NPS-CS-13-002

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

INFERRING INTERNET SERVER IPV4 AND IPV6 ADDRESS RELATIONSHIPS

by

Robert Beverly
Arthur Berger
Nicholas Weaver
Larry Campbell

June 21, 2013

THIS PAGE INTENTIONALLY LEFT BLANK

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

RDML Jan E. Tighe
Interim President

Douglas A. Hensler
Provost

The report entitled *"Inferring Internet Server IPv4 and IPv6 Address Relationships"* was prepared for and funded by National Science Foundation.

Further distribution of all or part of this report is authorized.

This report was prepared by:

Robert Beverly

Arthur Berger

Nicholas Weaver

Larry Campbell

Reviewed by:

Peter Denning, Chairman
Department of Computer Science

Released by:

Jeffrey D. Paduan
Dean of Research

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
21–6–2013	Technical Report	2012-05-01—2013-05-01

4. TITLE AND SUBTITLE

Inferring Internet Server IPv4 and IPv6 Address Relationships

5a. CONTRACT NUMBER

5b. GRANT NUMBER

CNS-1111445

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Robert Beverly, Arthur Berger, Nicholas Weaver, Larry Campbell

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

NPS-CS-13-002

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Science Foundation
Arlington, VA 22230

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

The views expressed in this report are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

14. 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 two novel techniques for finding DNS resolver sibling groups, one passive and one active. For 674k observed (IPv4, IPv6) address pairs, we find that 34% of the addresses are one-to-one, i.e., appear in no other pair. Yet there are also complex cases, where distributed DNS resolution creates interconnected sets of nameserver address pairs that can span continents and autonomous systems, complexity confirmed using active probing. 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, geolocation and performance measurement.

15. SUBJECT TERMS

IPv6 measurement, IPv4 to IPv6 association

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Robert Beverly
Unclassified	Unclassified	Unclassified	UU	53	19b. TELEPHONE NUMBER *(include area code)* (831) 656-2132

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

THIS PAGE INTENTIONALLY LEFT BLANK

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 two novel techniques for finding DNS resolver sibling groups, one passive and one active. For 674k observed (IPv4, IPv6) address pairs, we find that 34% of the addresses are one-to-one, i.e., appear in no other pair. Yet there are also complex cases, where distributed DNS resolution creates interconnected sets of nameserver address pairs that can span continents and autonomous systems, complexity confirmed using active probing. 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, geolocation and performance measurement.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1: Introduction

After languishing for decades, IPv6 [13] is experiencing renewed interest [18], in large part due to technical, economic, and political reasons [11, 30]. 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 will depend on IPv4, where hosts and infrastructure are commonly "dual-stacked." Understanding the relationship between, and interdependence 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 [35, 43]. However, less attention has been placed on characterizing IPv6 within Internet server infrastructure. Several initiatives [19, 20] and anecdotal evidence [36] suggest that IPv6 infrastructure deployment is proceeding well in advance of client adoption. For instance, large residential service provider networks [12], 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). For 674k observed (IPv4, IPv6) address pairs we find that 34% of the addresses are one-to-one, i.e., appear in no other pair.

This percentage increases to almost 50% when we aggregate IPv6 addresses into /64s. And there are also complex cases, where distributed DNS resolution creates interconnected sets of nameserver address pairs that can span continents and autonomous systems. To validate and better understand these passive results, we perform active DNS measurements. We find 73% of address pairs discovered via active probing also exist as pairs within our passive data. 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.

Associating IPv4 and IPv6 addresses, and finding infrastructure siblings has several important motivations:

1. ***Internet Evolution:*** As IPv6 deployment increases (or decreases), sibling associations will evolve. Our sibling discovery system can aid in tracking how server infrastructure of 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. Understanding the adoption and use of IPv6 over time is therefore important to policy and network operation, as well as to network researchers.
2. ***Measurement Research:*** Measurement research that aims to understand various performance and topological aspects of the IPv6 network, as compared to the existing IPv4 Internet [25, 35], 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.
3. ***Security:*** IPv6 presents several practical security challenges. Much of the IPv6 infrastructure is insecure in comparison to IPv4. Yet, not only are many old IPv4 attacks feasible in IPv6, IPv6 introduces new attack vectors [16]. The extent to which IPv4 infrastructure depends on IPv6, and vice-versa, therefore has unknown security and critical infrastructure protection implications. For example, whether an attack against the IPv6 address of an Internet web or DNS server impacts an organization's corresponding service for IPv4 depends on the sibling relationship. Further, siblings imply the potential for *correlated failures* that impact resilience. Whether IPv6 infrastructure is deployed on the same equipment that supports IPv4 is an important hypothesis we test in this work.
4. ***Reputation:*** Mapping IPv4 addresses to a reputation has emerged as an important security and anti-fraud feature, with several public and commercial offerings. For example, spam, malware, and phishing block lists (e.g., [38]) provide a binary indicator of whether an IPv4 address belongs to an abusive or compromised host, while other services pro-

vide reputation as a means to combat e-commerce fraud (e.g., [27]). Associating IPv6 addresses with their IPv4 counterpart provides one means to understand the reputation and past behavior of an IPv6 host.

5. ***Geolocation:*** Significant effort is spent on physically locating IPv4 addresses [41], thus, in the same vein as reputation, the geolocation of an IPv6 address can be informed by its associated IPv4 address if known. This last motivation is perhaps the most relevant to the present work, as the system described herein is used, *in a production capacity*, on the Akamai content distribution network in order to improve IPv6 geolocation [5].

Toward better understanding the relationship between IPv4 and IPv6, we make five 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 whether *targeted* candidate IPv4, IPv6 addresses are machine siblings.
4. Analysis of observed DNS resolver address pairs that finds a significant proportion, 34%, have a one-to-one association (almost 50% when the IPv6's are aggregated to /64 prefixes) and also finds complex, connected sets of address pairs.
5. Real-world deployment of the techniques to gather DNS and website siblings and equivalence classes.

The remainder of this report 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.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 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 Passive, Opportunistic DNS Technique

Clients rely on local recursive resolvers to perform DNS resolution. We seek to ascertain the IPv4, IPv6 addresses of dual-stacked DNS resolvers by inducing them to reveal their addresses as part of their natural lookup process. While this technique is passive and opportunistic in that a client resolver must contact our authoritative DNS, our deployment on the Akamai CDN permits collection of a large number of associations.

Our technique exploits: 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. Assume we control, and deploy our technique on, the DNS authority of `example.com`.

Consider a recursive DNS resolver servicing a local client requesting resolution of `www.a.example.com`. As depicted in Figure 2.1, the resolver requests resolution (typically an `A` record) for this host 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 [29]. Crucially, the `AAAA` records of the second-level `NS`, as returned by our first-level nameserver, 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.*

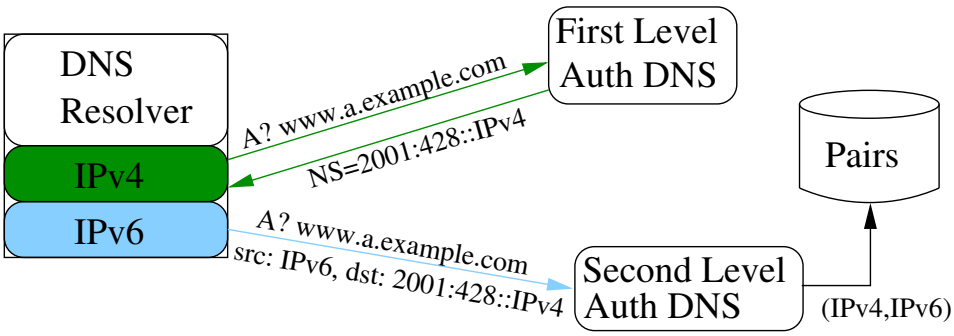


Figure 2.1: DNS Resolver IPv4/IPv6 Address Inference: A multi-level authority returns second-level nameserver `AAAA` records encoding a query's IPv4 source in the lower-order bits. The second-level nameserver associates IPv6 sources with the IPv4 encoded in the destination IPv6.

For example, Figure 2.1 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 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 as the second-level DNS accepts queries from an entire prefix; we use a /80 prefix.

Akamai Technologies has deployed this passive technique within its production infrastructure [34], making use of its pre-existing multi-level DNS hierarchy (whose primary purpose is to resolve names to CDN addresses that can best serve the client), and has activated it for a selection of domains. Akamai has used the resulting data collection during the revision of its request-routing software to support IPv6, and continues to use it as one of the inputs into Akamai's geolocation product, [5] where the extensive prior knowledge of IPv4 can be used to aid in the location of IPv6.

As a practical issue, the first-level authoritative DNS in Figure 2.1 does not know whether the client's nameserver is IPv6 capable, as the incoming DNS query arrives over IPv4. Thus, in Akamai's implementation, of eight `NS` records returned, only one or two of the associated glue records are `AAAA`. Also, even when the client's nameserver is dual stacked, it may not select the `AAAA`; selection order typically depends on historical load and performance characteristics.

Note that this technique is passive. The collection of address pairs is simply a by-product of the client's nameserver resolving a domain under our authority that the client requested. Since the entity collecting the data is not initiating the DNS query, the technique does not require that the nameserver be an open resolver.

Also note that this technique only checks whether the client's resolver can send IPv6, but does not check whether it can receive IPv6. If it can not, the query will timeout, and the resolver can fallback to use an IPv4 glue record, yet an (IPv4, IPv6) pair is still logged. This ambiguity does not occur with the active technique of §2.3.

Deployment on Akamai affords us a rich and diverse dataset; however the general technique could be implemented on other authoritative nameservers under common control along a DNS namespace hierarchy.

Advantageous Complications

Figure 2.1 illustrates the simple case where the DNS queries come from a single, dual-stacked machine. However, reality can be much more complex. Recall that the (IPv4, IPv6) addresses that are logged are addresses seen by the authoritative nameservers. They need not be associated with the same machine. It is well-known that DNS servers and resolvers experience significant load, and that many approaches exist for balancing that DNS query load [3, 4, 15, 33]. A rel-

atively common case is that a provider maintains a set of servers at a location that share the load, and also may share a cache or may distribute obtained resolutions. Further, DNS resolver implementations may perform record pre-fetching [4] to improve performance.

In Akamai's implementation, the NS records returned by the first-level Authoritative DNS have at TTL of 12 hours. Thus, if there is shared cache or if the results are distributed, then another nameserver may subsequently use them and send a query over IPv6, in which case the second-level will discover multiple IPv6's associated with a single IPv4. A short TTL on the final `A` or `AAAA` record (20 seconds in the Akamai implementation) increases reuse of the cached `NS` record. Furthermore, after the `NS` TTL expires, if some other nameserver in the cluster does the lookup at the first-level, then another IPv4 address could be seen by the authoritative nameserver. In which case the data set will contain multiple pairs which have IPv4 or v6 addresses in common, all of which are associated with a given cluster of DNS servers at a given location. This is an advantageous complication from a number viewpoints. For geo-location, one has discovered additional IPv6 addresses whose location can bootstrap from prior knowledge of IPv4. For CDN's that wish to associate clients to nameservers, one might generalize an observed set of (client address, nameserver address) pairs to any of these clients potentially being associated with any of these nameservers. From the viewpoint of survivablity, or robustness, one has gained some insight into how the ISP has chosen to architect its DNS.

However, clusters of DNS servers complicate the discovery of pairs of (IPv4, IPv6) addresses that are on the same interface or machine.

However, in §2.2 we present an active fingerprinting technique that seeks to determine whether or not candidate address pairs are indeed on the same machine, given one can send DNS queries from a vantage point for which the candidate resolvers will respond, which always pertains in the special case of open resolvers.

Another complication that occurs is when there are intermediary machines between the client's nameserver and the authoritative first and second-level nameservers. A common case is when a public DNS, e.g., [2], is used for the IPv6 but not the IPv4, or vice versa. The presence of intermediary machines is a negative complication from the viewpoint of wishing to associated addresses to a common machine or set a machines at a given location. Though, it is interesting from the viewpoint of discovery of DNS architecture and how the DNS is being used, as discussed in §3.1.

Note that when using the discovered (IPv4, IPv6) pairs for geo-location one needs to take into account that the data set will likely contain misleading information. However, this is not a new problem, but rather is a well-known attribute of many of the inputs used for geo-location; and one must sort, weigh, and filter conflicting information from multiple sources.

Example Nameserver Pairs

To illustrate the potential complexity of the associations we gather passively, we present several example nameserver pairs in this subsection. Often (to be quantified in §3.1) a given IPv4

address and a given IPv6 appear in only one pair. For example, the two addresses in the pair:

```
(119.63.216.69,  2402:7400:0:c::5)
```

are only observed in this pair. However, just as common are more complex relationships. For example the IPv6 address `2001:380:515:1::201` is observed in two pairs:

```
(210.227.79.198,  2001:380:515:1::201)
(210.227.79.230,  2001:380:515:1::201)
```

and these two IPv4 addresses are only seen for this IPv6. Similarly, the IPv4 address `120.119.28.2` is observed in four pairs:

```
(120.119.28.2, 2001:e10:c41:49:c5e5:281b:44f8:80bd)
(120.119.28.2, 2001:e10:c41:51:226:b0ff:fedc:f970)
(120.119.28.2, 2001:e10:c41:1::2)
(120.119.28.2, 2001:e10:c41:59:cabc:c8ff:fe92:e8d6)
```

and these four IPv6 addresses are only seen in these pairs. The associations can become more involved, for example the following six pairs:

```
(193.137.16.65,  2001:690:2280:1::65)
(193.137.16.65,  2001:690:2280:801::135)
(193.137.16.75,  2001:690:2280:801::135)
(193.137.16.75,  2001:690:2280:1::75)
(193.137.16.145, 2001:690:2280:801::135)
(193.137.16.145, 2001:690:2280:801::145)
```

form a connected set of associations, in that starting at any address and following the associations, one would eventually touch each of the addresses in this set.

Such sets of address pairs can be complex. In the 6-month data set there is one particularly large group consisting of about a third of the 674,000 address pairs, and spans hundreds of autonomous systems (ASes) and multiple continents. It occurs, at least in part, due to open, public nameservers such as GoogleDNS, and is of interest from the viewpoint of how DNS is being used, and is discussed in Section 3.1.

Bipartite Graph and Equivalence Classes
We abstract the observed nameserver pairs as a bipartite graph. The IPv4 addresses form one column of vertices. And the IPv6 addresses form the second column of vertices. And each of the observed address pairs is an edge of the graph. The graph is not fully connected, but rather consists of many subsets of connected edges.

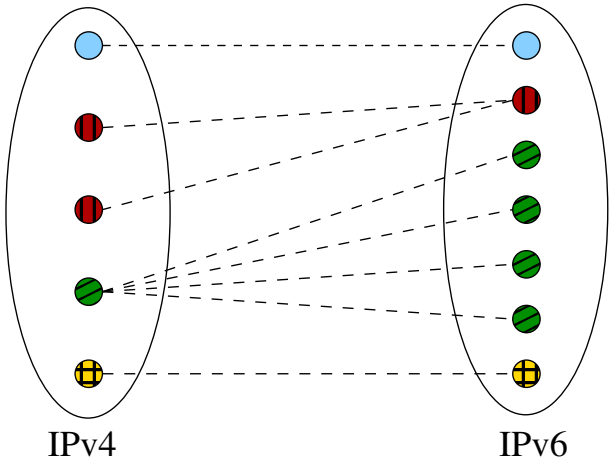


Figure 2.2: Equivalence Classes: The associations (edges) between IPv4 and IPv6 addresses form a bipartite graph. The connected components form equivalence classes of various sizes. In this hypothetical example, the nodes in each equivalence class are shaded uniquely. Two of the four equivalence classes are 1-1 (one IPv4, one IPv6 node), one is 2-1, and one is 1-4. Note that 4 of the 12 addresses, 33%, are in the 1-1 equivalence classes.

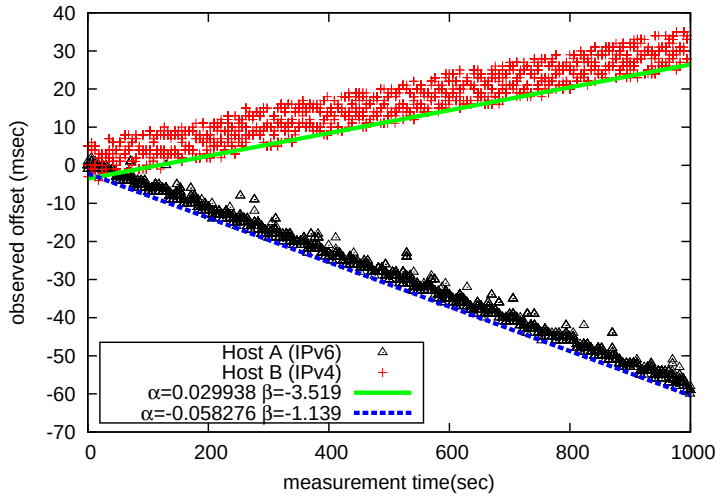
We call a set of connected edges (and the address pairs they represent) an "equivalence class," abbreviated "eq. class." Figure 2.2 illustrates a mini version of the graph, consisting of five IPv4 addresses, seven IPv6 addresses, and eight address pairs (edges). Let *m-n* denote an equivalence class containing *m* IPv4 and *n* IPv6 addresses. In Figure 2.2 the address pairs partition into four equivalence classes, two of which are *1-1*, one is *2-1* and one is *1-4*.

As address associations are collected over time, the equivalence classes change too. For example, suppose the system reports association ($v4'$, $v6'$). If neither address 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.

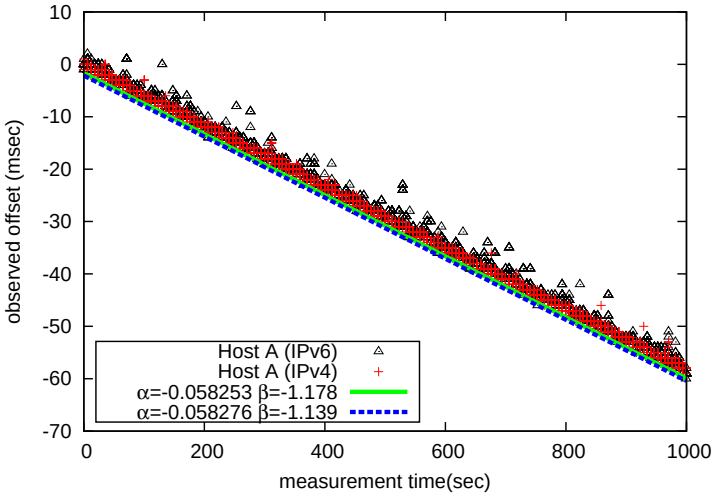
2.2 Targeted Fingerprinting Technique

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 [26] 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 [23]. While this technique



(a) False Siblings



(b) True Siblings

Figure 2.3: Targeted machine-sibling detection via TCP timestamp skew inference.

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 [21]. 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 [23], we compute the *observed offset* of each packet over time.

Given a sequence of offsets, we adopt the linear programming solution in [31] 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 2.3(a) and 2.3(b) graphically illustrate our approach using two hosts for which we know their ground-truth interface addresses. Figure 2.3(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 2.3(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 connection, e.g., a remote web server or a remote DNS server via our truncation method which we discuss next.

2.3 Active DNS Technique

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

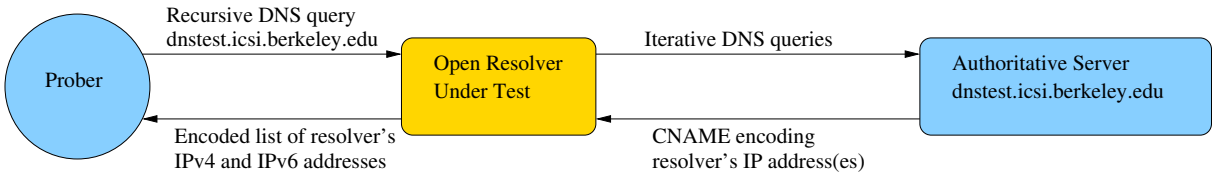


Figure 2.4: Active DNS Probing: We control both DNS probes and the authority of the DNS records being probed (shaded blue), thereby permitting testing of open DNS recursive resolvers in our dataset. The authoritative server is specially designed to: 1) force the recursive resolver to alternate between IPv4 and IPv6 network transport; and 2) encode the resolver's source IP addresses. The result of our DNS probes is a specially encoded list of the resolver's IPv4 and IPv6 addresses.

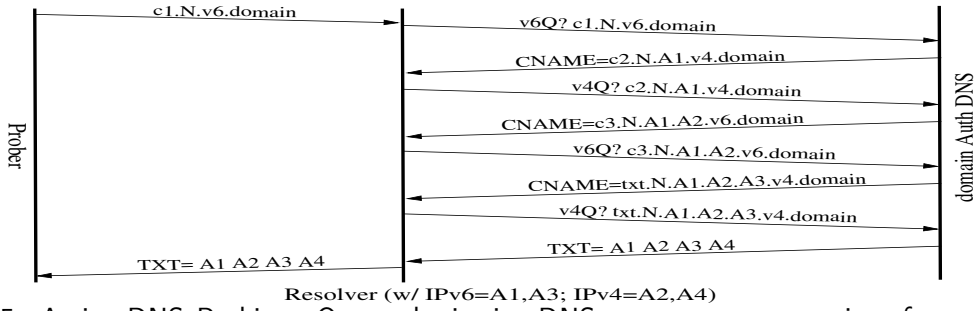


Figure 2.5: Active DNS Probing: Our authoritative DNS server returns a series of `CNAME` results with alternating IPv6 and IPv4 glue. We probe a resolver under test for our special domain, including a nonce N. State is maintained on addresses the resolver uses by encoding the IPs along the chain. The final result is the sequence of IPv4, IPv6 addresses used by the resolver (here $A1$,$A2$,$A3$,$A4$).

devices. We obtained about 500 addresses of such NATs from data collected by Netalyzr [24] which were both open *and* forward to a resolver supporting IPv6[1]. Additionally, ≈6600 of the IPv4 addresses and ≈2700 of the IPv6 addresses associated with the passively collected DNS resolvers are open and also respond to external requests. Thus there exist a meaningful pool of systems that process recursive DNS requests, either on their own or by forwarding to their configured recursive DNS resolver.

We actively probe these open DNS resolvers, issuing specially crafted queries for DNS records for which we are authoritative. In this way, we control both the DNS probes and the authority of the DNS records being probed, thereby permitting testing of open DNS recursive resolvers in our dataset. Figure 2.4 depicts the high-level active DNS probing methodology.

Our authoritative domains are served by a custom DNS server that is standards compliant [14]. Our authoritative server listens on both IPv4 and IPv6, uses both TCP and UDP. However, our server returns different results depending upon the incoming request. The server handles multiple domains that support either IPv4 or IPv6 requests, where the choice of domain impacts the IP protocol used by a recursive resolver. We can also force a resolver to connect to our DNS server via TCP by replying with `TRUNCATE` to UDP requests [8], which signals to the resolver

[1]There are several orders of magnitude more open and forwarding resolvers on the Internet. However, we are only interested in the subset of those which also support IPv6.

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 issue a series of queries that alternate between IPv4 and IPv6 for network transport. We maintain state between requests by specially encoding the returned results such that the final response to the recursive query is a "chain" of IPv4 and IPv6 addresses used by the resolver under test. Figure 2.5 provides an example timing diagram of the interaction of our prober and an authoritative DNS server with an open resolver whose addresses we wish to infer. In this example, the resolver has IPv6 addresses=$A1, A3$ and IPv4 addresses=$A2, A4$.

The prober queries the open resolver or forwarder for a single `TXT` record. The resolver can only fetch this name using IPv6, but instead of returning the record's 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.v4.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 [29], 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 [40].

Note that our methodology includes several techniques to ensure accuracy. First, each query includes a nonce that prevents effects due to DNS caching. Second, all state is maintained in the queries themselves, thereby removing the potential for miscorrelation of IPv4 and IPv6 addresses. For example, to infer a set of IPv4 and IPv6 addresses used by the Google public DNS resolvers, we query:

```
dig +short TXT @8.8.8.8 \
 cname1e6464.n123.v6.dnstest.icsi.berkeley.edu
```

where the GoogleDNS resolver, after stepping through the CNAME's, finally sends a query for the `TXT` record:

```
txt.n123.2607yf8b0y4004yc00yy153.64x233x168x86. \
2607yf8b0y4004yc00yy156.v4.dnstest.icsi.berkeley.edu.
```

The authoritative DNS, noting the three addresses contained in the requested domain, and noting the source address of the incoming query, then returns the TXT record that includes the nonce and the sequence of addresses that contacted our server to resolve the request:

```
"n123" \
"2607:f8b0:4004:c00::153" "64.233.168.86" \
"2607:f8b0:4004:c00::156" "64.233.168.85"
```

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. Thus, one can repeat the active DNS probes multiple times in order to gain a more complete picture of cluster structure when present. Since the DNS specification [14] 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.

[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.

CHAPTER 3:
Results

This section analyzes results from deploying the aforementioned techniques on the IPv4 and IPv6 Internet.

3.1 Passive Discovery of DNS Siblings

We examine (IPv4, IPv6) 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. Recall that we define an "Equivalence Class" to be a set of connected edges (address pairs) in a bipartite graph (§2.1). A *m-n* eq. class consists of *m* IPv4 and *n* IPv6 addresses, and a *1-1* eq. class is the simple, canonical case of an address pair where neither address has been observed in any other pair, and is consistent with (but not a guarantee of) the two addresses being assigned to a common resolver.

Figure 3.1 is a scatter plot of all eq. classes, using the relative frequency for color, of the eq. classes. Figure 3.1 shows the small eq. classes: *1-1*, *1-2* are the most popular, and shows there exists a broad range of sizes, with larger ones often representing a singular occurrence. We focus first on the *1-1* eq. classes, and then examine the larger ones.

Focus on 1-1 Equivalence Classes

The canonical case of a simple dual-stack server with a single routable IPv4 and IPv6 address would be observed as a *1-1* eq. classes. Though the converse need not hold as a *1-1* relationship may also be the public facing portion of a more complicated architecture. From the viewpoint of finding candidate dual-stack servers more *1-1* eq. classes is better.

As shown in the first row of Table 3.1, there are 553,000 addresses (IPv4 plus IPv6) in the 6-month dataset, and 34% of them eq. classes. Although, not a majority, it is still a substantial portion, and increases to almost 50% when we consider aggregation to prefixes.

Aggregation to Prefixes and AS's Frequently, multiple addresses of a non-*1-1* equivalence class reside in a given network prefix. Therefore, for each equivalence class, we examine aggregating addresses by prefix, thereby forming network-specific eq. classes. There is a trade-off regarding the chosen size of the prefix, where larger prefixes lead to more *1-1* eq. classes, but also increase the chance of pooling together unrelated equipment. For example, aggregating to prefixes as advertised by BGP would produce eq. classes with a natural association to the networks. However, the Network Operator that advertises a given prefix is not claiming that the addresses therein are located in a given Point of Presence, and are unlikely to be so for larger prefixes.

Here, we concern ourselves with *1-1* eq. classes whose addresses are likely candidates to be co-located. Therefore, we consider smaller prefixes to increase the chance of co-location. For IPv6,

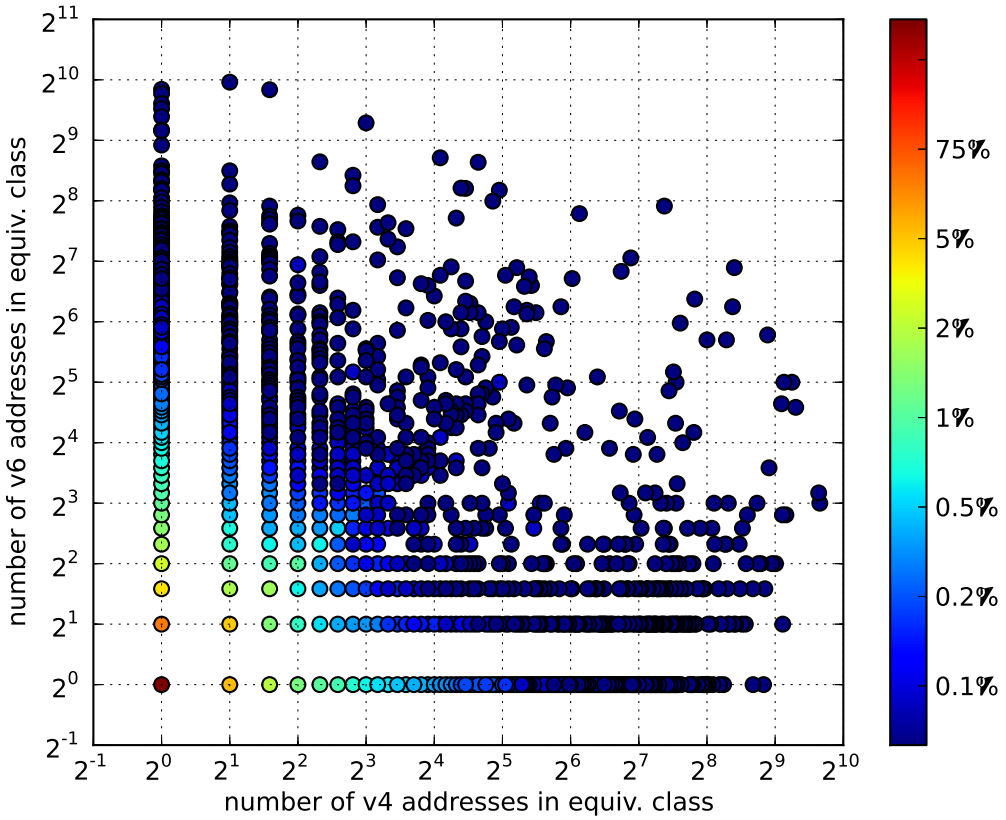


Figure 3.1: Scatter plot of number of IPv4 and IPv6 addresses in the equivalence classes

a natural choice is to aggregate across the interface ID in the lower 64 bits. Established practice is for IPv6 addresses in a given /64 to be on a common subnet, or may even be associated with a single interface where the interface ID is varied for anonymity, [32]. For IPv4 there is no natural choice for aggregation. Commonly, not always, operators assign to servers in a given rack (or subnet, or building) addresses that are close to one another in address space. A reasonable assumption (but for which there would be exceptions) is that addresses of servers in the same /30 would be in the same building. Assuming a /29 is a somewhat more aggressive, and so on. Fortuitously, given the arbitrariness in choice of prefix, we find that for the *1-1* eq. classes there is an insensitivity, at least up to /24's and /48's.

As an example of aggregation, the following five pairs:

```
(122.1.94.240, 2001:380:150::1053)
(122.1.94.240, 2001:380:11c::1053)
(122.1.94.240, 2001:380:11c::2053)
(122.1.94.242, 2001:380:11c::1053)
(122.1.94.243, 2001:380:11c::2053)
```

Table 3.1: Prevalence of 1-1 equivalence classes in the 6-month data set

Data Set	**Entity**	**Num of v4+v6 addresses / prefixes / AS's**	**% of v4+v6 addresses in 1-1 eq cls**
6-month data set	addresses	553,000	34%
6-month data set	prefixes - , /64	417,000	47.6%
6-month data set	prefixes /30, /64	396,000	47.9%
6-month data set	prefixes 27, /64	354,000	48.2%
6-month data set	prefixes /24, /64	280,000	48.6%
6-month data set	prefixes /24, /48	135,000	49.7%
6-month data set	prefixes /16, /32	29,000	62.4%
6-month data set	AS's	5,100	80%
restrict to last week	addresses	98,000	71%
restrict to last week	prefixes - , /64	80,000	79%
random subset; same num of pairs as in last week	addresses	84,800±400	51%±0.5%
random subset; same num of pairs as in last week	prefixes - , /64	70,500±400	65%±0.5%
Example in Fig 2.2	addresses	12	33%

form a *3-3* eq. class of addresses. When aggregated to prefixes /30, and /64 they become:

```
(122.1.94.240/30, 2001:380:150::/64)
(122.1.94.240/30, 2001:380:11c::/64)
```

a *1-2* eq. class of prefixes /30, /64. And when aggregated to AS's become `(AS-4713, AS-4713)`, a *1-1* eq. class of AS's.

For a more dramatic example, paired with `88.191.68.83` are 101 IPv6 addresses. All of the IPv6 addresses are in the single prefix `2a01:e0b:1:68/64`. The 6-month data set includes timestamps of when a given pair is first and last seen, and these IPv6's were almost always observed in disjoint 24-hour periods. This is likely an example where there is a single interface and the interface identification (the 64 lowest order bits) varies. Thus, whereas the addresses form a *1-101* equivalence class, after aggregating the IPv6's to /64 prefixes, we obtain *1-1* eq. class of prefixes.

The second through seventh rows of Table 3.1 report the aggregation to prefixes for different masks, where the equivalence classes are now of prefixes. When aggregating just the IPv6's and to /64's, the second row, the percent of addresses that are in the *1-1* eq. class jumps from 34% to 47.6%. When we also aggregate the IPv4's to prefixes, there is only a minor increase in the percentage even for aggregation to /24 and /48 respectively, which is rather aggressive from the viewpoint of addresses that are likely to be on the same subnet or location. (Aside: the

choice of prefix has a more noticeable impact for eq. classes greater than *1-1*; see §3.1.) Out of curiosity, when we do an extreme aggregation to /16 and /32 prefixes, we get that 62%.

From the viewpoint of finding (IPv4, IPv6) address pairs that are candidates for being on the same subnet, we observe that aggregating the IPv6's to /64's yields a useful number of addresses in the *1-1* equiv. class, almost 50%, and that there is little need for the somewhat more dubious step of also aggregating the IPv4's. See §**??** for further comments.

To better understand their distribution across networks, we lastly investigate aggregation to AS's. We exclude pairs where the IPv6 address is 6to4, as such addresses have ambiguous AS assignments. With 6to4 excluded, 80% of the addresses are in the *1-1* eq. class of AS's, Table 3.1. However, this includes the circumstance when the two AS's are not equal to each other. (This issue does not arise for prefixes, as the IPv4 is always distinct from the IPv6.) When the two AS's are different, by far the most common case is where the IPv6 AS is 6939, the tunnel broker Hurricane Electric. If we remove the address pairs of Hurricane Electric, then 51% of the remaining addresses are in the *1-1* of AS's where we impose the additional condition that the two AS's are equal to each other. Of the remaining *1-1* eq. classes of AS's where the two AS's are different, often the two AS's belong to the same organization. For example, of these AS pairs with distinct elements, the most popular AS is 7018, ATT. And of the AS's that are paired with 7018, 82% are AS 7132, SBIS-AS ATT Internet Services, and another 9% are AS 6389, BellSouth.net Inc. As an example of different organizations, when AS 3320, Deutsche Telekom, is in a *1-1* eq. class with a different AS, the most popular other AS is 8422 NetColonge, which suggest business relations besides what can be inferred from interconnection.

Restriction to a Shorter Observation Interval As the data set is a collection over a six month period, some address assignments and configurations likely changed during this period. These changes are reflected as larger eq. classes, as additional complexity, when in actuality, these large eq. classes may not have ever existed at a given moment in time. In this section, we examine this effect by considering a shorter observation interval.

However, when we restrict to a shorter time interval, we actually introduce *two* effects that reduce the larger eq. classes, and increase the proportion of addresses in the *1-1* eq. class: 1) less opportunity for changes in address assignments; and 2) a smaller data set. With a smaller data set, there is less opportunity for extant complexity to be revealed. (Imagine the extreme case of just a few observations, in which case large eq. classes are not possible.)

In order to separate these two effects, we also take random subsets of the full data set, where the subsets have the same number of pairs as present in the shorter observation interval. As the random subsets have address pairs from the full, 6-month observation interval, they exhibit just the first effect of a smaller data set.

The 6-month data set includes the time epoch the given pair was last observed. Thus, we can pick a time period, and then examine the subset of the pairs that are last observed during

that period. We pick the length to be a week, which is short enough that the opportunity for address reassignments is much less as compared with 6 months, and long enough that we get an appreciable number of pairs, to somewhat mitigate the impact of the smaller data set. Also, we pick the week to be final week of the 6-month period, as these are the pairs that are most active, and there are two to eight times as many pairs, 83,000, as compared with any other week on the six months.

Table 3.1 shows the results for the restriction to the final week, as well as for the random subsets, where for the latter, 10 subsets were taken and the table reports the mean values and the range. Table 3.1 shows that the fraction of addresses in the *1-1* eq. classes increases from 34% to 51% when we take the random subsets, which is the effect of a smaller data set. And the percentage increases further to 71% when we restrict to the final week, where the increment from 51% is the effect of reduced opportunity for changes in configuration. Likewise, when we consider the aggregation of IPv6's to /64 prefixes, the corresponding percentages are 47.6%, 65%, and 79%.

Other Measures for Popularity of Equivalence Classes

There are alternative ways to view the prevalence of *1-1* equivalence classes, besides the fraction of addresses. Other natural choices are: the fraction of equivalence classes, or the fraction of address pairs. For the mini graph in Figure 2.2, 2 of the 4 eq. classes (50%) are *1-1*, while 4 of the 12 addresses (33%) are in the *1-1* equivalence class, and 2 of the 8 address pairs (25%) are *1-1*.

For the 6-month data set, 77% of the eq. classes are *1-1*, 34% of the addresses are in the *1-1* equivalence class, Table 3.1, and just 14% of the address pairs are 1-1.

The percentage of eq. classes that are *1-1* is higher than for addresses or for pairs because each equivalence class is given equal weight when computing this percentage, and non-*1-1* classes contain more than two addresses and more than one address pair, sometimes hundreds.

The percent of addresses that are in *1-1* eq. classes is higher than for address pairs because, phrased informally, there are two addresses but just one pair in an *1-1* eq. class. Of the 673,784 address pairs, 93,832 (14%) are in a *1-1* eq. class. Whereas, of the 553,126 addresses, 2 * 93,832 (34%) are. More formally, the ratio of pairs to addresses is minimal, is 1/2, for the *1-1* eq. class, and increases for larger eq. classes. It is 2/3 for *1-2*, 3/4 for *1-3*, $n/(n+1)$ for *1-n*. For eq. classes with multiple IPv4's and IPv6's the ratio can be greater than 1. As discussed in Section 3.1 there is a huge eq. class where the ratio is 14.0.

In the following where we consider eq. classes greater than *1-1*, we want to talk about the size of an eq. class. In terms of addresses, the size of an *m-n* eq. class is naturally $m+n$. However, given the classification of *m-n* and when both *m* and *n* are > 1, there is not a natural definition in terms of number of pairs, because the number of pairs in an *m-n* eq. class can range between $m+n-1$ to $m*n$. For cases where, for a given *m* and *n*, there are multiple eq. classes in the data set, one alternative we examined was to count the number of pairs in each eq. class and then use the average for the size of the *m-n* eq. class. Another alternative is to drop the *m-n*

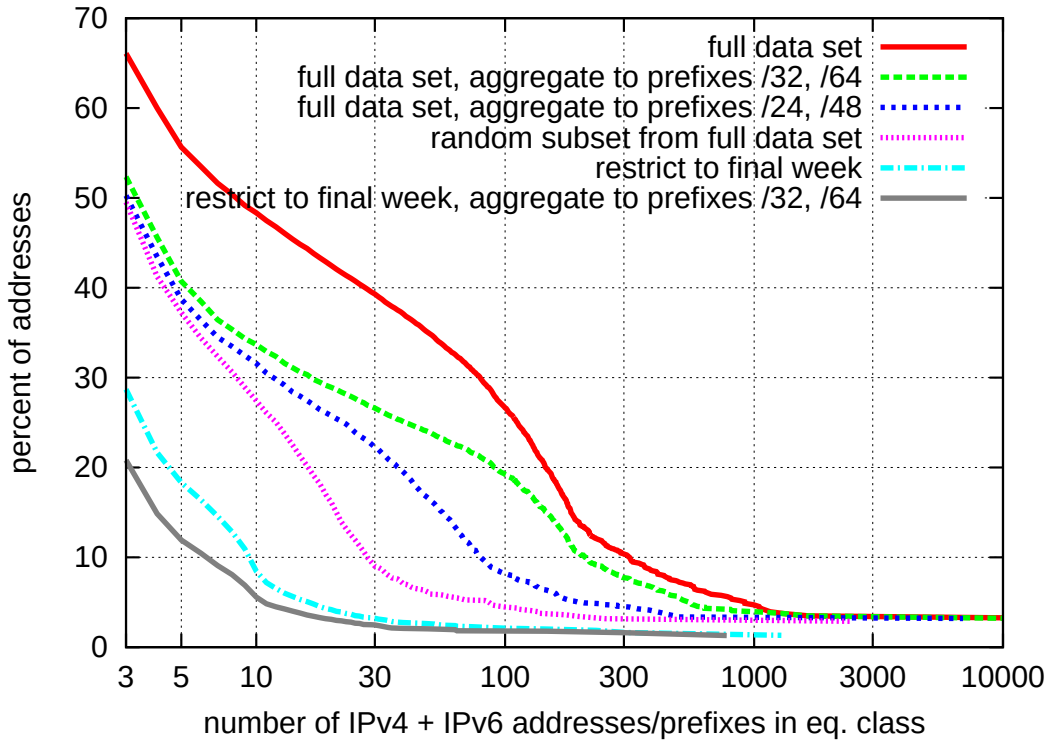


Figure 3.2: Percent of IPv4 + IPv6 addresses/prefixes in equivalence classes of at least a given size

classification, and just classify based on the number of pairs; or refine the classification with a third argument, *m-n-p*, where *p* is the number of pairs. In any case, for results such as Figure 3.2 we found that using address-pairs for the size of the eq. class did not lead to any additional insights beyond that obtained with using addresses. So, for simplicity in such figures we use addresses. When focusing on a particular eq. class, as in Section 3.1, we state both.

All Equivalence Classes

In this and the next three sub-sections, we shift the focus from the *1-1* eq. classes, and candidate address pairs that likely are on the same machine or subnet, to the non-*1-1* eq. classes.

Figure 3.2 shows the percent of addresses that are in eq. classes of at least a given size, where "size" is the sum of IPv4 plus IPv6 addresses (prefixes) in the eq. class. One can view Figure 3.2 as the complementary distribution of an address being in an equivalence class of a given size. For example, looking at the full data set, the red line, 40% of the addresses are in equivalence classes of size at least 27. While, when aggregating IPv6 to /64 prefixes, the green line, 40% of the addresses are in equivalence classes (of prefixes) of size at least 5. Note that with the x-axis beginning at 3, one can infer the percent of addresses in *1-1* eq. classes, e.g., the y-intercept of 66% means that 34% of the addresses are in an eq. class of size 2, as reported in Table 3.1. 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.

Figure 3.2 shows that although the aggregation of IPv6 to /64 noticeably reduces the tail of the distribution, still 20% of the addresses are in eq. classes of size 90 or more. As expected, with greater aggregation, to /24 and /48 prefixes, the tail of the distribution is further reduced. A more substantial reduction occurs when we take a random subset from the full data set, where the number of pairs in the subset equals the number of pairs observed in the final week, §3.1.[1] This reduction is due to the smaller sample size as compared to the full data set. That is, the additional observations in the full data set reveals substantial more complexity. We use the active technique of §2.3, to perform a more controlled data collection, over a short time period, and examine the additional complexity discovered with additional probing in §3.3. When we restrict the data to the final week, there is a significant reduction as compared with the random subset. Although the random subset has the same number of address pairs, it covers a period of six months, and thus there is a greater opportunity for changes in address assignments and configurations. Lastly, as expected, there is somewhat further reduction when the IPv6's are aggregated to /64's for the eq. classes of the final-week data set.

In all of the cases of Figure 3.2, there is a long tail. This is due to the tendency for there to be one eq. class that is much larger than the others, as discussed next.

"Mammoth" equivalence class

Here, we examine the largest equivalence class. As mentioned in §2.1, one of the equivalence classes is huge, *8037-9582*, which we call the "mammoth" eq. class – it contains 3% of the addresses and 37% of the address pairs. (The next largest eq. class contains 0.3% of the addresses and 0.2% of the pairs.) This huge eq. class, whose addresses span multiple continents and are in over 750 AS's, is of interest from the viewpoint of how DNS is being used and architected, and how such a diverse set addresses could be associated with one another (and clearly does not represent a set of load balanced nameservers at a single location).

The much larger percent of pairs, 37%, as compared with percent of addresses, 3%, occurs because a minority of the addresses are in many pairs. In terms of the bipartite graph, some of the vertices have many edges.

Figure 3.3 shows the complementary degree distribution of the vertices in the mammoth eq. class. That is, the percent of addresses (vertices) in the mammoth eq. class that are in at least a given number of pairs (edges). Figure 3.3 shows that the degree distribution for the IPv4 vertices is roughly the same as for the IPv6, though with a somewhat heavier tail. Although 54% of the addresses are in just one address pair, i.e., have degree one,[2] as seen from the y-intercept, 8% of the addresses are in at least 100 pairs, 4% in at least 300 pairs, and 1% in at least 500 pairs. As a comparison, for the full 6-month data set 89% of the addresses have degree one, and only 1% have degree at least 15.

[1] The plots for the different random subsets are very similar to one another.

[2] A degree one vertex in the bipartite graph does not imply that the vertex is in a a *1-1* eq. class. as the paired vertex can have degree greater than one.

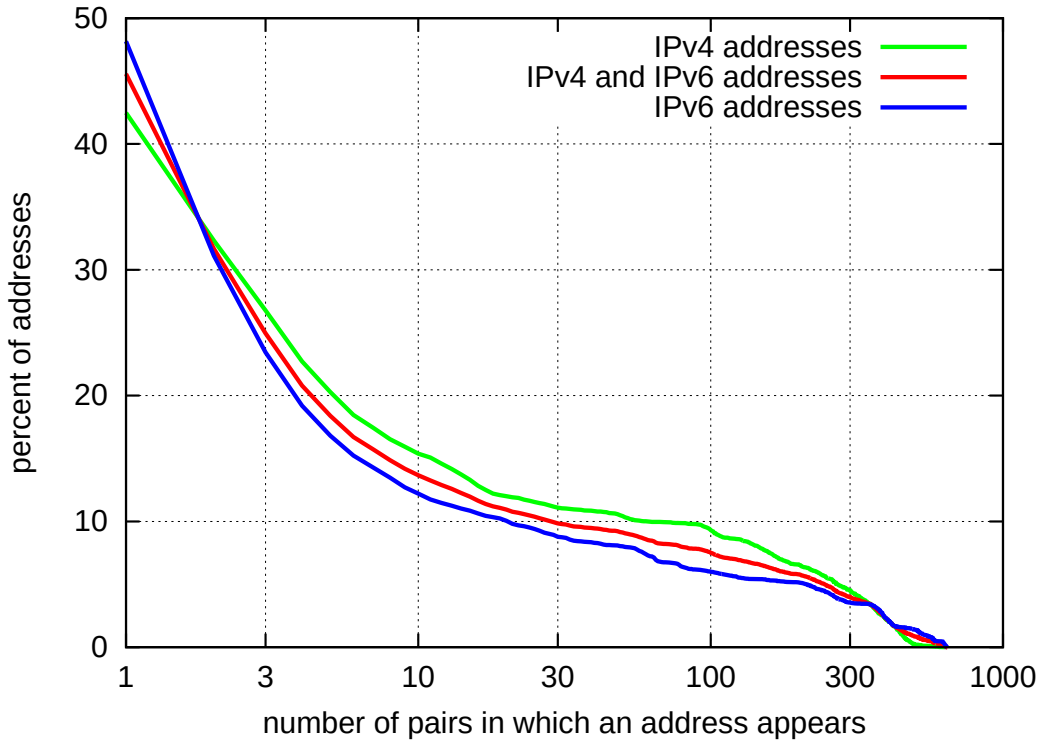


Figure 3.3: Percent of addresses in the "mammoth" eq. class that are in at least a given number of pairs

AS 15169, Google, plays a major role in this mammoth eq. class.[3] Although just 9% of the addresses in the mammoth eq. class are in AS 15169, 90% of the address pairs have one or both address(es) in that AS. 97% of the addresses that appear in at least 100 pairs are in AS 15169.

Whenever the IPv6 address is in AS 15169, then so is the paired IPv4. However, whenever the IPv4 address is in 15169, the paired IPv6 is not quite always in 15169 - 1% of the IPv6's are 6to4 and another 1% are scattered across a hundred different AS's. This is how addresses outside AS 15169 become a part of this eq. class. Some of these non-15169 IPv6 addresses may still be a part of Google's DNS architecture, assuming that Google has deployed servers in local ISP's with that ISP's addresses. And also, some may be separate from GoogleDNS - if the client's nameserver, using the public GoogleDNS for IPv4, obtained and cached the NS records returned by the first-level authoritative nameserver; and then for subsequent resolutions, sent a DNS query over IPv6 to the low-level authoritative nameserver.

6to4 addresses also play a significant role in the mammoth eq. class. Although 6to4 addresses are in only 6% of the pairs, they are 42% of the IPv6 addresses in this eq. class (an imbalance that is the opposite of AS 15169). Interestingly, for many of the 6to4 addresses, the embedded IPv4 address does not equal the observed paired IPv4. See §3.1 which discusses 6to4 in the full

[3]Other third-party DNS providers, such as OpenDNS and DynDNS, are present in the data set, but to a much lesser extent compared with GoogleDNS.

Table 3.2: Impact on the largest equivalence class of the 6-month data set when selected address pairs are removed

Action	After action, recompute eq. classes. Resulting largest eq. class contains: Num of IPv4 addr.	Num of IPv6 addr.	Num of addr. pairs
no change to data set	8,000	9,600	247,000
omit 6to4	4,000	5,100	230,000
omit AS 15169	5,600	4,900	18,000
omit 6to4 and AS 15169	900	2,000	3,400
random subset from full data set; same number of pairs as prior line	$2{,}623 \pm 214$	$2{,}626 \pm 335$	$78{,}778 \pm 603$

data set, not just the mammoth eq. class.

It is of interest to see what becomes the largest eq. class when, say, AS 15169 addresses are removed. Note that if one removes a set of addresses from the mammoth eq. class, the mammoth eq. class can partition into multiple smaller eq. classes - the remainder need not be a connected set of edges in the bipartite graph. Thus, we start with the full data set, and remove pairs that contain the addresses in question, and then with the reduced set of pairs, we recompute the equiv. classes and then note which is the largest one. Table 3.2 shows the results for selected removals.

As discussed in Section 3.1, some of the reduction is simply due to a smaller number of address pairs. Thus, ten random subsets from the full 6-month data set are taken, where the number of pairs in the subset is the same as in the fourth line of the Table. The mean value and range is reported in fifth line, and shows that the major cause for the lower complexity reported in line four is not due the smaller sample, but rather the selected pairs that are omitted.

Abnormal behavior with 6to4

Auto-tunneled [10] 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 the pairs where the IPv6 address is 6to4, 19% conform to the simple, canonical case where the embedded IPv4 address equals the paired IPv4 address, and the address pair is a *1-1* eq. class. Somewhat surprisingly, in 63% of the pairs where the IPv6 address is 6to4, the embedded IPv4 address is different from the paired address. Overall, any 6to4 traffic is probably a bug: Given a choice between an IPv4 address and a 6to4 address, a host should always prefer IPv4 unless executing a "happy eyeballs" strategy. Since the resolver also receives an IPv4 glue record, the resolver should prefer the IPv4 glue record, never attempting to connect to the DNS authority using 6to4.

Sometimes the 6to4 address is paired with hundreds of IPv4's. For example, `2002:aca8:101::aca8:101`

appears with 270 IPv4's. The embedded address is `172.168.1.1`, on AOL, and located in the USA. The paired IPv4's are mostly in Brazil, but also Germany, Saudi Arabia, and Turkey. This is not normal.

A likely explanation is some sort of buggy resolver software with a hardcoded 6to4 address. That we receive such 6to4 traffic also suggests that the 6to4 gateway processing this traffic isn't validating that the embedded IPv6 address corresponds to the IPv4 address. It is not likely to be an attack as, if an attacker took advantage of this, the query the attacker would employ would cause significantly more amplification than the name lookups present in an Akamai-ed domain.

Also, there are stranger cases where the 6to4 is again paired with many IPv4's but the embedded IPv4 is not routeable. For example 2002:6464:64cd::6464:64cd is paired with over 250 IPv4's and the embedded address is 100.100.100.205, or 2002:10::1:219:dbff:fef9:aa4f with embedded address 0.16.0.0. Again, this suggests buggy software misusing 6to4.

Although further investigation would be needed for a definitive explanation, it is clear that the collection of nameserver associations has turned out to have the additional feature of revealing abnormal activity.

IPv4 address within IPv6 address (non-6to4)
We mention briefly that of the few, 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.

We also confirm an interesting human-centric convention in at least one major ISP's DNS infrastructure that has been previously observed [16]. The operator assigns the lower 64 bits of the IPv6 address, using 16 bits to encode each octet of the IPv4 address, so that the hexadecimal values render as the decimal equivalent of the IPv4 address, such as `2001:558:1014:f:68:87:76:181`. which is paired with the IPv4 address `68.87.76.181`.

3.2 Validation Of Passive Technique

To assess the IPv4 to IPv6 associations inferred via the passive DNS technique, we examine two sources of external information about the resolvers: `PTR` records and software version.

DNS `PTR` records map an IPv4 or IPv6 address to a human readable name. We perform a DNS `PTR` query for every address in the 6-month dataset. For 14.5% of the pairs, both addresses have a `PTR` record. For these 97,821 pairs, Table 3.3 shows the percent of `PTR` record matches for the full data set, and then partitioned by those address pairs that are a *1-1* eq. class, and those in a larger eq. class. Of the addresses in *1-1* eq. classes, we see that 41% have `PTR` records that match exactly, as compared to only 1.5% of the non *1-1* eq. classes. Identical `PTR` records is supporting, though not definitive, evidence that the two addresses are either on the same server, or are the public-facing interfaces to a DNS server cluster.

Table 3.3: Agreement between associated IPv4 and IPv6 equivalence classes and their corresponding DNS PTR records.

Set	# Pairs with PTR	Exact Match	Second-Level Match
All	97,821	4821 (4.9%)	47,752 (48.8%)
1-to-1	8461	3500 (41.4%)	5679 (67.1%)
Non 1-to-1	89,360	1321 (1.5%)	42,073 (47.1%)

Table 3.4: Agreement between associated IPv4 and IPv6 equivalence classes and their corresponding bind.version.

Set	Num of Pairs with bind.version	Exact Match
All	8475	6951 (82.0%)
One-to-One	4650	4619 (99.3%)
Non One-to-One	3825	2332 (61.0%)

Next, we examine the agreement between the IPv4 and IPv6 `PTR` records for just the second-level domain portion of the name, e.g., `foo.bar.com` and `faz.bar.com` match at the second-level. Whereas an exact `PTR` record match might suggest that the IPv4 and IPv6 address correspond to a common host, a load-balanced or distributed resolver might assign different hostnames to different machines, albeit with the same second-level domain. Supporting this conjecture is that while only 1.5% of the non *1-1* eq. class pairs have exactly matching `PTR` records, 47.1% have a second-level `PTR` match. Whereas the difference is not as pronounced with pairs in *1-1* eq. classes, with 41.4% matching exactly and 61.1% matchng at the second-level. While DNS `PTR` records are frequently incorrect or misleading, these findings increase our confidence that our passive technique is discovering meaningful structure.

Lastly, we examine the version of software reported by the DNS resolvers discovered in the eq. classes of the 6-month dataset. To ascertain the software version, we perform a DNS Chaosnet class (`CHAOS`) `TXT` type query for the object `version.bind`. 35,921 of the addresses respond. An example response is:
"`9.7.0-P2-RedHat-9.7.0-10.P2.el5_8.1`"

For a baseline of what is the chance that two unrelated addresses have the same `version.bind`, we repeatedly randomly pick a IPv4 and a IPv6 address from this set (ignoring whether or not they are observed as a pair), which yields a 3% match rate.

For 8,475 of the address pairs, both addresses return a value for `version.bind`. As shown in Table 3.4, the version matches for 82% of these pairs, a value significantly higher than the 3% random baseline. Moreover, 99.3% of the *1-1* eq. classes have resolvers that report exactly the same version.

Together, these `PTR` and `version.bind` results help support the validity of our associations and equivalence classes.

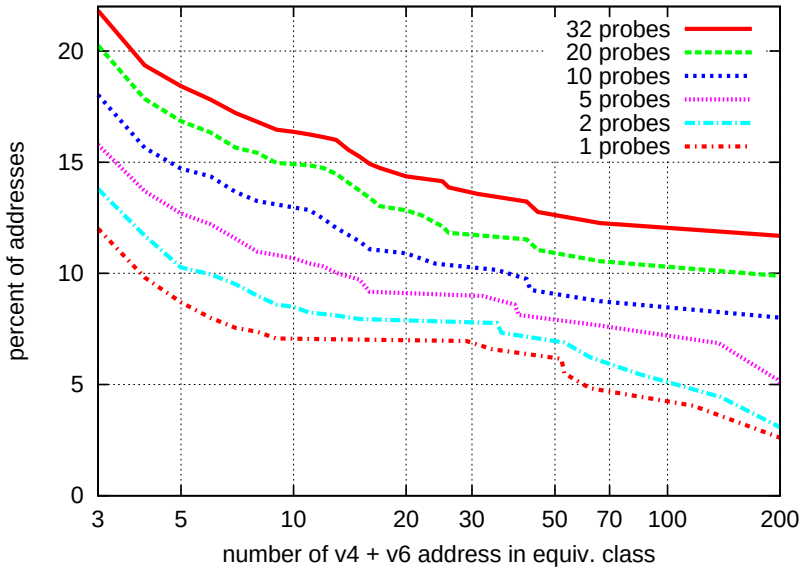


Figure 3.4: Percent of v4 + v6 address in equivalence classes of at least a given size, for open resolvers probed 32 times on September 14, 2012

3.3 Active Discovery of DNS Siblings

As the passive-DNS data set was collected over a six-month period, and contains large equivalence classes, discussed in Section 4.2, 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 v6/v4/v6/v4 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 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 3.4 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 3.4 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 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.

3.4 Active and Passive Validation

We examine the overlap between the nameserver pairs found by the active and the passive techniques. Even though the active technique probed a subset of the resolvers in the six-month collection from the passive technique, an address pair from the 4-tuples discovered by the active technique need not be present in the six-month dataset - one or both of the addresses may be absent; or if both are present, they need not be observed as a pair. We find that 73% of the actively discovered address pairs are also present as a pair in the six-month passive collection. As a somewhat looser criterion, for an additional 10% of the actively discovered pairs, although not present as a pair in the passive collection, both addresses are present and are in the same equivalence class in the passive collection. As for the remaining actively discovered pairs, for 11%, one or both of addresses are not present in the passive set. This is not that surprising as we consistently find that further probing yields new addresses. Lastly, in only 6%, are both addresses present in the passive collection and in different equivalence classes. 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 73% in perspective, we perform a 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 4-tuples into one of five bins. The first bin contains the first five 4-tuples from each resolver, the second bin contains the next five 4-tuples from each resolver, and so on. Thus, each bin has results from each of these resolvers. In each bin, we note the set of address pairs derived from in the 4-tuples.

We compare the overlap of the address pairs in the various bins to determine the overall temporal consistency of sibling sets obtain via repeated probing. Hypothetically, if an open resolver returns the same 4-tuple for each of the 25 probes, then all five bins would contain the same 4-tuples, and thus pairs, from that resolver. And if all of the resolvers returned consistent 4-tuples (though the 4-tuple may differ between the resolvers), then each of the five bins would contain the same set of address pairs, and the overlap between any two bins would be 100%. For each of the possible pairs of bins, we compute the percent of address pairs that are in common. We find an inter-bin consistency ranging from 76% to 79% with a median value of 78%. 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 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.

Table 3.5: 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%)

3.5 Web Server Machine Siblings

We evaluate our TCP-based clock skew machine-sibling detection technique on the Alexa [6] 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[4]. 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 3.5 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 [37], 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.

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 3.5 presents one example in our dataset to highlight the advantage of using the clock skew fingerprinting method. We probe `www.socialsecurity.gov` and receive

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

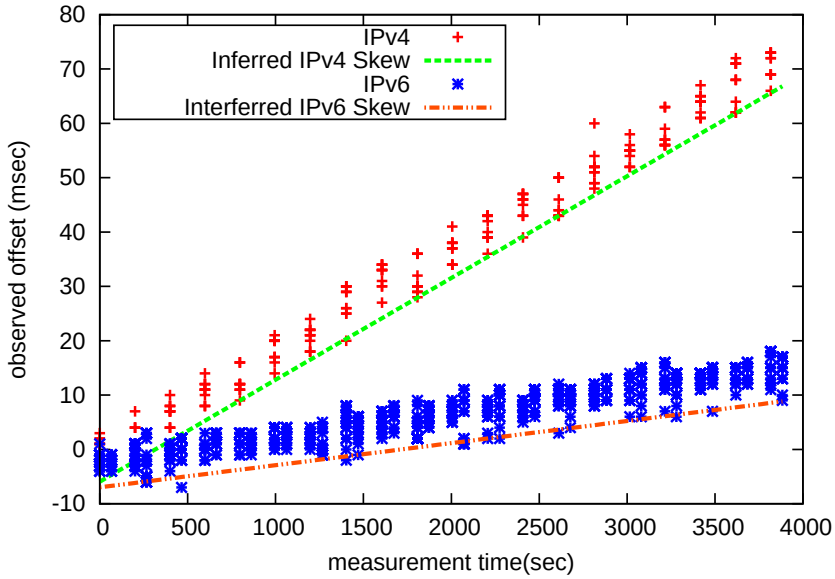


Figure 3.5: Non-Siblings: Inferred clock skew to `www.socialsecurity.gov` via IPv4 and IPv6

identical HTTP headers via either IPv4 or IPv6 in response. However, Figure 3.5 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 [28] 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 [9] to better understand these instances of divergent ASes between IPv4 and IPv6.

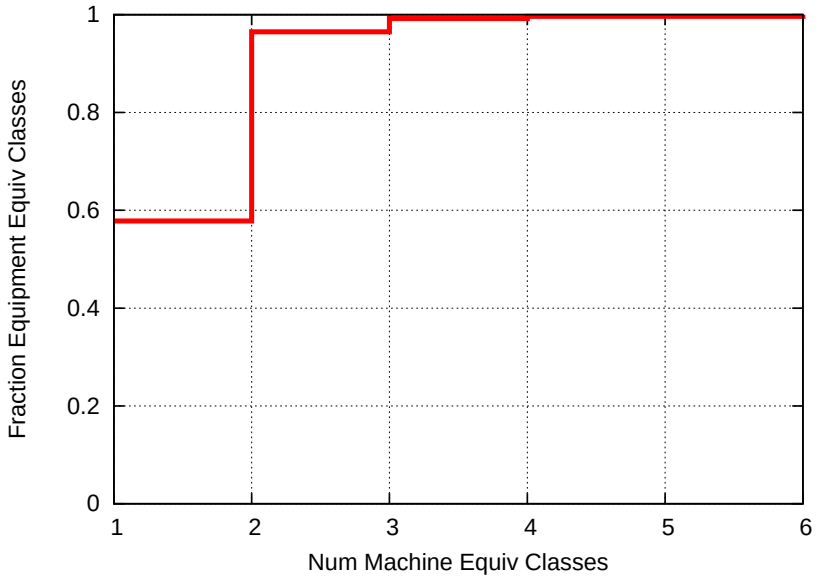


Figure 3.6: Relationship between equipment siblings and machine siblings, as inferred by applying TCP timestamp method on active DNS equivalence classes.

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

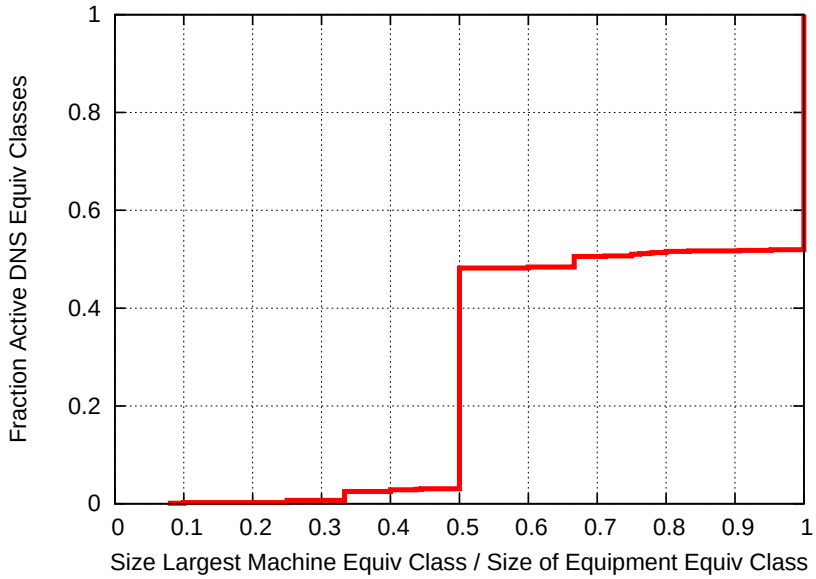


Figure 3.7: Relationship between equipment siblings and size of the largest inferred machine sibling group within the equipment equivalence classes.

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 3.6 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 3.7 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.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
Discussion

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

4.1 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 (§3.1), 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 [17], 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.

4.2 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.
2. Either IPv4 or IPv6 is behind a Network Address Translation (NAT) device [39], in which case we observe multiple addresses of the non-translated protocol, and one or two from the other.
3. Carrier grade NAT [22].
4. Middleboxes and load balancing, e.g., [3].
5. Auto-tunneling such as 6to4 and Teredo [10].
6. Hosts and devices that vary the lower order bits of the IPv6 address [32].
7. Public DNS services, e.g., [15].
8. Open, recursive nameservers [7].
9. Address changes over time [42].
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.

4.3 Relationship of Equivalence Classes to DNS Infrastructure

An interest in this paper is the discovery of IPv4, IPv6 address of resolvers that are in some way related to each other, such as being on the same server. With this aim, we first wish to discover the complexity that exists in the Internet of IPv4, IPv6 resolver associations, so as to avoid the false conclusion of a simple association when reality is more complicated. We then organize the set of discovered address pairs into equivalence classes, Section 2.1. In this Section we examine how to relate these eq. classes to the reality of the DNS infrastructure.

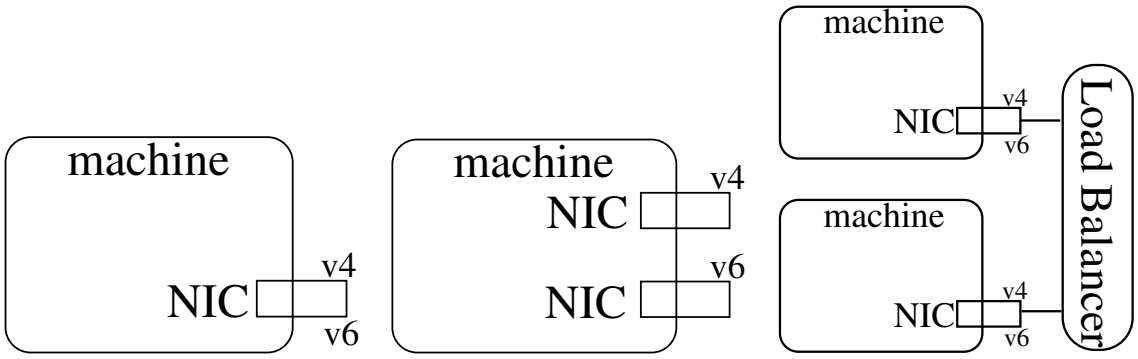
We note that the associations we observe are generated by different infrastructure configurations. To make better sense of siblings, we more carefully define three types:

1. Interface siblings
2. Machine siblings
3. Equipment siblings

As shown in Figure 4.1(a), interface siblings represent the canonical case of a dual-staked host with a single Network Interface Card (NIC). We term an IPv4 and an IPv6 address that are assigned to the same physical NIC "interface siblings." When not part of a more complicated architecture, interface siblings present as 1-1 eq. classes. However, because of the way in which DNS requests are commonly load-balanced, a request may induce responses from addresses of different interface siblings. Therefore, in this work, we explore methods to distinguish passively-gathered interface siblings from other types of siblings.

Similarly, a host may have dedicated NICs for each protocol as shown in Figure 4.1(b), a scenario that we expect to be more prevalent with network infrastructure such as servers and routers. We term the IPv4 address assigned to one NIC and the IPv6 address assigned to a different NIC of the same machine, "machine siblings." At this point in our research, we cannot distinguish between interface and machine siblings.

Lastly, as we discuss, the infrastructure generating the associations we observe can be complicated, including large clusters of functionality. For example, DNS cluster resolvers with load balancing and distributed caches, including pre-fetching – all designed to improve the performance of the time-critical name resolution functionality that is fundamental to the Internet.



(a) Interface Siblings (b) Machine Siblings (c) Equipment Siblings

Figure 4.1: Different types of sibling relationships.

Address that are associated from such distributed infrastructure we term "equipment siblings," as shown in Figure 4.1(c), where we restrict the term to cases where all of the equipment is operated by a single organization.

Natural candidates for interface or machine siblings are address pairs that are in the *1-1* eq. class and have remained so over a period of time, such as six-months, or remained so after repeated, active probing. Section 3.2 presents the encouraging result that of the pairs that are an *1-1* eq. class and where both addresses report a `version.bind`, for 99.3% of those pairs the two `version.bind`s are identical. Looking at all of the *1-1* eq. classes in the 6-month data set, a natural additional condition is whether the two addresses are in the same AS (which is a stricter condition than belonging to the same organization). If we exclude address pairs where the IPv6 is either 6to4 or belongs to the tunnel provider Hurricane Electric, AS 6939, then in 81% of the *1-1* eq. classes the two addresses are in the same AS. So, these are at least equipment siblings. This percentage is a lower bound on the percentage where the two addresses belong to the same organization. Of the cases where the AS's are distinct, the most popular case is where both AS's belong to ATT. Comcast is also popular.

There are cases where the two addresses definitely belong to different organizations. For example, paired with ATT are e.g. Comcast, Qwest, Verizon, California Education and Research Federation Network. One way this could occur is via multi-homing at the client's resolver, or at a gateway between the client's resolver and the authoriative DNS, and when the client's resolver chooses the AAAA record to query the second-level authoritative DNS, the path over IPv6 goes via a different carrier. [1]

[1] As an aside, there are also unusual pairings. in over 500 *1-1* eq. classes, the IPv6 address belongs to Eweka Internet Services, Netherlands, and the majority of the addresses are in a given /64, 2001:5c0:1400:a:8000/64, and where the paired IPv4's are in ISP's from all over, to mention a few: Telecom Italia, Deutsche Telekom, France Telecom, ASTRAL UPC Romania, Comcast, ATT, Qwest, Verizon, Vodaphone, Tunisia BackBone AS, Telefonica de Argentina, CANTV Servicios Venezuela, China Backbone CNC Group. We haven't thought of a standard architecture that would yield this. One possiblity is a probing experiment where the response from the first-level authoritative DNS to the diverse set of IPv4's is then forwarded to resolvers in Eweka Internet Services.

Also, it should be noted that some of the pairs where the IPv6 is 6to4 are also likely candidates for interface or machine siblings. Of the pairs where the IPv6 address is 6to4, 19% conform to the simple, canonical case where the embedded IPv4 address equals the paired IPv4 address, and the address pair is a *1-1* eq. class, Section 3.1. These are consistent with a single server that has obtained IPv6 connectivity via 6to4.

Considering eq. classes larger than *1-1*, it is still possible that in some cases the addresses are on the same machine. A particular case is where a single IPv4 is paired with multiple IPv6's all in the same /64 and where the different IPv6 addresses are observed in disjoint time intervals, which suggests that the IPv6 is being varied for anonymity.

Again excluding address pairs where the IPv6 is either 6to4 or belongs to Hurricane Electric, then of the eq. classes larger than *1-1*, 52% have all addresses in one AS, and can be inferred to be equipment siblings. Although smaller than the 81% for the *1-1* eq. classes, it is still a substantial portion. And additional equipment siblings can be identified from eq. classes where addresses in multiple AS's are actually allocated to the same organization.

Moreover, it is reasonable to infer equipment siblings that are a *subset* of an eq. class. For example, there is an eq. class *7-1151* that has all addresses in one AS, except one of the IPv4's is in an AS's of an unrelated organization. It is reasonable to infer an equipment sibling where the address pair containing this IPv4 is omitted. A more dramatic example is the Mammoth eq. class where Google's AS 15169 plays a major role, Section 3.1. In this eq. class, the addresses in 15169 are almost exclusively associated with just each another - only 2% of the IPv4's are paired with an IPv6 outside 15169 (and these non-15169's addresses are scatter across a hundred different AS's). The subset of the Mammoth eq. class consisting of pairs where both addresses are in AS 15169 would be an equipment sibling that spans continents. Also, it is possible that additional addresses in the Mammoth eq. class are part of this Google equipment sibling if Google has servers in ISPs' whose addresses are from the ISP.

In terms of the abstraction of the bipartite graph, Section 2.1, one could consider an enhancement where each edge is labelled by the two AS's, or better yet the organizations, of the vertices (addresses) of that edge. For many labels, the two AS's would be the same, but not always. Some eq. classes will have edges with different labels, and one could examine the sub-partitions consisting of edges with a common label.

Having discovered address pairs that we think are likely to be on the same server (based on being in a *1-1* eq. class and in the same AS) we can confirm this inference by using the targeted fingerprinting technique, Section 2.2, assuming we have a vantage point from which the server will respond. Likewise, we can expect that some of the address pairs within an equipment sibling are likely machine siblings, and we could use the targeted fingerprinting technique to search for them, again assuming suitable vantage points.

CHAPTER 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 [1,12]. 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.

THIS PAGE INTENTIONALLY LEFT BLANK

REFERENCES

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

[2] Public DNS: Performance Benefits, 2013. https://developers.google.com/speed/public-dns/docs/performance.

[3] Joe Abley. A software approach to distributing requests for DNS service, 2004. http://ftp.isc.org/isc/pubs/tn/isc-tn-2004-1.html.

[4] Bernhard Ager, Wolfgang Mühlbauer, Georgios Smaragdakis, and Steve Uhlig. Comparing dns resolvers in the wild. In *Proceedings of the 10th ACM IMC*, pages 15–21, 2010.

[5] Akamai. Edgscape geolocation, 2012. http://www.akamai.com/html/technology/products/edgescape.html.

[6] Alexa. Top 1,000,000 sites, 2012. http://www.alexa.com/topsites.

[7] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. RFC 4033 (Proposed Standard), March 2005. Updated by RFC 6014.

[8] R. Bellis. DNS Transport over TCP - Implementation Requirements. RFC 5966 (Proposed Standard), August 2010.

[9] Xue Cai, John Heidemann, Balachander Krishnamurthy, and Walter Willinger. Towards an AS-to-organization map. In *Proceedings of the 10th annual conference on Internet measurement*, pages 199–205, 2010.

[10] B. Carpenter and K. Moore. Connection of IPv6 Domains via IPv4 Clouds. RFC 3056 (Proposed Standard), February 2001.

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

[12] Comcast. IPv6 Information Center, 2012. http://www.comcast6.net/.

[13] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), December 1998.

[14] R. Elz and R. Bush. Clarifications to the DNS Specification. RFC 2181 (Proposed Standard), July 1997. Updated by RFCs 4035, 2535, 4343, 4033, 4034, 5452.

[15] Google. Public DNS, 2012. https://developers.google.com/speed/public-dns/.

[16] Marc Heuse. Recent advances in IPv6 insecurities. In *Chaos Communications Congress*, 2010.

[17] Hurricane Electric. IPv6 tunnel broker service, 2012. http://tunnelbroker.net/.

[18] Geoff Huston. IPv6 BGP Statistics, 2012. http://bgp.potaroo.net/v6/as2.0/.

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

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

[21] V. Jacobson, R. Braden, and D. Borman. TCP Extensions for High Performance. RFC 1323, May 1992.

[22] S. Jiang, D. Guo, and B. Carpenter. An Incremental Carrier-Grade NAT (CGN) for IPv6 Transition. RFC 6264 (Informational), June 2011.

[23] Tadayoshi Kohno, Andre Broido, and K. C. Claffy. Remote physical device fingerprinting. In *Proceedings of IEEE Security and Privacy*, pages 211–225, 2005.

[24] Christian Kreibich, Nicholas Weaver, Boris Nechaev, and Vern Paxson. Netalyzr: illuminating the edge network. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 246–259, 2010.

[25] Matthew Luckie and Bradley Huffaker. IPv6 deployment: trends and tidbits of 4,800 dual-stack ases. In *CAIDA AIMS Workshop*, 2012.

[26] Gordon Fyodor Lyon. *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. 2009.

[27] MaxMind. minfraud service, 2013. http://www.maxmind.com/en/ccv_overview.

[28] David Meyer. University of Oregon RouteViews, 2012. http://www.routeviews.org.

[29] P.V. Mockapetris. Domain names - implementation and specification. RFC 1035 (Standard), November 1987.

[30] Ram Mohan. Will U.S. Government Directives Spur IPv6 Adoption?, September 2010. http://www.circleid.com/.

[31] S.B. Moon, P. Skelly, and D. Towsley. Estimation and removal of clock skew from network delay measurements. In *Proceedings of INFOCOM*, volume 1, pages 227 –234, mar 1999.

[32] T. Narten, R. Draves, and S. Krishnan. Privacy Extensions for Stateless Address Autoconfiguration in IPv6. RFC 4941 (Draft Standard), September 2007.

[33] Nominum. Intelligent DNS, 2012. `http://www.nominum.com/`.

[34] Erik Nygren, Ramesh K. Sitaraman, and Jennifer Sun. The akamai network: a platform for high-performance internet applications. *SIGOPS Oper. Syst. Rev.*, 44(3):2–19, August 2010.

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

[36] Nadi Sarrar, Gregor Maier, Bernhard Ager, Robin Sommer, and Steve Uhlig. Investigating IPv6 traffic: what happened at the world IPv6 day? In *Proceedings of PAM*, 2012.

[37] Michael James Silbersack. Improving TCP/IP security through randomization without sacrificing interoperability. In *Proceedings of BSDCan*, 2006.

[38] Spamhaus, 2013. `http://www.spamhaus.org/`.

[39] P. Srisuresh and K. Egevang. Traditional IP Network Address Translator (Traditional NAT). RFC 3022 (Informational), January 2001.

[40] P. Vixie. Extension Mechanisms for DNS (EDNS0). RFC 2671 (Proposed Standard), August 1999.

[41] Yong Wang, Daniel Burgener, Marcel Flores, Aleksandar Kuzmanovic, and Cheng Huang. Towards street-level client-independent ip geolocation. In *NSDI'11. Proceedings of the 8thUSENIX conference on networked systems design and implementation*, pages 27–36, 2011.

[42] Yinglian Xie, Fang Yu, Kannan Achan, Eliot Gillum, Moises Goldszmidt, and Ted Wobber. How dynamic are ip addresses? In *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 301–312, 2007.

[43] Sebastian Zander, Lachlan L. H. Andrew, Grenville Armitage, Geoff Huston, and George Michaelson. Mitigating sampling error with measuring internet client ipv6 capabilities. In *Proceedings of the 12th ACM SIGCOMM conference on Internet measurement*, 2012.

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. National Science Foundation
   Arlington, Virginia
2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California