American Journal of Engineering Research (AJER)
e-ISSN: 2320-0847 p-ISSN : 2320-0936
Volume-6, Issue-4, pp-40-45
www.ajer.org
Research Paper Open Access

Performance evaluation of HTTP/2 in Modern Web and mobile Devices

Mukhtar Abdirahman Abdillahi[1], Ualikhan Dossetov[2], Ali Saqib[3]
[1, 2, 3](School of Computer Science and Engineering, Nanjing University of Science and Technology, P.R. China)

ABSTRACT : *http/2 is the new generation of protocol that Promises to fix http 1.x workloads by proposing new features and Help the speed of the web. Therefore, we try to discuss some current issues of the http 1.x, http/2 and http/2 over UDP. On the other hand, one of the big issues of the mobile devices is limited storage and processing capacity and short battery life time as a result of energy drain that is found when running programs that need intensive computations on the cellular devices. In this paper, we highlight whether http/2 can help to save energy consumption for the mobile devices and Improve performance of the web. We conducted the test on an iPhone device and open source server. We conclude our study that HTTP/2 has both better energy consumption and performance than HTTP/1.1.*

Keywords: *HTTP/2; HTTP/1.1; HTTP/1.0; Evolution; SPDY; UDP; QUIC; Energy Consumption*

I. INTRODUCTION

The evolution of web services into richer web pages introduces complexity, which has a repercussion on page load times. Indeed, in the late nineties, web pages were only made of text – with some color if you were lucky; but nowadays you deal with interactive web pages displaying hundreds of pictures and running multiple scripts. Every keystroke, every mouse movement is detected and impacts the rendering of the web page. This behavior has a cost, mainly because the traffic between the user and the server hosting the website gets really heavy [1].

The popularity of mobile devices (e.g., smartphones, and tablets) has dramatically increased, as millions of users are using these devices in their daily lives. These devices offer users the opportunity to have more computational power and even more capability in the palm of their hand than most users had on their desktop just a few years ago, it is reported that, as of 2014, more than 1.4 billion smartphones are being used globally [2], and that worldwide mobile data traffic grew approximately 70% [3]. That number was forecasted to reach 2.1 billion in 2016. Energy consumption, in addition to the mobile devices, has also become a subject of concern for large data centers consuming at least one percent of the world's energy [4]. HTTP has been the Internet backbone for a long time even with some known drawback and bottlenecks. In HTTP protocol (version 1.0), the client had to open a new connection on each request, after each response the connection should be closed. Tis affects the performance of the web page load due to a known TCP problem "TCP Slow Start issue" which is related to the requirement of round trips for each connection initiating and that leads to a slowness once the connection is opened. In this paper, the researchers show some new techniques that are implemented by several web client and server applications. These techniques include HTTP Pipelining, SPDY/HTTP2. HTTP pipelining has been implemented in several web browsers/web servers, and the result so far was better than HTTP 1.0, however, there are still some issues that not have been solved yet.

Therefore, the main aim of this paper was to list the drawbacks in HTTP 1.x from the saving Energy standpoint for mobile Devices, as well as performance perspective in addition to the other criteria. We will then discuss the new feature in HTTP 1.x that helps to improve the performance dramatically by presenting some benchmarks and technical details.

II. BACKGROUND STUDY: WEB PROTOCOLS AND WEB LATENCY

2.1 HTTP 1.0 and HTTP/2

The Hypertext Transfer Protocol is defined in the RFC as "an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through the extension of its request methods, error codes, and headers" [5].

HTTP has several bottlenecks such as the dependency on more than one connection for a parallel connection implementation. This leads to more problems such as extra round trips for each connection initiating (RTT)" [6]. The Hypertext Transfer Protocol (HTTP) was designed in the mid-nineties to communicate between browsers and web servers, has tried to adapt to the ever-changing use of the web. HTTP 1.0 was introduced in early 1996 when there was the onset of companies moving online for business. The popularity of use of HTTP has grown with over 75% of traffic on the internet being solely reliant on it [7]. HTTP 1.0 could only define up to 16 status codes which was a reserved number. The main limitation of using the 16 status codes was that there was poor resolution reporting that was noticed and thus there was the need to come up with the HTTP 1.1. HTTP 1.1 came with 24 status codes that were able to solve the previous limitations that HTTP 1.1 faced. Error reporting was done faster and there was easy detection of errors when they occurred [7].

Working on an experimental protocol called SPDY, which kept the HTTP semantics but solved the head of line blocking issue which limits HTTP/1.1. This resulted in several sub-versions (0.9, 1.0, 1.1) of the protocol. In 2009, Google started.

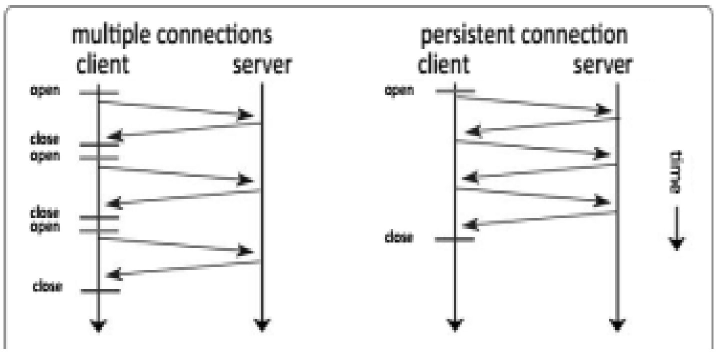

Figure 1: A comparison between HTTP 1.1 and HTTP 1.0 TCP connection: It shows the difference between reused connections and non-reused connection which is known as a persistent and non-persistent connection [8].

2.2 SPDY

Google recognized the degrading performance of Web applications [9], and in mid-2009 they announced a new experimental protocol called SPDY [10]. While still retaining the semantics of HTTP/1.1, SPDY introduced a framing layer on top of TLS persistent TCP connections to achieve multiplexing and request prioritization. It allowed SPDY to achieve one of its major design goal to reduce page load time by up to 50% [11]. SPDY reduced the amount of data exchanged through header compression, and features such as server push also helped to reduce latency. Despite its features and it being standardized, Google SPDY faced criticism [12]. It required TLS in its implementation. Organizations were hesitant to adopt it as it benefited Google Chrome Browser and was not an IETF standard. Moreover, its compression algorithm was found to be vulnerable to CRIME (Compression Ratio Info-leak Made Easy) and BREACH (Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext) attacks [12]. Consequently, as of March 2017, only 3.9% of the Web servers used SPDY [13]. It adds various features such as multiplexing of multiple requests, which is a parallel stream of request on a single TCP connection. It also adds a framing layer which is optimized for HTTP request-response multiplexing stream, with the keeping of backward compatibility of the current HTTP applications, so they can work over SPDY without affecting the applications that do not support SPDY protocol. SPDY, however, showed the need and possibility of a new protocol in place of HTTP/1.1 to improve Web performance.

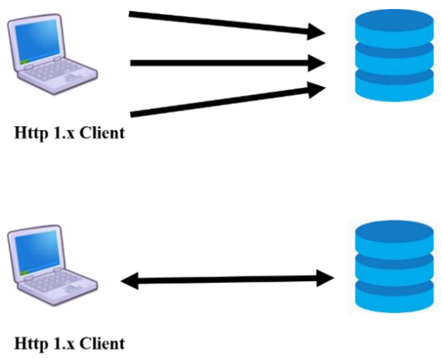

Figure 2: HTTP2.0 multiple connections vs. HTTP 1.x persistent connection

2.3 Http/2 and QUIC Protocol (HTTP over UDP)

In 2012, the Internet Engineering Task Force [14] decided to tackle the specification of a new version of HTTP, namely HTTP/2. The first HTTP/2 draft in 2012 was based on the SPDY protocol. Since then it has much evolved, always with the will to make HTTP/2 speed up the web page loading times. QUIC is Google's experimental, low-latency Internet transportation protocol over UDP, a protocol that is often used by gaming, streaming media and VoIP services. The name 'QUIC' stands for Quick UDP Internet Connection. UDP's (and QUIC's) counterpart in the protocol world is basically TCP (which in combination with the Internet Protocol (IP) makes up the core communication language of the Internet). UDP is significantly more lightweight than TCP, but in return, it features far fewer error correction services than TCP. This means that the sending server isn't constantly talking to the receiving server to check if packages arrived and if they arrived in the right order [15]. That's why UDP is great for gaming services. For the purposes of low overhead in order to reduce latency and if the server didn't receive your latest mouse movement, there's no need to spend a second or two to fix that because the action has already moved on. One wouldn't want to use it to request a website, though, because you couldn't guarantee that all the data would make it. Google aims to combine some of the best features of UDP and TCP with modern security tools under QUIC. On a typical secure TCP connection, it typically takes two or three round-trips before the browser can actually start receiving data. Using QUIC, a browser can immediately start talking to a server it has talked to before.

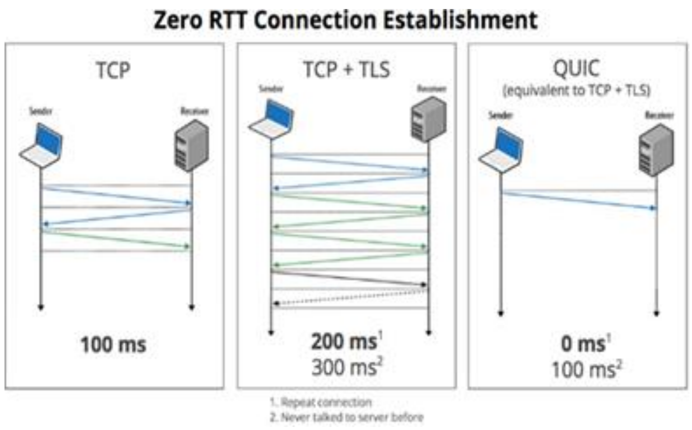

Figure 3: Showing a Typical TCP Connection and a QUIC Connection [15]

QUIC also introduces a couple of new features like congestion control and automatic re-transmission, making it more reliable that pure UDP. With SPDY, which later became the basis for the HTTP/2 standard, Google already developed another alternative protocol that had many of the same goals as QUIC, but HTTP/2 still runs over TCP and still runs into some of the same latency cost.

A. ***Multiplexing***: SPDY opens only one (or fewer compared to HTTP/1.1) TCP connection per sub-domain and the data stream is multiplexed over a single TCP connection. This avoids the Head of Line (HOL) blocking in resource level, reduces the time to open a new TCP connection, and reduces the impact of the TCP "slow start".

B. ***Header compression***: Sending redundant header information back and forth can be avoided by header compression. The size of request and response headers are reduced significantly by header compression.

C. ***Request prioritization***: All the high priority resources are requested earlier than other resources. This would allow web pages to render faster in bandwidth limited scenarios.

D. ***Server Push and Server Hint:*** Some resources are pushed to the client even before the client requests for them. This can vastly enhance the user experience as essential resources are rendered before requesting them. In case of server hint, some resources are marked with priority by the server and the client can choose to request or not request them based on the bandwidth limitations [16].

The main difference between HTTP/2 and SPDY comes from their header compression algorithms. HTTP/2 uses HPACK algorithm for header compression, which significantly reduces the GET request message sizes compared to SPDY, which uses DEFLATE. Both, HTTP/2 and SPDY use TCP as the transport layer protocol. TCP's "slow start" forces applications to open multiple TCP connections to achieve parallelism and higher performance. For example, TCP's Initial congestion Window (IW) decides the amount of data to be sent in the initial phase of TCP transaction. If the IW value is small then it takes multiple round trip times (RTT)s for the TCP to reach its maximum throughput.

Table 1: Shows the Advantages of SPDY and HTTP/2 in terms of the aforementioned Features.

SPDY	HTTP/2	Advantages
Header compression	Header compression	Reduced packet size
Multiplexing	Multiplexing	No HOL blocking
Request prioritization	Request prioritization	Faster rendering
Server push and Server Hint	Server push	Notification
Binary frames	Binary frames	Faster parsing

III. PERFORMANCE EVALUATION

In this Chapter, we provide insights to the methodology we used to compare the efficiency of HTTP/2 and HTTP. When experimenting these two protocols, we consider on two major specs: Energy Consumption on mobile phone, and Protocol Performance.

3.1 Energy Consumption:

Cell phones determine the vitality required for their operation from batteries. In the case of many consumer electronics devices, notably mobile phones, battery capacity is severely restricted due to constraints on size and weight of the device. This suggests that energy efficiency of these devices is crucial to their usability. Therefore, best management of power consumption of these devices is critical.

In the meantime, device functionality is growing fast. Modern high-end mobile phones combine the functionality of a pocket-sized communicating device with PC-like capabilities, resulting in what are generally referred to as smartphones. These incorporate varied functionality that is such as internet browsing, sound and video playback, voice communication, short-message and e-mail communication, media downloads, gaming and much more. The rich functionality deepens the need for effective energy management, and raises the pressure on battery lifetime. A core requirement of effective and efficient management of energy is an excellent understanding of where and the way the energy is used: how much of the system's energy is used up by which parts of the system and under what circumstances.

Lately some new research proved that the new Http/2 protocol helped to increase the battery lifetime of the mobile devices. One research paper demonstrates that HTTP/2 can conserve energy more when compared to the present HTTP 1.x protocol [16]. They compared a real-world example using Google.com and Twitter.com sites to value the client mobile programs (Figure 4). To Benchmark HTTP/2 protocol to HTTP 1.x protocol from the energy consumption standpoint, we set up a static server as well as a mobile client to run benchmarking of the brand-new protocol. We use iOS APIs and iPhone Device (Iphone6s with iOS 10.1.1) from the client side. For the server side, we chose open source server Nginx.

NGINX is known for its high performance, stability, rich feature set, simple configuration, and low resource consumption. To test the energy consumption, we use the new battery tracking feature that can be found in the recent iOS versions in every iPhone device. We run an http 1.x app for about 10 hours that sends continues multiple requests over the time and we see just how much percent is used up during the running time. We did the same on http/2. In order to make sure that the result are more accurate, we did this tests At least 2 times (figure 5).

3.2 Protocol Performance:

One of the key developments brought by HTTP/2 is the Header compression, the multiplexed streams, Binary Protocol, And the server push. These and other good changes enabled to attain great web pages loading results, including those having tons of added files attached to them (e.g. designs, scripts, pictures, fonts).

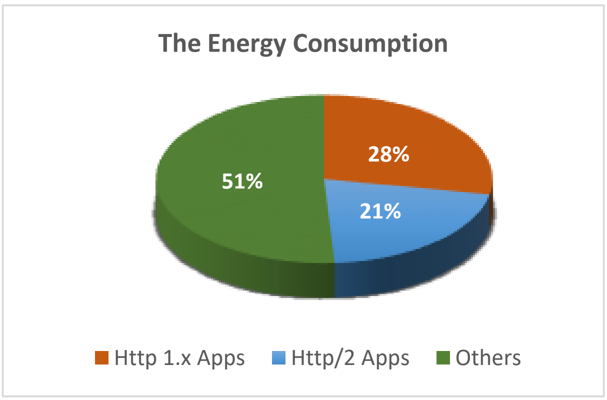

Figure 4: Http1 vs Http/2 energy consumption.

In the protocol, Performance Benchmark experiment we use Dummy website/web Page that's contains several images, css, and scripts Table 2. Our performance metric is the page load time. It's well known this metric is difficult to define and quantify accurately. In our evaluations, we rely on the times. The browser is instructed to get Navigate() command. The page is supposed to be loaded when the Navigate() call returns. To ensure that the results are even fair, we repeat this process until we see similar results to prevent the connection loss problems, and for each test we make sure we clear the cache in the browser. We use Nginx as a server as a http/2 web server and chrome developer's tool as a benchmarking tool.

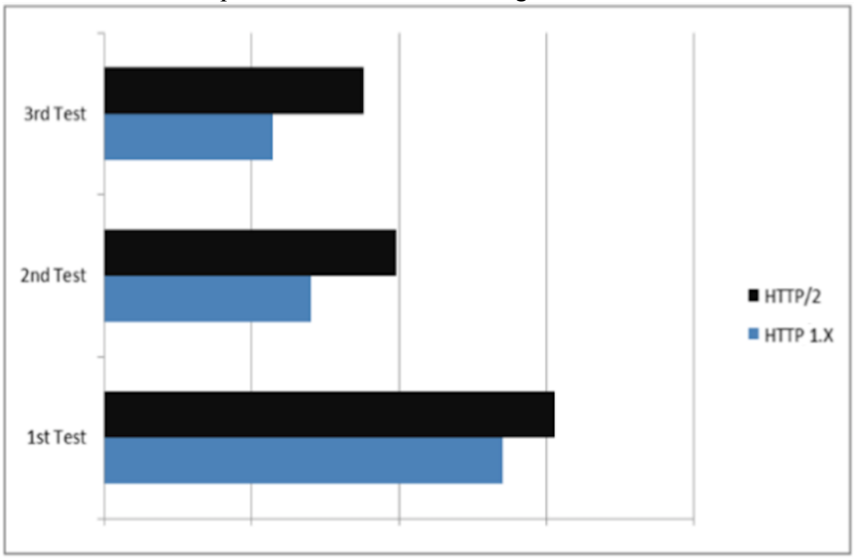

Figure 5: An experiment for the old HTTP vs. new HTTP/2. The final results shows that HTTP/2 is better than HTTP 1.x in the term of energy consumption.

We Quantify the performance by counting the number of requests per second as well as the result in figure 6 suggests that HTTP/2 could help enhancing the operation by up to 29% to 40% comparing to HTTP 1.x protocol. Generally, HTTP/2 primary development is focusing on web applications with so many resources instead of single web service (Figure 6).

Table 2: Composition of dummy web page.

File	File size
Index.html	13 KB
5 style sheets	1 KB
25 images	2.4 MB
4 JavaScript	0.9 KB

IV. CONCLUSION

Http/2 is newly designed protocol that intend to make our applications quicker, simpler, and much more robust by enabling us to reverse a lot of the HTTP/1.1 workarounds formerly done within our programs and address these issues within the transport layer itself. Better yet, in addition, it opens up lots of completely new opportunities enhance functionality and to optimize our applications and the Web. However, we've presented a performance research of http/2 protocol to prove that points. The main focus of this study was to evaluate the influence of http/2 on both of mobile phone Energy and the performance.

We first examined the Energy consumption of Http/2 compared to the http 1.x. additionally we Benchmark the speed and the performance of both of protocols. The results of experiments indicate that the new Protocol outperforms the Http 1.x in most cases.

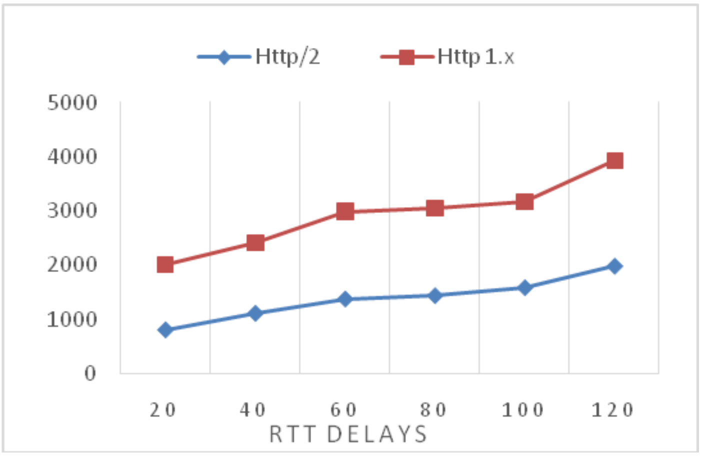

Figure 6: result of benchmarking http/1 against http/2.

REFERENCES

[1] H. De Saxce, I. Oprescu, and Y. Chen, “Is HTTP/2 really faster than HTTP/1.1?,” *Proc. - IEEE INFOCOM*, vol. 2015–Augus, pp. 293–299, 2015.

[2] A. Banerjee, L. K. Chong, S. Chattopadhyay, and A. Roychoudhury, “Detecting energy bugs and hotspots in mobile apps,” *Proc. 22nd ACM SIGSOFT Int. Symp. Found. Softw. Eng. - FSE 2014*, vol. 3, pp. 588–598, 2014.

[3] T. Cisco, “Cisco Visual Networking Index : Global Mobile Data Traffic Forecast Update , 2010 – 2015,” *Growth Lakel.*, vol. 2011, no. 4, pp. 2010–2015, 2011.

[4] H. Brotherton, *Data Center Energy Efficiency Calculator*. PhD thesis, Purdue University, 2014.

[5] F. Yang, P. Amer, J. Leighton, and M. Belshe, “A Methodology to Derive SPDY ’ s Initial Dictionary for Zlib Compression,” 2012.

[6] Mike Belshe and Roberto Peon, “SPDY Protocol - Draft 1 - The Chromium Projects,” 2012. [Online]. Available: https://www.chromium.org/spdy/spdy-protocol/spdy-protocol-draft1. [Accessed: 09-Dec-2016].

[7] David., “Difference between HTTP 1.0 and 1.1 | Difference Between.Net,” 2013. [Online]. Available: http://www.differencebetween.net/technology/protocols-formats/difference-between-http-1-0-and-1-1/. [Accessed: 09-Mar-2017].

[8] A. S. Alshammari and A. Al-Mogren, “HTTP/2 in Modern Web and Mobile Sensing-based Applications Analysis, Benchmarks and Current Issues,” *J. Electr. Electron. …*, vol. 5, no. 3, 2016.

[9] Google., “Make the Web Faster | Google Developers.” [Online]. Available: https://developers.google.com/speed/?csw=1. [Accessed: 21-Jan-2017].

[10] chrome, “A 2x Faster Web. [online] chromium blog. available at : http://blog.chromium.org/2009/11/2x-faster-web.html,” 2009.

[11] M. P. Matt Welsh, Ben Greenstein, “SPDY Performance on Mobile Networks | PageSpeed Insights | Google Developers,” 2012. [Online]. Available: https://developers.google.com/speed/articles/spdy-for-mobile. [Accessed: 05-Jan-2017].

[12] D. Stenberg, “HTTP2 Explained,” *Acm Sigcomm Comput. Commun. Rev.*, vol. 44, no. 3, pp. 120–128, 2014.

[13] “Usage Statistics of SPDY for Websites, Feb 2017.” [Online]. Available: https://w3techs.com/technologies/details/ce-spdy/all/all. [Accessed: 02-Mar-2017].

[14] “Internet Engineering Task Force (IETF).” [Online]. Available: https://www.ietf.org/. [Accessed: 11-Feb-2017].

[15] Frederic Lardinois, “Google Wants To Speed Up The Web With Its QUIC Protocol | TechCrunch,” 2015. [Online]. Available: https://techcrunch.com/2015/04/18/google-wants-to-speed-up-the-web-with-its-quic-protocol/. [Accessed: 13-Jan-2017].

[16] A. H. Shaiful Alam Chowdhury, Varun Sapra, “Is HTTP/2 More Energy Efficient Than HTTP/1.1 for Mobile Users?,” *PeerJ Prepr.*, p. 15, 2015.