Impacts of the HTTP/2 protocol for large scale web environments

Martin Leucht, James Gratchoff
Master SNE, UvA

March 11, 2015

1 Introduction

HTTP/1.1 is present since 1999 and it is such a big protocol and has been reviewed so many time that the IETF has split the original RFC (2616) into six different ones (7230-7235) describing the whole protocol in details. After more than 15 years of use it was time for a change. All the tricks not to slow the HTTP/1.1 protocol can be forgotten as on February 18th 2015, the specification for the new HTTP protocol HTTP/2 (and HPACK), has been formally approved by the IESG and is on the way to become an RFC standard. The main focus during the development period of the successor of HTTP/1.1 was to improve the performance, and thus provide a better user experience. The performance improvement is mainly based on how the packets are sent over the wire within a HTTP/2 session. HTTP/2 data is sent in binary format and is based on a multiplexing mechanism that allows a single connection for parallelism. Concerning security, HTTP/2 will not make the use of TLS mandatory. However, leading browsers firms, Mozilla Firefox and Google Chrome have already mentioned that HTTP/2 will only be implemented over TLS.
There are already web client and server implementations that support the final HTTP/2 specification. This research is intending to show what are the impacts of implementing HTTP/2 with TLS compared to HTTPS in a large environment.

2 Research Questions

To conduct the research the following research questions have been formulated:

How do the new features of the HTTP/2 protocol improve the performance for frequently visited webpages/webserver?

That question can be narrowed down to some sub questions:

- What are possible drawbacks that can occur for large web service providers when switching from HTTP/1.1 to HTTP/2 ?
- What is the predictable impact that could be related to changes in the infrastructure of Web service providers?
- What is the difference in bandwidth utilization between HTTP/1.1 and HTTP/2 considering the size ratio of header and data for multiple concurrent sessions on a server?
- How much can HTTP/2 decrease the amount of data when considering header compression and using multiple streams within one TCP Session compared to HTTP/1.1 on a server ?
- What is the impact regarding the response time experienced by a client when the number of concurrent requests / clients increases seen from three different geographical locations?
- What is the impact to the load and CPU utilization to the server when conducting the benchmark tests?

3 Scope

The scope of this research is to conduct benchmarks of the HTTP/1.1 and HTTP/2 protocol, using different static and dynamic parameters and compare them among each other. Static parameters

are the distance between client/server (RTT) and the page size, whereas dynamic parameters are the number of requests and the number of concurrent clients. Therefore we will implement three different HTML pages with different sizes and benchmark them from three different geographical locations. The test will be implemented using the secured version of the protocols with TLS enabled. We do not focus on new features of the HTTP/2 protocol like server push capabilities and flow prioritization.

4 Literature review

4.1 HTTP/1.1 drawbacks

Where is http/1.1 lacking in terms of performance and protocol implementation.

4.2 HTTP/2 improvements

What are the reasons to move to http/2 and how did it improve http/1.1.

4.2.1 Binary format

4.2.2 Multiplexing and priorities

4.2.3 Header compression

4.2.4 Flow control

4.2.5 Server push

4.3 Related work

The new HTTP2 protocol has been based on the SPDY protocol developed mainly by Google. As shown by Google [1], the SPDY protocol meets its expectations by reducing the loading time of web pages by 55%. Other people have tried to look into the protocol and one of the most interesting analysis has been done by Servy[2]. Servy evaluated the performance of the web servers implementing the SPDY protocol comparing it to HTTP/1.1 and HTTPS. The load testing tool used for this benchmark was the NeoLoad 4.1.2. His results showed that the implementation of SPDY increases by a factor of 6 the number concurrent of users possible before errors start showing up in comparison to HTTP and HTTPS. A contradictory study showing some boundaries of implementing SPDY has been done by Podjarny[3]. He shows that most of the websites use different domains and as SPDY works on a per-domain basis it does not necessarily help it to be faster. Finally, Wang et al.[4] have investigated the performance of SPDY for the improvements of the protocol compared to HTTP/1.1. This study highlights that SPDY is much faster since its benefits from the single TCP connection mechanism. However, they also mention that SPDY degrades under high packet loss compared to HTTP. Concerning the new standard HTTP/2 a few benchmarks have been performed by the creators of different client/server platforms. They reach the same conclusion for SPDY.
However, studies on comparisons between HTTP/2 (draft-ietf-httpbis-http2-14 [8]) and HTTP/1.1 with regards to concurrent clients and increasing amount of requests in different geographical locations and different page sizes or amount of elements a webpage contains, have not been conducted yet.

5 Requirements

In order to conduct this research we use one server that runs a webserver and responds to both versions of HTTP, HTTP/2 and HTTP/1.1 [7]. That instance will run on of our student servers. AWS medium-instances, provided by the Amazon Elastic Compute Cloud (EC2) [5] infrastructure, will be used as clients to perform several benchmark tests that try to cover real life scenarios.

6 Method

For our research we will setup three AWS Amazon micro instances (Europe/Asia/North America) that will act as HTTP clients (HTTP/1.1 and HTTP/2). The server itself, that will serve HTTP/1.1 and HTTP/2 requests is located in the OS3 lab and is one of our student server. The server will provide three different web pages that have different sizes (small/medium/large). As a server software for testing HTTP/2 (h2c-14 [8]) requests we will use nghttp [9]. For HTTP/1.1 requests we will use Apache2 [10]. In order to benchmark HTTP/2 we will use h2load [11] and for HTTP/1.1 Apache Benchmark (ab [12]). Despite the will not to use different benchmarks tools to minimize the repercussions on the results, we decided to use these tools because both provide similar output (data/header size ratio, requests per second, mean time for each request, etc) and are capable to simulate concurrent sessions.

7 Implementation

Three HTML of different sizes have been created to be compared. These pages reflect the sizes that can be encountered in most common websites and has been derived from top 100 websites statistics [13]. The pages size are:

title	size (bits)	number of requests
small.html	20458	3
medium.html	624048	8
large.html		49

Other fixed parameters have been set-up: $explanation of the other parameters$ Dynamic parameters $What are the parameters we will be playing with$

In order to have relevant results, the tests have been performed ten times and the average results will be presented in the next section. In order to make the benchmarking easier and more coherent some scripts have been implemented. $explanation of the scripts written by Martin here$

8 Results analysis

9 Conclusions

10 Further Work

References

[1] A 2x Faster Web. (2009). [online] Chromium Blog. Available at: http://blog.chromium.org/2009/11/2x-faster-web.html [Accessed 20 Feb. 2015].

[2] Servy, H. (2015). Evaluating the Performance of SPDY-enabled Web Servers. [online] Neotys.com. Available at: http://www.neotys.com/blog/performance-of-spdy-enabled-web-servers/ [Accessed 20 Feb. 2015].

[3] Podiatry, G. (2015). Guy's Pod Blog Archive Not as SPDY as You Thought. [online] Guypo.com. Available at: http://www.guypo.com/not-as-spdy-as-you-thought/ [Accessed 20 Feb. 2015].

[4] Wang et al. (2014). How Speedy is SPDY?, 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14). [online] usenix.org. Available at: https://www.usenix.org/system/files/conference/nsdi14/nsdi14-paper-wang_xiao_sophia.pdf [Accessed 20 Feb. 2015].

[5] Amazon Elastic Compute Cloud (Amazon EC2) (2015). Available at: http://aws.amazon.com/ec2/ [Accessed 20 Feb. 2015].

[6] Firefox Beta Notes (2015). Available at: https://www.mozilla.org/en-US/firefox/36.0beta/releasenotes/ [Accessed 20 Feb. 2015].

[7] HTTP/2 client/server implemenations (2015). Available at: https://github.com/http2/http2-spec/wiki/Implementations [Accessed 20 Feb. 2015].

[8] Hypertext Transfer Protocol version 2 draft-ietf-httpbis-http2-14 (2014). Available at: https://tools.ietf.org/html/draft-ietf-httpbis-http2-14 [Accessed 4. March 2015]

[9] HTTP/2 experimental server (2015). Available at: https://nghttp2.org/documentation/nghttpd.1.html [Accessed 4. March 2015]

[10] The Apache HTTP Server Project (2015). Available at: http://httpd.apache.org/ [Accessed 4. March 2015]

[11] Benchmarking tool for HTTP/2 and SPDY server (2015). Available at: https://nghttp2.org/documentation/h2load.1.html [Accessed 4. March 2015]

[12] Apache HTTP server benchmarking tool (2015). Available at http://httpd.apache.org/docs/2.2/programs/ab.html [Accessed 4. March 2015]

[13] Httparchive.org, (2015). HTTP Archive - Trends. [online] Available at: http://httparchive.org/trends.php [Accessed 11 Mar. 2015].