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

Martin Leucht, James Gratchoff
Master SNE, UvA

February 20, 2015

1 Introduction

Recently, the specifications for the new HTTP protocol HTTP/2 (and HPACK), the successor of the HTTP/1.1 protcocol have been formally approved by the IESG and will become an RFC standard soon. The main focus during the implementation period of the protocol was to improve the performance, and thus providing a better user experience. The performance improvement is mainly based on how the packets are send over the wire within a HTTP/2 session. HTTP/2 data is send in binary format and is based on multiplexing mechanism that allows a single connection for parallelism.
There are already web client and server implementations that support the final HTTP/2 specification.

2 Research Questions

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

- Are there specific usecases where the new features of the HTTP/2 protocol will provide major enhancements compared to the HTTP/1 protcol in large scale environments?
- How can we measure such performance improvements that are proposed by the new protocol?
- What are possible drawbacks that can occur for large web service providers when switching from HTTP/1.1 to HTTP/2 ?

- What are predictable impacts that are related to changes in the infrastructure of Web service provider?

3 Related work

The brand new IETF accepted HTTP2 protocol has been based on the SPDY protocol developed mainly by google. As shown by Google [?], the SPDY protocol is answering his expectation 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 is Servy's post. Servy evaluated the performance of the web servers implementing the SPDY protocol comparing it to HTTP1.1 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. The fact that SPDY is using a single connection for all requests induce that the clients are using one worker instead of six in HTTP/S. That makes the server able to handle more users with the same amount of worker. Servy also looked into the repercussion of the SPDY implementation in terms of CPU and memory consumption at the server side. Compared to HTTP, SPDY requests consume less memory but more CPU. However compared to HTTPS SPDY requests consume less memory and CPU usage. A contradictory study showing some boundaries of implementing SPDY has been done by Podjarny[?], he shows that most of the website uses different domains and as SPDY works on a per-domain basis it does not necessarily help it to be faster. Finally, Wang et al.[?] have investigated the performance of SPDY for the improvements of the protocol compared to HTTP. This study highlight that SPDY is much faster as benefiting from the single TCP connection mechanism. However they also mention that SPDY degrades under high packet loss compared to HTTP. Concerning the new standard HTTP2 a few benchmark have been done by the creator of the different client/server platforms. They all fall down in the same conclusion for SPDY. However no study shows he impact on the infrastructure.

4 Requirements

In order to conduct this research we need multiple server instances that run a webserver instance which respond to both versions of HTTP, HTTP/2 and HTTP/1.1 [?]. That instances will run on our student servers and on microinstances, using the Amazon Elastic Compute Cloud (EC2) [?] infrastructure to cover real life scenarios (e.g. RTT variances using different regions). Furthermore web client software that has already implemented HTTP/2 will be used [?].

5 Scope

6 Method

First, several webserver instances will be implemented in order to have multiple HTTP/2 and HTTP/1.1 enabled environments. Second, those instances will serve Web sites that have different characteristics (number of files, size of files) and different geographical locations to cover e.g. different Round Trip Times (RTTs). Finally we will conduct benchmarking based on that infrastructure to outline the improvements or drawbacks HTTP/2 provides in comparison to HTTP/1.1.

7 Expected results

References