September 2022
Geoff Huston

A Second Look at QUIC Use

A couple of months ago, in July 2022, I wrote about our work in measuring the level of use of QUIC in the Internet (*https://www.potaroo.net/ispcol/2022-07/quic.html*). Getting this measurement "right" has been an interesting exercise, and it's been a learning experience that I'd like to relate here. We'll start from the end of the previous article and carry on from there.

There's too little QUIC!

We've used APNIC Lab's Measurement platform where the measurement is embedded in a script within an online advertisement. The advertisement script directs the user to perform a number of URL fetches, and the servers that serve the referenced objects are instrumented to allow client capabilities and behaviours to be inferred from the server's actions.

In this case the client is directed to load a basic URL object (a minimal 1x1 pixel 'blot') where the domain name part of the URL is unique to each individual measurement. To set up a QUIC measurement we've taken the following steps:

- used the `nginx` server v1.23.1 with QUIC support included,
- used a URL domain name with a defined HTTPS RR Type whose value is `alpn="h3", and`
- used an Alternative Service directive on the content, namely `Alt-Svc: h3=":443"`, which is intended to direct the client to use HTTP/3 for subsequent retrievals.

Normally this last measure, the Alternative Service directive, would be largely ineffectual. Each client receives the ad as a single event and the script directs each ad to load once, so a client should not be performing a second load of the URL. In this case we've used a variant of the script that directs the client to wait for 2 seconds and then repeat the load of this URL. It is assumed that this delayed repeat would be sufficient for the client to act on the Alternative Service directive.

The QUIC measurement commenced at the start of June 2022. In this measurement we measure both the number of users who query for an HTTPS record and the number of users who use HTTP/3 (QUIC) to retrieve the URL.

Figure 1 compares the HTTP/3 retrieval rate using these two triggering processes. If the client does not use HTTP/3 in the first retrieval, but changes to use HTTP/3 in the second retrieval we assume it used the Alt-Svc. If the client used HTTP/3 in the initial retrieval, then we assume it used the DNS HTTPS mechanism.

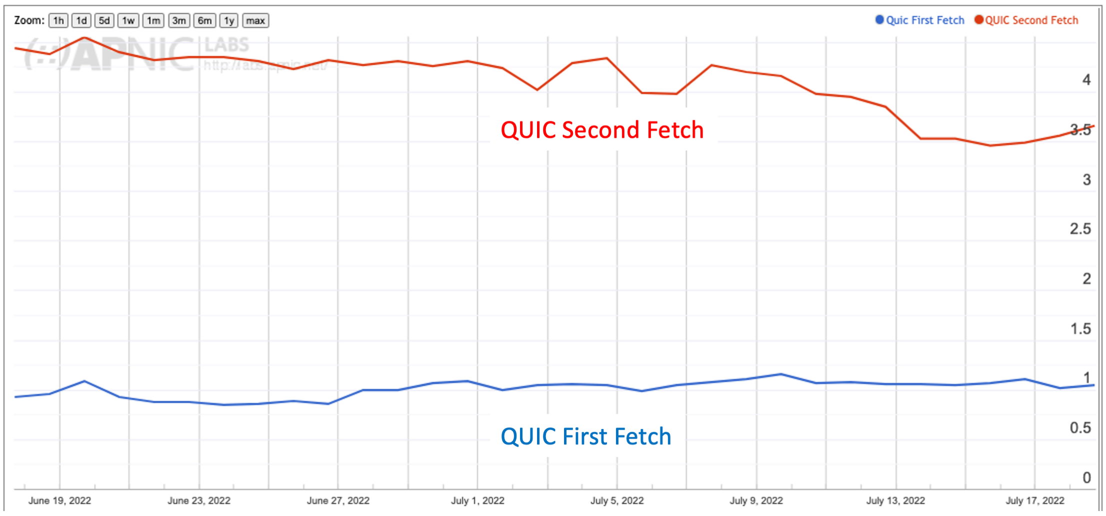

Figure 1 – QUIC Use for First and Second Fetches

The issue is that the second fetch rate is just too low. Google announced that the Chrome browser was adding support for QUIC in October 2020 (https://blog.chromium.org/2020/10/chrome-is-deploying-http3-and-ietf-quic.html), using the Alt-Svc directive to trigger QUIC. Cloudflare in its Radar Report (https://radar.cloudflare.com/) observes that 30% of sessions use QUIC, although it's unclear if the site is referring to traffic volume or session counts. In any case, a 30% use of QUIC is way more than between 3% and 4%.

What's going wrong?

Increase the number of repeats

Our first though was that the second repeat, scheduled 2 seconds after the first fetch was just not enough. The NGINX server will prefer to keep a session with the client open if it can, in order to amortise the cost of the TLS session establishment, and HTTP/2 can support this session reuse. Perhaps a single repeat fetch was not enough.

We increased the number of repeat fetches from 1 to 7, making a total of 8 fetches in all. The scheduling interval between fetches was kept at 2 seconds. This change improved the picture, lifting the second fetch HTTP/3 query rate from 4% to 26% (Figure 2).

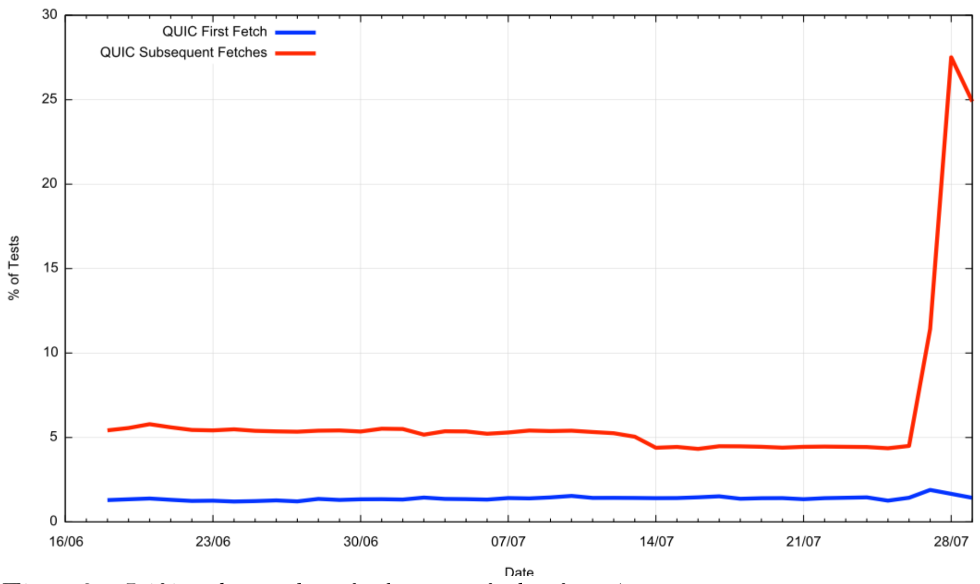

Figure 2 – Lifting the number of subsequent fetches from 1 to 7

The result was an obvious improvement, but it is still below the Cloudflare measurement. We were seeing around half of the Chrome browsers still not shifting to HTTP/3 across the repeat interval.

Adjust the server's keepalive timer

Persistent connections in HTTP help amortise the overheads of establishing a TCP connection (and a TLS connection for HTTPS) over a number of subsequent fetches from the same client (https://nginx.org/en/docs/http/ngx_http_core_module.html#keepalive_timeout). When a fetch is complete the server will keep the connection open for a further number of seconds (the "keepalive" interval) before shutting down the session. This improves the responsiveness of the server for the client at the expense of some additional memory state in the server to keep the session open.

In our case this server behaviour is not exactly what we want for the clients using the `Alt-Svc` directive to switch to HTTP/3. We are looking for the server to close the HTTP/2 session that was used for the initial fetch, and then have the browser client open up a new session, hopefully over QUIC and HTTP/3 for the second and subsequent fetches every time, rather than the more intermittent behaviours we are seeing across the 7 repeat fetches.

We tried setting the Nginx server's keepalive timer to 0 seconds, but this had the unintended side effect of disabling all QUIC support in the server. Clearly this was not the intended outcome!

We then raised the keepalive timer to 1 second, on the basis that a non-zero value would enable QUIC support. The result of these actions on the count of seen QUIC sessions is shown in Figure 3.

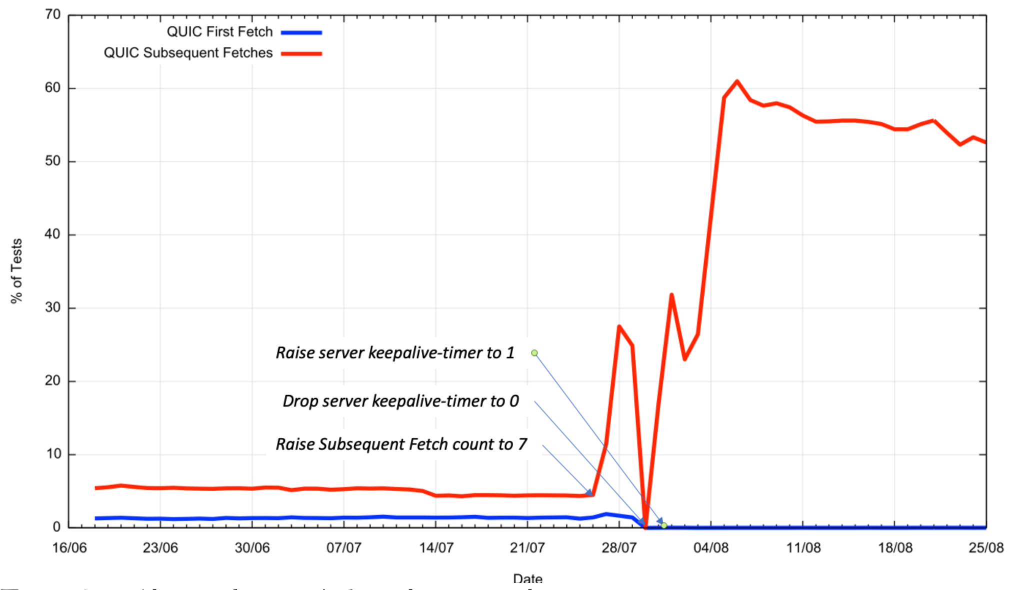

Figure 3 – Altering the server's keepalive time value

Clearly for the Chrome-like behaviour that is triggered by the `Alt-Svc` directive this has had the desired effect. The subsequent fetch cycle has seen the level of use of QUIC rise to over 50% of the tests.

It should be noted that slightly more than half of the QUIC-capable samples switched to use QUIC on the second fetch, the remainder switched on the third or later fetches, and in around 10% of fetch sequences clients switched back to use TLS over TCP. It was also noted that this 1 second timer value had the side effect of disabling all use of QUIC on the first fetch (the use that was triggered by the DNS HTTPS resource record, seen predominately in the Safari browsers).

It appears that the browsers that use the DNS lookup of the HTTPS record are experiencing the issue that the QUIC connection is shutting down before it can get established.

The IETF specification of QUIC (https://www.rfc-editor.org/rfc/rfc9000) includes an Idle Timeout (Section 10.1 of RFC 9000), and it is possible that the server's keepalive timer value is being used by the underlying QUIC transport as well, and the QUIC code is performing an early close of the QUIC session before the HTTP context can be established.

Further adjustment the server's keepalive timer

To test this theory, we adjusted the servers' keepalive parameter up from 1 second to 20 seconds.

At this point the measurement script is performing 7 repeat fetches, scheduled at 2 second intervals. The server's keepalive timeout is set to 20 seconds. Also, to speed up the DNS-triggered path we've added `ipv4hint` and `ipv6hint` fields to the HTTPS record in addition to the `alpn="h3"` field, allowing the client to bypass an additional DNS query, assuming that the client code will accept these fields in place of a further explicit DNS query.

These changes appeared to have addressed the major issues we've been having with low QUIC counts (Figure 4).

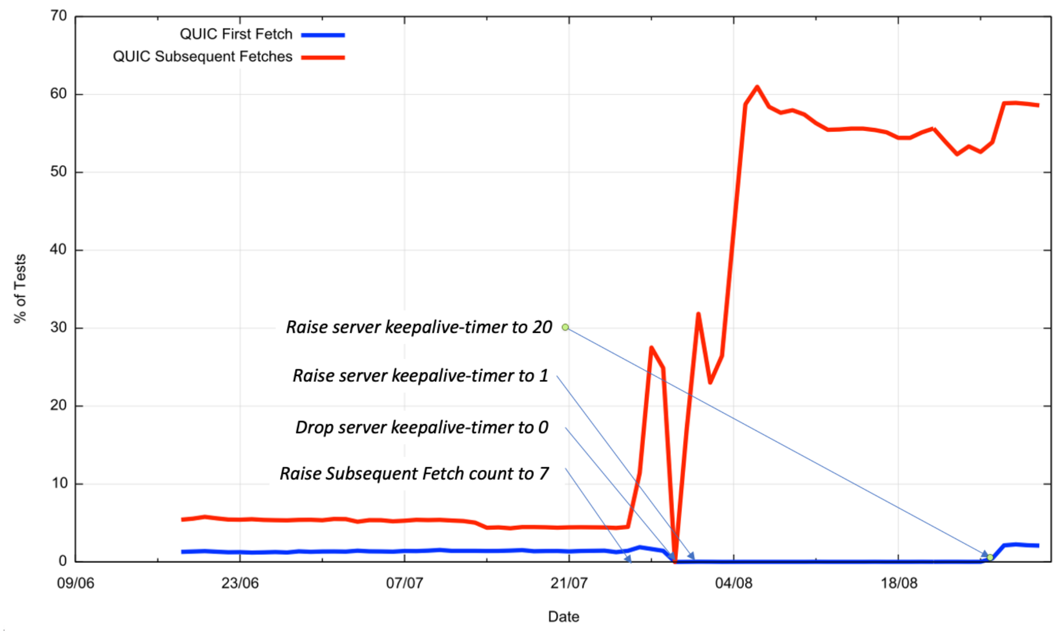

Figure 4 – Altering the server's keepalive time value to 20 seconds

The longer keepalive value allows the initial fetch to use QUIC, assuming that the client is one that uses the DNS HTTPS lookup, so we are one more seeing a first fetch value of 1.9% of samples using QUIC as a global average, while some economies are seeing use of QUIC on first fetch higher than 6%. Its difficult to tell whether this rate could be pushed to higher values or not by further exploration of the keepalive timers and the scheduling of the subsequent fetches in the measurement script. The data we do not have any access to is that of the failed connection attempts for QUIC where the outbound path from the client to the server discards UDP packets addressed to port 443. Overly enthusiastic local firewall filter rules have had quite notable impact on the robustness of other services (such as the use of IPv6-in-IPv4 tunnelling using 6to4), and we have no direct way of looking at the outbound behaviour of QUIC packets, so the robustness of this approach is challenging to measure.

In any case, a deployment rate across the entire internet of more than 50% is no mean feat. The world map of QUIC support shows this level of support for QUIC in in almost all economies, which the only major economy with a level of QUIC support below 20% being China (Figure 5).

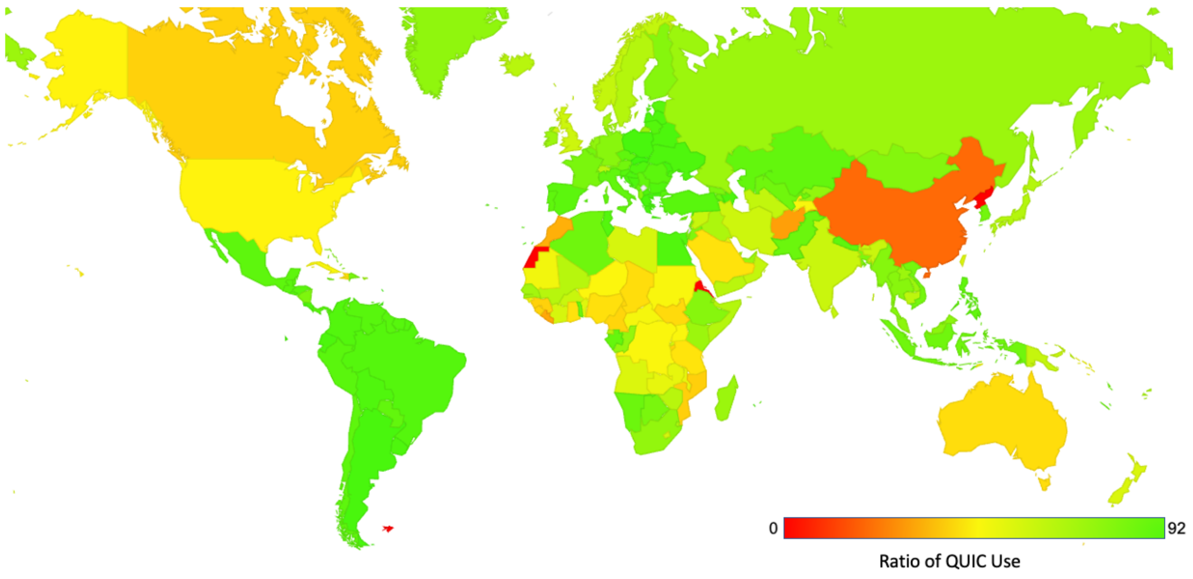

Figure 5 – QUIC use per economy – August 2022

Observations

If a server supports delivery of content over QUIC, then will clients use QUIC?

The answer appears to be "yes", in that a majority of browser clients will use QUIC when it is offered, but there are a number of caveats to this positive response. Given the majority of browser clients use Chrome, and Chrome still uses the `Alt-Svc` directive to switch to HTTP/3 for subsequent fetches, this implies that both the server and the content need to enable QUIC use, as well as requiring that the content is constructed to enable the use of multiple fetches in sequence, and, as we've found, this may require some careful tuning of the keepalive parameter to allow the client to switch over to use QUIC for subsequent fetches.

From this respect, the DNS HTTPS record approach looks preferable as a triggering mechanism for QUIC, allowing the QUIC attributes of faster connection establishment, 0-RTT session reestablishment, superior multi-session support and full encryption of the end-to-end transport parameters to be used from the point of first contact between client and server.

However, deploying change is becoming slower on the Internet, in line with its continued growth. In this case the change is not only in the behaviour of the client browser set (which is not a large set), nor the collection of servers (again, with the centrality pressures in mind, this is not a particularly large set). While QUIC has some obvious advantages in terms of offering a faster and more secure experience to the user, it appears that the downside lies in the cost of making changes to the various provisioning systems for the DNS and coordinating DNS published capabilities with server-side capabilities. But this is perhaps not as much as a barrier to widespread adoption as it might appear.

In a scan of the Tranco domain name list (https://tranco-list.eu/list/K2QGW, 31 August 2022), of the top 250K names in that list some 12.5% of these domain names have an HTTPS record. Most of these HTTPS records have a similar format, including address hint field values that point to Cloudflare servers, so in some ways this does not indicate a general adoption of HTTPS records in servers, but the use of this construct by a single CDN platform that is widely used. In Cloudflare's case the names of the services hosted by Cloudflare are served by authoritative servers also operated by Cloudflare, so the inclusion of the appropriate HTTPS records is relatively straightforward. Given the almost universal desire to speed up the elements of HTTP content delivery it's likely that other content delivery platforms will also adopt QUIC support in the near future.

The next step for the general adoption of QUIC in the browser world may well lie in the timing of the code release for the Chrome browser to add the HTTPS DNS record as the trigger to enable QUIC at the time of the initial fetch.

The APNIC Labs QUIC Use measurement is available at https://stats.labs.apnic.net/quic.

Disclaimer

The above views do not necessarily represent the views or positions of the Asia Pacific Network Information Centre.

Author

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

www.potaroo.net