What Is HTTP/3 – Lowdown on the Fast New UDP-Based Protocol

In November 2018, the Internet Engineering Task Force (IETF) met in Bangkok, and a new Internet-Draft was adopted. The QUIC transport protocol, an HTTP/2 successor, was renamed to HTTP/3. HTTP/3 builds on UDP, and is already being used by prominent internet companies such as Google and Facebook. If you're using Chrome and connecting to a Google service, you're probably already using QUIC.

The new version of the HTTP protocol benefits from the bare-metal, low-level UDP protocol, and defines many of the new features which were in previous versions of HTTP at the TCP layer. This provides a way of solving constraints within the existing internet infrastructure.

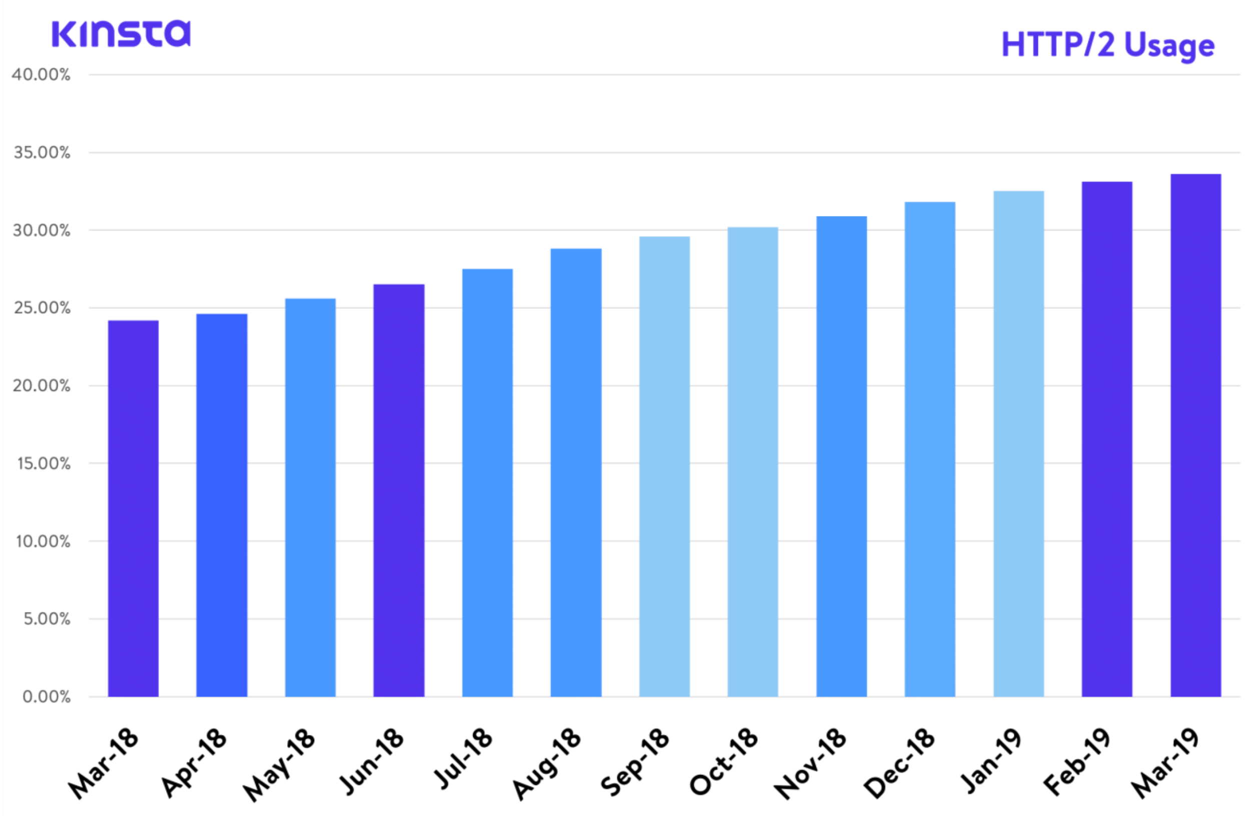
The first results are promising, and when the Internet-Draft by IETF expires, in June 2019, we can expect HTTP/3 to be promoted as a new, third-generation HTTP standard.

Just like with HTTP/2, HTTP/3 will again build on these achievements to help speed up the web. 🚀

HTTP/3 Is Coming

Some say that the web industry's hunger for more speed and lower latency is only matched by Google Chrome's hunger for more RAM.

In 2016, we published an article about HTTP/2, a standard that, according to W3Techs, currently has around a 34% world adoption rate. And according to Can I use, it's also supported by all modern web browsers. Yet here we are, writing an article about the next version of the protocol, HTTP/3.



HTTP/3 is, at the time of this writing, an IETF Internet-Draft or ID, which means that it is currently under consideration for an upcoming internet standard by the Internet Engineering Task Force – an international *internet standards body*, in charge of defining and promoting agreed upon internet protocol standards, such as TCP, IPv6, VoIP, Internet of Things, etc.

It is an open body which unites the web industry and facilitates discussion about the direction of the internet.

Currently, the *ID* phase of HTTP/3 is the last phase before proposals are promoted to the level of RFCs, or Request-for-Comments, which we can consider, for all intents and purposes, official internet protocol definitions. They are then implemented by all major internet players.

This means that HTTP/3 is to become an official standard once the draft expires later this year (June 2019).

What is HTTP/3 - In Layman's Terms

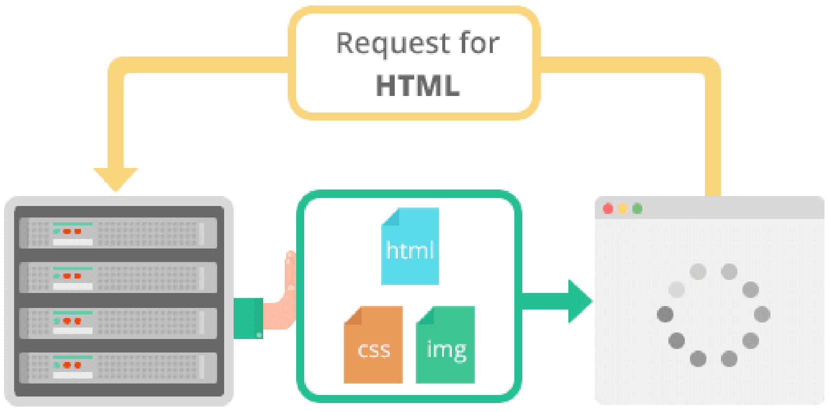
HTTP/3 is the third version of the Hypertext Transfer Protocol (HTTP), previously known as HTTP-over-QUIC. QUIC (Quick UDP Internet Connections) was initially developed by Google and is the successor of HTTP/2. Companies such as Google and Facebook have already been using QUIC to speed up the web.

A Bit of Background – It Started with HTTP/2

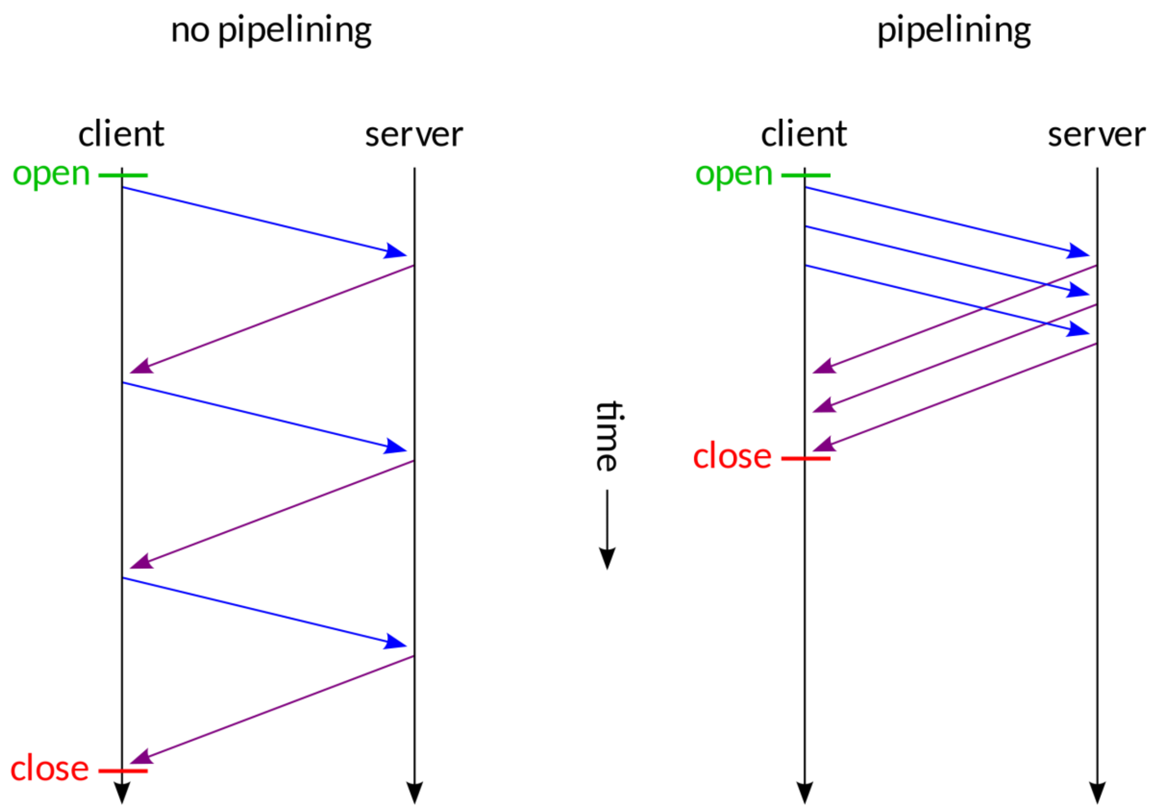
At Kinsta we are obsessed with squeezing every last millisecond from our stack, whether it's taking advantage of the newest version of PHP, delivering data over Google Cloud Platform's premium tier network, or caching assets on our HTTP/2 CDN.

HTTP/2 brought some serious improvements with non-blocking downloads, pipelining, and server push which has helped us overcome some limitations of the underlying TCP protocol. It allowed us to minimize the number of request-response cycles and handshakes.

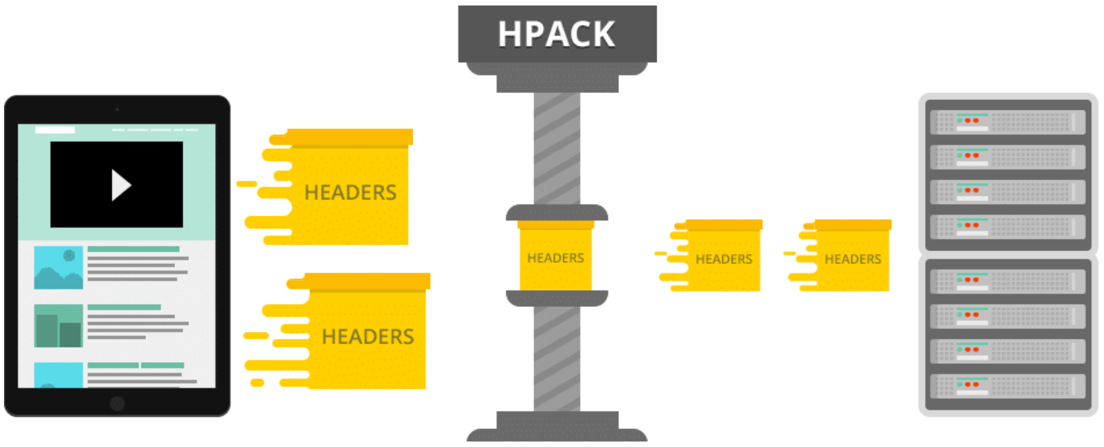
HTTP/2 made it possible to push more than one resource in a single TCP connection – multiplexing. We also got more flexibility in the ordering of static downloads, and our pages are now no longer constrained by a linear progression of the downloads.



In practice, this means that now one large javascript resource does not necessarily equal a choke point for all the other static resources waiting their turn.



Add to these things HTTP/2's header HPACK compression and default binary format of data transfer, and we have, in many cases, a significantly more efficient protocol.



Major browser implementations made it a requirement for websites to implement encryption – SSL – to be able to reap the benefits of HTTP/2 – and sometimes this incurred a computation overhead that rendered speed improvements unnoticeable. There were even some cases where users reported slowdown after transitioning to HTTP/2.

Let’s just say that the early days of adoption of this version were not for the weak of heart.

The NGINX implementation also lacked the server-push feature, relying on a module. And NGINX modules are not your usual Apache drop-in modules that you can just copy – NGINX has to be recompiled with these.

While some of these issues are solved now, if we look at the entire protocol stack, we see that the primary constraint lies on a lower level than HTTP/2 dared to venture.

To elaborate this, we will dissect today’s internet protocol stack from its bottom layer to the top. If you want to learn more about the background of HTTP/2, make sure to check out our ultimate HTTP/2 guide.

Internet Protocol (IP)

The Internet Protocol (IP) defines the bottom-part of the entire internet topology. It’s the part of the internet stack that is, we can safely say, really not negotiable without changing everything, including replacing the entire hardware infrastructure, from routers to servers and even the machines of end-users.

So, while the protocol overhaul may be due, such a far-reaching endeavor is not on the horizon at this time, mainly because we haven’t come up with a viable, groundbreaking, yet backward-compatible alternative.

Underneath it, there is link layer – the part of the protocol which is “bare metal” so to say.

On a side note, a convincing case for a complete overhaul can be seen from the speed with which creators of IPFS (interplanetary file system) managed to close an ICO funding that made them 250 mil USD within one month.

We can trace the beginnings of the IP protocol back to 1974, to a paper published by the Institute of Electrical and Electronics Engineers and authored by Vint Cerf and Bob Cahn. It detailed packets being sent over a network, routing them across IP addresses, and numerically defined addresses of nodes in a network/networks. The protocol defined the format of these packets, or datagrams – its headers and payload.

After the RFC 760 definition from 1980, the IETF settled with the definition widely used to this day, in its Request For Comments 791. This is the fourth version of the protocol, but we could say it's the first production version.

It uses 32-bit addresses, which sets constraint to the number of addresses to around 4 billion. This limitation is the explanation for the mystery of why non-business internet users get "dynamic IP addresses" by their ISPs, and a static IP is considered an "added value" and often subject to extra charges.

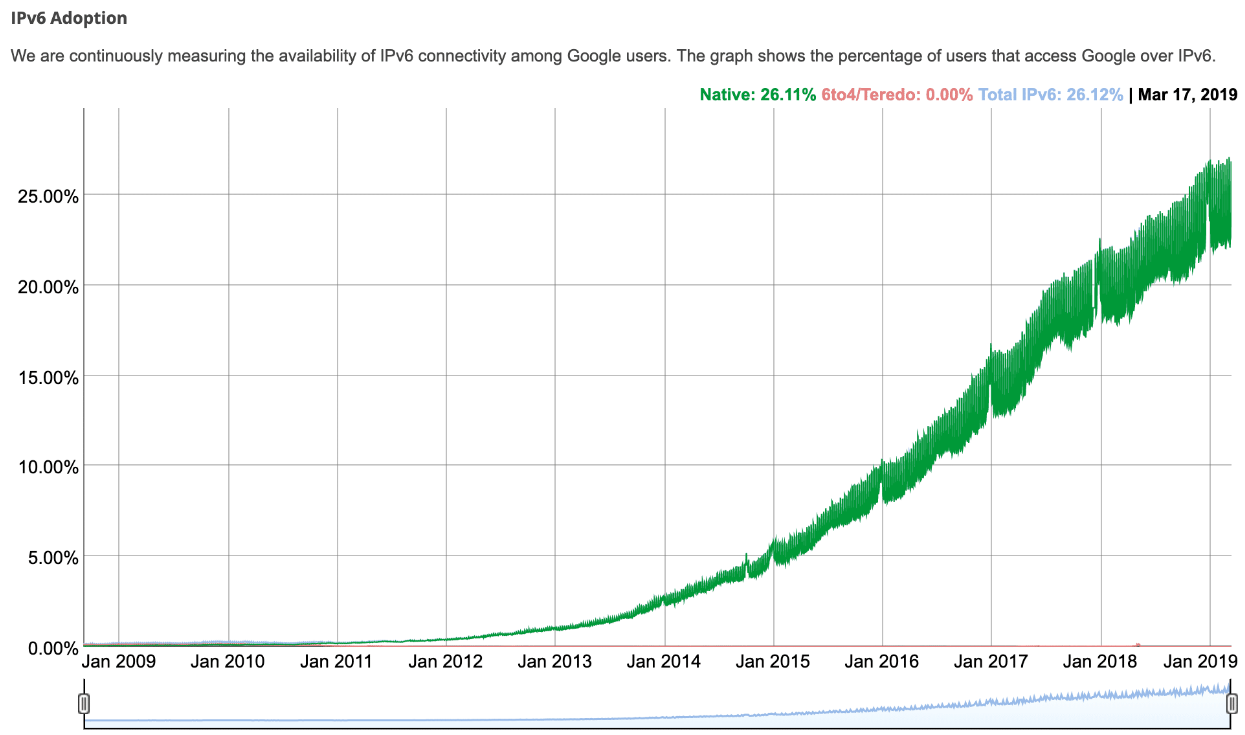
They are rationing.

It wasn't long until it was realized that 32-bit addresses are not enough, and the shortage was looming, so many RFCs were published trying to deal with this. Although these solutions are widely used today, and are part of our daily lives, it's probably safe to say these amount to hacks.

Internet Protocol version 6 or IPv6 came as a way to address these limitations, including to gradually be adopted over the previous version. It was made a Draft Standard document for the IETF in 1998, and was raised to an Internet Standard in 2017.

While IPv4 address space was limited by its 32-bit address length, IPv6 standard was given 128 bits, or 3.4 * 10 ^ 38 possible addresses. This should be enough to last us for quite some time.

According to Google and IPv6 connectivity among its users, IPv6 adoption is just over 25% as of March 2019.


IPv6 Adoption

We are continuously measuring the availability of IPv6 connectivity among Google users. The graph shows the percentage of users that access Google over IPv6.

IP is a rudimentary layer of the internet stack, defining most basic things, without guarantees of delivery, data integrity, or the ordering of transmitted packets. On its own it's unreliable. The header format of IPv4 provides for header checksum, which the transmission nodes use to verify the integrity of the header. This makes it different from the IPv6 version, which relies on the link layer underneath, enabling it to be faster..

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                 Example Internet Datagram Header
```

Understanding the Role of TCP and UDP

Now it's time to explore where HTTP/3 fits in with TCP and UDP.

TCP

While IP is the underlying layer of all of our online communications today, TCP (Transmission Control Protocol) is a higher-level part of the internet protocol suite, providing the reliability that is needed for the web, mail, file transfer (FTP) – for application layers/protocols of the internet.

This includes multi-step connection establishment, with handshakes, assured order of packets, and retransmission of lost packets. It provides feedback (Acks) of delivery to the sender and so on. There is also checksum computation to detect errors.

All these things indicate a lot of steps that make TCP a reliable protocol, making it a foundation of the most notorious internet services we use today.

Its specification dating back to 1974 (RFC 675) and 1981 (RFC 793) hasn't changed substantially to this day.

The reliability that TCP provides doesn't, however, come without a cost. The overhead of all the roundtrips required by handshakes, delivery feedbacks, ordering guarantees, and checksums that could be considered weak and redundant. It has made TCP a bottleneck of the modern protocol stack. HTTP/2 has reached a plateau of speed improvements that can be achieved on top of TCP.

Changing the TCP in any substantial way is not a straightforward endeavor, because the protocol is, as part of the TCP/IP stack that goes back all the way to the '70s. It's deeply embedded into operating systems, device's firmware, etc.

UDP

UDP (User Datagram Protocol) is also one of the parts of the Internet Protocol Suite, with its specification dating back to 1980 (RFC 768).

It is, as the name suggests, a datagram-based connectionless protocol. Which means there are no handshakes and there are no assurances of ordering or delivery. This means that any possible steps for ensuring delivery, data integrity, and other things are left to the application layer. This means that an application building on top of UDP can cherry-pick strategies it will employ depending on the concrete case, or it can possibly leverage elements of the link layer, like checksums, to avoid overhead.

Because UDP is widespread just like TCP, it makes it possible to achieve improvements without requiring wide change of firmware on all the devices connected to the internet, or significant changes in the operating systems.

Deployment of new protocols is hampered by many firewalls, NATs, routers and other middle-boxes that only allow TCP or UDP are deployed between users and the servers they need to reach. – HTTP/3 explained

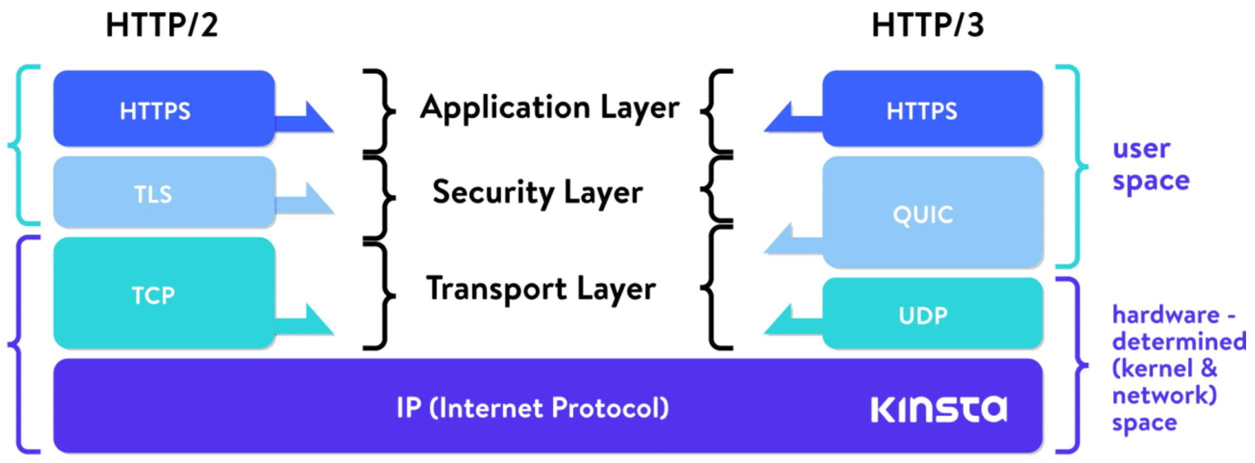
This thread on Hacker News can help us begin to understand the reasoning behind building the new HTTP version on top of the existing network stack, rather than reinventing it (although there is more to it than that).

UDP packet format specification is rather minimal, it's header consists of the source port, destination port, length, in bytes, of packet header and packet data, and checksum. Checksum can be used to verify data-integrity both for header and data part of the packet.

Checksum is optional when the underlying protocol layer is IPv4, and mandatory with IPv6. So far, UDP has been used for things like computer systems clock synchronization (NTP), VoIP applications, video streaming, DNS system, and DHCP protocol.

QUIC and HTTP/3

QUIC (Quick UDP Internet Connections) was first deployed by Google in 2012. It redefines boundaries of network layers, relying on lower-level UDP protocol, redefining handshakes, reliability features, and security features in "user-space," avoiding the need for upgrading kernels of internet-wide systems.



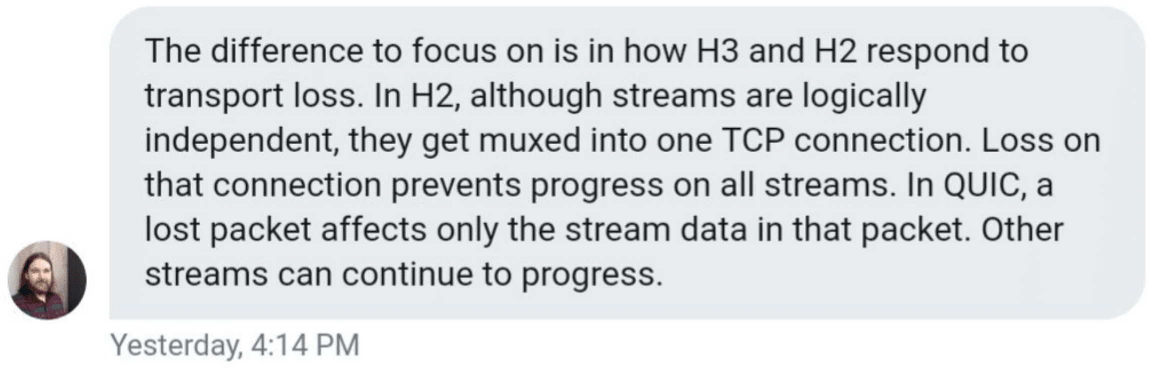
Just like with HTTP/2, an advancement which was spearheaded by Google's SPDY or speedy, HTTP/3 will again build on these achievements.

While HTTP/2 did give us multiplexing, and mitigate head-of-line-blocking, it is constrained by TCP. You can use a single TCP connection for multiple streams multiplexed together to transfer data, but when one of those streams suffers a packet loss, **the whole connection (and all its streams) are held hostage,** so to say, until TCP does its thing (retransmits the lost packet).

This means that all the packets, even if they are already transmitted and waiting, in the buffer of the destination node, are being blocked until the lost packet is retransmitted. Daniel Stenberg in his book on http/3 calls this a "TCP-based head of line block." He claims that, with 2% packet loss, users will do better with HTTP/1, with six connections to hedge this risk.

QUIC is not constrained by this. With QUIC building on the on connectionless UDP protocol, the concept of connection does not carry the limitations of TCP and failures of one stream do not have to influence the rest.

As Lucas Pardue from Cloudflare put it:



The difference to focus on is in how H3 and H2 respond to transport loss. In H2, although streams are logically independent, they get muxed into one TCP connection. Loss on that connection prevents progress on all streams. In QUIC, a lost packet affects only the stream data in that packet. Other streams can continue to progress.

Yesterday, 4:14 PM

With a focus on UDP *streams*, QUIC achieves multiplexing without having to piggyback on one TCP connection. QUIC builds its *connection* on a higher level than TCP. New streams within QUIC connections are not forced to wait for the others to finish. QUIC connections also benefit from doing away with TCP handshake overhead, which reduces latency.

Folks at Cisco made an interesting video explaining TCP's 3-way handshake.

While QUIC does away with TCP reliability features, it makes up for it above the UDP layer, providing retransmitting of packets, ordering and so on. Google Cloud Platform introduced QUIC support for their load balancers in 2018 and saw an **improvement in mean page load time by 8% globally**, and up to 13% in regions where latency is higher.

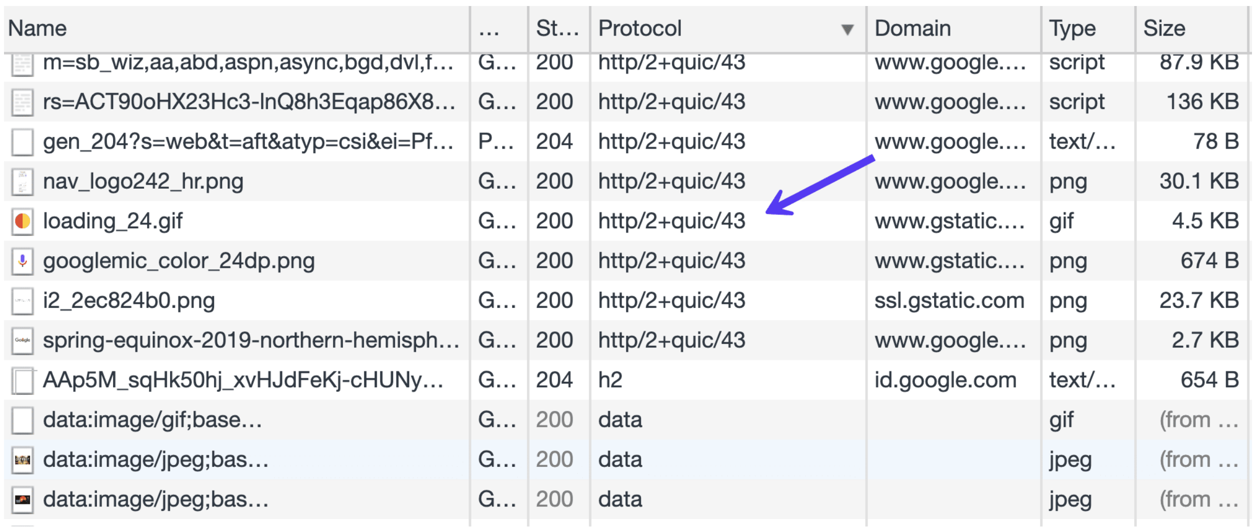
Between Google Chrome, YouTube, Gmail, Google's search and other services, Google was able to deploy QUIC on a nice chunk of the internet, without waiting for IETF. Google's engineers claim that in 2017, 7% of the internet traffic was already conducted over QUIC.

Google's version of QUIC was focused on just HTTP transport, using HTTP/2 syntax. People from IETF (those in charge of standardizing QUIC), decided that IETF version of QUIC should be able to transport more than just HTTP. For the time being, however, any work on non-HTTP protocols over QUIC is on hold.

One more thing IETF's working group decided is that the standardized version is going to use TLS 1.3 encryption instead of Google's custom solution. TLS 1.3, compared to the older versions, also contributes to protocol speed, as its handshakes require fewer roundtrips. **Kinsta supports TLS 1.3 on all of our servers and our Kinsta CDN.**

Right now, Google continues to use its own version of QUIC in its product, while directing it's development efforts toward the IETF standards. Most of the other internet players are building on top of the IETF version (the two differ in some other aspects beside encryption).

If we open Chrome Dev Tools, and load some of Google's products, like Gmail, in the Protocol column of the Network tab, we will see a lot of resources being loaded via Google's version of the QUIC protocol. This is also the case for Google's products like Analytics, Google Tag Manager, etc.



Name	…	St…	Protocol	Domain	Type	Size
m=sb_wiz,aa,abd,aspn,async,bgd,dvl,f…	G…	200	http/2+quic/43	www.google.…	script	87.9 KB
rs=ACT90oHX23Hc3-lnQ8h3Eqap86X8…	G…	200	http/2+quic/43	www.google.…	script	136 KB
gen_204?s=web&t=aft&atyp=csi&ei=Pf…	P…	204	http/2+quic/43	www.google.…	text/…	78 B
nav_logo242_hr.png	G…	200	http/2+quic/43	www.google.…	png	30.1 KB
loading_24.gif	G…	200	http/2+quic/43	www.gstatic.…	gif	4.5 KB
googlemic_color_24dp.png	G…	200	http/2+quic/43	www.gstatic.…	png	674 B
i2_2ec824b0.png	G…	200	http/2+quic/43	ssl.gstatic.com	png	23.7 KB
spring-equinox-2019-northern-hemisph…	G…	200	http/2+quic/43	www.google.…	png	2.7 KB
AAp5M_sqHk50hj_xvHJdFeKj-cHUNy…	G…	204	h2	id.google.com	text/…	654 B
data:image/gif;base…	G…	200	data		gif	(from …
data:image/jpeg;bas…	G…	200	data		jpeg	(from …
data:image/jpeg;bas…	G…	200	data		jpeg	(from …

Cloudflare recently published a very extensive update about the standardization progress.

While UDP does provide QUIC and HTTP/3 some inherent advantages, it also brings some challenges. TCP has been the mainstream protocol for years, while UDP has not, so operating systems and the software stack for it, in general, is not as optimized. Consequently, there is much higher CPU load/requirements with QUIC, by some estimates, twice as much as with HTTP/2.

Optimizations go deep down to the kernel of operating systems, and different routers and devices firmware. This Red Hat tuning guide may shed more light on the topic for those more technically inclined.

We could say that QUIC attempts to re-engineer TCP features on top of a more minimal, and more flexible protocol.

QUIC connections, which we mentioned earlier, combine TLS and transport handshakes. Once established, they are identified by unique CIDs (connection IDs). These IDs persist across IP changes and can help to secure uninterrupted downloads on, for example, a switch from 4G to WiFi. This is relevant, particularly because more and more internet traffic is conducted on mobile devices. Questions may arise whether this element is conceived by Google to facilitate better user-tracking across different connections and internet providers.

TLS is mandatory, and is meant to make it hard for devices in the middle to tamper with, or sniff the traffic. That is why it is not rare to see firewall providers and vendors like Cisco seeing the UDP protocol as a problem, and to provide ways to disable it. It is harder for middlemen to inspect and supervise or filter QUIC traffic.

QUIC streams are sent over QUIC connections, uni-direction or bi-directional. Streams have IDs, that identify the initiator, and whether the stream is uni-directional or bi-directional, and also serve in-stream flow-control.

While QUIC is a transport-layer protocol, HTTP is the layer above that, an application-layer protocol, or application protocol.

Since backward-compatibility is of the utmost importance, the IETF promoted the implementation of HTTP/3 will include the old version (HTT1 or HTTP/2) in the response. It will include a header which informs the client that HTTP/3 is available, along with port/host information, as described in RFC 7838.

This is different from HTTP/2, in which transport can be negotiated within the TLS handshake. But since IETF has all but adopted QUIC-based HTTP/3 as the next standard, we can expect web clients to anticipate HTTP/3 support more and more. It is possible for clients to cache data from previous HTTP/3 connections, and to connect directly (zero-round-trip, or 0-RTT) on subsequent visits to the same host.

Summary

There are those who think that, with HTTP/2 standard not being adopted yet fully, it may be too early to push for HTTP/3 (version three). This is a valid point, but, as we mentioned, this protocol has already seen wide-scale tests and implementations. Google began testing it as early as 2015, as well as Facebook in 2017.

Since then, other players have joined the standardization efforts, such as Akamai and Mozilla. At the last IETF hackathon in November 2018, the list of attendees showed interest in QUIC by companies such as Facebook, Apple, Google, Mozilla, NetApp, and LiteSpeed Tech. There were some promising tests, and it looks like LiteSpeed might be the first major server vendor with a functioning HTTP/3 server. Cloudflare is also currently running QUIC in beta.

Shortly after this, QUIC was renamed to HTTP/3 in IETF's Internet Draft. It will expire at the end of June 2019, and we can expect the RFC, or the final standard sometime in July.

This year will be exciting, as we can expect to see the move by major software vendors to implement the new standard.