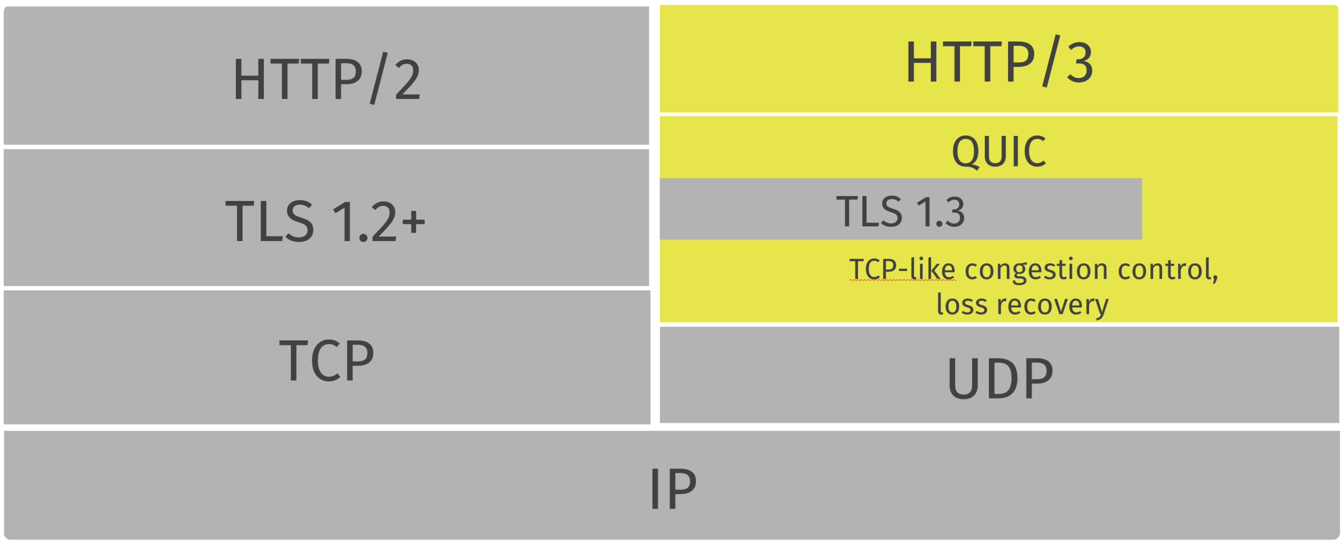
HTTP/3 explained

Protocol features

The QUIC protocol from a high level.

Illustrated below is the HTTP/2 network stack on the left and the QUIC network stack on the right, when used as HTTP transport.



HTTP/3 over QUIC stack overview

UDP

Transfer protocol over UDP

QUIC is a transfer protocol implemented on top of UDP. If you watch your network traffic casually, you will see QUIC appear as UDP packets.

Based on UDP it also then uses UDP port numbers to identify specific network services on a given IP address.

All known QUIC implementations are currently in user-space, which allows for more rapid evolution than kernel-space implementations typically allow.

Will it work?

There are enterprises and other network setups that block UDP traffic on other ports than 53 (used for DNS). Others throttle such data in ways that makes QUIC perform worse than TCP based protocols. There is no end to what some operators may do.

For the foreseeable future, all use of QUIC-based transports will probably have to be able to gracefully fall-back to another (TCP-based) alternative. Google engineers have previously mentioned measured failure rates in the low single-digit percentages.

Will it improve?

Chances are that if QUIC proves to be a valuable addition to the Internet world, people will want to use it and they will want it to function in their networks and then companies may start to reconsider their obstacles. During the years the development of QUIC has progressed, the success rate for establishing and using QUIC connections across the Internet has increased.

Reliable

While UDP is not a reliable transport, QUIC adds a layer on top of UDP that introduces reliability. It offers re-transmissions of packets, congestion control, pacing and the other features otherwise present in TCP.

Data sent over QUIC from one end-point will appear in the other end sooner or later, as long as the connection is maintained.

Streams

Similar to SCTP, SSH and HTTP/2, QUIC features separate logical streams within the physical connections. A number of parallel streams that can transfer data simultaneously over a single connection without affecting the other streams.

A connection is a negotiated setup between two end-points similar to how a TCP connection works. A QUIC connection is made to a UDP port and IP address, but once established the connection is associated by its "connection ID".

Over an established connection, either side can create streams and send data to the other end. Streams are delivered in-order and they are reliable, but different streams may be delivered out-of-order.

QUIC offers flow control on both connection and streams.

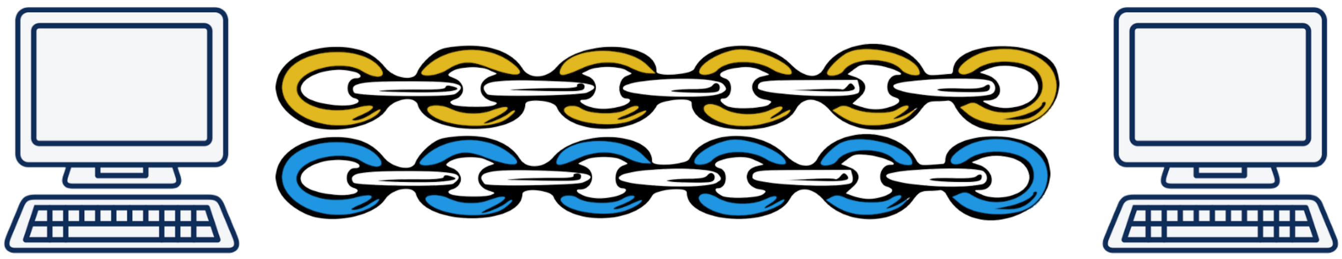
See further details in connections and streams sections

In Order

QUIC guarantees in-order delivery of streams, but not between streams. This means that each stream will send data and maintain data order, but each stream may reach the destination in a different order than the application sent it!

For example: stream A and B are transferred from a server to a client. Stream A is started first and then stream B. In QUIC, a lost packet only affects the stream to which the lost packet belongs. If stream A loses a packet but stream B does not, stream B may continue its transfers and complete while stream A's lost packet is re-transmitted. This was not possible with HTTP/2.

Illustrated here with one yellow and one blue stream sent between two QUIC end-points over a single connection. They are independent and may arrive in a different order, but each stream is reliably delivered to the application in order.


two QUIC streams between two computers

Fast handshakes

QUIC offers both 0-RTT and 1-RTT connection setups, meaning that at best QUIC needs no extra round-trips at all when setting up a new connection. The faster of those two, the 0-RTT handshake, only works if there has been a previous connection established to a host and a secret from that connection has been cached.

Early data

QUIC allows a client to include data already in the 0-RTT handshake. This feature allows a client to deliver data to the peer as fast as it possibly can, and that then of course allows the server to respond and send data back even sooner.

TLS 1.3

The transport security used in QUIC is using TLS 1.3 (RFC 8446) and there are never any unencrypted QUIC connections.

TLS 1.3 has several advantages compared to older TLS versions but a primary reason for using it in QUIC is that 1.3 changed the handshake to require fewer roundtrips. It reduces protocol latency.

The Google legacy version of QUIC used a custom crypto.

Transport and application

The IETF QUIC protocol is a transport protocol, on top of which other application protocols can be used. The initial application layer protocol is HTTP/3 (h3).

The transport layer supports connections and streams.

The legacy Google version of QUIC had transport and HTTP glued together into one single do-it-all and was a more special-purpose send-http/2-frames-over-udp protocol.

HTTP/3 over QUIC

The HTTP layer, called HTTP/3, does HTTP-style transports, including HTTP header compression using QPACK - which is similar to the HTTP/2 compression named HPACK.

The HPACK algorithm depends on an *ordered* delivery of streams so it was not possible to reuse it for HTTP/3 without modifications since QUIC offers streams that can be delivered out of order. QPACK can be seen as the QUIC-adapted version of HPACK.

Non-HTTP over QUIC

The work on sending protocols other than HTTP over QUIC has been postponed until after QUIC version 1 has shipped.