From gQUIC to IETF QUIC and Beyond

Ian Swett ianswett@google.com

MileHighVideo, July 2019

Oveview

1. QUIC overview of QUIC
2. IETF QUIC compared to gQUIC
3. gQUIC to IETF QUIC migration at Chrome and Google

A QUIC History - SIGCOMM 2017

Protocol for HTTPS transport, deployed at Google starting 2014

Between Google services and Chrome / mobile apps

Improved application performance

YouTube Video Rebuffers: 15 - 18%

Google Search Latency: 3.6 - 8%

35% of Google's egress traffic (7% of Internet)

IETF QUIC working group formed in Oct 2016

Modularize and standardize QUIC

The QUIC Transport Protocol: Design and Internet Scale Deployment

Key QUIC Features

Always encrypted end-to-end

Multistreaming transport with no head of line blocking

0RTT connection establishment

Better loss recovery and flexible congestion control

Supports mixing reliable and unreliable transport features

Improved privacy and reset resistance

Connection migration

gQUIC compared to IETF QUIC

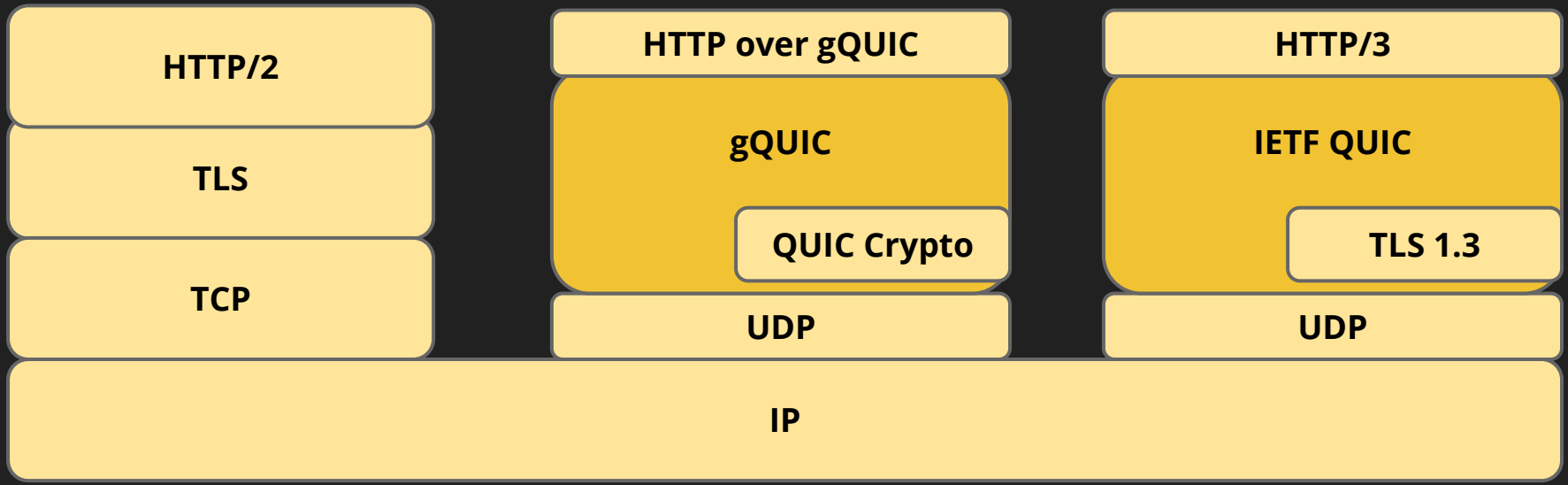
What is gQUIC? IETF QUIC? HTTP/3?

gQUIC: The Google QUIC project that is 7+% of public internet traffic today
Refers to the transport on top of UDP, QUIC Crypto, and HTTP mapping

IETF QUIC: The IETF standardized version of QUIC
Refers to the transport on top of UDP using a TLS 1.3 handshake

HTTP/3: The name for the HTTP mapping that lives natively on IETF QUIC

HTTP/2 vs gQUIC vs HTTP/3



What Changed from gQUIC to IETF QUIC/HTTP/3?

Every bit on the wire, some key changes:

	gQUIC	**IETF QUIC/H3**
Connection IDs	8 byte symmetric	Variable length asymmetric
Packet Numbers	Visible	Encrypted
Streams	Bidirectional	Uni and Bidirectional
Crypto Handshake	QUIC Crypto	TLS 1.3
HTTP Header Compression	HPACK	QPACK

How do I retrieve SNI from gQUIC?

This is a VERY frequently asked question.

If you don’t already extract SNI from gQUIC I recommend waiting for IETF QUIC.

How do I retrieve SNI from IETF QUIC?

This is also a VERY frequently asked question.

1) SNI will only be in Initial long header packets from client to server
2) Decrypt those packets(AES-GCM-128) using the server connection ID + QUIC version specific nonce (Full description in tls)
   Every new version of QUIC requires an updated key(or more)
3) Inside will be a CRYPTO frame which carries the TLS 1.3 ClientHello
4) Use existing TLS 1.3 parsing code to extract SNI

Suggestion: Whenever possible, use destination IP(s) instead

Full details in the resolution of Managability Issue #75

Migrating from gQUIC to IETF QUIC

Where is Google/Chrome now?

gQUIC **v46** is default enabled, v39 and v43 are still supported on the server

- Invariants-3 compatible with transport-19 packet types
- Future version will be invariants-6 compatible

V46 only supports 8 byte CIDs client -> server and 0 byte server -> client

Where is Google/Chrome now?

gQUIC **v46** is default enabled, v39 and **v43** are still supported on the server

- Invariants-3 compatible with transport-19 packet types
- Future version will be invariants-6 compatible

V46 only supports 8 byte CIDs client -> server and 0 byte server -> client

This section is about v43 -> v46 (invariants-03)

A document on the v43 to v46 header change is here

QUIC identification

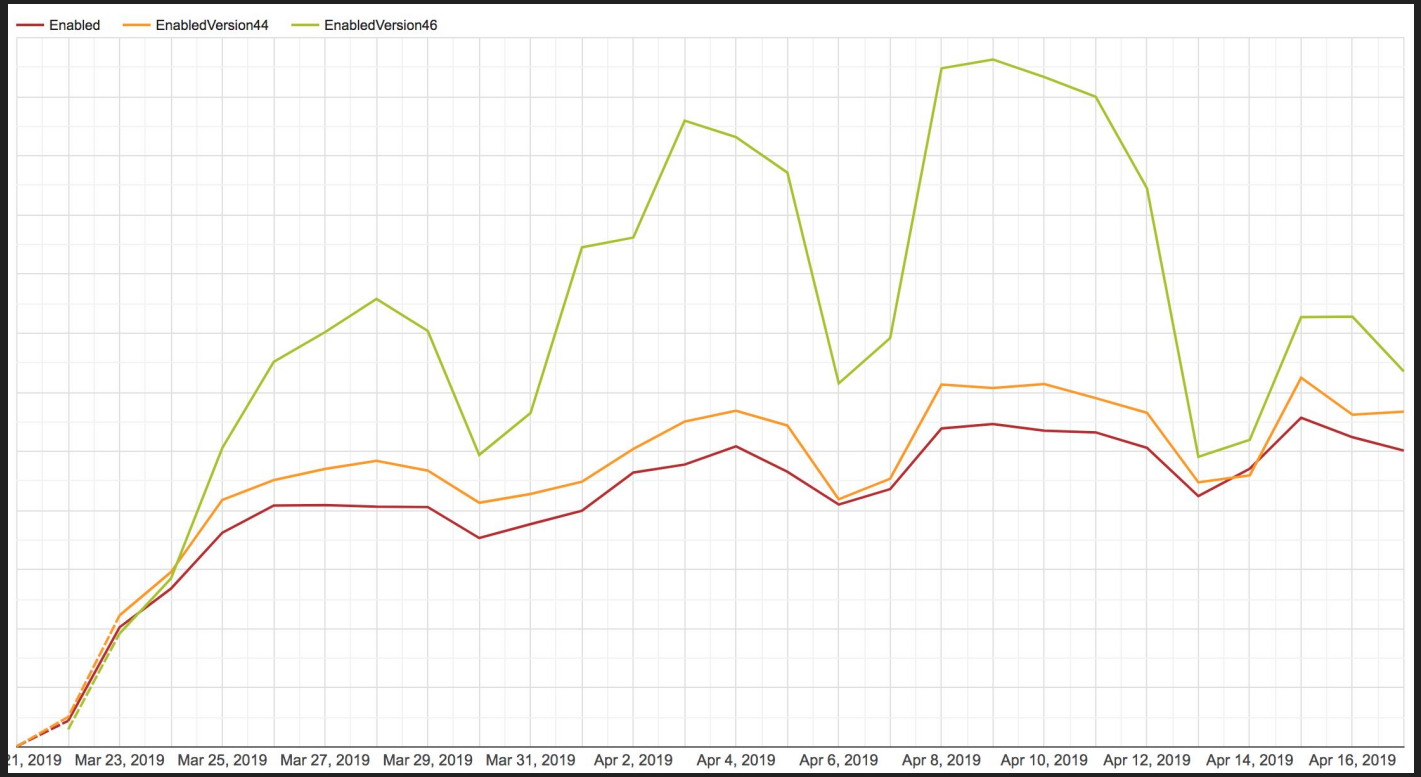
v46: Large increase in post-handshake blackholing

v46 blackholing almost 2x of v43

v46: Large increase in post-handshake blackholing

Suddenly improved April ~13th!

- Not server
- Not Chrome
- ?



What is TOO_MANY_RTOs?

On the 5th RTO, close the connection

Enabled by default on Chrome Desktop

Definitely a heuristic, but it’s better than nothing

- A great proxy for sudden blackholing
- But when were the connections being blackholed?

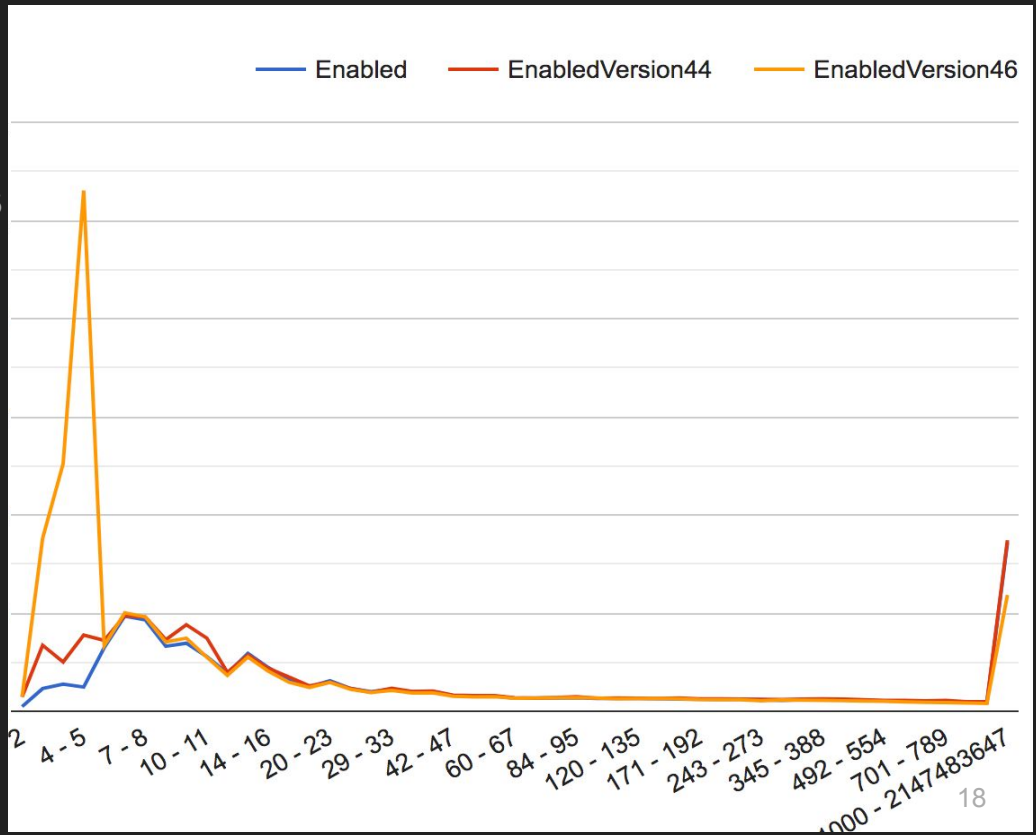
Note: gQUIC 5 RTOs ~= 7 PTOs because our 5 RTOs are after 2 TLPs are sent

When in a connection?

HUGE spike of TOO_MANY_RTOS at 2, 3 and 4 packets

Almost identical from 7 packets

# of packets received before TOO_MANY_RTOS



Turned out to be middlebox QUIC identification

Suddenly improved when a vendor updated their QUIC identification

Most users updated weekly, but some updated less frequently(ie: quarterly)

This caused a multi-week issue which was eventually diagnosed

How to block QUIC*

If QUIC is going to be blocked, ensure all packets in at least one direction are completely blocked.

Anything else causes **very** user visible failures**

* For any connection based transport, not just QUIC

** HTTP/3 clients are expected to have a fallback, others may not

Antivirus QUIC blocking

Suddenly, QUIC usage among Windows users dropped measurably!

Eventually traced it to a single AV company

- At the time, v46 was not blocked
- v46 was default enabled, and then v46 was blocked :(

Slight change in SNI location

Middleboxes have started inspecting gQUIC SNI in some locations

Most deployers haven’t told Google/YouTube*
so breakage is a risk as gQUIC -> IETF QUIC

Expect 2+ more changes that are likely to break SNI extraction
before IETF QUIC v1

*If you’d like to be notified of these changes, please contact me

What’s Next?

gQUIC is now closer to IETF QUIC, with many changes to go

- Some are visible to passive observers, so something will break…

IETF QUIC v2 is likely to target WebRTC and possibly Multipath and FEC

More Information

Questions?

IETF WG Page
Base IETF drafts: transport, recovery, tls, http, qpack
Ops IETF drafts: applicability, manageability
Chromium QUIC Code: cs.chromium.org