QUIC

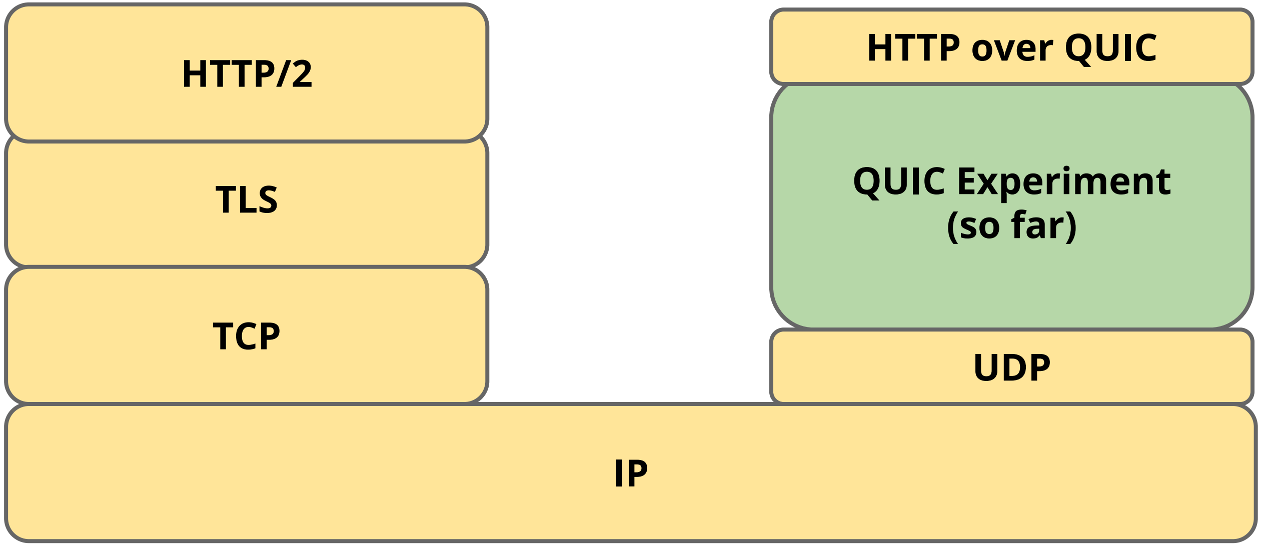
A New Internet Transport

Presenter: Jana Iyengar

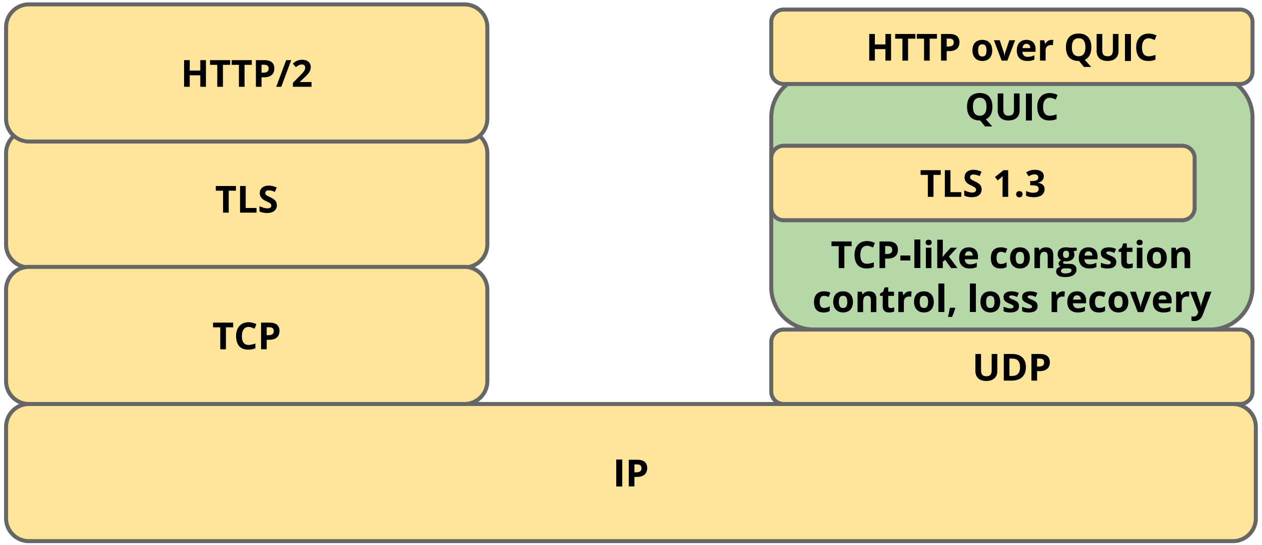
QUIC and the IETF

Nov 2013	Early design and experience (TSVAREA)
Mar 2015	QUIC handshake (SAAG)
Mar 2015 onwards	Replacing QUIC's handshake with TLS1.3
July 2015	BarBoF, experimental results
Nov 2015	Cubic bug in QUIC, TCP (TCPM)
July 2016	**BoF** Now!

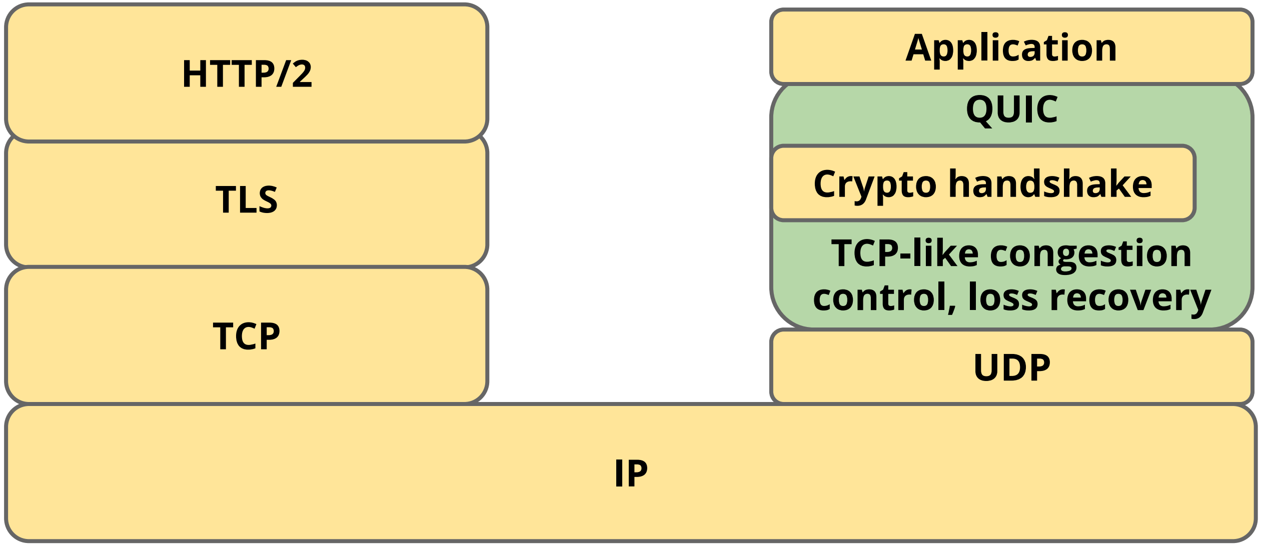
The QUIC Experiment



The IETF Proposal



Standardized QUIC



QUIC Design Aspirations

- Deployability and evolvability

QUIC Design Aspirations

- Deployability and evolvability
- Low latency connection establishment

QUIC Design Aspirations

- Deployability and evolvability
- Low latency connection establishment
- Multistreaming and per-stream flow control

QUIC Design Aspirations

- Deployability and evolvability
- Low latency connection establishment
- Multistreaming and per-stream flow control
- Better loss recovery and flexible congestion control

QUIC Design Aspirations

- Deployability and evolvability
- Low latency connection establishment
- Multistreaming and per-stream flow control
- Better loss recovery and flexible congestion control
- Resilience to NAT-rebinding

QUIC Design Aspirations

- Deployability and evolvability
- Low latency connection establishment
- Multistreaming and per-stream flow control
- Better loss recovery and flexible congestion control
- Resilience to NAT-rebinding
- Multipath for resilience and load sharing

Deployability and Evolvability

Uses UDP as the substrate

enables deployment through various middleboxes

userspace implementation enables rapid deployment

Deployability and Evolvability

Uses UDP as the substrate

enables deployment through various middleboxes

userspace implementation enables rapid deployment

Version negotiation

enables protocol wire format evolution

Deployability and Evolvability

Uses UDP as the substrate

enables deployment through various middleboxes

userspace implementation enables rapid deployment

Version negotiation

enables protocol wire format evolution

Fully authenticated and mostly encrypted headers

avoids network ossification

QUIC Streams

Multiplexed streams within a transport connection

- multiple streams avoids HoL blocking
- shared congestion control and loss recovery
- two levels of flow control: stream and connection

Congestion Control & Loss Recovery

QUIC builds on decades of experience with TCP

Congestion Control & Loss Recovery

QUIC builds on decades of experience with TCP

Incorporates TCP best practices

TCP-like congestion control (NewReno, Cubic)

Congestion Control & Loss Recovery

QUIC builds on decades of experience with TCP

Incorporates TCP best practices

TCP-like congestion control (NewReno, Cubic)

FACK, TLP, F-RTO, Early Retransmit, …

(also, time-based loss detection)

Congestion Control & Loss Recovery

QUIC builds on decades of experience with TCP

Incorporates TCP best practices

TCP-like congestion control (NewReno, Cubic)

FACK, TLP, F-RTO, Early Retransmit, …

(also, time-based loss detection)

Richer signaling than TCP

Richer Signaling Than TCP

Retransmitted packets consume new sequence number

- no retransmission ambiguity
- prevents loss of retransmission from causing RTO

Richer Signaling Than TCP

Retransmitted packets consume new sequence number

- no retransmission ambiguity
- prevents loss of retransmission from causing RTO

More verbose ACK

- TCP supports up to 3 SACK ranges
- QUIC supports up to 256 ACK ranges
- explicit packet receive times
- enables ACK decimation

QUIC Implementations

Chromium (open source)

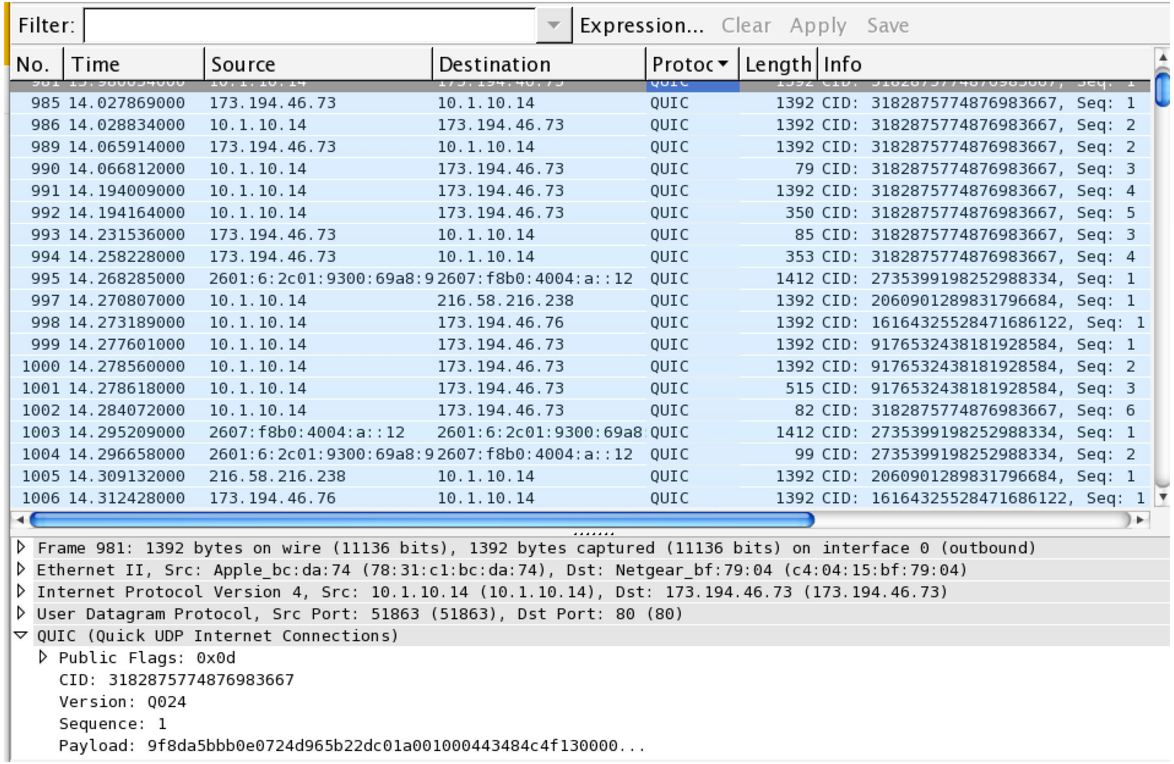
https://cs.chromium.org/chromium/src/net/quic/

quic-go (open source implementation in Go)

https://github.com/lucas-clemente/quic-go

Christian Huitema's implementation

Debugging Tools: Wireshark



Debugging Tools: Chrome

chrome://net-internals