July 2025
Geoff Huston

Triggering QUIC

Last month, in June 2025, I reported on our progress with the adoption of the QUIC transport protocol. Here I would like to look at the mechanisms used to trigger a client application (typically a browser) to connect to the server using the QUIC transport protocol.

A conventional way to introduce a new transport service is to let applications initiate a connection using the new service. If the connection is unresponsive, then the application can back off to use the old transport service. In the case of QUIC, this would be a QUIC connection request made via a UDP packet directed to the server's port 443, and if the client doesn't receive a response after some timeout interval it will fall back to TCP (and TLS) by sending a SYN packet to the server's TCP destination port 443.

This connection process can be slow, particularly if the server or the intervening network does not support QUIC transport. As the client isn't necessarily aware of the round-trip time to the server, the client would normally err on the side of caution and wait at this juncture for at least one or two seconds before abandoning the connection attempt. QUIC was designed to increase the responsiveness of services over the network, not to introduce new sources of delay. A client could could use a smaller timeout interval, but this would be prone to false signals, where the client falls back to TCP unnecessarily. A client could set up a "connection race", sending a QUIC connection packet and a TCP SYN packet, and follow through by completing the connection with the first protocol to respond, but this would impose additional load on the server as it is forced to set up two connection contexts for every client connection.

One way to establish if a QUIC connection is viable without paying a time penalty is for the server to signal the capability to use QUIC to the client in the first (TCP/TLS) connection, allowing the client to initiate a QUIC session on the second and subsequent connections. This *second-use* approach is defined in an Internet standard for *Alternative Services* (RFC 7838), which is a means to list alternative ways to access the same resources, using the HTTP header fields.

For example,

```
alt-svc: quic=":443"; ma=2592000;
```

This indicates that the same material is accessible using QUIC over port 443. The "ma" field is the time to keep this information cached in the local client (in seconds), which in this case is 30 days. The semantics of this header is that this is not an unconditional directive, and the client is free to ignore this information and continue with the existing TCP/TLS protocol choice.

However, it's not quite as simple as it may sound. The HTTP protocol introduced the concept of session persistence in HTTP/1.0 so that the underlying transport session is kept open for some *keepalive interval*, allowing reuse of the TCP and TLS state without the overheads of establishing a new session. The implication for the *second-use* signaling approach was that the original HTTP session was kept open across multiple requests, indefinitely deferring any subsequent connection that would use the QUIC transport protocol.

This form of triggering the use of QUIC has been bundled into the Chrome browser for more than a decade.

When Apple introduced support for QUIC in its Safari browser they used a different trigger, namely the DNS. The IETF has introduced the SVCB record in the DNS, allowing a number of connection profile items to be stashed in the DNS (RFC 9460). SVCB records facilitate the discovery of alternative endpoints for a service, allowing clients to choose the most suitable endpoint based on their needs and capabilities. There is the HTTPS DNS record, which is the SVCB record for use for HTTPS service connections. Relevant here is the Application-Layer Protocol Negotiation (ALPN) field where the token **`h3`** denotes a server capability to support QUIC (HTTP/3) (RFC 9114).

Safari browsers that support the use of QUIC will query the DNS for an HTTPS record, and if there is an **`alpn`** field that specifies **`h3`** in this record, then the browser can immediately proceed with opening a QUIC connection to the server, based on the knowledge that the server has indicated that it can support QUIC connections.

This form of triggering the use of QUIC has been bundled into the Safari browser for more than a year.

This means that we have two triggers for the use of QUIC, one as a "second use" in-band alternative service directive imbedded in the HTTPS protocol exchange, predominately used by the Chrome browser, and the DNS HTTPS trigger, predominately used by the Safari browser.

Measuring QUIC

The conventional ways to measure the uptake of a technology such as QUIC is to set up a service that supports QUIC in additional to conventional TLS over TCP and count the relative use of each transport protocol to access the service.

If we had access to a well-used service platform that supported both transport protocols, we could certainly perform this measurement. This is what Cloudflare appears to have done, and the results are visible on the Cloudflare Radar report. A screenshot of the relative use of QUIC (HTTP/3) in the overall traffic mix seen by Cloudflare is shown in Figure 1.

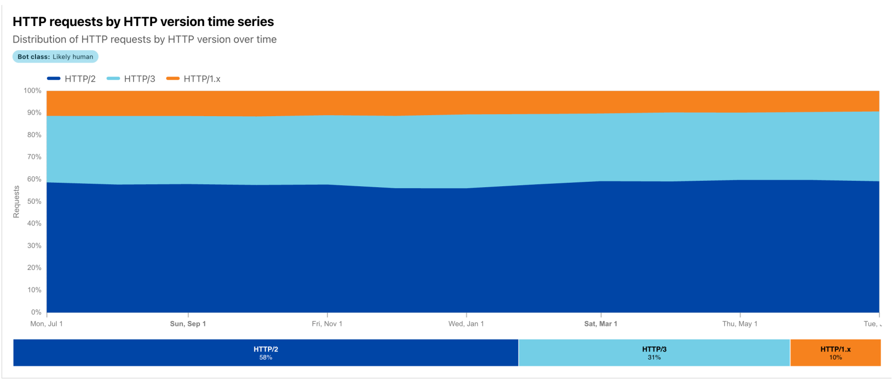


Figure 1 – Use of QUIC as reported by Cloudflare Radar

Some 30% of the HTTP requests seen by Cloudflare use the QUIC protocol. But what does this mean? Does all content served by Cloudflare use the **alt-svc** directive to indicate that the server is capable of serving content via QUIC? Do they also use an HTTPS DNS record for all of their content? Or for just some content? Or none? Without a basic understanding as to how the use of QUIC is being triggered by Cloudflare, this 31% result is less helpful in terms of understanding the overall picture of QUIC deployment.

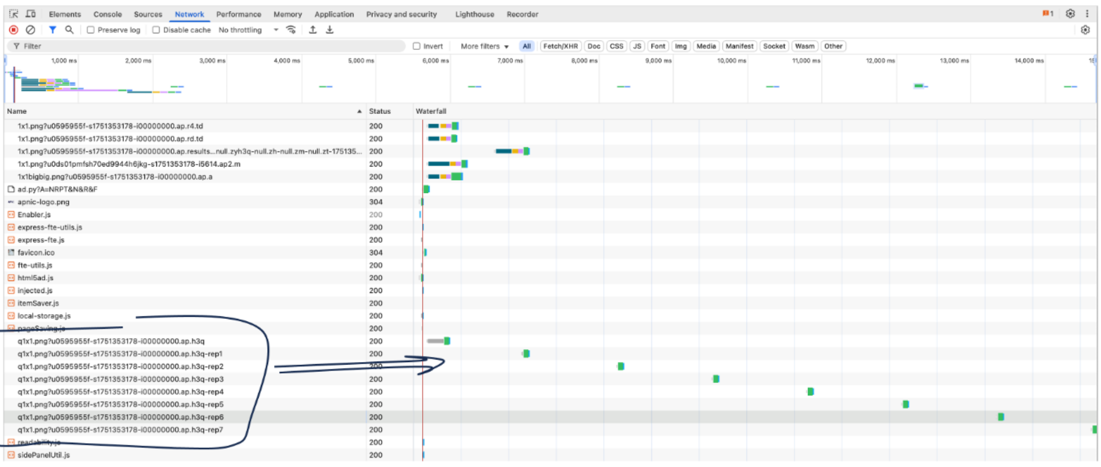
This is another classic example of a producer/consumer relationship in terms of adoption of a technology. A content server sees little in the way of marginal benefit in supporting delivery of content over QUIC if few clients support this transport protocol, and clients see little in the way of marginal benefit of adoption of QUC if there is scant content that can be accessed in this manner.

Without some form of initial impetus such a situation can lead to a situation of stasis through mutual deadlock, with clients and servers awaiting a move by the other party. This is by no means uncommon within the Internet, and we've seen a similar standoff in the adoption of DNSSEC, with adoption of DNSSEC signing and DNSSEC validation each dependant on the other. Similarly, the adoption of IPv6 has been through a similar journey, with IPv6 support in clients and networks awaiting support of IPv6 delivery of content and services, while the content and service delivery side was awaiting the deployment of IPv6 support in client systems.

Successful self-propelled adoption appear to require that both clients and servers generate marginal benefits for themselves through the adoption of the technology.

What we do know is that the Chrome browser has been supporting the QUIC protocol for at least five years now (see the October 2020 Chrome announcement). Chrome appears to enjoy a market share in the context of the public Internet of some 65% of the user base (source: statscounter). It's not clear to me how much content served by Cloudflare uses this **alt-svc** directive, and perhaps equally importantly it's unclear as to how users interact with these services, and in particular how they formulate repeated requests to the same named content server. Both HTTP/1.0 and HTTP/2 allow for session persistence, so if a client performs a bunched cluster of requests to the same server it's likely that all the requests will be handled within the same underlying initial TCP session. With **alt-svc**, it's the second and subsequent connection that can use the QUIC protocol.

If the aim here is to measure the *potential* to use QUIC with the Chrome browser, then the measurement technique needs to be carefully structured to trigger session termination and the initiation of a new session, so that the client will then make use **alt-svc** directive. With the QUIC measurement conducted by APNIC Labs we use a sequence of 7 repetitions of the query, spaced in two second intervals, with a one second keepalive timer. The browser diagnostic screen showing the spacing of these repeated queries is shown in Figure 2.



Name	Status
1x1.png?u0595955f-s1751353178-i00000000.ap.r4.td	200
1x1.png?u0595955f-s1751353178-i00000000.ap.rd.td	200
1x1.png?u0595955f-s1751353178-i00000000.ap.results...null.zyh3q-null.zh-null.zm-null.zt-175135...	200
1x1.png?u0ds01pmfsh70ed9944h6jkg-s1751353178-i5614.ap2.m	200
1x1bigbig.png?u0595955f-s1751353178-i00000000.ap.a	200
ad.py?A=NRPT&N&R&F	200
apnic-logo.png	304
Enabler.js	200
express-fte-utils.js	200
express-fte.js	200
favicon.ico	304
fte-utils.js	200
html5ad.js	200
injected.js	200
itemSaver.js	200
local-storage.js	200
pageSaving.js	200
q1x1.png?u0595955f-s1751353178-i00000000.ap.h3q	200
q1x1.png?u0595955f-s1751353178-i00000000.ap.h3q-rep1	200
q1x1.png?u0595955f-s1751353178-i00000000.ap.h3q-rep2	200
q1x1.png?u0595955f-s1751353178-i00000000.ap.h3q-rep3	200
q1x1.png?u0595955f-s1751353178-i00000000.ap.h3q-rep4	200
q1x1.png?u0595955f-s1751353178-i00000000.ap.h3q-rep5	200
q1x1.png?u0595955f-s1751353178-i00000000.ap.h3q-rep6	200
q1x1.png?u0595955f-s1751353178-i00000000.ap.h3q-rep7	200
readability.js	200
sidePanelUtil.js	200

Figure 2 – HTTP request repetition to trigger the use of the ***alt-svc*** *directive*

What we anticipate here is that the initial request will be served using TCP, and the subsequent requests will be served using QUIC.

In addition, the use of QUIC may be triggered using a DNS HTTPS resource record, with the value `alpn="h3"`. Queries for HTTPS records are now running at 10% of all queries seen in the APNIC measurement, with A queries at 45% and AAAA queries at 16%.

The overall picture for *potential* QUIC use in the public Internet, as measured by APNIC Labs, is shown in Figure 3. The blue line in this figure is the proportion of observed clients who use QUIC on the first retrieval. As we've already noted, this initial retrieval is likely to be using the DNS HTTPS query to trigger the use of QUIC. The subsequent retrievals using QUIC where the initial retrieval was via HTTP/1,2 is likely to have been triggered using the **alt-svc** directive.

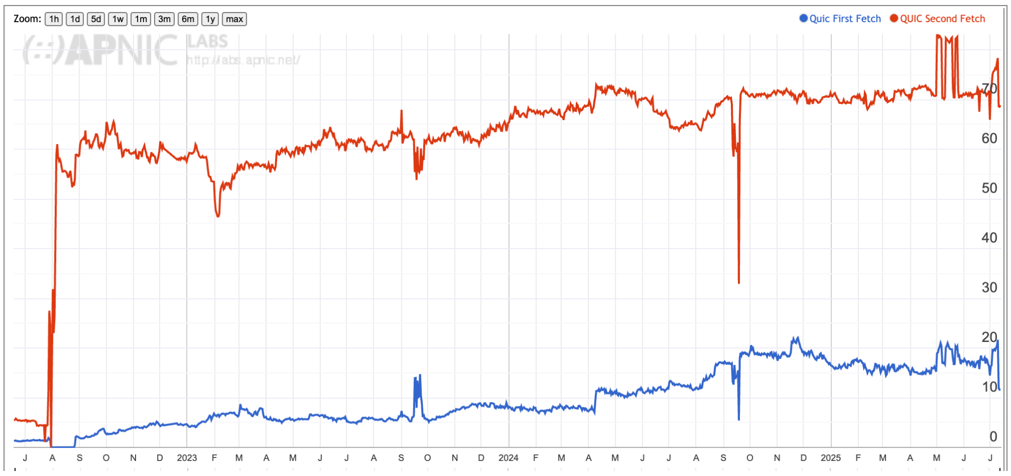


Figure 3 – Time series in Measuring QUIC

Can we associate these trigger mechanisms with the browsers? By using the User Agent string in the observed web fetches, we can associate particular behaviours with browser types.

Who is Using the HTTPS Record?

Figure 4 shows the time series of the use of the HTTPS query on a daily basis for all browser types (blue line), and a further breakdown into the relative use by Safari browsers, Chrome and others.

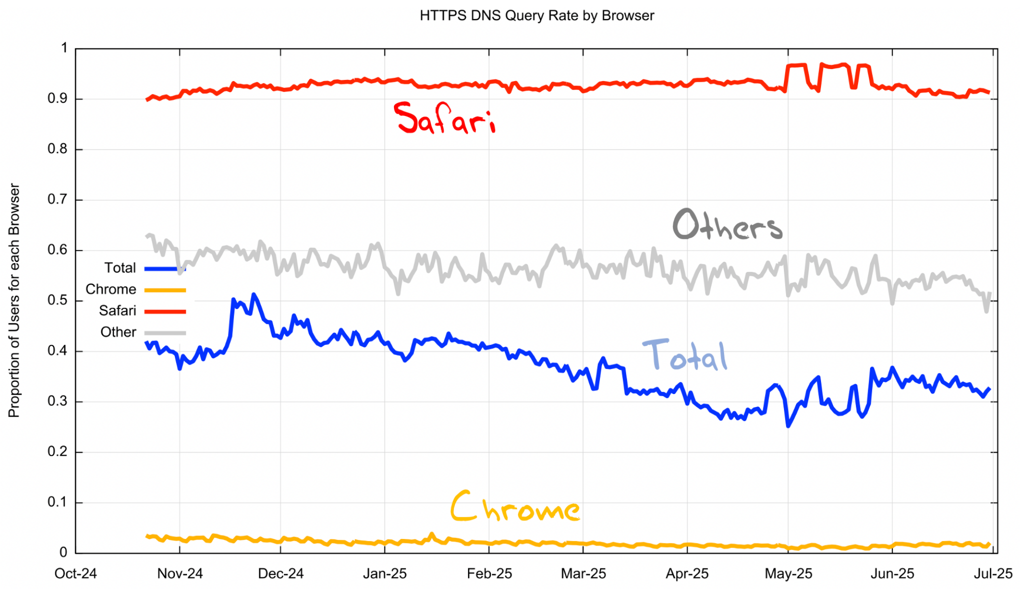


Figure 4 – Relative use of the HTTPS query by browser type

This figure shows that total some 30% - 40% of sampled clients generate an HTTPS query (blue), where 90% of Safari browsers generate an HTTPS query, and 0.3% of Chrome browsers generate an HTTPS query. The mix of other browser types has a 60% relative use of the HTTPS query.

Let's look at Safari behaviour in a little more detail, looking at the "conversion rate" from the HTTPS query to a fetch using the QUIC protocol. Figure 5 shows the relative proportion of Safari browsers who make an HTTPS query. Over the past 7 months this has been a relatively steady 90% of browsers making this DNS query. Figure 5 also shows the rate of these browsers performing the subsequent web object fetch using QUIC. Over the same period this QUIC use rate is between 50% to 60% of Safari browsers.

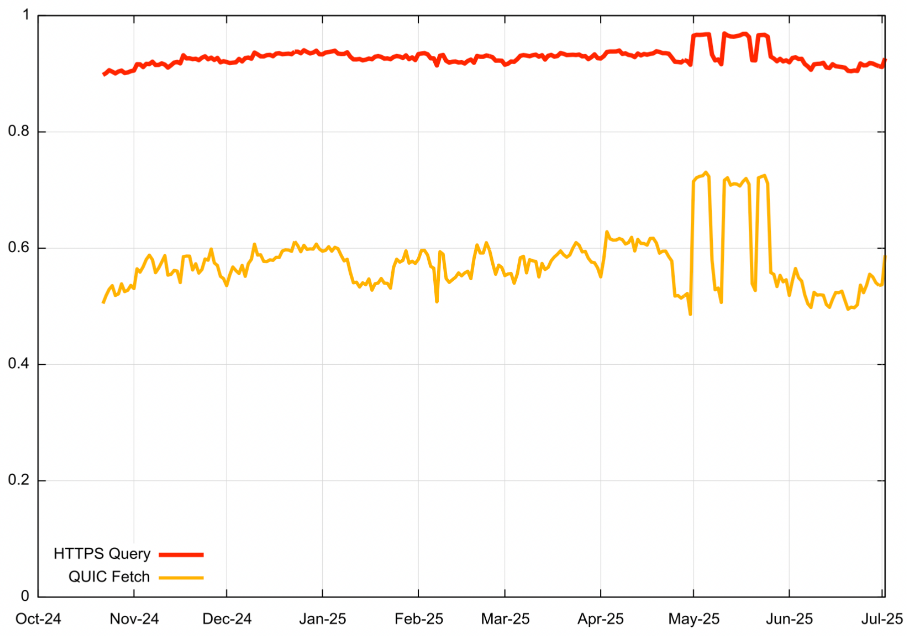


Figure 5 – Relative use of the HTTPS query and QUIC fetch for Safari Browsers

In asking why this is happening, one theory is that the DNS infrastructure is blocking HTTPS query type responses. Given that Safari browsers ask for A, AAAA (when IPv6 is available) and HTTPS records, the absence of an HTTPS response would not be noted as unusual. The A and AAAA responses would guide Safari to use either HTTP/1.0 or HTTP/2 to perform the web object retrieval.

To test this theory, we ran a modified experiment where the DNS name only had an HTTPS resource record. No A and no AAAA. To provide the client with the address of the web object we used the **ipv4hint** and **ipv6hint** attributes of the HTTPS record:

```
test_name   IN  HTTPS   1  .  alpn="h2,h3" ipv4hint=192.0.2.1 ipv6hint=2001:db8::1
```

The result of this experiment, run over a 24-hour period, is shown in Table 1:

	HTTPS and alt-svc										
	All			Chrome			Safari			Other	
DNS query (any)	13,177,108			9,487,295			3,602,160			87,653	
DNS Query HTTPS	3,701,867	28.1%		155,926	1.6%		3,491,146	96.9%		54,795	62.5%
QUIC First	2,837,425	21.5%		111,679	1.2%		2,693,192	74.8%		32,554	37.1%
HTTP/1,2 First	10,243,793	77.7%		9,293,260	98.0%		896,028	24.9%		54,505	62.2%
	HTTPS-only										
	All			**Chrome**			**Safari**			**Other**	
DNS query (any)	13,177,108			9,487,295			3,602,160			87,653	
DNS Query HTTPS	3,708,895	28.1%		157,695	1.7%		3,506,664	97.3%		44,536	50.8%
QUIC First	2,710,668	20.6%		4,793	0.1%		2,701,516	75.0%		4,359	5.0%
HTTP/1,2 First	770,205	5.8%		1,164	0.0%		768,351	21.3%		690	0.8%

Table 1 – Use of HTTPS query for various browser types

It's evident from this table that only a very small set of Chrome users make an HTTPS query, and of these even fewer followup with a QUIC fetch. Given that the only way that the browser can obtain address information is via this HTTPS query in the HTTPS-only experiment, it's also evident that HTTPS DNS responses are making it back to the end client, and the decision to use HTTP/1,2 for the first fetch in 21% of cases is a local browser decision.

Who Uses the alt-svc directive?

The same question can be asked of the alt-svc directive. It's clear that this is used by the Chrome browser, but does the Safari browser use the **alt-svc** directive when no HTTPS DNS record is present?

To measure this, we used an experiment that had no HTTPS record, the web object contained the **alt-svc** directive, and the experiment performed a number of repeated fetches. The results are shown in Table 2.

	alt-svc only										
	All			Chrome			Safari			Others	
Samples	14,163,673			9,788,178			4,251,430			124,065	
HTTP/1,2 First Fetch	14,055,816	99.2%		9,787,962	100.0%		4,151,937	97.7%		115,917	93.4%
QUIC 1st Fetch	107,857	0.8%		216	0.0%		99,493	2.3%		8,148	6.6%
QUIC 2nd Fetch	9,183,332	64.8%		8,966,915	91.6%		122,086	2.9%		94,331	76.0%

Table 2 – Use of alt-svc directive for various browser types

It's evident from this data that the Safari browser makes no use of the alt-avc directive, and relies solely on the alt-svc directive to switch to use the QUIC protocol.

Conclusions and Questions

If you want to serve content over QUIC you need to support **both** QUIC trigger methods. You need to provision a DNS HTTPS record that specifies support of the HTTP/3 protocol (QUIC) and use an alt-svc directive in the HTTP content headers to signal QUIC capability to Safari and Chrome clients respectively.

Why doesn't the Chrome browser also use a DNS HTTPS query? Are they concerned about the significantly greater DNS query load that would result from such a change in this extensively used browser? And why doesn't Safari also use the **alt-svc** directive as a trigger to use QUIC?

And while we have questions, why do 24% of Safari clients not perform a QUIC fetch despite a HTTPS record indicating that the server supports QUIC?

"The good thing about standards is that there are so many to choose from!" is a somewhat disparaging comment that is commonly attributed to Andrew Tanenbaum. The IETF's RFC series has just published RFC 9813, and it would an insane flight to fancy to think that all these specifications are mutually consistent. They're not.

We have two standardised mechanisms used to trigger the use of the QUIC protocol. RFC 7838 describes HTTP Alternative Services, used by the alt-svc directive that is embedded in the HTTP headers of the content being serves. RFC 9460 specifies the DNS HTTPS Resource Record. HTTPS records allow a service to be provided from multiple service delivery points, each with associated parameters such as the transport protocol.

What we don't have is mutual interoperability. A client that is configured to look for the **alt-svc** directive will not necessarily query for a DNS HTTPS resource record, as there is no standard specification that mandates such an action. The same holds in the other direction, in that a client that scans for HTTP alternative services is not required by any standard specification to query for a DNS HTTPS resource record as well.

Many years ago, we were accustomed to criticising the OSI protocol suite on the basis that it standardised both connection-oriented and connectionless transport protocols. Two standards-compliant OSI endpoints were unable to interoperate if one endpoint exclusively used connection-oriented transport services and the other opted to use connectionless transport services. We find ourselves in a similar situation here with the triggering of the QUIC transport service and the use of HTTP alternative services and the DHS HTTP Resource Record.

The aim of QUIC is to improve the speed and efficiency of transactions, so it is hardly reasonable to standardise a "probe and wait" form of connection initiation. The desired QUIC triggering mechanism is one based on a positive indication from the content server that is is capable of supporting a QUIC connection, so when a client initiates a QUIC session it does so on the basis of a clear indication that the server supports the use of this protocol. Both the alt-svc direction and the HTTPS DNS resource record provide this functionality.

However, they have quite different properties. The client will be unaware that there is an **alt-svc** directive this is applicable to this resource until it has performed an initial fetch using the default transport protocol (in the HTTP context this implies a fetch over HTTP/1.0 or HTTP/2). Both of these HTTP protocol versions feature *session persistence*, where the overhead of establishing a TCP session and an overlay TLS session can be amortised across multiple individual HTTP fetch operations. The **alt-svc** approach is fast, lightweight and imposes little in the way of external side effects on other systems. However, if objective here is to facilitate a transition to use QUIC as broadly as possible, then the **alt-svc** is not exactly going to achieve that outcome.

The DNS HTTPS resource record can be folded into the connection initiation process, where in addition to the address record queries for the A and AAAA resource records, the HTTPS record can be used to signal the capability to use QUIC from the outset. The DNS operates within the constraint that each query contains just a single resource record. To query for a name's IPv4 and IPv6 addresses requires two query/response DNS transactions. They can be performed in parallel, but they remain separate DNS transactions. Adding a HTTPS query adds a third query to be used in the connection process. There is an obvious impact on the DNS infrastructure here with an increase in the DNS query load in line of the uptake of client systems that use this form of QUIC triggering.

Today we have a split environment in the web client world. Some two thirds of the client base use the Chrome browser and also exclusively use the **alt-svc** directive. Slightly under one quarter of the client base use the Safari browser, which exclusively uses the HTTPS DNS resource record.

The failure of to reach alignment on a single trigger system for QUIC in the client world implies that the cost burden of this failure is imposed on the service platforms, where both QUIC trigger systems need to be supported.

If the ultimate aim of the IETF's Standards process is to provide effective and efficient interoperability for both clients and servers alike, then in this case (and in other similar cases) the IETF's Standards Process is failing us. Diversity of approaches is fine in terms of overall resilience of an engineered system, but at a cost where every actor is placed in a position of having to support every approach to maximise interoperability. The case can be made that an effective standards process is more about making choices between functionally equivalent approaches, and far less about developing a huge suite of diverse ways of achieving the same outcome. I suspect that the IETF has already undergone this transition, and its core deliverable today is the standard themselves, and not necessarily an efficient and interoperable network platform as instantiated in the form of the public Internet!

Disclaimer

Author

Geoff Huston AM, M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region.

www.potaroo.net