Server Siblings: Identifying Shared IPv4/IPv6 Infrastructure via Active Fingerprinting

Robert Beverly[1], Arthur Berger[2]

[1] Naval Postgraduate School, Monterey, CA
[2] MIT CSAIL / Akamai, Cambridge, MA
rbeverly@nps.edu, awberger@csail.mit.edu

Abstract. We present, validate, and apply an active measurement technique that ascertains whether candidate IPv4 and IPv6 server addresses are "siblings," i.e., assigned to the same physical machine. In contrast to prior efforts limited to passive monitoring, opportunistic measurements, or end-client populations, we propose an *active* methodology that generalizes to all TCP-reachable devices, including servers. Our method extends prior device fingerprinting techniques to improve their feasibility in modern environments, and uses them to support measurement-based detection of sibling interfaces. We validate our technique against a diverse set of 61 web servers with known sibling addresses and find it to be over 97% accurate with 99% precision. Finally, we apply the technique to characterize the top ∼6,400 Alexa IPv6-capable web domains, and discover that a DNS name in common does not imply that the corresponding IPv4 and IPv6 addresses are on the same machine, network, or even autonomous system. Understanding sibling and non-sibling relationships gives insight not only into IPv6 deployment and evolution, but also helps characterize the potential for correlated failures and susceptibility to certain attacks.

1 Introduction

While significant prior research has characterized the evolution, routing, and performance of IPv6 [6, 15, 5], less attention has been given to understanding whether IPv6 infrastructure is being deployed using separate hardware or by adding IPv6 to existing machines. I.e., are providers using separate IPv4 and IPv6 servers to host the same web content, or using single "dual-stacked" servers?

Given an IPv4 and IPv6 address, we seek to infer whether they belong to interfaces on the same physical machine. We term such cross-protocol associated addresses server "siblings." To accurately determine sibling and non-sibling relationships, we leverage prior work on device fingerprinting to perform active measurements of TCP option signatures (coarse-grained) [10] and TCP timestamp clock skew (fine-grained) [9].

The prevalence of shared IPv6 infrastructure has important policy and Internet evolution implications [3]. Moreover, for network operators and researchers, the way in which IPv6 is deployed has particular impact on measurement and security. A potential, future application for the methods herein is for IPv6 geolocation, where prior knowledge of the corresponding IPv4 sibling can be leveraged.

Note that making such inferences based on a common Domain Name System (DNS) name can be dubious. As shown in Section 4, a DNS name in common does not imply that the IPv4 and IPv6 addresses are on the same interface, machine, or even autonomous system (AS).

A second area of interest is IPv6 security, as the deployment and maintenance of firewalls, filtering, and intrusion detection systems on IPv6 lags, while tunnels and transition mechanisms facilitate alternate data paths for application-layer attacks. Furthermore, not only are many old IPv4 network-layer attacks feasible in IPv6, IPv6 introduces new attack vectors [7]. The extent to which IPv4 infrastructure depends on IPv6, and vice-versa, therefore has unknown security implications. Whether an attack against the IPv6 address of an Internet web or DNS server impacts an organization's corresponding service for IPv4 depends on whether it is dual-stacked. Further, dual-stacked servers imply the potential for *correlated failures* that impact survivability.

Toward identifying shared IPv4/IPv6 infrastructure, our contributions are:

1. A reappraisal of the current feasibility of Kohno's 2005 physical device fingerprinting [9] method using TCP clock skew.
2. Integration to, and enhancement of, various fingerprinting methods to actively, rather than passively, associate IPv4 and IPv6 server addresses.
3. Evaluation on ground-truth data, with $> 97\%$ accuracy and 99% precision.
4. Real-world measurements of siblings and non-siblings among the Alexa top websites, characterizing a portion of Internet IPv6 infrastructure.

2 Background

Inferring IPv4 and IPv6 host associations has largely been confined to client populations using passive, opportunistic measurements. For instance prior projects have used web-bugs, javascript, or flash object to determine the prevalence of IPv6 connectivity and associate IPv4 and IPv6 addresses of connecting clients [17, 14]. In contrast our technique is active and we study servers.

Our prior work also examines IPv4/IPv6 associations, but is limited to DNS resolvers [2]; the techniques herein are more general and can be performed actively, on-demand. By operating at the transport layer we can actively probe any listening TCP service to test whether a candidate IPv4 and IPv6 address belong to the same device.

At its heart, our work relies on the rich history of prior research in network fingerprinting. Network fingerprinting is a common technique that relies on implementation and configuration-specific characteristics to uniquely identify devices. We leverage the fact that any application or transport-layer fingerprint will be common to the lower level network protocol, whether IPv4 or IPv6. We use coarse-grained active operating system (OS) fingerprinting, e.g. [10], to eliminate clearly unrelated IPv4 and IPv6 addresses. However, OS fingerprinting alone does not provide sufficient granularity to accurately classify true siblings as the set of possible OSes is small relative to the set of possible addresses.

We therefore leverage previous work on physical device fingerprinting [9]. Kohno's technique measures a machine's clock drift by obtaining TCP-layer

Table 1. Properties of the four datasets probed

	Dataset	Hosts	# v4 AS	# v6 AS	Countries	# Option Signatures
1)	Ground Truth	61	34	34	19	13
2)	Alexa embedded	1050	85	80	31	30
3)	Alexa non-CDN	1533	629	575	69	73
4)	Alexa CDN	230	59	55	18	29

timestamps from the remote machine. While this technique has been used in the past, we apply it in a new context and reappraise its feasibility 10 years later. More importantly, skew-based fingerprinting has been primarily used on network clients, rather than servers. We find several interesting server-specific behaviors, e.g. load-balancing, that we take into account. Second, we enhance and combine the technique with other fingerprinting methods. We then evaluate the accuracy of our technique on a distributed set of ground-truth web servers. Last, we apply the method to the new problem of actively interrogating remote IPv4 and IPv6 endpoints over TCP to determine if they are server siblings.

3 Methodology

Our methodology uses active fingerprinting at the TCP layer, as a host's TCP stack is common to both the underlying IPv4 and IPv6 stack. We combine several of such fingerprinting techniques to achieve the best accuracy. Our resulting active method can be run on-demand to provide a server sibling test.

A networked server may have one or more interfaces, each with one or multiple addresses. An interface's addresses can be IPv4, IPv6, or a combination. Our TCP fingerprinting techniques attempt to determine whether a given IPv4 and IPv6 address share a common TCP stack. If the determination is "yes," then we are confident (see §4 on ground truth) that the two address are on the same server (and in practice likely the same interface), and we classify the address pair as siblings. If the determination is "no," then we are confident that the addresses are on separate interfaces, and most likely separate machines, and we classify the address pair as non-siblings.

3.1 Datasets

This work considers four datasets shown in Table 1. First, a ground-truth dataset (1) where the IPv4 and IPv6 addresses are known to be co-located on the same dual-stacked host. Then, for the subset of the Alexa [1] top 100,000 sites with both `A` and `AAAA` records in the DNS, we partition into set (2) sites where the IPv4 address is embedded in the corresponding IPv6 address. And for sites not in (2), partition into datasets: (3) those not part of a Content Distribution Network (CDN), and (4) those part of a CDN.

To develop and refine our association inference algorithm, we utilize ground-truth data consisting of 61 hosts with known IPv4/IPv6 association. While this set is relatively small, it spans 34 ASes and 19 countries. Importantly, it allows us to test not only our algorithm's recall (ability to identify true siblings), but also its precision (ability to identify ∼1,800 possible combinations of non-siblings).

We query the DNS for the `A` and `AAAA` records of the Alexa hosts as retrieved in April, 2014. If the query returns multiple DNS records, we retain only the first. We perform the DNS resolution only once in order to obtain the IPv4 and IPv6 addresses. The remainder of our experimentation involves directly probing IPv4 and IPv6 addresses; the DNS is not subsequently consulted as to avoid dynamics due to DNS or DNS load-balancing.

A total of 6,387 sites in the Alexa top 100,000 have both IPv4 and IPv6 addresses. We remove 22 sites that return non-global IPv6 addresses, e.g. "`::`." Because multiple sites can be hosted on one server, we reduce this set to 3,986 unique IPv4/IPv6 address pairs. Further, since the Alexa list is comprised of popular web sites, these sites are frequently part of a CDN. We observe that many sites use anycast, as inferred by collecting RTTs from geographically dispersed vantage points and finding those sites with RTTs that are not physically possible without anycast. We remove these sites from our analysis as to not conflate the effects of anycast with our inferences, leaving 2,813 unique address pairs.

When part of a CDN, the same website is often hosted on multiple machines distributed across sites or geographic regions. We therefore separate the Alexa hosts into those that are part of a CDN versus those that are not. To distinguish CDN site, we query the DNS for the site from five geographically dispersed vantage points. If we obtain different `A` or `AAAA` records from multiple vantage points, we label the site as belonging to a CDN. In addition, if the site's DNS CNAME corresponds to a well-known CDN, we place it in the CDN dataset.

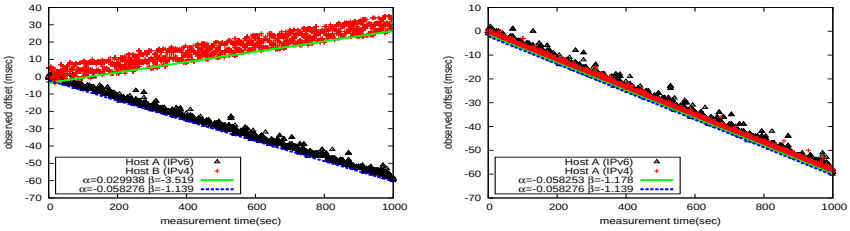
Last, we create the "embedded" dataset. In practice, IPv4 addresses are frequently embedded in IPv6 addresses in different ways. We include instances where the IPv4 address is embedded as four bytes, e.g. `162.159.243.37` and `2400:cb00:2048:1::a29f:f325`, or where the IPv4 base-10 representation is used as a base-16 sequence, e.g. `142.8.72.175` and `2a01:f1:d0:dc0:142:8:72:175`.

Table 1 characterizes the distribution of hosts in each dataset, including the number of IPv4 and IPv6 ASes they represent as inferred from the routeviews global BGP table [12] from the same day as our Alexa site list (April 14, 2014), as well as the geographic distribution as determined by maxmind [11].

3.2 TCP Option Signature

Modern TCP stacks make common use of TCP options, especially options in [8]. While options are standardized, the order and packing of those options is implementation dependent, thereby providing a well-known operating system-granularity fingerprint [10]. For example, FreeBSD in our dataset returns: `<mss 1460, nop, wscale 3, sackOK, TS>` whereas a Linux machine returns: `<mss 1460, sackOK, TS, nop, wscale 4>`.

To form the signature, we preserve the option order, and strip the integer value of the MSS and timestamp options. While the IPv6 MSS is frequently 20 bytes less than the IPv4 MSS (to accommodate the extra 20 bytes of IPv6 header), this is a loose rule in our ground-truth. Some hosts connect via tunnels, with a lower IPv6 MSS, while some hosts support jumbo-grams only for IPv4.



(a) Non-Siblings (Different Hosts) (b) Siblings (Same Host)

Fig. 1. Timestamp drift of candidate siblings.

While coarse-grained, the variability of the TCP options signature provides a good first-order filter. Table 1 reports the number of unique TCP option signatures observed for each of the datasets.

3.3 TCP Timestamp Skew

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 [8]. 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 [9], for each IPv4 packet we compute the *observed offset* or *skew*: $s_i^4 \equiv (T_i^4 - T_0^4) - (t_i^4 - t_0^4)$ and likewise for each IPv6 packet, $s_i^6 \equiv (T_i^6 - T_0^6) - (t_i^6 - t_0^6)$.

Given a sequence of skews, we compute drift via the linear programming solution in [13] to determine a line that is constrained to be under the data points, but minimizes the distance to the data points. We obtain:

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

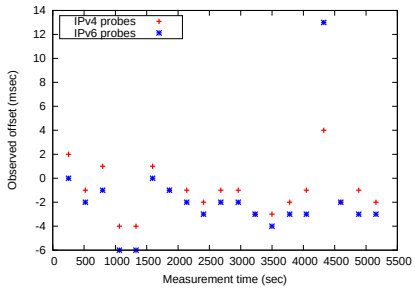
I.e., two lines, one corresponding to the interrogation of I_4 and one to I_6 that lower-bounds the set of offset points observed. The angle θ between them is:

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

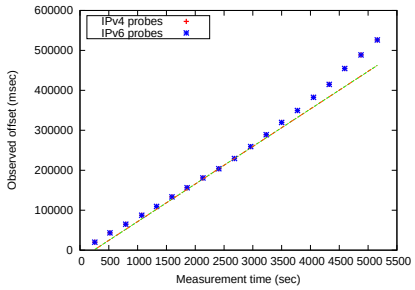
If $\theta < \tau$, then I_4 and I_6 are inferred to be siblings, where τ is a threshold. Empirically, we find that $\tau = 1.0$ degree is sufficiently discriminating.

Figures 1(a) and 1(b) illustrate the approach using two hosts for which we know their ground-truth interface addresses. Figure 1(a) displays the observed drift from interrogating Host A's IPv6 interface as compared to Host B's IPv4 interface. We observe not only different drift, 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 are *non-siblings* ($\theta \geq \tau$).

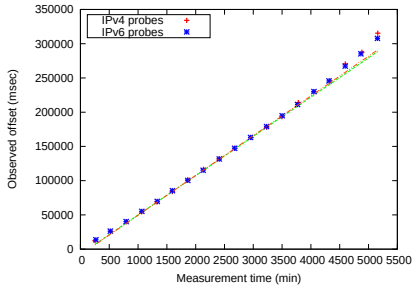
In contrast, Figure 1(b) displays a *sibling* relationship. In this experiment, we probe the same host (A) via its IPv4 and IPv6 interfaces. We observe nearly identical inferred skew (the linear programming solution determined as $\alpha_4 = -0.058253, \beta_4 = -1.178$ and $\alpha_6 = -0.058276, \beta_6 = -1.139$; $\theta = 1.3 \times 10^{-3}$).



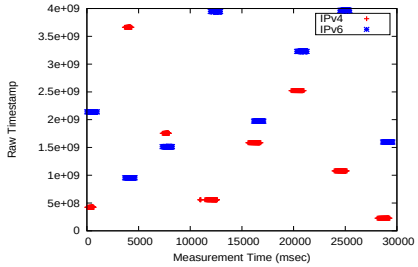
(a) Negligible Drift



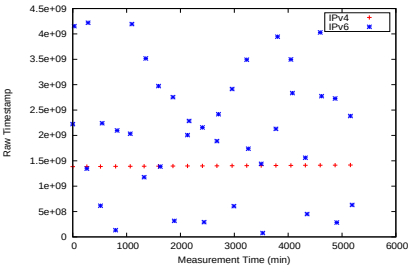
(b) Siblings



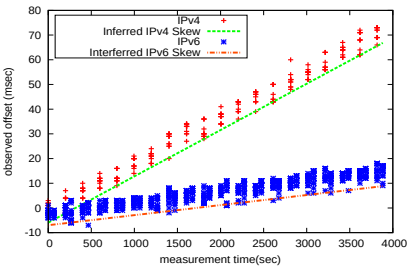
(c) Non-Siblings



(d) FreeBSD random offset makes timestamps non-monotonic across flows



(e) `www.caida.org` timestamps. IPv4 timestamps are monotonic, but random for IPv6 due to a proxy.



(f) Non-siblings: Inferred clock drift to `www.socialsecurity.gov` via IPv4 and IPv6

Fig. 2. Examples of insufficient drift angle, necessitating point distance (§3.4) (a-c) and complicated association inferences (d-f).

3.4 TCP Timestamp Point Distance

In our ground-truth testing of the TCP timestamp skew, we make three general observations: i) some machines now have clocks with negligible drift (e.g. Figure 2(a)); ii) some clocks we observe exhibit non-linearity in their skews (e.g. Figure 2(b)); and iii) the observed skew of two distinct machines, but with the same OS and located in the same rack, can be very similar (e.g. Figure 2(c)).

These complicating factors, which Kohno did not observe in 2005, motivate a second test on the TCP timestamps: pair-wise point distance. For each IPv4 packet, with arrival time t_i^4, we find the IPv6 packet whose arrival time is closest

(either before or after), say it is packet j, with arrival time t_j^6. We define the absolute value of difference in skews of these two packets to be the pair-wise point difference for IPv4 packet i: $diff(i) = |s_i^4 - s_j^6|$.

After some experimentation, we find that the median of the $diff(i)$'s to be most useful. Figure 2(c) illustrates the merit of the point distance method. The plotted IPv4 and IPv6 skews are from two different, but identical, physical machines in the same data center. The timestamp drifts appear very similar and yield $\theta = 0.358$ degrees, which is less than the τ threshold. Thus, with the skew inference alone, these two addresses would erroneously be inferred to be siblings. However, the point distance correctly rejects them: the median difference is above a chosen threshold of 100 msec.

3.5 Full Algorithm

Algorithm 1 presents pseudocode for the logic to infer whether I_4 and I_6 are siblings. First, we probe I_4 and I_6 over time to obtain the TCP option signatures $Signature^{4,6}$ and vectors of skew measurements $\boldsymbol{s^{4,6}}$. The first condition (line 3) is to test whether the option signatures differ (§3.2), in which case we infer that the addresses are non-siblings and terminate.

We observe that the options returned by various TCP stacks can be divided into five cases: 1) no options returned; 2) timestamp not present in options; 3) timestamps non-monotonic between connections; 4) timestamps are random; 5) timestamps are monotonic. Lines (4-8) tests for these cases. Non-monotonic timestamps can occur when I_4 or I_6 are addresses of a front-end load balancer and the clocks of the machines behind the load balancing are not precisely synchronized. In this case, the timestamps of a single flow are monotonic, but can be non-monotonic across connections. In addition, we also observe TCP stacks where the timestamp always starts at 1 for each connection.

Next are random timestamps. Some TCPs, notably BSD-based [16], randomize the initial TCP timestamp values on a per-flow basis. As shown in Figure 2(d), the timestamps increase linearly from some random offset for each connection and are not monotonic across flows. When one addresses presents randomized values and the other does not, we infer a non-sibling relationship.

Algorithm 1 Siblings(I^4, I^6)

```
1: (s^4, Signature^4) ⇐ probe(I^4)
2: (s^6, Signature^6) ⇐ probe(I^6)
3: if Signature^4 ≠ Signature^6 then return false
4: for case in 'missing', 'non-mono', 'rand' do
5:     if case(s^4) = True and case(s^6) = True then
6:         return unknown
7:     if case(s^4) = True or case(s^6) = True then
8:         return false
9: (α_4, α_6) = slopes(s^4, s^6)
10: θ = angle(α_4, α_6)
11: if θ < τ then return true
12: else return false
```

These cases present both a difficulty and an opportunity. When the timestamps from both the IPv4 and IPv6 address are non-monotonic, missing, or random, we cannot infer a definitive relation and classify their association as "unknown." However, if one protocol matches one of the cases and the other protocol does not, we conclude that the addresses are not related.

As a real-world example, consider the raw timestamps from the IPv4 and IPv6 addresses of `www.caida.org` in Figure 2(e). While the IPv4 timestamps increase monotonically with a constant skew, the IPv6 timestamps are random. In addition, the TCP option signatures were different. Correspondence with the system administrators revealed that the IPv6 address was a separate machine that acted as a proxy for the IPv4 web server.

Note that application layer fingerprints, for instance the HTTP headers in our experiment, are not a reliable sibling detection mechanism. Figure 2(f) presents one example in our dataset to highlight our use of granular fingerprints. We probe the site and receive identical HTTP headers via either IPv4 or IPv6 in response. However, the drift-based inference clearly shows these as non-siblings.

Lastly, we perform the skew-based inference (§3.3), lines 9 - 12. When testing Alg. 1 against ground truth, we find corner cases where either the algorithm could not make a determination, or was incorrect. With some experimentation, we determine some additional, simple logic that improved the results. However, this logic relies on some rather arbitrary parameter values. We include it here as an optional enhancement, Alg. 2. We believe that further refinement is possible.

Algorithm 2 Optional, enhancement to Algorithm 1

10: $median \Leftarrow$ point distances $diff(i)$'s
11: $range \Leftarrow$ max minus min of observed skews
12: **if** $range < 100$ **then return** unknown
13: **if** $(|\alpha_4| \leq 0.0001)$ or $(|\alpha_6| \leq 0.0001)$ **then**
14: **if** $median \leq range/10$ **then return** true
15: **else return** false
16: **if** $median > 100$ **then return** false
17: return to Algorithm 1 at step 10.

The core reason the drift inference works is that, in the common-case, the remote server's TCP timestamp clock is less accurate than the prober host's packet capture clock. However, in contrast to prior work, we find that for a subset of the machines we probe, the TCP timestamps are set by a clock that is as stable as that of the probing machine, such that the only source of skew comes from probing latency variation (e.g. Figure 2(a)). As an alternative, we compute the median of the point distances (§3.4) in line 10 of Alg. 2, and the *dynamic range* of the skews, defined as: the largest skew observed over time, for either IPv4 or IPv6, minus the smallest skew, line 11. (In the plots of skew, the range is the largest y-coordinate minus the smallest.) If the dynamic range is below a threshold, we cannot obtain a reliable skew fingerprint, as in Figure 2(a),

and classify the relationship as unknown, line 12. Similarly, if either the IPv4 or IPv6 slope (α_4 or α_6) is below a threshold $minslope = 0.0001$, we consider the skew-based inference unreliable, line 13. In this case, if the median point distance is an order of magnitude less than the dynamic range we associate the IPs (lines 14-15). Last, if the median point distance is >100 ms, we infer non-siblings.

A limitation of our technique is that we require the ability to negotiate a TCP connection with the remote device, i.e. the remote machine must be listening on a publicly accessible TCP port. As applied to common server infrastructure, e.g. remote web or DNS servers, this does not present a practical limitation.

4 Results

This section analyzes results from deploying the aforementioned technique on our datasets, including ground-truth and the larger IPv4 and IPv6 Internet.

4.1 Ground Truth Validation

To validate the accuracy of our technique, we examine the ground truth dataset described in §3.1. We perform multiple rounds of testing. While the data provides us with true associations, for evaluation purposes, we also test false associations in each round. These known non-siblings are formed by randomly associating a non-associated IPv6 site with each IPv4 site. In this fashion, we test both type I and type II errors.

Table 2. Relative Ground Truth Performance of Sibling Classifiers

Algorithm	**Accuracy**	**Precision**	**Recall**	**Specificity**	**Unknown**
TCP Opts	82.2%	74.1%	98.2%	66.8%	0.0%
Kohno	90.6%	82.3%	97.0%	86.4%	27.8%
Alg 1	94.2%	93.6%	91.4%	96.0%	22.4%
Alg 1&2	97.4%	99.6%	93.1%	99.8%	29.4%

We wish to understand discriminative power of both the original Kohno timestamp skew algorithm, as well as our enhancements, in distinguishing siblings from non-siblings. First, we look at using TCP options as a classifier alone. As shown in Table 2, TCP options yield an accuracy of 82.2% with 74.1% precision, 98% recall, and 67% specificity. (Where precision is the fraction of identified siblings that are truly siblings, recall is the fraction of all ground-truth siblings classified as siblings, and specificity measures the ability to identify non-siblings). Thus, while the option signature alone does not provide sufficient granularity, it eliminates non-siblings with minimal overhead (just a single TCP ACK packet from the IPv4 and IPv6 target).

We next examine Kohno's original timestamp skew algorithm alone, without consideration of TCP options. Over ten rounds, we obtain an accuracy of 90.6% with 82.3% precision, 97.0% recall and 86.4% specificity. We then examine Algorithm 1 and the combined Algorithms 1 and 2 as detailed in §3.5. We see that

Table 3. Alexa Machine-Sibling Inferences

	Dataset(Table 1)		
Inference	**non-CDN**	**CDN**	**Embedded**
Siblings			
- v4/v6 drift match	816 (53.2%)	55 (23.9%)	978 (93.1%)
Non-Siblings			
- v4 and v6 opt sig differ	229 (14.9%)	14 (6.1%)	22 (2.1%)
- v4 or v6 missing	70 (4.6%)	11 (4.8%)	7 (0.7%)
- v4 or v6 random	23 (1.5%)	13 (5.7%)	1 (0.1%)
- v4 or v6 non-monotonic	52 (3.4%)	47 (20.4%)	1 (0.1%)
- v4/v6 drift mismatch	35 (2.3%)	13 (5.7%)	0 (0.0%)
Unknown			
- v4 and v6 missing	196 (12.8%)	6 (2.6%)	26 (2.5%)
- v4 and v6 random	32 (2.1%)	25 (10.9%)	6 (0.6%)
- v4 and v6 non-monotonic	78 (5.1%)	45 (19.6%)	9 (0.9%)
- v4 or v6 unresponsive	2 (0.1%)	1 (0.4%)	0 (0.0%)
Total	1533 (100%)	230 (100%)	1050 (100%)

each provides increasingly accurate sibling classification, with the full algorithm yielding an accuracy of 97.4%, with 99.6% precision, 93.1% recall, and 99.8% specificity over the ten rounds of testing. However, some of this accuracy comes at the expense of our full algorithm labeling 29.4% of the hosts as "unknown" as it cannot make a definitive determination.

4.2 Web Server Machine Siblings

As an initial application of our sibling detection technique, we characterize sibling relationships among a subset of important Internet infrastructure, Alexa [1] top 100,000 websites as gathered, resolved, and probed in April, 2014 (details of dataset in §3.1). We perform our probing from a host with high-speed, native IPv6 connectivity. To remain inconspicuous, we probe at a low rate. We fetch the root HTML page from each site's IPv4 and IPv6 interfaces once every ∼3.5 hours over ∼17 days.

We then apply our inference Algorithm 1 and 2 to the datasets in Table 1. As described in §3, there are a variety of potential outcomes. For each of the three Alexa datasets, we divide the inferences into three major categories in Table 3: siblings, non-siblings, and unknown.

In aggregate, we find 53.2% of the IPv4/IPv6 addresses of non-CDN, 23.9% of CDN, and 93.1% of embedded are siblings via the full Algorithm 1 and 2. Fully 42.6% of the CDN, and 26.7% of the non-CDN have addresses we infer to be non-siblings. While we expect a high proportion of siblings among sites with embedded addresses, 3.0% are non-sibling–underscoring the fact that addresses alone do not imply the same machine. And we cannot definitively determine 20% of the non-CDN, 33.5% of the CDN, and 3.9% of the embedded sites.

The largest contributing subset of non-monotonic timestamps are CDN sites – as we might expect due to the various forms of load balancing inherent in CDN architectures. A non-trivial fraction of non-CDN and CDN sites have miss-

ing timestamps. We learned via personal communication with an operator that missing timestamps in one case were due to a front-end load balancing device; similar middlebox issues [4] likely cause the missing timestamps observed here.

Among the sibling and non-sibling populations, we examine the origin AS of the prefixes to which the addresses belong from the routeviews [12] 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. As shown in Table 4, 21.8% of the non-siblings in our non-CDN dataset are in different ASes, as compared to 10% of the siblings. Siblings may be in different ASes when an organization uses IPv6 tunnels or a different AS for IPv6. By contrast, 97.3% of the inferred siblings among the embedded sites are within the same AS. Only 51% of the non-siblings among the CDN sites reside within the same AS. Manual investigation of some of the siblings in different ASes reveals that the ASes belong to the same organization.

Table 4. Alexa Machine-Sibling AS Agreement

	Fraction of matching (I^4, I^6) ASNs		
Inference	**non-CDN**	**CDN**	**Embedded**
Siblings	90.0%	83.6%	97.3%
Non-Siblings	78.2%	51.0%	87.1%
Unknown	91.6%	62.3%	78.0%

5 Conclusions and Future Work

We developed, validated, and applied a method for using TCP-layer fingerprinting techniques to identify IPv4 and IPv6 addresses that belong to the same machine. By combining coarse and fine-grained TCP-layer fingerprinting, we identify server "siblings." We can imagine several other applications of sibling interface identification: predicting correlated failures or similar behaviors under attack (and whether the IPv4 and IPv6 interfaces share fate); IPv6 geolocation that leverages knowledge of the corresponding IPv4 address; and comparing IPv4 and IPv6 path performance, by providing certainty as to whether a measurement end-point is common; and more generally, understanding how IPv6 and IPv4 network infrastructures are co-evolving at a macroscopic level. Although we applied our technique to web servers, it generalizes to any device with a listening TCP service, including DNS, email, and peer-to-peer services.

Although our technique validated surprisingly well for our diverse set of ground truth, we see at least three areas for improvement. First, the optional enhancement algorithm (Alg. 2) we used to classify problematic cases contains parameters and thresholds that may overfit our data. A larger ground-truth dataset would support further refinement and higher confidence in our inferences. Second, although we detect certain instances of TCP load-balancing by observing multiple monotonic sequences with different initial offsets, it would be better to use reverse-proxy detection techniques to discern cases where a TCP-splitting proxy sits in front of the interrogated web server.

Last, our preliminary sensitivity results show that our inferences are stable even with fewer data points and over shorter time frames. Our technique can make some sibling inferences quickly, with only a few TCP observations, whereas others require samples across longer time periods. We leave a complete temporal sensitivity analysis to future work.

Acknowledgments

Thanks to kc claffy, Justin Rohrer, Nick Weaver, and Geoffrey Xie for invaluable feedback. This work supported by in part by NSF grant CNS-1111445 and Department of Homeland Security (DHS) S&T contract N66001-2250-58231. Views and conclusions are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. government.

References

1. Alexa: Top 1,000,000 sites (2014), `http://www.alexa.com/topsites`
2. Berger, A., Weaver, N., Beverly, R., Campbell, L.: Internet Nameserver IPv4 and IPv6 Address Relationships. In: Proceedings of the ACM Internet Measurement Conference. pp. 91–104 (2013)
3. claffy, k.: Tracking IPv6 evolution: data we have and data we need. SIGCOMM Comput. Commun. Rev. 41(3), 43–48 (Jul 2011)
4. Craven, R., Beverly, R., Allman, M.: A Middlebox-cooperative TCP for a Non End-to-end Internet. In: Proceedings of ACM SIGCOMM. pp. 151–162 (2014)
5. Czyz, J., Allman, M., Zhang, J., Iekel-Johnson, S., Osterweil, E., Bailey, M.: Measuring IPv6 Adoption. In: Proceedings of ACM SIGCOMM. pp. 87–98 (2014)
6. Dhamdhere, A., Luckie, M., Huffaker, B., Elmokashfi, A., Aben, E., et al.: Measuring the deployment of IPv6: topology, routing and performance. In: Proceedings of the ACM Internet Measurement Conference. pp. 537–550 (2012)
7. Heuse, M.: Recent advances in IPv6 insecurities. In: Chaos Communications Congress (2010)
8. Jacobson, V., Braden, R., Borman, D.: TCP Extensions for High Performance. RFC 1323 (May 1992)
9. Kohno, T., Broido, A., Claffy, K.C.: Remote physical device fingerprinting. In: Proceedings of IEEE Security and Privacy. pp. 211–225 (2005)
10. Lyon, G.F.: Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning (2009)
11. Maxmind: IP Geolocation (2014), `http://www.maxmind.com`
12. Meyer, D.: University of Oregon RouteViews (2014), `http://www.routeviews.org`
13. Moon, S., Skelly, P., Towsley, D.: Estimation and removal of clock skew from network delay measurements. In: Proceedings of INFOCOM. vol. 1 (mar 1999)
14. RIPE NCC: World IPv6 Day Measurements (2011), `http://v6day.ripe.net`
15. Sarrar, N., Maier, G., Ager, B., Sommer, R., Uhlig, S.: Investigating IPv6 traffic: what happened at the world IPv6 day? In: Proceedings of PAM (2012)
16. Silbersack, M.J.: Improving TCP/IP security through randomization without sacrificing interoperability. In: Proceedings of BSDCan (2006)
17. Zander, S., Andrew, L.L., Armitage, G., Huston, G., Michaelson, G.: Mitigating Sampling Error when Measuring Internet Client IPv6 Capabilities. In: Proceedings of the ACM Internet Measurement Conference. pp. 87–100 (2012)