What is QUIC?
WARNING: PORTIONS OF THIS DOCUMENT ARE OUT OF DATE

QUIC (**Q**uick **U**DP **I**nternet **C**onnections) is a new transport protocol for the internet, developed by Google.

QUIC solves a number of transport-layer and application-layer problems experienced by modern web applications, while requiring little or no change from application writers. QUIC is very similar to TCP+TLS+HTTP2, but implemented on top of UDP. Having QUIC as a self-contained protocol allows innovations which aren't possible with existing protocols as they are hampered by legacy clients and middleboxes.

Key advantages of QUIC over TCP+TLS+HTTP2 include:

- Connection establishment latency
- Improved congestion control
- Multiplexing without head-of-line blocking
- Forward error correction
- Connection migration

Connection Establishment

For a complete description of connection establishment, please see the QUIC Crypto design document. Briefly, QUIC handshakes frequently require zero roundtrips before sending payload, as compared to 1-3 roundtrips for TCP+TLS.

The first time a QUIC client connects to a server, the client must perform a 1-roundtrip handshake in order to acquire the necessary information to complete the handshake. The client sends an inchoate (empty) client hello (CHLO), the server sends a rejection (REJ) with the information the client needs to make forward progress, including the source address token and the server's certificates. The next time the client sends a CHLO, it can use the cached credentials from the previous connection to immediately send encrypted requests to the server.

Congestion Control

QUIC has pluggable congestion control, and provides richer information to the congestion control algorithm than TCP. Currently, Google's implementation of QUIC uses a reimplementation of TCP Cubic and is experimenting with alternative approaches.

One example of richer information is that each packet, both original and retransmitted, carries a new sequence number. This allows a QUIC sender to distinguish ACKs for retransmissions from ACKs for originals and avoids TCP's retransmission ambiguity problem. QUIC ACKs also explicitly carry the delay between the receipt of a packet and its acknowledgment being sent,

and together with the monotonically-increasing sequence numbers. This allows for precise roundtrip-time calculation.

Finally, QUIC's ACK frames support up to 256 NACK ranges, so QUIC is more resilient to reordering than TCP (with SACK), as well as able to keep more bytes on the wire when there is reordering or loss. Both client and server have a more accurate picture of which packets the peer has received.

Multiplexing

One of the larger issues with HTTP2 on top of TCP is the issue of head-of-line blocking. The application sees a TCP connection as a stream of bytes. When a TCP packet is lost, no streams on that HTTP2 connection can make forward progress until the packet is retransmitted and received by the far side - not even when the packets with data for these streams have arrived and are waiting in a buffer.

Because QUIC is designed from the ground up for multiplexed operation, lost packets carrying data for an individual stream generally[1] only impact that specific stream. Each stream frame can be immediately dispatched to that stream on arrival, so streams without loss can continue to be reassembled and make forward progress in the application.

Forward Error Correction

In order to recover from lost packets without waiting for a retransmission, QUIC can complement a group of packets with an FEC packet. Much like RAID-4, the FEC packet contains parity of the packets in the FEC group. If one of the packets in the group is lost, the contents of that packet can be recovered from the FEC packet and the remaining packets in the group. The sender may decide whether to send FEC packets to optimize specific scenarios (e.g., beginning and end of a request).

Connection Migration

QUIC connections are identified by a 64 bit connection ID, randomly generated by the client. In contrast, TCP connections are identified by a 4-tuple of source address, source port, destination address and destination port. This means that if a client changes IP addresses (for example, by moving out of Wi-Fi range and switching over to cellular) or ports (if a NAT box loses and rebinds the port association), any active TCP connections are no longer valid. When a QUIC client changes IP addresses, it can continue to use the old connection ID from the new IP address without interrupting any in-flight requests.

---

[1] Truth in advertising: QUIC compresses HTTP headers via HTTP2 header compression, which imposes head-of-line blocking for header frames only. FEC is expected to mitigate this. It is also possible to address with HTTP2 header compression and extra implementation logic.