

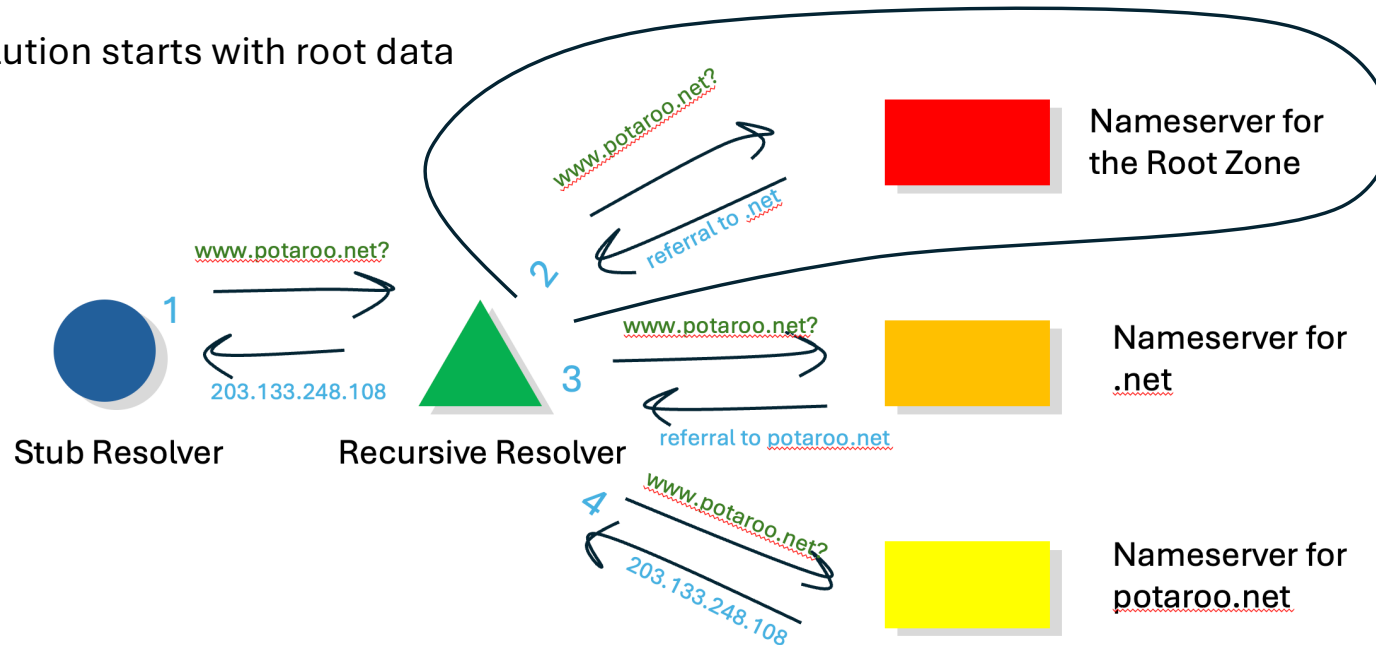
LocalRoot++

Making Localroot the default in
the DNS

Geoff Huston

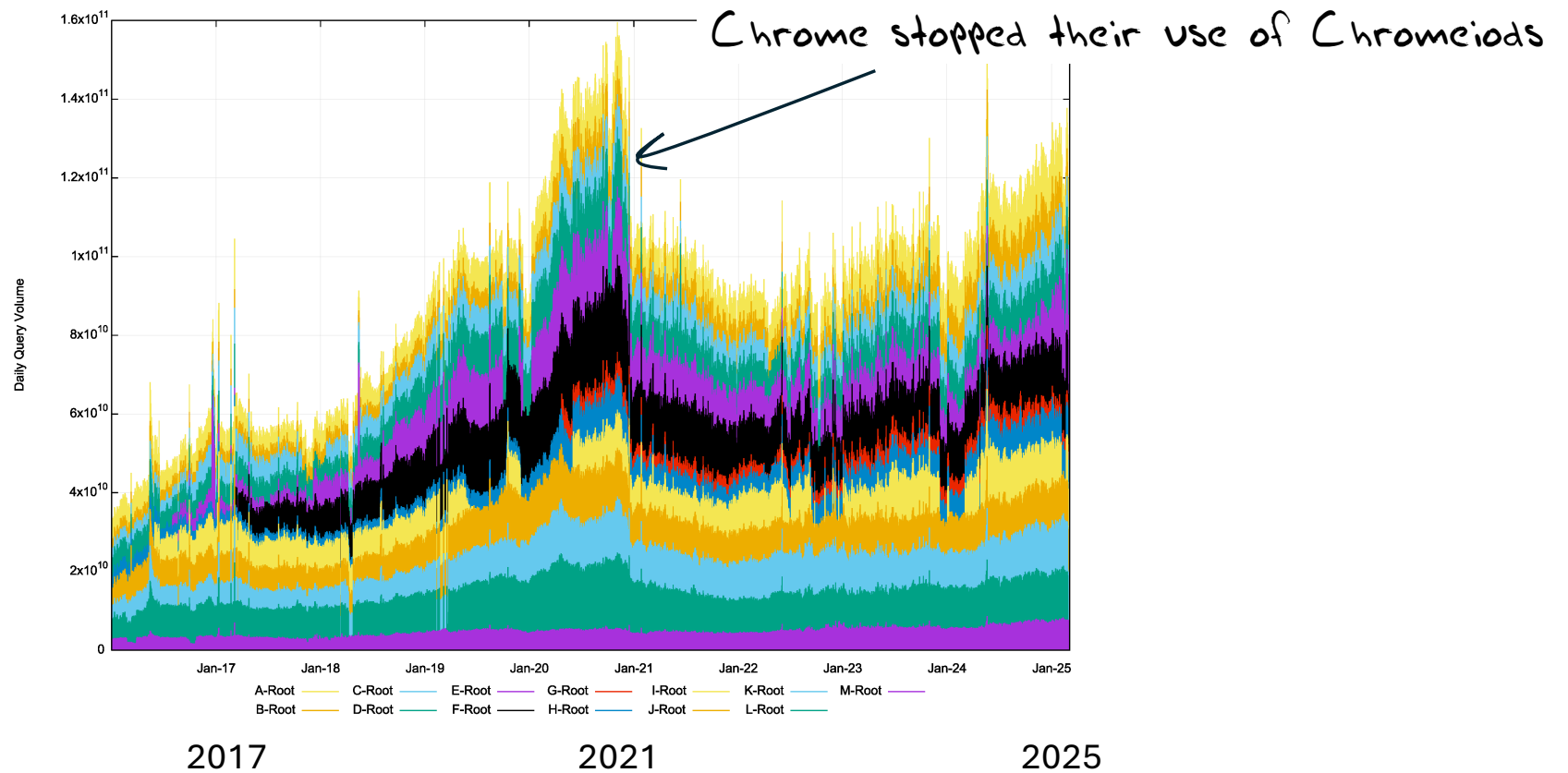
The Role of the Root in the DNS

Every DNS resolution starts with root data



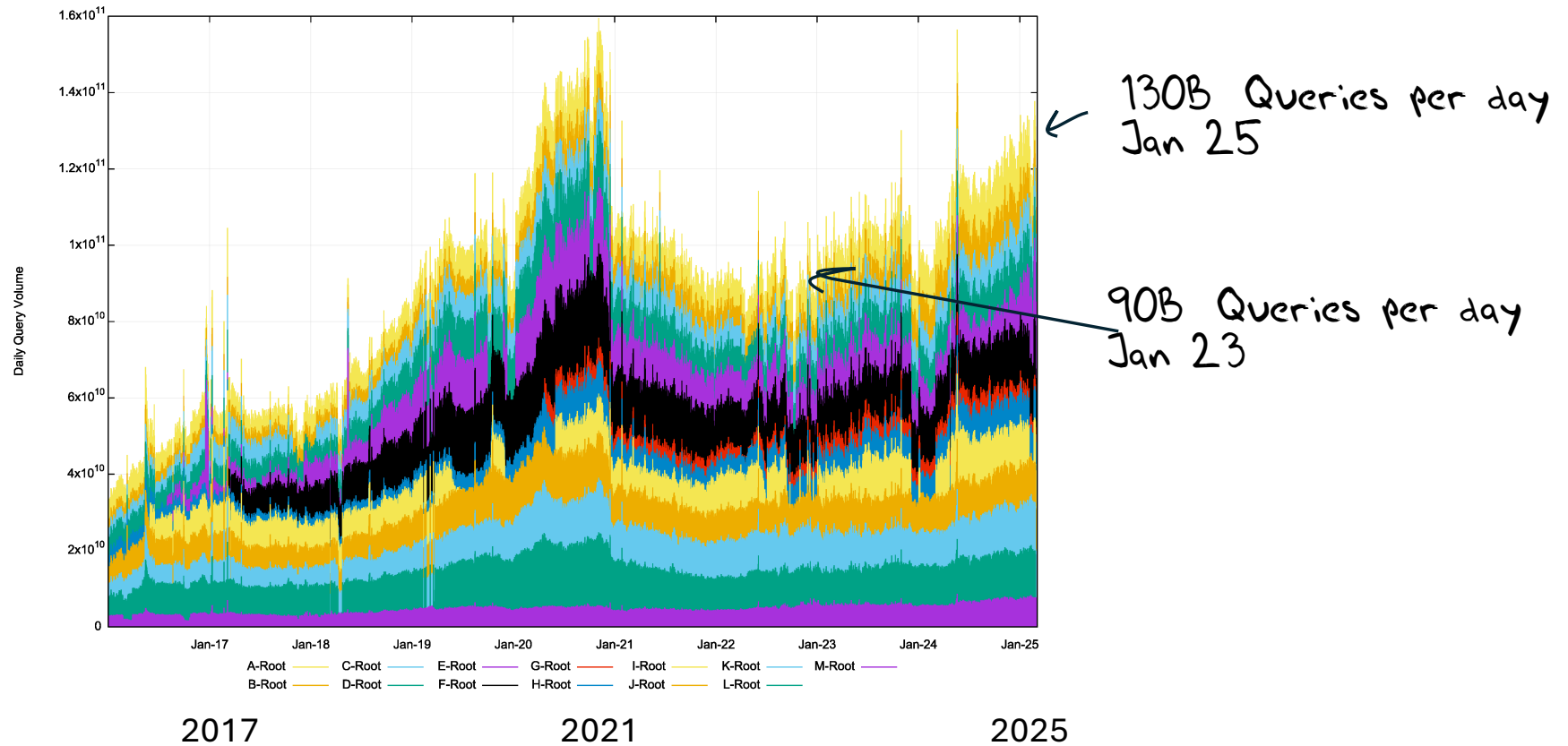
But you knew all this!

Root Query Load over time



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

Root Query Load over time



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!

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 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?
- Somehow, we need to factor in the apparent need to escalate the investment of resources that are in effect donated into 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 bytes these days
- Adding more named servers to the root does not necessarily add useful resilience nor does it add useful capacity

More Root 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

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 increases overall system capacity and improves service resilience

Anycast doesn't necessarily result in even load balancing across servers

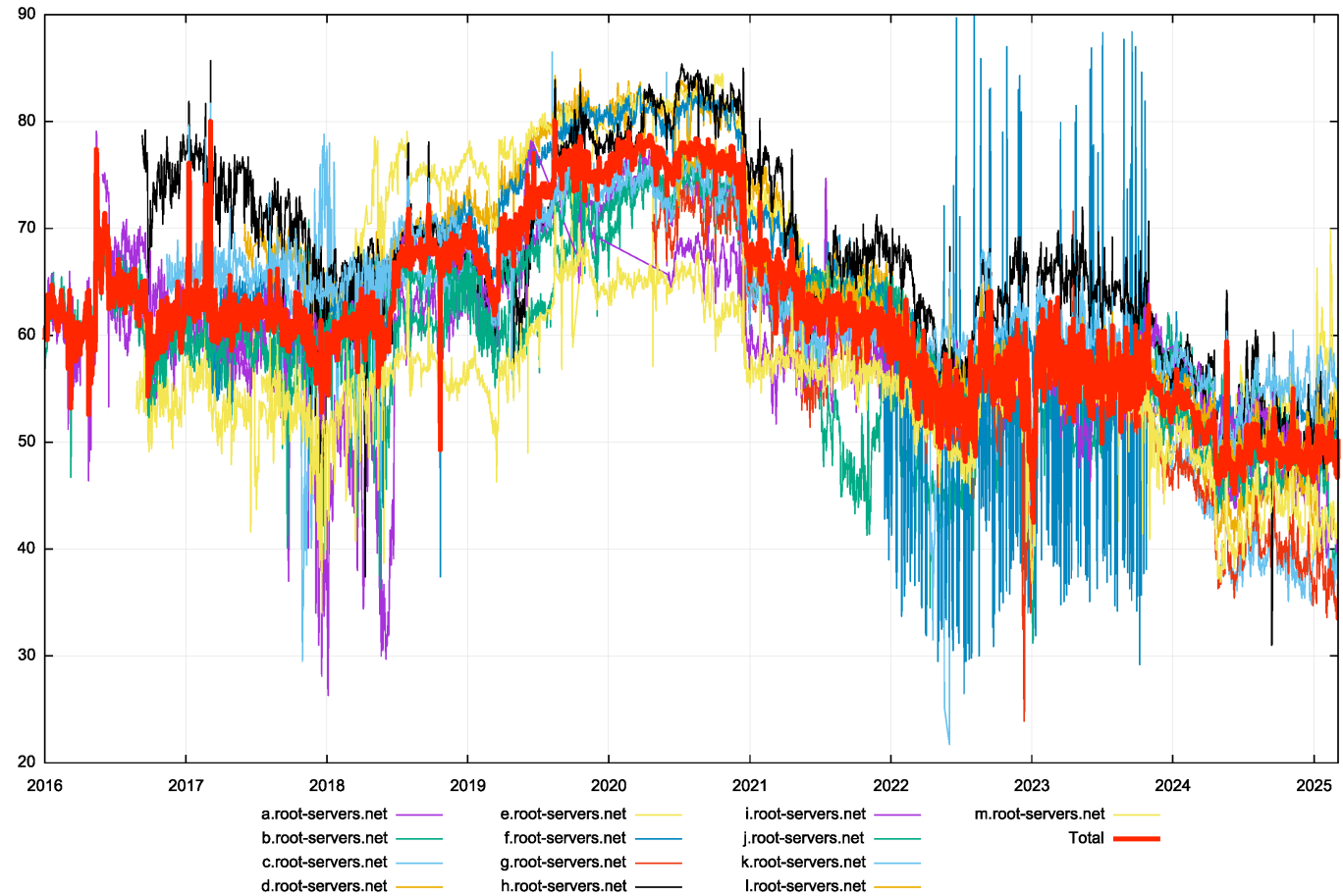
Anycast Site Counts for Root Servers, March 2025 (<https://root-servers.org/>)

More Root Service Platforms

- Even this anycast form of expanding the distributed root service may not be enough in the longer term.
- If we are facing 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 anycast replicated root servers expand indefinitely?
- Or should we look elsewhere for scaling solutions?

Caching for Negative Responses

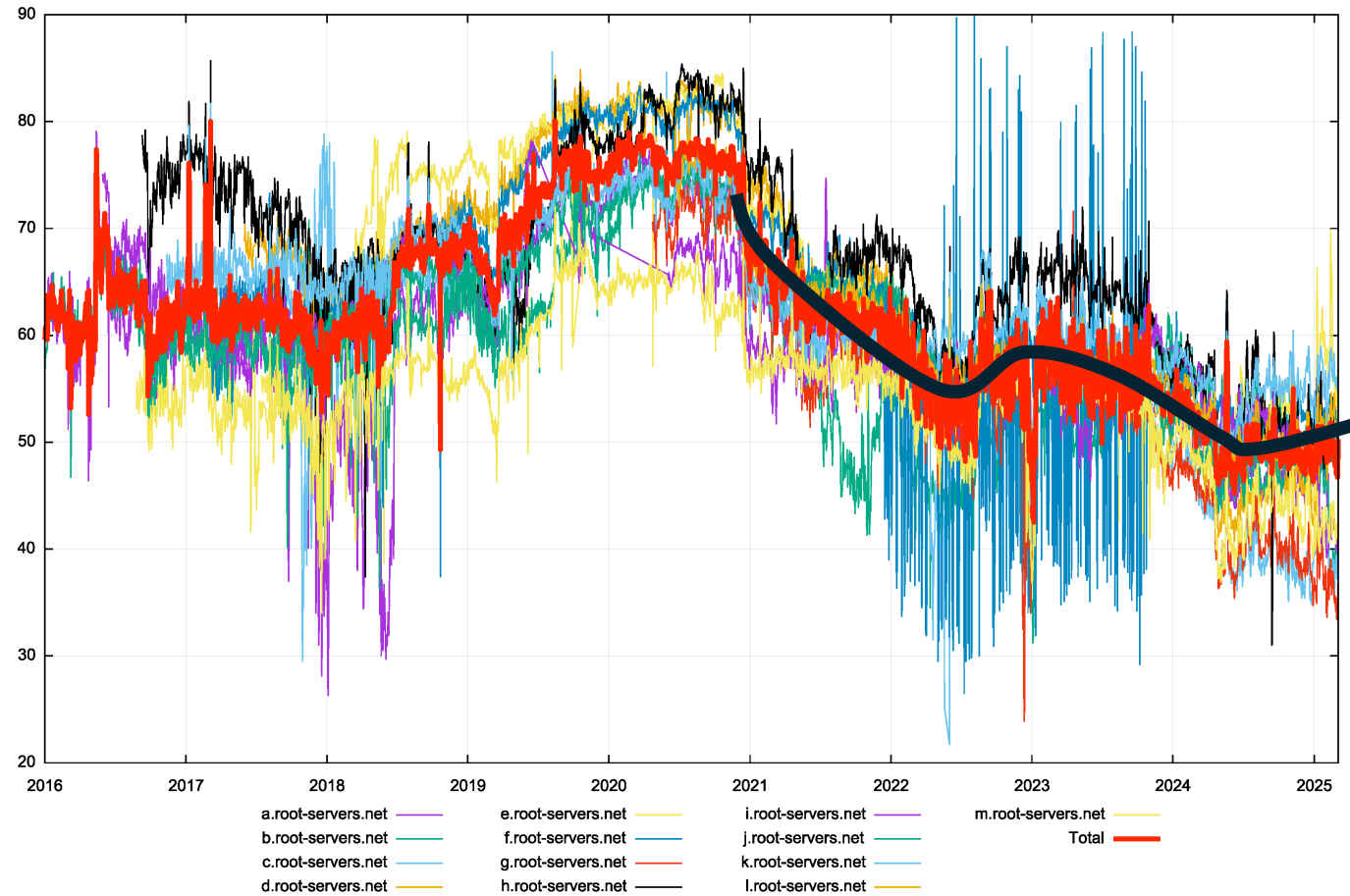
- One half of all queries to the Root Servers are answered with NXDOMAIN
- RFC8198 describes a way to cache the negative space in the root zone by leveraging the NSEC records



% of responses that are NXDOMAIN - RSSAC002 data

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

Aren't we negative caching already?

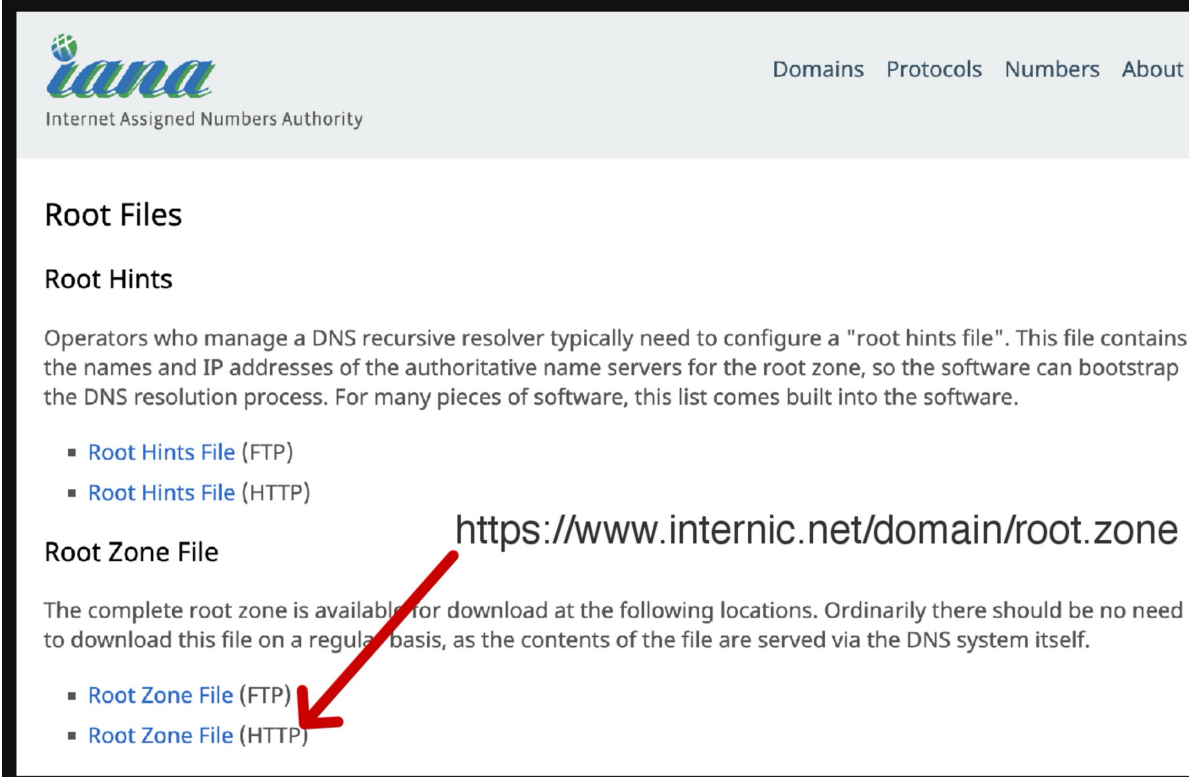
- Bind supports this function as of release 9.12.
- Unbound supports this as of release 1.7.0.
- Knot resolver supports this as of version 2.0.0.

- 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 record, 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



The screenshot shows the IANA website header with the logo and navigation links. The main content area is titled "Root Files" and "Root Hints". It explains the purpose of a "root hints file" and lists two options: "Root Hints File (FTP)" and "Root Hints File (HTTP)". Below this, the "Root Zone File" section explains that the complete root zone is available for download and lists two options: "Root Zone File (FTP)" and "Root Zone File (HTTP)". A red arrow points from the URL <https://www.internic.net/domain/root.zone> to the "Root Zone File (HTTP)" link.

Root Files

Root Hints

Operators who manage a DNS recursive resolver typically need to configure a "root hints file". This file contains the names and IP addresses of the authoritative name servers for the root zone, so the software can bootstrap the DNS resolution process. For many pieces of software, this list comes built into the software.

- [Root Hints File](#) (FTP)
- [Root Hints File](#) (HTTP)

Root Zone File

The complete root zone is available for download at the following locations. Ordinarily there should be no need to download this file on a regular basis, as the contents of the file are served via the DNS system itself.

- [Root Zone File](#) (FTP)
- [Root Zone File](#) (HTTP)

<https://www.internic.net/domain/root.zone>

Cache the ENTIRE Root Zone

How?

Bind config:

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

Knot config:

```
modules.load('prefil')
Prefill.config({ ['.'] = { url='https://www.internic.net/domain/root.zone', interval=86400 }})
```

Unbound config:

```
auth-zone:
  name: "."
  url: "https://www.internic.net/domain/root.zone"
  fallback-enabled: yes
  for-downstream: no
  for-upstream: yes
  zonefile: "root.zone"
  prefetch: yes
```

Why?

- Increased resolution speed
 - It's a local lookup every time
- Improved privacy
 - No root queries leak!
- Decreased root zone attack surface
 - Non-existent root label queries are answered directly by the resolver and not passed into the DNS
- Reduced external service dependence in the DNS

Why don't we...

- Make a prefetch of the root zone **the default action** for recursive resolvers?
- And fall back to incremental queries only when/if the prefetch fails

draft-wkumari-dns-localroot-bcp to be discussed at IETF124
proposes using a BCP to change the default behaviour of DNS
resolvers to make localroot the default action

Thanks!