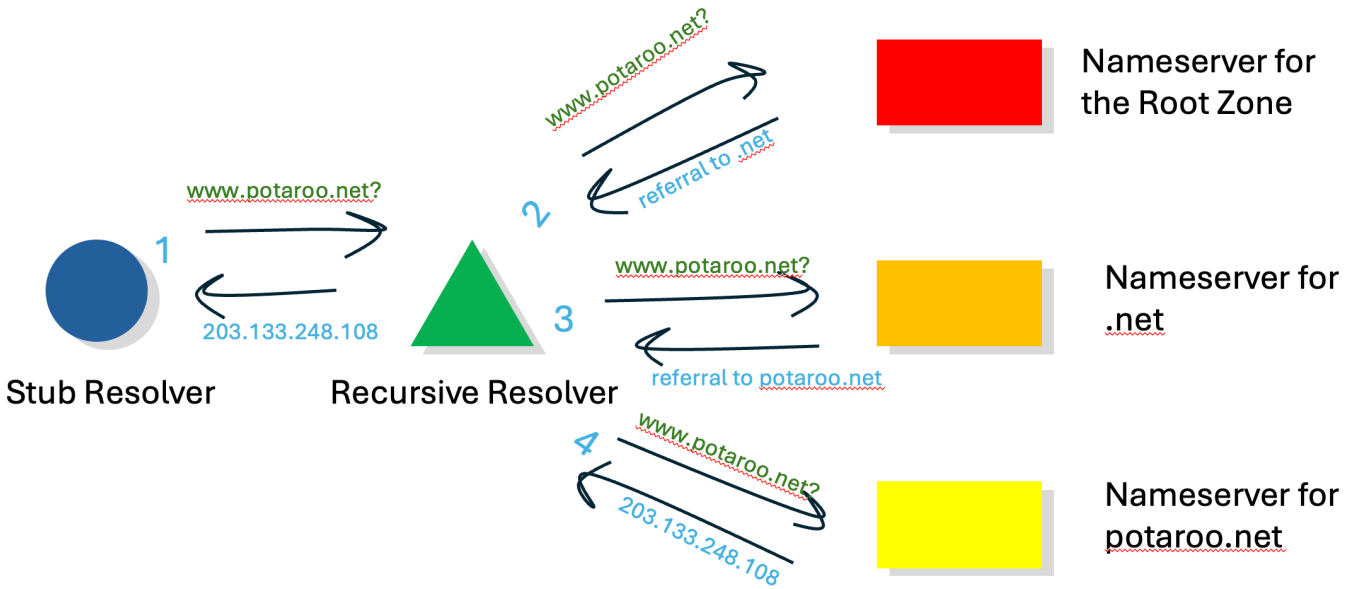
Some Thoughts on the Root of the DNS

Geoff Huston

Disclaimer

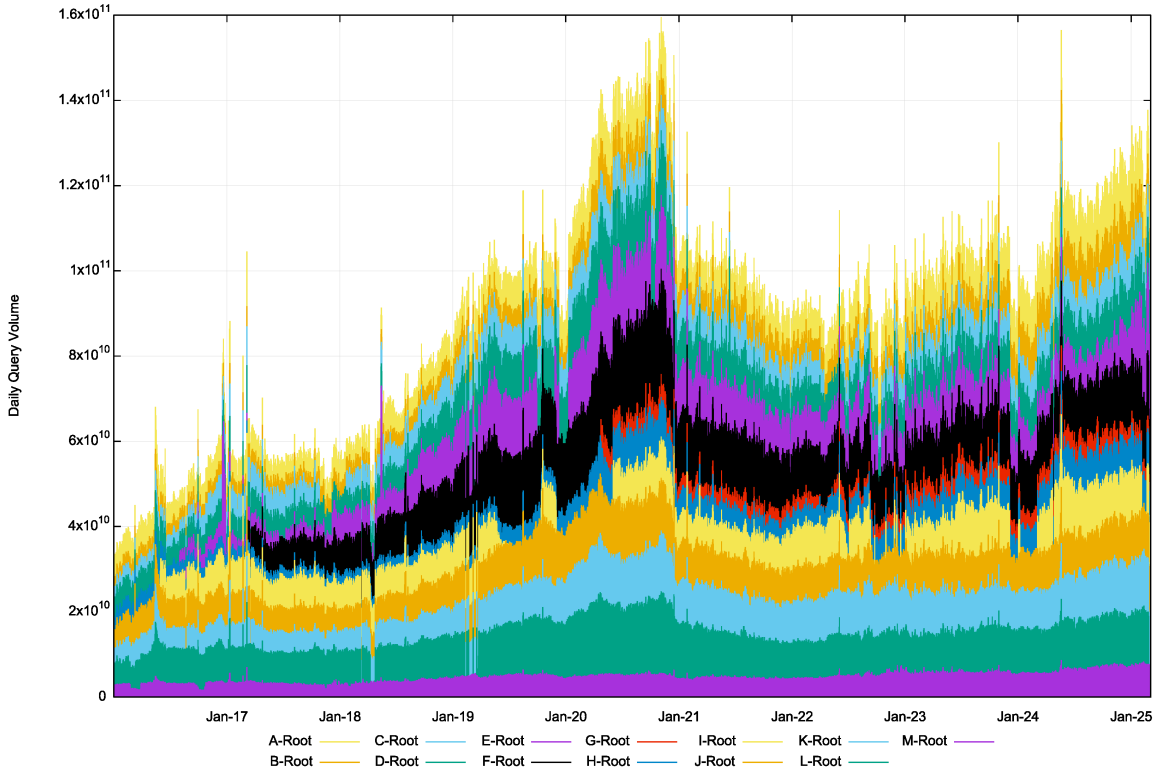
- I’m a member of a couple of community groups that are concerned with aspects of the DNS Root service, namely:
  - the Root Zone Evolution Review Committee (RZERC), which I currently chair, and
  - the Root Service Governance Working Group (RSGWG) where I am an IAB nominated member.
- The views here are purely my own views and do not necessarily reflect the working positions of either of these bodies!

The Role of the Root



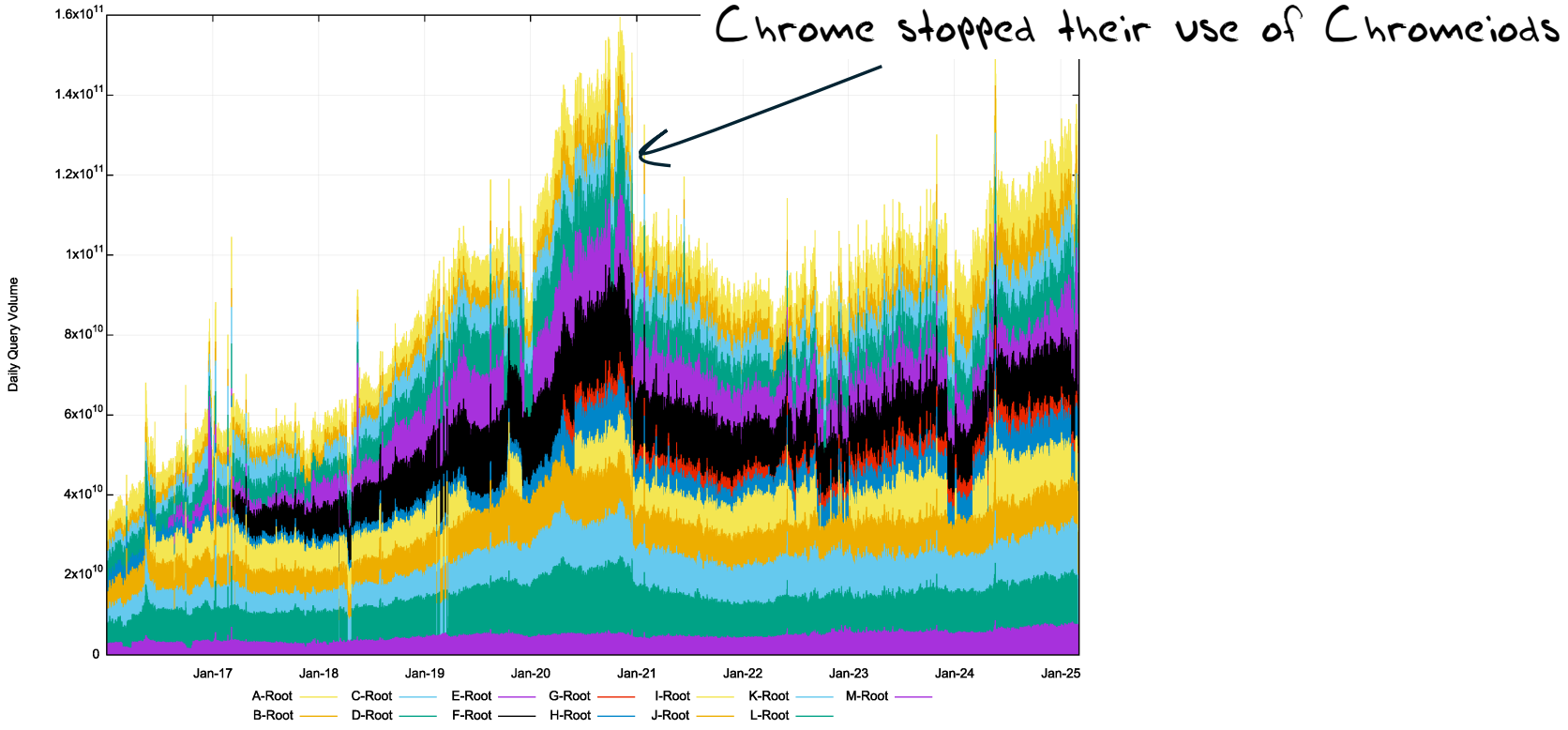
I really don’t need to explain this figure to this audience!

Root Query Load



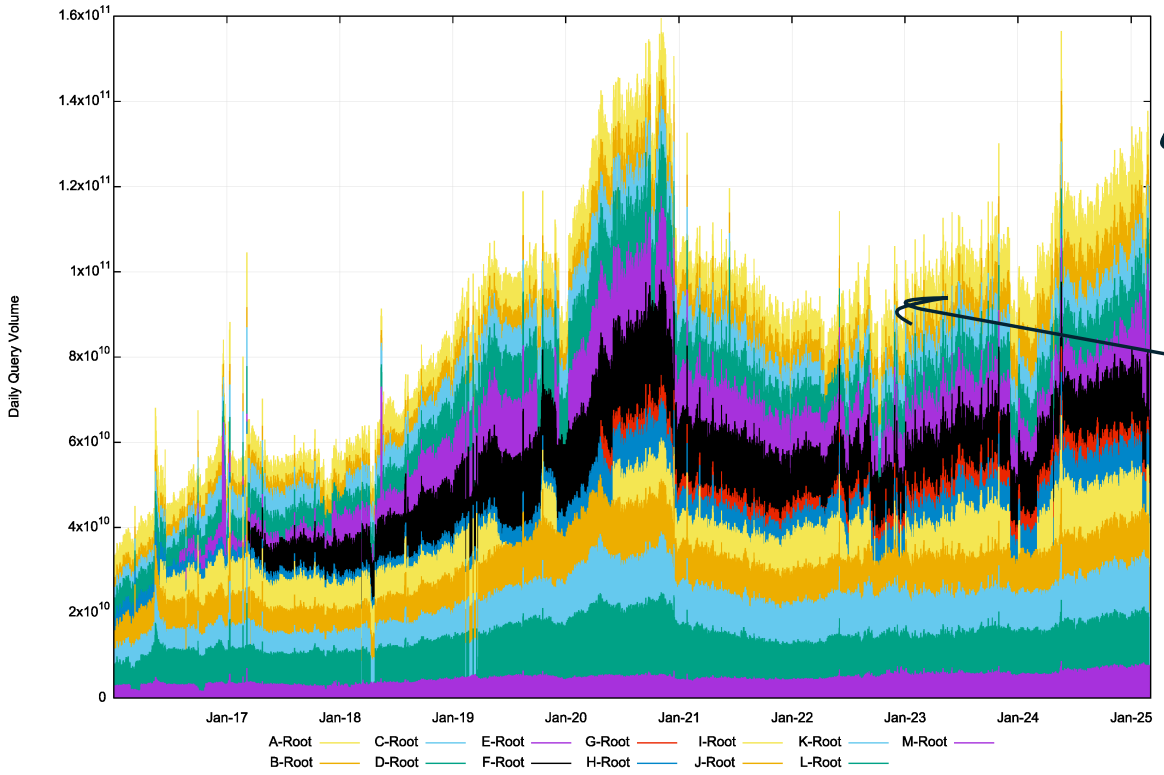
From https://github.com/rssac-caucus/RSSAC002-data

Root Query Load



From https://github.com/rssac-caucus/RSSAC002-data

Root Query Load



130B Queries per day
Jan 25

90B Queries per day
Jan 23

From https://github.com/rssac-caucus/RSSAC002-data

Root Growth

- That's a 40% growth over two years!
  - No other aspect of the Internet's common infrastructure service portfolio has grown by the same relative volume over this period
  - Indeed, many of the Internet's metrics are showing signs of market saturation
- I thought that queries to the root servers were only used to prime recursive resolvers on startup, and refresh expired cached entries
  - Which makes this growth profile challenging to explain!
- How should we respond to this inexplicable growth trend?

Aside: The Economics of the DNS

- Conventionally, when a good is consumed the consumer pays the producer a fee to compensate for the cost of production
  - Increasing consumption generates additional fees which can fund higher production volumes
- BUT DNS queries are essentially unfunded
  - ISPs bundle the cost of operation of their in-house recursive resolver into their access fee
  - No recursive resolver pays authoritative servers to answer queries about the domains that they serve
  - If there is a revenue stream, it comes from the DNS zone administrators who are paying for these nameservers to serve their zone.
  - Except for the root zone
  - Which no one pays for!

The Economics of the DNS Root

- In a market economy, a monopoly supplier of a critical resource is able to extract a monopoly rental from all others, while customers cannot seek relief through competitive offerings because of the very nature of the monopoly.
- Today's world looks to market regulators and the associated public regulatory frameworks to protect markets from such forms of abuse.
- In the Root Service function, we find a service that is both universal across the entire collection of individual public regimes, and a collective monopoly.
- A self-imposition by these operators to provide a freely offered service is perhaps not the only possible response to counter such risks of potential abuse of role, but so far, the ethos of these twelve independent root service operators has proved to be an adequate and sufficient measure to counter potential market abuse.
- But "free" is hard to scale.

The Inherent Contradiction

- How are we ensuring that the root zone service can continue to grow in capacity in response to this resumption in the growth of query rates?
- Yet factor in the apparent need to escalate the investment of resources that are in effect donated in the DNS to operate this service by this small collection of root service operators?

How can we further scale the root service?

More Named Root Servers

- Why not just expand the number of named root services from 13 to some a larger number?
  - 13 was based on a non-fragmented priming response in an IPv4 only environment
  - When we moved to dual stack operation it was no longer possible to keep the response with 512 octets
    - A priming response is 811 byes
- The Yeti exercise showed that a larger set of root nameservers was feasible

More Named Root Servers

- Most TLDs operate with 6 or 4 named nameservers. Is there a scenario when you **need** 13 named root nameservers?
  - And if you needed that many nameservers then would 14 provide more resilience? Or 15? Or more?
  - When do you call a halt?
- When do recursive resolvers simply give up?
  - Most resolvers have a “work limit” in attempting to resolve a name, and will commonly just give up after 7 – 10 seconds
- Adding more servers to the root does not necessarily add useful resilience to name resolution nor scale the service delivery capability

More Service Platforms

Root	Sites
A	59
B	6
C	13
D	220
E	328
F	359
G	6
H	12
I	85
J	148
K	131
L	123
M	23
Total	1,513

Anycast Site Counts for Root Servers, March 2025 (https://root-servers.org/)

Another option is to use the inherent parallelism that's obtained through anycast, and this has been enthusiastically embraced by the root name service operators

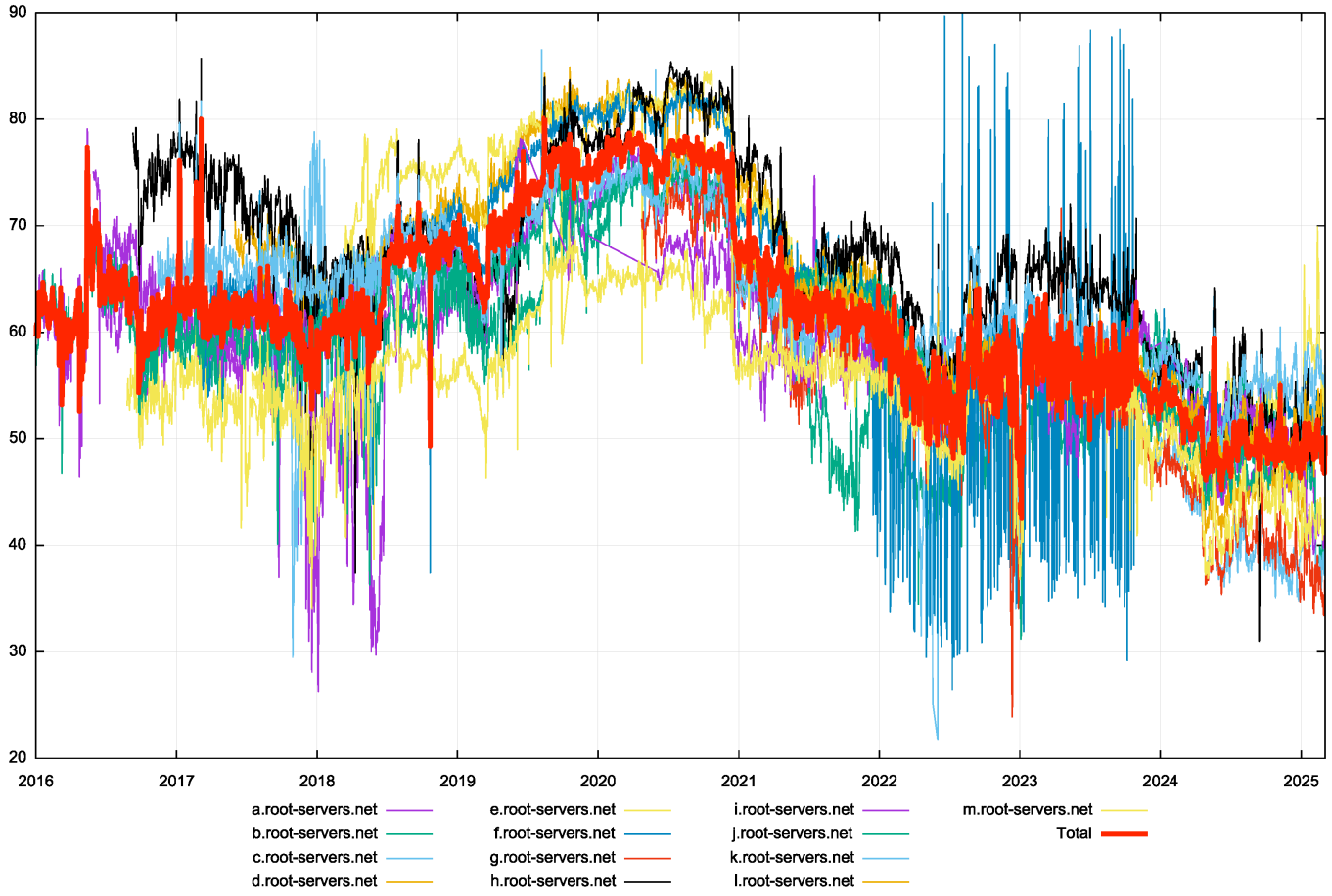
Anycast does not result in even load balancing across servers, but does increase overall system capacity and improves service resilience

More Service Platforms

- Even this anycast form of expanding the distributed root service may not be enough in the longer term.
- With a 25% compound annual query growth rate, then in four years from now we may need double the root service capacity from the current levels, and in a further four years we'll need to double it again. Exponential growth is a very harsh master.
- Can this anycast model of replicated root servers expand indefinitely?
- Or should we look elsewhere for scaling solutions?

Caching for Negative Responses

- One half of all queries to the Root Servers are answered with NXDOMAIN
- RFC8198 describes a way to cache the negative space in the root zone by leveraging the NSEC records



% of responses that are NXDOMAIN - RSSAC002 data

Are we negative caching already?

- Bind supports this function as of release 9.12.
- Unbound supports this as of release 1.7.0.
- Knot resolver supports this as of version 2.0.0.

- But the queries at the root zone keep growing despite the declining proportion of queries resulting in an NXDOMAIN response!
- How can we increase the effectiveness of local caching and reduce the query dependence on the root zone servers?

Cache the ENTIRE Root Zone

- Rather than caching individual entries from the root zone why not configure recursive resolvers to cache the ENTIRE root zone
- Its tiny - 2.2Mbytes!
- It's signed with a ZONEMD records so the resolver can validate the authenticity and currency of the root zone
- It's private – queries to the root zone don't leak beyond the recursive resolver
- This approach is documented as RFC 8806 (using AXFR - zone transfer over TCP)

Cache the ENTIRE Root Zone

- Who needs to change?
  Just recursive resolvers!

```
 // prime the server with knowledge of the root servers
zone "." {
        type mirror;
};
```

- Why not make this the default behaviour for all recursive resolvers?

- While we are at it, rather than imposing this load on the root servers, why not leverage the massive investment in CDN capability that has occurred over the past decade or so?
  - From a CDN perspective the incremental load of serving the root zone as a URL is all but invisible!

A Root Anycast URL?

- A couple of the root servers already use the services of a CDN provider to augment their own anycast platform with the CDN's capabilities
- Why not serve the Root zone as a web object and allow **any** CDN to serve the root zone as a web object?

  http://1.2.3.4/root_zone.txt

- And also, (to counter query thrashing) add a minor tweak to the root zone management practices that root zone updates will be applied no more frequently than once per 24 hours

Next Steps?

- An extension to RFC8806 allowing an HTML fetch as an alternative to DNS AXFR
- An RFC describing the rationale for defining this mode as the default mode for recursive resolvers

Discussion