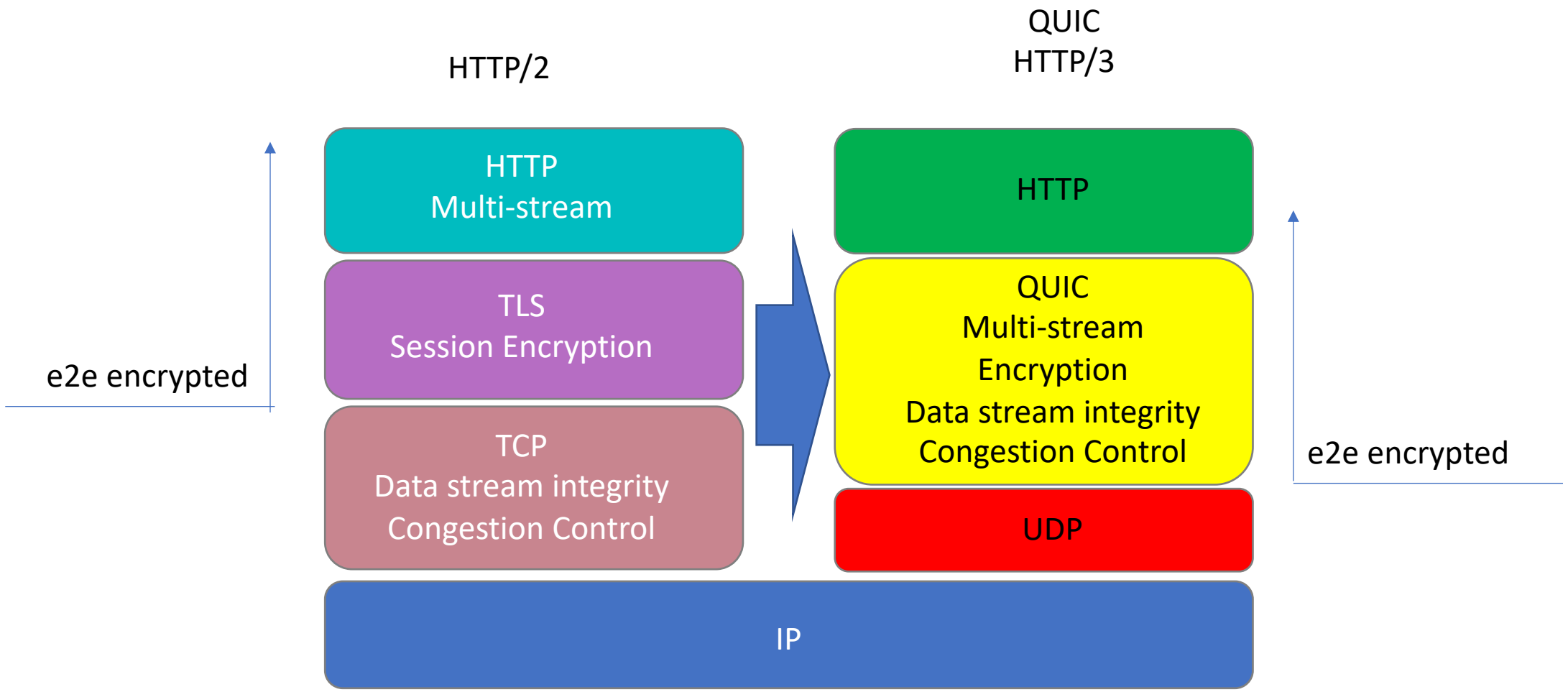
QUIC

Geoff Huston AM

APNIC

QUIC is

QUIC is a mashup of TCP and TLS



TCP is..

A transport protocol that constructs a reliable full duplex adaptive streaming service on top of an unreliable IP datagram service

- Uses a coordinated state between the two end systems without any network intervention or mediation
- Uses a sliding window to allow lost data to be resent
- Uses ACK-clocking to regulate the sending behaviour to match network path capacity estimate

TCP is NOT…

- Fully independent of the underlying platform's transport services
- Fully multi-stream (it has head-of-line blocking)
- Fully multi-path (yes, MP-TCP exists, but there are some outstanding issues here!)
- Address agile
- Free from on-the-wire network intervention (TCP control parameters are sent in the clear)
- Has e2e encryption as a second step / afterthought
- Everything for everyone – it relies on the application to perform data framing and in-band control

QUIC IS…

Constructed upon a transport level framing protocol that offers applications access to the basic IP datagram services offered by IP through the use of UDP

- All other transport services (data integrity, session control, congestion control, encryption) are shifted upwards in the protocol stack towards the application. A host platform may provide a QUIC API as part of the host library, but the application can also provide its own QUIC service independent of the host

QUIC is…

So much more than just “*encrypted TCP over UDP*”

- Support for multi-stream multiplexing that avoids head-of-line blocking and exploits a shared congestion and encryption state
- Faster - Combines transport and encryption setup exchange in a single 3-way exchange at session start, and supports fast reopen
- Customisable - QUIC implementations can use individual flow controllers per flow
- QUIC places its transport control fields inside the encryption envelope, so QUIC features minimal exposure to the network
- Supports record and Remote Procedure Call service models as well as bit-streaming and datagram services

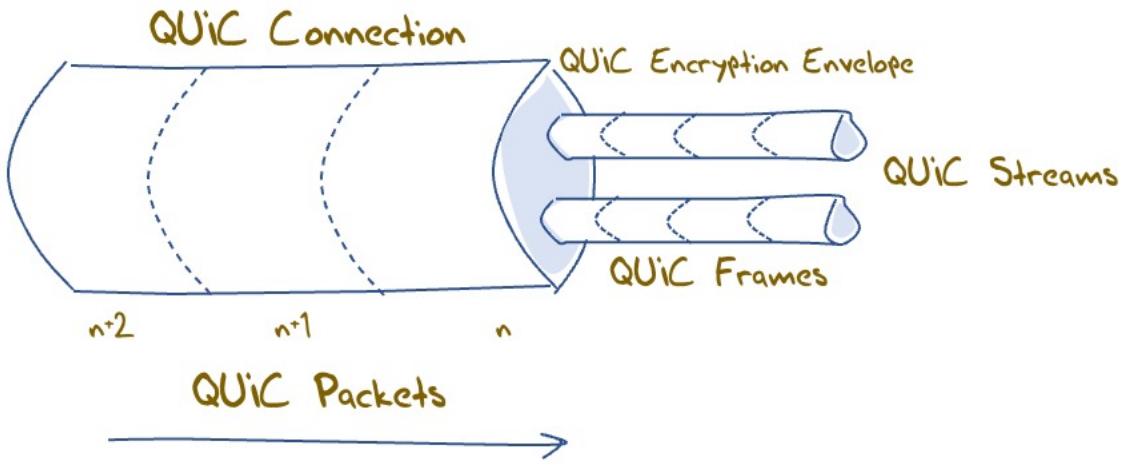
QUIC is address agile

- NATs are potentially hostile to QUIC because of the outer UDP wrapper
  - A NAT may rebind a QUIC session (shift the externally visible address/port of a host during a session), as NATs are not generally aware of UDP streaming states
- QUIC uses a persistent “connection ID”
  - If a host receives a QUIC frame with the same connection ID and a new source IP address / port it will send a challenge by way of a random value that should be echoed back. This is all performed within the e2e encryption envelope. That way a QUIC e2e session can map into new address/port associations on the fly

QUIC also…

- Is IP **fragmentation intolerant** – QUIC uses PMTUD, or defaults to 1,200 octet UDP payloads
- **Never retransmits a QUIC packet** – retransmitted data is sent in the next QUIC packet number – this avoids ambiguity about packet retransmission
- Extends TCP SACK to **256 packet number ranges** (up from 3 in TCP SACK)
- Separately encrypts each QUIC packet – no inter-packet dependencies on decryption
- May load multiple QUIC packets in a single UDP frame

QUIC flow structuring



A QUIC connection is broken into "streams" which are reliable data flows – each stream performs stream-based loss recovery, congestion control, and relative stream scheduling for bandwidth allocation

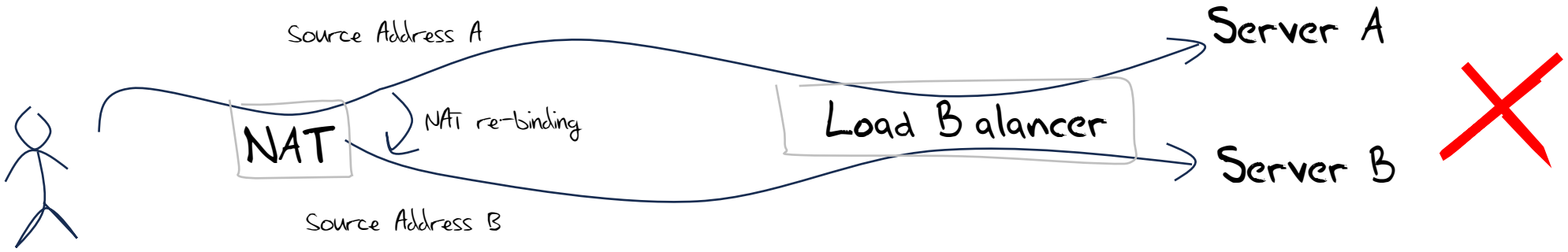
QUIC also supports unreliable encrypted datagram delivery

QUIC and Remote Procedure Calls

- By associating each RPC request/reply with a new stream, QUIC can support asynchronous RPC transactions using reliable messaging
  - This can handle lost, mis-ordered and duplicated RPC messages without common blocking or throttling

QUIC and Load Balancing

- This assumes that a front-end load balancer is capable of performing load balancing on UDP flows using the UDP connection 5-tuple
- If the remote end performs NAT rebinding the load balancer will be thrown by this shift, and it has no direct visibility into the e2e session to uncover the connection ID
- Using UDP to carry sustained high-volume streams may not match the internal optimisations used in server content delivery networks



QUIC and Load Balancing

- This assumes that a front-end load balancer is capable of performing load balancing on UDP flows using the UDP connection 5-tuple
- If the remote end performs NAT rebinding the load balancer will be thrown by this shift, and it has no direct visibility into the e2e session to uncover the connection ID
- Using UDP to carry sustained high-volume streams may not match the internal optimisations used in server content delivery networks

- **If we really want large scale QUIC with front-end load balancing and if we still need to tolerate NATs then we will need to think about how the end point can share the connection ID state with its front-end load balancer, or how to terminate the QUIC session in the front-end and use a second session to a selected server**

QUIC and DOS

- Very little lies outside the encryption envelope in QUIC
- Which means all incoming packets addressed to the QUIC port need to be decrypted
- But the QUIC session uses symmetric crypto so the packet decode overhead is far smaller than an asymmetric crypto load for the same packet rate

- It’s not the best answer, but it’s not disastrous either!

QUIC is:

- A logical evolutionary step for transport services, providing more flexibility, faster connection setup, and a larger set of transport services

- It’s what we should expect from a capable modern transport protocol!

Measuring QUIC

Triggering QUIC in HTTP

Use the DNS to trigger QUIC:

- Set up an HTTPS record for each server name, with value: **`alpn="h3"`**

Use content-level controls to trigger QUIC:

- Add **`Alt-Svc: h3=":443"`** to the HTML headers

(This second method requires a subsequent query in a distinct HTTP session to allow the client to use the Alt-Svc capability.)

Triggering QUIC in HTTP

Use the DNS to trigger QUIC:

- Set up an HTTPS record for each server name, with value: **`alpn="h3"`**

Use content-level controls to trigger QUIC:

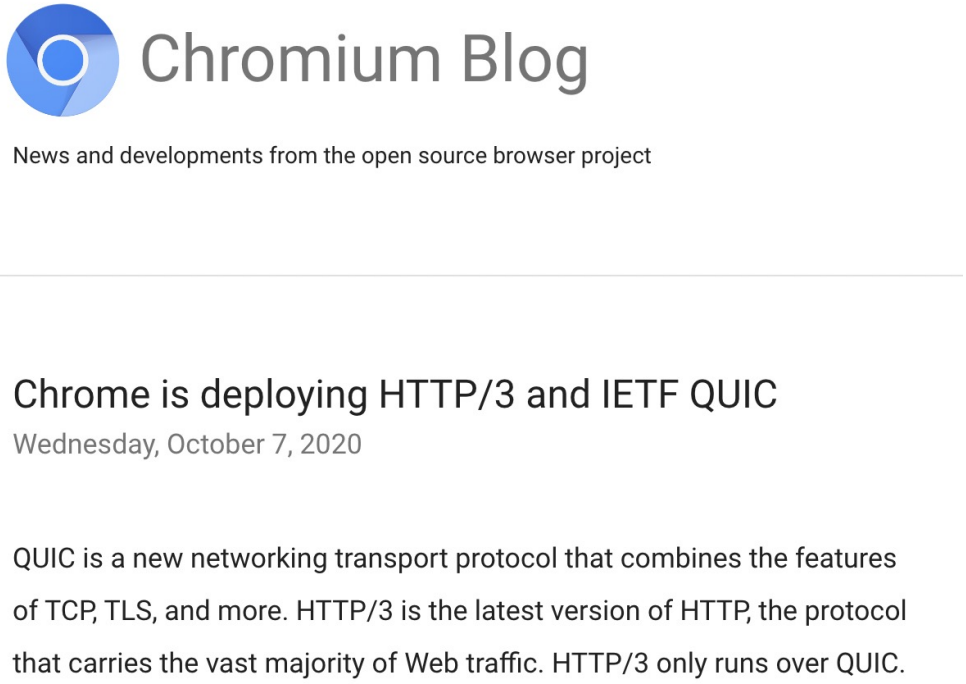
- Add **`Alt-Svc: h3=":443"`** to the HTML headers

First Fetch

Second Fetch

Setting Expectations

- Chrome has a dominant share of browser instances - roughly, some 65%*
- And Chrome has been supporting a switch to QUIC via the Alt-Svc directive since 2020



Chromium Blog

News and developments from the open source browser project

Chrome is deploying HTTP/3 and IETF QUIC

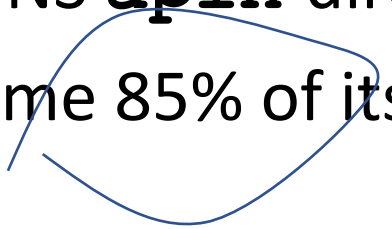
Wednesday, October 7, 2020

QUIC is a new networking transport protocol that combines the features of TCP, TLS, and more. HTTP/3 is the latest version of HTTP, the protocol that carries the vast majority of Web traffic. HTTP/3 only runs over QUIC.

** Oberlo.com*

Setting Expectations

- Chrome has a dominant share of browser instances - roughly, some 65%*
- And Chrome has been supporting a switch to QUIC via the Alt-Svc directive since 2020
- And Apple Safari is now supporting QUIC, using the DNS **`apln`** directive
- So a QUIC-aware server platform should be seeing some 85% of its sessions using QUIC – right?

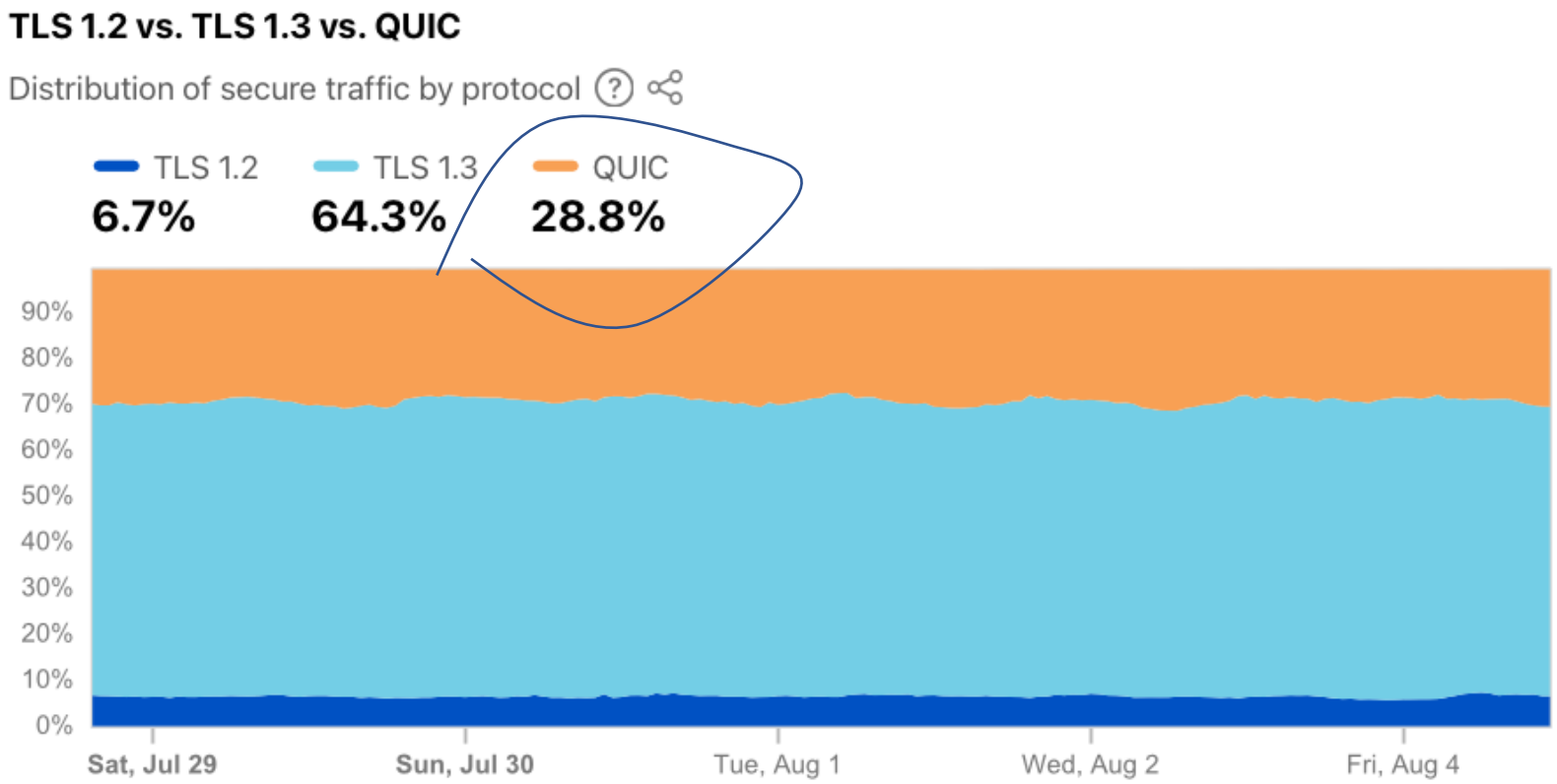


** https://gs.statcounter.com/browser-market-share*

Cloudflare's Numbers

Cloudflare reports a far lower level of QUIC use

Cloudflare Radar



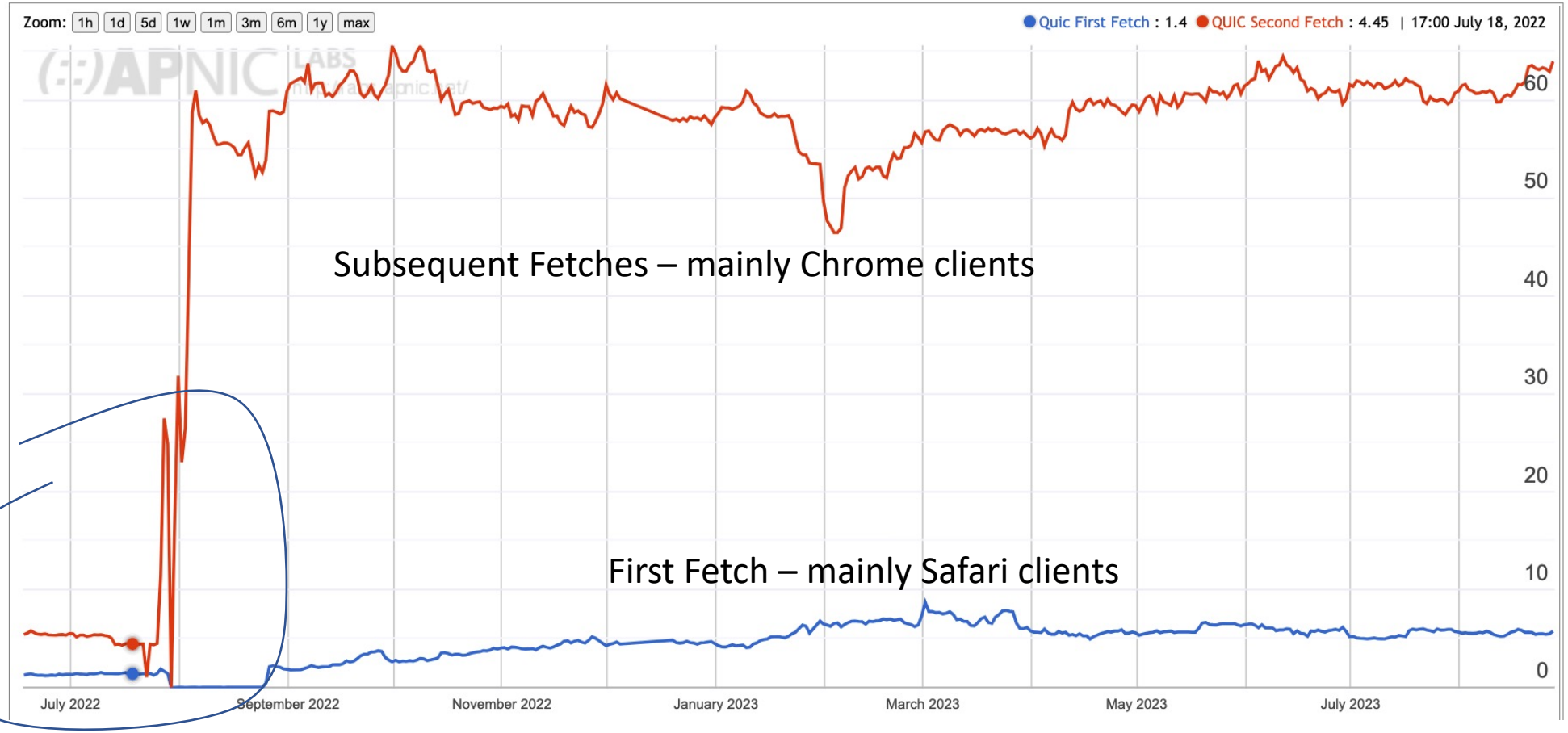
APNIC's QUIC measurement

- We have configured a server to support QUIC sessions
- We support both DNS and content triggers
- The content trigger requires us to measure across multiple fetches within each measurement
  - Which means that we need to carefully set the HTTP/2 session keepalive timer to make this work as intended

Server Session Keepalive Timers

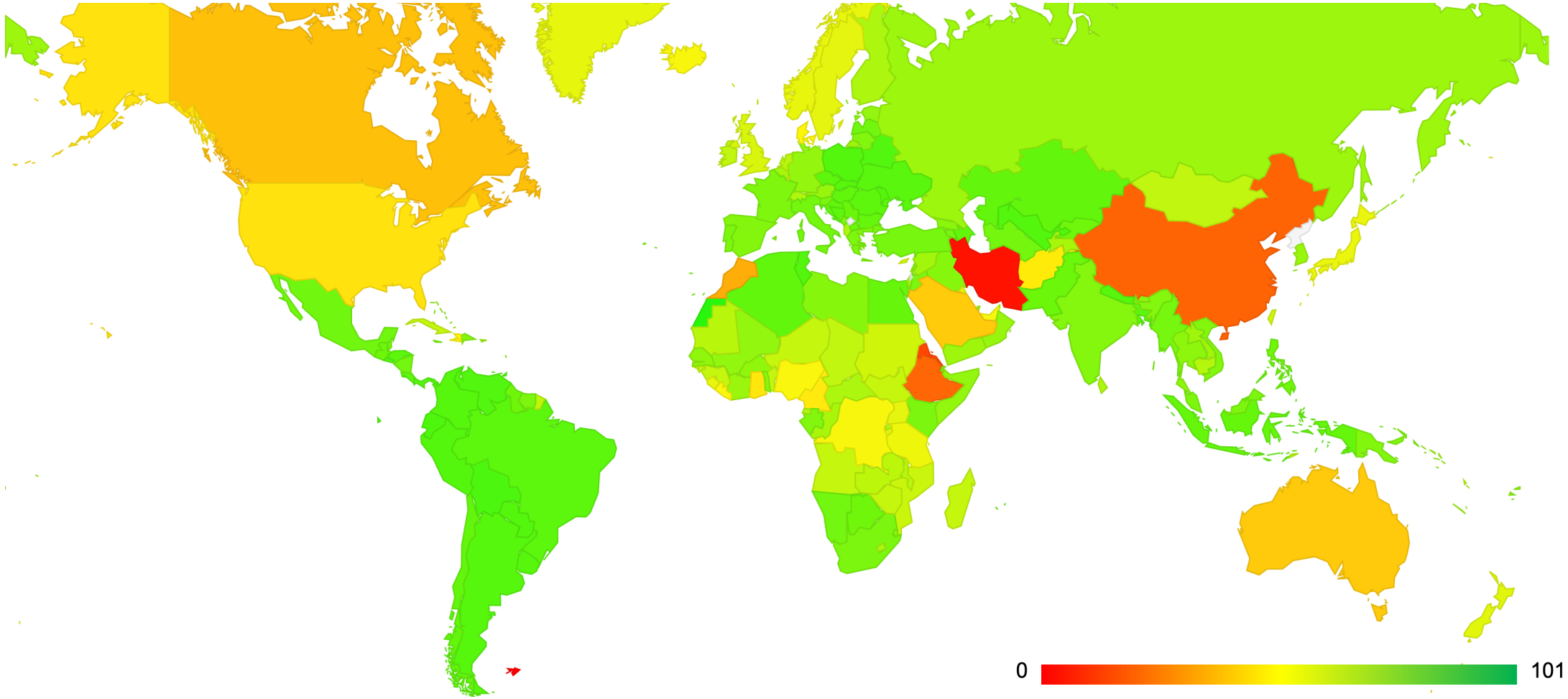
- After much searching under many rocks we were advised that a **server keepalive** timer value of 1 second is too small, as the server drops the QUIC connection too aggressively and the browser client then drops back to using HTTP/2
- The default value of 65 seconds for the server keepalive interval seems to be too long
- So we used a **server keepalive** value of 20 seconds…

QUIC Use

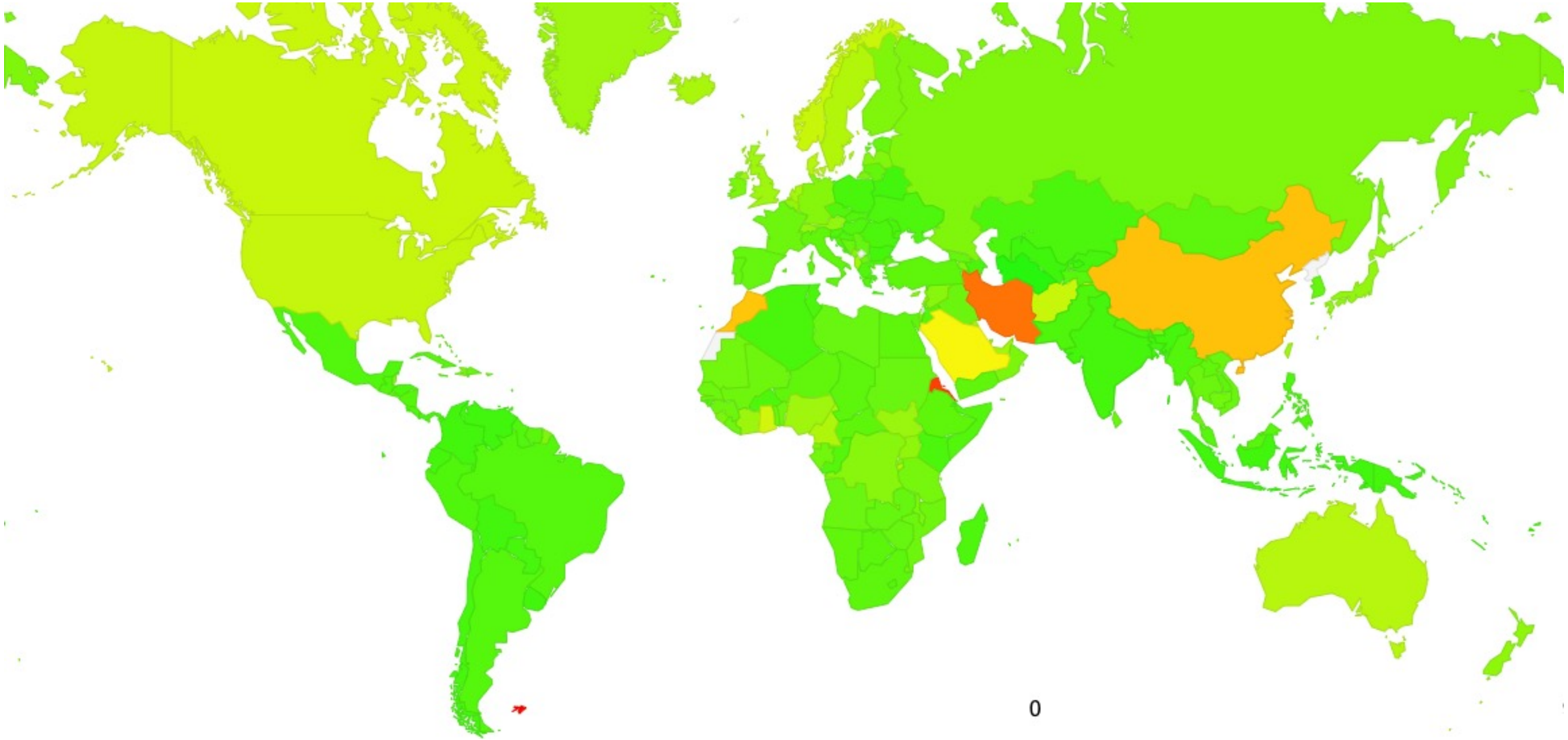


Playing with keepalive parameters!

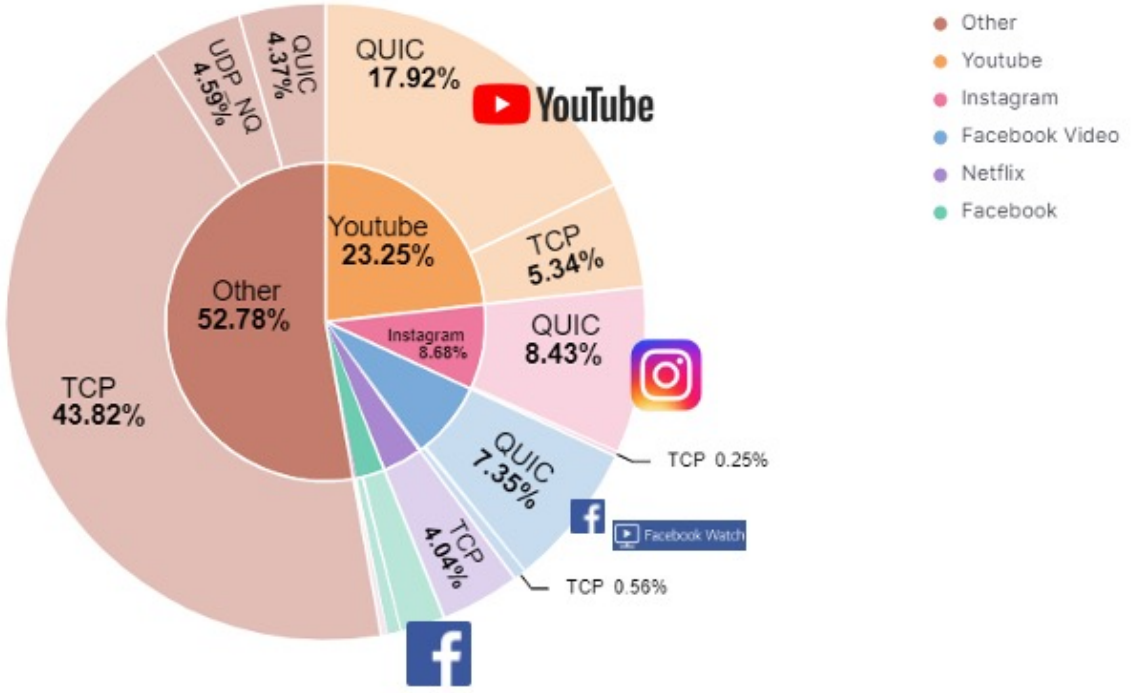
QUIC Use - July 2023



QUIC Use - May 2024



Network Traffic Volume



*source EU Operator 2022

Presentation to RIPE 86: The New Encrypted Protocol Stack and How to Deal with it – Bart van de Velde, Cisco

Why is QUIC important?

Because QUIC is fast

Because QUIC encrypts everything

- No visible transport control settings
- No visible Server Name Indication in the crypto-setup
- No visible traffic profile other than inter-packet timing
- And if you use a MASQUE-based VPN then there no residual visibility!

Because QUIC is an application capability

- QUIC can interact with the platform through the UDP API, so all of QUIC can be implemented within the application. This gives the application more control over its service outcomes and reduces external dependencies

What does this mean for TCP?

It’s not looking all that good for TCP’s prospects

- QUIC not only does faster start up, but it supports multi-channel in a frictionless manner
- QUIC resists network operator efforts to perform traffic shaping through direct manipulation of TCP control parameters
- QUIC allows the application service provider to control the congestion behaviour of its sessions

What does this mean for TCP?

Normally you would expect any transition from TCP to QUIC to take forever

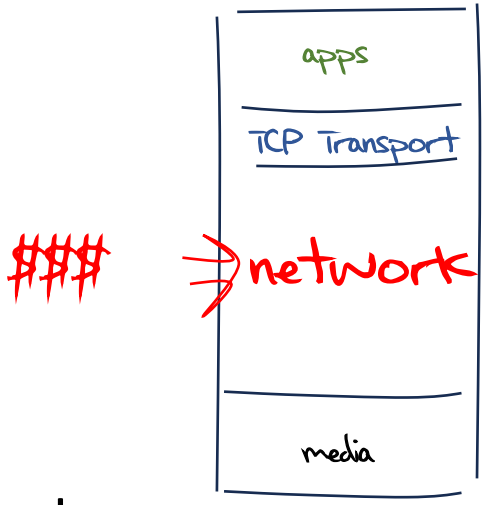
BUT:

- QUIC gives benefit to adopters through more responsive web services
- QUIC does a better job of hiding content, which is a benefit to the service operator
- QUIC has fewer external dependencies
- QUIC can be deployed on a piecemeal basis

So it all may be over for TCP in a very small number of years!

What does this mean for the Internet?

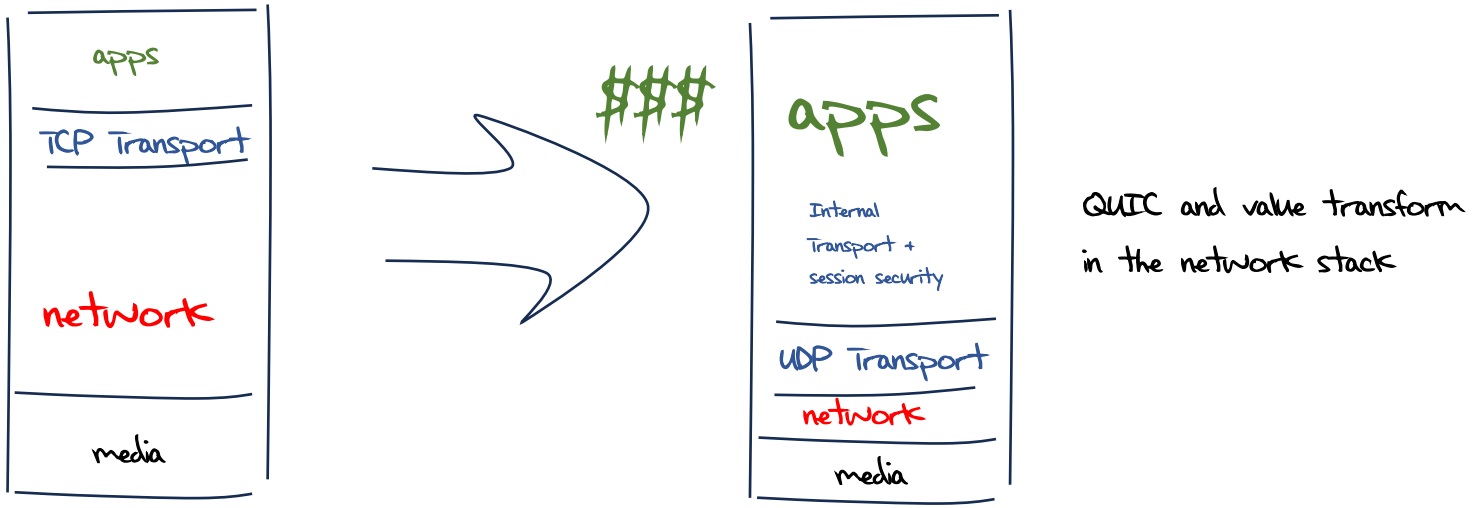
- IP was a ***network*** *protocol* that provided services to attached devices
- The network service model used by IP was minimal
  - Packets may be dropped, fragmented, duplicated, corrupted and/or reordered on their path through the network
  - It's left to the edge systems to recover from this network behaviour.
- Efforts to expand the network's role have foundered
  - QoS has just got nowhere!
  - Various forms of source-directed forwarding are resisted by network operators who want control over traffic engineering
- Networks took up a role of defending the network resource against aggressive application behaviour
- Some networks enabled user surveillance



The new Networking Space

And this is why QUIC is so interesting – it is pushing both network carriage and host platform into commodity roles in networking and allowing applications to effectively customize the way in which they want to deliver services and dominating the entire networked environment

QUIC is the application's view of what Transport should be!



What does this mean for the Internet?

- The relationship between applications, hosts and networks has soured into mutual distrust and suspicion
- The application now defends its integrity by wrapping up as much of the service transaction with encryption and indirection
- QUIC (and MASQUE) is an intrinsic part of this process of wrapping up traffic in encryption and redirection
- For the network operator there is little left to see
- And I suspect that there is no coming back from here!

What can a Network Operator Do?

- When **all** customer traffic is completely obscured and encrypted?
  - Traffic Shaping?
  - Regulatory Requirements for traffic interception?
  - Load Balancing / ECMP

The new Internet Space

“What you can’t dominate, you commoditise*”

- Vertically integrated service providers have faded away into history - the deregulated competitive service industry continues to specialize rather than generalize at every level
- Carriage is no longer an inescapable monopoly - massively replicated content can be used as a substitute for many carriage service elements
- Control over the platform is no longer control over the user. Operating systems have been pushed back into a basic task scheduling role, while functions are being absorbed into the application space

* A related quote is Peter Thiel’s “Competition is for losers!”

Thanks!