Oct 13, 2020

How HTTP/3 Changes the Web

Lecturer: Dr.Amit Dvir

Submitters: Gil Matsliah - 312480791

Amit Rash - 311235881

Table of Contents

Introduction

HTTP – Hyper Text Protocol is an application layer protocol used for data communication for the World Wide Web, its development was initiated by Tim Berners-Lee in 1989.

The protocol works in request-response manner as the client request objects like html files, jpg, gif, or any other type, the server answers with a response containing the state of the request and the information requested in the body of the response.

HTTP had a few versions released over the years as the need for faster communication only increased with the increasing size of data needed to transfer.

A new version

of the HTTP protocol, HTTP/1.1, was released in 1999 it used Transmission Control Protocol (TCP in short) as the transport protocol, this version of the protocol is still wildly used today, it usually establishes 6 TCP Connections working in parallel to serve 6 requests at once each serving one request at a time, the improvement it introduced was the concept of “keep alive”, previous versions opened a new connection each time a new object was needed to be transferred, which required each of those times a TCP three-way handshake and in the case that the site is encrypted also a TLS handshake both resulting in a delay at every new request. each new TCP connection was also required to go through “slow start”, a process that TCP congestion control uses to determine the amount of data that could be transferred at once without issues.
“keep-alive” solved that issue by keeping the connection alive after finishing the transfer and keeping the connection alive for other requests that could be made in the future and by that eliminating the need for a new handshake procedure for each requested file and also saving time with the “slow start” every new connection needs to go through reducing the overall delay compared to previous versions.
HTTP/1.1 also introduced pipelining, allowing to request multiple HTTP requests on a single TCP connection reducing the loading times especially on high latency connections, the requests were responded in a first in first out manner so the feature needed to be used carefully as all the requests needed to be answered fully and in the order that they were requested.

Fig.1: represents one TCP connection while using HTTP/1.1, usually there are 6 of these while using HTTP/1.1

In 2015 HTTP/2 was formally released and it offered quite a lot of new functions that could speed up the web, first it took a new approach for transferring information, on previous versions you could only transfer at a maximum 6 objects at once (one object per TCP connection, this limitation was called “HTTP had of line blocking") this version introduced the concept of streams, those streams were executed using one TCP connection, each having an ID used to identify itself within the TCP connection and each could transfer an object, one TCP connection could be used for all requests and responses eliminating the delay of opening a number of TCP connections and also speeding up the process by executing a huge number of requests in parallel instead of the six at a maximum that could be executed in previous versions.

HTTP/2 also introduced the feature HTTP PUSH that could be utilized by sending information before a request is made to the client to reduce latency if the server “assumes” a request will be made to an object after a certain request.

Another change HTTP/2 brought was the change from being text based in previous versions to binary based, this change minimized the size of information transferred and also helped in preventing text based issues like a blank line, spaces and more. HTTP/2 didn’t force encryption but it was mostly implemented with TLS so it was considered a safer protocol in general.

Fig.2: Illustration of how each version of the protocol works

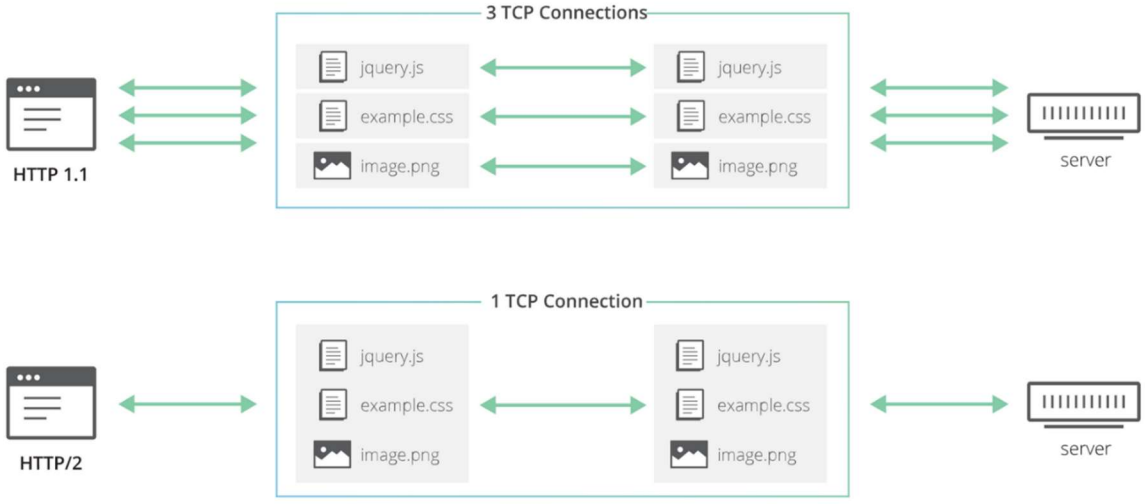


Fig.3: how each version of the protocol actually works

Recently, researchers started to work on HTTP/3. Still in its early stages and not finalized yet, HTTP/3 unlike previous versions of the protocol using TCP as its transmission protocol, HTTP/3 uses the QUIC protocol.

First, we need to understand what is the QUIC transmission protocol and what it brings to the table as it's the biggest factor in the advantages of HTTP/3 compared to HTTP/2.

QUIC is a new transfer protocol based on UDP developed by google later splits to another branch, IETF, that aims to achieve the same reliability as the TCP protocol. QUIC manages its streams within one QUIC connection in a similar way to the TCP protocol while also getting the speed advantage of UDP as there's less delay while establishing a connection.

QUIC has encryption built into it eliminating the delay occurred by the TLS handshake that was necessary while using TCP and also gives the option for 0 handshakes if the client requesting the data is already known to the server (using an identifier sent by the client) while also promising the security while using it unlike previous versions of HTTP that used TCP combined with a much faster establishment of the connection reducing the delay by a big margin.

Another advantage QUIC offers compared to TCP is its ability to continue transferring information even in the case that one of the streams within the connection encountered some problem, unlike TCP that affects **all** the streams within the connection if an error occurred giving QUIC a huge advantage in the case of a problematic connection between the client and the server eliminating TCP head of line blocking (problems in one stream affects all the streams within the TCP connection until its solved).

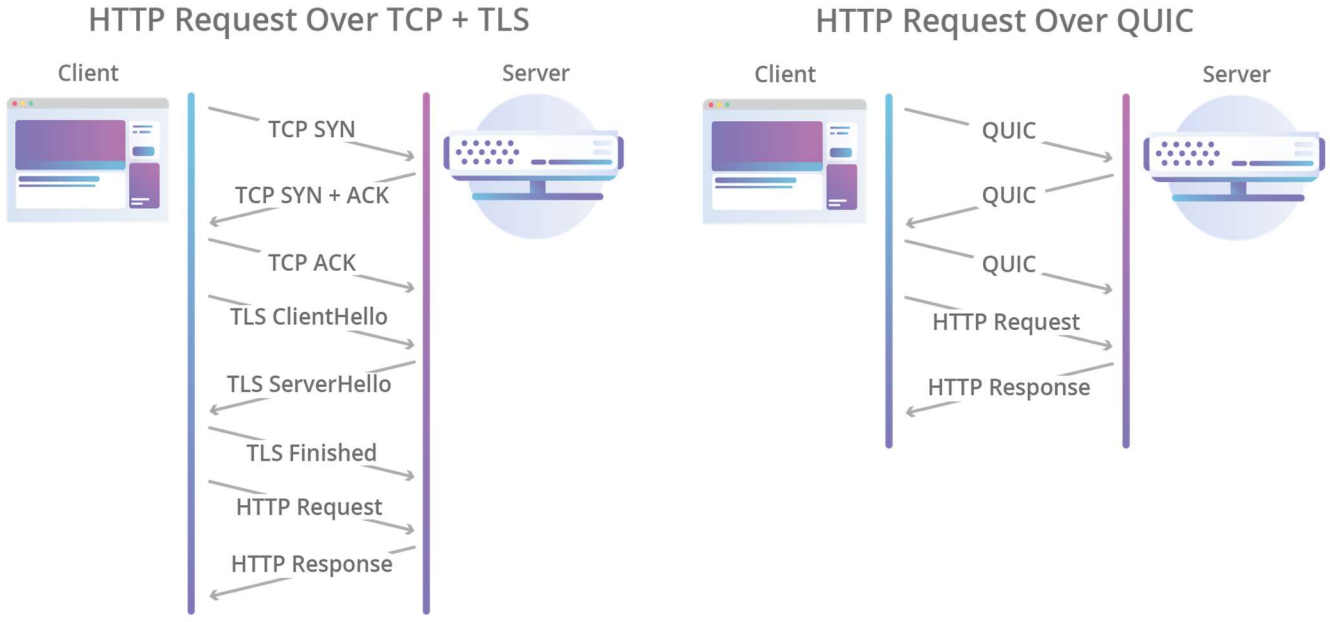


Fig.4: TCP HTTP connection establishment and request compared to QUIC

In a nutshell, QUIC combines the best of both the TCP protocol and the UDP protocol, it's a transmission protocol that can be used in many ways and it wasn't developed only to be used with HTTP, its faster than TCP while also offering the same level of reliability while also promising the encryption of the connection.

While it seems like HTTP/3 is just HTTP/2 done over QUIC that's not really the case and some changes were made to the protocol.
HTTP/3 dropped some features of HTTP/2 as some of them were already built in the QUIC protocol like QUIC streams reducing its complexity and also addressing the unorganized order of requests-responses of the protocol as QUIC does not guarantee the order of the responses within the connection a fact that required a new header to tackle this problem.

HTTP/3 is still quite young and its integration within the industry is growing, this version of the protocol is already supported by some of the popular browsers like Google Chrome and Mozilla Firefox.

In our project we will test the performance of the HTTP versions under various network conditions and will evaluate each version and how each of them works under different problems with several different websites.

We would like to know how each scenario of problems in the connection affects each version and measure its performance and see each version ups and downs.

Our Methodology:

- Creating different websites with a different number of objects.
- Building a server using a different version of HTTP for each of the websites.
- Simulating different problems in the connection.
- Analyzing the results.

Literature Review

In paper [10] we see tests of web page loading time using SPDY and HTTP/1.1. The result shows that SPDY can either help or sometimes hurt the load time compared to when browsers use HTTP. Most of the performance impact was because SPDY uses a single TCP connection in a similar way to HTTP/2, if the network connection is stable with low percentage of loss the single connection performs quite well but when the connection is problematic in a similar way to HTTP/2 all the communication is being hurt because of its use of TCP as its transport protocol and only using one connection at that.

we can see at [6] an evaluation of HTTP/2 new features tested through synthetic and real web pages under emulated and real Internet conditions. It found that HTTP/2 usually has better loading times compared to HTTP/1.1. it also shows that HTTP/2 is vulnerable to packet loss due to TCP congestion control that can lead to worse performance in a more impactful way compared to HTTP/1.1 as it only uses one TCP connection compared to the usual six that HTTP/1.1 uses the same as SPDY that was its main inspiration.

Trying to find new ways to overcome TCP limitations we can see that [7] investigates QUIC and compares it to HTTP/1.1 and SPDY, both using TCP as the transport protocol while QUIC is based on UDP, [7] shows the benefits of transporting HTTP traffic over QUIC as it reduces the overall page retrieval time compared to HTTP/1.1 and shows how QUIC outperforms SPDY in the case of a lossy connection.

More experiments performed to measure QUIC performance compared to the usual TCP we used until now, like article [8] that checks the impact of QUIC on networked multimedia quality of experience.

It checks how QUIC is performing in comparison to TCP on popular uses like browsing google or watching a video on YouTube.

The results weren’t as great as the authors expected them to be because google promised a big gain by using QUIC instead of TCP but in their testing there wasn’t much difference by using QUIC instead of TCP from quality of experience standpoint.

The biggest factor in their study that probably influenced the result is probably the low number of objects that were needed to be requested in the tests they performed as the biggest gain QUIC offers is achieved with large number of objects that needs to transferred and that advantage should be even more dramatic if the network has issues like packet loss. When these problems aren’t at play testing QUIC the results can be negligible compared to TCP as those are TCP biggest downfalls that gave the motivation for HTTP/3 use of QUIC as its transport protocol.

Giving us a better perspective on how the development of HTTP/3 is going we can look at paper [9] that checks HTTP/3 at an earlier point than our project and compares its performance to HTTP/2 using lighthouse, an open source tool developed by google to audit the web page performance, giving us an opportunity to see how HTTP/3 progressed so far as it’s only a draft for now and isn’t finalized yet and changes that will be made can lead to drastic change in performance.

In their testing they saw a general trend of HTTP/3 performing worse compared to HTTP/2 while using specific simulated network scenario and they offered their point of view as to why this happens like the fact that HTTP/3 implementations are still early, experimental and more, giving a reason as to why its performing worse than HTTP/2 at its current stage as it promised many advantages.

Our Experiment

For our experiment we used the open source web server “Caddy” that supports the three versions of HTTP we tested in our experiment (1.1, 2 and 3) to host the websites, both HTTP/2 and HTTP/1.1 were used with TLS to give a more comparable results (HTTP/3 is encrypted by default because of its use of QUIC as its transport protocol).
Caddy uses IETF-standard-draft version of QUIC for its HTTP/3 implementation.

We created three different websites each containing a different number of objects to be requested and one generic webpage we downloaded from the web, in total we tested 4 websites that represent different levels of objects to load.

We used Google Chrome to test each website with all three versions of HTTP tested and used Linux traffic control to configure the kernel packet scheduler to simulate problems with the network connection.

All experiments were tested using the public network one of which using cellular network, the generic website, as it’s used quite a lot today and looking at its performance is important because of the lossy connection of cellular networks.

Server (PC)
ISP: Hotnet
CPU: Intel i5-4570
Ram: 8GB ddr3
OS: ubuntu 20.04

Client (Laptop – Lenovo ideapad Y700)
ISP: Bezeq International
CPU: Intel i7-6700HQ
Ram: 16GB ddr4
OS: ubuntu 20.04

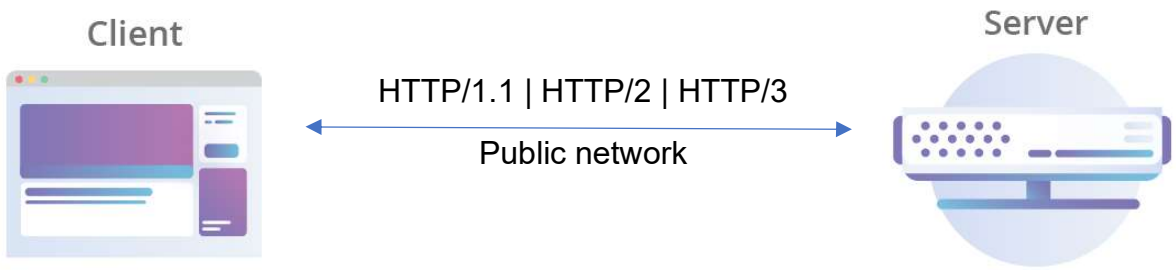


Fig.5: measurement setup

Results

Each graph shows the measurements of total average in seconds the website takes to fully load using a specific version of HTTP with a specific website we built (except the generic website used in the cellular test) featuring a specific number of objects of a specific size allowing us to see how each protocol behaves in many connection conditions with each website and measure its performance.

A simple site we made for testing, contains 50 gifs

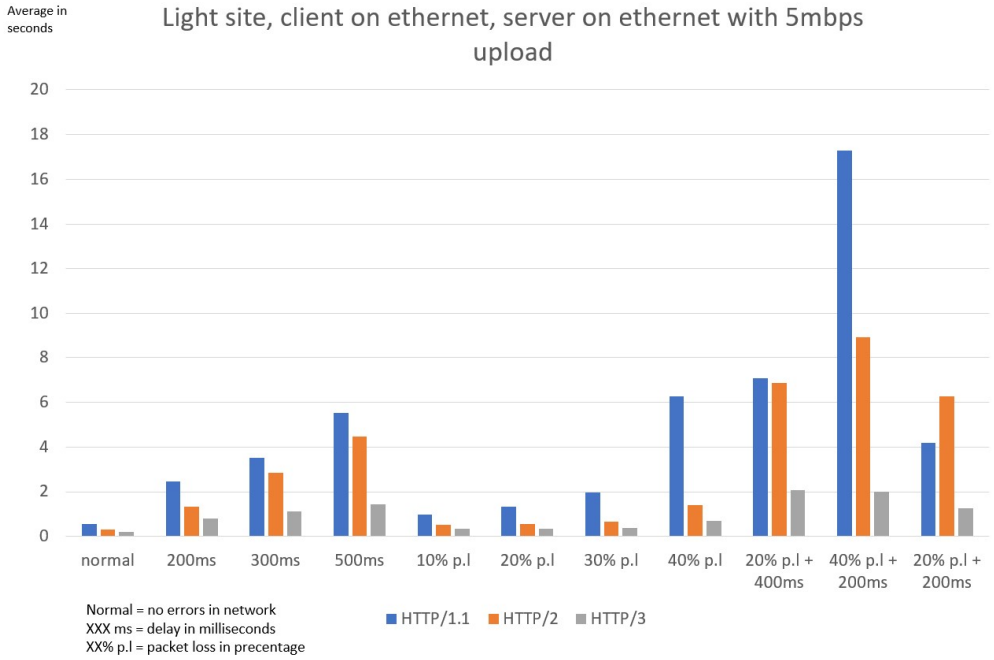


Fig.6: Light site, client and server using ethernet, server upload speed 5mbps

In this test we ran a template of the site we created containing 50 gifs, a light weight site without many requests to be made.
while the site loaded fast running with no issue's on all 3 versions we can see that when we simulate issues thing are affected in a different way on all 3 versions.

When we simulated network delay we can see that while HTTP/1.1 took the biggest hit in load times on average the gap widened on average as the delay got bigger while we see that HTTP/3 took a much smaller hit to its performance than the previous versions.

Packet loss looked a bit different, while HTTP/1.1 slowed as the percentage of the loss grew we payed special attention to HTTP/2 to see if the packet loss would affect it more than HTTP/1.1 as it uses only 1 TCP connection and the packet loss could affect all the streams within it resulting in a worse result than HTTP/1.1 as TCP congestion control could lead to worse behavior, on average that wasn't the case but we can clearly see the advantage of HTTP/3 at play here.
By using QUIC as its transmission protocol only the stream that encountered a packet loss stopped and the rest continue to work, that resulted in almost no change to the speed of loading the sites with different packet loss percentage.

While combing both packet loss and delay we can start to see that HTTP/2 performance started to be as good or even worse than HTTP/1.1, while analyzing the traffic we saw that as the issues grew HTTP/2 performance slowed by a big margin a big factor to that was the fact that if a packet was lost combined with the delay led to extreme slowdowns in some cases even worse than the previous version.
HTTP/3 again showed its strengths in the combined scenarios, by using QUIC the issues with the network were much less noticed as it handled them much better than TCP did resulting in much better load times.

A simple site we made for testing, contains 200 gifs

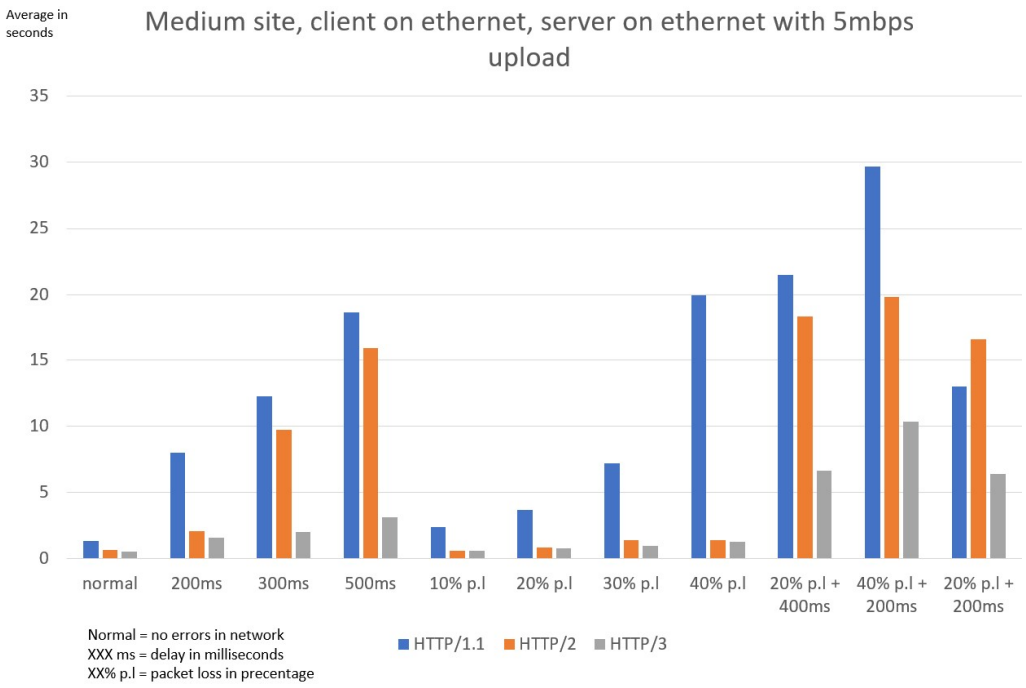


Fig.7: medium site, client and server using ethernet, server upload speed 5mbps

In this part of the test we ramped up the number of objects the site contained and by doing so increased the number of requests being issued by the client.

Even with no issues with the connection we can see here that HTTP/1.1 start's to be slower compared to HTTP/2 and HTTP/3 as they can issue all the requests in parallel compared to the 6 that can be issued at once in HTTP/1.1.

We see a similar trend here as HTTP/1.1 has a hard time with all the simulated issues especially with packet loss compared to the newer versions and while HTTP/2 is generally giving better result to the load times in a similar way to the previous test as the issues with the connection got to the extreme cases of the test the HTTP/2 protocol performed much worse.

As the load got bigger we can see HTTP/3 start to struggle a bit more compared to the lighter site but once again gave much better results compared to previous versions as it handles issues in the connection much better.

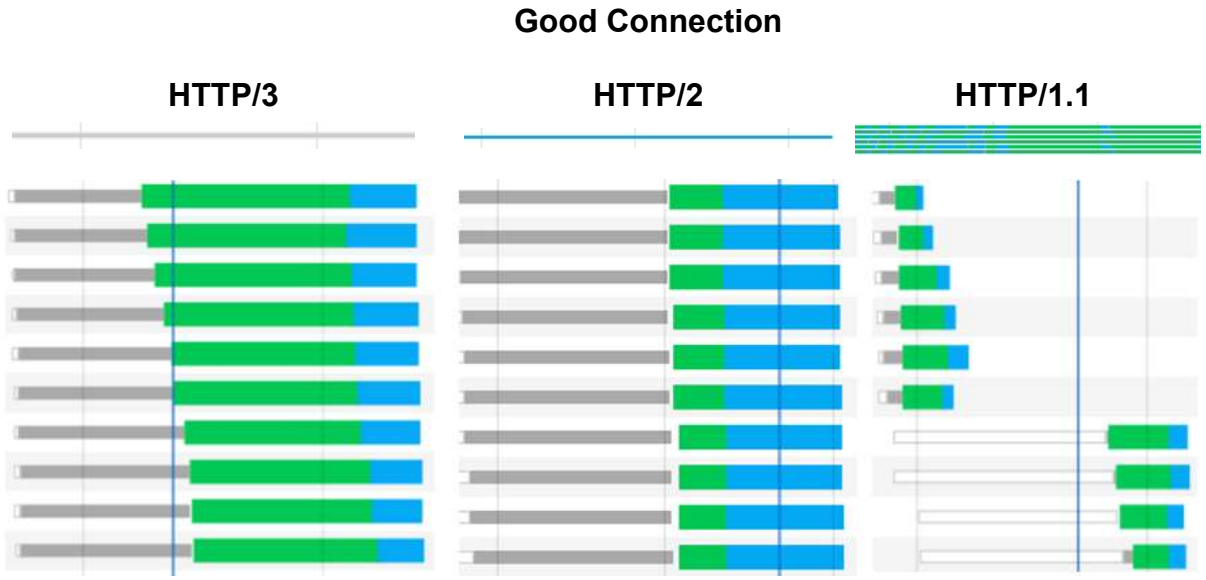


Fig.8: client-server with normal conditions waterfall, as seen from google chrome

The picture above shows how each protocol works under good network conditions with no issues, HTTP head of line blocking can be seen clearly while using HTTP/1.1, the maximum requests working in parallel is 6 while using this version, the maximum number of TCP connections HTTP/1.1 uses at once, while each TCP connection can transfer one object at a time.
HTTP/2 and HTTP/3 shows the advantages of multiplexing as they can work with a much bigger number of objects at once while only using one connection for all of the requests solving HTTP head of line blocking, the main disadvantage of HTTP/1.1.

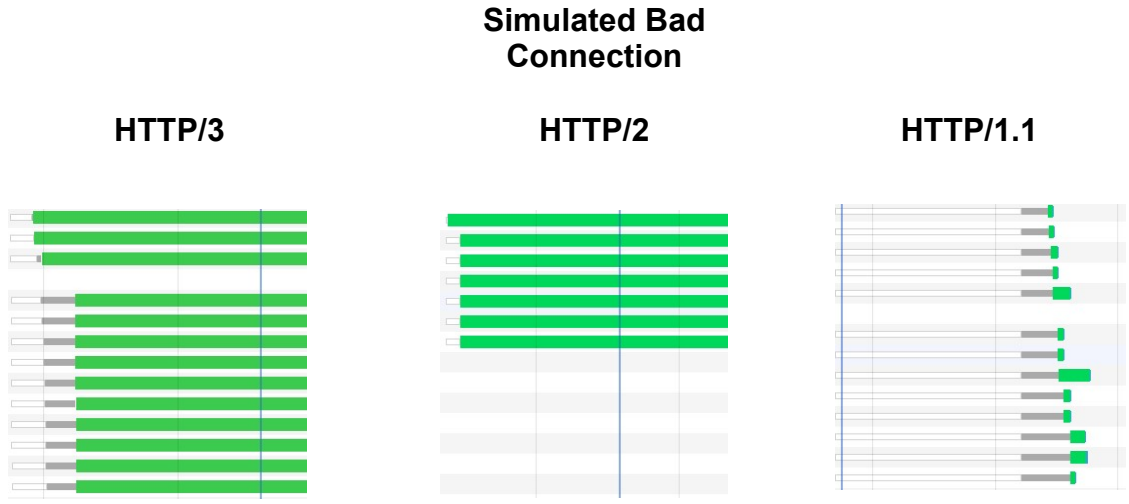


Fig.9: client-server with simulated conditions, packet loss and delay waterfall, as seen from google chrome

While HTTP/2 is generally faster than HTTP/1.1 the fact that it uses only one TCP connection can result in a bigger slowdown as HTTP/1.1 can keep the 5 other connections going as seen in the picture above while HTTP/2 stops completely all communication until the packet is retrieved (TCP head of line blocking).
HTTP/3 while also getting the advantage of multiplexing also has the advantage of keeping the other QUIC streams going in case of a issue with one stream within it as

seen in the picture above, one of the big advantages of using QUIC as the transport protocol.

A simple site we made for testing, contains 400 gifs

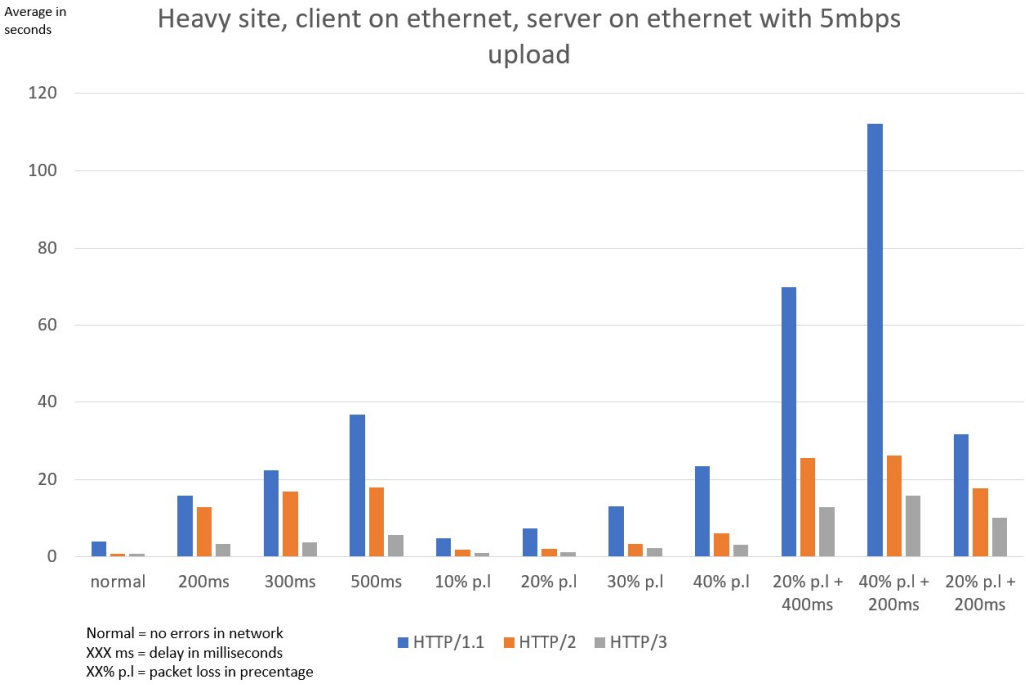
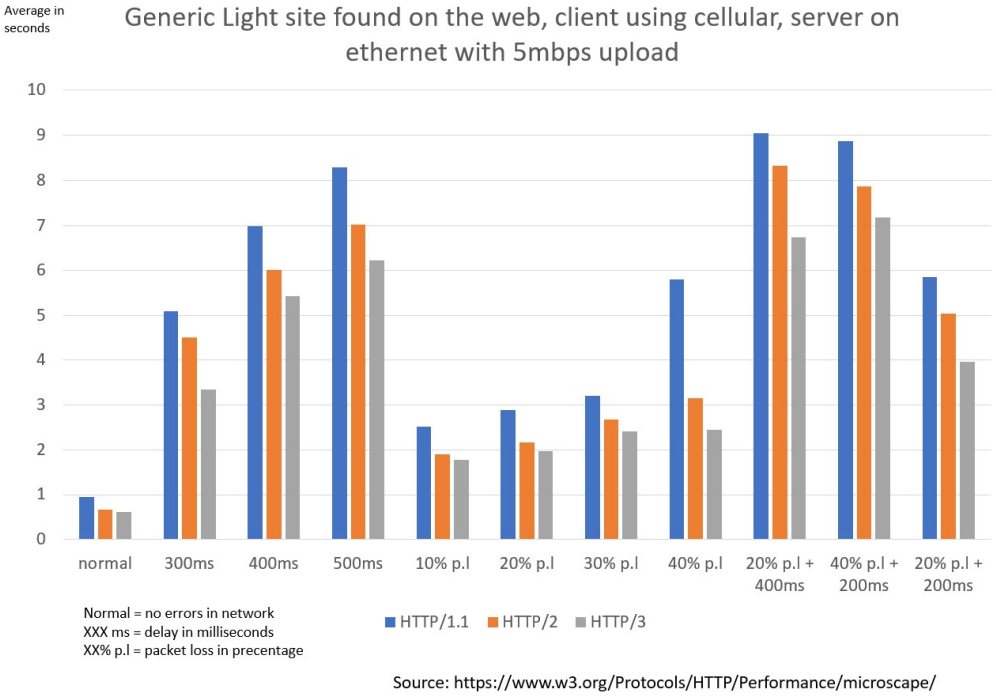


Fig.10: heavy site, client and server using ethernet, server upload speed 5mbps

This part of the test was used to take each protocol to its limits, 400 requests were made at once, HTTP head of line blocking showed HTTP/1.1 limits as on average the time it took to load the page was longer by a big margin compared to HTTP/2 and HTTP/3 even when the network had no issues.

We can see almost the same trend as before but with a much bigger penalty rendering HTTP/1.1 and HTTP/2 almost unusable if the network suffers from issues in this test as the loading times take a huge hit compared to HTTP/3.

Generic site contains some text and gifs, most of them are very light weight.



Source: https://www.w3.org/Protocols/HTTP/Performance/microscape/

Fig.11: generic site, client using cellular and server using ethernet, server upload speed 5mbps

In this part of the experiment we used a generic website we downloaded from the web, we used cellular network to test another option that's used quite a lot today to test the website using the 3 protocols.

The site is light weight and doesn't require many requests to be made from the client, this fact resulted in close results between all 3 protocols while the network was working without issue's as HTTP/1.1 didn't take much longer than HTTP/2 and HTTP/3 even though only 6 object can be requested and transferred at once.

That could also be seen while we simulated issues with the network as the result stayed pretty close, HTTP/3 still was the best performer but it wasn't as drastic compared to previous experiments, the small number of object that needed to be transferred playing a role as multiplexing didn't yield much better results.

Another factor that could influence the performance of this part of the test is that cellular networks are much more prone to issues than the Wired network we usually use with computers today as the delay is bigger when using cellular and potentially some issues with reception.

*the cellular network used in this part of the test is Partner 4G.

Conclusion

From the results of our experiment tests we can see that HTTP/3 eliminates HTTP/2 biggest downfalls as it much less affected by network issues compared to HTTP/2 that in some scenarios could perform even worse than HTTP/1.1 (can be seen in some scenarios in our testing and also on: Is the HTTP/2 really faster than HTTP/1.1 paper) while also enjoying the biggest advantages of HTTP/2 like multiplexing while also promising a secure connection as it's a default of QUIC.

All in all HTTP/3 is the next big step for the web, it offers better performance than previous versions in all the scenarios we tested , bringing especially better experience for the more isolated places around the globe that have troubled network infrastructures as it could be a night and day difference compared to the protocols we used until now giving everyone the chance to enjoy a better, faster and secure web experience.

References

[1] Linux traffic control:
https://netbeez.net/blog/how-to-use-the-linux-traffic-control/
This article gives a simple explanation on how to simulate network issues using Linux packet scheduler.

[2] Caddy web server:
https://caddyserver.com/
Caddy web server is the open source software we used to host the server it offers automatic certificates using Let's Encrypt, reverse proxy and much more.

[3] HTTP/3: the past, the present, and the future:
https://blog.cloudflare.com/http3-the-past-present-and-future/
This article contains a lot of information on how and why HTTP kept evolving and why there was a need for a new version and what are its features.

[4] HTTP/3 overview:
https://www.slideshare.net/bagder/http3-129039527
This slideshow offers a more simple approach to explain the HTTP/3, the past and more.

[5] HTTP/2 vs HTTP/3:
https://blog.cloudflare.com/http-3-vs-http-2/
Article containing explaining some of the benefits of HTTP/3 compared to HTTP/2.

[6] Is HTTP/2 really faster than HTTP/1.1:
https://ieeexplore.ieee.org/document/7179400

[7] HTTP over UDP: an experimental investigation of QUIC:
https://www.researchgate.net/publication/280547975_HTTP_over_UDP_an_experimental_investigation_of_QUIC

[8] Is QUIC becoming the New TCP? On the Potential Impact of a New Protocol on Networked Multimedia QoE:
https://ieeexplore.ieee.org/document/8743223

[9] An Early Benchmark of Quality of Experience Between HTTP/2 and HTTP/3 using Lighthouse:
https://www.researchgate.net/publication/340475942_An_Early_Benchmark_of_Quality_of_Experience_Between_HTTP2_and_HTTP3_using_Lighthouse

[10] How Speedy is SPDY?
https://www.usenix.org/system/files/conference/nsdi14/nsdi14-paper-wang_xiao_sophia.pdf