Inferring Internet Server IPv4 and IPv6 Address Relationships

Robert Beverly
Naval Postgraduate School

Arthur Berger
MIT CSAIL / Akamai

Nicholas Weaver
ICSI / UCSD

Abstract

While IPv6 is finally experiencing non-trivial deployment, IPv4 and IPv6 are expected to co-exist for the foreseeable future, implying dual-stacked devices, and protocol inter-dependence. We develop and deploy a system for characterizing the association between IPv4 and IPv6 addresses (”siblings”) within network server infrastructure, with specific focus on Internet DNS and web servers. We develop novel active and passive techniques for finding DNS resolver sibling groups, and find only 14% are one-to-one, primarily due to complex deployment and inter-relationship between IPv4 and IPv6. We then describe a targeted method to actively interrogate candidate (IPv4, IPv6) pairs to determine if they are assigned to the same device. We find that the IPv4 and IPv6 addresses of Internet servers frequently belong to different interfaces, machines, and even autonomous systems. Our results have important implications on network resilience, security, and performance measurement, as well as the evolution of IPv6.

1 Introduction

After languishing for decades, IPv6 [11] is experiencing renewed interest [15], in large part due to technical, economic, and political reasons [9, 26]. However, IPv4 and IPv6 are expected to co-exist for the foreseeable future. For instance, transition strategies, practical constraints, and inertia imply that portions of the IPv6 infrastructure may depend on IPv4, where hosts and infrastructure are commonly “dual-stacked.” Understanding the relationship between, and inter-dependence of, IPv4 and IPv6 infrastructure, and its evolution over time, is an open problem important to network structure, resilience, security, and performance measurement.

We examine the problem of associating *infrastructure* IPv6 addresses with IPv4 addresses. Specifically, this paper focuses on IPv4 and IPv6 address relationships among Internet DNS and web servers. Previous measurement research has explored IPv6 adoption and penetration of the broader client population, for instance via web instrumentation [37, 31]. However, less attention has been placed on characterizing IPv6 within Internet server infrastructure. Several initiatives [16, 17] and anecdotal evidence [32] suggest that IPv6 infrastructure deployment is proceeding well in advance of client adoption. For instance, large residential service provider networks [10], and content providers [1] fully support IPv6, but are only conservative in making it publicly available.

We term the relationship of associated IPv4 and IPv6 addresses “siblings.” In the case of a client, the relationship is typically straightforward, with a dual-stacked host producing a 1-to-1 sibling relationship. However, server infrastructure is significantly more complex. “Equivalence classes” of IPv4 and IPv6 addresses can result from e.g. a cluster of dual-stacked machines that perform load-balancing. We therefore further distinguish a IPv4-IPv6 association corresponding to a cluster or group of cooperating machines, even if physically distributed, as “equipment siblings.” IPv4 and IPv6 addresses that belong to a single physical machine are thus “machine siblings.” Our measurement techniques allow us to differentiate between equipment and machine siblings, revealing important properties of the underlying web and DNS infrastructure and the manner in which IPv6 is employed.

We develop three novel measurement systems, both active and passive, for characterizing Internet infrastructure IPv4 and IPv6 addresses: i) passive DNS using a two-level DNS hierarchy that encodes IPv4 addresses within IPv6 nameserver records; ii) active DNS whereby we probe resolvers to induce various lookup behaviors; iii) targeted server (DNS and web) which measures TCP timestamp clock skew to generate a physical device fingerprint.

Our passive technique for finding candidate siblings of DNS resolvers provides a new method to characterize a critical portion of the Internet infrastructure. We deploy the opportunistic system on a large commercial Content Distribution Network (CDN) to gather approximately 674,000 (IPv4, IPv6) address pairs. We find only 14% of the pairs map as a bijection, primarily due to load-balancing and a small number of very large DNS cluster resolvers as deployed by large providers. Overall, inter-relationship between IPv4 and IPv6 in this domain is particularly complex. To validate and better understand these passive results, we perform active DNS measurements. We find 75% of address 4-tuples discovered via active probing also exist within our passive data and are consistent with our inferred sibling groupings. Lastly, our targeted method allows us to actively interrogate candidate IPv4, IPv6 server addresses to determine if they correspond to machine siblings. We tie our passive and active DNS measurements by using the targeted technique to refine equipment sibling equivalence classes to more precise machine sibling equivalence classes.

Finding infrastructure siblings has several important motivations. First, as IPv6 deployment increases, the sibling associations will evolve. Our sibling discovery system can aid in tracking how the IPv6 network spreads, where we expect, at least initially, it to largely overlap with the IPv4 network, and with time, be increasingly autonomous, and where the future state is unclear.

Second, the extent to which IPv4 infrastructure depends on IPv6, and vice-versa, has security and critical infrastructure protection implications that are currently unknown. For example, an attack against the IPv6 address of an Internet web or DNS server may or may not impact that server's IPv4 service. Further, siblings imply the potential for correlated failures. Whether IPv6 infrastructure is deployed on the same equipment that supports IPv4 is an important hypothesis we test in this work.

Finally, measurement research that aims to understand various performance and topological aspects of the IPv6 network, as compared to the existing IPv4 Internet [31, 22], requires sibling discovery. Specifically, a common measurement technique is to probe IPv4 and IPv6 destinations that have the same DNS name, presuming that those addresses correspond to the same end-host. As shown in §3.5 however, a DNS name in common does not imply that the IPv4 and IPv6 addresses are hosted on the same interface, machine, or even autonomous system (AS). Among popular Alexa IPv6 websites, we find over 25% are not machine-siblings to their IPv4 counterpart, and 43% of non-siblings reside in different ASes. Our targeted sibling detection can identify such cases, thereby preventing potentially erroneous measurements intending to compare IPv4 and IPv6 performance or paths.

Toward better understanding the relationship between IPv4 and IPv6, we make four primary contributions:

1. A new passive technique for *opportunistically* pairing IPv4 and IPv6 addresses of DNS resolvers.
2. A novel approach for *active* DNS resolver probing to discover equipment siblings.
3. The first method to test if *targeted* candidate IPv4, IPv6 addresses are machine siblings.
4. Real-world deployment of the techniques to gather DNS and website siblings and equivalence classes.

The remainder of this paper is organized as follows. Section 2 describes the three measurement techniques comprising our system in detail as well as our deployment. We present results in §3 and discuss findings in §4. Finally, we conclude in Section 5.

2 Methodology

Our system includes three novel measurement techniques. First, we develop an *opportunistic* method to passively discover candidate siblings of DNS resolvers. Second, we provide the first reliable *targeted* sibling detection technique, which can be run on-demand, by using physical device fingerprinting. Third, we create a custom DNS server that permits *active* measurement of DNS resolvers. Our active DNS technique ties the first two techniques by permitting targeted fingerprinting of DNS resolvers to refine equipment siblings into clusters of machine siblings. This section describes each of our system's techniques in detail.

2.1 Opportunistic DNS Technique

We opportunistically find candidate siblings of DNS resolvers by exploiting: 1) a two-level authoritative DNS resolution, and 2) the ability to encode an IPv4 address in the lower order bits of an IPv6 address.

Clients rely on local resolvers to perform DNS resolution. We seek to ascertain the IPv4, IPv6 siblings of dual-stacked DNS resolvers. Assume a recursive DNS resolver servicing a local client requesting resolution of `www.a.example.com`. As depicted in Figure 1, the resolver requests resolution (typically an `A` record) for this domain from the first-level authoritative nameserver via an IPv4 DNS query. The first-level nameserver responds with the second-level `NS`, and corresponding "additional" `A` and `AAAA`, records [25]. Crucially, the `AAAA` records of the second-level DNS, as returned by our first-level DNS, are formed dynamically. The first-level DNS *encodes a query's IPv4 source address in the lower-order bits of the response's additional* `AAAA` *record.*

For example, Figure 1 shows the first-level DNS responding to a DNS resolver's query with the authoritative `NS` record of the second-level DNS, including an additional `AAAA` record for the second-level nameserver

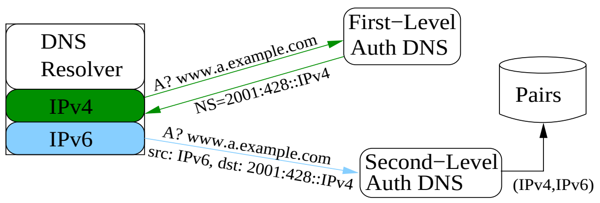


Figure 1: DNS Resolver Siblings: A multi-level authority returns second-level nameserver `AAAA` records encoding a query's IPv4 source in the lower-order bits. The second-level nameserver opportunistically associates IPv6 sources with encoded IPv4 destinations.

that includes the querying resolver's IPv4 source address in the low-order bits. The recursive DNS resolver may use either the `A` or `AAAA` address for the second-level resolution. When the latter is used and the second-level DNS receives the query from the DNS resolver, it pairs the IPv6 source address of that query with the IPv4 address encoded in the IPv6 destination of the query. Note that the dynamically generated `AAAA` nameserver record is valid[1].

We deploy this system within Akamai [30] to a selection of domains using its pre-existing multi-level DNS hierarchy (whose primary purpose is to resolve domain names to addresses that can best serve the client). Deployment on Akamai affords us a rich and diverse dataset, however the general technique can be implemented on any two authoritative nameservers under common control along the same DNS namespace hierarchy.

Note however that the technique could be implemented even on a single box, with multiple addresses, where the IPv4 glue address for the NS record for `example.com` is distinct from the IPv4 and IPv6 glue addresses for the NS record that causes the lookup to the second level.

2.2 Targeted Fingerprinting Technique

Network fingerprinting is a common technique that relies on implementation and configuration-specific characteristics to identify devices. Fingerprinting may be active or passive, and different techniques permit different fingerprint granularity. For instance, active operating system fingerprinting [23] may aid in eliminating false siblings, but is unlikely to provide sufficient granularity to gain confidence in a true sibling relationship as the set of possible operating systems is small relative to the set of possible interfaces.

Instead, we utilize previous work on *physical* device fingerprinting [20]. While this technique has been used in the past, we apply it in a novel context to obtain machine siblings. Observe that any application or transport-layer fingerprint will be common to the lower level network protocol, whether IPv4 or IPv6. In particular, we use evidence of clock skew, visible from TCP-layer timestamps, to remotely identify devices. By actively communicating with a pair of remote IPv4 and IPv6 endpoints over TCP, we infer whether they are siblings based on the similarity of their clock skews.

Define a candidate pair as (I_4, I_6). We periodically connect to a running TCP service on I_4 and I_6 and negotiate the TCP timestamp option [18]. We receive a sequence of time-stamped packets along with their arrival time relative to our prober. Let t_i^4 be the time at which the prober observes the i'th IPv4 packet from I_4 and t_i^6 be the observed time of the i'th IPv6 packet from I_6. Similarly, let T_i^4 and T_i^6 be the timestamp contained in the TCP options of the i'th packet from I_4 and I_6 respectively. Following the technique in [20], we compute the *observed offset* of each packet over time.

Given a sequence of offsets, we adopt the linear programming solution in [27] to determine a line that is constrained to be under the data points, but minimizes the distance to the data points. We then obtain:

$$y_4 = \alpha_4 x + \beta_4 \quad \text{and} \quad y_6 = \alpha_6 x + \beta_6$$

I.e., we determine two lines, one corresponding to the interrogation of I_4 and one to I_6 that lower-bounds the set of offset points observed. We determine the angle θ between the two lines.

$$\theta(\alpha_4, \alpha_6) = \tan^{-1}\left|\frac{\alpha_4 - \alpha_6}{1 + \alpha_4\alpha_6}\right|$$

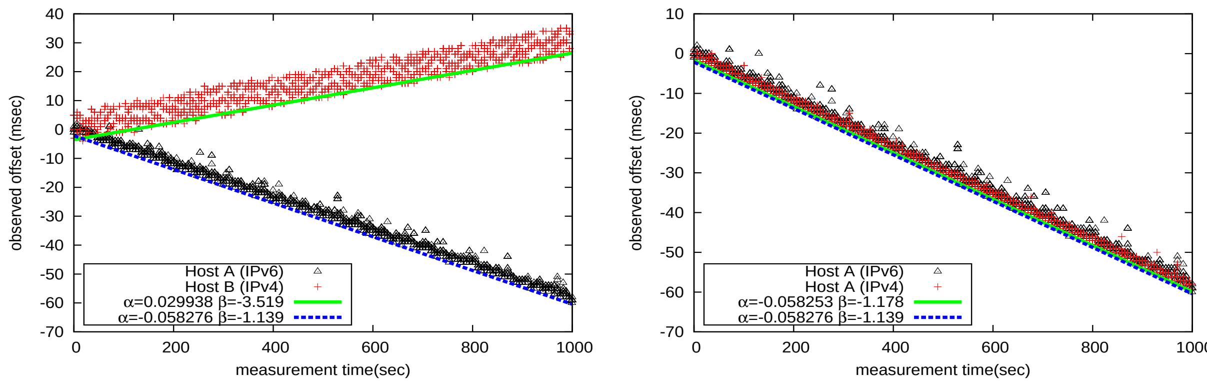
If $\theta < \tau$, then I_4 and I_6 are siblings, where τ is a threshold. Empirically, we find that $\tau = 1.0$ is sufficiently discriminating.

Figures 2(a) and 2(b) graphically illustrate our approach using two hosts for which we know their ground-truth interface addresses. Figure 2(a) displays the observed skew from interrogating Host A's IPv6 interface as compared to Host B's IPv4 interface. We observe not only different skews, but see that the clocks on the respective host are drifting in opposite directions and have different resolutions. Hence, we infer that the IPv4 and IPv6 interfaces interrogated are *false siblings* ($\theta \geq \tau$).

In contrast, Figure 2(b) displays a *true sibling* relationship. In this experiment, we probe the same host (A) via its IPv4 and IPv6 interfaces. We observe nearly identical inferred skew (the linear programming solution determined as $\alpha_4 = -0.058253, \beta_4 = -1.178$ and $\alpha_6 = -0.058276, \beta_6 = -1.139$; $\theta = 1.3 \times 10^{-3}$).

A limitation of our technique is that it is limited to interfaces for which the prober can establish a TCP con-

[1]The second-level DNS accepts queries from an entire prefix; we use a /80 to encode IPv4 sources in 48 bits.



(a) False Siblings (b) True Siblings

Figure 2: Targeted machine-sibling detection via TCP timestamp skew inference.

nection, e.g. a remote web server or a remote DNS server via our truncation method which we discuss next.

2.3 Active DNS Measurements

In addition to the opportunistic, passive DNS data collection, we develop an active DNS measurement to both validate and better understand our passive results. This active DNS measurement ties the previous two techniques together by permitting us to perform the targeted TCP-based timestamping on a subset of the DNS resolvers obtained passively to further evaluate siblings.

There are numerous open DNS forwarders on the public Internet, mostly misconfigured NAT devices. We obtained about 500 addresses of such NATs from data collected by Netalyzr [21] which were both open and forwarded to a resolver which supports IPv6. Additionally, ≈6500 of the IPv4 addresses and ≈2600 of the IPv6 addresses associated with the passively collected DNS resolvers are open and also respond to external requests. Thus there exist a large pool of systems which will process recursive DNS requests, either on their own or by forwarding to their configured recursive DNS resolver.

We actively probe these open systems, issuing specially crafted queries for DNS names for which we are authoritative. Our authoritative domains are served by a custom DNS server that is compliant with the DNS standard [12]. This server listens on both IPv4 and IPv6, uses both TCP and UDP, and reacts differently depending upon the incoming request. The server handles multiple domains that support either IPv4 or IPv6 requests, where the choice of domain selects the IP protocol used by the recursive resolver. We can also force a resolver to contact our server via TCP by always replying with `TRUNCATE` to UDP requests [6], which signals to the resolver that it must retry using TCP.

We initiate queries to the open and forwarding resolvers. The results from our DNS server for the queried object induce the resolver under test to issues a series of queries that alternate between IPv4 and IPv6. We maintain state between requests by specially encoding the returned results such that the end result is a "chain" of IPv4 and IPv6 records that we observe a remote system using to resolve the record. Figure 3 shows a timing diagram of the interaction of our prober and authoritative DNS with a resolver that has IPv6 addresses=$A1, A3$ and IPv4 addresses=$A2, A4$.

An individual measurement queries the open resolver or forwarder for a single `TXT` record. The resolver can only fetch this name using IPv6, but instead of just returning a value, our server returns a canonical name (`CNAME`) alias. This `CNAME` encodes the IPv6 address which contacted our server; for example an IPv6 address `2001:f8b0::91` is encoded into the `CNAME`: "`2001yf8b0yy91.nonce.dnstest.icsi.berkeley.edu`."

This returned `CNAME` exists within the IPv4-only domain. The next `CNAME` redirects back to IPv6, encoding both IPs. After following another `CNAME` back to the IPv4 domain, our server finally returns a `TXT` record reporting the sequence of four IP addresses which contacted our server. Note that while DNS authority servers may typically include multiple records in a single returned result, our server only returns one result at a time in order to force multiple lookups and infer the chain. Our `CNAME` encoding scheme, combined with DNS message compression [25], ensure that, even in the worst case ASCII IPv4 and IPv6 encoding expansion, our chains of length 4 are less than 512 bytes. As 512B is the limit for DNS over UDP, we ensure that our chains rely on neither truncation nor EDNS0 [35].

An example lookup using our deployed authority to query the Google Public DNS is:

```
dig +short TXT @8.8.8.8
cname1e6464.nonce.v6.dnstest.icsi.berkeley.edu
```

which returns the nonce and the sequence of addresses that contacted our server to resolve the request.

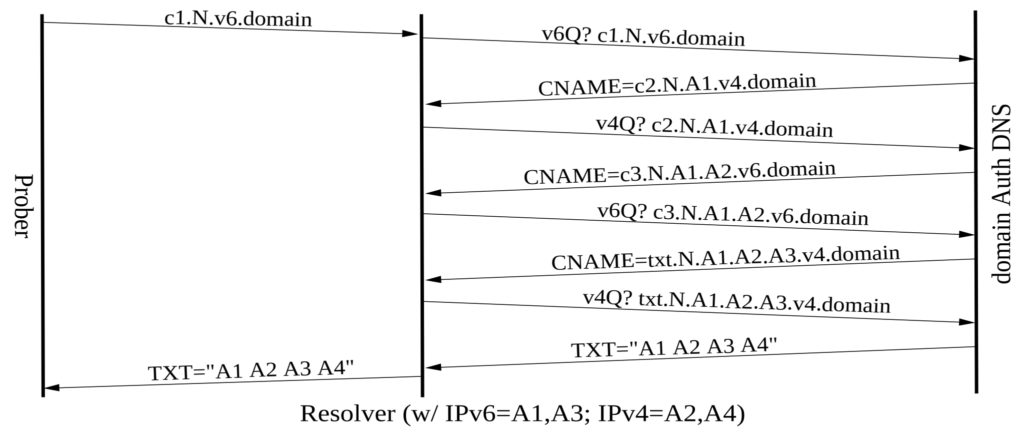

c1.N.v6.domain
v6Q? c1.N.v6.domain
CNAME=c2.N.A1.v4.domain
v4Q? c2.N.A1.v4.domain
CNAME=c3.N.A1.A2.v6.domain
v6Q? c3.N.A1.A2.v6.domain
CNAME=txt.N.A1.A2.A3.v4.domain
v4Q? txt.N.A1.A2.A3.v4.domain
TXT="A1 A2 A3 A4"
TXT="A1 A2 A3 A4"
Prober
domain Auth DNS
Resolver (w/ IPv6=A1,A3; IPv4=A2,A4)

Figure 3: Active DNS Probing: Our authoritative DNS server returns a series of CNAME results with alternating IPv6 and IPv4 glue. We probe a resolver under test for our special domain, including a nonce N. State is maintained on addresses the resolver uses by encoding the IPs along the chain. The final result is the sequence of IPv4, IPv6 addresses used by the resolver (here $A1, A2, A3, A4$).

As we will show in §3, many large-scale resolvers are actually clusters, not individual systems. A cluster might be behind a single publicly facing IP address with load distributed among multiple backend machines, or might encompass multiple publicly visible IP addresses. We therefore repeat our active DNS probes 32 times in order to gain a more complete picture of cluster structure when present. Since the DNS specification [12] requires that the recursive resolver process the entire CNAME chain, these four IP addresses should represent the same "system" responsible for completing the DNS resolution.[2] The replies themselves have a 0 second TTL and the request contains a counter, thus a resolver should never cache the result.

We also conduct a series of requests that force truncation. These use individual queries over both IPv4 and IPv6 where the authority server forces each request to retry using TCP. During these scans, we collect packet traces on our server in order to capture the TCP timestamps. Although the recursive resolver's eventual reply is over UDP, the extra latency required to complete these lookups may be easily observed (e.g. even by external parties by querying `txt.v4t.dnstest.icsi.berkeley.edu` and `txt.v4.dnstest..` several times and comparing the results.)

Although limited to probing IPv6-capable resolvers that either directly accept requests or where we know a forwarder exists (thus excluding most corporate networks), the active measurement has several advantages over the passive measurements. This technique forces the resolver to use IPv6 (instead of relying on a resolver's preference for IPv6 over IPv4). Since the measurements all occur within a short time window, this measurement is not affected by network changes. It also produces a set of up to four siblings, allowing it to more effectively and precisely map the structure of a cluster resolver. Finally, by combining the DNS CNAME measurement with our use of truncation to force TCP lookups, we are able to even more precisely characterize siblings within clusters. While the passive measurements identify common clusters, the active measurements enable use to identify common systems.

3 Results

This section analyzes results from deploying the aforementioned techniques on the IPv4 and IPv6 Internet. However, we begin by concretely defining our notion of sibling "equivalence classes."

3.1 Sibling Equivalence Classes

To better understand sibling relationships, we conceptualize the associations as a bipartite graph. The set of discovered IPv4 and IPv6 addresses (nodes) are connected by edges as our techniques discover associations. We define an *equivalence class* that encompasses siblings, i.e. IPv4 and IPv6 addresses connected in the graph. Let *m*-*n* denote an equivalence class containing *m* IPv4 and *n* IPv6 addresses. To illustrate, Figure 4 provides an example with 5 IPv4 addresses, 7 IPv6 addresses, and 8

[2] We initially noticed a complication where a NAT forwarder could send an initial lookup to our server but, after receiving our CNAME reply, it queries instead a configured recursive resolver for the CNAME. We observe that most such systems are not IPv6 capable, so we changed our query's order from V4/V6/V4/V6 to V6/V4/V6/V4 to prevent a NAT from initiating the request directly, forcing it to contact the configured recursive resolver.

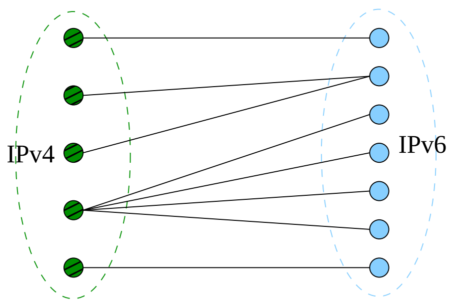


Figure 4: Example equivalence classes, various ways of measuring 1-1 equivalences are given in Table 1.

address pairings (edges). The address pairs partition into 4 equivalence classes, two of which are *1-1*, one is *2-1* and one is *1-4*. 4 of the 12 addresses (33%) are in the *1-1* equivalence class, while 2 of the 8 address pairs (25%) are *1-1*. The canonical case of a simple dual-stack server with a single routable IPv4 and IPv6 address represents a *1-1* sibling equivalence class. Though the converse need not hold as a *1-1* relationship may also be the public facing portion of a more complicated architecture.

3.2 Passive DNS Equipment Siblings

We examine (v4, v6) DNS resolver address pairs as collected by the Akamai network, using our passive technique of §2.1, over a six month period from 17 Mar 2012 to 13 Sep 2012. Akamai observes a significant cross-section of global DNS traffic in its role as a large CDN; the dataset includes resolvers from over 213 countries [3] and contains: 674,000 (v4, v6) pairs with 271,000 unique IPv4 and 282,000 unique IPv6 addresses.

It is well-known that DNS servers and resolvers experience significant load, and that many approaches exist for balancing DNS query load [29, 2]. Thus, while we expect many instances of 1-to-1 IPv4 to IPv6 mapping, we also discover complex IPv4 and IPv6 interrelation among DNS resolvers and resolver clusters. Note: we do not automatically presume that an equivalence class of the DNS servers is an equipment sibling, as it is possible that the machines associated with the addresses that are grouped into in the equivalence class may not be cooperating in any sense, but rather that some of the recorded addresses are from some intermediary boxes uncoordinated with the actual resolver, for example.

Frequently, multiple addresses of a non-*1-1* equivalence class reside in a given network prefix (e.g. `network/mask`). We therefore examine aggregating addresses by prefix, thereby forming network-specific equivalence classes. Given that larger prefixes lead to greater simplification, but also increase the chance of pooling together unrelated equipment, we chose a /24 for IPv4 and a /64 for IPv6. We obtain different results depending on whether prefix aggregation is performed before or after forming equivalence classes. Table 1 shows *1-1* equivalence class prevalence for the passive-DNS data set, as well as for prefix aggregation before and after computing classes, and, for reference, the example of Figure 4.

Table 1: Prevalence of 1-1 equivalence classes

Data Set	**Num of pairs**	**% of eq cls that are 1-1**	**% of v4+v6 in 1-1 eq cls**	**% of pairs in 1-1 eq cls**
Addresses	674,000	77%	34%	14%
Aggregate to prefixes (before)	238,000	67%	31%	18%
Aggregate to prefixes (after)	260,000	83%	55%	39%
Restrict to last week and aggregate to prefixes (after)	49,000	92%	83%	75%
Aggregate to AS's (after)	55,000	95%	92%	89%
Example in Fig 4	8	50%	33%	25%

Perhaps counter intuitively, aggregating addresses to prefixes before forming equivalence classes sometimes leads to classes that have more pairs. For example, when aggregating a v4 address to its /24 prefix, there may be other v4 nameservers in that prefix which are in other equivalence classes, and some of the v6 nameservers associated with these other v4 nameservers are in /64 prefixes different from those in the original equivalence class, and this can continue, thereby creating a larger class. In contrast, if address-based equivalence classes are subsequently aggregated to prefixes (the "after" case), then, by construction, the number of classes remains the same, but the number of pairs within a class can only decrease.

As shown in Table 1, 77% (93,832 of 122,610) of the per-address equivalence classes are *1-1*. Since each equivalence class has equal weight when computing this percentage, and since non-*1-1* classes contain more than two addresses and more than one address pair, percentages for the latter two are lower. A significant percentage of addresses and prefixes are in non-*1-1* classes. In particular, for the 553,126 addresses (sum of IPv4 and IPv6), the percent of v4+v6 addresses in *1-1* equivalence classes is only 34% (2 * 93,832 / 553,126). For the 673,784 address pairs, the percentage is only 14% (93,832 / 673,784).

When aggregated to prefixes (after) these percentages increase, but still only 39% of prefix pairs are *1-1*. Figure 5 is a scatter plot, using the relative frequency for color, of the per-address equivalence classes. We explore

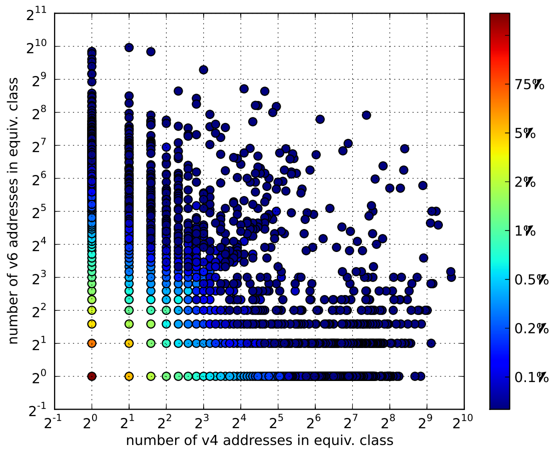


Figure 5: Scatter plot of number of v4 and v6 addresses in the equivalence classes

potential causes for this broad range of *m*-*n* classes, as well as the significant reduction in complexity when aggregating to autonomous systems (ASes), in §4.

3.2.1 IPv4 address within IPv6 address

Auto-tunneled [8] addresses raise a natural question whether the embedded IPv4 address (i.e. the 32 bits that come after the "`2002:`") matches the paired IPv4 address from the discovery technique. Of pairs including a 6to4 address, the embedded IPv4 address equals the paired address for 37% of the pairs, i.e. 63% do not. The weak matching between paired and embedded address is not surprising given that a network may have multiple IPv4 addresses distinct from the embedded IPv4 gateway address. Of matching cases, 21% embed the paired address twice: both after the "`2002:`" and in the lowest order 32 bits. Of 92 pairs with Teredo IPv6 addresses, (`2001:0000::/32`) only one has an embedded IPv4 equal to the paired IPv4. Of the pairs where the IPv6 address is neither 6to4 nor Teredo, just 0.6% have an embedded IPv4 equal to the paired IPv4. Of these, 57% embed the IPv4 address in the lowest 32 bits. For 29%, the highest bit of the embedded IPv4 is at bit position 96 and the rest are scattered across bit positions 48, 56, 64, and 104; and 11 pairs embed twice, at positions 96 and 32.

We also observe an interesting human-centric convention in at least one major ISP's DNS infrastructure. Rather that encoding the IPv4 address directly in the IPv6 address, the network assigns the lower 64 bits of the IPv6 address so that the hexadecimal values render as the decimal equivalent of the IPv4 address, such as `68.87.76.181` which has an IPv6 address of e.g. `2001:558:1014:f:68:87:76:181`.

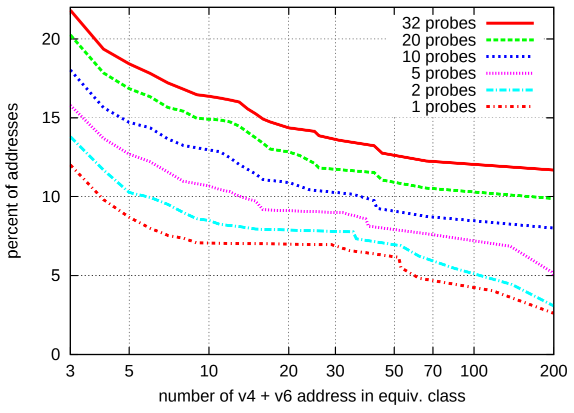


Figure 6: Percent of v4 + v6 address in equivalence classes of at least a given size, for open resolvers probed 32 times on September 14, 2012

3.3 Active DNS Siblings

As the passive-DNS data set was collected over a six-month period, and contains large equivalence classes, discussed in Section 4.3, we next turn to understanding the results gathered over a much shorter time frame using the active DNS measurement technique of §2.3. We first determine the open resolvers in the passive-DNS data set, for which we find 6,581 IPv4 and 2,658 IPv6. We probe each of these open resolvers via active DNS measurement 32 times on Sept 14, 2012, requesting resolution for our special domain which induces chains of lookups across IPv4 and IPv6 DNS glue. Thus, we obtain a data set over a short period of time, about 2 hours, and for which the same set of nameservers is probed repeatedly. Each 4-tuple of v4/v6/v4/v6 yields either 1, 2, or 4 (v4, v6) address pairs, depending on whether the two v4's and two v6's are the same.

We first consider the marginal benefit of *repeated* active probing of the open resolvers. We examine those resolvers for which the 4-tuple is obtained at least 20 times and then compute equivalence classes as obtained by different amounts of probing. Figure 6 shows the percent of addresses that are in equivalence classes of at least a given size, as a function of the number of probes. One can view Figure 6 as the complementary distribution of the size of the equivalence classes. For example, in the case of 32 probes, 15% of the addresses are in equivalence classes of size at least 15. Note that with the x-axis beginning at 3, one can infer the percent of addresses in *1-1* equivalence classes, e.g. the y-intercept of 23% means that 77% of the addresses are in an equivalence class of size 2. The curves that are higher up in the Figure indicate that, overall, the equivalence classes are larger and more complex – the distribution has a heavier tail. The lowest curve in the plot is of equivalence classes

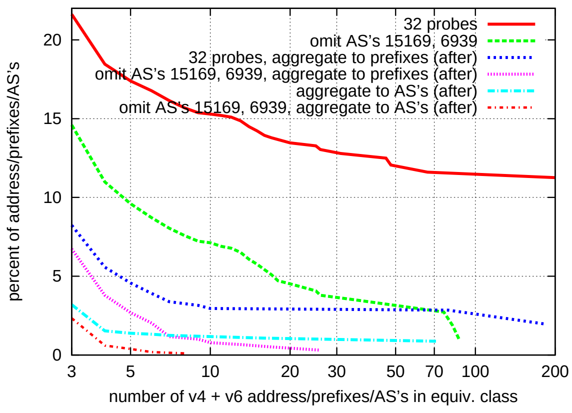


Figure 7: Percent of v4 + v6 address/prefixes in equivalence classes of at least a given size, for open resolvers probed 32 times on September 14, 2012

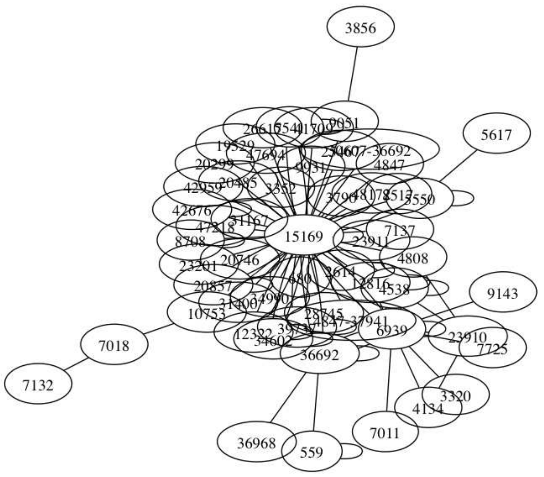


Figure 8: Cluster plot of AS's in 51-20 equivalence class

from those 4-tuples discovered as a result of just the first probe to each of the nameservers. The next curve above that one is from those 4-tuples discovered after the first two probes, and likewise, for the first five, ten, twenty, finally all 32 probes. The Figure shows that with increased sampling, additional nameservers are discovered, and the equivalence classes overall become larger.

Looking at the AS's of the nameservers in the set of 32 probes, we find that two play a significant role in the complexity of the equivalence classes: AS 15169 (Google) and AS 6939 (Hurricane Electric). Figure 7 again examines the actively collected DNS siblings, and shows the impact of excluding these AS's and/or aggregating to prefixes, or to AS's (after). The red (top) line is the same as in Figure 6. Removing the two AS's (the green line) significantly reduces the complexity. If the addresses are then aggregated to prefixes, the resulting equivalence classes of prefixes (the dark blue and purple lines), as expected, are further simplified. Lastly, when the addresses are aggregated to AS's, the resulting equivalence classes of AS's (the light blue and dash-dot red lines), are simpler still.

When AS 15169 and 6939 are included, one of equivalence classes of AS's is significantly bigger than all the others, and is of size *51-20*. Figure 8 shows the structure of this equivalence class. Note that AS 15169 plays a dominant role - if just one resolver in an AS uses Google's public DNS for either v4 or v6, then that AS becomes paired with 15169. Off from the center, towards the lower right of the Figure, AS 6939, Hurricane Electric [14], which provides 6in4 tunnels, plays a secondary role in linking together other AS's. When these two AS's are excluded, this large equivalence class is no longer present, yielding the dash-dot red line in Figure 7.

3.4 Active and Passive Validation

We examine the overlap between the nameserver pairs found by the active and the passive techniques. Of actively discovered address pairs, 75% of the pairs are present within a single passively-discovered equivalence class (of any size). In other words, we verify that 75% of the active sibling pairs are constituents of an equivalence class created via the passive technique. This overlap between two different methodologies for finding siblings, given the large difference in data collection duration, helps validate correctness of the two techniques.

To put this value of 75% in perspective, we perform an analogous calculation using partitions of the active-DNS data set itself in order to understand the consistency of the results over time. In particular, we consider siblings pairs received from open and forwarding resolvers that obtain at least twenty-five 4-tuples via the active technique. For all such resolvers, we temporally divide received sibling pairs from each into bins. The first bin contains the first 10 sibling pairs received from each resolver, the second bin the next 10 pairs from each resolver, and so on. Thus each bin will have results from all of these resolvers, and there will be results from each of the resolvers in each of the bins. We compare the address pair overlap between bins to determine the overall temporal consistency of sibling result sets obtain via repeated probing. Hypothetically, if each open resolver returns the same 4-tuple as a result of each probe, then each bin would contain the same set of pairs, and the overlap is 100%. For each of possible pairs of bins, we compute the percent of address pairs that are in common. We find an inter-bin consistency ranging from from 75% to 77% with a median value of 77%. Different partitions of the multiple probes yield similar results, ranging from 65% to 85% with median values in the mid 70's. We are encouraged that, across the multiple active probes sent over

Table 2: Alexa 100K Targeted Machine-Sibling Inference

Case	**Count**
v4 and v6 non-monotonic (possible siblings)	109 (7.8%)
v4 or v6 non-monotonic (non-siblings)	140 (10.0%)
v4 and v6 no timestamps (possible siblings)	94 (6.7%)
v4 or v6 no timestamps (non-sibling)	101 (7.2%)
Skew-based siblings	839 (60.0%)
Skew-based non-siblings	115 (8.3%)
Total	1398 (100%)

a short time span, we obtain the same level of consistency as when we compare this set of recent probes with the passive collection done over a six-month period.

3.5 Web Server Machine Siblings

We evaluate our TCP-based clock skew machine-sibling detection technique on the Alexa [4] top 100,000 websites. For each Alexa website, we record the DNS `A` and any `AAAA` record on May 3, 2012. From the top 100,000 sites, we find 1398 ($\approx$ 1.4%) with IPv6 DNS records[3]. We then repeatedly probe each site via IPv4 and IPv6 using HTTP by fetching the root HTML page. Probing proceeds sequentially using the deterministic addresses; no DNS lookups are performed during fetching. We conservatively use $\tau = 1.0$ degree in applying the method of §2.2.

Table 2 depicts the machine sibling inference of the 1398 Alexa sites advertising IPv4 and IPv6 records in the DNS. We identify 839 siblings (60.0%), 356 non-siblings (25.5%), and 210 possible siblings (14.5%).

Two exceptions can cause our inference technique to fail: the timestamps are not monotonic across TCP flows, or the TCP timestamp option may fail to be negotiated. A total of 195 sites (14.0%) did not negotiate timestamps for at least one of the addresses. We learned via personal communication with a website operator that the lack of timestamps in this one case is due to a front-end load balancing device; we surmise that similar middleboxes are the cause of the remaining instances missing timestamps in the TCP negotiation. When neither the IPv6 nor the IPv4 address support timestamps, we can only weakly conclude that they are possible siblings. However, if one address supports timestamps and the other does not, they are positively non-siblings.

Second, some TCPs, notably BSD-based [33], randomize the initial TCP timestamp values on a per-flow basis. Again, when one of the two addresses presents randomized values and the other does not, we infer a non-sibling relationship.

[3]IPv6 yield is slightly higher for Alexa top 10K: $\sim$ 2%.

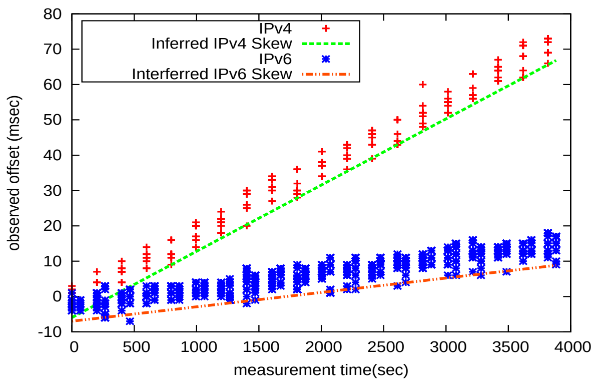


Figure 9: Non-Siblings: Inferred clock skew to `www.socialsecurity.gov` via IPv4 and IPv6

It is important to note that while it is sometimes possible to infer siblings on the basis of application layer fingerprints, for instance the HTTP headers in our experiment, this is not a reliable mechanism. Figure 9 presents one example in our dataset to highlight the advantage of using the clock skew fingerprinting method. We probe `www.socialsecurity.gov` and receive identical HTTP headers via either IPv4 or IPv6 in response. However, Figure 9 suggests that the IPv4 and IPv6 addresses are not machine-siblings – further implying that these two addresses will not have correlated failures and will behave differently under attack than if they were addresses of the same machine.

Among our inferred sibling and non-sibling populations, we examine the origin autonomous system (AS) of the routed prefixes to which the addresses belong, as viewed from the public routeviews [24] BGP table. The origin AS of the corresponding IPv4 and IPv6 addresses of a website allow us to determine whether non-siblings are within the same network, if not the same host – and better understand the impact of measurement studies that are not mindful of siblings (e.g. those that simply pick a server's destination IPv4 and IPv6 addresses and measure path or performance properties, ignoring that the addresses may be on different machines or networks).

Of the 1398 websites in the Alexa top 100,000 with IPv6 records, 246 (17.6%) have IPv4 and IPv6 addresses that reside in different ASes. 49 of 115 skew-based non-siblings (43%) are in different ASes, while 106 of 839 skew-based siblings (12.6%) are in different ASes. Many of the non-siblings in different ASes are due to content distribution network (CDN) effects, where for instance, the IPv4 version of the site is hosted by the CDN, while the IPv6 version of the site is not. Manual investigation of instances of siblings in different ASes reveals some that are different, but related to the same organization. In future work, we plan to use AS-to-organization mappings [7] to better understand these instances of divergent ASes between IPv4 and IPv6.

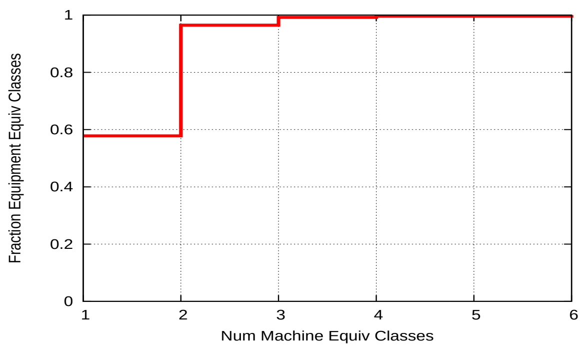


Figure 10: Relationship between equipment siblings and machine siblings, as inferred by applying TCP timestamp method on active DNS equivalence classes.

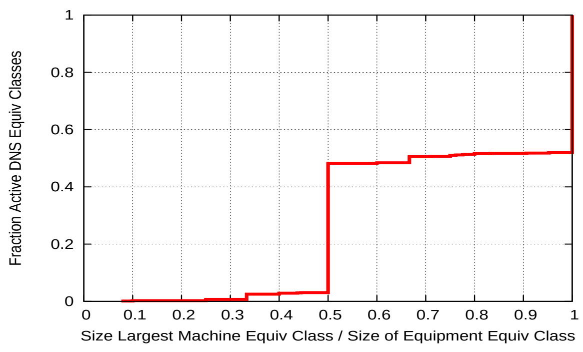


Figure 11: Relationship between equipment siblings and size of the largest inferred machine sibling group within the equipment equivalence classes.

3.6 DNS Machine Siblings

Finally, we note that the passively and actively collected DNS siblings may be, and often are, equipment siblings – i.e. a group of machines operating to service DNS requests. Similarly, a single IPv4 or IPv6 address may represent the outward facing address for a larger group of machines behind that address. To better differentiate such cases, we seek to identify DNS *machine* siblings, i.e. IPv4 and IPv6 addresses that exist on the same physical machine, among the previously inferred equipment sibling equivalence classes.

We therefore tie the passive and active DNS data collection with our TCP timestamp-based sibling inference mechanism by inducing remote resolvers to initiate TCP connections via truncation as described in §2.3.

While actively probing DNS resolvers, we capture all TCP packets. We wish to determine whether the equivalence classes obtained from active probing can be reduced to machine siblings. To cull machine siblings, we determine the timestamp skew for each IP address within an active DNS equivalence class. We compare the skew of an IP in question with all machine siblings of the existing equivalence class. An IP is added to the machine sibling group with the smallest $\theta < 1.0$, i.e. added to the mostly closely matching sibling group. If $\theta >= 1.0$, we create a new machine sibling group with a single IP. This creation of machine sibling equivalence classes continues until all IPs of the equipment equivalence class are clustered. We repeat for all equipment equivalence classes.

For example, the largest equivalence class from active probing includes 172 IP addresses, 78 of which are IPv6. Of these 172 IPs, we identify six different machine sibling groups. The largest machine sibling group consists of 131 IPs, while the second largest contains 6 IPs. 26 IPs have non-monotonic timestamps, while 1 IP did not negotiate timestamps.

We examine the relationship between the equipment siblings as inferred by active DNS probing versus machine siblings. Figure 10 provides the cumulative fraction of DNS equivalence classes versus the number of machine equivalence classes in the equipment class. We observe that nearly 60% of the DNS equivalence classes are in fact machine siblings, i.e. all of the IPs within the equivalence class belong to the same machine as inferred by our timestamp method. Another 38% of the equivalence classes have at most two groups of equipment siblings. Fewer than 1% of the actively collected siblings correspond to 3 or more machine sibling groups.

Next, we examine the size of the machine sibling groups within the active DNS equipment equivalence classes. Figure 11 shows the cumulative fraction of DNS equivalence classes versus the ratio of the largest inferred machine sibling group to the size of the original equipment equivalence class. For example, in the aforementioned equivalence class, the ratio is $0.76 = 131/172$.

We see that for approximately 50% of the equipment equivalence classes, the ratio is 1.0, indicating that the equivalence class is covered by a single machine sibling group. Approximately 45% have a ratio of 0.5, meaning that the largest machine sibling group accounts for half of the total IPs within the equipment equivalence class.

4 Discussion

In this Section, we discuss additional details and implications of our methodology and results. In particular, we examine temporal effects, AS partitioning of IPv4 and IPv6 addresses, and root causes for the large equivalence classes we observe.

4.1 Temporal Effects

As address associations are collected over time, the equivalence classes change too. For example, suppose the system reports association ($v4'$, $v6'$). If neither addresses is present in the existing equivalence classes, we create a new equivalence class containing just this pair. If only one of these addresses was observed previously, an existing equivalence class is increased. If both addresses are previously known and are in the same equivalence class, then possibly a new edge is added to the graph. Lastly, if both addresses are previously known, but are in different equivalence classes, then the two prior equivalence classes are merged into one, with this new address pair forming the joining edge.

If an address pair remains in a 1-1 class over a period of time (six months for the passive dataset), then we gain confidence that these candidate siblings are at least equipment-siblings. Equipment-siblings also likely pertain to the *m-n* equivalence classes that collapse to *1-1* when the addresses are aggregated (after) to prefixes – note the increase in percentage from 14% to 39% in Table 1.

Similarly, with the targeted fingerprinting technique, a common clock skew may imply machine-siblings, although with two physical interfaces on a single machine that are separately assigned IPv4 and IPv6 addresses. We hope to refine our notion of equipment and machine siblings as we gain more insight into such relationships.

4.2 AS's of DNS address pairs

In contrast to the fingerprinted address pairs of Alexa (§3.5), for which there is not an implied coupling, we might expect a stronger AS match between address pairs of the nameservers involving DNS resolution. However, of the resolver address pairs in the dataset (§3.2), excluding those whose IPv6 address is 6to4 or Teredo, 31% have addresses in distinct ASes, as opposed to 18% of the Alexa sites. If we also exclude those pairs whose IPv6 address is in AS 6939, an IPv6 tunnel broker [14], the percentage is still 25%. Manual checks on the more popular pairs with different ASes reveal companies using separate ASes for IPv4 and IPv6, or those likely having a business relationship, where possibly the network equipment is owned by one company, but one of addresses is registered with the other. The caveats in §3.5 again apply.

4.3 Large Equivalence Classes

There are a variety of potential reasons why address pairs may form equivalence classes larger than *1-1*:

1. An interface with more than one IPv4 and/or more than one IPv6 address.

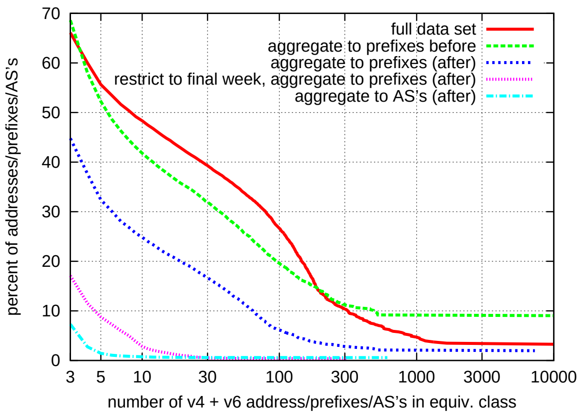


Figure 12: Percent of v4 + v6 address/prefixes in equivalence classes of at least a given size

2. Either IPv4 or IPv6 is behind a Network Address Translation (NAT) device [34], in which case we observe multiple addresses of the non-translated protocol, and one or two from the other.
3. Carrier grade NAT [19].
4. Middleboxes and load balancing, e.g. [2].
5. Auto-tunneling such as 6to4 and Teredo [8].
6. Hosts and devices that vary the lower order bits of the IPv6 address [28].
7. Public DNS services, e.g. [13].
8. Open, recursive nameservers [5].
9. Address changes over time [36].
10. Mechanisms that subvert the multi-level hierarchy, such as manual or automated probes from different locations directed to specific nameservers.
11. Shared caches. If NS records returned from the first-level are saved in a shared cache (with TTL of 12 hours for the present data set) and subsequently used by another nameserver that sends a query over v6, then the second-level will discover multiple v6's associated with a single v4. A short TTL on the final A or AAAA record (20 seconds in the data set) increases reuse of the cached NS record. Furthermore, after the NS TTL expires and some other nameserver using the shared cache does the lookup at the first-level, then another v4 address could be added to the equivalence class. Considering subsets of the six-month data set, e.g. the final week, the classes are simpler, though still complex.

Figure 12 is analogous to Figure 7 except that it analyzes the passive-DNS data set. Note the larger range on the x-axis as compared with Figure 7. Figure 12 shows that although the aggregation to prefixes (after the computation of the equivalence classes), the dark-blue line, substantially reduces the size, still 20% of the prefixes are in equivalence classes of size 19 or more and 10% in equivalence classes of size 64 or more. When we restrict the data to the final week, and aggregate to prefixes, there is again a significant reduction, though it is unclear how much is due to simply less data, and how much to removal of old associations that are no longer active. When we aggregate the full data set to AS's, the equivalence classes become rather small, except, as was the case the active-DNS measurement, for a single large one, which here is of size *412-200*. Again, as before, the dominant AS's are 15169 and 6939.

A strong trend in our investigation is that with additional data collection, greater complexity appears. This is indicated in Figure 6 where repeated lookups by the same set of open resolvers reveal new nameserver addresses, and the resulting equivalence classes become larger as the number of lookups goes from 1 to 32. This is also indicated in Table 1 and Figure 12 where the set of prefixes seen in the final week is substantially less than over the sixth-months, and the equivalence classes are substantially smaller (though for this longer time span some of the address pairings may no longer be active). The new nameservers that are seen are often, but not always, within the same /24 or /64 as those previously observed, as one would expect from a bank load balanced nameservers.

5 Conclusions

This paper examines the relationship between IPv4 and IPv6 addresses of Internet server infrastructure. Our primary contribution is a methodology for characterizing the inter-relation of IPv4 and IPv6 among Internet DNS and web servers. We deploy both active and passive measurement techniques to discover groups of equipment equivalence classes, and then tie the techniques together with physical TCP fingerprinting in order to discover more granular machine equivalence classes.

While prior work explores IPv6 client adoption and penetration, to our knowledge this paper is the first to take a comprehensive look at the server-side where IPv6 deployment is active [10, 1]. Characterizing server IPv6 addresses and their relation to IPv4 is important in: i) tracking the evolution of IPv6; ii) understanding the potential for correlated failures and security risks when IPv4 and IPv6 services are physically colocated; and iii) preventing erroneous Internet measurements intending to compare the performance of IPv4 and IPv6 paths.

We develop and deploy three novel measurement systems: i) a passive DNS collection using a two-level DNS hierarchy that encodes IPv4 addresses within IPv6 nameserver records; ii) an active DNS probing system that induces a combination of IPv4 and IPv6 DNS resolver lookups in a single resolution operation and can also force resolvers to utilize TCP; and iii) an active TCP physical device fingerprinting technique that more precisely identifies IPv4 and IPv6 addresses present on the same machine.

We find significant complexity, as measured by large equivalence classes in both the active and passive data sets, between IPv4 and IPv6 associations. Much of this complexity is attributable to large DNS resolver clusters used by large providers. Further complicating "clean" association of addresses are instances where operators employ shared caches, load balancing, NAT, carrier grade NAT, IPv6 address randomization, or mixtures thereof.

While we examine servers in-depth, the relationship between IPv4 and IPv6 router addresses is an important infrastructure component we plan to address in future work to better understand topological differences.

The primary implication of our work is an under-appreciated fact: that the IPv4 and IPv6 addresses of Internet servers frequently belong to different interfaces, machines, and even autonomous systems. We hope that our results illuminate not only some of the underlying complexity between IPv4 and IPv6 as deployed in the Internet today, but also properties to protect critical infrastructure and methodologies for conducting sound IPv4/IPv6 comparison measurements.

References

[1] IPv6 Implementors Conference, 2010. `https://sites.google.com/site/ipv6implementors/2010/agenda`.

[2] ABLEY, J. A software approach to distributing requests for DNS service, 2004. `http://ftp.isc.org/isc/pubs/tn/isc-tn-2004-1.html`.

[3] AKAMAI. Edgscape geolocation, 2012. `http://www.akamai.com/html/technology/products/edgescape.html`.

[4] ALEXA. Top 1,000,000 sites, 2012. `http://www.alexa.com/topsites`.

[5] ARENDS, R., AUSTEIN, R., LARSON, M., MASSEY, D., AND ROSE, S. DNS Security Introduction and Requirements. RFC 4033 (Proposed Standard), Mar. 2005. Updated by RFC 6014.

[6] BELLIS, R. DNS Transport over TCP - Implementation Requirements. RFC 5966 (Proposed Standard), Aug. 2010.

[7] CAI, X., HEIDEMANN, J., KRISHNAMURTHY, B., AND WILLINGER, W. Towards an AS-to-organization map. In *Proceedings of the 10th annual conference on Internet measurement* (2010), pp. 199–205.

[8] CARPENTER, B., AND MOORE, K. Connection of IPv6 Domains via IPv4 Clouds. RFC 3056 (Proposed Standard), Feb. 2001.

[9] CLAFFY, K. Tracking IPv6 evolution: data we have and data we need. *SIGCOMM Comput. Commun. Rev. 41*, 3 (July 2011), 43–48.

[10] COMCAST. IPv6 Information Center, 2012. `http://www.comcast6.net/`.

[11] DEERING, S., AND HINDEN, R. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), Dec. 1998.

[12] ELZ, R., AND BUSH, R. Clarifications to the DNS Specification. RFC 2181 (Proposed Standard), July 1997. Updated by RFCs 4035, 2535, 4343, 4033, 4034, 5452.

[13] GOOGLE. Public DNS, 2012. `https://developers.google.com/speed/public-dns/`.

[14] HURRICANE ELECTRIC. IPv6 tunnel broker service, 2012. `http://tunnelbroker.net/`.

[15] HUSTON, G. IPv6 BGP Statistics, 2012. `http://bgp.potaroo.net/v6/as2.0/`.

[16] ISOC. World IPv6 Day, 2011. `http://www.internetsociety.org/ipv6/archive-2011-world-ipv6-day`.

[17] ISOC. World IPv6 Launch, 2012. `http://www.worldipv6launch.org`.

[18] JACOBSON, V., BRADEN, R., AND BORMAN, D. TCP Extensions for High Performance. RFC 1323, May 1992.

[19] JIANG, S., GUO, D., AND CARPENTER, B. An Incremental Carrier-Grade NAT (CGN) for IPv6 Transition. RFC 6264 (Informational), June 2011.

[20] KOHNO, T., BROIDO, A., AND CLAFFY, K. C. Remote physical device fingerprinting. In *Proceedings of IEEE Security and Privacy* (2005), pp. 211–225.

[21] KREIBICH, C., WEAVER, N., NECHAEV, B., AND PAXSON, V. Netalyzr: illuminating the edge network. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement* (2010), pp. 246–259.

[22] LUCKIE, M., AND HUFFAKER, B. IPv6 deployment: trends and tidbits of 4,800 dual-stack ases. In *CAIDA AIMS Workshop* (2012).

[23] LYON, G. F. *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. 2009.

[24] MEYER, D. University of Oregon RouteViews, 2012. `http://www.routeviews.org`.

[25] MOCKAPETRIS, P. Domain names - implementation and specification. RFC 1035 (Standard), Nov. 1987.

[26] MOHAN, R. Will U.S. Government Directives Spur IPv6 Adoption?, Sept. 2010. `http://www.circleid.com/`.

[27] MOON, S., SKELLY, P., AND TOWSLEY, D. Estimation and removal of clock skew from network delay measurements. In *Proceedings of INFOCOM* (mar 1999), vol. 1, pp. 227 –234.

[28] NARTEN, T., DRAVES, R., AND KRISHNAN, S. Privacy Extensions for Stateless Address Autoconfiguration in IPv6. RFC 4941 (Draft Standard), Sept. 2007.

[29] NOMINUM. Intelligent DNS, 2012. `http://www.nominum.com/`.

[30] NYGREN, E., SITARAMAN, R. K., AND SUN, J. The akamai network: a platform for high-performance internet applications. *SIGOPS Oper. Syst. Rev. 44*, 3 (Aug. 2010), 2–19.

[31] RIPE NCC. World IPv6 Day Measurements, 2011. `http://v6day.ripe.net`.

[32] SARRAR, N., MAIER, G., AGER, B., SOMMER, R., AND UHLIG, S. Investigating IPv6 traffic: what happened at the world IPv6 day? In *Proceedings of PAM* (2012).

[33] SILBERSACK, M. J. Improving TCP/IP security through randomization without sacrificing interoperability. In *Proceedings of BSDCan* (2006).

[34] SRISURESH, P., AND EGEVANG, K. Traditional IP Network Address Translator (Traditional NAT). RFC 3022 (Informational), Jan. 2001.

[35] VIXIE, P. Extension Mechanisms for DNS (EDNS0). RFC 2671 (Proposed Standard), Aug. 1999.

[36] XIE, Y., YU, F., ACHAN, K., GILLUM, E., GOLDSZMIDT, M., AND WOBBER, T. How dynamic are ip addresses? In *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications* (2007), pp. 301–312.

[37] ZANDER, S., ANDREW, L. L. H., ARMITAGE, G., HUSTON, G., AND MICHAELSON, G. Mitigating sampling error with measuring internet client ipv6 capabilities. In *Proceedings of the 12th ACM SIGCOMM conference on Internet measurement* (2012).