September 2023
Geoff Huston

DNS is the new BGP

AUSNOG'23 was held in September. As usual, the meeting had a diverse collection of presentations on network technology, operational practices, engineering, and experiences. One of these presentations, by Cloudflare's Tom Peseka, was on the subject of service routing, highlighting the ways in which today's service platform attempt to optimise the user experience through service routing.

The classical model of the Internet (yes, I think the Internet is now old enough to have a "classical" model that refers to the original design and operational practices from some time back before, say, the year 2000!) was that you looked up the DNS with the name of the service or resource you wanted to connect to, the DNS duly returned an IP address, and then you opened an IP transport connection with that IP address. The quality of the service that ensued between you as the user and the service point was largely a network issue, which was influenced by the quality of the network path. That, in turn, was an outcome of the routing process. BGP was not just a reachability protocol, but also a traffic engineering protocol. Connecting customers to services was a BGP problem!

One of the big changes within the Internet over the last decade or so has been the shift to replicated services. Rather than operating a single service delivery point and use the network to connect the service's users to this delivery point, this service replication allows each individual service point to be positioned closer to clusters of users. The question now becomes who (and how) selects the "best" service point to use in response to each user's service request.

This is not a trivial question by any means. Given two network paths from a client to two different service instances how can we possibly compare them? Is the *best* path the *shortest* path? What is the unit of distance here to measure *short* and *long*? Do we need to map each point to its physical location coordinates and measure distance as a straight line on a map? Or is distance a measurement by packet transmission delay so that we use the network path, not the geographic path. Is *best* related to performance? Should a higher capacity path be preferred to a lower capacity path? Such questions abound, and simple and useful answers that are generally applicable are just not obvious. And even if we could agree on a metric for *best* for network paths (and we can't) then how do hosts and services perform this measurement?

There have been a few ways to respond to this over the years as we gain more experience with large scale service platforms and service routing.

Anycast

The first approach is to simply defer the problem to the inter-domain routing protocol and allow BGP to determine the *best* service point for each client. This is the anycast approach. Every individual service point shares a common IP address, and the advertisement of the same IP address into the routing system at multiple points will effectively segment the common network into areas of affinity to each service point. The network's routing system will direct each client to the "closest" instance of the service. The metric of distance used by BGP is the AS Path length, so BGP will select the path that minimizes the number of transit networks between the user and the service point.

Anycast has been used extensively in the root server system of the DNS, where each of the 13 uniquely addressed root services are not single DNS servers but are composed of a current total of 1,751 separate instances of a root service, organised into 13 anycast clusters.

The advantage of the anycast approach is that service points can be added and removed dynamically. There is no need for any form of service orchestration or coordination, because in an anycast context the BGP routing system performs this function. New instances will create an affinity are when the route is announced, and retiring instances will have their service affinity area absorbed into those of the remaining service set.

The issue with this approach is that it can be a relatively coarse form of service differentiation. In AS Path terms, the Internet is not *long and stringy* but *short and dense*. That means that AS Path length does not necessarily produce optimal outcomes, particularly in a sparsely populated service set.

The lack of direct control over steerage of clients to service points is also seen as a disadvantage by some service operators. What if the service operator wished to ensure some the overall quality of the service by attempting to balance the load across all server instances? Anycast can't achieve that. Or selecting the server that is reachable using the lowest delay, or the highest path capacity. BGP does not select routes based on path delay or path capacity.

If the service operator wants a greater level of control over the service selection, then it may choose to use some form of service routing.

Service Routing with the DNS

One approach is to use the DNS as the way to perform service routing. This approach uses distinct IP addresses for each service instance, each corresponding to a different location in the network.

How is this different to the conventional operation of the DNS? When a client asks the DNS for the IP address of a named service the DNS will by default just give IP addresses as the answer to the query. If there is more than one answer it will give them all! Here's an example of this behaviour, where a query for the name servers of the root zone of the DNS is answered with a list of all 13 root servers, and the 26 IPv6 and IPv4 addresses of each of the root servers are included in the response.

```
$ dig . NS

;; ANSWER SECTION:
.                       385969  IN      NS      k.root-servers.net.
.                       385969  IN      NS      c.root-servers.net.
.                       385969  IN      NS      g.root-servers.net.
.                       385969  IN      NS      e.root-servers.net.
.                       385969  IN      NS      f.root-servers.net.
.                       385969  IN      NS      a.root-servers.net.
.                       385969  IN      NS      h.root-servers.net.
.                       385969  IN      NS      i.root-servers.net.
.                       385969  IN      NS      j.root-servers.net.
.                       385969  IN      NS      d.root-servers.net.
.                       385969  IN      NS      l.root-servers.net.
.                       385969  IN      NS      m.root-servers.net.
.                       385969  IN      NS      b.root-servers.net.

;; ADDITIONAL SECTION:
a.root-servers.net.     385969  IN      AAAA    2001:503:ba3e::2:30
b.root-servers.net.     385969  IN      AAAA    2001:500:200::b
c.root-servers.net.     385969  IN      AAAA    2001:500:2::c
d.root-servers.net.     385969  IN      AAAA    2001:500:2d::d
e.root-servers.net.     385969  IN      AAAA    2001:500:a8::e
f.root-servers.net.     385969  IN      AAAA    2001:500:2f::f
g.root-servers.net.     385969  IN      AAAA    2001:500:12::d0d
h.root-servers.net.     385969  IN      AAAA    2001:500:1::53
i.root-servers.net.     385969  IN      AAAA    2001:7fe::53
j.root-servers.net.     385969  IN      AAAA    2001:503:c27::2:30
k.root-servers.net.     385969  IN      AAAA    2001:7fd::1
l.root-servers.net.     385969  IN      AAAA    2001:500:9f::42
m.root-servers.net.     385969  IN      AAAA    2001:dc3::35
a.root-servers.net.     385969  IN      A       198.41.0.4
b.root-servers.net.     385969  IN      A       199.9.14.201
c.root-servers.net.     385969  IN      A       192.33.4.12
d.root-servers.net.     385969  IN      A       199.7.91.13
e.root-servers.net.     385969  IN      A       192.203.230.10
```

```
f.root-servers.net.       385969   IN      A       192.5.5.241
g.root-servers.net.       385969   IN      A       192.112.36.4
h.root-servers.net.       385969   IN      A       198.97.190.53
i.root-servers.net.       385969   IN      A       192.36.148.17
j.root-servers.net.       385969   IN      A       192.58.128.30
k.root-servers.net.       385969   IN      A       193.0.14.129
l.root-servers.net.       385969   IN      A       199.7.83.42
m.root-servers.net.       385969   IN      A       202.12.27.33
```

This comprehensive response is uniquely unhelpful in terms of service routing, as it provides no indication of which of these servers (or, more accurately, which specific instances of these services, as they are all, in fact, anycast clouds) is the better service for the client to use.

What DNS-based service routing does is to filter this list of all service delivery points, and only offer the client the one IP address of the service point that is considered to be optimal for the querier. This is achieved by performing a service location processing step at the authoritative DNS server when forming responses to each and every query. The authoritative DNS server returns a single IP address, where the associated server instance represents a *best service* selection for the client. In effect the client is being routed to a server not by the routing system, but by the DNS itself.

It's the DNS that performs service routing, not the BGP routing protocol.

For example, here’s the resolution of an anycast-hosted service, hosted on the Akamai platform:

```
$ dig www.sbs.com.au

;; ANSWER SECTION:
www.sbs.com.au.           300      IN      CNAME   www.sbs.com.au.edgekey.net.
www.sbs.com.au.edgekey.net. 300    IN      CNAME   e7065.b.akamaiedge.net.
e7065.b.akamaiedge.net.   20       IN      A       23.195.84.241


$ traceroute 23.195.84.241
traceroute to 23.195.84.241 (23.195.84.241), 64 hops max
  1   10.109.0.1 (10.109.0.1)  6.186ms  2.353ms  3.980ms
  2   61.206.224.254 (61.206.224.254)  6.286ms  4.569ms  4.621ms
  3   61.206.224.165 (61.206.224.165)  7.752ms  6.966ms  6.968ms
  4   219.110.0.229 (core1-kote.bb.itscom.jp)  8.141ms  31.571ms  4.863ms
  5   219.110.0.241 (asbr2-kote.bb.itscom.jp)  7.273ms  5.718ms  5.563ms
  6   210.171.224.189 (210.171.224.189)  8.485ms  38.239ms  11.497ms
  7   *  *  *

$ ping www.sbs.com.au
PING e7065.b.akamaiedge.net (23.195.84.241): 56 data bytes
64 bytes from 23.195.84.241: icmp_seq=0 ttl=57 time=10.943 ms
64 bytes from 23.195.84.241: icmp_seq=1 ttl=57 time=6.409 ms
64 bytes from 23.195.84.241: icmp_seq=2 ttl=57 time=13.810 ms
64 bytes from 23.195.84.241: icmp_seq=3 ttl=57 time=13.763 ms
^C--- e7065.b.akamaiedge.net ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 6.409/11.231/13.810/3.016 ms
```

This is an Australian content service, but the query was made in Japan. The individual service name (www.sbs.com.au) is lifted out of the control of the parent domain zone due to the first CNAME record, taking the recursive resolver to an edgekey.net domain (www.sbs.com.au.edgekey.net). At this point DNS routing is invoked, and the IP address of the recursive resolver making the query is matched against the set of servers that can provide a service for this particular service, and then the generic mapped service name is then mapped to a specific service in the second CNAME (e7065.b.akamaiedge.net). When the client contacts this server, it will use the service name identifier www.sbs.com.au, allowing the server to select the appropriate TLS context and service context. This will be the case irrespective of whether the session uses TLS over TCP or TLS embedded within QUIC. In this case the DNS steering service has selected a server that is located very close to the client, some 7 IP hops away with just 6ms round trip delay, as shown in the traceroute and ping records.

This is confirmed with a query to an online geolocation service:

```
$ curl -s ipinfo.io/23.195.84.241
{
   "ip": "23.195.84.241",
   "hostname": "a23-195-84-241.deploy.static.akamaitechnologies.com",
   "city": "Tokyo",
  "region": "Tokyo",
   "country": "JP",
   "loc": "35.6895,139.6917",
   "org": "AS20940 Akamai International B.V.",
   "postal": "101-8656",
   "timezone": "Asia/Tokyo",
}
```

The assumption behind this approach is that each client is sufficiently close to their recursive resolver such that when authoritative server for the service name calculates the *closest* server instance based on the IP address of the recursive resolver, then the outcome is relatively accurate for the client.

Fastly uses a similar approach. The following sequence shows the resolution of a Fastly-hosted service.

```
$ dig www.theguardian.com
www.theguardian.com.      6989     IN       CNAME    dualstack.guardian.map.fastly.net.
dualstack.guardian.map.fastly.net. 30 IN A 151.101.89.111

$ curl -s ipinfo.io/151.101.89.111 | grep city
  "city": "Osaka",

$ dig +short whoami.akamai.net
220.247.144.122

$ curl -s ipinfo.io/220.247.144.122 | grep city
  "city": "Osaka",
```

Fastly map this URL to a server located in Osaka, based on the geolocation of the client's DNS recursive resolver, which is also geolocated to Osaka.

The advantage of this approach is that the steering of clients to servers is under the direct control of the service provider, not the network provider(s). This allows the service provider the ability to exercise some level of control over the resultant service delivery quality. The caveat here is that the recursive resolver will cache the DNS outcome and will reuse this DNS response for future clients of the same recursive resolver for the DNS cache lifetime. If the service provider wants to achieve even finer levels of control over this DNS steering, then it could use a lower TTL in the DNS response. For example, in the example above the Akamai DNS response has a TTL of 20 seconds. However, the TTL value is only a suggestion in the DNS, rather than a strictly enforced rule, and recursive resolvers have been observed to hold onto responses for longer periods in order to improve DNS performance through longer cache retention.

Service Routing in the DNS with Client Subnet

While this assumption of the colocation of client and recursive resolver was reasonable in the early days of DNS-based service steering, the rise of the use of open recursive resolvers has undermined aspects of this approximate co-location assumption.

A client may access a remote open recursive resolver with the inevitable result that a DNS-level service steering would make a poor choice of server to use for the service transaction.

Let's look at the example for the Guardian service with Fastly, but this time we will use three different resolvers, namely a stub (in-host) resolver, a local recursive resolver, and Cloudflare's 1.1.1.1 open recursive resolver:

```
# stub
$ dig +short www.theguardian.com @::1 | awk '{ print "curl -s ipinfo.io/" $1 " | grep city"}' | /bin/sh
  "city": "Tokyo",
# ISP
$ dig +short www.theguardian.com | awk '{ print "curl -s ipinfo.io/" $1 " | grep city"}' | /bin/sh
  "city": "Osaka",
# Cloudflare Open
$ dig +short www.theguardian.com @1.1.1.1 | awk '{ print "curl -s ipinfo.io/" $1 " | grep city"}' | /bin/sh
  "city": "San Francisco",
# Google Open
$ dig +short www.theguardian.com @8.8.8.8 | awk '{ print "curl -s ipinfo.io/" $1 " | grep city"}' | /bin/sh
  "city": "San Francisco",
```

The host is located at Kyoto, so it's likely that Osaka is the best outcome here, but the difference between Osaka and Tokyo is small enough that either result would be acceptable. But if I choose to repeat the query using Cloudflare's 1.1.1.1 open resolver, or Google's 8.8.8.8 public resolver, then the response teleports me to San Francisco!

This non-optimal DNS-steering outcome led to a practice shared between some open resolver operators and DNS-steering service platforms where the open resolver attaches some details of the source of the original DNS query to the query that the recursive resolver passes to the service's authoritative name server. Having the full client address attached to the DNS query was considered to be overly compromising of user privacy (as the implication that this query data effectively forms a record of which user has interests in which Internet resources), so the practice was to blur it slightly by attaching a subnet parameter which included the client's IP address. This was achieved by having the recursive resolver using the extension mechanism for DNS (EDNS) and placing the subnet of the original client query into the query being made to the authoritative server. This way the authoritative server could perform a selection based not on the IP address of the recursive resolver, but on the subnet of the end client.

The RFC describing this procedure, EDNS Client Subnet, RFC7871, has some disparaging commentary on this process: "We recommend that the feature be turned off by default in all nameserver software." Pretty stern stuff for a technical specification! Nevertheless, a number of content service platforms, including Akamai and Fastly, appear to use this approach of customising the response from the authoritative name servers, using the DNS as the service routing protocol rather than any of the conventional routing protocols.

Service Routing within the Application

The issue with service routing via the DNS is that it cannot easily be customised to individual clients. Once the response is passed back to the recursive resolver, the service provider has no control over which clients are passed the IP address of this particular service instance. If a server fails it may take some time for that IP address to expire from the various DNS caches, and load balancing across service instances is very challenging.

This leads to the next approach, which is that of performing service routing within the application. Here is an example of the resolution of the service name of network, made from a client located in Japan.

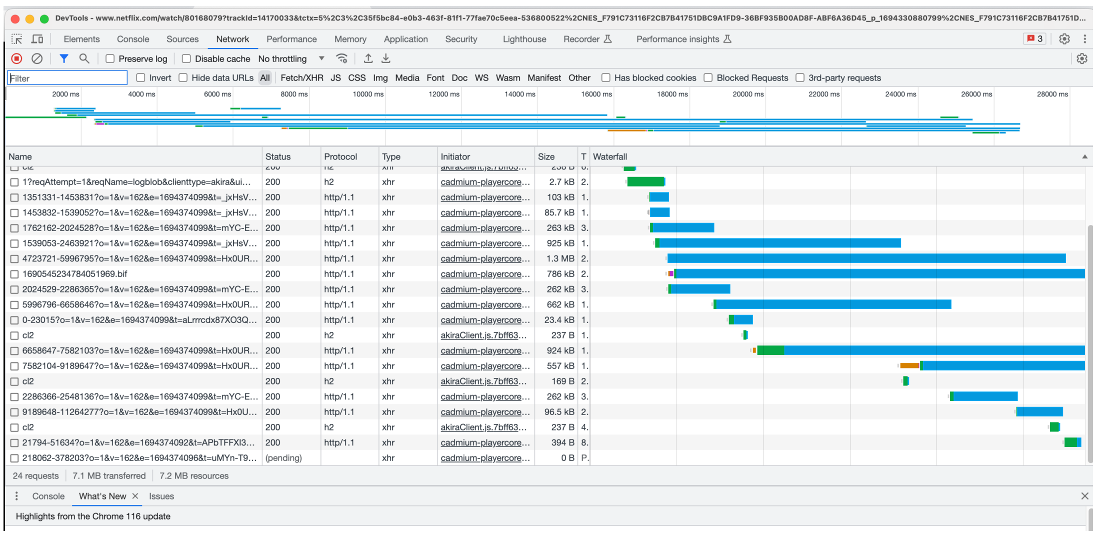
```
$ dig www.netflix.com

;; ANSWER SECTION:
www.netflix.com. 268      IN       CNAME    www.dradis.netflix.com.
www.dradis.netflix.com.   48       IN       CNAME    www.us-west-2.internal.dradis.netflix.com.
www.us-west-2.internal.dradis.netflix.com.       48       IN       CNAME       apiproxy-website-nlb-prod-1-
bcf28d21f4bbcf2c.elb.us-west-2.amazonaws.com.
apiproxy-website-nlb-prod-1-bcf28d21f4bbcf2c.elb.us-west-2.amazonaws.com. 48 INA 44.242.13.161
apiproxy-website-nlb-prod-1-bcf28d21f4bbcf2c.elb.us-west-2.amazonaws.com. 48 INA 52.38.7.83
apiproxy-website-nlb-prod-1-bcf28d21f4bbcf2c.elb.us-west-2.amazonaws.com. 48 INA 44.240.158.19

;; AUTHORITY SECTION:
elb.us-west-2.amazonaws.com. 150 IN        NS       ns-424.awsdns-53.com.
elb.us-west-2.amazonaws.com. 150 IN        NS       ns-748.awsdns-29.net.
elb.us-west-2.amazonaws.com. 150 IN        NS       ns-1283.awsdns-32.org.
elb.us-west-2.amazonaws.com. 150 IN        NS       ns-1870.awsdns-41.co.uk.
```

These service addresses geolocate to a data server located in Oregon, in the US. This makes no sense for a client that is located in, say, Japan. But the Netflix service is not a monolithic service. Each element of a video stream is effectively broken up into *chunks* and the application in the client side interacts with the server side to determine the optimal location for serving each *chunk*. In Netflix's terminology this approach is termed "Open Connect".

In the developer screenshot of a browser playing a Netflix video stream it's clear that the movie has been broken into a set of such *chunks*.



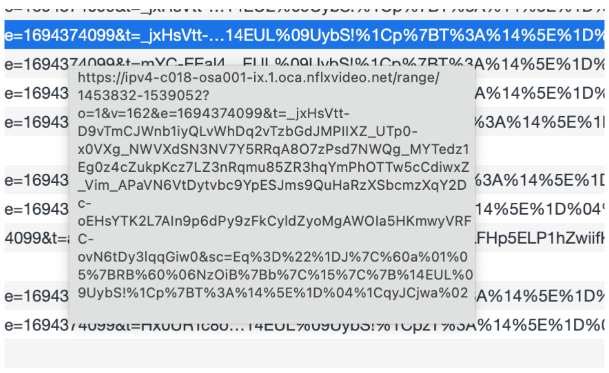
DevTools - www.netflix.com/watch/80168079?trackId=14170033&tctx=5%2C3%2C35f5bc84-e0b3-463f-81f1-77fae70c5eea-536800522%2CNES_F791C73116F2CB7B41751DBC9A1FD9-36BF935B00AD8F-ABF6A36D45_p_1694330880799%2CNES_F791C73116F2CB7B41751D…

Name	Status	Protocol	Type	Initiator	Size	T
1?reqAttempt=1&reqName=logblob&clienttype=akira&ui…	200	h2	xhr	cadmium-playercore…	2.7 kB	2.
1351331-1453831?o=1&v=162&e=1694374099&t=_jxHsV…	200	http/1.1	xhr	cadmium-playercore…	103 kB	1.
1453832-1539052?o=1&v=162&e=1694374099&t=_jxHsV…	200	http/1.1	xhr	cadmium-playercore…	85.7 kB	1.
1762162-2024528?o=1&v=162&e=1694374099&t=mYC-E…	200	http/1.1	xhr	cadmium-playercore…	263 kB	3.
1539053-2463921?o=1&v=162&e=1694374099&t=_jxHsV…	200	http/1.1	xhr	cadmium-playercore…	925 kB	1.
4723721-5996795?o=1&v=162&e=1694374099&t=Hx0UR…	200	http/1.1	xhr	cadmium-playercore…	1.3 MB	2.
1690545234784051969.bif	200	http/1.1	xhr	cadmium-playercore…	786 kB	2.
2024529-2286365?o=1&v=162&e=1694374099&t=mYC-E…	200	http/1.1	xhr	cadmium-playercore…	262 kB	3.
5996796-6658646?o=1&v=162&e=1694374099&t=Hx0UR…	200	http/1.1	xhr	cadmium-playercore…	662 kB	1.
0-23015?o=1&v=162&e=1694374099&t=aLrrrcdx87XO3Q…	200	http/1.1	xhr	cadmium-playercore…	23.4 kB	1.
cl2	200	h2	xhr	akiraClient.js.7bff63…	237 B	1.
6658647-7582103?o=1&v=162&e=1694374099&t=Hx0UR…	200	http/1.1	xhr	cadmium-playercore…	924 kB	1.
7582104-9189647?o=1&v=162&e=1694374099&t=Hx0UR…	200	http/1.1	xhr	cadmium-playercore…	557 kB	1.
cl2	200	h2	xhr	akiraClient.js.7bff63…	169 B	2.
2286366-2548136?o=1&v=162&e=1694374099&t=mYC-E…	200	http/1.1	xhr	cadmium-playercore…	262 kB	3.
9189648-11264277?o=1&v=162&e=1694374099&t=Hx0U…	200	http/1.1	xhr	cadmium-playercore…	96.5 kB	2.
cl2	200	h2	xhr	akiraClient.js.7bff63…	237 B	4.
21794-51634?o=1&v=162&e=1694374092&t=APbTFFXI3…	200	http/1.1	xhr	cadmium-playercore…	394 B	8.
218062-378203?o=1&v=162&e=1694374096&t=uMYn-T9…	(pending)		xhr	cadmium-playercore…	0 B	P.

24 requests | 7.1 MB transferred | 7.2 MB resources

Highlights from the Chrome 116 update

The full URL of a *chunk* is shown in the next figure. The Netflix server has determined the location of the client device, and now is sending the client a specific sequence of URLs that are served in a location close to the client.



e=1694374099&t=_jxHsVtt-…14EUL%09UybS!%1Cp%7BT%3A%14%5E%1D%

```
https://ipv4-c018-osa001-ix.1.oca.nflxvideo.net/range/
1453832-1539052?
o=1&v=162&e=1694374099&t=_jxHsVtt-
D9vTmCJWnb1iyQLvWhDq2vTzbGdJMPlIXZ_UTp0-
x0VXg_NWVXdSN3NV7Y5RRqA8O7zPsd7NWQg_MYTedz1
Eg0z4cZukpKcz7LZ3nRqmu85ZR3hqYmPhOTTw5cCdiwxZ
_Vim_APaVN6VtDytvbc9YpESJms9QuHaRzXSbcmzXqY2D
c-
oEHsYTK2L7AIn9p6dPy9zFkCyldZyoMgAWOIa5HKmwyVRF
C-
ovN6tDy3lqqGiw0&sc=Eq%3D%22%1DJ%7C%60a%01%0
5%7BRB%60%06NzOiB%7Bb%7C%15%7C%7B%14EUL%0
9UybS!%1Cp%7BT%3A%14%5E%1D%04%1CqyJCjwa%02
```

```
$ dig +short ipv4-c018-osa001-ix.1.oca.nflxvideo.net
45.57.83.136
$ curl -s ipinfo.io/45.57.83.136 | grep city
  "city": "Osaka",
```

For larger service transactions that are not interactive, such as streaming downloads, this shift from a single initial geolocation function that directs the client to a service instance for the entirety of the subsequent download into a set of separable server decisions for each chunk of the downloaded content is an important change. This removes the indirection and caching side-effects that accompany the DNS-based content steering and allows the client to directly interact with the service orchestrator so that instances of poor outcomes in delivery of content components can allow dynamic redirection to a difference server instance.

This approach used by YouTube is a hybrid approach, using both DNS steering and breaking the content into chunks that are served from nearby content caches. The service uses DNS steering to route the client to a collection of front-end service units that are located close to the client. Here are four examples:

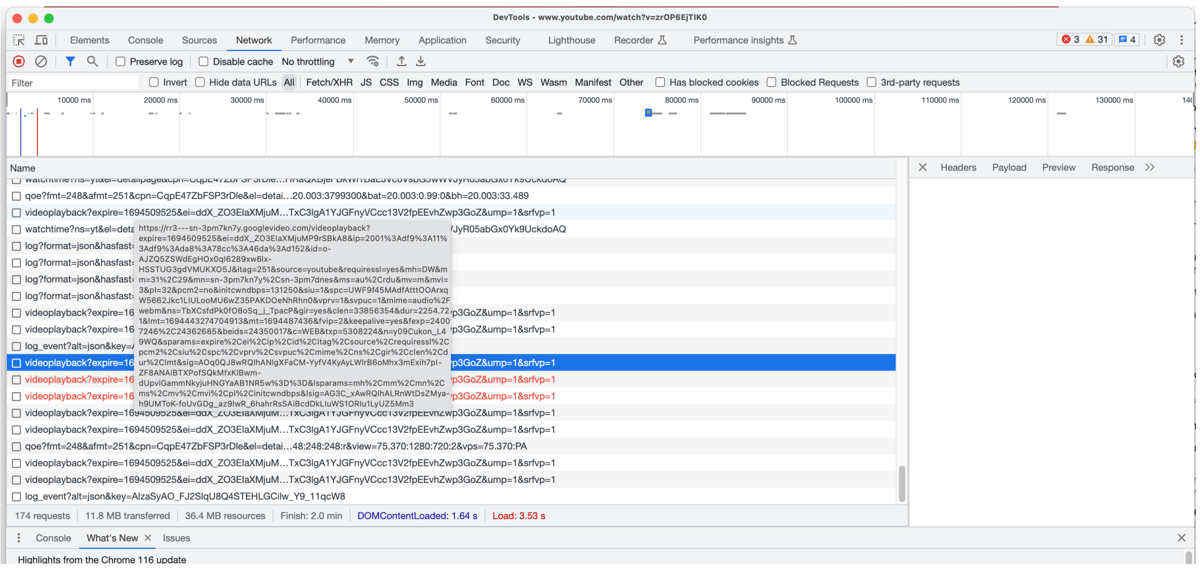
```
# query from Hong Kong
dig +short www.youtube.com | awk '{print "curl -s ipinfo.io/" $1 "| grep city"}' | /bin/sh | uniq -c
     16   "city": "Hong Kong",

# query from US (Dallas):
$ dig +short www.youtube.com | awk '{print "curl -s ipinfo.io/" $1 "| grep city"}' | /bin/sh | uniq -c
     16   "city": "Dallas",

# query from EU (Frankfurt an Main)
$ dig +short www.youtube.com | awk '{print "curl -s ipinfo.io/" $1 "| grep city"}' | /bin/sh | uniq -c
     16   "city": "Frankfurt am Main",

# query from Singapore
$ dig +short www.youtube.com | awk '{print "curl -s ipinfo.io/" $1 "| grep city"}' | /bin/sh | uniq -c
     16   "city": "Singapore",
```

A Youtube video stream is broken into chunks, similarly to Netflix. Each chunk served from a content cache service located close to the user.



```
DevTools - www.youtube.com/watch?v=zrOP6EjTIK0

Name
qoe?fmt=248&afmt=251&cpn=CqpE47ZbFSP3rDle&el=detai…20.003:3799300&bat=20.003:0.99:0&bh=20.003:33.489
videoplayback?expire=1694509525&ei=ddX_ZO3EIaXMjuM…TxC3lgA1YJGFnyVCcc13V2fpEEvhZwp3GoZ&ump=1&srfvp=1
watchtime?ns=yt&el=deta                                                             /JyR05abGx0Yk9UckdoAQ
log?format=json&hasfast=
log?format=json&hasfast=
log?format=json&hasfast=
log?format=json&hasfast=
videoplayback?expire=16                                                             p3GoZ&ump=1&srfvp=1
videoplayback?expire=16                                                             p3GoZ&ump=1&srfvp=1
log_event?alt=json&key=/
videoplayback?expire=16                                                             p3GoZ&ump=1&srfvp=1
videoplayback?expire=16                                                             p3GoZ&ump=1&srfvp=1
videoplayback?expire=16                                                             p3GoZ&ump=1&srfvp=1
videoplayback?expire=1694509525&ei=ddX_ZO3EIaXMjuM…TxC3lgA1YJGFnyVCcc13V2fpEEvhZwp3GoZ&ump=1&srfvp=1
videoplayback?expire=1694509525&ei=ddX_ZO3EIaXMjuM…TxC3lgA1YJGFnyVCcc13V2fpEEvhZwp3GoZ&ump=1&srfvp=1
qoe?fmt=248&afmt=251&cpn=CqpE47ZbFSP3rDle&el=detai…48:248:248:r&view=75.370:1280:720:2&vps=75.370:PA
videoplayback?expire=1694509525&ei=ddX_ZO3EIaXMjuM…TxC3lgA1YJGFnyVCcc13V2fpEEvhZwp3GoZ&ump=1&srfvp=1
videoplayback?expire=1694509525&ei=ddX_ZO3EIaXMjuM…TxC3lgA1YJGFnyVCcc13V2fpEEvhZwp3GoZ&ump=1&srfvp=1
log_event?alt=json&key=AIzaSyAO_FJ2SlqU8Q4STEHLGCilw_Y9_11qcW8

https://rr3---sn-3pm7kn7y.googlevideo.com/videoplayback?
expire=1694509525&ei=ddX_ZO3EIaXMjuMP9rSBkA8&ip=2001%3Adf9%3A11%
3Adf9%3Ada8%3A78cc%3A46da%3Ad152&id=o-
AJZQ5ZSWdEgHOx0ql6289xw6lx-
HSSTUG3gdVMUKXO5J&itag=251&source=youtube&requiressl=yes&mh=DW&m
m=31%2C29&mn=sn-3pm7kn7y%2Csn-3pm7dnes&ms=au%2Crdu&mv=m&mvi=
3&pl=32&pcm2=no&initcwndbps=131250&siu=1&spc=UWF9f45MAdfAtttOOArxq
W5662Jkc1LIULooMU6wZ35PAKDOeNhRhn0&vprv=1&svpuc=1&mime=audio%2F
webm&ns=TbXCsfdPk0fO8oSq_j_TpacP&gir=yes&clen=33856354&dur=2254.72
1&lmt=1694443274704913&mt=1694487436&fvip=2&keepalive=yes&fexp=2400
7246%2C24362685&beids=24350017&c=WEB&txp=5308224&n=y09Cukon_L4
9WQ&sparams=expire%2Cei%2Cip%2Cid%2Citag%2Csource%2Crequiressl%2C
pcm2%2Csiu%2Cspc%2Cvprv%2Csvpuc%2Cmime%2Cns%2Cgir%2Cclen%2Cd
ur%2Clmt&sig=AOq0QJ8wRQIhANlgXFaCM-YyfV4KyAyLWlrB6oMhx3mExih7pl-
ZF8ANAiBTXPofSQkMfxKlBwm-
dUpviGammNkyjuHNGYaAB1NR5w%3D%3D&lsparams=mh%2Cmm%2Cmn%2C
ms%2Cmv%2Cmvi%2Cpl%2Cinitcwndbps&lsig=AG3C_xAwRQIhALRnWtDsZMya-
h9UMToK-foUvGDg_az9lwR_6hahrRsSAiBcdDkLIuWS1ORlu1LyUZ5Mm3

174 requests | 11.8 MB transferred | 36.4 MB resources | Finish: 2.0 min | DOMContentLoaded: 1.64 s | Load: 3.53 s

Highlights from the Chrome 116 update
```

```
$ dig rr3---sn-3pm7kn7y.googlevideo.com
rr3---sn-3pm7kn7y.googlevideo.com. 1008    IN CNAME rr3.sn-3pm7kn7y.googlevideo.com.
rr3.sn-3pm7kn7y.googlevideo.com. 1009 IN A 172.217.137.200

$ curl -s ipinfo.io/172.217.137.200 | grep city
  "city": "Osaka",
```

In this case a video stream initiated in the city of Kyoto, Japan is served from a content cache located in Osaka, Japan.

The advantage of this from of steerage is that the service response can be tailored to individual clients, rather than having a DNS-based response that impacts all end clients of a particular recursive resolver. The approach offers the service platform considerably greater levels of flexibility in managing the elements of the aggregate service delivery platform.

Measuring Distance

How can you determine the distance between two points on the Internet?

A long-standing approach has been to measure the delay a packet experiences to be passed from a source to a destination and for the response to be passed back to the original source. This is the so-called *ping time*. Irrespective of their geographical locations and the straight-line distance, the *ping time* measures the network propagation delay which is directly related to achievable performance that can be realised by transport protocols. It's a highly relevant measurement if you are trying to understand distance in the sense of achievable service performance. The problem with this kind of metric is that it's an active metric based on probing where one end of the pairwise measurement measures the time distance to the other. It's not an additive metric and it's not a cacheable metric, as the traffic load state on the path impacts each measurement. It also takes time to execute the ping probes.

Some DNS resolver implementations use a variant of this form of measured end-to-end delay to select the best authoritative server for a domain when multiple such servers are configured in the DNS. The resolver and make an initial selection and will keep track of the query/response delay. The resolver will periodically query the other name servers for this zone, and switch to a lower delay server if this periodic probing exposes a more responsive server. Aside from this DNS use of probed round-trip-time delay, there does not appear to be much use of this technique of determining distance.

At the other end of the spectrum is geographic distance. There are a number of databases, some public some accessible by subscription and some private that map IP addresses to geographic coordinates. This can be political, as in country, province/state and city, or Lat/Long coordinates. A common resource is

Maxmind, but there are others, such as ipinfo.io used in the examples here that provide similar information. A matching system based on city or state/province may produce acceptable outcomes, or a distance calculation based on lat/long distance, assuming that the coordinates are reasonably accurate in the first place.

The assumption behind the use of geolocation to inform a service steering decision is that network interconnectivity is highly meshed. There have been notable failures in such meshed interconnectivity assumptions, where the packet path may involve a large detour between two geographically adjacent points. A distance calculation may simply write this off as a small-scale anomaly in the overall distance-based service steering, or fold in a set of specific exceptions.

The advantage of this approach is that it can be implemented efficiently and operate within the timescale of a DNS transaction. Given that one side of the calculation are the known locations of service points the entire set of distance calculations can be mapped into a collection of lookup tables.

There is also the approach of in-session service refinement when using a deconstructed content model, such as the one used by Netflix. The session can interleave test elements from neighbouring service delivery points to test if the alternate points can deliver a better service experience. If so then the service platform can change the affinity between the client and a service delivery point to improve the overall service outcome.

Service Delivery in a Service Dense World

The result of a number of decades of Moore's Law at play in the silicon world has meant that the unit price of processing and storage solutions have plummeted while the availability has increased dramatically. We no longer use a network to bring users up to the service portal, but instead use the abundance of processing, storage and capacity to replicate the service across the network, so that in effect the service has come to the user.

This is no longer a carriage and a routing problem. It's a service selection problem, and the steering of the user to the best service portal is best achieved by using the DNS as the way of linking users and services.

This approach of DNS steering in the service delivery world has some pretty profound implications for the network, in aspects of architecture, design, robustness and utility. If we no longer rely on carriage services to sustain service quality, then we are no longer reliant on the routing system to produce optimal outcomes in terms of path quality. The true value of adding robust security to the routing environment dissipates in accordance with our reduced dependence on routing as a service platform.

Our critical reliance shifts towards the DNS with the use of the DNS as the service selector. It's part of a larger picture of the Internet in lifting the intrinsic value of the networked environment out of the network itself and positioning the application layer as the centrepiece of value in the digital environment. It may well be that the picture of the concentration of value into the small collection of content behemoths in today's world is still in its early days and the process will produce even further concentration and yet greater levels of centrality and dominance in coming years before any new model emerges to displace it. Considering that the former giants of the industrial age are still with us today, the prospects of the current digital giants securing a bountiful future for themselves for some decades to come is looking pretty solid in my view!

Disclaimer

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