Partitioning the Internet using Anycast Catchments

Kyle Schomp
Akamai Technologies
kschomp@akamai.com

Rami Al-Dalky
Case Western Reserve University
rami.al-dalky@case.edu

ABSTRACT

In anycast deployments, knowing how traffic will be distributed among the locations is challenging. In this paper, we propose a technique for *partitioning* the Internet using passive measurements of existing anycast deployments such that all IP addresses within a partition are routed to the same location for an arbitrary anycast deployment. One IP address per partition may then represent the entire partition in subsequent measurements of specific anycast deployments. We implement a practical version of our technique and apply it to production traffic from an anycast authoritative DNS service of a major CDN and demonstrate that the resulting partitions have low error even up to 2 weeks after they are generated.

CCS CONCEPTS

• **Networks → Network measurement**; **Routing protocols**; **Network dynamics**;

KEYWORDS

Anycast, Measurement

1 INTRODUCTION

Anycast is a popular [7, 17, 24] technique for distributing traffic among multiple physical locations, also known as points of presence, or *PoPs*, on the Internet. IP Anycast is frequently used to provide authoritative and recursive DNS services (e.g., [1, 18, 25]), as well as by content delivery networks (CDNs) (e.g., [9, 16, 22]). By advertising, via BGP [26], the same IP prefix from each PoP, traffic from any source on the Internet is routed to one of the PoPs according to properties of the path between the source and the PoPs. Routers along that path select between multiple competing routes to the anycast prefix according to a combination of best practices and unique policies enacted by individual autonomous systems (ASs).

Knowing how traffic will be distributed among the PoPs advertising an anycast prefix a priori is a challenging problem due to incomplete knowledge of AS relationships and the varying policies implemented by ASs, which can be complex [6]. As a direct result, there is a great deal of operational complexity in monitoring and modifying the advertisements of production anycast services [2]. Operators tune advertisements to both improve performance by routing sources to a nearby PoP and balance load by shifting traffic away from overloaded PoPs. Without the ability to predict to which PoP sources will be routed, advertisement changes are often made without complete knowledge of their impact.

Recently, Verfploeter [12] is proposed as a powerful method to measure how traffic will be split among PoPs of an anycast service *before* it receives production traffic. By active scanning with ICMP Echo Requests *from* the anycast IP address to target IP addresses on the Internet, Verfploeter identifies which PoP will receive traffic from each target by where the Echo Reply is returned. Techniques for scanning the entire IPv4 Internet exist [13], however are likely to produce incomplete results because many hosts do not respond to ICMP [19]. Further, scanning all of IPv4-space is onerous and does not scale well to many parties conducting their own scans, and – looking forward – full scans of IPv6-space are impractical. The authors of [12] propose scanning one IP address per /24 prefix on the Internet chosen to be representative of the entire /24 prefix, leveraging work in [15]. This method, however, will result in (*i*) excess scanning when prefixes less specific than 24-bits are all routed to the same PoP and (*ii*) inaccurate representation when IP addresses within the same /24 prefix are routed to different PoPs, which we demonstrate occurs in Section 4.

In this paper, we propose a novel technique using passive measurements of existing anycast deployments that splits IP-space into *consistently routed* partitions, i.e., all IP addresses within a partition are routed to the same PoP for an arbitrary anycast deployment. Then, one IP address per partition may be selected and used in active measurements such as [12] to represent all IP addresses within the partition. We make the following contributions:

- Propose a theoretical approach using deployed anycast services for creating a partitioning of the Internet along differences in routing to anycast PoPs.
- Present a practical implementation of the approach that solves real-world problems with production anycast data. We validate and measure the error in the resulting partitions, showing it's effectiveness at predicting which PoP anycast traffic will be routed to over time and across anycast deployments.

We use the following terminology in the paper. All IP addresses routed to the same PoP for a given anycast prefix are in the same *catchment*. The set of PoPs and policies used to advertise a given anycast prefix are an anycast *deployment*. The PoP that a given IP address is routed to on an anycast prefix is the *sink*.

2 THEORETICAL APPROACH

Anycast deployments split the Internet into the catchments for each PoP in the deployment. The catchments are a function of the deployment itself and routing on the Internet. Our general idea is to combine the catchment information from multiple anycast deployments to learn about the underlying Internet topology and routing policies that form catchments independently of the anycast deployments themselves. By combining the catchment information from enough anycast deployments, we create a mapping of IP addresses into partitions where the IP addresses within a partition are always part of the same catchment regardless of anycast deployment.

First, consider a simple deployment of a single anycast prefix a_1 advertised from two PoPs, l_1 and l_2. We use two PoPs for ease of exposition, but the method generalizes to any number of PoPs. Logically, the Internet is split into the catchment for sink l_1 and the catchment for sink l_2, defined as $P_{a_1 \to l_1}$ and $P_{a_1 \to l_2}$, respectively. Note that the two catchments need not be composed of contiguous IP-space. Thus, $P_{a_i \to l_i}$ in general represents a set of prefixes.

Next, take a second anycast prefix a_2 again advertised from two topologically different PoPs, l_3 and l_4. Again, the Internet is split in two: $P_{a_2 \to l_3}$ and $P_{a_2 \to l_4}$. Together, a_1 and a_2 split the Internet into 4 possible partitions: $P_{a_1 \to l_1, a_2 \to l_3}$, $P_{a_1 \to l_2, a_2 \to l_3}$, $P_{a_1 \to l_1, a_2 \to l_4}$, and $P_{a_1 \to l_2, a_2 \to l_4}$. Any partitions that are empty can trivially be ignored.

Using n anycast deployments, $a_1, a_2, a_3, ..., a_n$, the Internet is split into up to 2^n non-empty partitions. As $n \longrightarrow \infty$, the number of partitions is bounded by the number of IP addresses on the Internet. Practically, however, we expect the number of partitions to be far less as many hosts on the Internet will be routed to the same sink due to the hosts residing in the same network and their traffic traversing the same paths. We call the discovered partitions *consistently routed* because all IP addresses within the partition are always routed to the same PoP regardless of anycast deployment.

The catchment of any IP address within a partition is representative of all IPs in the partition. Thus, one IP address may be selected from within each partition and used in subsequent measurements with the understanding that the observed routing of the selected IP address applies to the entire partition.

Note that many types of network routing changes caused by traffic engineering will not perturb the mapping of IP addresses to partitions. For example, an AS re-routing traffic from one path to another does not invalidate the mapping because all IP addresses within any partition should still be routed to the *same* sink. Conversely, if the AS begins splitting traffic among multiple paths, then a partition may incorrectly include IP addresses routed to different sinks.

3 IMPLEMENTATION

In this section, we describe a method to generate the partitions using real world data from a production system and discuss algorithmic adjustments to the theory needed to address real world problems.

3.1 Dataset

To create the partitions as described in Section 2, we use logs captured from an anycast authoritative DNS service of a major content delivery network (CDN). The service has 22 IPv4 anycast prefixes, each used for authoritative hosting of a wide variety of DNS zones and advertised from many, globally distributed PoPs. The PoPs advertising different anycast prefixes are typically distinct, but in some cases overlap. Each PoP has on average dozens of BGP peers making the 22 anycast deployments examples of "many-provider" configurations [23]. Logs from the authoritative nameservers within the PoPs include the source IP address and the destination IP address—which is from one of the 22 anycast prefixes—of the DNS queries. From the nameserver that logs the query, we can infer the sink PoP of the source IP address for the anycast prefix. We use these DNS traffic logs to create the partitions, but note that the partitions could be generated from logs for any type of anycast traffic, including web access. We collect 5 days of logs and summarize the data in Table1 as datasets D_0 through D_*. Further, the D_{IPv6} dataset is collected from logs of the 22 IPv6 anycast prefixes used by the same anycast authoritative DNS service (discussed in Section 4.4), and the R dataset is collected from an anycast recursive resolver service (discussed in Section 4.5).

Name	Date	Queries	Source IPs	/24 Prefixes	ASNs	Countries
D_0	2019-02-07	129B	5.1M	1.5M	44K	240
D_{1d}	2019-02-08	139B	5.0M	1.5M	41K	243
D_{1w}	2019-02-14	158B	5.5M	1.6M	42K	243
D_{2w}	2019-02-21	168B	5.4M	1.6M	44K	240
D_*	2018-06-14	138B	5.8M	1.6M	47K	238
R	2019-02-21	2.26B	385K	107K	2952	84
D_{IPv6}	2020-03-24	17.4B	236K (/64)	104K (/48)	6839	171

Table 1: Datasets used in this paper

Since our theoretical approach expects data from all IP addresses on the Internet, we attempt to estimate how complete our datasets are. In addition to the number of DNS queries in each dataset and the number of source IP addresses, we compute the number of /24 prefixes that the source IP addresses are within, use Team Cymru [29] to determine the ASN, and use the commercially available geolocation service EdgeScape [14] to determine the country code of each source IP address in our datasets. D_0 through D_* all have similar properties. First, in terms of IP addresses, our datasets cover roughly 0.1% of the approximately 3.7B total IPv4 addresses excluding reserved space [21] and 0.5% of the 1.1B IPv4 addresses estimated to be in use as of 2014 [32]. Clearly, our datasets are not exhaustive. However, the number of active /24 prefixes on the Internet is estimated as 4.8M in 2013 [10] and 6.3M in 2014 [32] via passive measurement, and 5.3M in 2017 [4] via active scanning, of which our datasets cover between 24% and 31%. The number of active /24 prefixes is likely higher today, but we could not find a more recent census. Further, 65K ASNs appear in routing tables [5] and our datasets cover 63-72% of that number. Finally, EdgeScape recognizes 248 country codes of which 96-98% are covered in our datasets. Thus, we conclude that, while our datasets do not include all IP addresses, the coverage of our datasets is sufficient to produce interesting and meaningful results, yet with some error. We attempt to quantify the error in Section 4.2.

3.2 Algorithm

In this section, we describe a practical method of generating the anycast partitions. The input to our algorithm is the total hits (DNS queries) arriving at each PoP from each IP address to each anycast prefix over a 24 hour period. We use a time window of one day to avoid missing parts of the world due to diurnal usage patterns. Further, we show in Section 4.2 that 24 hours is sufficient to predict the catchments observed in the subsequent day with low error. The algorithm involves three steps.

Step 1. First, dealing with each anycast deployment individually, we find *max-prefixes*, the least-specific prefixes covering source IP addresses all routed to the same sink. The intuition for this step is that IP addresses not observed in the dataset are likely routed to the same sink as observed IP addresses within the same max-prefix. Using a greedy approach, the algorithm loops through each source IP address r in turn and, using r as a root, finds the covering prefix of r where (i) all covered IP addresses are routed to the same sink, (ii) none of the covered IP addresses are already covered by a different max-prefix, and (*iii*) the covering prefix cannot be further expanded without violating (*i*) or (*ii*).

However, requiring that all IP addresses are routed to the same sink can be too strict. In the D_0 dataset, we observe that 890K (17%) IP addresses are routed to multiple PoPs for the same anycast prefix over the 24 hour period, i.e., change catchment during the day. There are many possible causes for this observation, some long time scale and some short, including:

- Rare route instability [20, 30, 31] produces frequent changes between multiple paths.
- Maintenance events both in the PoPs and in networks along the path can cause all traffic routed to one PoP to shift to another for time periods ranging from minutes to hours.
- The DNS queries from (predominantly) recursive resolvers nearly always use an ephemeral UDP port number for each query [27]. Thus, traffic engineering techniques that hash the port numbers (e.g., equal-cost multi-path) may spread traffic from a single IP address among multiple PoPs on a packet-by-packet basis.
- Other traffic engineering that causes traffic to shift between routes at specific times, e.g., due to traffic volume increases during peak times.

We make no attempt to distinguish between these causes of routing changes, noting that all sources of routing changes are to be expected in production systems. Practical implementations of our methodology must contend with routing changes and we design our algorithm to explicitly take them into account as follows.

Algorithm 1 shows a refined step 1 to deal with route changes. Instead of looking for covering prefixes where all covered IP addresses are routed to a single PoP, we relax the constraint by looking for covering prefixes where all covered IP addresses are routed to the same set of PoPs over the day. Thus, if the root switches sink from A to B during the 24-hours, we look for other IP address that have also switched between the same two sinks during the day.

Further, we observe that many IP addresses send few queries throughout the day, 44% of IP addresses in the D_0 dataset sent less than 10 DNS queries to at least one anycast prefix. We cannot be confident that we observe all routing changes that impacted these IP addresses throughout the 24-hour period. Thus, we may only have samples in our dataset from when the IP addresses were in the catchment of A or B and not both. To handle low sampling, the algorithm iterates through IP addresses in order of descending total hits to build covering prefixes based upon roots where we have higher confidence in our observations due to abundant samples. Next, the covering prefix is expanded to cover IP addresses if the set of PoPs that they are routed to intersects with the set of PoPs to which the root is routed. Finally, we add a threshold minimum number of hits within a covering prefix and keep expanding the covering prefix if below the threshold to limit the generation of small max-prefixes that represent very few samples. Empirically, we find a low threshold value of approximately 10 is sufficient and produces reasonable results.

Algorithm 1: STEP 1. Find the least-specific prefixes covering IP addresses all routed to the same sink(s)

```
Input: $S \leftarrow$ list of all IPs
Input: $h[s,p] \leftarrow$ # of hits from IP $s$ to PoP $p$
Input: T $\leftarrow$ a threshold value
covered[$s$] $\leftarrow$ false for $s$ in $S$
for root in S ordered by hits desc do
    if covered[root] then
        continue
    max-prefix $\leftarrow$ root
    $P_r \leftarrow$ set $p$ where $h[\text{root},p] > 0$
    for $r_{cover} \leftarrow$ 31 to 1 bit prefix of root do
        $S_r \leftarrow$ set $s$ in $S$ covered by $r_{cover}$
        if covered[$s$] for any $s$ in $S_r$ then
            break
        $t \leftarrow \sum h[s,p]$ for $s$ in $S_r$
        if $t \geq T$ & $h[s,p] > 0$ for $p$ not in $P_r$ & $s$ in $S_r$ then
            break
        covered[$s$] $\leftarrow$ true for $s$ in $S_r$
        max-prefix $\leftarrow r_{cover}$
    yield max-prefix
```

Step 2. Next, the algorithm merges the max-prefixes for each anycast deployment together to form consistently routed prefixes. If there is a conflict between the max-prefixes (i.e., one max-prefix is a subnet of another), the more-specific one wins. For example, in the scenario where 1.2.3.0/24 and 1.2.3.0/26 both appear in the merger, 1.2.3.0/24 is split into 1.2.3.0/26, 1.2.3.64/26, and 1.2.3.128/25. After this step, the remaining prefixes, *consistent-prefixes*, each contain IP addresses that are consistently routed, with some introduced error.

Step 3. The consistent-prefixes are collected to form partitions by matching the sinks of the consistent-prefixes across all anycast deployments. Using the above example again, 1.2.3.64/26 and 1.2.3.128/25 become part of the same partition.

4 ANALYSIS

The partitioning created from the D_0 dataset contains 484K consistent-prefixes. Figure 1 shows their lengths. The most common prefix length is 22-bits but, surprisingly, there are many more specific than 24-bits including nearly 12K /32 prefixes, i.e., individual IP addresses that are routed differently than the IP addresses numerically next to them. Some of these IP addresses are operated by our institution and we are able to manually verify their routing behavior. We confirm that the hosts in the entire covering /24 prefix are routed via one of two peering links depending upon the value of the most specific bit in the IP address, generating many /32 consistent-prefixes. Thus, every other IP address in the /24 prefix is routed identically and the /32 consistent-prefixes readily merge into partitions in the final step of our algorithm. In total, 245K (4.8%) of the IP addresses in the D_0 dataset are covered by consistent-prefixes that are more specific than 24-bits, structure that would be lost if a single IP address is used as representative of the entire /24 prefix, as in [12].

The 484K consistent-prefixes collect into 212K partitions in the final step of our algorithm. We explore the consistency of the partitions by computing the number of ASNs each partition covers and find that 94% of partitions consist of IPs from a single ASN, and

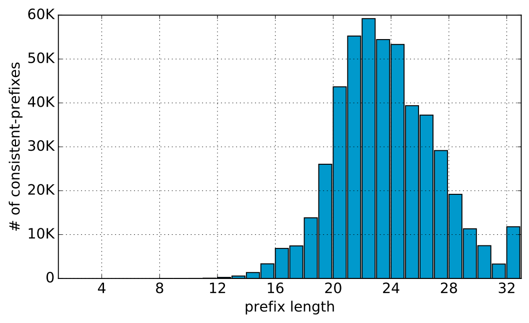

Figure 1: Lengths of consistent-prefixes from D_0

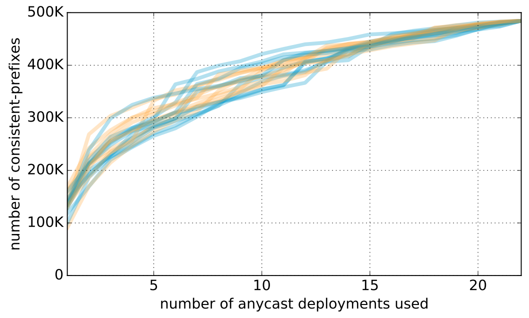

Figure 2: Number of consistent-prefixes for different random permutations of the 22 anycast deployments used

98% consist of no more than two ASNs. We expect different ASNs would produce different routing and this result agrees with that intuition. Conversely, the IP-space of 37% of the 44K ASNs in D_0 is split across multiple partitions, higher than observed in [12] likely due to the larger number of PoPs in our study. This result is also expected as it is common for ASNs to have multiple peering links as in the /32 prefix example above.

4.1 Is 22 Anycast Prefixes Enough?

In Section 2, we propose the use of a potentially infinite number of anycast deployments to generate the partitions. Practically, however, the number of anycast deployments used is finite and we have data available from 22. To answer whether 22 anycast deployments is sufficient to produce partitions with the desired properties, we generate partitionings with subsets of the 22 anycast deployments where the set size varies from 1 to 21 anycast deployments. Figure 2 shows the number of consistent-prefixes found as a function of the number of anycast deployments used. Each line (20 total) is a different random permutation of the 22 anycast deployments. We expect the slope of the curve to approach zero as the number of anycast deployments becomes sufficient to discover all consistent-prefixes. However, the curves clearly still have a positive slope approaching 22 deployments. Thus, 22 anycast deployments is insufficient to discover all consistent-prefixes and the partitioning is incomplete. The next sections quantify the resulting error.

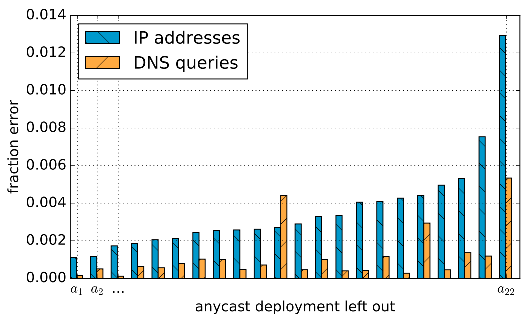

Figure 3: Cross validation of partitions where one anycast deployment is left out at a time

4.2 Estimating Error in Partitions

The method of generating the partitions in the presence of routing changes described in Section 3 introduces some error. However, measuring that error is not straightforward. Recall that the partitions indicate that all IP addresses should be consistently routed, not to which sink they are routed. Therefore, we compute error as follows. For each partition, first calculate the most frequent sink (or sinks in the presence of routing changes) in terms of DNS queries. Any IP addresses in the partition routed to a different sink add error. In terms of IP addresses, the fraction error is the sum of the number of erroneous IP addresses across partitions divided by the total IP addresses. However, since the volume of DNS queries per IP address is skewed, we also calculate error in terms of DNS queries by summing the erroneous DNS queries across partitions and dividing by the total hits in the dataset. In the D_0 dataset, the error is 0.15% in IP addresses and 0.03% in DNS queries, showing that the method for dealing with routing changes adds a small amount of error.

The finite number of anycast deployments and the lack of traffic from all IP addresses are also sources of error. Next, we perform cross validation to estimate the predictive capability of the partitions on IP addresses and anycast deployments not part of the training data. Note that 34% of the IP addresses in the D_0 dataset only ever sent DNS queries to a single anycast deployment. We perform cross validation by withholding data for one anycast deployment at a time, generating partitions using the remaining 21 anycast deployments, and then calculate error on the withheld data. The resulting errors per anycast deployment left out are shown in Figure 3. The maximum error is 1.29% in IP addresses and 0.53% in DNS queries, while the average error is 0.36% in IP addresses and 0.12% in DNS queries. The error remains low despite the introduction of IP addresses and anycast deployments not in the training data and the variation in the error indicates that some anycast deployments include more distinct catchment information than others, likely due to varied application use and the peers present at the deployments PoPs.

To be useful for many applications, the partitions generated at time t must still be able to accurately predict behavior at time $t + \Delta$. We investigate the error in the partitions over time by using other datasets in Table 1. Figure 4 shows the error for values of Δ from 1 day (1d) to 8 months (8m) where the partitions are generated

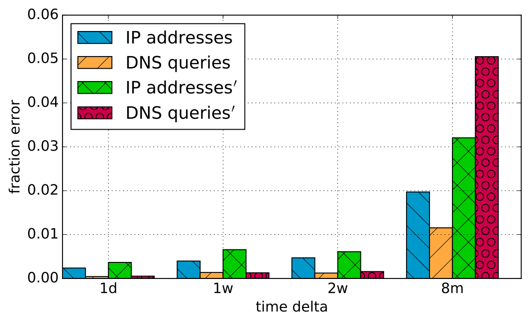

Figure 4: Error in the partitions over time

from the D_0 dataset and then applied to future datasets. Note that 8 months is included as an extreme example and is exceptional in that the partitions are generated using the D_* dataset and applied to the D_{1w} dataset. The bars labeled *IP addresses* and *DNS queries* correspond to our two measures of error above. We introduce two new measures of error in *IP addresses′* and *DNS queries′* where the IP addresses over which the error is calculated are restricted to only those in the dataset used to compute error but not in the dataset used to generate the partitions, and thus anticipated to be more prone to error. The error after 8 months is predictably large, suggesting that the partitions should be recomputed over time. However, at a 2 week Δ – which we believe to be a reasonable lifespan – the error by all metrics remains below 1% despite only 56% of the IP addresses in the D_{2w} dataset are present in the D_0 dataset and 28% of the ones present are routed to different sinks. Thus, the routing policies driving consistent routing in our partitioning are relatively stable over at least two weeks even though source IP addresses and the sinks of consistent-prefixes are themselves unstable.

4.3 Alternative Methods

Here, we compare the above error rates to those computed using alternative methods. We compare our method using a 1 week Δ to two alternative methods: (*i*) split the dataset into /24 prefixes as in [12] and (*ii*) use the prefixes visible in BGP route tables obtained via Team Cymru [29]. The latter uses the reachability information advertised by other ASNs to split up the IP addresses in the dataset accordingly. To compare the methods, we must consider both error rate and number of partitions as there is a trivial, and useless, solution when the Internet is split into /32 prefixes. Such a solution has zero error but both increases the number of IP addresses (and measurements) needed to represent the Internet and cannot predict the routing of any unresponsive IP addresses.

Table 2 in the first three rows shows the methods in order of descending number of partitions. The "/24 Prefixes" method produces low error but at the cost of a very large number of partitions. Contributing to the error is the observation above that 4.8% of IP addresses are not part of the same catchment as other IP addresses in their /24 prefix. The "BGP Prefixes" method creates fewer partitions but the error is very large. This is expected as route tables contain routes *to* an IP address whereas our goal is to estimate routing *from* the IP address and routing need not be symmetric. In

	Method	# Partitions	Error (IPs)	Error (Queries)
IPv4	/24 Prefixes	1.55M	0.77%	0.44%
	BGP Prefixes	229K	5.46%	6.70%
	Anycast Partitions	212K	0.13%	0.04%
IPv6	/48 Prefixes	104K	0.41%	7.01%
	Anycast Partitions	42.0K	0.39%	0.05%
	BGP Prefixes	13.4K	7.45%	10.13%

Table 2: Comparison with alternative methods

all, 17.1K (7.5%) of the 229K prefixes in BGP tables are split between multiple catchments, showing that IP addresses within BGP route prefixes are frequently *not* similarly routed. Of the three methods, the anycast partitioning generates both the fewest partitions and the lowest error.

4.4 IPv6

Our analysis has focused exclusively on IPv4. However, our methodology is equally applicable to IPv6. We investigate IPv6 using the D_{IPv6} dataset, which is much smaller than the other datasets captured from the authoritative DNS service due to limited IPv6 support in recursive resolvers. Running the partitioning algorithm on this dataset produces 42K consistent-prefixes and 19K partitions. Table 2 in the bottom three rows compares the anycast partitioning method to splitting the dataset by /48 prefixes and prefixes in BGP route tables as described above. For IPv6, anycast partitioning does not generate the fewest partitions but does have the lowest error. We believe these results show that our technique will become increasingly preferable as IPv6 usage grows and scanning IP-space – even one IP address per active /48 prefix – becomes infeasible.

4.5 Use Across Client Populations

As shown in Section 3.1, none of our datasets have complete coverage of the Internet. Thus, the partitions generated may only be representative for the subset of IP-space used in their creation. In particular, the IP addresses in authoritative DNS service logs are typically recursive resolvers. Thus, the partitions may be specific to recursive resolvers and miss catchment structure for other client populations.

To explore this notion, we use a dataset of DNS logs from an anycast recursive resolver service where the clients are stub resolvers, likely located in edge or home networks. Shown in Table 1 as R, we measure the overlap between R and D_{2w}, which were collected on the same day. As expected, we find that only 6% of IP addresses and 35% of their covering /24 blocks in R are also in the D_{2w} dataset. For comparison, 60% of IP addresses and 83% of /24 blocks in the D_{2w} dataset are also in the D_{1w} dataset, a week apart. The error on R using partitions from D_{1w} is 8% in IP addresses and 3% in DNS queries, much higher than the 0.13% and 0.04%, respectively, reported in Table 2. Thus, we conclude that the partitions must be generated from the traffic of the same client populations where the partitions are intended to be used.

5 RELATED WORK

de Vries et al. [12] propose a technique for measuring anycast catchments of responding IP addresses on the Internet, sampling one IP address per /24 prefix as in [15]. Our work is complementary to their work, showing that if the sampling is done strategically using passive measurement from deployed anycast services then both the number of samples and the error in estimated catchments can be reduced. While a /24 prefix is the smallest routable prefix on the Internet, traffic from a /24 prefix may still be split among anycast catchments which will be missed by sampling one IP address per /24 prefix. Sermpezis and Kotronis [28] propose a method that can infer catchments with uncertainty. The uncertainty in the inferred catchments depends upon the amount of data available on the network graph and the policies implemented by ASs. In contrast, our method has no reliance on knowledge of the network graph or specific policies, instead relying upon passive measurement from deployed anycast services. Both methods aim for the same goal and which method will produce a more accurate depiction of catchments likely depends upon the data available for analysis. de Oliveira Schmidt at al. [11] show that surprisingly few intelligently placed PoPs are needed to provide good global latency for an anycast service. Our work, in conjunction with the scanning proposed in [12], can aid in intelligently placing future PoPs by analyzing the catchments formed by existing PoPs. Many works evaluate anycast deployments using probing traffic [3, 7, 8]. We take the opposite approach, evaluating catchments by passive measurement of traffic to anycast deployments.

6 CONCLUSION & FUTURE WORK

In this paper, we propose a technique for partitioning the Internet using passive measurements of existing anycast deployments that splits IP-space into *consistently routed* partitions, i.e., all IP addresses within a partition are part of the same catchment for an arbitrary anycast deployment. To implement the technique, we present an algorithm for merging the catchments of multiple anycast deployments together. The resulting partitions can predict consistent routing with low error even 2 weeks after the partitions are generated. Finally, in comparison to alternative methods, we demonstrate that the partitions generated from anycast catchments are both fewer – and thus able to represent the Internet with less active measurements – and lower error.

We highlight the following areas for future work. (*i*) Routing changes in the training data pose a problem for our methodology. We propose a technique described in Section 3.2 to deal with routing changes and demonstrate good results using it. However, other solutions producing better results may exist. (*ii*) We focus on the use of the partitions for predicting anycast catchments. However, changes in catchment are also indicative of routing changes in general. Knowing the boundaries within IP-space where routing changes occur is likely beneficial in other measurement studies as well. (*iii*) Multiple anycast prefixes were available to us for our analysis, however multiple anycast prefixes is not inherently necessary to produce partitions. A single anycast prefix could be used, iterating through multiple deployments instead. This may allow use of many more than the 22 anycast deployments in this paper which, as noted in Section 4.1, is too few. There remains, however, a need for multiple distinct deployments.

REFERENCES

[1] 1.1.1.1 2019. *Cloudflare 1.1.1.1 Public Recursive Resolver.* Retrieved June 2019 from https://1.1.1.1/

[2] Joe Abley and K Lindqvist. 2006. *Operation of Anycast Services.* RFC 4786. https://tools.ietf.org/html/rfc4786

[3] Hitesh Ballani, Paul Francis, and Sylvia Ratnasamy. 2006. A Measurement-based Deployment Proposal for IP Anycast. In *ACM Internet Measurement Conference.*

[4] Shehar Bano, Philipp Richter, Mobin Javed, Srikanth Sundaresan, Zakir Durumeric, Steven J Murdoch, Richard Mortier, and Vern Paxson. 2018. Scanning the Internet for Liveness. *ACM SIGCOMM Computer Communication Review* (2018).

[5] Tony Bates, Philip Smith, and Geoff Huston. [n.d.]. *CIDR Report.* Retrieved June 2019 from http://www.cidr-report.org/as2.0/

[6] Matthew Caesar and Jennifer Rexford. 2005. BGP Routing Policies in ISP Networks. *IEEE Network* (2005).

[7] Danilo Cicalese, Jordan Augé, Diana Joumblatt, Timur Friedman, and Dario Rossi. 2015. Characterizing IPv4 Anycast Adoption and Deployment. In *ACM Conference on emerging Networking EXperiments and Technologies.*

[8] Danilo Cicalese, Diana Joumblatt, Dario Rossi, Marc-Olivier Buob, Jordan Augé, and Timur Friedman. 2015. A Fistful of Pings: Accurate and Lightweight Anycast Enumeration and Geolocation. In *IEEE Conference on Computer Communications.*

[9] CloudFlare 2019. *Cloudflare, Inc.* Retrieved June 2019 from https://www.cloudflare.com/

[10] Alberto Dainotti, Karyn Benson, Alistair King, Michael Kallitsis, Eduard Glatz, Xenofontas Dimitropoulos, et al. 2013. Estimating Internet Address Space Usage through Passive Measurements. *ACM SIGCOMM Computer Communication Review* (2013).

[11] Ricardo de Oliveira Schmidt, John Heidemann, and Jan Harm Kuipers. 2017. Anycast latency: How Many Sites are Enough?. In *Passive and Active Measurement Conference.*

[12] Wouter B De Vries, Ricardo de O Schmidt, Wes Hardaker, John Heidemann, Pieter-Tjerk de Boer, and Aiko Pras. 2017. Broad and Load-Aware Anycast Mapping with Verfploeter. In *ACM Internet Measurement Conference.*

[13] Zakir Durumeric, Eric Wustrow, and J Alex Halderman. 2013. ZMap: Fast Internet-wide Scanning and Its Security Applications. In *USENIX Security Symposium.*

[14] EdgeScape 2019. *Akamai EdgeScape Geolocation Service.* Retrieved June 2019 from https://developer.akamai.com/edgescape

[15] Xun Fan and John Heidemann. 2010. Selecting Representative IP Addresses for Internet Topology Studies. In *ACM Internet Measurement Conference.*

[16] Fastly 2020. *Fastly, Inc.* Retrieved March 2020 from https://www.fastly.com

[17] Danilo Giordano, Danilo Cicalese, Alessandro Finamore, Marco Mellia, Maurizio M Munafò, Diana Zeaiter Joumblatt, and Dario Rossi. 2016. A First Characterization of Anycast Traffic from Passive Traces. In *Network Traffic Measurement and Analysis Conference.*

[18] Google Public DNS 2019. *Google Public DNS.* Retrieved June 2019 from https://developers.google.com/speed/public-dns/

[19] John Heidemann, Yuri Pradkin, Ramesh Govindan, Christos Papadopoulos, Genevieve Bartlett, and Joseph Bannister. 2008. Census and Survey of the Visible Internet. In *ACM Internet Measurement Conference.*

[20] James Hiebert, Peter Boothe, Randy Bush, and Lucy Lynch. 2006. Determining the Cause and Frequency of Routing Instability with Anycast. In *Asian Internet Engineering Conference.*

[21] Internet Assigned Numbers Authority. [n.d.]. *IPv4 Address Space Registry.* Retrieved June 2019 from https://www.iana.org/assignments/ipv4-address-space/ipv4-address-space.xhtml

[22] Limelight 2019. *Limelight Networks.* Retrieved June 2019 from https://www.limelight.com/

[23] Stephen McQuistin, Sree Priyanka Uppu, and Marcel Flores. 2019. Taming Anycast in the Wild Internet. In *Proceedings of the Internet Measurement Conference.* 165–178.

[24] Craig Partridge, Trevor Mendez, and Walter Milliken. 1993. *Host Anycasting Service.* RFC 1546. http://www.rfc-editor.org/rfc/rfc1546.txt

[25] Quad9 2019. *Quad9 DNS Service.* Retrieved June 2019 from https://www.quad9.net/

[26] Yakov Rekhter, Tony Li, and Susan Hares. 2006. *A Border Gateway Protocol 4 (BGP-4).* RFC 4271. https://tools.ietf.org/html/rfc4271

[27] Kyle Schomp, Tom Callahan, Michael Rabinovich, and Mark Allman. 2014. Assessing DNS Vulnerability to Record Injection. In *Passive and Active Measurement Conference.*

[28] Pavlos Sermpezis and Vasileios Kotronis. 2019. Inferring Catchment in Internet Routing. *ACM Measurement and Analysis of Computing Systems* (2019).

[29] Team Cymru 2019. *Team Cymru - IP to ASN Mapping.* Retrieved June 2019 from http://www.team-cymru.com/IP-ASN-mapping.html

[30] Lan Wei and John Heidemann. 2017. Does Anycast Hang Up on You?. In *Network Traffic Measurement and Analysis Conference.*

[31] Yang Richard Yang, Haiyong Xie, Hao Wang, Avi Silberschatz, Arvind Krishnamurthy, Yanbin Liu, and Li Erran Li. 2005. On Route Selection for Interdomain Traffic Engineering. *IEEE Network* 19, 6 (2005), 20–27.

[32] Sebastian Zander, Lachlan LH Andrew, and Grenville Armitage. 2014. Capturing Ghosts: Predicting the Used IPv4 Space by Inferring Unobserved Addresses. In *ACM Internet Measurement Conference.*