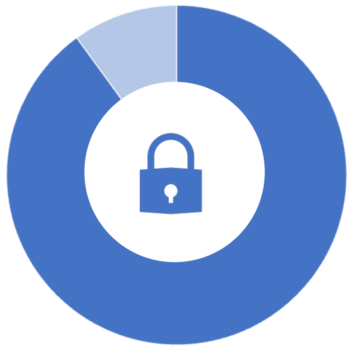
Goodbye TCP?

Geoff Huston AM
APNIC Labs

What have we done to the Internet?

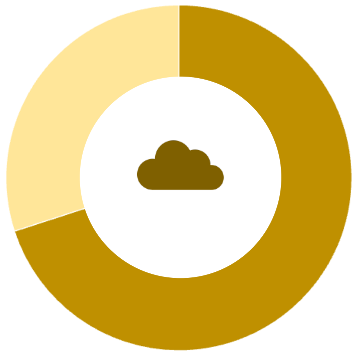
More than 90% of all Internet traffic uses encrypted payloads



What have we done to the Internet?

More than 90% of all Internet traffic uses encrypted payloads

More than 70% of all Internet traffic is sourced from Cloud servers

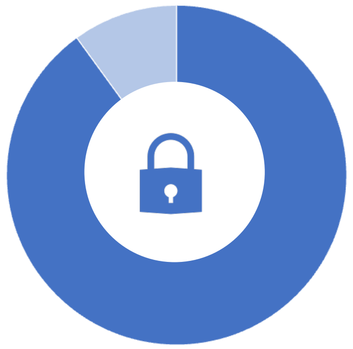


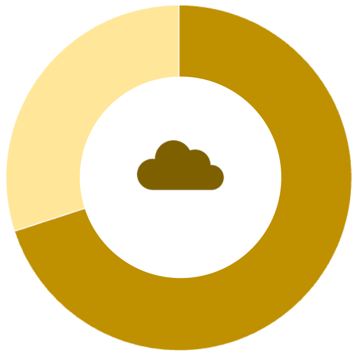
What have we done to the Internet?

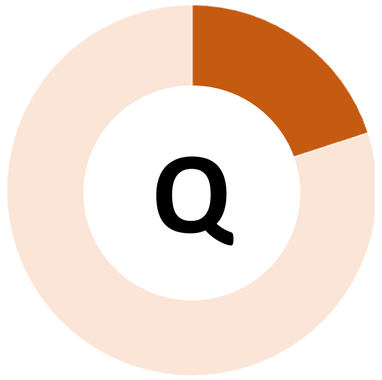
More than 90% of all Internet traffic uses encrypted payloads

More than 70% of all Internet traffic is sourced from Cloud servers

More than 20% of all traffic by volume uses QUIC







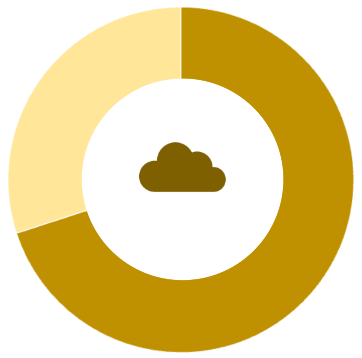
* https://ripe86.ripe.net/wp-content/uploads/presentations/49-The_New_Encyrpyted_Stack_RIPE.pdf

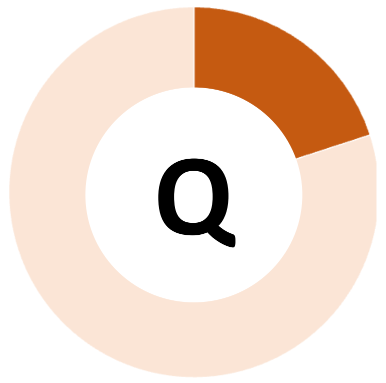
What have we done to the Internet?

More than 90% of all Internet traffic uses encrypted payloads

More than 70% of all Internet traffic is sourced from Cloud servers

More than 20% of all traffic by volume uses QUIC





Whatever is happening to TCP on the Internet?

* https://ripe86.ripe.net/wp-content/uploads/presentations/49-The_New_Encyrpyted_Stack_RIPE.pdf

Why was the Internet Protocol so special?

- What made it so radically different from other network protocols of the 1980’s?
- Why are we still using IP today?

- The answer is **TCP**!

TCP is..

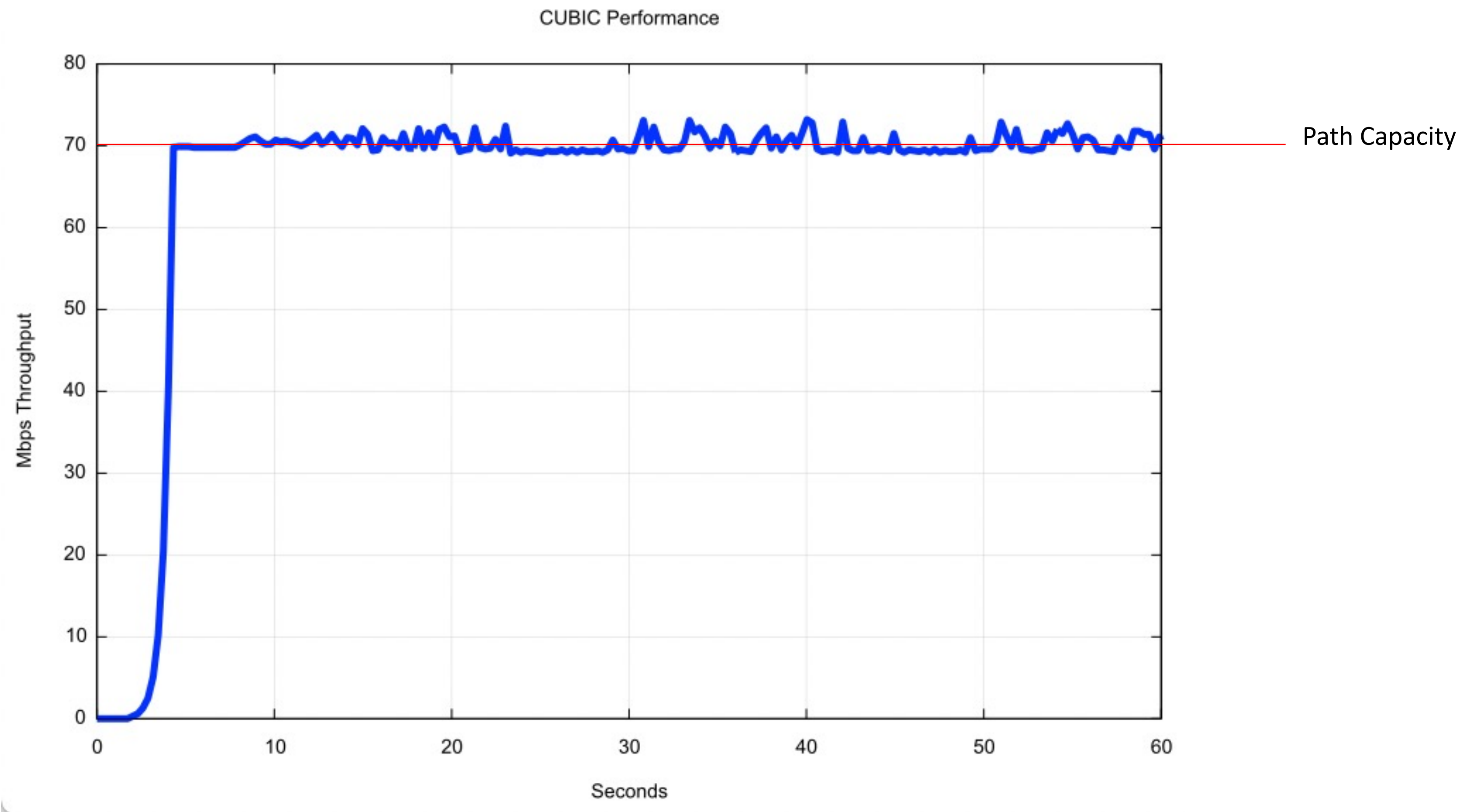
A transport protocol that constructs a reliable full duplex adaptive streaming service constructed on top of an unreliable IP datagram service

- 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
- Has been tweaked to support* multi-stream and RPC transport models

* to some degree!

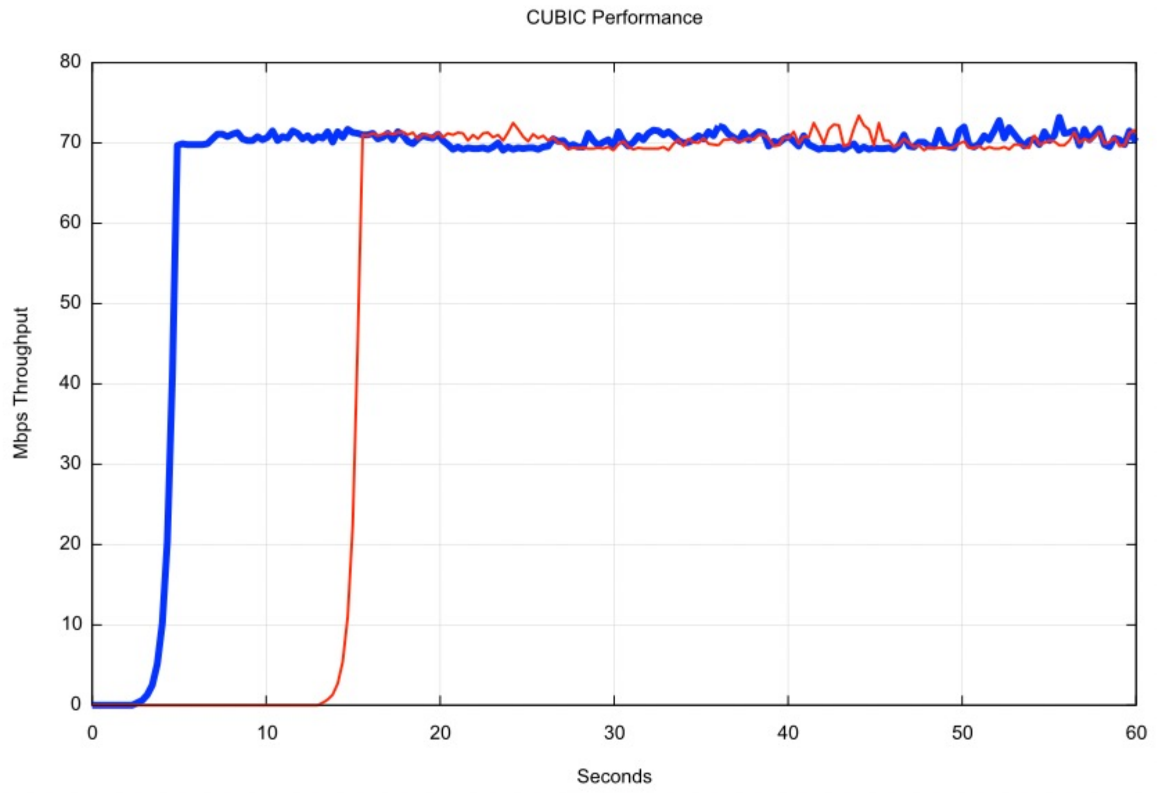
TCP is highly efficient

- TCP will attempt to use all available network bandwidth



TCP is friendly!

- TCP performs self-moderation to share available network capacity fairly across active TCP sessions



TCP is flexible

- TCP behaviour is controlled by the server, not the client
- The same client TCP engine can be used for a variety of server-side congestion control algorithms
    - Reno – classic AIMD
    - CUBIC
    - BBR
    - MulTCP
    - …
- Different servers can use different TCP behaviours to achieve different service outcomes

Pushing TCP harder

We have been prodding and poking at TCP for decades:

- **T/TCP** – TCP for Transactions which attempted to stash the payload and the response into the TCP handshake (since discarded due to its truly woeful security model!)
- **Multipath TCP** – spread TCP session traffic across multiple outbound interfaces to allow for host-based load sharing and session resiliency (largely abandoned after just one large scale implementation)
- **Initial Window inflation** – improve short flow performance by commencing the flow with a larger initial window size
- **Cached TCP performance** – cache the achieved TCP performance to a server and start the next TCP session in congestion avoidance, bypassing initial slow start
- **ECN signalling** – have the network mark packets when queries are forming in the network path, allowing hosts to react prior to packet loss events

But

- There are some performance behaviours that you just can’t fix using TCP tweaking

TCP isn't…

- Fully independent of the underlying platform's transport services
- Fully multi-stream (TCP has head-of-line blocking)
- Free from on-the-wire network intervention (TCP control parameters are sent in the clear)
- Has e2e encryption as a second step / afterthought
- Can't support reliable datagram behaviour only with full session support
- Relies on the application to perform data framing and in-band control
- **TCP isn't everything for everyone**

Choices:

- Keep adjusting TCP at the edges and leave the basic TCP behaviour alone

or

- Pull TCP functionality up into the application and implement more radical changes to the transport behaviour at the application level

QUIC is…

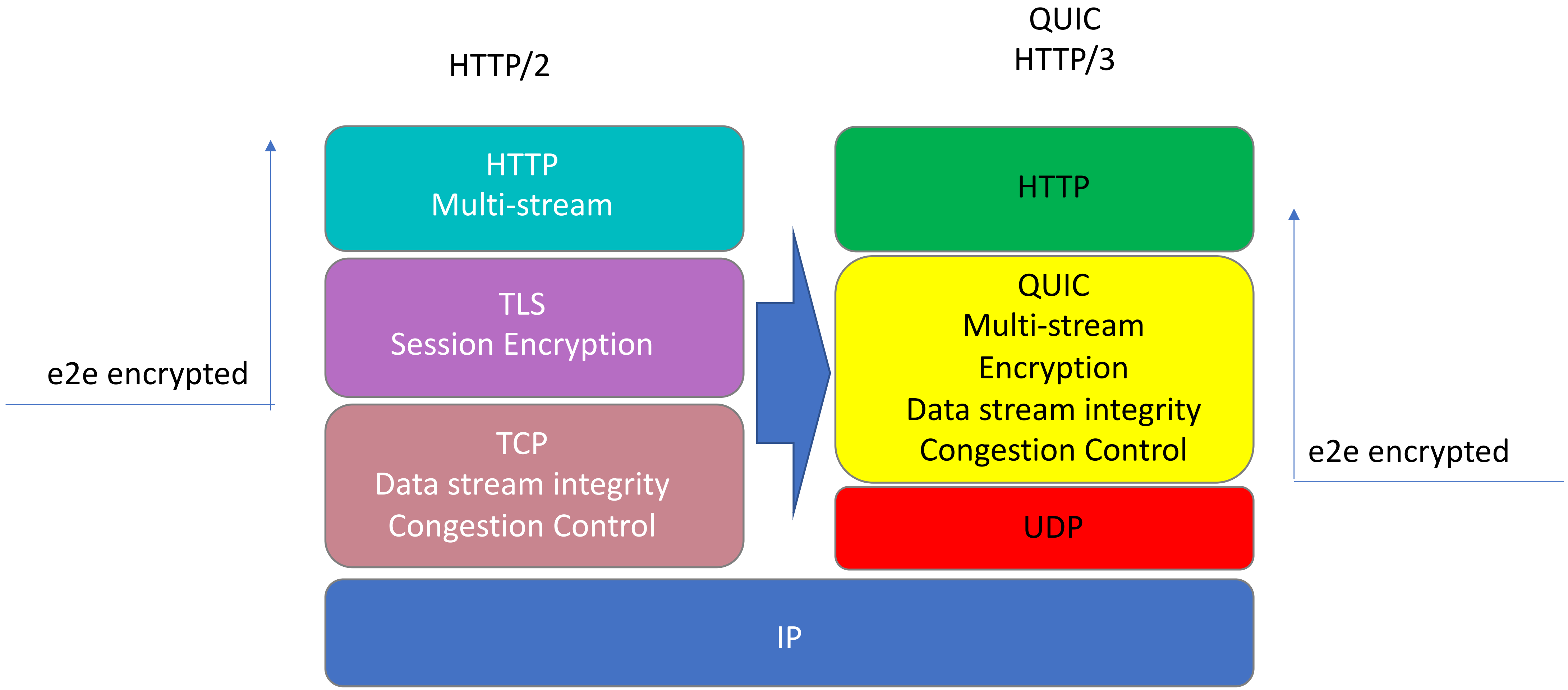
The second path!

Constructed upon a base datagram framing protocol through the use of UDP, placing all transport functions into the UDP payload

- All end-to-end transport services (data integrity, session control, congestion control, encryption) are shifted towards the application.
- A platform may provide a QUIC API, but the application can also provide its own service

QUIC is…

- TLS transport layered over a UDP substrate



QUIC is…

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

- More Flexible - 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
- Customisable - QUIC implementations can use individual flow controllers per flow
- QUIC places its transport control fields inside the encryption envelope, so QUIC has minimal exposure to the network
- Supports record and RPC service models as well as streaming and datagram

QUIC is…

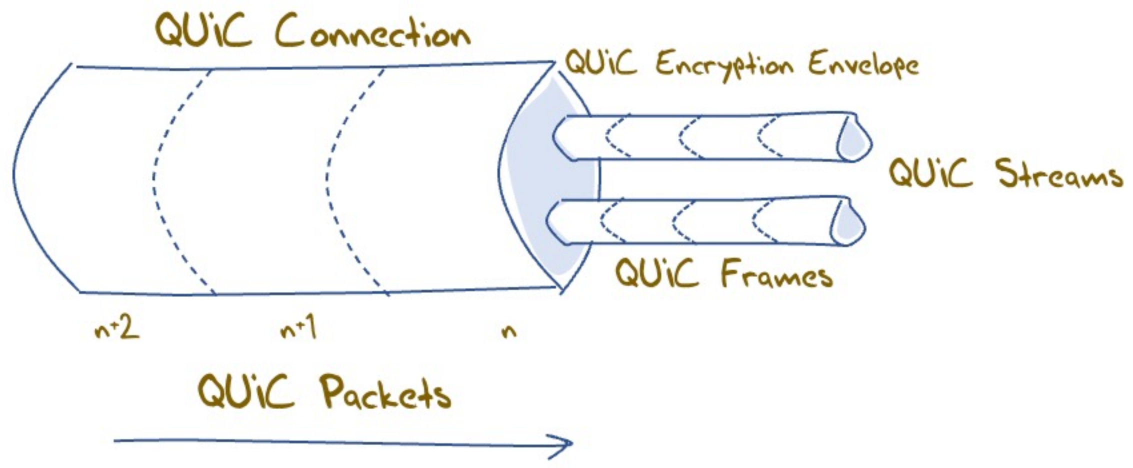
Endpoint address agile

- NATs are potentially hostile to QUIC because of the outer UDP wrapper
  - A NAT may rebind (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 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 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)
- Separately encrypts each QUIC packet
- May load multiple QUIC packets in a single UDP frame (countering some forms of network level packet fingerprinting efforts)

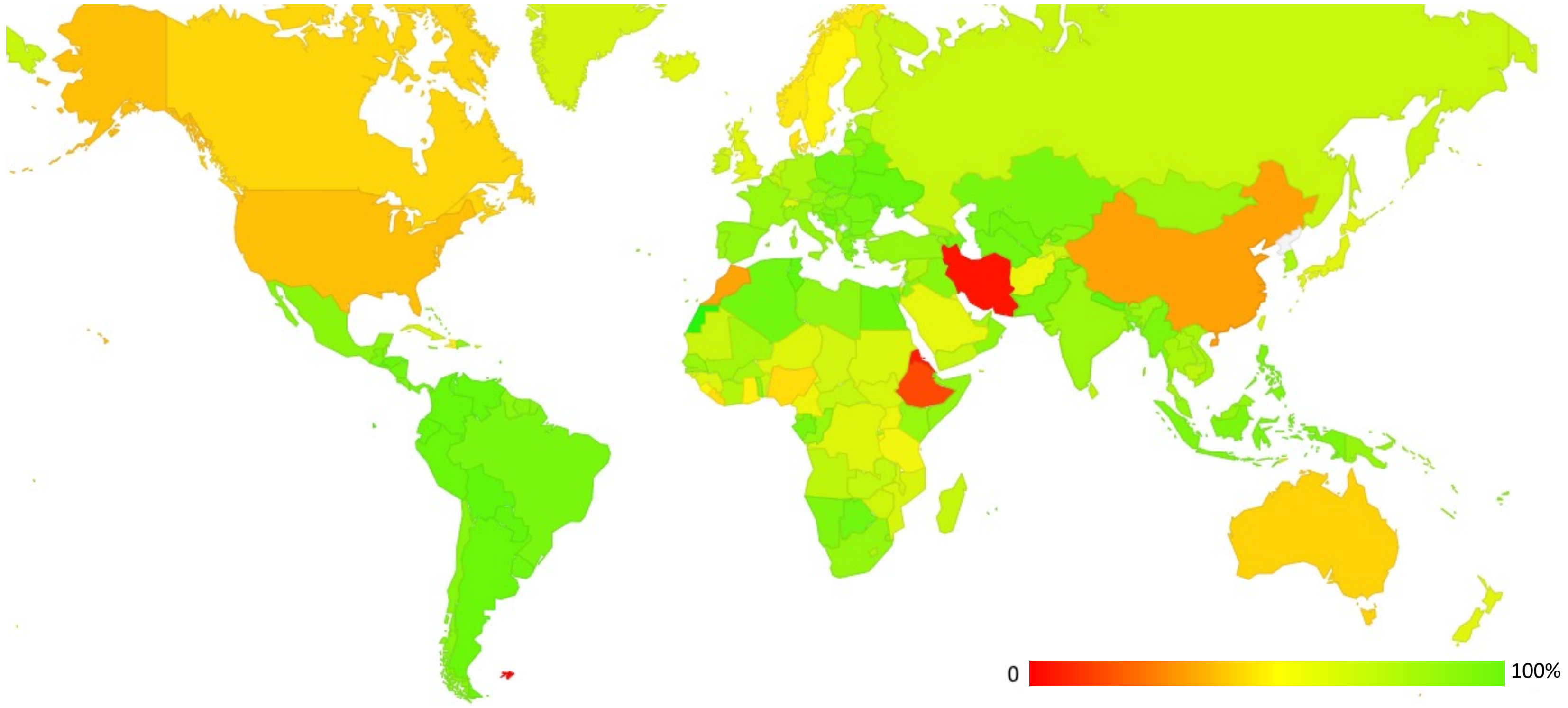
QUIC flow structuring



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

Where can we see QUIC capability today?

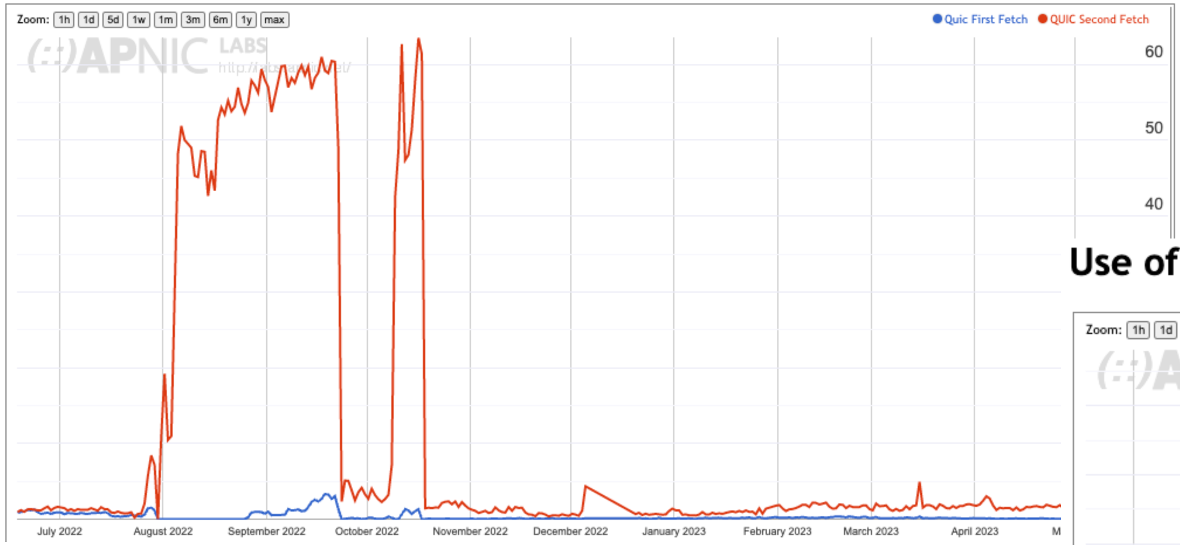


https://stats.labs.apnic.net/quic

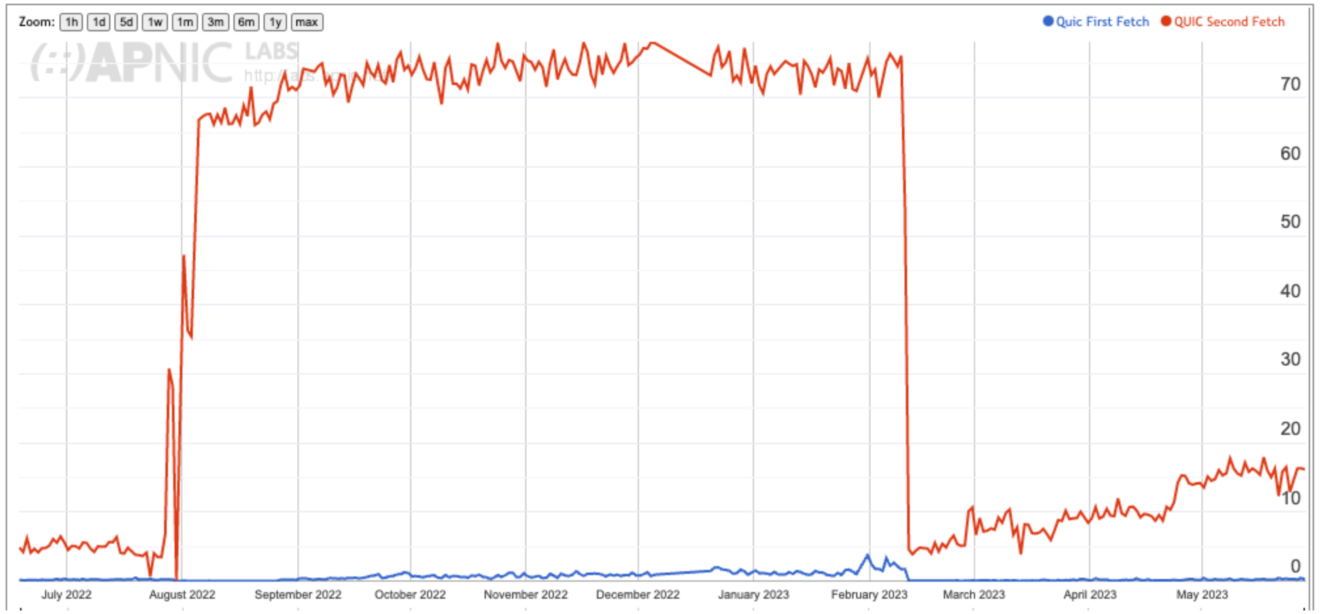
July 2023

Almost Everywhere

Use of HTTP/3 for Iran (Islamic Republic of) (IR)

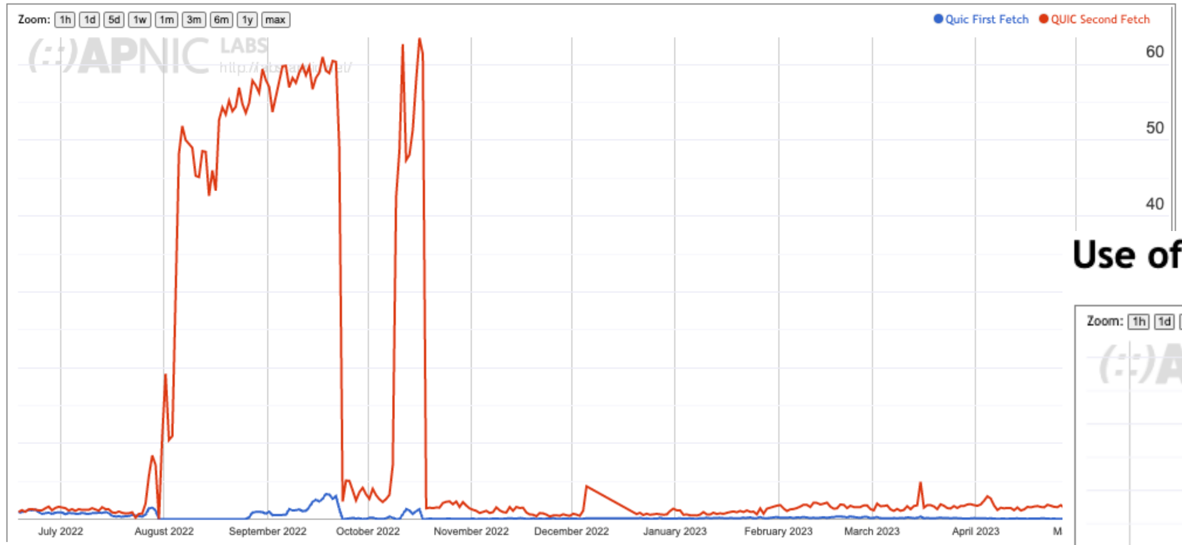


Use of HTTP/3 for Ethiopia (ET)

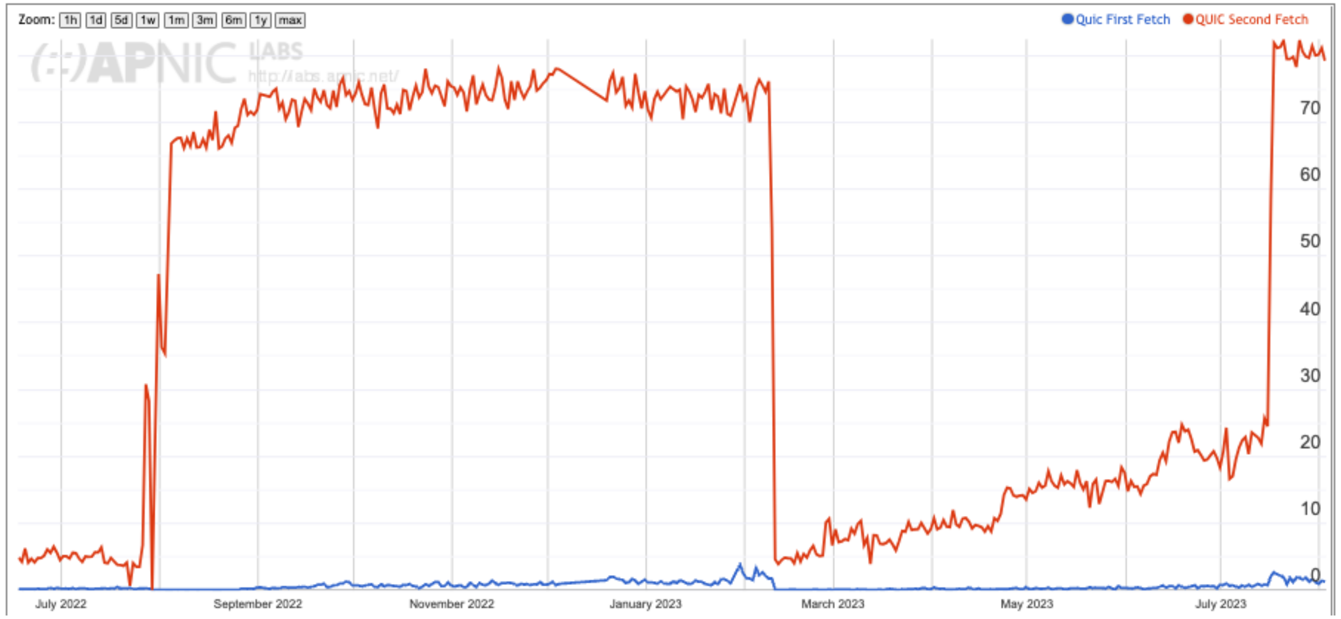


Almost Everywhere

Use of HTTP/3 for Iran (Islamic Republic of) (IR)

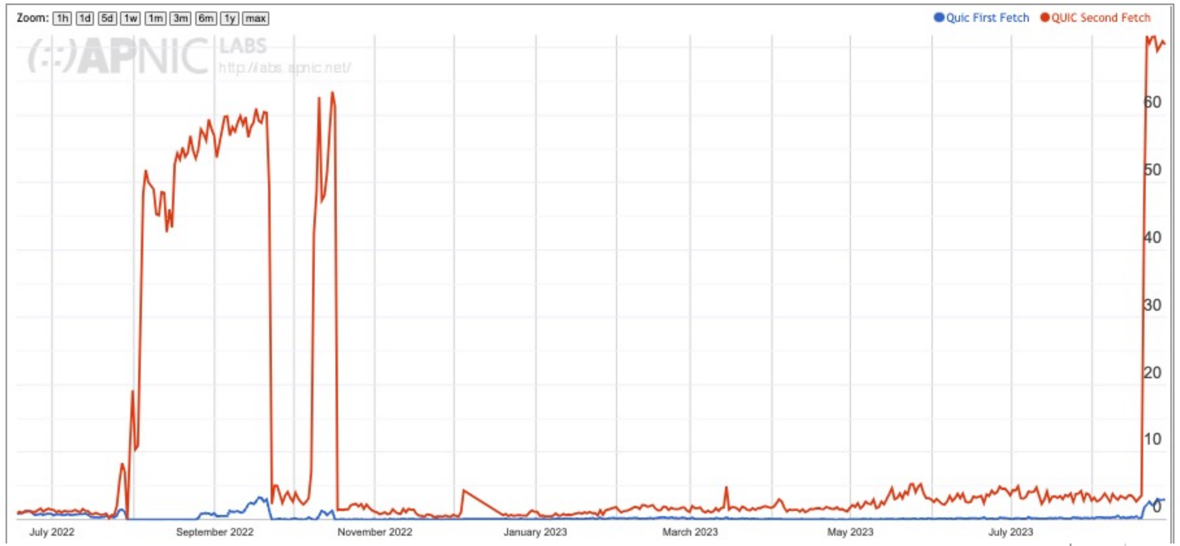


Use of HTTP/3 for Ethiopia (ET)

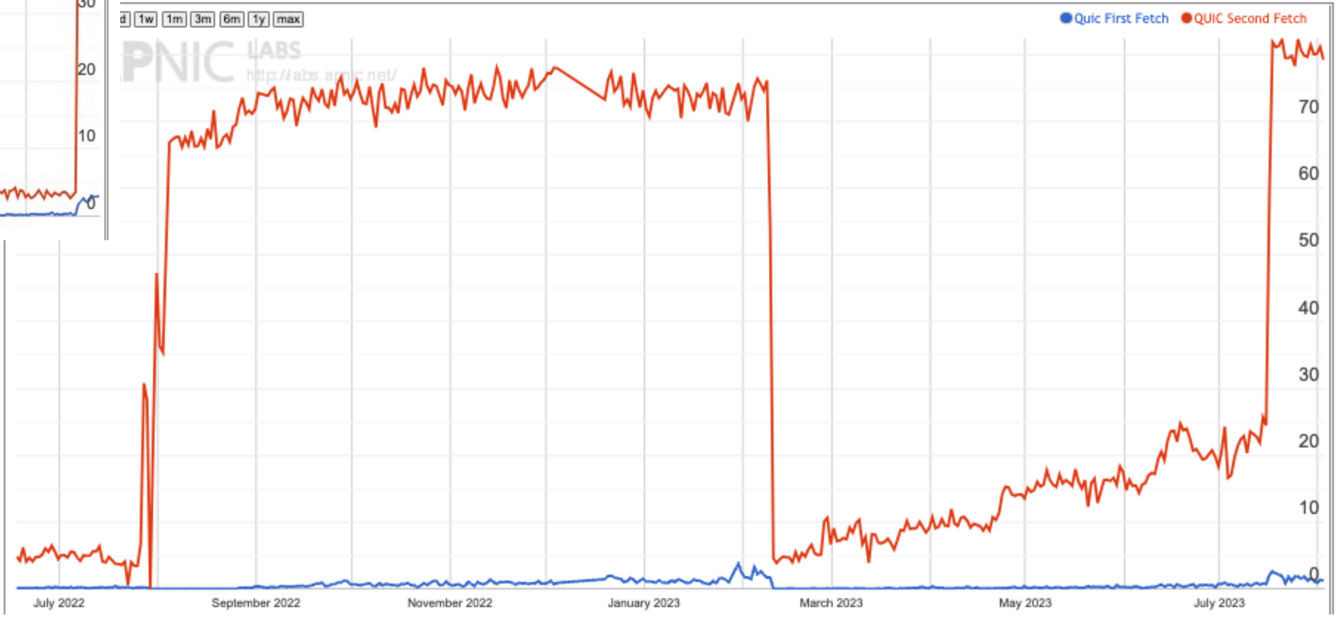


Almost Everywhere

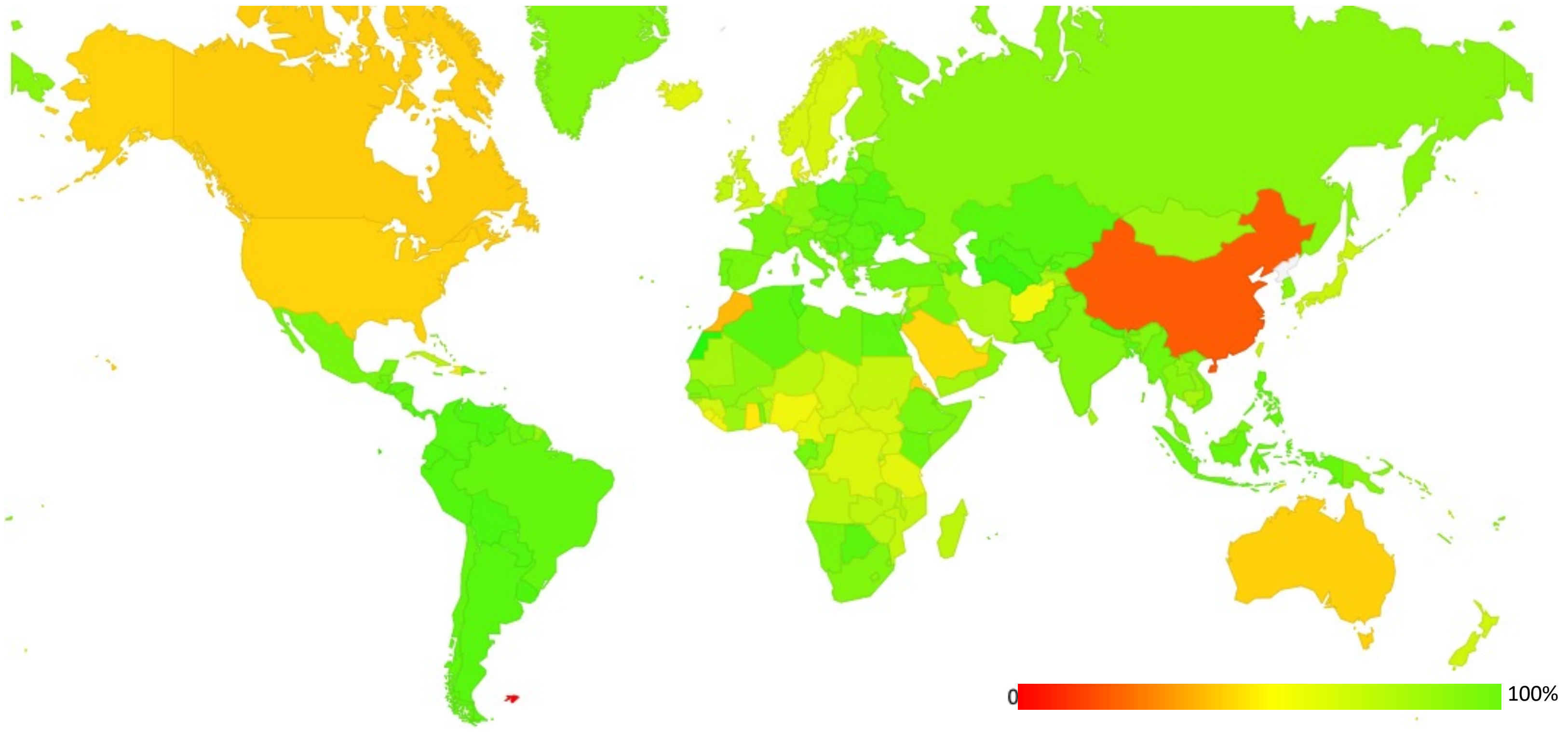
Use of HTTP/3 for Iran (Islamic Republic of) (IR)



HTTP/3 for Ethiopia (ET)



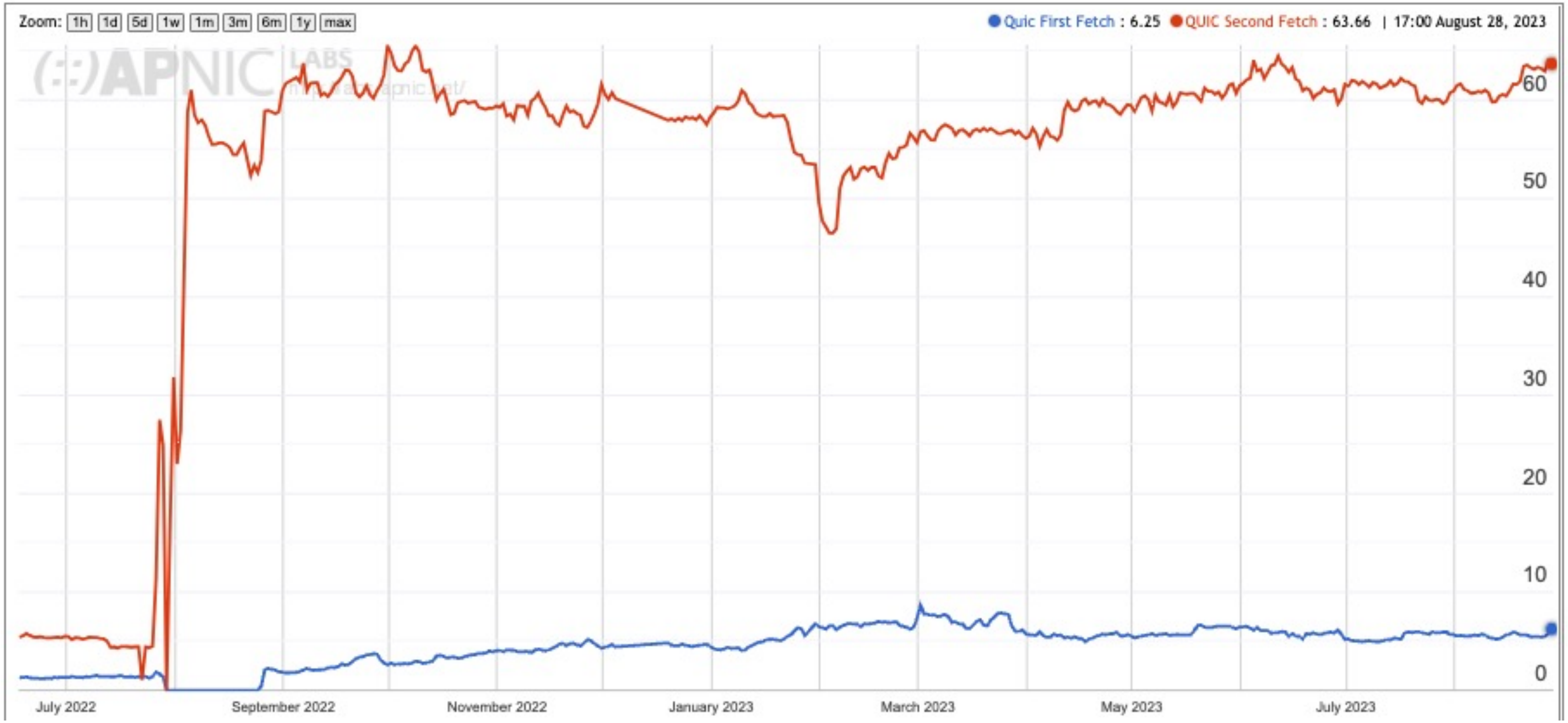
Almost Everywhere



https://stats.labs.apnic.net/quic

September 2023

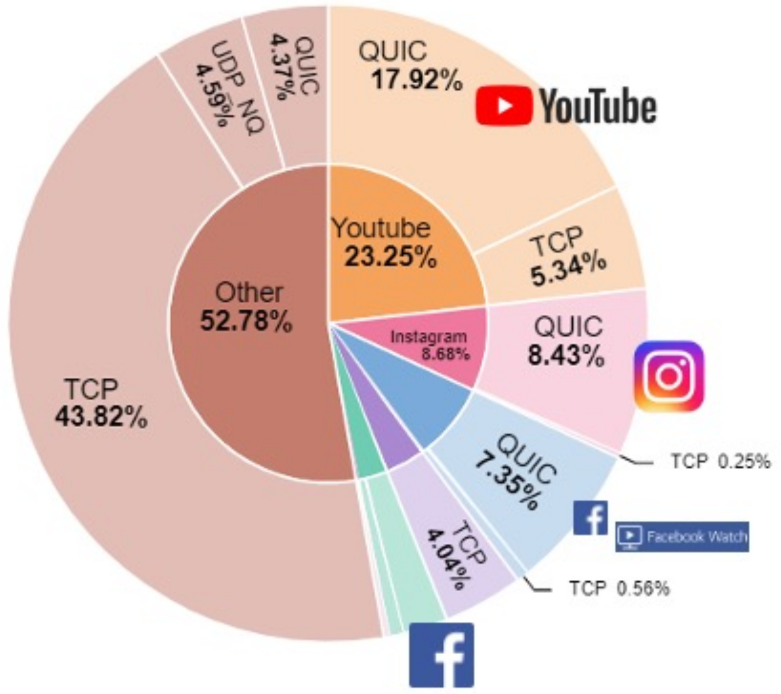
Overall Uptake



Measuring QUIC Use

- This is a **capability** measure – how many users can use QUIC is the server advertises that is can serve web content using QUIC (HTTP/3)?
- The measurement uses both HTTPS capability in the DNS (used by Apple Safari clients) and content-tagging with Alt-Svc directive (used by Chrome clients)

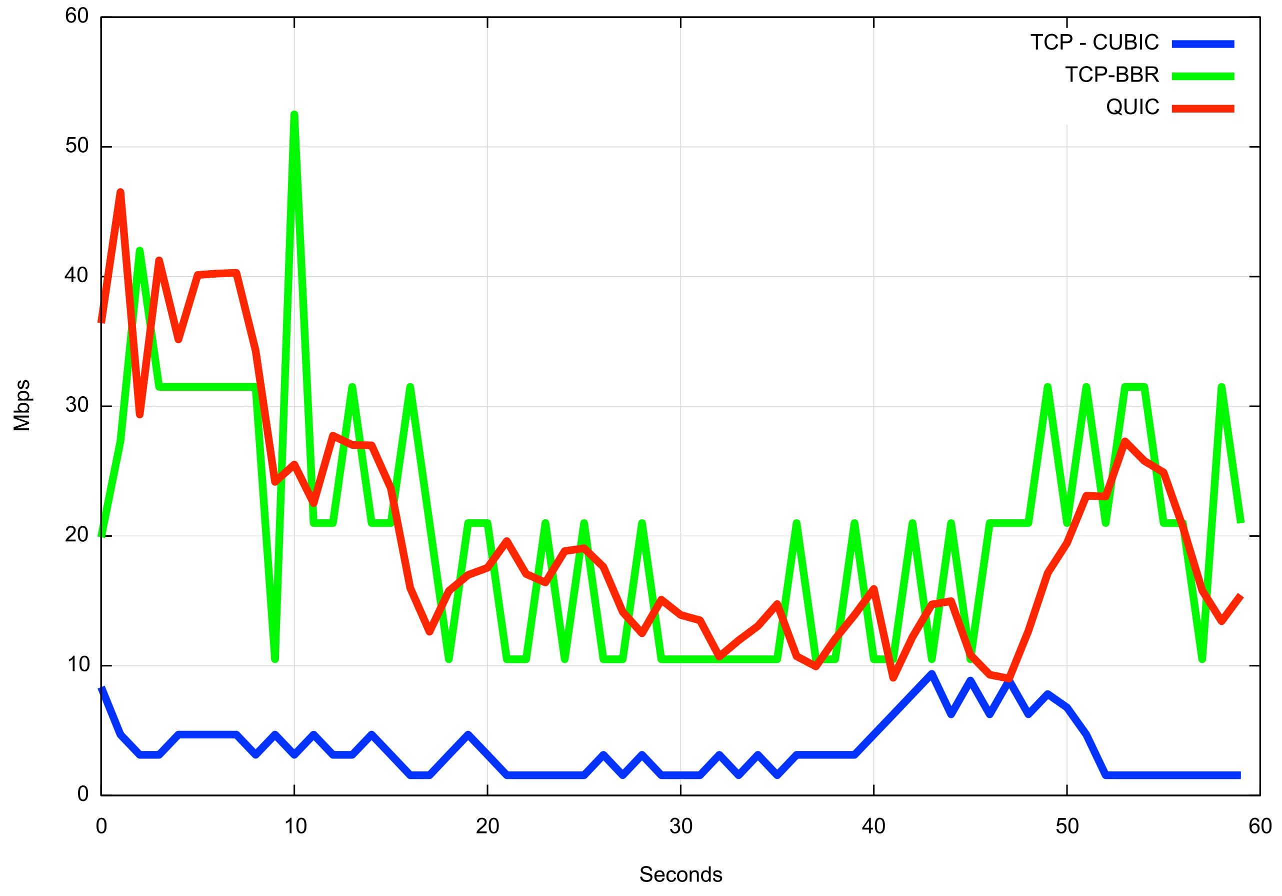
Other Measures: Network Traffic Volume



- ● Other
- ● Youtube
- ● Instagram
- ● Facebook Video
- ● Netflix
- ● Facebook

*source EU Operator 2022

Measuring QUIC Performance



In this test (between the same endpoints) over a Starlink circuit, TCP CUBIC underperforms badly, while TCP BBR and QUIC both perform reasonably well

Why is QUIC important?

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

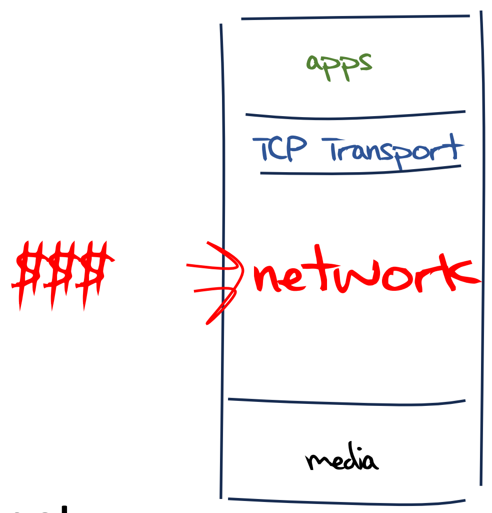
BUT:

- 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?

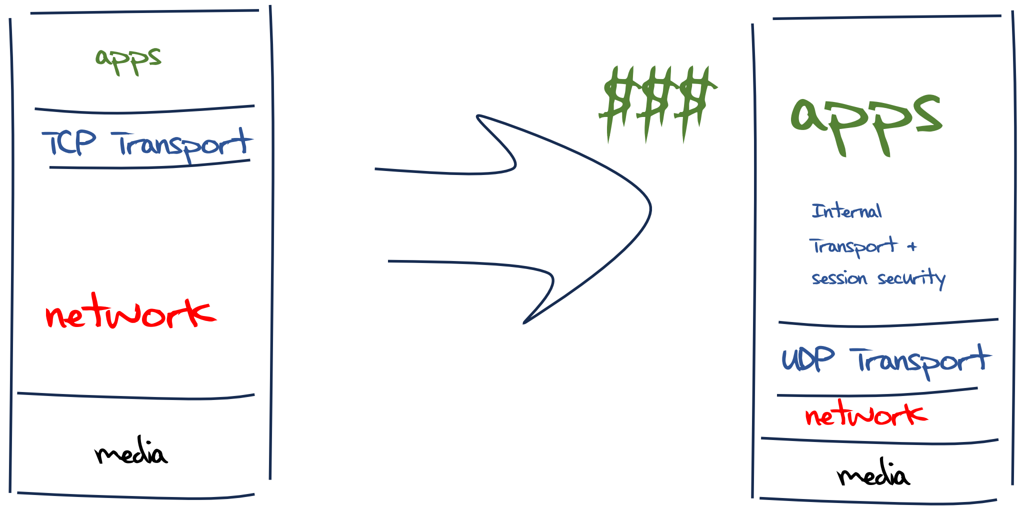
- IP was a ***network*** *protocol* that provided services to attached devices
- The network service model used by IP was minimal
  - Packets may be dropped, fragmented, duplicated, corrupted and/or reordered on their path through the network
  - Its left to the edge systems to recover from this network 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 Network Space

And this is why QUIC is so interesting – it is pushing both carriage and 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!



QUIC and value transform
in the network stack

What does this mean for the Internet?

- The relationship between 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!”

Comments / Questions?