Characterization of Collaborative Resolution in Recursive DNS Resolvers

Rami Al-Dalky[1] and Kyle Schomp[2(✉)]

[1] Case Western Reserve University, Cleveland, USA
rami.al-dalky@case.edu
[2] Akamai Technologies, Cambridge, USA
kschomp@akamai.com

Abstract. Recursive resolvers in the Domain Name System play a critical role in not only DNS' primary function of mapping hostnames to IP addresses but also in the load balancing and performance of many Internet systems. Prior art has observed the existence of complex recursive resolver structures where multiple recursive resolvers collaborate in a "pool". Yet, we know little about the structure and behavior of pools. In this paper, we present a characterization and classification of resolver pools. We observe that pools are frequently disperse in IP space, and some are even disperse geographically. Many pools include dual-stack resolvers and we identify methods for associating the IPv4 and IPv6 addresses. Further, the pools exhibit a wide range of behaviors from uniformly balancing load among the resolvers within the pool to proportional distributions per resolver.

Keywords: DNS · Resolver pools · Dual-stack

1 Introduction

The Domain Name System (DNS) [15] is the component of the Internet that maps human readable names to IP addresses. Traditionally, the DNS is considered to contain three components: (*i*) stub resolvers running on end-user devices that receive resolution requests from apps and forward DNS queries to (*ii*) recursive resolvers that perform the resolution process by iteratively querying (*iii*) authoritative nameservers that are each responsible for zones (or domains) within the DNS hierarchical namespace.

Because of the DNS' vital role in Internet transactions, it is also a convenient choice for implementing traffic management strategies, i.e., load balancing and replica selection can be implemented by authoritative nameservers handing out different hostname to IP address mappings as a function of recursive resolver source IP address and time. Several major content delivery networks (CDNs) [3–5] operate using DNS as the method to assign clients to edge servers. Because there is no direct communication between the end-user devices and the authoritative nameservers in DNS, the location and network connectivity of the

R. Beverly et al. (Eds.): PAM 2018, LNCS 10771, pp. 146–157, 2018.
https://doi.org/10.1007/978-3-319-76481-8_11

end-user device must be inferred from that of the recursive resolver. There is a mechanism for recursive resolvers to attach end-user information to DNS queries [13], but the adoption of the mechanism is still low [16,19] so recursive resolvers remain a frequently used surrogate for end-users. As such, understanding their behavior is of critical importance.

Prior art [10,18] notes that the DNS ecosystem has grown more complex than the early three component model: now recursive resolvers often act in "pools" [10]. Indeed, prior art observe that multiple resolvers may participate in a *single* resolution. The proliferation of public resolution services [6,7] are major use cases for more complex resolver architectures, as the scaling requirements of such systems are substantial.

The proliferation of recursive resolver pools has implications to the efficient functioning of CDNs as pools further obfuscate the association of end-user device to recursive resolver. Unfortunately, little is known about the structure and behavior of pools. In this work, we present what is to the best of our knowledge the first attempt to characterize recursive DNS resolver pools as observed by authoritative nameservers. Our key contributions are:

- **Determine the frequency of pooling behavior and the size of existing pools.** We find use of pools is common, with 71.4% of DNS queries in our dataset originating from pools. Further, pool sizes vary widely with some operators using pools of 2 resolvers and others using pools of hundreds[1].
- **Identify key characteristics of pools including IP, AS, and geographic diversity.** Pools often cover large portions of IP-space with 40% of IPv4 pools distributed within a /16 CIDR block or larger. At the same time, however, pools rarely cross network operator boundaries. We also observe that 10% of pools have large distances between the resolvers in the pool, potentially confusing or misleading efforts to geolocate end-user devices behind the pool.
- **Tangentially, discover dual-stacked resolvers and novel ways to associate IPv4 to IPv6 addresses.** We find many pools of 2 IP addresses are actually dual-stack resolvers and observe that patterns in IPv4/IPv6 address assignment can aid in identifying dual-stack configurations.
- **Classify pools according to several observed behaviors.** We find that pools utilize a wide range of behaviors to distribute DNS queries within the pool. We identify several behaviors including uniform load balancing, offloading, and various other uneven distributions.

The rest of this paper is organized as follows. In Sect. 2, we provide a brief summary of related work. In Sect. 3, we describe our methodology and present the experimental apparatus, dataset, and post-processing steps. Section 4 contains a characterization of pools by network properties. Section 5 classifies the pools by behavior and we draw our conclusions in Sect. 6.

[1] We identify resolvers by IP address and there may not be a one-to-one relationship between hardware and IP address. Regardless, our study reflects what authoritative nameservers observe.

2 Related Work

To the best of our knowledge, we contribute the first assessment of the characteristics of recursive resolver pools. However, several works [10,18] have observed the presence of resolver pools through active probing with CNAME redirections. Alzoubi et al. [10] called the collaborative pools behavior a multiport behavior and interpreted it as either a single multiport machine or load balancing across a resolver farm. Moreover, Schomp et al. [18] looked at the resolver pools from the resolver client perspective by studying the number of recursive resolvers used per client and the geographical distance between clients and recursive resolvers.

While examining DNS pools, we find many dual-stack recursive resolvers. Berger et al. [11] associate IPv4 and IPv6 addresses in DNS queries to find dual-stack machines. In the presence of pools, the authors associate sets of IPv4 and IPv6 addresses rather than identify individual dual-stack resolvers. Other research [12,17] focuses on identifying IPv4 and IPv6 dual-stack machines using TCP options and timestamps, but both methods have limitations when applied to DNS recursive resolvers. First, many resolvers are not open to answer queries from arbitrary sources on the Internet meaning active scanning techniques will miss many recursive resolvers. Second, the techniques require TCP which is a backup transport protocol for DNS and not all TCP implementations support TCP timestamp option. Our technique for discovering pools and dual-stack resolvers does not require any special support from the target resolvers.

3 Dataset & Methodology

We discover pools of recursive resolvers by first discovering pairs of collaborating resolvers and then grouping the pairs together. To find pairs of collaborating resolvers, we use DNS queries for instrumented hostnames. Resolving one of the hostnames induces a resolver to send *two* DNS queries to our authoritative nameservers, as described below. If the resolver is part of a pool, the DNS queries may arrive at the authoritative nameservers from different source IP addresses, offering an opportunity to capture a pair of collaborating recursive resolvers. Below, we describe our dataset, how we extract pairs from the dataset, and then how to form pools from the pairs.

Our dataset consists of DNS query logs from the authoritative nameservers of a major CDN. For a small fraction of Web requests, the CDN platform injects a javascript library [2] that initiates a DNS resolution for an instrumented hostname under the CDN's control. The hostname encodes the end-user device's public IP subnet, and resolves to a CNAME record—a DNS record that indicates a hostname is an alias of another hostname—for a second hostname (also under the CDN's control) that also encodes the end-user's public IP subnet. Thus, the resolution looks like:

$$n1.encoded(x.x.x.x/y).example.com \rightarrow n2.encoded(x.x.x.x/y).example.com$$

The DNS queries for both hostnames are recorded in the logs including the source IP address and a timestamp of when the query was received truncated to the

Table 1. Description of the dataset.

Description	Number
DNS queries	820M
Unique resolvers	429K
Resolver pairs	109M
Unique pairs	1.16M
Singletons	398K
Non-singletons	762K
Groups of resolver pairs	421K
Singletons	360K
Initiator pools	61.5K

Table 2. Top 10 countries with largest number of observed resolvers

Rank	Country	Resolvers
1	US	153K
2	DE	27K
3	BR	24K
4	GB	16K
5	RU	16K
6	CA	15.6K
7	JP	14K
8	AU	10.8K
9	IN	10K
10	IT	8.7K

second. We collect 1 week of logs, July 12–19 2017, containing 820M queries from 429K unique recursive resolver IP addresses. Table 1 follows the breakdown of our dataset in the remainder of this section. Using the EdgeScape [1] geolocation database, we find the recursive resolvers span 27294 ASNs and 234 countries[2]. Table 2 lists the top 10 countries by number of observed resolvers. The top 10 ASNs by number of observed resolvers account for 82.6K (19%) of the total.

Next, we group the queries that are part of the same resolution into pairs (Q_1, Q_2) to extract pairs of collaborating resolvers. Queries that are part of the same resolution are identified by the tuple: encoded end-user subnet, query type (A for IPv4 address or AAAA for IPv6 address) and timestamp. This, however, may not be a unique key because (*i*) multiple end-users in the same subnet may resolve the same hostname at roughly the same time, (*ii*) multiple recursive resolvers may "race" to return an answer fastest to the same end-user, or (*iii*) recursive resolvers may re-resolve the hostname, possibly due to prefetching. The third category can be particularly troublesome due to DNS TTL violations [18] where recursive resolvers may re-resolve only one of the two hostnames in the series, even though both hostnames have the same authoritative DNS TTL. To eliminate noise from these sources, we employ a sliding window of 11 s: $[i-5, i+5]$. If in second i, there is a matching pair of queries, Q_1 and Q_2, we check in the window $[i-5, i]$ for any other queries like Q_1. Similarly, we check the window $[i, i+5]$ for any other queries like Q_2. If we find either, then the pair is discarded because we cannot identify which queries should be paired. The window of 11 s was chosen to allow for up to a 5 s resolver timeout and retry, which is the default DNS timeout value in Linux [14].

The source IP addresses of each pair of queries (Q_1,Q_2) produce an ordered pair of related recursive resolver IP addresses (R_1, R_2). In total, we find 109M

[2] We report the results from EdgeScape, but note that the ASNs matched exactly with what is reported by Team Cymru [8] and countries disagreed for only 166 IP addresses.

sample ordered pairs consisting of 1.16M unique pairs (R_1, R_2), 66% of which were sampled more than once. We exclude 7.3K samples from 5.6K unique pairs where one of the resolver IP addresses belongs to the ASNs of Google Public DNS [6] or OpenDNS [7] and the other does not. All but 853 of the unique pairs are cases of an ISP's resolvers off-loading queries to Google or OpenDNS. We exclude these pairs as they contaminate our pool size measurements (see Sect. 4.1). The remaining 853 pairs exhibit Google Public DNS or OpenDNS off-loading queries to a third party. We suspect that these may be error introduced by query pairs outside our 11 second window. The pairs account for less than 0.1% of all our samples, so we exclude them as well. Of the unique pairs, 762K are non-singletons—$R_1 \neq R_2$—which we use as the basic unit for constructing pools. Note that singleton pairs may still be part of a pool since resolvers may collaborate in only a portion of resolutions.

Next, we merge samples together to form pools. Care must be taken in constructing the pools, because the relationships may not be symmetric. Consider the case of three recursive resolvers: x, y, and z. With observed pairs (x, y) and (x, z), we cannot conclude a direct relationship between y and z. Similarly, if we also observe pair (y, z), we still cannot conclude that all three are members of the same pool, as z may have no affiliation with x and y. Therefore, we opt to take a conservative approach and preserve directional relationships. We group all samples with the same initiator R_1. Continuing the above example, we generate the grouping $(x{:}y, z)$ where the resolver on the left-hand side *uses* all of the resolvers on the right-hand side. From our dataset, we find 421K groups of which 360K represent singletons, i.e., resolvers that did not ever use another resolver R_2 in our dataset. Excluding those, we are left with 61.5K groups that generated 780K (71.6%) of unique pairs and 77.9M (71.4%) of the total samples in our dataset. From here on, we exclude singletons from our analysis and refer to the remaining 61.5K groups as "initiator pools", or just pools where contextually clear.

4 Characterizing Resolver Pools by Network Properties

In this section, we breakdown the initiator pools by network properties. We attempt to find common network structure and characterize the pools by those structures.

4.1 Initiator Pool Size

Here, we explore the size of the discovered pools based on the number of recursive resolver IP addresses in the pool. The number of samples per pool is defined as the summation of the number of samples per unique pair in the pool. Many pools (17.5%) are only sampled once. As a result, our ability to discover the actual size of the pool is limited due to low sampling and our pool size results are a lower bound. However, since our dataset is driven by end-user action, the number of samples per pool correlates with the number of end-users behind the pool. Thus, a more frequently used pool is likely higher sampled and our measurement of size more accurate.

Figure 1 shows the number of recursive resolver IP addresses per pool. As shown in the figure, most pools are small with 38.7K (63%) of pools contain 2 resolvers. We observe that 21.5K (35%) pools with 2 resolvers contain one IPv4 and one IPv6 address and we explore them in more detail in Sect. 5.1. The largest pool we discovered consists of 317 IP addresses contained within 5 IPv4 /24 CIDR blocks and 8 IPv6 /64 CIDR blocks. All blocks belong to ASN 15169, Google Inc. In all, 85% of the pools consist of less than 10 resolvers.

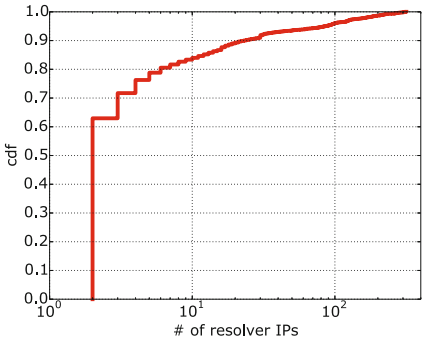

Fig. 1. Size of the initiator pools in number of resolver IP addresses

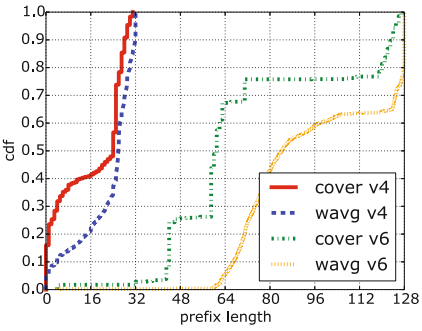

Fig. 2. Prefix length of the most specific CIDR block covering all IP addresses in the initiator pool

In comparison with previous work, Alzoubi et al. [10] observed that 90% of the discovered pools have at most 3 resolvers while a single pool consists of over 22K resolvers. We attempt to replicate their findings using their methodology for constructing pools with our dataset, but do not find a single "megapool". The largest pool we discover is caused by offloading by many third parties to Google and OpenDNS (Sect. 3), and we therefore suspect that the difference between our dataset and the dataset of Alzoubi et al. is reduced observations of offloading to Google and OpenDNS.

4.2 IP-Space Distribution

Next, we investigate how concentrated initiator pools are in IP-space. Intuitively, we expect collaborating resolvers to be closely concentrated, e.g., Google Public DNS publishes a list of whole /24 CIDR blocks that are used in recursive resolution [6]. In this section, we measure how similar the IP addresses of resolvers within a pool are to one another.

First, we calculate the length of the prefix for the most specific CIDR block that covers all IP addresses in the pool. Consider a pool with two IP addresses: 1.2.3.0 and 1.2.3.128. The longest common prefix of the two IP addresses is 24-bits, thus, the covering prefix length is 24-bits. For IPv4, the prefix length varies from 32-bits (indicating a single IP address in the pool) to 0-bits (indicating

that even the leading bit does not match). The values have a similar meaning for IPv6, but extend to a maximum value of 128-bits due to the larger address size. For this analysis, we discard unique pairs where the IP versions do not match and don't plot pools without any unique pairs remaining. The filtering leaves 39.3K initiator pools, 37.7K are IPv4 and 1.6K are IPv6 pools.

The covering prefix lengths for IPv4 and IPv6 are shown in Fig. 2 in lines "cover v4" and "cover v6", respectively. In IPv4, we find that 48% of pools are covered by a prefix shorter than 24-bits and further 38% are covered by a prefix shorter than 8-bits. This indicates that a large fraction of initiator pools are greatly distributed in IP-space. There is also a large amount of variability in the prefix length as demonstrated by the relatively smooth curve. In IPv6 on the other hand, there are 4 clear typical prefix lengths: 44, 60, 70, and 120-bits. The pools with prefix lengths of 44 and 60-bits are operated by Google, while the pools with prefix lengths of 70-bits are operated by AT&T. The prefix lengths greater than 120-bits come from a variety of operators. This result highlights the differing policies network operators apply when assigning IP addresses.

The covering prefix length is susceptible to outliers, however. For example, in a pool of 10 resolvers where 9 match to 24-bits but 1 resolver only matches to 8-bits, the covering prefix length is still 8-bits. Therefore, we next compute the weighted average prefix length between the initiator and each of the other resolvers in the pool using the relative number of samples per unique pair as the weights. The result of this computation is plotted in lines "wavg v4" and "wavg v6" and show a frequently more specific prefix length than the covering prefix length. This indicates that (*i*) resolver usage within a pool is frequently not uniform (see Sect. 5.2), and (*ii*) resolvers closer in IP-space are used more frequently than resolvers further apart. This could be a preference choice, e.g., resolver operators prefer to off-load to nearby capacity, but will off-load to equipment further away if necessary (see Sect. 4.4).

4.3 Autonomous System Distribution

The previous section noted that initiator pools can be dispersed in IP-space. In this section, we endeavor to determine if the pools encompass multiple operators. First, we look at the number of autonomous systems (ASs) per initiator pool and observe that 15.2% of pools are in more than one AS and 0.7% are in more than two ASs. In the most extreme case, one initiator pool consists of 10 recursive resolvers each in a different AS. All of the ASs are Russian and each uniquely identifies a distinct Russian city.

We focus on the 8.9K (14.5%) pools in 2 ASs here and manually compare the WHOIS entries of the most commonly occurring AS pairs. The most frequently occurring pair of ASs is 7018 and 7132 which occur together in 553 pools, and both are operated by AT&T. The second most frequently occurring pair of ASs are operated by Sprint Corporation: 10507 and 3651 occur together in 201 pools. Noting the exception of Google and OpenDNS usage by third parties which we filter (see Sect. 3), we conclude that other collaboration between recursive resolvers from unrelated ASs is rare.

4.4 Geographic Distance Within Pools

In this section, we investigate the geographic distribution of recursive resolvers within a pool. Large distances between recursive resolvers can have ramifications for any system attempting to geolocate end-users by the recursive resolver that they use, e.g., CDNs that attempt to map end-users to nearby replicas for performance reasons. We attempt to determine whether recursive resolvers in an initiator pool are all in the same location, and, if not, how far the recursive resolvers are from the initiator. We calculate the Great Circle distance between the initiator and the resolvers in the pool using the geolocation information provided by EdgeScape [1]. We consider three methods for calculating the distance within a pool:

1. Minimum distance between the initiator and any resolver in the pool (the lower bound).
2. Maximum distance between the initiator and any resolvers in the pool (the upper bound).
3. Weighted average distance between the initiator and the resolvers in the pool using the number of samples per unique pair as the weights.

Figure 3 provides the distributions of those distances. We notice that 6.2K (10%) of the pools have a weighted average distance more than 160 km (100 miles) and those pools represent 3.6M (3.3%) of our total samples. This means that the majority of pools—producing 96.4% of the samples in our dataset—consist of resolvers that are close to each other geographically. We examine the pools in the tail with weighted average distance more than 160 Km and observe 319 pools where all IP addresses are within 66.102.0.0/20 and geolocate across the US[3]. The IP addresses reverse resolve to *google-proxy-*${*IP*}*.google.com*, indicating that they are Google proxies [9]. This suggest that Google proxies (*i*) perform recursive resolution themselves rather than rely upon Google's DNS infrastructure, and (*ii*) collaborate amongst themselves.

The geographically distributed pools add extra time to the resolution process as off-loading a follow-up query necessitates further network delay. Moreover, distance within the pool complicates end-user mapping in CDNs as discussed before. We observe that pool intra-distance is small for the majority of pools, a positive result for CDNs.

5 Classifying Resolver Pools by Behavior

In this section, we provide a classification of the pools by how DNS queries are distributed among the resolvers within the pool. Unfortunately, low sampling makes identifying behavior in many pools difficult. Consequently, we limit this section to the study of highly sampled pools, reducing our dataset to the 18.7K (30%) pools with at least 100 samples. The threshold was chosen because the

[3] We manually verified that our example cases are approximately located where EdgeScape reports by using ping measurements from nearby landmark locations.

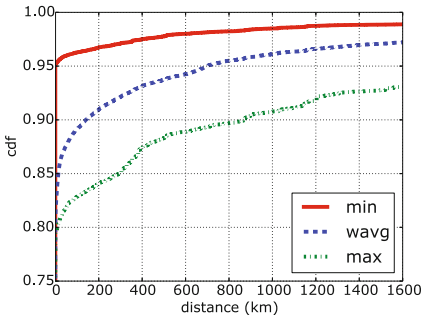

Fig. 3. Tail of the distribution of distances within pools

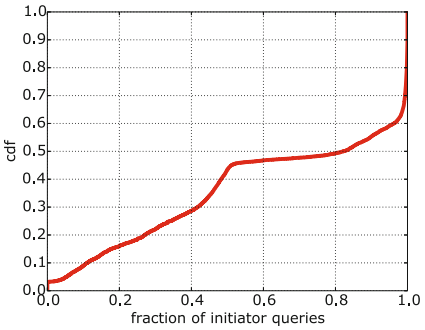

Fig. 4. The number of samples in an initiator pool that are initiator to initiator

distribution of samples per pool surrounding the threshold is smooth. We classify the pools into 4 categories described below: (*i*) dual-stack resolvers, (*ii*) uniform load balancing, (*iii*) off-loading and (*iv*) others. Table 3 contains a breakdown of the categories.

5.1 Dual-Stack Resolvers

We observe that 6.8K (36%) of the pools contain exactly 2 recursive resolver IP addresses where one address is IPv4 and the other is IPv6, and hypothesize that these are actually dual-stack recursive resolvers that switch between interfaces during resolution. To test this theory, we attempt to match the IPv4 and IPv6 addresses by patterns in the IP assignments: (*i*) the IPv4 octets embedded as the final 4 hextets (e.g., *1.2.3.4* and *89ab::1:2:3:4*), or (*ii*) the final IPv4 octet equal to the final IPv6 hextet (e.g., *1.2.3.4* and *89ab::4*). Of the 6.8K potential dual-stack resolvers, 696 match (*i*) and another 1.3K match (*ii*). Interestingly, We also observe 4 cases where the full IPv4 address is embedded within the IPv6 address, but not in the final 4 hextets (e.g., *1.2.3.4* and *89ab::1:2:3:4:5678*). From manual inspection of the remaining 4.8K potential dual-stack resolvers, we observe incremental IP assignment patterns among the pools within the same AS that also aid in positively identifying dual-stack resolvers. For example, in AS 46690, Southern New England Telephone Company, IP assignment appears incremental in both IPv4 and IPv6, but shifted: *w.x.y.z* forms a pool with *a:b:c::${z+C}* where C is a constant. Anecdotally, we observe similar patterns in several ASs.

Next, we note that the IPv4 interface is heavily favored for transport in the pools that are potentially dual-stack resolvers. For each pool, we calculate the ratio of DNS queries using the IPv4 interface versus the IPv6 interface, and find the median ratio is 11:1. Only 718 (10.6%) of the pools favor the IPv6 interface over the IPv4 interface. Our findings here may be impacted by our measurement apparatus, as the DNS zone we use to collect the dataset has more IPv4 delegation records than IPv6 records: 11 and 2, respectively. Depending upon

Table 3. Breakdown of pools by classification

Description	Number of pools	Percentage of pools
Total discovered Pools	61.5K	-
with ≥100 samples	18.7K	30%
Dual-stack resolvers	6.8K	36%
Uniform load-balance pools	2.5K	14%
Off-loading pools (rare)	4.9K	26%
Off-loading pools (frequent)	300	<1%
Other	4.3K	23%

recursive resolver policy, the imbalance may cause resolvers to prefer reaching our authoritative servers over IPv4 (e.g., if the resolver selects a delegation via round robin), thus impacting the number of samples per network protocol.

Finally, we find that in 506 (7.4%) of the pools, the IPv4 and IPv6 addresses are in different ASs operated by the same company. For instance, we find pools belonging to Frontier Communications that exhibit an incremental IP assignment pattern where the IPv6 address is in AS 5650 and the IPv4 addresses is in AS 3593. Thus, having IPv4 and IPv6 addresses in different ASs does not infer that they do not both belong to a dual-stack machine. As future work, we plan to apply the patterns above to identifying dual-stack resolvers within pools of more than 2 resolver IP addresses.

5.2 Load Balancing, Off-Loading and Other Pools

Turning to the 11.9K pools that are not dual-stack resolvers, we classify them into three categories based on the scheme used by the initiator to distribute queries among the resolvers in its pool. Recall that each unique pair has an associated number of samples. Therefore, we can compute the fraction of observations for each resolver within the pool. First, we check for uniform load across the resolvers using the chi-squared test (χ^2) for uniformity. Using a standard 5% significance level, we reject the null hypothesis—that the distribution is uniform—if the p-value is less than 0.05. Otherwise, we conclude the pool is a uniform load balancing pool. Approximately 2.5K (14%) of the pools are balancing the load evenly among the resolvers within the pool.

Next, we explore the 9.5K pools where the null hypothesis is rejected. We observe that in many pools the initiator uses itself much more frequently than other resolvers in the pool. We compare the number of samples in the pool that are initiator to initiator, (x, x), with the number of samples for any other unique pair in the pool, (x, y). The line in Fig. 4 shows the ratio initiator to initiator samples over the maximum samples of any other unique pair. An x-axis value of 0.5 indicates that the initiator uses itself at least as often as any other resolver in the pool. There is a clear behavioral shift at greater than 0.5. We choose the threshold $x = 0.8$ to separate the pools into classes. The 4.9K (26%) pools

where $x \geq 0.8$ we term off-loading, as the initiator prefers to use itself, but will off-load queries to other resolvers less frequently. The frequency of off-loading differs widely across pools. In the extreme, we observe 2 recursive resolvers in AS 1221—Telstra Corporation—that use each other in only 8 out of 340K samples. We postulate that resolvers like Telstra's are using a failover behavior when a DNS query is unsuccessful, possibly due to packet loss. Unfortunately, we are not able to identify the reason why queries are off-loaded from our dataset, but note that it is not a function of domain name, as all queries in our dataset are for a single domain.

At the far left of Fig. 4, there is another behavioral shift where the initiator never uses itself in 300 (<1%) of pools. In 219 of the pools, the initiator is an IPv6 resolver, and we therefore conclude that IPv4 transport preference is a main cause of the behavior.

The remaining 4.3K (23%) pools that lie in the middle, we classify as other because they exhibit a variety of behaviors. One behavior is a load distribution which is uneven, e.g., AS 20057—AT&T Mobility—uses a peer structure of IPv4 resolvers where each initiator has a peer and distributes queries roughly 52% to 48% to itself and the peer, respectively. We note our p-values for the AS 20057 pools is roughly 0.002, well below the significance level. In a slightly more dramatic example, AS 4780—Digital United—uses a distribution of 58% to 42% between their resolver peers. Larger pools of greater than 2 resolvers also use complex policies where the fraction of use varies per resolver in the pool, e.g. in a pool of 3 resolvers, they are each used in 50%, 30%, and 20% of samples. Another behavior is a combination of off-loading with uniform load balancing. We observe multiple pools in Level3's AS where all resolvers within the same /24 CIDR block are uniformly load balanced, and resolvers within a second /24 CIDR block are used roughly once out of 500 samples. The range of behaviors within the other classification comes near to the range of operators, thus we do not attempt to further refine this classification.

6 Conclusion

This paper examines the characteristics of recursive resolver pools. We find the resolver pools are not trivial and a large fraction of DNS queries originate from pools of resolvers. First, we examine the characteristics of the pools based on general network properties. We find that the pools are varied in size and confined within an operator's network. Further, we find that a large portion of pools are distributed in IP space and 10% of the discovered pools are geographically distributed. Next, we classify the resolver pools based on their operational behavior. We identify dual-stack resolvers by looking into pools of 2 resolvers which have both IPv4 and IPv6 addresses and find that 36% of the pools are dual-stack resolvers. We observe that there are different assignment patterns—varying from operator to operator—that can be used to associate IPv4 and IPv6 addresses. Further, we classify the pools into 3 major categories based on the distribution of DNS queries among the resolvers within the pool. We find that 14% of the

pools uniformly distribute the load among the resolvers within the pools. Moreover, in 26% of the pools, we observe that the initiator resolver prefers to handle the resolution by itself but in some cases it decides to off-load queries to other resolvers in its pool. Finally, 23% of the pools tend to have a wide range of behaviors which varies depending on the operator.

References

1. Akamai EdgeScape (2017). https://www.akamai.com/us/en/products/web-performance/
2. Akamai Real User Monitoring (2017). https://www.akamai.com/us/en/resources/real-user-monitoring.jsp
3. Akamai Technologies, Inc. (2017). https://www.akamai.com/
4. Cloudflare, Inc. (2017). https://www.cloudflare.com/
5. Fastly, Inc. (2017). https://www.fastly.com/
6. Google Public DNS (2017). https://developers.google.com/speed/public-dns/
7. OpenDNS (2017). https://www.opendns.com/
8. Team Cymru IP To ASN Mapping (2017). http://www.team-cymru.org/IP-ASN-mapping.html
9. Agababov, V., Buettner, M., Chudnovsky, V., Cogan, M., Greenstein, B., McDaniel, S., Piatek, M., Scott, C., Welsh, M., Yin, B.: Flywheel: Google's data compression proxy for the mobile web. In: NSDI, vol. 15, pp. 367–380 (2015)
10. Alzoubi, H.A., Rabinovich, M., Spatscheck, O.: The anatomy of LDNS clusters: findings and implications for web content delivery. In: Proceedings of the 22nd International Conference on World Wide Web, pp. 83–94. ACM (2013)
11. Berger, A., Weaver, N., Beverly, R., Campbell, L.: Internet nameserver IPv4 and IPv6 address relationships. In: Proceedings of the 2013 Conference on Internet Measurement Conference, IMC 2013, pp. 91–104. ACM, New York (2013)
12. Beverly, R., Berger, A.: Server siblings: identifying shared IPv4/IPv6 infrastructure via active fingerprinting. In: Mirkovic, J., Liu, Y. (eds.) PAM 2015. LNCS, vol. 8995, pp. 149–161. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-15509-8_12
13. Contavalli, C., van der Gaast, W., Lawrence, D., Kumari, W.: Client Subnet in DNS Queries. RFC 7871, RFC Editor, May 2016. https://tools.ietf.org/html/rfc7871
14. Cui, H., Biersack, E.: Trouble shooting interactive web sessions in a home environment. In: Proceedings of the 2nd ACM SIGCOMM workshop on Home networks, pp. 25–30. ACM (2011)
15. Mockapetris, P.: Domain names - implementation and specification. STD 13, RFC Editor, November 1987. http://www.rfc-editor.org/rfc/rfc1035.txt
16. Otto, J.S., Sánchez, M.A., Rula, J.P., Bustamante, F.E.: Content delivery and the natural evolution of DNS: remote DNS trends, performance issues and alternative solutions. In: Proceedings of the 2012 ACM Conference on Internet measurement conference, pp. 523–536. ACM (2012)
17. Scheitle, Q., Gasser, O., Rouhi, M., Carle, G.: Large-scale classification of IPv6-IPv4 siblings with variable clock skew. In: 2017 Network Traffic Measurement and Analysis Conference (TMA), pp. 1–9. IEEE (2017)
18. Schomp, K., Callahan, T., Rabinovich, M., Allman, M.: On Measuring the client-side DNS infrastructure. In: Proceedings of the 2013 Conference on Internet Measurement Conference, IMC 2013, pp. 77–90. ACM, New York (2013)
19. Sudrajat, F.U.: The State of Adoption of DNS ECS Extension on the Internet. Master's thesis, Case Western Reserve University (2017)