December 2024
Geoff Huston

DNS Nameservers

It is common folklore in the Domain Name System that a delegated domain name must be served by 2 or more nameservers. The logic for this is based in a desire for service resilience. If one server is unreachable then hopefully the other is not, and recursive resolvers when presented with a list two or more nameservers for a domain will work through this list until it queries a nameserver that is responsive.

This simple description raises a couple of questions. Firstly, when presented with a list of nameservers for a domain how do recursive resolvers respond? Do they send queries to all of the nameservers at once? Or do they serialise their actions in looking for a responsive nameserver? Secondly, if these queries are serialised, then how can a domain administrator organise the zone's nameservers to maximise both DNS resolution performance and service resilience?

Recent Work

To assist with answering out questions there are a couple of recent(ish) papers that address parts of this topic. The first paper dates from 2017, "Recursives in the wild: Engineering Authoritative DNS Servers", by a group of researchers at SIDN Lans and USC's Information Sciences Institute (https://ant.isi.edu/~johnh/PAPERS/Mueller17b.pdf). They motivate their study by observing that "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." By the time that this work was undertaken the use of anycast in the DNS was well-established operational practice, and the conclusion from the paper is that: "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."

There is a current Internet Draft document, "Secure Nameserver Selection Algorithm for DNS Resolvers" by researchers from Tsinghua University, Alibaba Cloud and Salesforce that focuses on the behaviour of recursive resolvers (https://datatracker.ietf.org/doc/draft-zhang-dnsop-ns-selection/). This draft provides a description of the nameserver selection algorithms used in a number of commonly used recursive resolvers (Bind 9, Unbound, Knot, PowerDNS, Microsoft DNS). This document also notes that: "Nameserver selection algorithms employed by DNS resolvers are not currently standardized in the DNS protocol".

This is the case. The only reference to nameserver selection int he RFC series is in RFC 1034, which has the advice that "The sorting [of nameservers] … may involve statistics from past events, such as previous response times and batting averages." (RFC 1034, "Domain Names – Concepts and Facilities, Paul Mockapetris, 1987). Yes, the term "batting average" is used here, without further elucidation. Presumably it means some record of responsiveness to respond to queries, although other interpretations are equally possible!

Some Simple Tests of Resilient Resolver Behaviour

How do recursive resolvers behave where attempting to resolve a domain name served by multiple nameservers where none of the nameservers are responsive? Table 1 shows the behaviour of the **Bind 9** resolver when presented with a variable collection of IPv4-only nameservers.

Servers	Queries	Duration (secs)	Servers Queried
1	3	7.2	1
2	6	8.1	2
3	8	10.0	3
4	7	10.0	4
5	7	8.8	5
6	8	9.6	6
7	7	8.8	6
8	7	8.8	6
9	7	8.8	6
10	7	8.8	6
11	8	9.6	6
12	7	9.2	6
13	8	10.0	7

Table 1 - Bind 9 Query behaviour with unresponsive IPv4-only nameservers

In the case of a single non-responsive nameserver, the resolver will halt after 3 queries, taking a total of 7.2 seconds. For more servers the resolver appears to bring resolution to a halt after some 9 - 10 seconds. The resolver appears to query 6 nameservers, except for the case when there were 13 nameservers, when 7 nameservers were queried. In each case the resolver was "cold started" with a clear cache.

Does the picture change when using unresponsive dual stacked nameservers? Table 2 shows the outcome of this test, and the results are largely the same.

Servers	V4 Queries	V6 Queries	Queries	Duration (secs)	Servers Queried
1	3	3	6	8.0	1
2	4	3	7	8.1	2
3	4	4	8	9.6	3
4	5	3	8	9.7	4
5	3	4	7	8.8	5
6	3	5	8	9.6	6
7	3	6	9	8.8	5
8	2	6	8	8.8	5
9	3	4	7	8.8	6
10	4	5	9	10.0	6
11	3	4	7	8.8	6
12	4	4	8	9.6	6
13	5	7	12	10.0	7

Table 2 - Bind 9 Query behaviour with unresponsive Dual Stack nameservers

The conclusion to be drawn from this measurement is that there is little to be gained in terms of resilience in using more than 6 nameservers when using Bind 9 as a recursive resolver.

Each recursive resolver implementation has chosen to implement failover and resilience in a different manner. Tables 3 and 4 show the results of the same experiment with unresponsive nameservers, using the **Unbound** recursive resolver.

Servers	Queries	Duration (secs)	Servers Queried
1	11	46.6	1
2	21	55.4	2
3	37	331.7	3
4	52	327.7	4
5	65	428.4	5
6	72	360.0	6

7	21	131.6	2
8	63	534.5	6
9	107	533.5	9
10	110	230.4	10
11	109	395.6	11
12	135	541.6	12
13	124	188.8	13

Table 3 - unbound Query behaviour with unresponsive IPv4-only nameservers

Servers	V4 Queries	V6 Queries	Queries	Duration (secs)	Servers Queried
1	10	11	21	84.2	1
2	13	11	24	229.1	2
3	33	12	45	336.4	3
4	28	16	44	391.2	4
5	61	51	112	735.5	5
6	57	61	118	585.3	6
7	63	43	106	258.5	6
8	71	78	149	1,577.0	7
9	67	47	114	177.5	8
10	66	48	114	717.8	8
11	79	58	137	203.6	9
12	67	62	129	223.1	8
13	67	62	129	246.5	9

Table 4 - unbound Query behaviour with unresponsive Dual Stack nameservers

Clearly unbound is a far more persistent algorithm, spacing out its queries in increasing intervals (increasing in doubling steps with 3, 6, 12 and 24 second between queries) for a period of some minutes. In general, stub resolvers typically stop waiting for a response from a recursive resolver after 10 seconds, so it is not exactly clear of the incremental benefit in having the recursive resolver persist for many more minutes. It strikes me as a measure that has little, if any, positive features. The original querier has given up after 10 seconds, so no client is waiting for a response, and the additional queries to unresponsive nameservers appear to simply add to the level of DNS background noise.

There is also the additional worrisome issue of DOS amplification where a single query from a stub resolver to an unbound recursive resolver can generate over 100 queries. Is this age of acute sensitivity to various forms of denial-of-service vulnerabilities, this behaviour pattern appears to be far more of a harmful liability than a sign of virtuous persistence! I have no idea why the Unbound support folk still have this behaviour in their implementation. Bind 9's approach of abandoning a resolution process after some 9 seconds appears to match stub resolver behaviour and shows an adequate level of persistence in attempting to deal with unresponsive nameservers.

Large-Scale Measurements of Recursive Resolver Behaviour

Multiple nameservers are intended to provide both resilience and improved performance. There appears to be a general expectation that recursive resolvers have implemented some interpretation of the general objectives of nameserver selection and will attempt to prefer querying the recursive resolver with the fastest response. The following descriptions of recursive resolver nameserver selection is drawn from a current Internet Draft document, "Secure Nameserver Selection Algorithm for DNS Resolvers" by researchers from Tsinghua University, Alibaba Cloud and Salesforce (https://datatracker.ietf.org/doc/draft-zhang-dnsop-ns-selection/).

Bind 9 uses an estimate of the smoothed Round Trip Time (SRTT) for each nameserver, and selects the nameserver with the lowest SRTT for the next query. Nameservers that have not been queried are assigned an initial SRTT of between 1 to 32ms. Also, all nameservers that are not selected have their

SRTT values decreased. Over time such nameservers will be selected and queried, and the SRTT value is reset to the actual query/response time as a result.

Unbound uses a different strategy. The resolver randomly selects from all candidate nameservers, excluding a nameserver from this candidate set only when its SRTT exceeds the fastest one by more than 400ms.

The Knot resolver attempts to balance a strong preference to use the closest nameserver, and a need to check on all other nameservers to see if the reachability of any of these nameservers has changed. Some 95% of cases a Knot resolver will query the lowest latency server and randomly choose from the other servers some 5% of cases.

PowerDNS applies a "decay" function to the SRTT of each nameserver during each query, ensuring that with more cases the unused nameserver's SRTT will decay to the point when it is the most preferred/

Let's put this theory to the text by using a large-scale measurement of the interaction between recursive resolvers and authoritative servers. We will enrol recursive resolvers by using an advertisement-based measurement platform, presenting the measurement system with some 20M – 30M individual measurement cases per day. The measurement script presents the user's system with a unique DNS name (so that caching cannot influence the results), where the name is drawn from a uniquely-named domain that is served by four dual-stack unicast geographically dispersed nameservers.

The locations of each of these nameservers, and the RTT path times between each pair of these servers is shown in Figure 1.

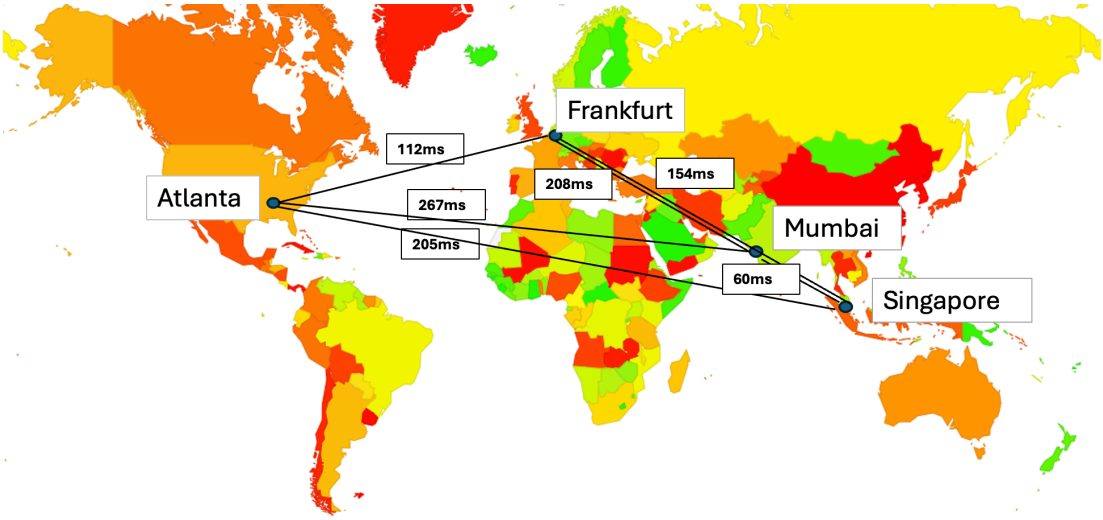

Figure 1 – Placement of unicast nameservers for the resolver measurement

What we expect to see if recursive resolvers behave with a balance of performance and resilience would be for the resolver to have a strong "attachment" to the nameserver that has the fastest DNS response times, and a regular periodic sweep across the other nameservers to confirm that the selected server remains the fastest to respond. A plot of the nameserver selection of this "ideal" recursive resolver is shown in Figure 2. The servers labelled in the figure are *in* for Mumbai, *eu* for Frankfurt, *am* for Atlanta and *ap* for Singapore.

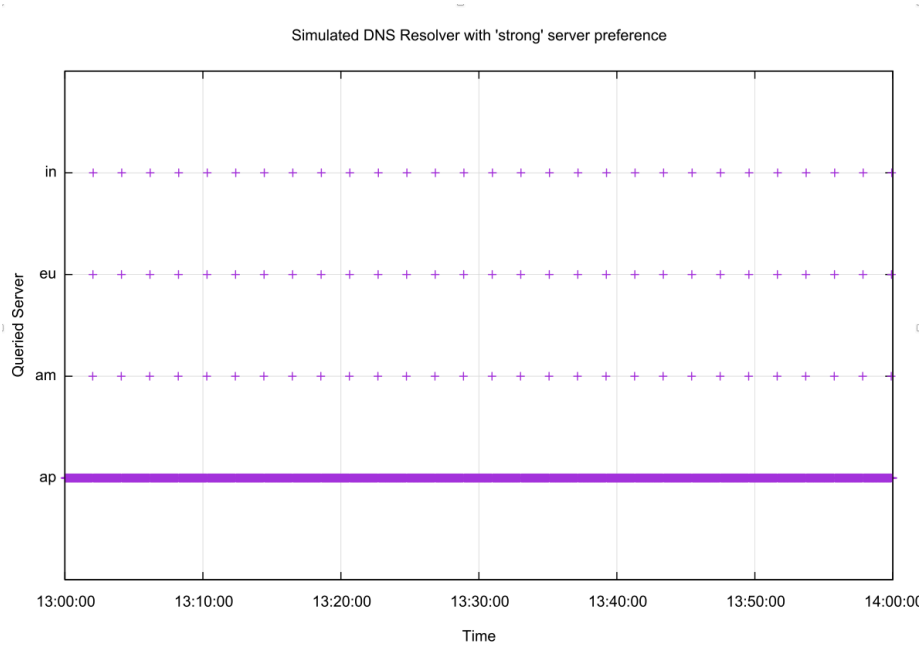

Figure 2 – Simulation of a recursive resolver under a constant query load with strong attachment behaviour

We can compare this idealised behaviour against a couple of widely used recursive resolver implementations. Figure 3 shows the query behaviour of a recursive resolver located in Australia, running Bind 9 when querying for names that are served by this measurement rig. Unlike the earlier tests when looking at the resilience behaviour of recursive resolvers, in this case the nameservers are always reachable and responsive. The test generates unique-named queries at a rate of 1 per second and we are looking at the selection of a nameserver used to form the recursive resolver's response.

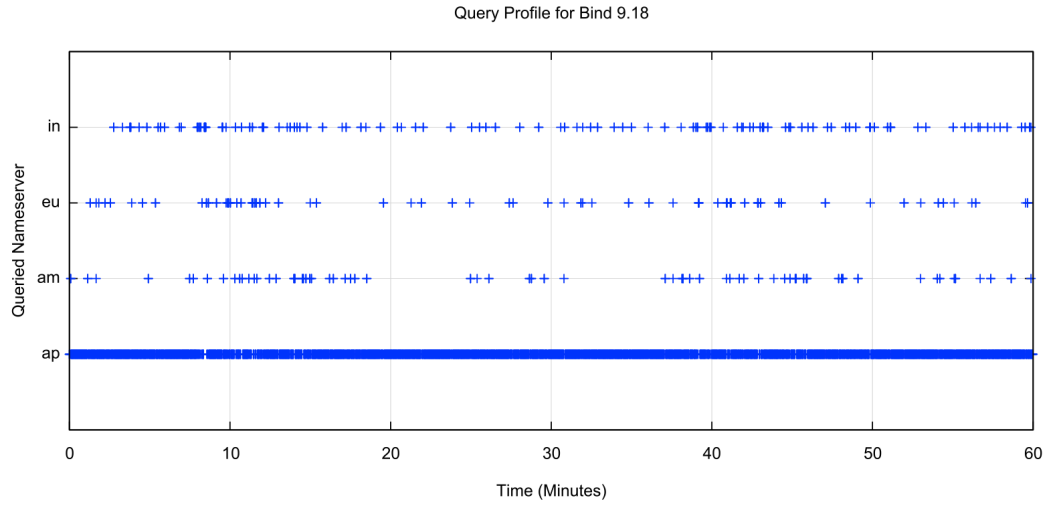

Figure 3 – Nameserver selection of a Bind 9 recursive resolver

Unbound, on the other hand, uses a nameserver selection algorithm with a far coarser level of granularity of 400ms. This roughly corresponds to the diameter of the public Internet, so in practice the selection algorithm corresponds to a random choice with no particular bias to use faster resolvers. Figure 4 shows the behaviour of this test using a recursive resolver running Unbound 1.17.1.

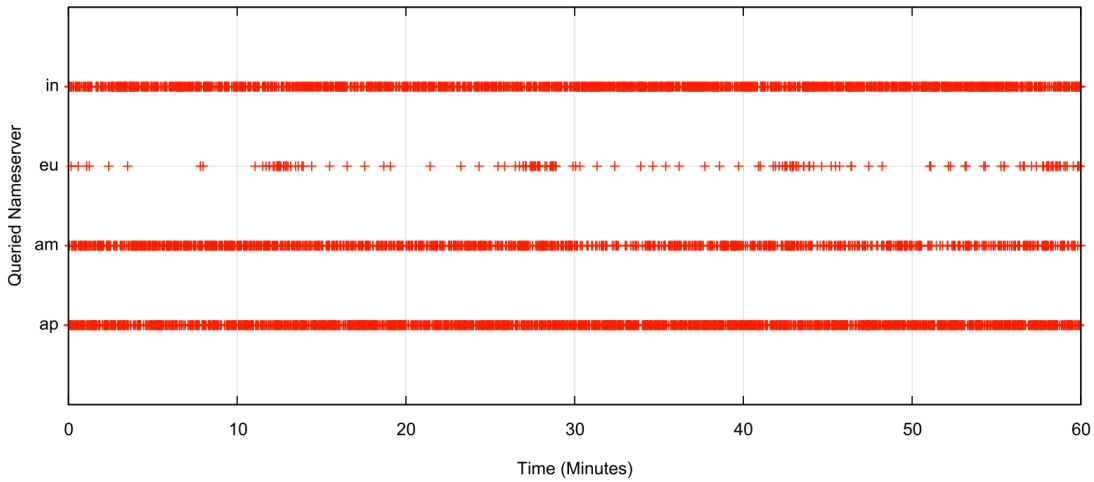

Figure 4 – Nameserver selection of an Unbound 1.17.1 recursive resolver

In the case of the Unbound resolver, there is no strong signal that the resolver prefers to use the fastest nameserver. The RTT times for each of the servers are: 104ms for *ap*, 170ms for *in*, 217ms for *eu* and 278ms for *am*. Why the resolver prefers to use the most distant server, *am*, in preference to the *eu* server is unclear to me.

Using the same technique, we can look at the two most popular open resolvers, the Google Public DNS service operated from 8.8.8.8 and Cloudflare's 1.1.1.1 service. Both of these resolver services are operated as an anycast service, so the test system's DNS query should reach an instance of the anycast set at a location close to the test system, in Australia. Repeated queries would be directed to a resolver located in a similar location as the initial resolver which implies that if there is a bias toward maximising performance, then the recursive resolver would prefer to query the nearest server. What we observe for the Google service is shown in Figure 5 and in Figure 6 for the Cloudflare service.

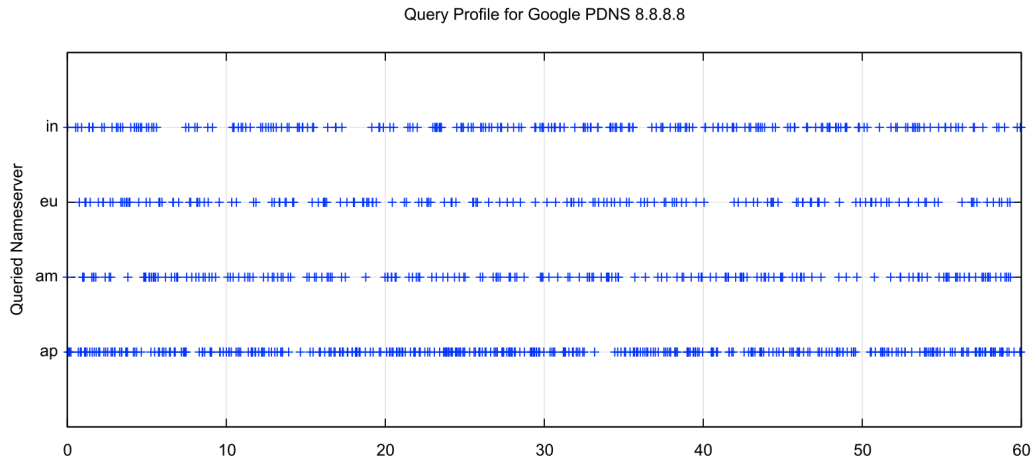

Figure 5 – Nameserver selection for the Google 8.8.8.8 resolver

It is evident that there is no strong attachment preference in the Google service. The nearest service, the *ap* service in Singapore is queried at a higher rate than the server located in Mumbai, and the servers in Frankfurt are queried less frequently, but the bias is not a pronounced one.

This relative preference is also visible in the Cloudflare 1.1.1.1 query profile, where the query volume appears to be inversely related to the RTT for each server relative to the original query location in Australia.

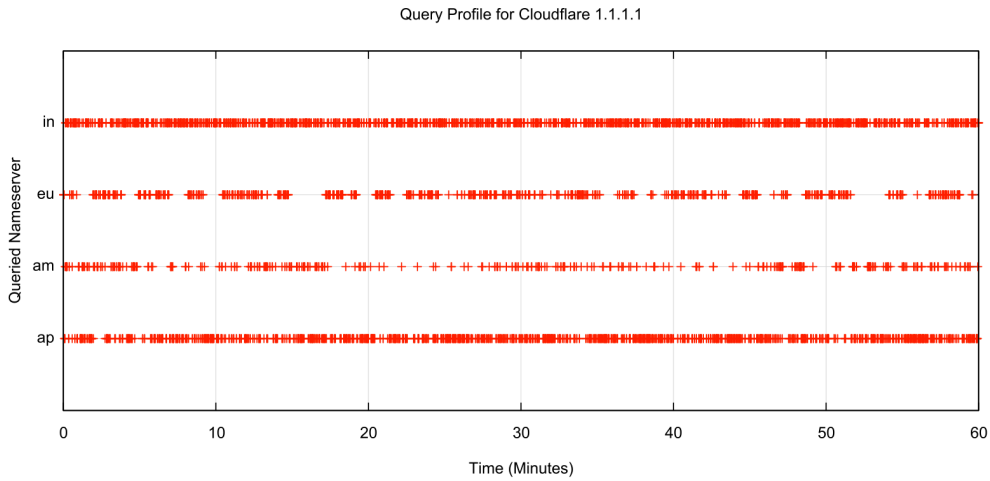

Figure 6 – Nameserver selection for the Cloudflare 1.1.1.1 resolver

The larger scale measurement involved running the measurement over an 8-day period from the 1st to the 8th December 2024. We look at the resolvers that query the four nameservers in our measurement, and use a similar presentation format as already use above the show the nameserver selection preference for each of the recursive resolvers.

The most used resolver is one operated by Bharti Airtel in India, a network that has an estimated 31M end users (https://stats.labs.apnic.net/cgi-bin/aspop?c=IN&d=16/12/2024). The query distribution for a 60-minute period is shown in Figure 7.

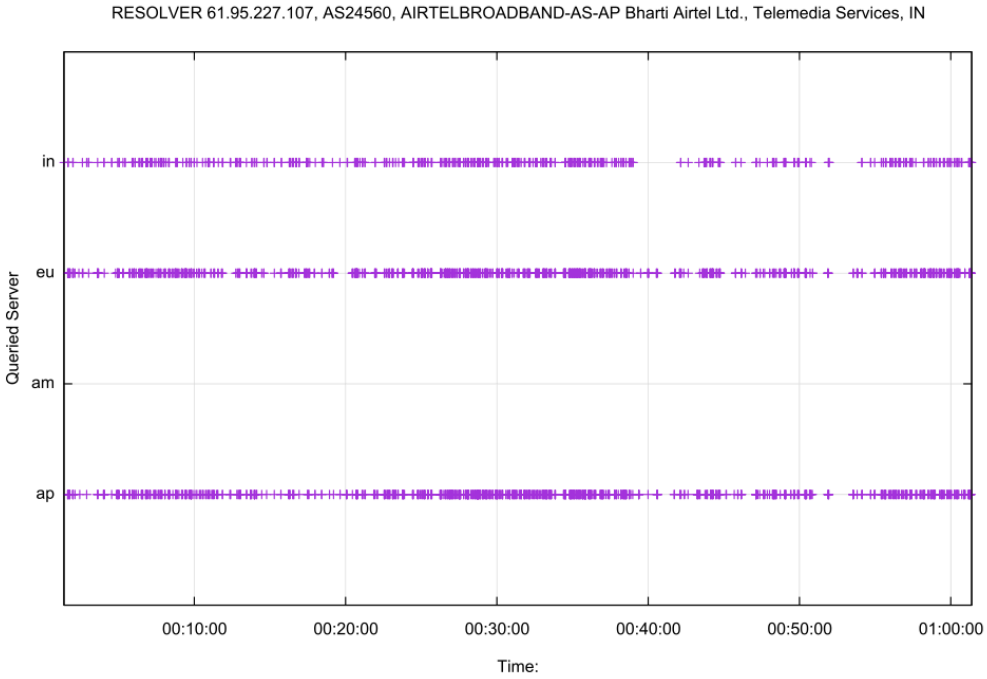

Figure 7 – Nameserver selection for 61.95.227.107, operated by Bharti Airtel, India

At this level of granularity it is evident that the resolver has a preference to direct queries to the Singapore server (*ap*), some 31ms distant by RTT, not to the apparently closer server located in Mumbai, which has a RTT of 27ms. The other server used by this resolver in Frankfurt is 145ms distant. Curiously, there were no queries sent to the server located in Atlanta over this period.

The summary of the 8-day measurement period for this resolver is shown in Table 5. As the queries are not uniformly spread in time, instead of using a query count as an indicator of preference we'll use a matric of an *attachment time* where the time interval between two successive queries to the same server is added to the total *attachment time* for that server. The *Attachment Ratio* is the relative amount of time that the resolver was attached to this server.

Server	Queries	Attachment Time (secs)	Longest Attachment Interval (secs)	RTT (ms)	Attachment Ratio
Atlanta	87	1	0	243	0%
Singapore	611,992	202,012	725	31	48%
Frankfurt	581,799	183,051	637	145	44%
Mumbai	300,751	32,153	580	27	8%

Table 5 – 8-day Query Profile for 61.95.227.107, operated by Bharti Airtel, India

This presentation clearly shows that this resolver, located in India, has an equal preference to query servers located in Singapore and Frankfurt.

This per-resolver attachment ratio now gives is an approach to represent the attachment preference for the 100 resolvers that had the highest query counts in this measurement.

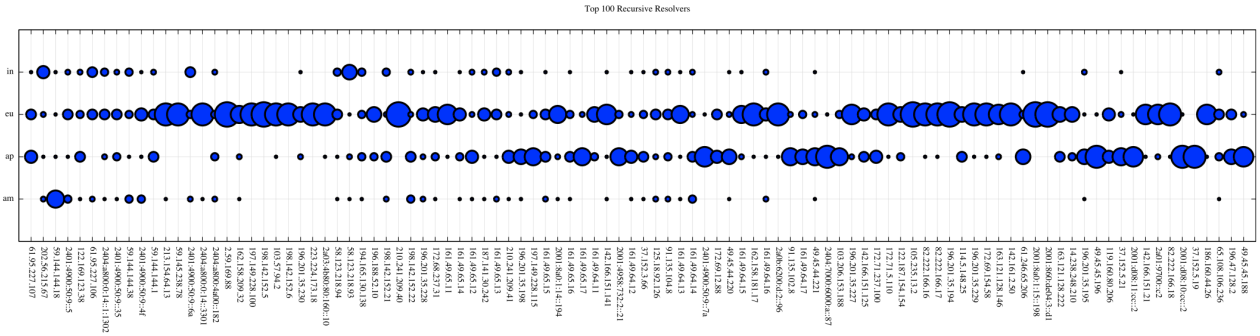

Figure 8 – Attachment Profile for the top 100 resolvers

Each vertical line is the measurement for a single resolver, and the radius of the circles for each server is the attachment ratio. The larger the circle, the greater the attachment ratio. Figure 8 clearly shows a preference to query the servers located in Frankfurt and Singapore, but it is not clear whether this preference is based on the distribution of placement of advertisements or the underlying distribution of user populations.

Over this 8-day period 166,580 resolver IP addresses were observed to query the domain name. To perform this form of query analysis for attachment its necessary, and for this it is helpful to have a relatively dense sequence of queries. The top 1,000 resolver IP address that ask the most queries vary from 1,494,629 to 36,482 queries, or an average of one query every half a second to one query every twenty seconds over the eight-day period. This set of the top 1000 resolver IP addresses account for 27% of the total query volume.

If a resolver queries a single server for more than 60% of the time, then we will term this a "strong attachment" query behaviour. A total of 616 resolver IP addresses show a strong attachment from this set of 1,000 resolvers. If no server has more than 40% of attachment time, then we can term this as "no attachment" preference. A total of 53 resolver IP address show no attachment preference.

The next question is how good is this attachment preference? To perform this analysis, we can measure the RTT from each server to the resolver, and compare the RTT's against the attachment result.

For example, the IPv6 address 2a01:e00:ffff:53:2::13 is a recursive resolver operaed by the ISP Free.FR in France. The ping RTT measurements for this resolver IP address are: Atlanta – 95.3ms, Singapore - 307.5ms, Frankfurt - 9.8ms and Mumbai - 241.6ms. If this resolver were to make an accurate selection of the closest server, then it would select the server located in Frankfurt. The queries made by this resolver show that the server located in Atlanta was used for 70% of the time, which is some 85ms more distant than the closest resolver. So the "error" in this resolver's selection process is 85ms.

We can apply this technique to the 661 resolver IP addresses that show a strong attachment preference. Of the 661 resolvers, some 498 respond to ping requests. Of these 499 resolvers, 199 had made a strong attachment to the server with the lowest ping time (39%). The remaining 299 resolvers have selected a more distant resolver. The average error time interval for these resolvers was 142.3ms. The distribution of these error values is shown in Figure 9. There is a strong signal of an error time of less than 120ms, which is the case for 175 of these resolvers. The highest error value was 434ms.

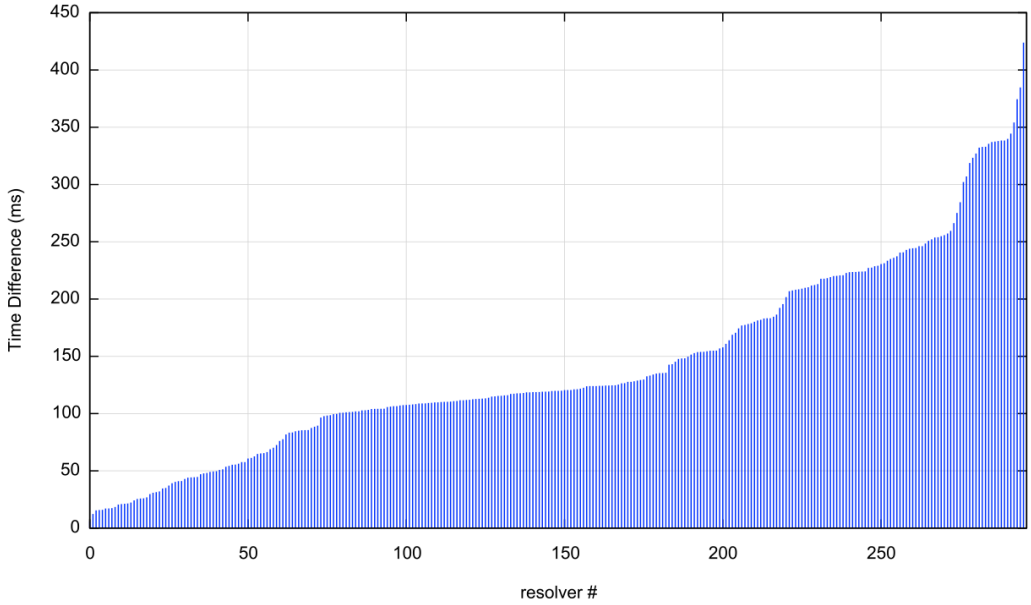

Figure 9 – Selection mismatch RTT time error distribution for 299 resolvers

Conclusions

What us this data telling us?

A domain publisher cannot rely on the server selection algorithm used by today's recursive resolvers to make an optimal selection from a dispersed set of unicast authoritative nameservers. If name resolution performance is an important factor, then the domain publisher needs to look to anycast solution, preferably with a highly diverse collection of points of presence within the anycast constellation. Anycast pushes the responsibility of the nameserver selection process away from the recursive resolver and into the routing system. If the anycast distribution is sparse or concentrated in a small region, then the relatively coarse BGP route selection algorithm of AS Path Length may not produce optimal outcomes. Widely dispersed anycast networks will have a better outcome here.

How many nameservers is enough? If the domain is using anycast as a method to ensure the performance of name resolution, the major motivation behind the choice of the number of nameservers is that of resilience. Using two or more nameservers from the same anycast service platform achieves little in the way of additional resilience. A hybrid approach of using both anycast and unicast nameservers can improve resilience, but at the cost of potential compromise in resolution performance through poor resolver selection of nameservers. There is a practical limit to the number of nameservers that most recursive resolvers will query, so the additional marginal improvement in resilience decreases with each additional nameserver when the domain is served by more than two nameservers, unicast or anycast.

This analysis suggests that the optimal approach is for a domain when considering both performance and operational resilience is for the domain to be served by two distinct dual-stack diverse anycast nameservers.

If the domain being served is DNSSEC-signed using front end-signers, then this advice needs some further qualification, as the scenarios related to key management and coordination between the two anycast platform providers can quickly become quite complex. But that scenario is perhaps best left to a future article!

Disclaimer

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

Author

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

www.potaroo.net