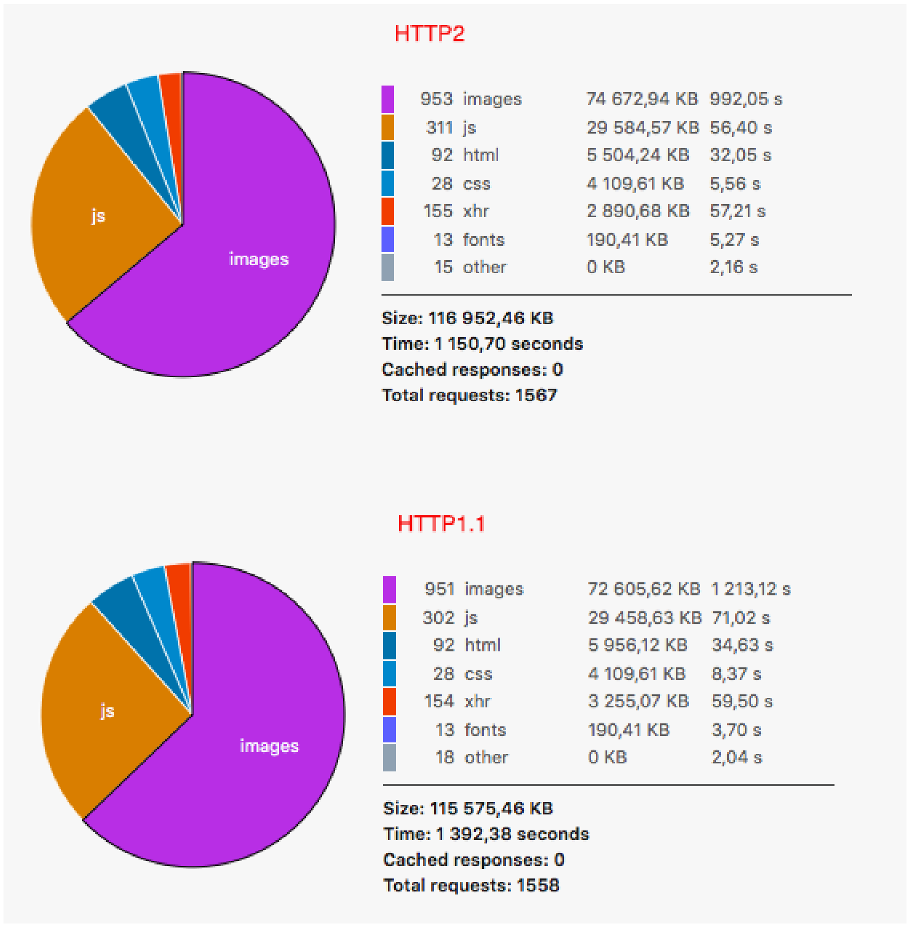
[2] Abram Hindle, Alex Wilson, Kent Rasmussen, E. Jed Barlow,

	HTTP1.1	**HTTP2**
Mean (W)	2.987	2.995
Std.dev (W)	0.984	1.007
Total (J)	1114.2	1117.2

Table 1: Energy consumption of HTTP1.1 and HTTP2

Web service	**RTT (ms)**	**RTT std. dev (ms)**	**Packet loss**
Yahoo	158	32	4.7%
Instagram	116	10	0.3%
Flickr	132	12	0.3%

Table 2: Latency of our websites



HTTP2

🟪	953	images	74 672,94 KB	992,05 s
🟧	311	js	29 584,57 KB	56,40 s
🟦	92	html	5 504,24 KB	32,05 s
🟦	28	css	4 109,61 KB	5,56 s
🟥	155	xhr	2 890,68 KB	57,21 s
🟦	13	fonts	190,41 KB	5,27 s
⬜	15	other	0 KB	2,16 s

Size: 116 952,46 KB
Time: 1 150,70 seconds
Cached responses: 0
Total requests: 1567

HTTP1.1

🟪	951	images	72 605,62 KB	1 213,12 s
🟧	302	js	29 458,63 KB	71,02 s
🟦	92	html	5 956,12 KB	34,63 s
🟦	28	css	4 109,61 KB	8,37 s
🟥	154	xhr	3 255,07 KB	59,50 s
🟦	13	fonts	190,41 KB	3,70 s
⬜	18	other	0 KB	2,04 s

Size: 115 575,46 KB
Time: 1 392,38 seconds
Cached responses: 0
Total requests: 1558

Figure 4: The better latency of HTTP2 comes with the price of performance.

Joshua Charles Campbell, and Stephen Romansky. Greenminer: A hardware based mining software repositories software energy consumption framework. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014, pages 12–21, New York, NY, USA, 2014. ACM.

[3] Q-Success. Usage of HTTP/2 for websites, 2016. `http://w3techs.com/technologies/details/ce-http2/all/all`. Accessed 30.04.2016.

[4] Varun Sapra Shaiful Alam Chowdhury and Abram Hindle. Is HTTP/2 More Energy Efficient Than HTTP/1.1 for Mobile Users? *PeerJ Preprints*, 2015.

[5] Daniel Stenberg. HTTP2 Explained. *ACM SIGCOMM Computer Communication Review*, 44(3):120–128, 2014.