Summary. This paper discusses the influence of the HTTP protocol on the performance of Web applications, ranging from HTTP/1.1 through SPDY, and ending with HTTP/2. The custom study of the influence of HTTP/2 on the user-perceived start time of the exemplary Single Page application in simulated Cable, DSL, LTE and 3G networks is described.

Keywords: HTTP/2, user-perceived web application performance

Streszczenie. W artykule omówiono wpływ protokołu HTTP na wydajność aplikacji webowych, począwszy od HTTP/1.1 poprzez SPDY, a kończączyjszy na HTTP/2. Opisano autorskie badania wpływu protokołu HTTP/2 na postrzegany przez użytkownika czas uruchamiania przykładowej aplikacji Single Page w symulowanych sieciach Cable, DSL, LTE i 3G.

Słowa kluczowe: HTTP/2, postrzegana przez użytkownika wydajność aplikacji webowej
1. The influence of HTTP protocol on the performance of web applications

1.1. Introduction

The specification of the commonly used HTTP/1.1 was made in 1997, and although a lot has changed since then, the fundamental bases of its performance remained unchanged. This has forced developers to design and implement web applications in such a way that the negative influence of the imperfection of the protocol on performance was as small as possible. It has led to the creation of a set of the so-called good practices which improve the speed of operation of the application:

- The number of queries to DNS servers should be as small as possible,
- The number of HTTP requests should be as small as possible,
- A distributed network of servers should be used (Content Delivery Network) – the server is then physically closer to the client, which results in the reduction of delay,
- Multiple requesting of the same resources should be prevented through the use of a cache.
- Resources should be compressed (gzip),
- HTTP redirects should be avoided.

The above principles stem directly from the fact that the transmission delay has the greatest influence on page load time and following them reduces the number of requests between the client and the server. They are – therefore – universal patterns that will always positively affect the performance of a web application. In addition, however, a number of other principles have been developed, which are basically to bypass a problem, not to provide a solution, for example domain sharding, or concatenation, composition and inlining of resources.

A major influence on improving the performance of web applications, as perceived by users, was the increase in the size of the Initial Congestion Window in TCP standard. A group of scientists working for Google conducted a study on the influence of the value on delays, which led to the creation of a document which suggests increasing that value from the current 3 to 10, which resulted in improved performance through the reduction in the Round Trips needed to send the complete resource from the server to the client [1]. The change was actually introduced in April 2013 (in RFC 6928 document) and significantly contributed to the improvement of web applications’ performance, but it still did not solve many fundamental HTTP/1.1 problems.
1.2. SPDY vs. HTTP/2

In 2009, Google made available the source code and documentation of experimental SPDY protocol. The project objectives as set by its developers were:

- Reduction in page load time by 50%,
- Simplicity of implementation – TCP use requires no changes in the existing network infrastructure,
- No need to make changes to existing sites by their creators – the only required changes are the changes on the client's and the server's side,
- Making the work on the protocol available to all the interested to jointly solve the problem of network delays – work on the principles of creating an open source by the community and professionals.

To achieve the intended goals, the following technical assumptions were identified:

- Enabling sending multiple HTTP requests simultaneously using a single TCP connection,
- Reducing bandwidth usage through the compression of headers and the elimination of the unnecessary ones,
- Creating a protocol which is efficient and easy to implement,
- Using SSL as a transport protocol for better compatibility with the existing network infrastructure – despite SSL overhead in terms of delays, the creators motivated their selection by the conviction that the future of the web is secure connections,
- Allowing the server to initiate connections with the client and pushing data whenever possible [2].

Shortly after the SPDY announcement, its authors published the documentation, source code, and preliminary studies, which showed that it was possible to reduce page load time by an average of 55% [3].

The protocol began to gain popularity, and three years later, in 2012, it was already supported by Chrome, Opera and Firefox as well as implemented on the server side of the most popular websites, such as Google, Facebook or Twitter. As it turned out, thanks to SPDY, the actual performance benefits were achieved. Observing these changes, the HTTP Working Group released the first version of the new HTTP/2 in early 2012. Following the discussions on the document, it was decided that the starting point for the work on the new protocol will be SPDY specification [4].

The first version of HTTP/2 documentation [5] (then denoted as HTTP/2.0) lists its most important assumptions:
• Reducing delays as perceived by users in relation to HTTP/1.1 in a visible and measurable manner,
• Solving the problem of head-of-line blocking present in HTTP/1.1,
• No need to open multiple connections to the server to handle requests simultaneously, which, in turn, leads to better use of TCP, especially in the context of network congestion control mechanisms,
• No semantic changes in relation to HTTP/1.1 in terms of methods, status codes, URI identifiers and header fields (where possible),
• Clearly defining the interactions between HTTP/2.0 and HTTP/1.1, especially for intermediaries,
• Clearly defining the places of the potential protocol extension and the rules of their proper use.

It can be noticed that the objectives of the new protocol correspond with the familiar HTTP/1.1 problems and weaknesses, assuming no requirements to change lower level protocols.

SPDY protocol was still being developed alongside HTTP/2, constituting a research field of some kind and the driving force for the new protocol. Many of the functionalities implemented and explored in SPDY were taken over by HTTP/2 [4]. With the introduction of HTTP/2 in 2015, SPDY ceased to be developed.

Currently, most websites are optimized for HTTP/1.1, but the support for HTTP/2 is constantly growing. Most of the popular sites already support it. On November 16, 2016, almost 149,000 domains offered full support, more than 180,000 partial support and nearly 242,000 heralded their support [6].

On the browser market, all the major developers (Chrome, Opera, Firefox, IE 11, Edge and Safari) introduced HTTP/2 support at the end of 2015.

1.3. Experiments with HTTP/2

Golang programmers [7] implemented an interesting environment for on-line experimentation. At http2.golang.org the load times of an image cut into 180 parts for HTTP/1.1 and HTTP/2 can be compared for the delay values of 0 ms, 30 ms, 200 ms and 1 second. A significant improvement in load times through the use of HTTP/2 can be observed. The authors conducted an experiment in which, for the 200 ms delay and HTTP/1.1 the DOM object loaded in 1.288 ms, and the entire image in as much as 11.451 ms. In HTTP/2 the DOM object loaded in 1.220 ms and the image in 1.875 ms, which is more than 5 times faster (results for 10 samples for each protocol).
The above experiments do not meet the requirements of scientific research, but they give a picture of performance improvement achieved through the use of HTTP/2 without making any additional changes and improvements.

In 2014, Saxcé, Oprescu and Chen [8] performed experiments with HTTP/2. They adopted a methodological approach, taking the newly introduced features into consideration: compression, multiplexing, server push and priority. They highlighted that HTTP/2 is negatively impacted by packet loss which is characteristic of cellular networks. They also experimented on server push and priority to understand their behavior.

Varvello et al. [9] built a measurement platform that monitors HTTP/2 adoption and performance across the Alexa top 1 million websites on a daily basis. They reported their findings from an 11-month measurement campaign (November 2014 ÷ October 2015). For the most part, websites do not change as they move from HTTP/1.1 to HTTP/2 – current web development practices like inlining and domain sharding are still present. These practices make HTTP/2 more resilient to losses and jitter. In all, they find that 80% of websites supporting HTTP/2 experience a decrease in page load time compared with HTTP/1.1 and the decrease grows in mobile networks.

In 2015, Lönn and Stenberg [10] conducted an experiment designed to compare the performance of HTTP/1.1 and HTTP/2. The authors downloaded resources from amazon.com pages, put them on a local server and in this way simulated server's responses under controlled conditions. The average page load time was measured, depending on the delay. For HTTP/2 protocol the results were by 50÷70% better.

Han, Hao and Qian [11] introduced MetaPush – a novel server push framework aiming at reducing web page load time. The key idea is to strategically leverage server push, a built-in feature in HTTP/2, to preemptively push web pages' metadata, which can later be leveraged by the client to early-fetch critical resources. They demonstrated MetaPush is particularly suitable for cellular networks, providing negligible bandwidth overhead (around 0.4%), improved page load time (up to 45%), and reduced radio energy utilization (up to 37%) over HSPA+ networks.

In 2016, Stepnjak and Nowak [12] analyzed performance of SPA-based Web systems. The research revealed that the server push mechanism of HTTP/2 protocol implemented in the mod_http2 module of Apache web server slowed down the process of loading the Single Page application and, at the time of its development, was not suitable to increase SPA performance.

From the analysis of the publications made by the authors it results that there has been no reliable scientific studies of the influence of HTTP/2 on the user-perceived start time of
application. It was the authors' motivation to carry out their own research described in the next sections.

2. Tools

2.1. WebPagetest

WebPagetest is used for measuring and analyzing the performance of web pages. It is an open source project supported by Google to increase the speed of web applications. It was originally created as an internal tool for AOL so that developers could explore how efficiently their pages work in the end-user's eyes. Currently, the source code of the tool is available on the GitHub platform under the BSD license. Anyone can run a private instance of WebPagetest [13-15].

The location of the WebPagetest in relation to other computer software is shown in Fig. 1. The role of DummyNet is described in the next section.

WebPagetest offers a number of metrics. According to the authors, four of them are important from the point of view of planned research:

- **Load Time** – full page load time. It is one of the most researched performance metrics. Even though it is not the most essential in the context of user-perceived performance, its omission in the research would be an oversight and the results would be incomplete,

- **Time To First Byte** (TTFB) – the time measured between sending the first request to the server and receiving the first byte of the response. Measured as a control metric of appropriate server configuration, research results should be similar for the same network emulation settings, regardless of the protocol or pushing resources,

- **Time To Start Render** (TTSR) – the time elapsed before the appearance of the first visible web page item on the screen. It is an important metric in terms of performance as perceived by the user. The faster the user sees the loading page elements, the greater is the
feeling that the page loading process runs smoothly. Too long time, during which the page remains blank, may make the user leave the page,

- **Speed Index** – the most important of the assessed metrics. Its aim is to reflect the speed of loading the visible elements of the page. Its low value means better webpage performance. It stems from the need to measure the user experience in the context of webpage performance. To calculate this metric, the degree of filling the screen and the time are considered. The Speed Index is the "area above the curve" calculated in ms and using $0.0 \div 1.0$ for the range of visually complete page (1). An example area is shown in Fig. 2.

$$\text{Speed Index} = \int_{0}^{\text{end}} 1 - \frac{\text{VC}}{100},$$

(1)

where \(\text{end}\) denote end time in milliseconds and \(\text{VC}\) denote % visually complete page.

![Graph of Speed Index](image)

Fig. 2. Graphic representation of the Speed Index metric [13]
Rys. 2. Graficzna reprezentacja metryki Speed Index [13]

### 2.2. DummyNet

Originally designed for testing networking protocols, **DummyNet** is a live network emulation tool simulating queue and bandwidth limitations, delays, packet losses, and multipath effects. In addition it implements various scheduling algorithms and can be used on the computer running the user's application, or on external boxes acting as routers or bridges [16, 17].

The application runs within the operating system and intercepts selected traffic in encounters through the network stack. It sends packets to objects called pipes implementing a set of queues, a scheduler, and a link. In this way, it simulates, among others, bandwidth, delay, loss rate, queue size and scheduling policy.
Traffic selection is accomplished by means of the *ipfw* firewall, which is the main user interface for *DummyNet*. For example, the following commands can simulate an ADSL link on user's computer:

```
ipfw add pipe 1 in
ipfw add pipe 2 out
ipfw pipe 1 config bw 10Mbit/s delay 1000ms
ipfw pipe 2 config bw 1Mbit/s delay 1000ms
```

When *WebPagetest* integrates with *DummyNet*, it uses predefined connectivity profiles.

3. Research

3.1. The analyzed webpage

The webpage selected to be used to conduct experiments is a template for *Single Page* site type comprising of 20 resources with a total volume of 617 kB (Fig. 3).

```
index.html
 | \vendor\bootstrap\css\bootstrap.min.css
 | \vendor\font-awesome\css\font-awesome.min.css
 | \vendor\font-awesome\fonts\fontawesome-webfont.woff2
 | \vendor\magnific-popup\magnific-popup.css
 | \css\creative.min.css
 | \img\header.jpg
 | \img\portfolio\thumbnails\1.jpg
 | \img\portfolio\thumbnails\2.jpg
 | \img\portfolio\thumbnails\3.jpg
 | \img\portfolio\thumbnails\4.jpg
 | \img\portfolio\thumbnails\5.jpg
 | \img\portfolio\thumbnails\6.jpg
 | \vendor\jquery\jquery.min.js
 | \vendor\bootstrap\js\bootstrap.min.js
 | \js\jquery.easing.min.js
 | \vendor\scrollreveal\scrollreveal.min.js
 | \vendor\magnific-popup\jquery.magnific-popup.min.js
 | \js\creative.min.js
```

Fig. 3. Dependency tree of the assessed page's sources
Rys. 3. Drzewo zależności źródeł badanej strony

The template is from https://startbootstrap.com/. Its use was motivated by the popularity of *Single Page* applications. The choice of this particular template was motivated by its high number of downloads at the moment of research planning.

No changes in the source code were made. The *HTTP/2 push* mechanism was controlled directly by the configuration of the *Apache* server. All the external resources to which the
webpage sources referred were also downloaded and placed on the server used for research. In this way the influence of external factors on the results was minimized.

3.2. Research position

The research was carried out using two computers connected through the local network. Both the computers had Windows 7 operating system installed. Computer #1 was working as a server, while computer #2 was configured to allow the use of research tools.

On computer #1 2.4.18 version of Apache server was installed [18]. The decision was motivated by the popularity of this solution and the fact that from that version the full support for the module supporting HTTP/2 including the push mechanism was introduced. At the time of research, such support was not offered by either nginx server or Microsoft IIS.

The first experiments showed that all of the assessed metrics deteriorated when configuring a server using HTTP/2 module in comparison with the results obtained for HTTP/1.1.

![Diagram of research site configuration](attachment://research_site_config.png)

The probable cause of such behavior was the older version of the kernel (Windows 7), in which the value size of the TCP initial congestion window is set to 3 [4]. It made the effect of the TCP slow start much more evident in the case of communication via HTTP/2, in which only one TCP connection is opened, than in the case of HTTP/1.1, when a browser opens up to 6 parallel TCP connections. Therefore, the decision was made to use VirtualBox software to install a virtual instance of Ubuntu Server 16.10. This is version of the system with the size value of TCP Initial congestion window set to 10. In the case of Ubuntu it was also decided to use Apache in its 2.4.18 version. The final configuration of the research site is shown in Fig. 4.

On computer #2 a private instance of WebPagetest tool was installed. It allowed for greater flexibility in its configuration, such as no limitations in the amount of research to 10
repetitions and the ability to use the built-in BatchTool which enables writing simple scripts to automate experiments. The package also includes the DummyNet network traffic emulator. It was able to emulate any bandwidth or network delay during the research.

### 3.3. The conducted experiments

The decision was made to carry out research for 5 server configurations:

- HTTP/1.1 without optimization strategies,
- HTTP/2 without the push mechanism,
- HTTP/2 with pushing visible background image when a part of the page is loaded, and the associated file containing CSS rules,
- HTTP/2 with pushing all CSS files, and all files containing JavaScript (JS) scripts,
- HTTP/2 with pushing all CSS files, all JS scripts and background image visible when loading a part of the page.

Each of the above configurations was examined for 4 network emulation settings. Delay and bandwidth values are modeled on the original settings of the tool at [www.webpagetest.org](http://www.webpagetest.org):

- Cable – 5Mb/s bandwidth, 28 ms delay,
- DSL – 1.5Mb/s bandwidth, 50 ms delay,
- LTE – 12Mb/s bandwidth, 70 ms delay,
- 3G – 1.6Mb/s bandwidth, 300 ms delay.

For each combination 25 repetitions were performed. Only the first page loading was researched. Google Chrome version 54.0.2840.87 was used. As the web browsers support HTTP/2 only for encrypted connections, it was decided to examine HTTP/1.1 protocol for such a connection, too. For this purpose, an appropriate certificate was generated using OpenSSL software.

All the twenty-five item sets of results obtained for the four metrics were analyzed in the R environment in order to evaluate the normal distribution hypothesis. In view of the fact that for about half of the sets it was possible to reject the null hypothesis, it was decided to operate on median values in further analyses. Such an analysis is also supported by the fact that WebPagetest uses this measure when presenting the results (on the main screen the median run is always presented). All the median values obtained are presented in Fig. 5.
3.4. Analysis of results

3.4.1. Speed Index Metric

The research revealed that the Speed Index metric value was the highest in the case of HTTP/1.1 for networks with more bandwidth – Cable (5 Mb/s) and LTE (12 Mb/s). For networks characterized by lower bandwidth, i.e. DSL (1.5 Mb/s) and 3G (1.6 Mb/s) in the

![Graphs showing Speed Index, TTFB, and Load Time for Cable, DSL, LTE, and 3G networks.](image)

Fig. 5. Results for Cable, DSL, LTE and 3G networks
Rys. 5. Wyniki dla sieci Cable, DSL, LTE i 3G
case when among pushing resources no background image was found, the value of \textit{Speed Index} was lower during the evaluation of HTTP/1.1 than in the case of HTTP/2. But those were never relatively large differences – with the largest amounting to 10%.

At the same time for the two networks (DSL, 3G), the most prominent was the positive effect of pushing the background image. This phenomenon is due to the fact that the pushed resource loaded faster. This has a significant influence on the user experience – a large part of the visible area of the page is available to them earlier and focuses their attention. Giving higher priority to elements of the page (in the context of the loading order), with which the user can interact is a known technique to improve the comfort of use of a website. To this purpose, the push mechanism of HTTP/2 may be used.

\subsection*{3.4.2. Time To Start Render Metric}

The strategy of pushing CSS files and JavaScript scripts turned out to be the best for \textit{Time To Start Render} metric. Not much worse, and in some cases even slightly better times were obtained for HTTP/2 without the use of server push. The total volume of pushed resources for the said strategy was the smallest as it did not include the background image, which translated directly into the value of the discussed metric.

It should be noted that in the case of a lower bandwidth network (DSL and 3G) the value of \textit{Time To Start Render} metric was higher when pushing background image than in the case of research carried out for HTTP/1.1. This behavior is caused by the fact that the pushed resources were linked to the first server response. The background image file size is relatively large which in effect extends the time needed to download the resource. The lower the bandwidth the more noticeable the time extension. As the rendering starts after the first response is obtained, it eventually leads to longer \textit{Time To Start Render}.

\subsection*{3.4.3. Time To First Byte Metric}

The value of \textit{Time To First Byte} metric is stable when assessing a particular type of network, regardless of the protocol used or pushing resources, as expected. This situation can be attributed to the fact that the resources were pushed only after receiving client GET header for the resource index. The value of the said metric was only influenced by the parameters of the emulated network.

Noteworthy is the fact that the values of the metric are higher than in the case of research carried out initially on a server that was running under Windows 7 operating system. Extending the \textit{Time To First Byte} was probably caused by the overhead of the virtual instance of the Ubuntu Server operating system.
3.4.4. Load Time Metric

In the case of Load Time metric, regardless of the emulated network, the longest recorded loading times are evident for HTTP/1.1. When using HTTP/2 for all types of networks, with the exception of Cable, page load time when pushing resources was shorter than without the push mechanism. The configuration of this network (Cable) is characterized by the lowest value of delays (28 ms). The pushing of resources reduces the number of requests, which has a positive impact on the Load Time. The greater the delay, the greater the benefits. When the delay value is low, the gain may not be noticed. Nevertheless, Load Time is not expected to be longer while using server push. Therefore, to confirm the effect of the delay value on the page load time more detailed research should be carried out.

3.5. Conclusions

The majority of metrics researched in almost all cases improved when using HTTP/2 as compared with the evaluations conducted for HTTP/1.1. It should, however, be borne in mind that the researched site was in no way specifically optimized when carrying out experiments in the case of server configuration for HTTP/1.1. Exactly the same optimizations in the source code (i.e. CSS files at the beginning, JS scripts at the end) were used for both versions of the protocol and all the pushing strategies. Changes were made only in the server configuration.

Pushing the background image, visible when loading the rest of the page, had a positive influence on the Speed Index metric, especially for networks characterized by lower values of bandwidth. The push mechanism can be used as a tool to optimize web pages by pushing key resources that the user should see as soon as possible. At the same time, the deterioration of the value of the Time To Start Render metric when pushing bulky items, such as images should be borne in mind.

In the conditions under which the research was conducted, the use of HTTP/2 always had a positive influence on page load time compared to the time obtained for HTTP/1.1.

In the early stages of the implementation of HTTP/2 and the push mechanism, both on the browsers' and servers' side, there were problems [12]. The authors, however, did not observe similar issues. This may indicate the fact that manufacturers of such software improved their products for the use with HTTP/2 and the server push mechanism.
4. Summary and directions for further research

On the basis of the analysis of the results obtained it can be concluded that the greatest influence on the start time perceived by the user would be pushing CSS resources and background image file of the page. Using this strategy, we recorded a significant drop in the value of the Speed Index metric, especially for links with less bandwidth.

Research should be extended with experiments for browsers by other manufacturers, e.g. Mozilla Firefox, Internet Explorer, or Edge. It would also be reasonable to study the effect of pushing resources on the page load time for the network characterized by low values of delays (in order to remove the doubts described in paragraph 3.4.4.).

The experiments were conducted in an isolated environment and the characteristics of the network were emulated using DummyNet software used by the WebPagetest instance. It would be worthwhile to conduct research under the actual network conditions.

Although the development of implementation of HTTP/2 on both the server and the browsers, the push mechanism of resources is still part of an experimental protocol [19]. Replicating the research at a later time might confirm the influence of mature implementation of this functionality on performance. In addition, not all servers possess server push implementation (e.g. nginx). It would be worthwhile to conduct research at the moment of introduction of the pushing resources mechanism there.

BIBLIOGRAPHY


8. Saxcé H., Oprescu I., Chen Y.: Is HTTP/2 Really Faster than HTTP/1.1?. Proc. IEEE Conference on Computer Communications Workshops, INFOCOM, 2015, p. 293–299, doi.org/10.1109/INFCOMW.2015.7179400


Omówienie

W pierwszej sekcji przedstawiono tzw. zbiór dobrych praktyk wypracowanych w celu poprawy szybkości funkcjonowania aplikacji webowych wykorzystujących protokół HTTP/1.1. Wyspecyfikowano cele projektowe i założenia techniczne dla protokołu SPDY. Przedłożono najważniejsze założenia protokołu HTTP/2 oraz opisano znane autorom eksperymenty przeprowadzone względem protokołu HTTP/2.

Druga sekcja zawiera opis wykorzystanych przez autorów narzędzi: WebPagetest (rys. 1) – miernik i analizator wydajności stron webowych oraz DummyNet – symulator ograniczeń sieci komputerowych. Do badań autorzy wybrali cztery metryki oferowane przez WebPagetest: Load Time, Time To First Byte, Time To Start Render oraz Speed Index. Istotę tej ostatniej ilustruje zależność (1) oraz rys. 2.

Trzecia sekcja została poświęcona badaniom przeprowadzonym przez autorów. Znajduje się tam diagram przedstawiający drzewo zależności źródeł badanej aplikacji Single Page (rys. 3) oraz schemat konfiguracji stanowiska badawczego (rys. 4). Podczas eksperymentów zasymulowano działanie aplikacji w sieciach Cable, DSL, LTE i 3G. Wyniki zwizualizowano na rys. 5. Większość badanych metryk prawie we wszystkich przypadkach uległa poprawie przy użyciu protokołu HTTP/2 w stosunku do badań przeprowadzonych dla protokołu HTTP/1.1. Zauważono, że wypychanie obrazu tła, widocznego podczas wczytywania reszty strony, miało pozytywny wpływ na metrykę Speed Index, szczególnie dla sieci cechujących się gorszymi wartościami przepustowości. Należy jednak mieć na uwadze pogorszenie wartości metryki Time To Start Render przy wypychaniu dużych objętościowo elementów, takich jak obrazy.

W podsumowaniu zauważono, że badania należałyby rozwinać o eksperymenty dla przeglądarek innych producentów niż Google. Ponieważ nie wszystkie serwery mają zaimplementowany mechanizm server push, celowe byłoby przeprowadzenie badań w momencie jego pojawienia się.

Addresses

Jakub PROKOPIUK: Acturis Poland Sp. z o.o., ul. Rzeźnicza 32-33, 50-130 Wrocław, Poland, jakub.prokopiuk@acturis.com

Ziemowit NOWAK: Wrocław University of Science and Technology, Faculty of Computer Science and Management, ul. Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland, ziemowit.nowak@pwr.edu.pl