Authoritative Nameserver Selection and Recursive Resolvers

Geoff Huston, Joao Damas
APNIC Labs

OARC 44, February 2025

Nameserver Questions

- When presented with multiple authoritative nameservers / multiple IP addresses how do recursive resolvers behave?
- How do we maximize performance and resilience in authoritative nameserver configurations?

Recent(ish) work

"Recursives in the Wild: Engineering Authoritative DNS Servers"
Muller, Moura, Schmidt, Heidemann, 2017

https://ant.isi.edu/~johnh/PAPERS/Mueller17b.pdf

"To meet their goals of minimizing latency and balancing load across NSes and anycast, operators need to know how recursive resolvers select an NS, and how that interacts with their NS deployments."

"... all name servers in a DNS service for a zone need to be consistently provisioned (with reasonable anycast) to provide consistent low latency to users."

Recent(ish) work

"Secure Nameserver Selection Algorithm for DNS Resolvers"
Zhang, Liu, Song, Huque, October 2024

https://datatracker.ietf.org/doc/draft-zhang-dnsop-ns-selection/

"Nameserver selection algorithms employed by DNS resolvers are not currently standardized in the DNS protocol"

The document contains an informal description of the selection algorithms used in a number of commonly used recursive resolvers (Bind 9, Unbound, Knot, PowerDNS, Microsoft DNS)

And the RFC's say:

RFC1034: “The sorting [of nameservers]… may involve statistics from past events, such as previous response times and batting averages.”

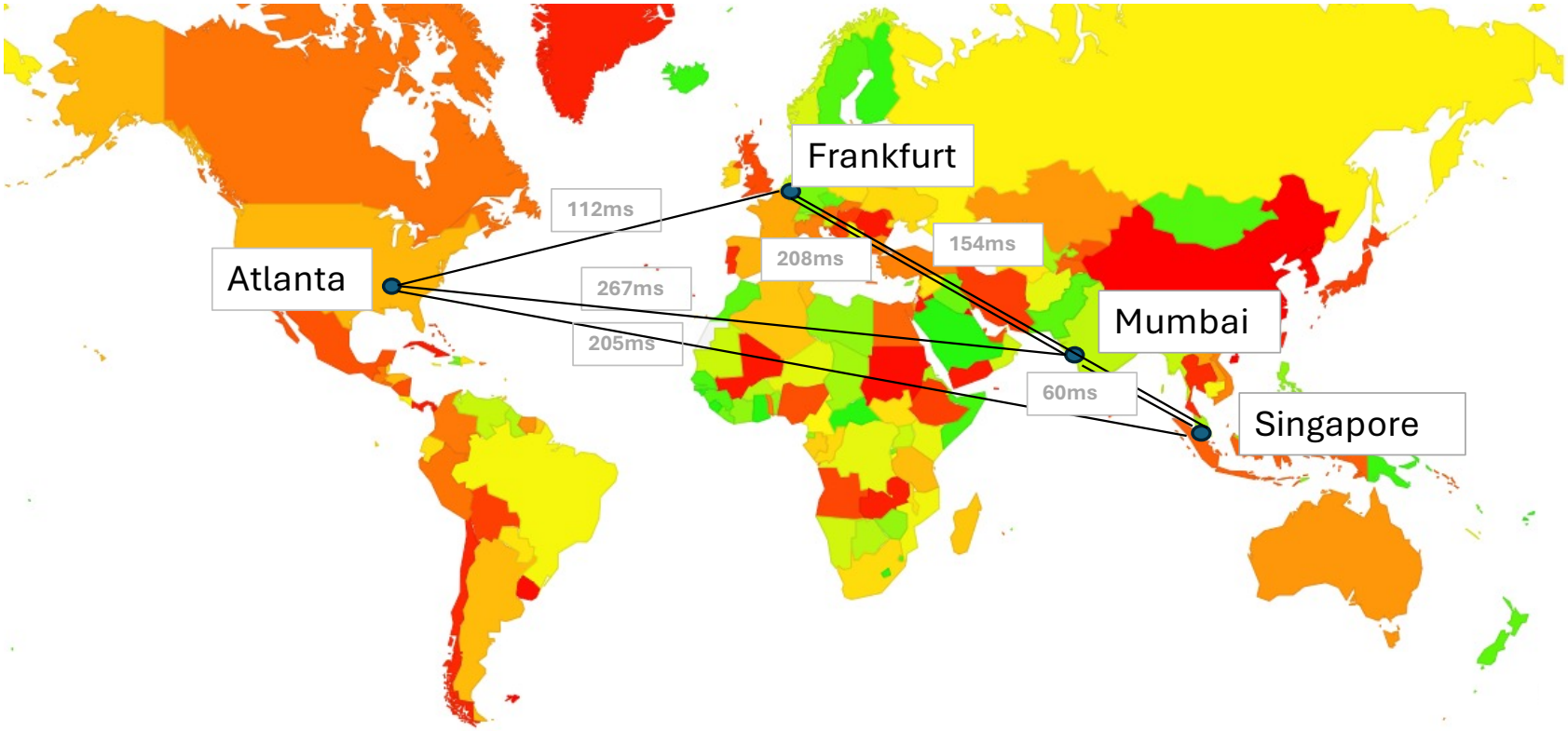
Batting averages????

Our Measurement

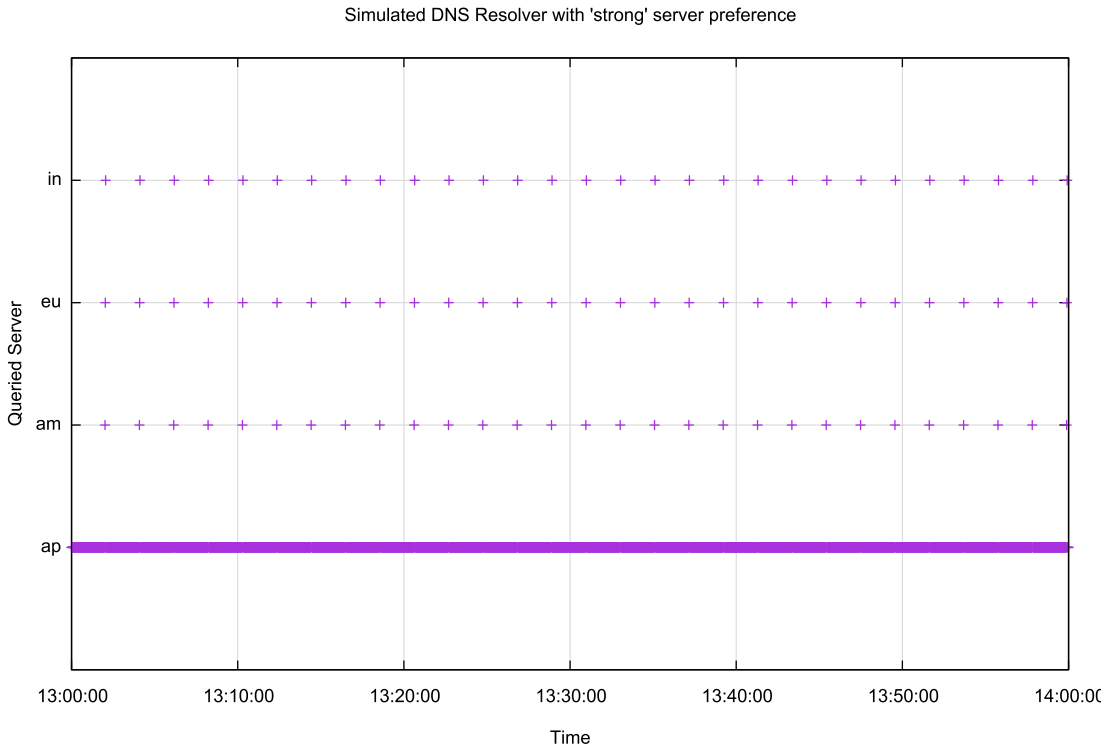
- Use a domain with four unicast authoritative nameservers
- Direct client systems to resolve unique DNS names using these nameservers
- Track queries to each nameserver from each visible recursive resolver

Our Measurement

- Use a domain with four dual-stack unicast authoritative nameservers



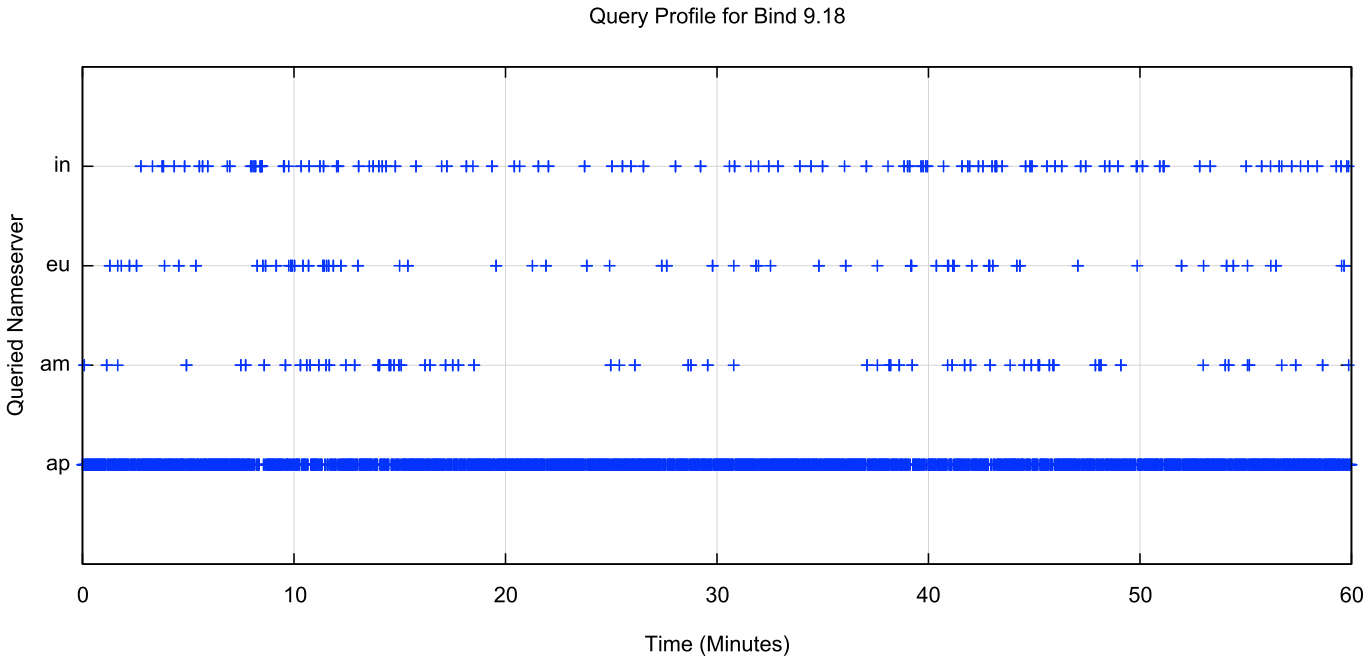
What are we expecting to see?



We anticipate seeing a “strong” preference to query the authoritative server with the lowest response time, and a regular querying of all other authoritative servers to see if their response times have changed.

Some Lab Tests: Bind 9

- Observing Bind 9 against the four nameservers using 1 query per second (test resolver is located in au)

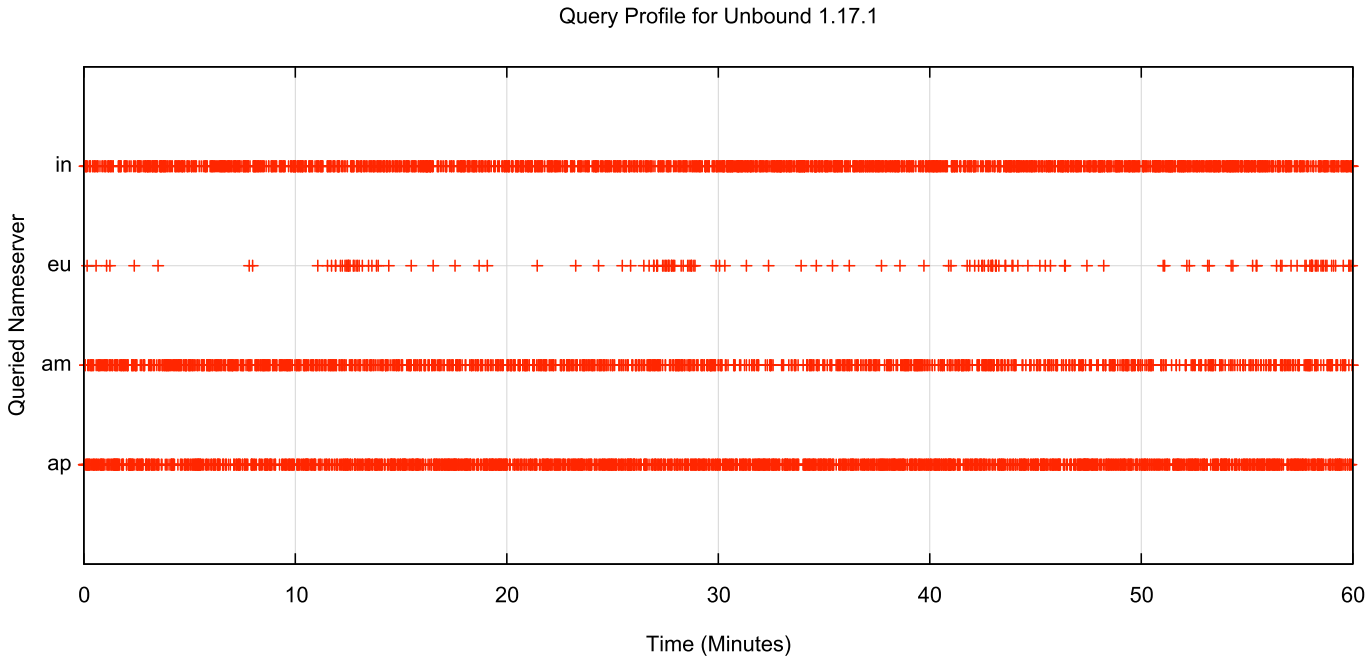


The resolver tests the other authoritative servers to ensure that its is “attached” to the fastest server.

This ‘other server’ test appears to be irregular, with a slight bias towards the second-closest server

Some Lab Tests: Unbound

- Observing Unbound against the four nameservers using 1 query per second (test resolver located in au)

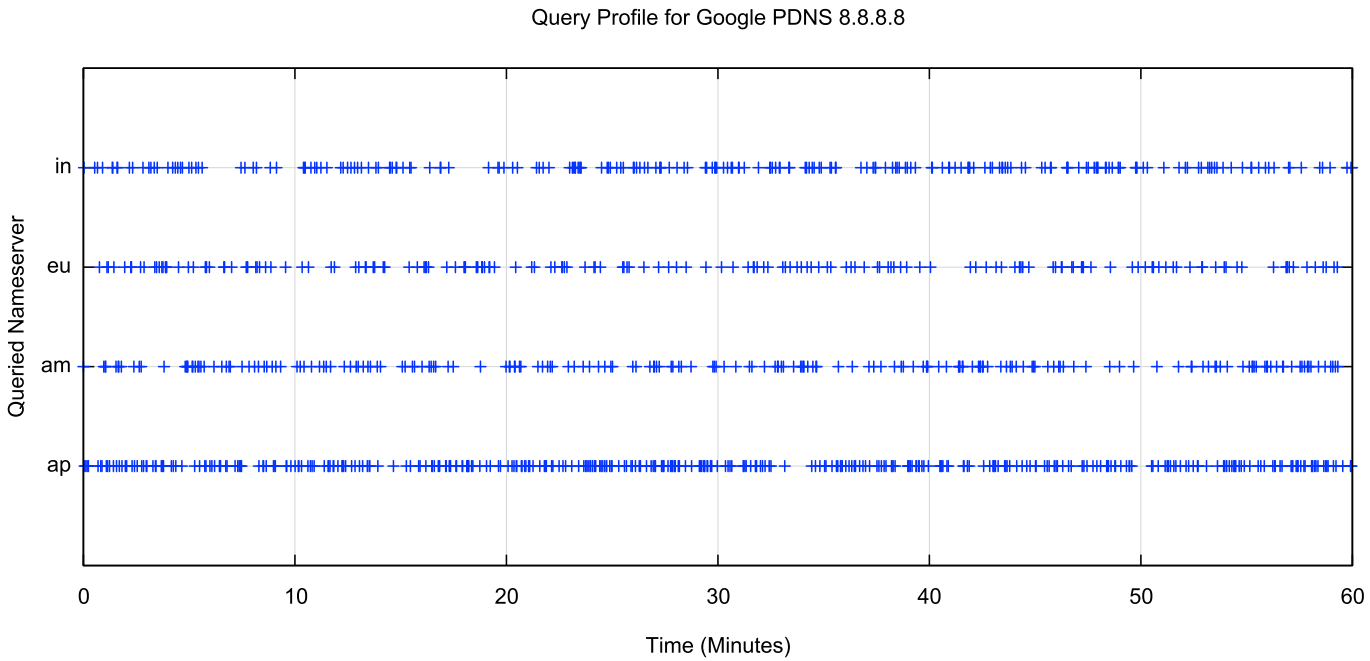


Unbound is not so clearly attached to the closest server (ap) and also queries Mumbai and Atlanta consistently.

The non-selected resolver (Frankfurt) is queried irregularly

Some Lab Tests: Google 8.8.8.8

- Observing Google 8.8.8.8 against the four nameservers using 1 query per second (stub resolver located in au)

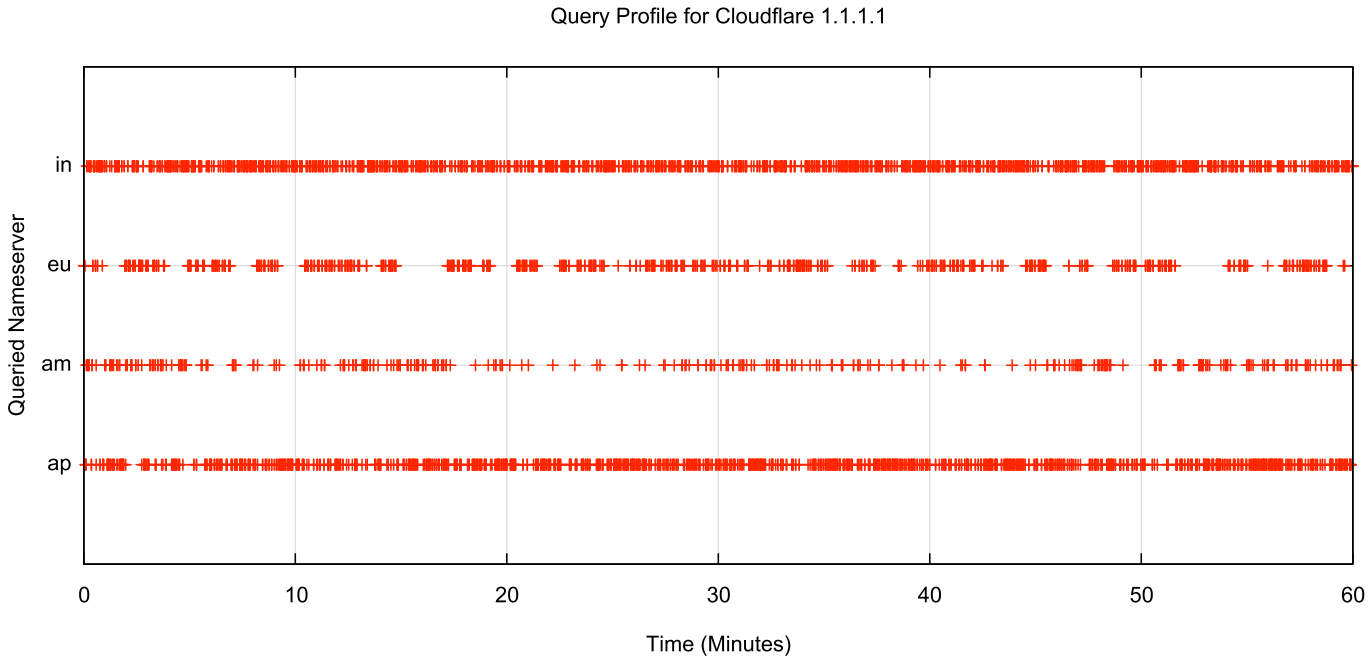


The Google resolver appears to query all instances regularly.

There appears to be a slight preference to use the Singapore (ap) server

Some Lab Tests: Cloudflare 1.1.1.1

- Observing Cloudflare's 1.1.1.1 resolver against the four nameservers using 1 query per second (stub resolver located in au)



There is a visible preference to use the servers located in Singapore (ap) and Mumbai (in)

There is no common behaviour

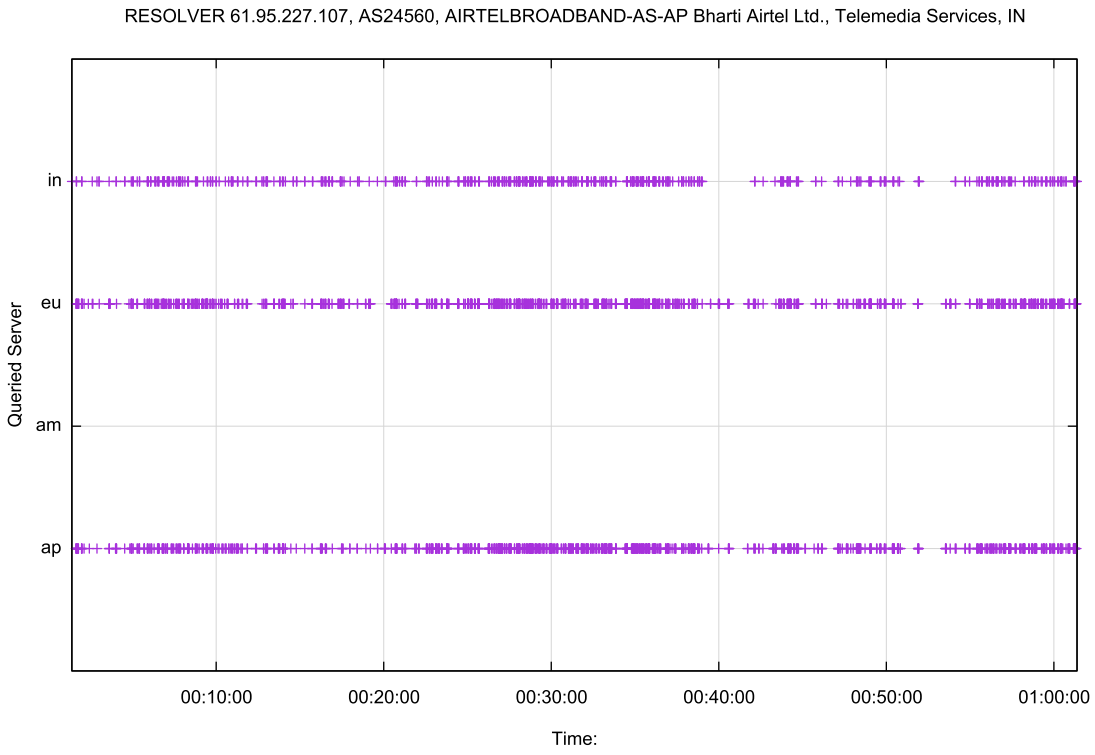
In this small sample set we are not seeing a common behaviour across recursive resolvers

- To optimize resolution performance the recursive resolver would prefer to direct all its queries to the authoritative server that responds in the shortest time
- However, the resolver would also like to track all other servers to ensure that its preferred choice of server is still optimal
- It is not clear what units of time are used compare servers' response performance, as some resolvers appear to treat a subset of servers with diverse query/response times as equivalent

Let's scale up the measurements

- We'll use an ad-based measurement platform to enrol some 25M stub resolvers per day and observe the interaction between the various recursive resolvers used by these stub resolvers and their behaviour against these four unicast authoritative servers

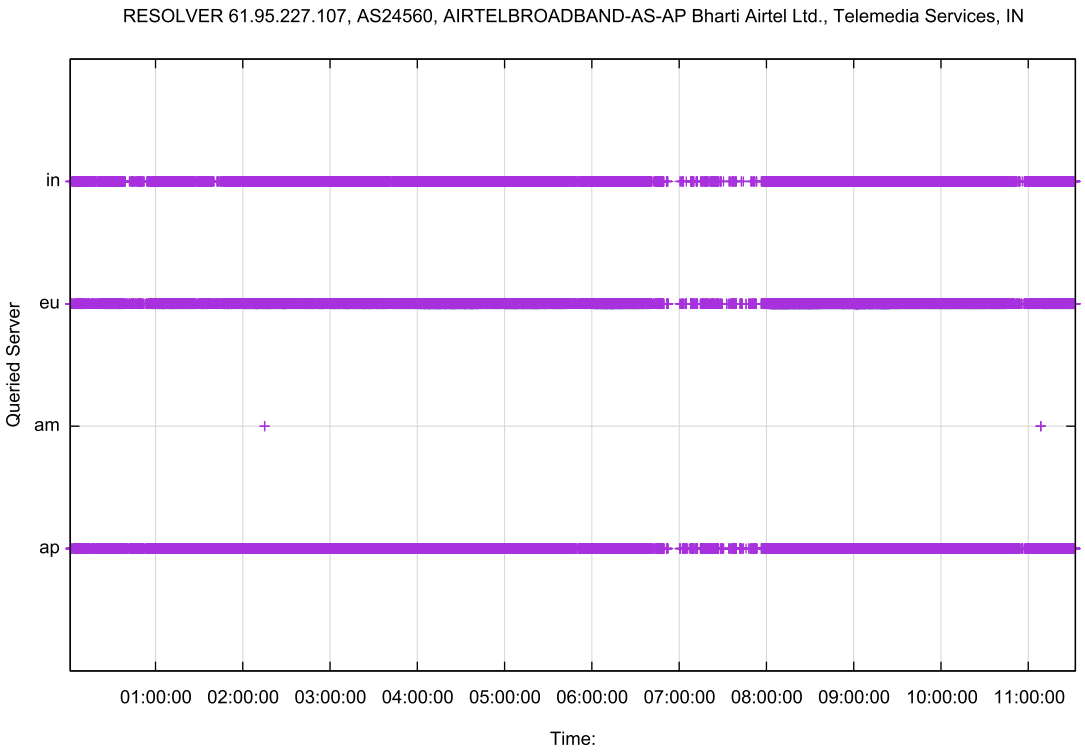
What we see - 1 Hour Profile



This is a resolver operated by Bharti Airtel in India:

- It appears to spread its query load roughly equally across the servers in Mumbai, Frankfurt and Singapore
- But none to Atlanta!

What we see - 12 Hour Profile



This resolver passed 2 queries to the server located in Atlanta just 2 times in this 12 hour period

From this graph it is not evident if there is any preference between the other three nameservers – but maybe summary numbers can be more informative

A one-week profile of this resolver

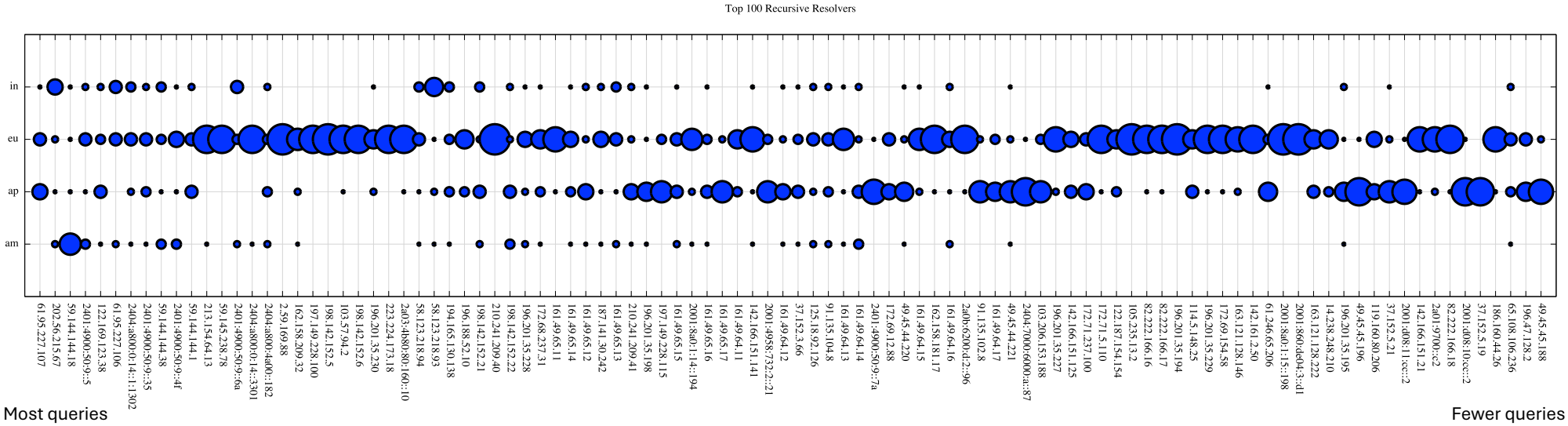
How many seconds did the resolver "latch" onto this server?

What was the longest period spent asking ONLY this server

Server	Queries	Attachment Time (secs)	Longest attached Interval (secs)
Atlanta	87	1	0
Singapore	611,992	202,012	725
Frankfurt	581,799	183,051	637
Mumbai	300,751	32,153	580

A large recursive resolver, located in India, appears to prefer distant servers that are located in Singapore and Frankfurt over a server located in Mumbai

8 Day Profile of 100 Resolvers



The 100 recursive resolvers that processed the largest number of queries in an 8-day period (1 Dec – 8 Dec)

The size of the point indicates the relative amount of time the resolver appears to “latch” onto an individual server

What proportion of Resolvers show "Attachment Preference"?

- Use the most active 1,000 recursive resolvers
- Define an "***attachment preference***" as maintaining an authoritative server selection for more than 60% of the time
  - 616 out of these 1,000 recursive resolvers show a **strong attachment** preference
- Define "***NO attachment preference***" as having the major attachment for no more than 40% of the time
  - 53 out of the 1,000 recursive resolvers show a **no attachment** preference

How "good" is this attachment preference?

- Do recursive resolvers who have a strong attachment end up attaching to the server that is “nearest” to them?

- Let’s combine ping RTT measurements with these resolver / server measurements

For example:

2a01:e00:ffff:53:2::13 is a recursive resolver operated by free.fr in France

RTT Measurements for this resolver. The *ping* measurements for this resolver are:

Atlanta – 95.3ms

Singapore - 307.5ms

Frankfurt – 9.8ms

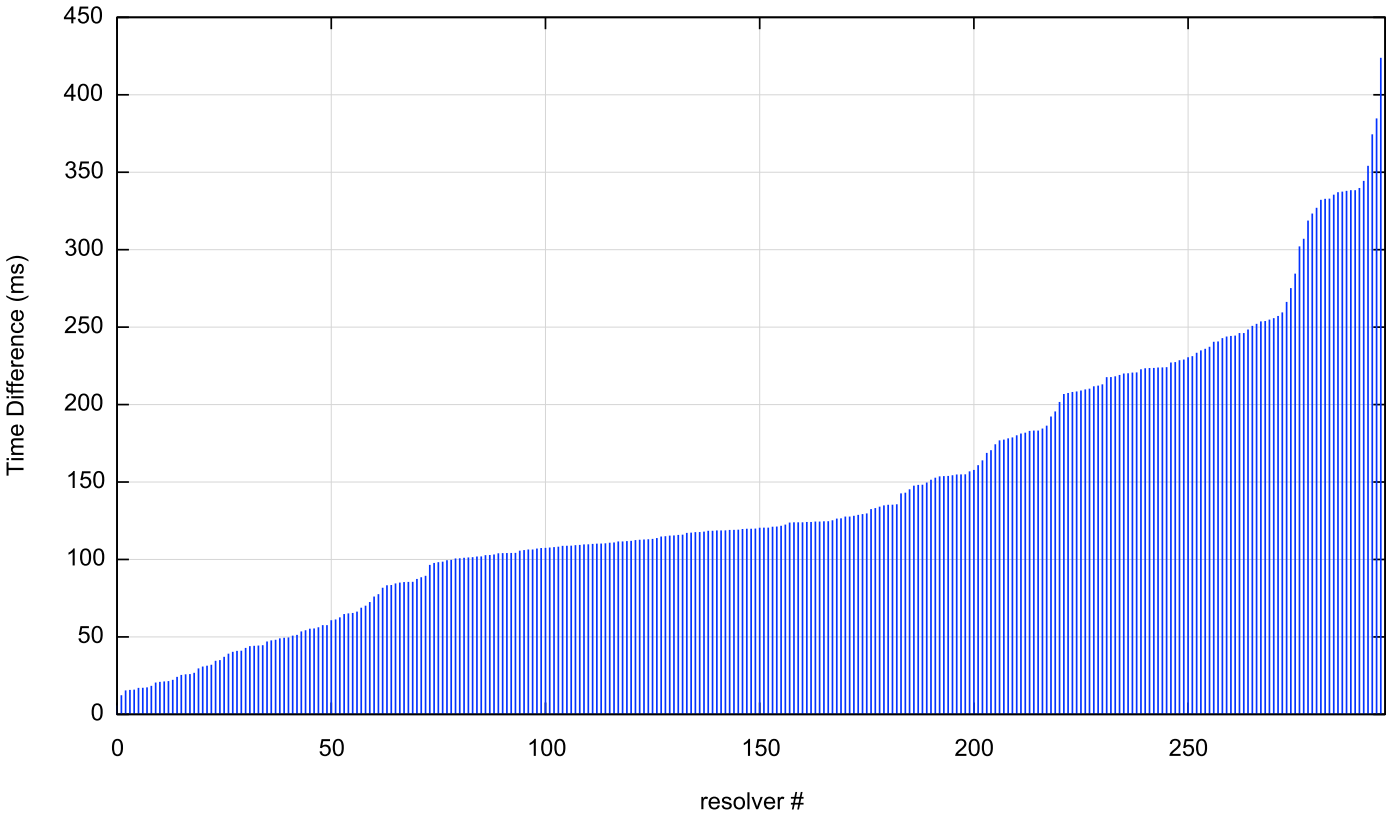
Mumbai – 241.6 ms

This resolver (located in Paris, France) was observed to query the Atlanta server 70% of the time, a resolver that was 85ms further “away” than the closest resolver (located in Frankfurt)

Results

- Of the 661 recursive resolvers that showed strong “attachment” for a single resolver (queried one server more than 60% of the time)
- 498 recursive resolvers responded to ping requests
- Of these 498 resolvers:
  - 199 resolvers chose the server that had the lowest ping time
  - The other 299 chose a more distant server
    - The average additional RTT between the chosen server and the closest server for these 299 resolvers was 142.3ms

Distribution of "Mismatch Times"



There is a strong signal of a time mismatch of <= 150ms

Is this indicative of some form of rounding of the resolvers' internal recursive-to-authoritative delay timers of to a unit of 150 ms increments?

What is this data telling us?

- You can't rely on the preference algorithm used by today's recursive resolvers to make an optimal selection across a dispersed set of unicast servers that will select the fastest authoritative server
  - i.e. even if you do a great job of deploying diverse unicast nameservers, recursive resolvers will likely muck it up and make sub-optimal selections of their "preferred" nameserver!

How should you deploy a set of nameservers?

Assuming that you want to optimize both performance and availability…

- A distributed collection of unicast nameservers is **not** going to give the best possible result
  - Many recursive resolvers are not only poor at latching on to the fastest nameserver, but they latch onto a more distant nameserver, giving a worse resolution performance for non-cached name resolution
- Perhaps a better approach is to use anycast nameservers
- What do other domains do for their nameserver configuration?

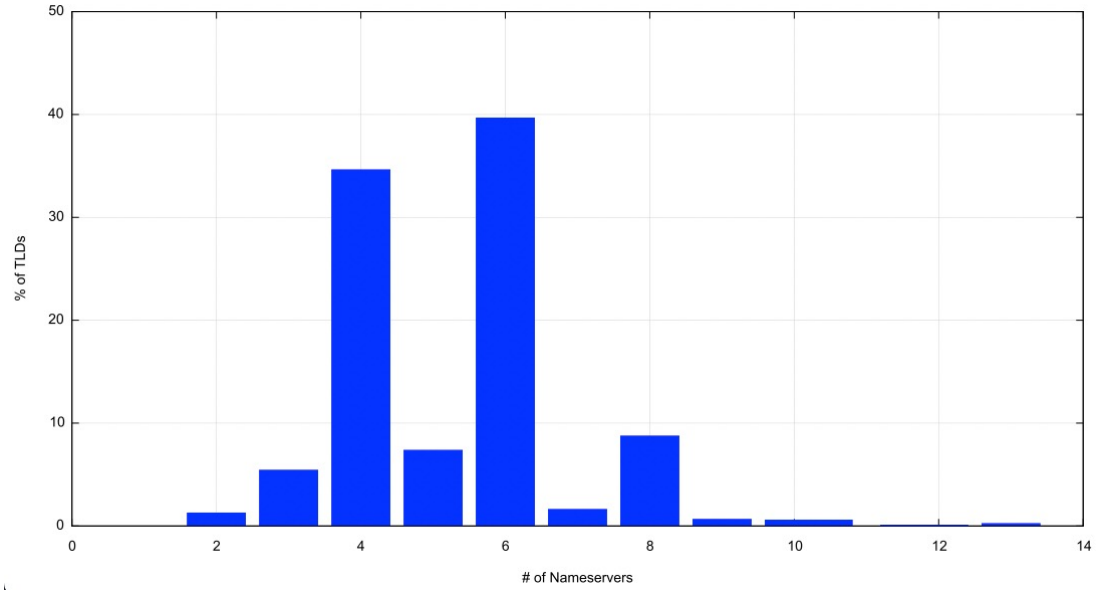
Let's look the Root Zone

- A small set of relatively intensely used domains whose performance and reliability is (supposedly) critical

Profile of the Use of Nameservers

Root Zone:

- TLDs: 1,445
- Authoritative Nameservers: 5,998
- Average Nameservers per TLD: 4.2
- Distribution of Nameservers per TLD show strong preference for 4 or 6 nameserver names
- Dual Stack Nameservers: 5,687
- IPv4-only Nameservers: 321
- IPv6-only Nameservers: 3



Unicast vs Anycast - Root TLD Nameservers

How many of these IP addresses are carried in an anycast cloud?

Unicast vs Anycast

How many of these IP addresses are carried in an anycast cloud?

- Query the IP address from a diverse set of locations(Atlanta, Frankfurt, Sao Paulo, Singapore, Australia)
- If the Name Server ID (NSID) is constant when queried from a diverse set of queries, then its reasonable to assume that the server's IP address is **not** part of an anycast cloud (as we anticipate that different anycast instances will give a different NSID response)
- If the variance of DNS query times is not sufficiently large, then its reasonable to assume that the server's IP address is part of an anycast service cloud.

Unicast vs Anycast - Root TLD Nameservers

- Unique IP addresses of nameservers: 9,014

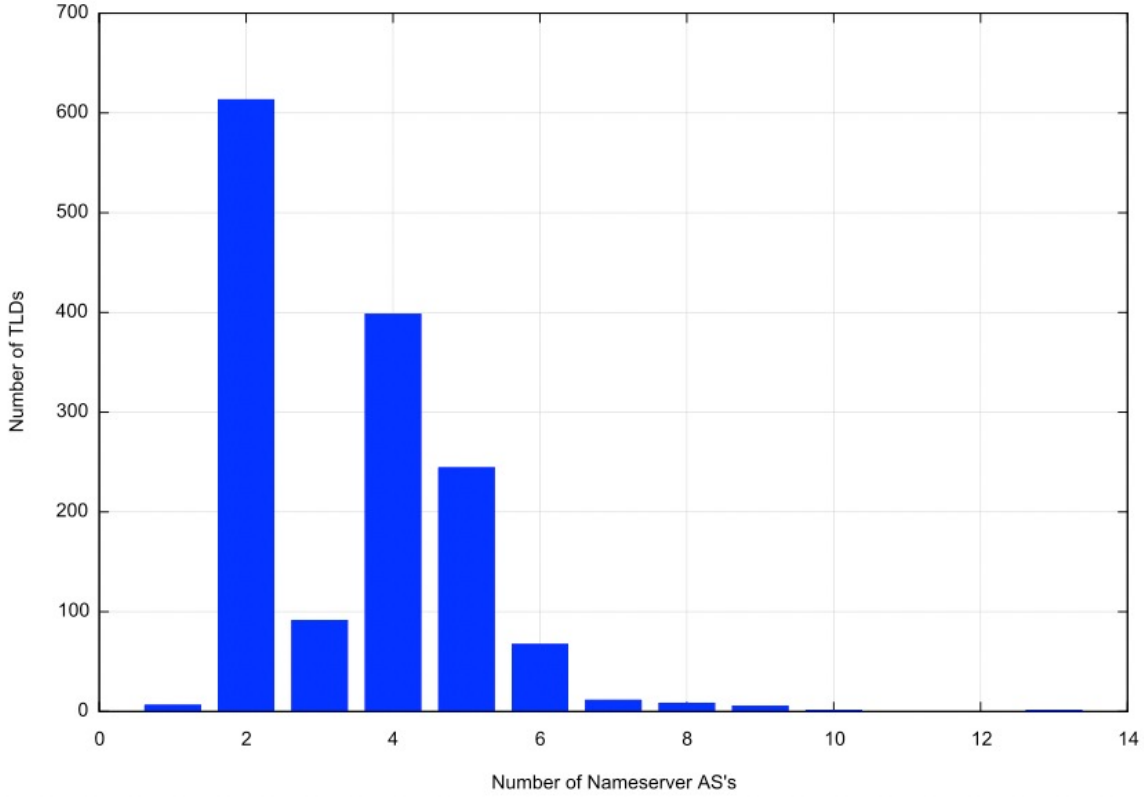
- Unicast IP addresses: 587
- “Limited” Anycast addresses: 5,868 (rtt variance > 150ms)
- **“Diverse” Anycast IP addresses: 2,559** (rtt variance < 150ms)

Unicast vs Anycast

- TLDs served by only unicast nameservers: 8
- TLDs served by a mix of unicast and anycast servers: 378
- TLDs served by anycast servers only: 1,067
  - TLDs served only by diverse anycast: 289
  - TLDs served only by limited anycast: 202
  - TLDs served by limited and diverse anycast: 576

AS Diversity

- In the root zone there are just 6 TLDs where all the zone's nameservers are held in a single AS (and only 1 in a diverse anycast configuration)
- On average each TLD has 10.1 distinct IP addresses of nameservers
- On average each TLD has nameservers located in 3.4 origin Ases



Observations (1/3)

- DNS recursive resolvers **do not**, on average, make an accurate selection of the “fastest” nameserver
  - Which negates any potential performance benefits of a nameserver deployment approach of a geographically diverse collection of unicast nameservers
  - Many recursive resolvers appear to use a timer with a granularity of around 150ms to select the “fastest” nameserver

Observations (2/3)

- Anycast nameserver deployments can produce better outcomes, but the effectiveness of this approach depends on the density of the anycast constellation
  - Limited density anycast constellations may produce sub-optimal outcomes for some clients

Observations (3/3)

Resilience can be provided through the use of multiple anycast service platforms

- How many such platforms is “optimal” is an open question
- More is not necessarily incrementally better, which leads to a suggestion of the use of 2 or 3 diverse anycast platforms, depending on the level of failover resilience you are after

BUT - DNSSEC

- The combination of multiple diverse nameserver providers, a DNSSEC-signed domain name, and online signing introduces some operational fragility, particularly when managing key rollover and algorithm rolls
- This is still a not well charted space and operational incidents that impact TLD availability still occur

Questions?