HTTP/1.1's successors - New features and performance considerations

Lucca Greschner
Stuttgart Media University

March 30, 2024

Abstract

HTTP/2 and HTTP/3, the first new specifications of the Hypertext Transfer Protocol since 1999, promise to improve web browsing performance. This paper tries to summarize and evaluate the features and performance improvements HTTP/2 and HTTP/3 bring compared to HTTP/1.1. To do this, various research papers conducting experiments to measure HTTP performance were surveyed and put in context. The comparison of the surveyed papers showed that HTTP/2 and HTTP/3 show performance benefits in most cases but can also have a negative impact on performance in others.

1 Introduction

HTTP/1.0 was specified in 1996 [10]. Back then, two out of 100 people in the world used the internet. [18] In 2023, the amount of internet users is estimated to be 5.2 billion people. [5]

Since the introduction of HTTP/1.0, HTTP received three updates: HTTP/1.1, HTTP/2 and HTTP/3. The latter two protocol version where standardized 2015 and 2022 respectively. [15, 1]

This paper aims to summarize the improvements made by the newer HTTP revision HTTP/2 and HTTP/3, show the rationale behind revising the HTTP protocol and survey different researchers' results on performance comparisons of the different revisions of HTTP.

2 Related works

In 2015, Gaetano Carlucci, Luca De Cicco and Saverio Mascolo experimentally investigated QUIC as a possible replacement for TCP in the HTTP protocol. [3]

Maarten Wijnants et al. surveyed different user agent implementations of HTTP/2 prioritization. [20]

Martino Trevisan et al. collected data on HTTP/3 adoption and performance in different situations. [17]

Konrad Wolsing et al. did a comparison of QUIC-based HTTP and HTTP/2 using an optimized TCP stack. [21]

Justus Wendroth and Benedikt Jaeger compared HTTP/1.1, HTTP/2 and HTTP/3 regarding their features and performance. They compare the protocols while altering the parameters latency, packet loss and usage. [19]

Sanae Rosen et al. analyzed the impact of the server push feature introduced in HTTP/2. They focused on mobile performance by conducting measurements of page load time using various connection types. [14]

Mohammed Rajiullah et al. compared web page performance using the HTTP/1.1 protocol over TLS, HTTP/2 and HTTP over QUIC. [13]

Prasenjett Biswal et al. measured performance of HTTP over QUIC as proposed by Google and compared them to HTTP/2 performance. [2]

3 Problems with HTTP/1.1

HTTP/1.1 suffers from application-layer head-of-line-blocking (HoLB). [15] This means, that a blocking request can delay the fulfillment of another pending request. Besides that, HTTP/1.1 does not allow for multiplexing, needing for multiple connections to send requests concurrently. This poses a

problem, as TCP cannot handle congestion control over multiple connections. [1]

4 HTTP/2

HTTP/2 uses standard HTTP semantics but aims to improve performance through mapping existing HTTP semantics to an underlying layer that allows for optimizations of the transmission. The HTTP/2 protocol is defined through RFC 9113 [15]. Unless explicitly stated, this section exclusively refers to that definition.

4.1 Multiplexing

HTTP/2 brings the ability to multiplex transmissions over a single TCP connection by using the concept of streams.

The smallest unit of communication within HTTP/2 is called a frame. These frames are encoded in a binary representation and contain an encoded part of a HTTP message. [6]

Every HTTP message is sent through an independent sequence of frames called a stream of which multiple can be open within a HTTP/2 connection concurrently. Having multiple streams in one TCP connection allows for frames of different streams to be interleaved so that the transmission of a HTTP message cannot block the transmission of another one. [6]

Interleaving the frames sent, solves HoLB on the application-layer as seen in HTTP/1.1. However, RFC 9113 explicitly states that head-of-line-blocking within TCP is not addressed with HTTP/2. The concept of streams also solves the need for multiple TCP connections to send requests concurrently as seen in HTTP/1.1.

4.2 Field compression

HTTP/2 incorporates a feature called field compression using the HPACK algorithm, defined in RFC 7541. [11] The header fields compressed using HPACK are name-value pairs. In previous HTTP versions they were called headers. [15]

HPACK compresses header fields using two indexing tables. One being the static table which includes commonly used header fields and their values, the other being the dynamic table which is generated upon compression. [11] In these tables, the information about previously sent header fields is saved. With this knowledge, header fields of the following message are encoded differentially. [6]

4.3 Prioritization

The HTTP/2 protocol allows for prioritization which allows more important streams to finish before less important streams. Prioritization in HTTP/2 is managed through dependencies. If a stream depends on another, it is called its child.

There are two types of dependencies: Exclusive and Non-exclusive. Former meaning that only one stream can be dependent on its parent stream. In the case of non-exclusive streams however, the parent stream could theoretically have infinite child streams.

As Marten Wijnants et al. pointed out, there are two naive ways to arrange dependencies between two HTTP/2 streams. One approach being, having only exclusive dependencies and the other being only non-exclusive dependencies. [20]

4.4 Server Push

Server Push is a concept introduced in HTTP/2. Server Push allows a HTTP/2 server to send data to a client that is likely to be requested next [1]. For example, if the client sends a request for the HTML content of a website, the server might send it the necessary JavaScript and CSS files to display the website correctly.

4.5 Limitations of HTTP/2

HTTP/2 has the limitation of head-of-line-blocking on the transport layer. As a single TCP connection is used to transfer data, the loss of a TCP packet leads to blocking of all following packets. [2]

5 HTTP/3

5.1 QUIC

Other than HTTP/1.1 and HTTP/2, HTTP/3 is based on QUIC. [1] QUIC is a transport layer protocol based on UDP, standardized through RFC 9000 and incorporates security features such as the TLS handshake. [7] To transmit data, it uses the

concept of streams. Note, that in difference to HTTP/2 streams, QUIC streams are a transport layer feature. Streams in QUIC consist of ordered bytes that can be either sent uni- or bidirectionally. Through switching TCP for QUIC as the transport layer protocol underneath HTTP, HTTP/3 mitigates the head-of-line-blocking issue on the transport layer with HTTP/2. [17]

5.2 0-RTT handshake

QUIC introduces 0-RTT handshakes that can be used in HTTP/3. [7, 1] 0-RTT handshakes allow a client to establish an encrypted connection to a server and send data immediately. 0-RTT handshakes need the client to have had prior communication with the server. [16] This means, 0-RTT handshakes could have the potential of improving performance when reconnecting with a known server.

5.3 Field compression

HTTP/3 uses field compression to use bandwidth more efficiently. Other than HTTP/2 which uses the HPACK algorithm [15], HTTP/3 uses a variation of HPACK called QPACK [1]. QPACK is defined in RFC 9204, [8] is derived from HPACK and optimized so that less head-of-line-blocking occurs due to QUIC not guaranteeing in-order delivery. The standard is designed so that implementations can balance compression ratio and the chance of head-of-line-blocking. [1, 8]

6 Performance considerations

6.1 Effects of latency

Corbel et al. investigated page download times using HTTP/1.1 and HTTP/2. They artificially increased network delay in order to increase latency and measured the page load time using both protocols separately. Throughout all measurements, they found the average page download time with HTTP/2 to be lower than with HTTP/1.1 in all but one test cases. [4]

Trevisan et al. used BrowserTime to compare the performance of the HTTP/1.1, HTTP/2 and HTTP/3 protocol. Similar to R. Corbel, et al. they increased latency and measured the page load time of 14707 websites visited multiple times. [17] Their measurements show that HTTP/1.1 has the worst average page load time under high latency conditions, while HTTP/3 has the best. HTTP/2 lies in between. In their further analysis Trevisan et al. focus on the differences between HTTP/2 and HTTP/3. They found that HTTP/3 performed better than HTTP/2 in 50% of cases with no latency added while being slower than HTTP/2 in the other cases. With an added latency of 200ms they found 81% of websites loading faster using HTTP/3 than using HTTP/2.

So, while Corbel et al. saw improvements of HTTP/2 over HTTP/1.1 in most cases, Trevisan et al. could see a clear advantage of HTTP/2 over HTTP/3 only when artificially adding latency.

6.2 Effects of bandwidth

Trevisan et al. also compared the performance of HTTP/2 and HTTP/3 under varying bandwidth conditions. [17] When limiting bandwidth to 1 Mbit/s, they found HTTP/3 to have a lower page load time with 69% of tested websites, whereas in the test cases with 2 Mbit/s and 5 Mbit/s they haven't found a clear trend in which protocol version performs best.

6.3 Effects of packet loss

Other than with bandwidth and latency, Trevisan et al. didn't find any performance benefits of HTTP/3 in high packet loss scenarios. [17]

Corbel et al. measured page download time for 1% packet loss under varying network delay. On average, HTTP/2 had a 15% lower page download time than HTTP/1.1. The ratio of page download time peaked at a network delay of 10 milliseconds where HTTP/2 was measured to have half the page download time as HTTP/1.1 did. When conducting further measurements using higher packet loss rates they observed HTTP/2 to have a higher resilience against packet loss than HTTP/1.1. [4]

6.4 Effects of server push

Rosen et al. investigated the performance impact of the server push feature introduced in HTTP/2 under various network conditions and configurations.

[14] They found that server push was more successful in reducing page load time when only css and javascript files were pushed rather than all content on the website. Furthermore, they saw the most improvements in mobile network conditions like LTE and Wifi. However, they realize that HTTP/2 server push is rarely used in real world scenarios. Out of the top 10,000 sites they found only 5 to use HTTP/2 server push. This could be due to the configuration needed for HTTP/2 server push to work. The server needs to be configured so that it knows which resources to push to clients.

Even though Rosen et al. found performance gains of HTTP/2 server push, this probably has practically no real world implications as Google's Chrome had support for HTTP/2 server push removed in 2022. Chrome developers also found that support of server push has dropped to 0.7% of all HTTP/2 pages. [12]

6.5 Mobile networking

Rajiullah et al. analyzed the speed of HTTP/2 and HTTP/1.1 over TLS varying the connection type through multiple mobile network providers in four European countries.

To quantify the speed they used the RUMSpeedIndex as metric. RUMSpeedIndex is a metric that quantifies the speed of page rendering relying on events announced by the browser. They conducted their measurements on a selection of 20 of the most popular websites supporting HTTP/2. In most cases, they found that HTTP/2 had the same speed as HTTP/1.1 did, while HTTP/2 was faster in approximately the same amount of cases as HTTP/1.1 was faster in. [13]

The only case where HTTP/2 had significant speed gains was Reddit in which around 50% of tests on mobile networking reported HTTP/2 to be faster than HTTP/1.1. However, on wired networking, they only experienced HTTP/2 to be faster in approximately 30% of tests. [13]

7 Conclusion

The newer revisions of the HTTP protocol, HTTP/2 and HTTP/3, bring multiple features that promise to improve efficiency and performance. However, not all of them were realized. In the case of server push, as of 2022 the feature is considered not to be useful in real world applications.

The other promises seem to be fulfilled in parts as the surveyed studies saw better performance for HTTP/2 over HTTP/1.1 under most conditions. Conversely, Trevisan et al. found that HTTP/3 did perform worse than HTTP/2 in some cases. [17]

Furthermore, Marx et al. found significant differences in implementations of the QUIC and HTTP/3 protocols. [9]

The comparison of the different results showed that in most cases the newer revisions of the HTTP protocol perform better than HTTP/1.1. The advantage of HTTP/2 over HTTP/1.1 was seen to be greater than the advantage of HTTP/3 over HTTP/2. However, the results vary heavily depending on outer circumstances such as latency, bandwidth and implementation.

The surveyed experiments showed that HTTP/2 and HTTP/3 bring benefits for lower bandwidths and high latencies. In case of HTTP/2 a higher resilience against packet loss was observed by Corbel et al. As similar conditions are seen in mobile networks, [4] HTTP/2 and HTTP/3 could in theory improve the loading times of web pages on mobile devices.

The research conducted by Rajiullah et al. showed that this theory couldn't be proven in all cases for HTTP/2 in real life conditions. They found that performance differences were not equally prominent with different tested websites. Under exclusion of errors in measurement, this could either mean that variations in configuration or differences in page content could lead to different performance metrics.

7.1 Potential for further research

While the surveyed articles provided a helpful basis to evaluate the improvements brought by HTTP/2, HTTP/3 and QUIC, a concrete assessment of performance cannot be made as many variables haven't been researched yet. One of them being the influence of implementation and configuration differences on performance.

Another aspect not researched yet is the performance and efficiency implications of the newer revisions of HTTP on servers.

References

[1] Mike Bishop. HTTP/3. RFC 9114, June 2022.

[2] Prasenjeet Biswal and Omprakash Gnawali. Does quic make the web faster? In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2016.

[3] Gaetano Carlucci, Luca De Cicco, and Saverio Mascolo. Http over udp: an experimental investigation of quic. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 1–6, April 2015.

[4] Romuald Corbel, Emile Stephan, and Nathalie Omnes. Http/1.1 pipelining vs http2 in-the-clear: Performance comparison. In *2016 13th International Conference on New Technologies for Distributed Systems (NOTERE)*, pages 1–6, 2016.

[5] DataReportal, Meltwater, and We Are Social. Digital 2023: April global statshot report. `https://www.statista.com/statistics/617136/digital-population-worldwide/`, April 2023.

[6] Ilya Grigorik. Making the web faster with http 2.0: Http continues to evolve. *Queue*, 11(10):40–53, oct 2013.

[7] Jana Iyengar and Martin Thomson. QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000, May 2021.

[8] Charles 'Buck' Krasic, Mike Bishop, and Alan Frindell. QPACK: Field Compression for HTTP/3. RFC 9204, June 2022.

[9] Robin Marx, Joris Herbots, Wim Lamotte, and Peter Quax. Same standards, different decisions: A study of quic and http/3 implementation diversity. In *Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC*, EPIQ '20, pages 14–20, New York, NY, USA, 2020. Association for Computing Machinery.

[10] Henrik Nielsen, Roy T. Fielding, and Tim Berners-Lee. Hypertext Transfer Protocol – HTTP/1.0. RFC 1945, May 1996.

[11] Roberto Peon and Herve Ruellan. HPACK: Header Compression for HTTP/2. RFC 7541, May 2015.

[12] Barry Pollard. Removing http/2 server push from chrome. `https://developer.chrome.com/blog/removing-push/`, August 2022.

[13] Mohammad Rajiullah, Andra Lutu, Ali Safari Khatouni, Mah-Rukh Fida, Marco Mellia, Anna Brunstrom, Ozgu Alay, Stefan Alfredsson, and Vincenzo Mancuso. Web experience in mobile networks: Lessons from two million page visits. In *The World Wide Web Conference*, WWW '19, pages 1532–1543, New York, NY, USA, 2019. Association for Computing Machinery.

[14] Sanae Rosen, Bo Han, Shuai Hao, Z. Morley Mao, and Feng Qian. Push or request: An investigation of http/2 server push for improving mobile performance. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 459–468, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee.

[15] Martin Thomson and Cory Benfield. HTTP/2. RFC 9113, June 2022.

[16] Martin Thomson and Sean Turner. Using TLS to Secure QUIC. RFC 9001, May 2021.

[17] Martino Trevisan, Danilo Giordano, Idilio Drago, and Ali Safari Khatouni. Measuring http/3: Adoption and performance. In *2021 19th Mediterranean Communication and Computer Networking Conference (MedComNet)*, pages 1–8, 2021.

[18] International Telecommunication Union. Ict developments over time. `https://www.itu.int/ITU-D/ict/statistics/ict/`, July 2008.

[19] Justus Wendroth and Benedikt Jaeger. A Brief Overview on HTTP, November 2022.

[20] Maarten Wijnants, Robin Marx, Peter Quax, and Wim Lamotte. Http/2 prioritization and its impact on web performance. In *Proceedings of the 2018 World Wide Web Conference*,

WWW '18, pages 1755–1764, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee.

[21] Konrad Wolsing, Jan Rüth, Klaus Wehrle, and Oliver Hohlfeld. A performance perspective on web optimized protocol stacks: Tcp+tls+http/2 vs. quic. In *Proceedings of the Applied Networking Research Workshop*, ANRW '19, pages 1–7, New York, NY, USA, 2019. Association for Computing Machinery.