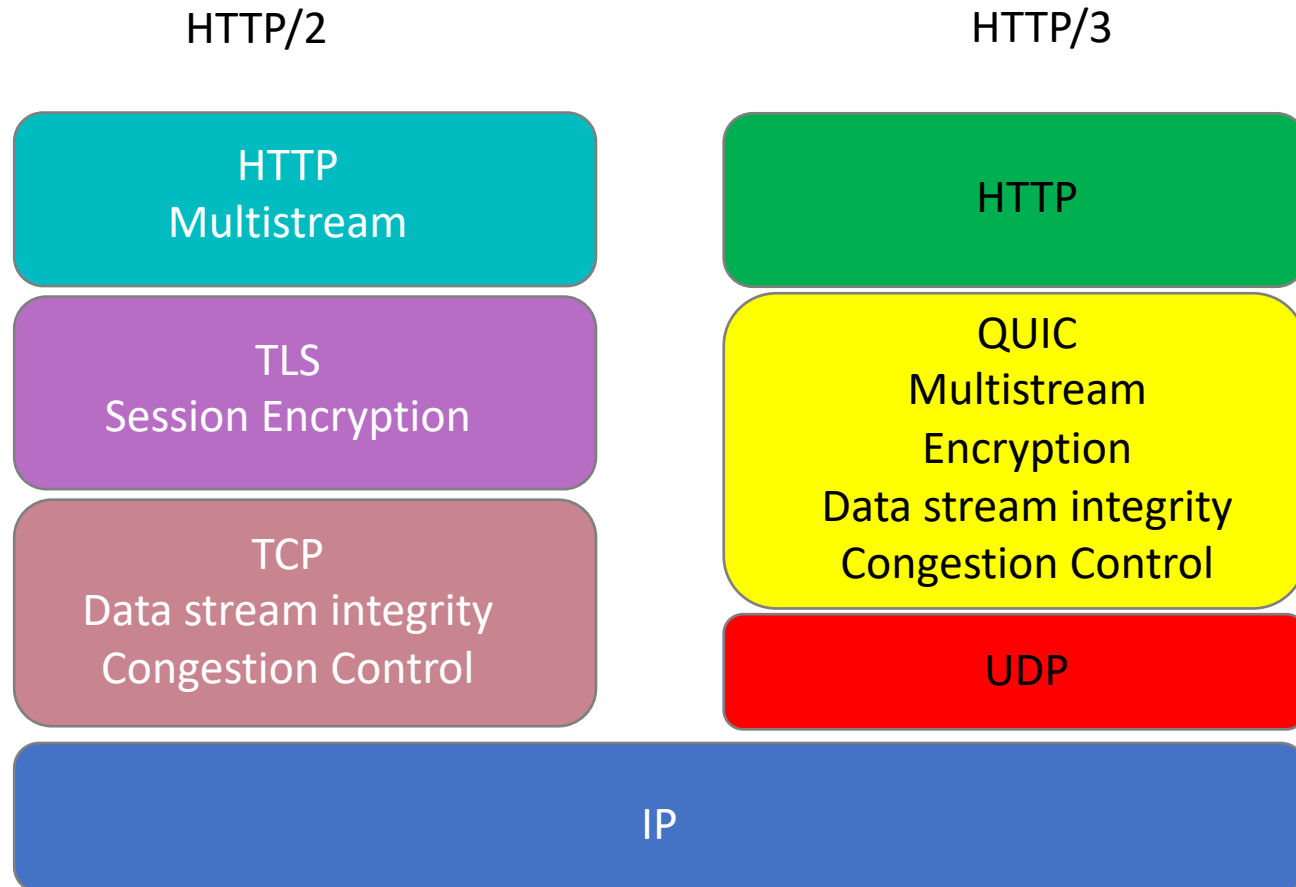


# A Quick look at QUIC

Geoff Huston, Joao Damas,

APNIC Labs

# QUIC is...



# QUIC is...

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) are shifted towards the application

Support for multi-stream multiplexing that avoids head-of-line blocking and exploits a shared congestion and encryption state


QUIC also places the transport control fields inside the encryption envelope, so QUIC has minimal exposure to the network

What TCP needs to be for the Internet of 2022!

# Looking for QUIC

- At APNIC we use Ads to perform large scale measurements of network service capabilities as seen by users
  - IPv6 deployment
  - DNSSEC validation
  - Fragmentation
- Can we use this measurement platform to see the level of use of QUIC in today's network?

# Setting up QUIC

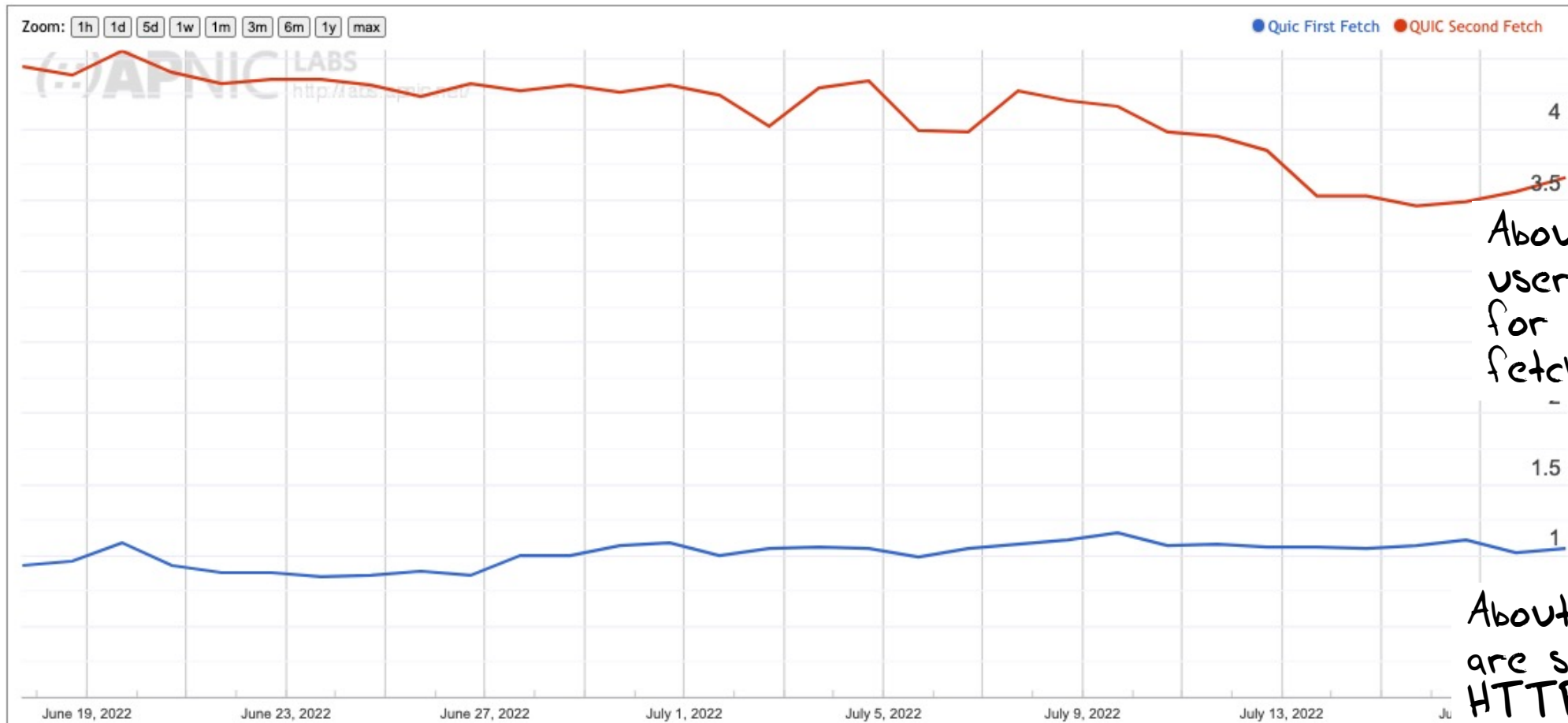
- Server:
  - nginx v1.21.7 with QUIC functions included
- DNS:
  - Set up an HTTPS record for each URL with value: **alpn="h3"**
- Content:
  - **Alt-Svc: h3=" :443"** 

(This second method requires a subsequent query to allow the client to use the Alt-Svc capability. We perform a 2-second delayed second query for this URL in the measurement experiment approximately one fifth of the time. We keep the domain name constant and vary the URL arguments to detect the second fetch.)

# Setting up QUIC

- Server:
    - nginx v1.21.7 with QUIC functions included
  - DNS:
    - Set up an HTTPS record for each URL with value: **alpn="h3"**
  - Content:
    - **Alt-Svc: h3=":443"**
- First Fetch
- Second Fetch

# QUIC Use - June/July 2022



About 3.5÷ of users use HTTP/3 for the second fetch

About 1÷ of users are seen to use HTTP/3 on first fetch

# This result looks wrong!

- Some 90% of the browsers we “see” via the ad campaign identify themselves as Chrome
- 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.



# This result looks wrong!

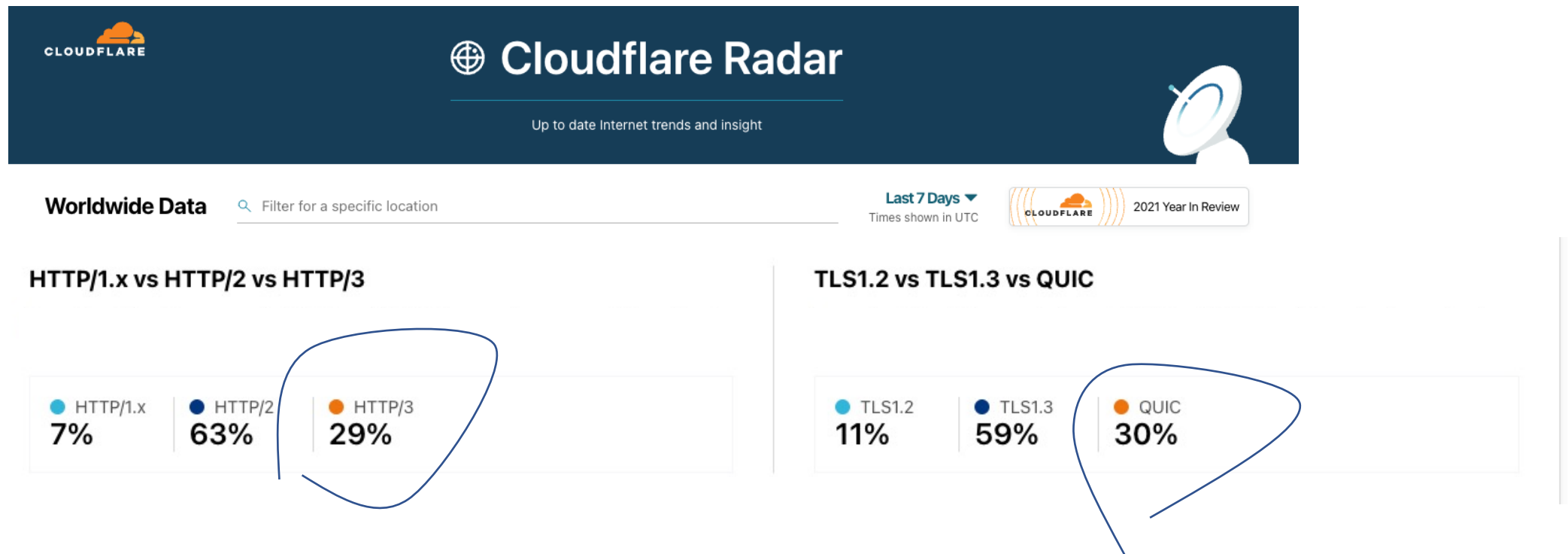
- Some 90% of the browsers we “see” via the ad campaign identify themselves as Chrome
- And Chrome has been supporting a switch to QUIC via the Alt-Svc directive since 2020
- So we should be seeing a far higher level of QUIC use than 3.5%

Hmm

- What do others see?

# Cloudflare's Numbers

Our QUIC use numbers are **far lower** than other published measures



# Maybe 2 fetches is not enough?

- So we changed the experiment to fetch the same URL 7 times with a 2 second pause between each fetch
- Surely this would flush out QUIC use!

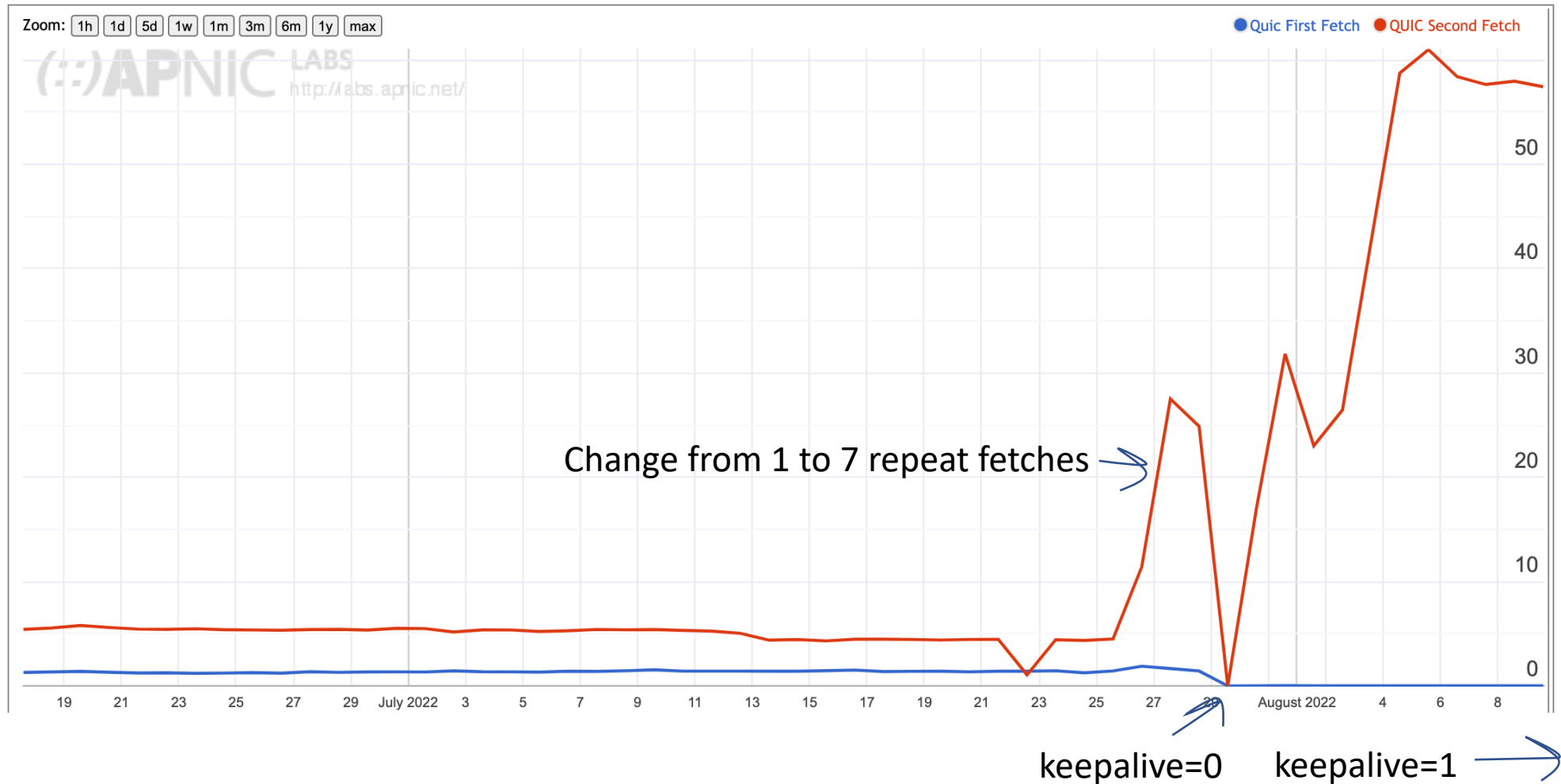
# Nope!

- No change!
- The problem appears to be related to HTTP/2 and persistent connections
- When the browser performs the followup fetch the connection is likely still open and then the browser will prefer to use the open connection over opening up a new QUIC connect
- So on the NGINX server let's set the keepalive session parameter to 0 seconds
- Yes?

# Still Nope!

- That was worse!
- We didn't see any use of QUIC at all
- Nothing. Nada. Not even a little bit.
  
- Seems that if you set keepalive to zero then NGINX disables QUIC completely!
- So we then set the session keepalive parameter to 1 second
- Better?

# QUIC Use - June - August 2022



# Yes!

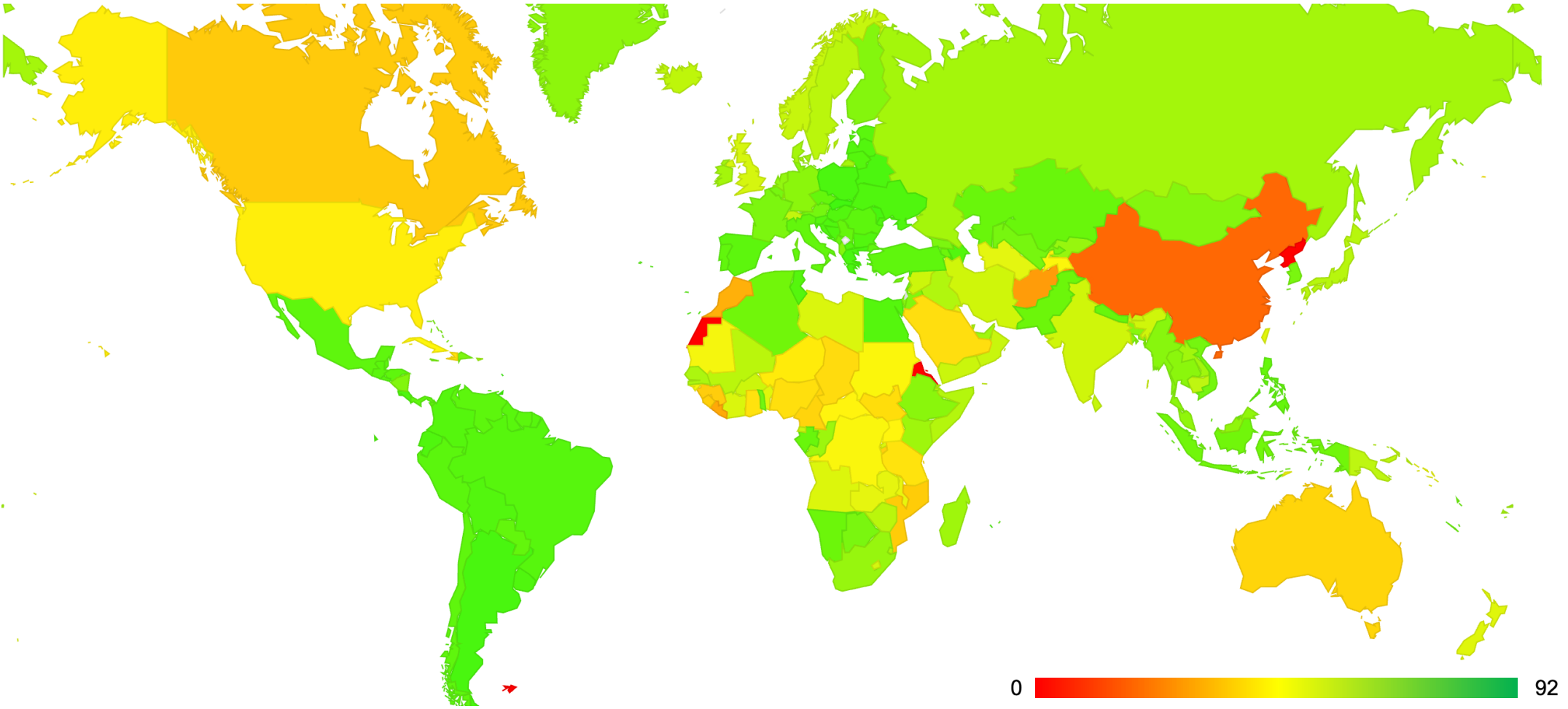
- We are seeing a 57% QUIC on the repeat fetches, corresponding to a rate of 63% QUIC use of Chrome clients

# Yes, and No

- We are seeing a 57% QUIC on the repeat fetches, corresponding to a rate of 63% QUIC use of Chrome clients
- But at the same time the first fetch use dropped from 1% to minimal levels
- Which appears to be an issue with Safari, and iOS (and MAC OS)
- Chrome and Firefox were also behaving erratically across the 7 repeat fetches flipping between HTTP/2 and HTTP/3



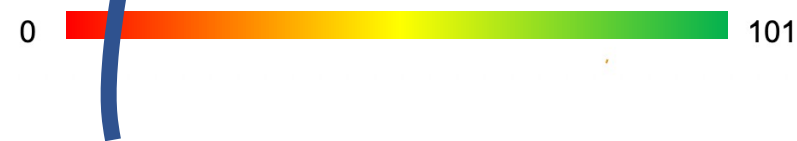
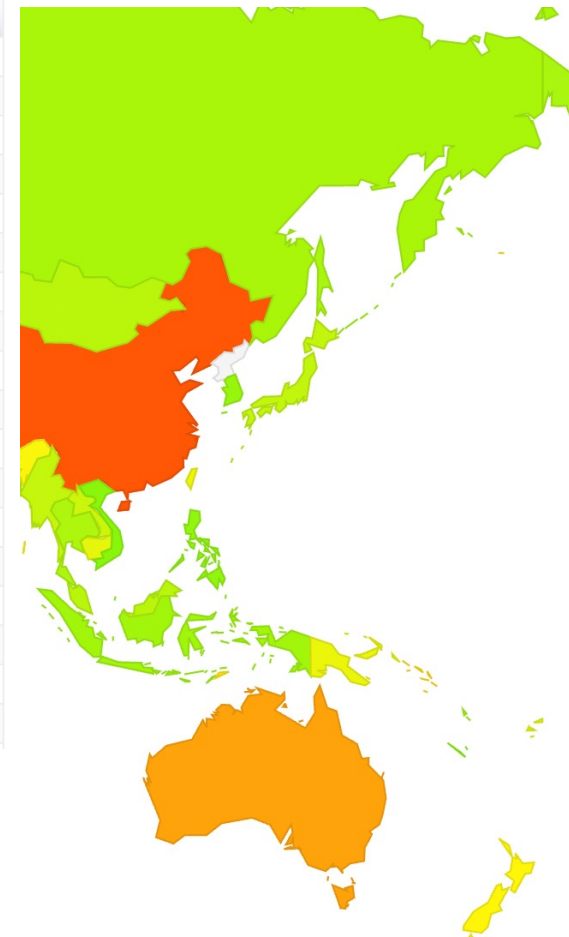
# QUIC Use - August 2022



0  92

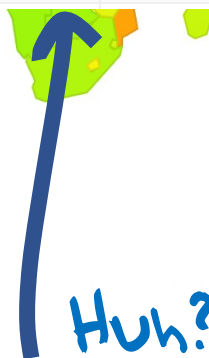
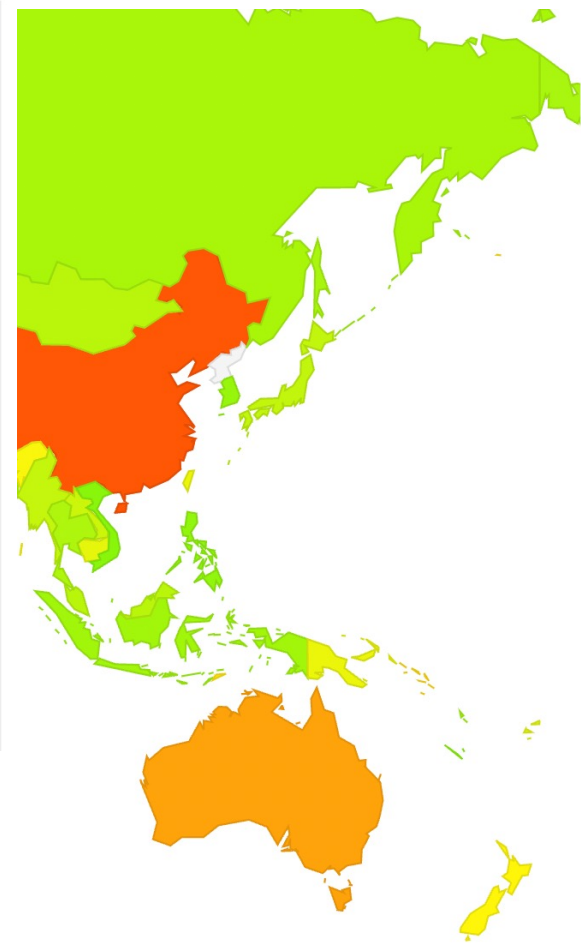
# QUIC Use - August 2022

Code	Region	HTTP/3 on First Query	HTTP/3 on Second Query ▼
EH	Western Sahara, Northern Africa, Africa	0.0%	100.0%
SM	San Marino, Southern Europe, Europe	0.0%	92.2%
MQ	Martinique, Caribbean, Americas	0.0%	88.8%
BL	Saint Barthelemy, Caribbean, Americas	0.0%	87.5%
AX	Aland Islands, Northern Europe, Europe	0.0%	87.1%
MD	Republic of Moldova, Eastern Europe, Europe	0.0%	86.4%
AD	Andorra, Southern Europe, Europe	0.0%	85.9%
SK	Slovakia, Eastern Europe, Europe	0.0%	84.8%
RE	Reunion, Eastern Africa, Africa	0.0%	84.7%
GP	Guadeloupe, Caribbean, Americas	0.0%	84.6%
MF	Saint Martin (French part), Caribbean, Americas	0.0%	84.6%
AR	Argentina, South America, Americas	0.0%	84.3%
TT	Trinidad and Tobago, Caribbean, Americas	0.0%	84.3%
AI	Anguilla, Caribbean, Americas	0.0%	84.2%
UA	Ukraine, Eastern Europe, Europe	0.0%	83.9%
UY	Uruguay, South America, Americas	0.0%	83.9%
EC	Ecuador, South America, Americas	0.0%	83.8%
PL	Poland, Eastern Europe, Europe	0.0%	83.6%



# QUIC Use - August 2022

Code	Region	HTTP/3 on First Query	HTTP/3 on Second Query
EH	Western Sahara, Northern Africa, Africa	0.0%	100.0%
SM	San Marino, Southern Europe, Europe	0.0%	92.2%
MQ	Martinique, Caribbean, Americas	0.0%	88.8%
BL	Saint Barthelemy, Caribbean, Americas	0.0%	87.5%
AX	Aland Islands, Northern Europe, Europe	0.0%	87.1%
MD	Republic of Moldova, Eastern Europe, Europe	0.0%	86.4%
AD	Andorra, Southern Europe, Europe	0.0%	85.9%
SK	Slovakia, Eastern Europe, Europe	0.0%	84.8%
RE	Reunion, Eastern Africa, Africa	0.0%	84.7%
GP	Guadeloupe, Caribbean, Americas	0.0%	84.6%
MF	Saint Martin (French part), Caribbean, Americas	0.0%	84.6%
AR	Argentina, South America, Americas	0.0%	84.3%
TT	Trinidad and Tobago, Caribbean, Americas	0.0%	84.3%
AI	Anguilla, Caribbean, Americas	0.0%	84.2%
UA	Ukraine, Eastern Europe, Europe	0.0%	83.9%
UY	Uruguay, South America, Americas	0.0%	83.9%
EC	Ecuador, South America, Americas	0.0%	83.8%
PL	Poland, Eastern Europe, Europe	0.0%	83.6%



# Yes, and No

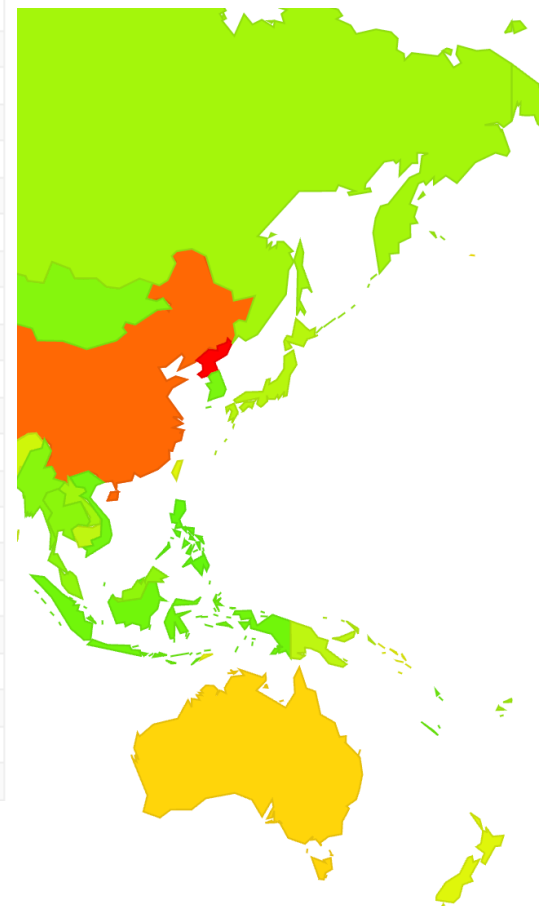
- We are seeing a 57% QUIC on the repeat fetches, corresponding to a rate of 63% QUIC use of Chrome clients
- But at the same time the first fetch use dropped from 1% to minimal levels
- Which appears to be an issue with Safari, and iOS (and MAC OS)
- Interestingly, Safari technology preview (release 150, safari 16.0, webkit 17614.1.22.1.1) does not show this behaviour within our test rig
- So we were unsure what is going on here between Safari clients and our NGINX-based QUIC server

# Its all about Keepalive Timers

- After much searching under many rocks we are advised (**many** thanks to Ryan Hamilton, Martin Thompson and Tommy Pauly for various clues at this stage) that a server keepalive timer value of 1 is also a Really Bad setting!
- The server is dropping the QUIC connection too aggressively and the browser client drops back to HTTP/2
- The default value of 65 seconds for the server keepalive interval seems to be too long
- And 1 second is too short
- So now let's try a value of 20 seconds...

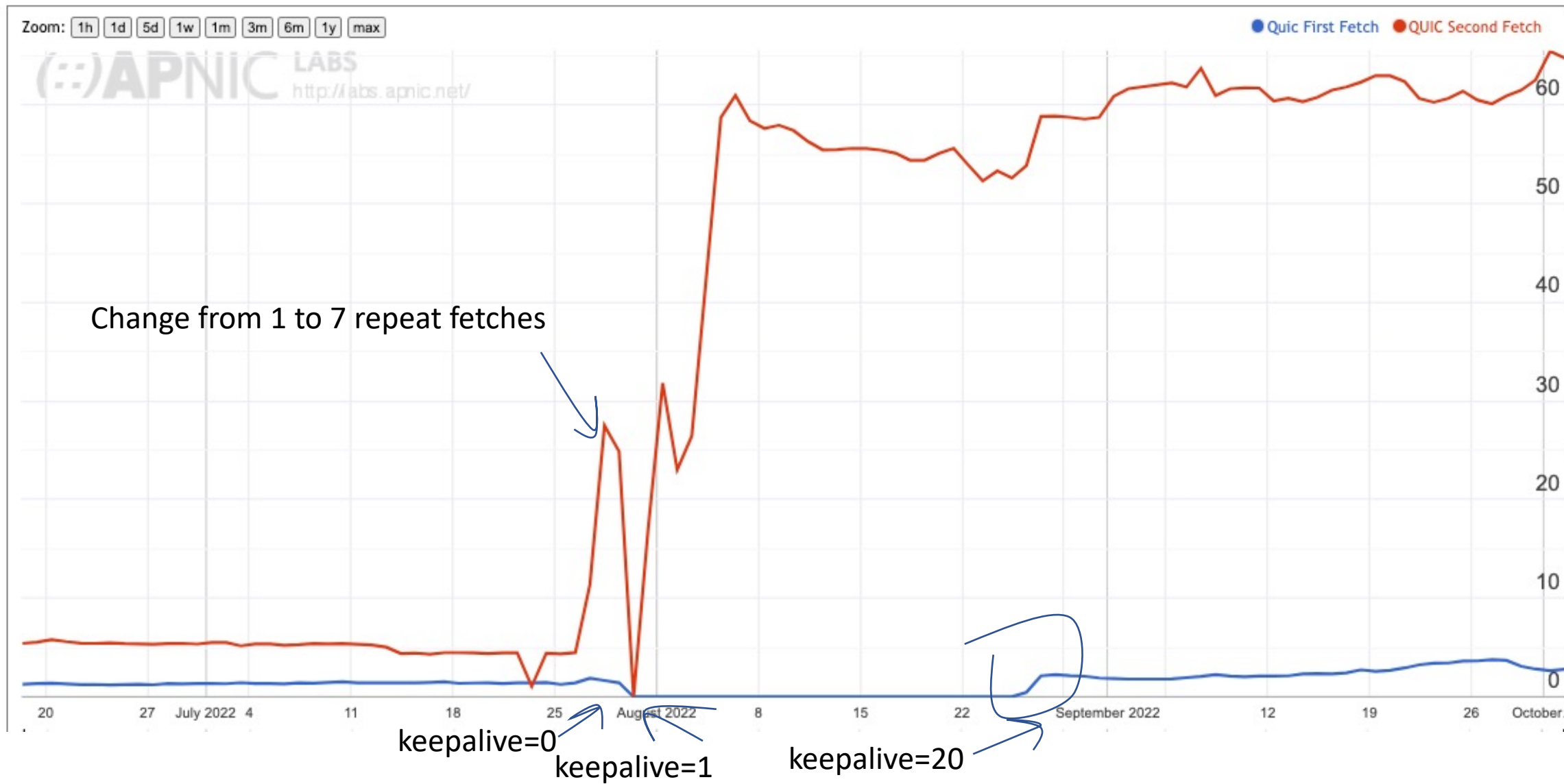
# Quic Use - September 2022

Code	Region	HTTP/3 on First Query	HTTP/3 on Second Query
IM	Isle of Man, Northern Europe, Europe	16.5%	45.0%
GB	United Kingdom of Great Britain and Northern Ireland, Northern Europe, Europe	16.0%	57.1%
JE	Jersey, Northern Europe, Europe	13.6%	55.8%
OM	Oman, Western Asia, Asia	13.5%	65.1%
AX	Aland Islands, Northern Europe, Europe	11.7%	55.6%
GG	Guernsey, Northern Europe, Europe	10.7%	46.7%
SE	Sweden, Northern Europe, Europe	10.7%	61.2%
MO	Macao Special Administrative Region of China, Eastern Asia, Asia	10.5%	66.7%
CH	Switzerland, Western Europe, Europe	9.8%	68.3%
KW	Kuwait, Western Asia, Asia	9.7%	61.1%
NO	Norway, Northern Europe, Europe	9.5%	64.7%
US	United States of America, Northern America, Americas	9.4%	48.8%
TW	Taiwan, Eastern Asia, Asia	9.2%	61.6%
AU	Australia, Australia and New Zealand, Oceania	9.1%	53.4%
BH	Bahrain, Western Asia, Asia	8.9%	65.5%
FO	Faeroe Islands, Northern Europe, Europe	8.7%	64.3%
JM	Jamaica, Caribbean, Americas	8.4%	69.5%
GI	Gibraltar, Southern Europe, Europe	8.3%	61.2%
MC	Monaco, Western Europe, Europe	8.2%	63.1%
QA	Qatar, Western Asia, Asia	8.0%	64.9%
DK	Denmark, Northern Europe, Europe	7.7%	70.1%
GU	Guam, Micronesia, Oceania	7.5%	66.1%





# Quic Use - September 2022



# Some Questions:

1. Which clients are performing QUIC and why?
2. What are the QUIC MSS values?
3. What is the QUIC connection failure rate?
4. Is QUIC faster than HTTP/2 + TLS?



# 1. OS Clients\* performing QUIC

	TCP/TLS	QUIC on First Fetch	QUIC on Second Fetch
iOS	5.5%	↑ <b>93.3%</b>	16.1%
Mac OS	1.0%	2.8%	0.6%
Android	84.5%	1.7%	↑ <b>77.9%</b>
Win	5.5%	1.4%	4.3%
Linux	0.4%	0.2%	0.2%
Others	3.1%	0.6%	0.9%
	100.0%	100.0%	100.0%

\* Based on reported browser string

# 1. Browser Clients\* performing QUIC

	TCP/TLS	QUIC on First Fetch	QUIC on Second Fetch
Chrome	91.8%	4.1%	↑ <b>81.7%</b>
Safari	4.3%	↑ <b>93.3%</b>	↑ <b>16.1%</b>
Firefox	0.8%	2.4%	1.0%
Edge	0.7%	0.0%	0.5%
Opera	0.2%	0.1%	0.6%
Others	2.2%	0.1%	0.1%
	100.0%	100.0%	100.0%

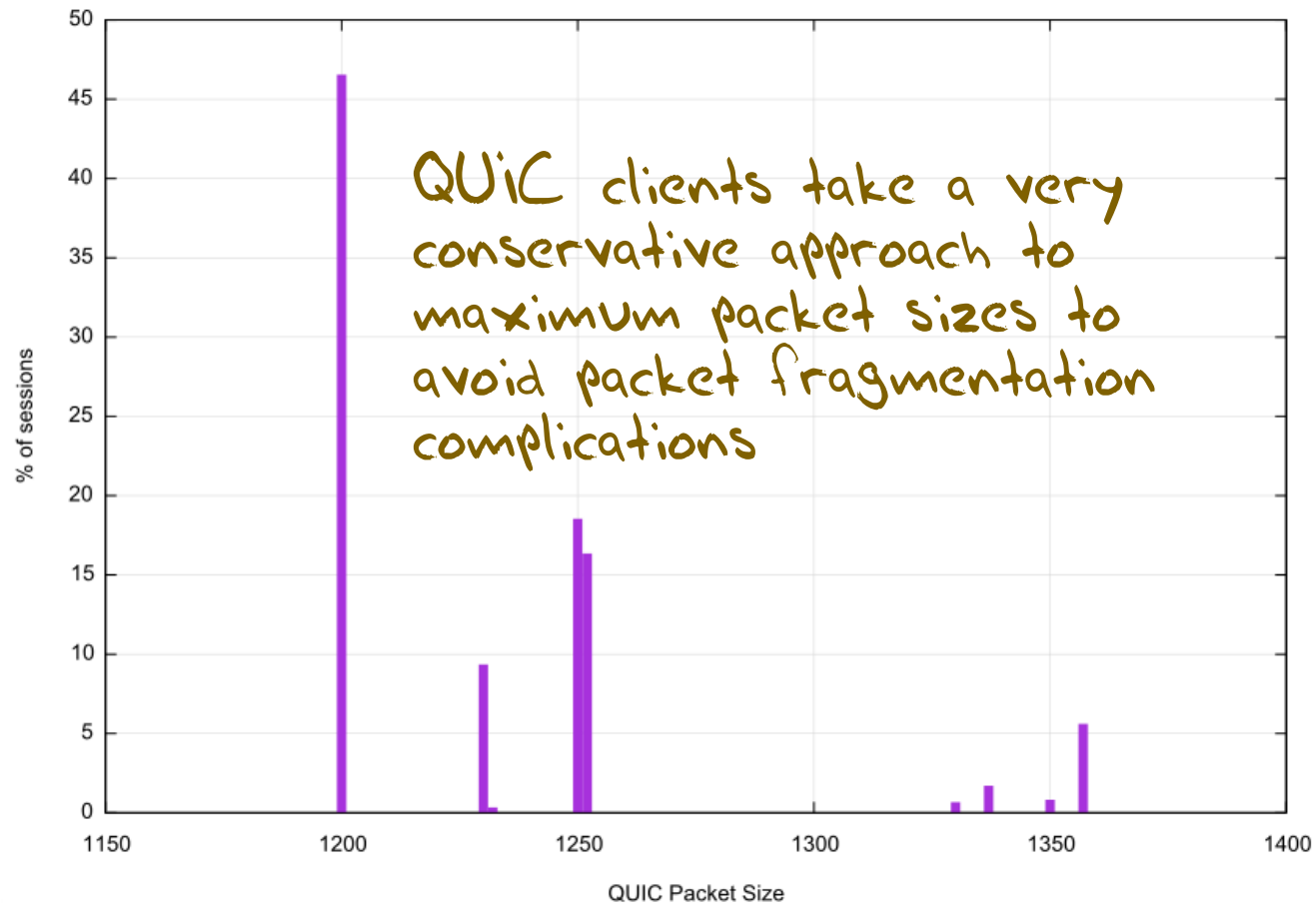
\* Based on reported browser string

# 1. Who does QUIC and why?

Apple Safari clients use a **DNS HTTPS** query and *some* of these clients then follow up with a fetch over QUIC. The observed DNS HTTPS query to QUIC fetch conversion rate was relatively small.

Chrome clients use the **Alt-Svc** field as a QUIC trigger for *most* clients. The observed QUIC conversion rate was high, but not universal.

## 2. QUIC Packet Size distribution



Maximum Packet Sizes used in QUIC sessions:

1,200 octets – 46.6%

1,250 octets – 18.5%

1,252 octets – 16.4%

# 3. QUIC Connection Loss

In this measurement framework we cannot measure client -> server loss, but we can measure server-> client loss by looking for incomplete QUIC initial connections that do not complete

(this form of connection loss could be due to the client filtering incoming UDP port 443 packets)

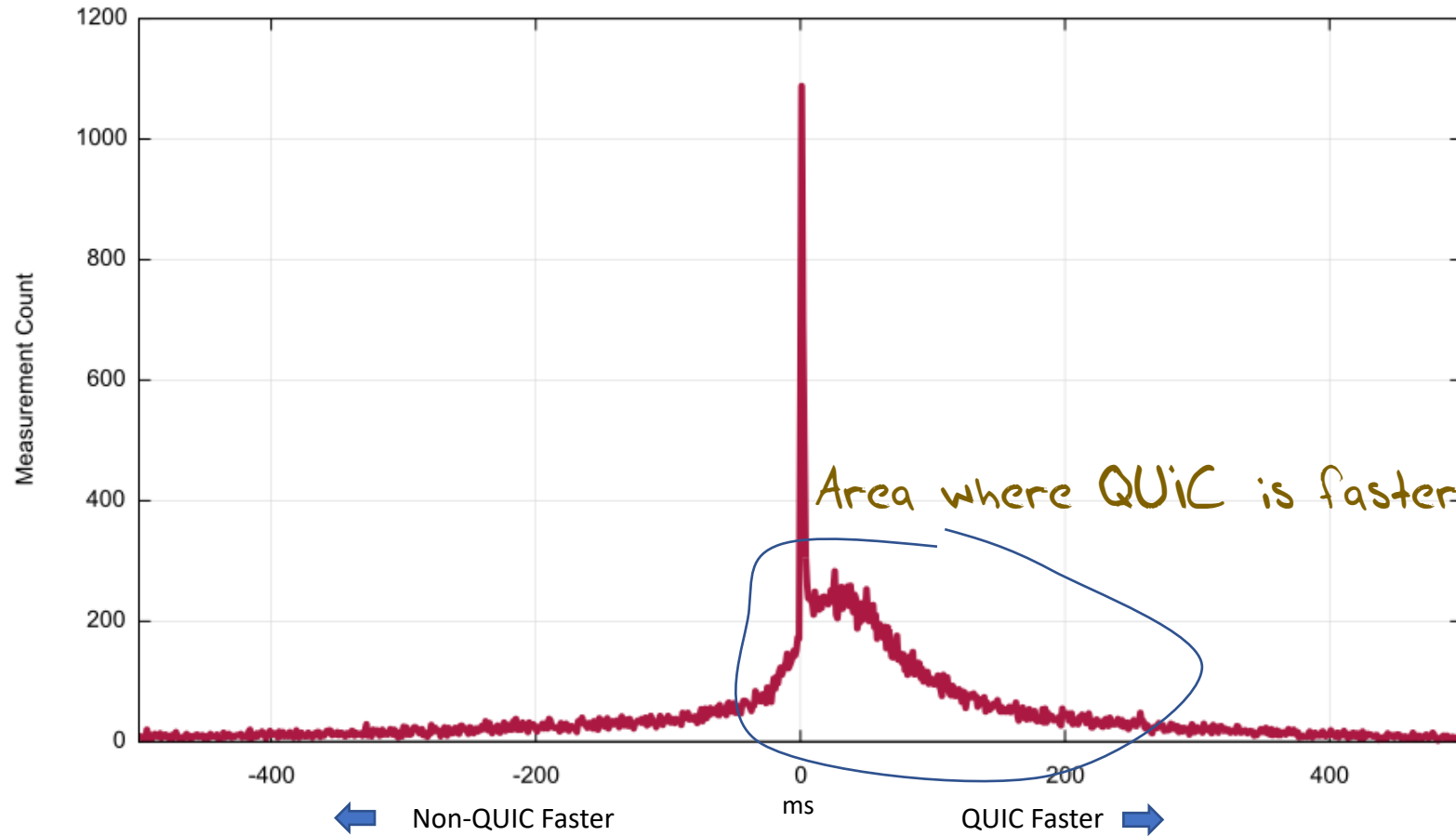
Initial QUIC Connections:	19,211,357
Failed Connections:	46,645
Failure Rate:	<b>0.24%</b>

## 4. Is QUIC Faster?

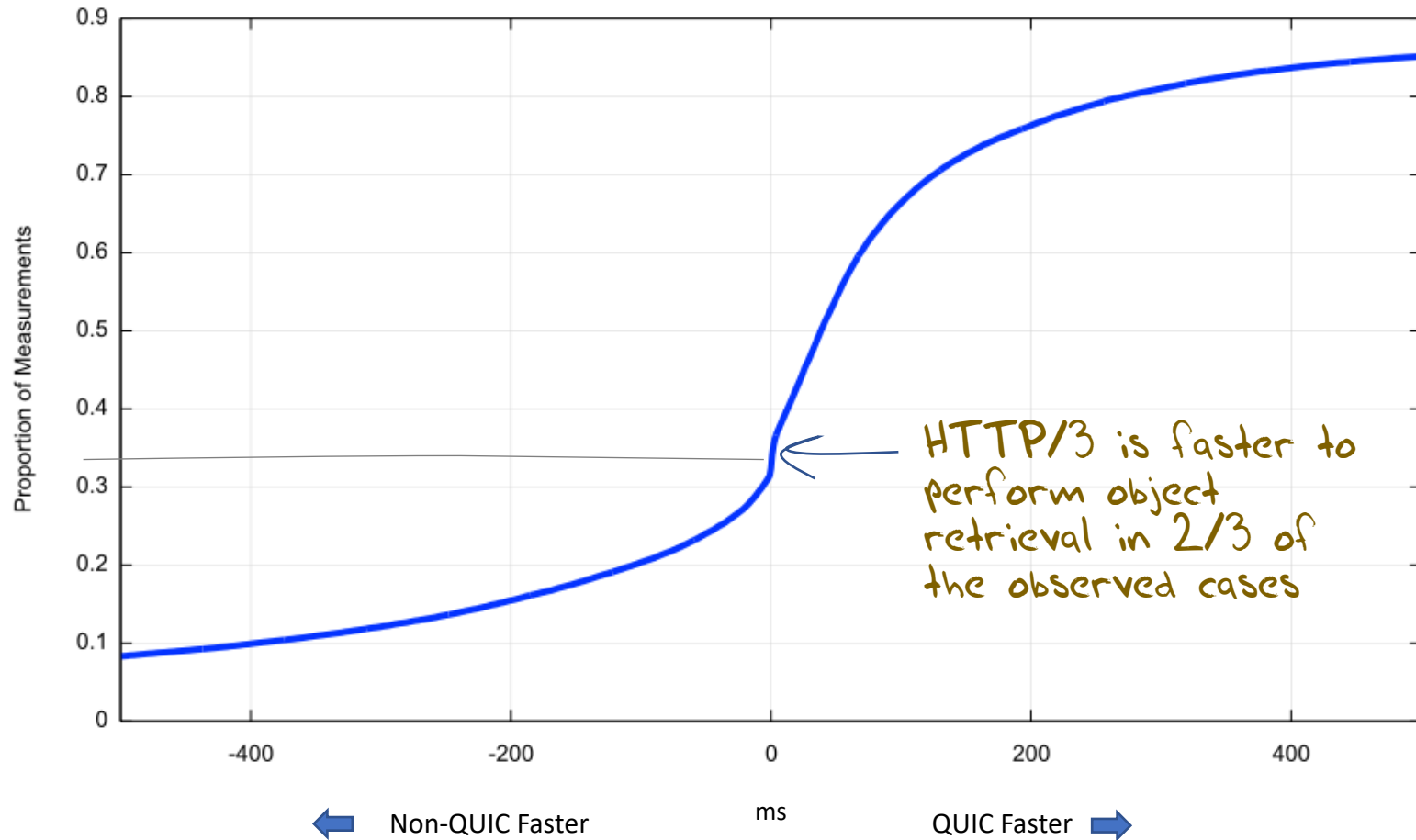
Let's compare the user-measured time to load an object using HTTP/2 and the same user's measured time to load the same object using HTTP/3

- There are a number of variables in the user time measurement, including varying time penalties relating to the internal task scheduling within the browser, but these individual factors should be cancelled out over a large enough sample set

# 4. TCP/TLS vs QUIC speed difference



# 4. Cumulative Distribution





# Some Answers:

1. Which clients are performing QUIC?

**The recent change appears to relate predominately to iOS 15.x clients (iPhones and iPads) using HTTPS queries and selectively performing an object retrieval over HTTP/3 at a rate of approximately 1 in 5**

2. What are the QUIC MSS values?

**Most QUIC clients limit their total IP packet size to a max of either 1,250 or 1,252 octets. Largest observed packet was 1,357 octets**

3. What is the QUIC connection failure rate?

**Extremely small at 0.24%. This falls within the bounds of experimental error in this experiment's framework.**

4. Is QUIC faster than HTTP/2 + TLS?

**Yes, more than 2/3 of the time QUIC will complete in less elapsed time than the equivalent HTTP/2 retrieval**

Thanks!

**Ongoing HTTP/3 Measurement Report at APNIC Labs:  
<https://stats.labs.apnic.net/quic>**