# Evaluating Anycast in the Domain Name System

Xun Fan John Heidemann Ramesh Govindan
University of Southern California / Information Sciences Institute and Computer Science Dept.

***Abstract*—IP anycast is a central part of production DNS. While prior work has explored proximity, affinity and load balancing for some anycast services, there has been little attention to third-party discovery and enumeration of components of an anycast service. Enumeration can reveal abnormal service configurations, benign masquerading or hostile hijacking of anycast services, and help characterize anycast deployment. In this paper, we discuss two methods to identify and characterize anycast nodes. The first uses an existing anycast diagnosis method based on CHAOS-class DNS records but augments it with traceroute to resolve ambiguities. The second proposes Internet-class DNS records which permit accurate discovery through the use of existing recursive DNS infrastructure. We validate these two methods against three widely-used anycast DNS services, using a very large number (60k and 300k) of vantage points, and show that they can provide excellent precision and recall. Finally, we use these methods to evaluate anycast deployments in top-level domains (TLDs), and find one case where a third-party operates a server masquerading as a root DNS anycast node as well as a noticeable proportion of unusual DNS proxies. We also show that, across all TLDs, up to 72% use anycast.**

## I. Introduction

Rapid response and high availability requires that large network services be distributed widely, often with a single logical service is provided by distributed replicas accessed using a single logical identifier. Content delivery networks (for example, [13]), mirroring services (for example, [12]), URNs [38], and IP anycast [33] all fit this model.

In IP anycast, as standardized by the IETF [1], [33], an anycast service operates on a single IP address, but multiple *anycast nodes* replicate that service at different physical locations. Each node may be implemented with one or more *servers* (a physical or virtual machine), each of which listens to the anycast address and often also one or more unicast addresses. Standard interdomain routing directs clients to the nearest replica and handles fail-over to other nodes as required. (We review details and terms in Section II.)

Anycast is used for many core services of the Internet today. It is widely used for DNS [19]: as of April 2011, 10 out of 13 root name servers employ anycast [35]. Other uses include discovering IPv6-to-IPv4 relay routers [21] and sinkholes [18] and for load distribution [14], [39]. Anycasted services benefit from anycast's load balancing and ability to mitigate denial-of-service attacks [1], and research proposals have discussed improvements to scale to many anycast destinations [23].

The use of anycast for core Internet services suggests we need to understand its performance, use, and robustness. In this paper, we focus on understanding anycast use in DNS. Extensive prior work (Section VI) has measured server proximity, the affinity between clients and anycast services, and the performance of load balancing of anycasted DNS. However, to date there has been no effort to discover, map, and fully enumerate anycast use in DNS. As we show in Section V, such a capability can aid in diagnosing abnormal name service configurations, and help understand anycast deployment.

The first contribution of our work is to evaluate different approaches to automatically *discover* and *enumerate* all nodes of an anycast service. To understand the challenges in anycast discovery, we first taxonomize anycast deployment configurations (Section II). Anycast discovery is challenging because anycast configurations can be complex, existing diagnosis methods are not standardized that leads to measurement ambiguity, and the visibility of anycast servers can be topologically scoped requiring a large number of vantage points (unique observation locations).

We then discuss the design of *two methods to enumerate anycast nodes*. The first method uses an existing anycast diagnosis technique based on CHAOS-class TXT DNS records [41], but augments it with traceroute to identify non-cooperative anycast nodes and resolve ambiguities (Section III-A). This approach requires specific measurement support, sometimes limiting its coverage. Our second method (Section III-B) proposes the use of Internet-class (IN) TXT DNS records to enable the use of tens of thousands of recursive DNS servers as vantage points.

A careful validation of these methods, using 60k and 300k vantage points, reveals interesting trade-offs (Section IV). CHAOS queries issued from 60k Netalyzr clients discover (measured using *recall* from information retrieval) 93% of F-root anycast servers. However, because the CHAOS query format is not standardized, different providers use different conventions to identify anycast servers; this results in a measurement ambiguity that can be resolved using traceroute probes. A smaller scale experiment on PlanetLab using 238 nodes reveals that *precision* of CHAOS queries can be improved from 65% to 89% using traceroute, and to 100% if the provider's CHAOS labeling conventions are known. Finally, we show that, up to 90% recall is possible on-demand, when we shift to IN queries and 300k recursive DNS servers, as evaluated on the AS112 anycast service.

More important, we find that 10,000 or more vantage points are required to reach a recall of 80% for either method (Section IV-B). For context, almost all prior work on anycast performance (the exception being [8]) has used only hundreds of vantage points. Interesting future work may be to examine whether their conclusions would be significantly altered by a broader perspective as suggested by our approaches.

Our second contribution is to understand how anycast is used in practice over many services (Section V). Until recently, the AS112 anycast service used manual methods to track their extent of deployment; our evaluations find that the manual list is out-of-date and incomplete, with about 26% of listed nodes

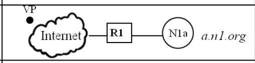
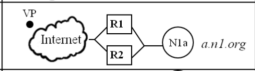
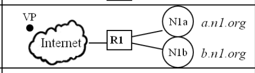
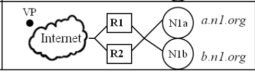
| | | Node topology | *Label* | PR |
|---|---|---|---|---|
| T1 | One router<br>One server |  | *1* | 1 |
| T2 | Multi-router<br>One server |  | *1* | >1 |
| T3 | One router<br>Multi-server |  | *>1* | 1 |
| T4 | Multi-router<br>Multi-server |  | *>1* | >1 |

**Fig. 1:** *Anycast node configurations. Observed labels in italics and penultimate-hop traceroute routers in bold. "VP" indicates vantage points, "R1, R2" indicates penultimate routers, "N1a, N1b" indicates servers in the same node.*

no longer operational. Recently, AS112 operators have adopted a discovery method similar to what we propose. Second, we evaluate anomalous anycast usage (Section V-A). We found one third-party DNS server *masquerading* as an anycast node for a public root server, and hundreds of users observe what are likely in-path proxies. This demonstrates the importance of dynamic discovery methods to audit critical Internet infrastructure. Finally, in Section V-C, we apply anycast discovery to servers for all top-level domains, showing that up to 72% of TLDs may now be using anycast. Thus, our methods can lead to new insights about anycast usage and, in the future, enable an understanding of how this usage evolves over time.

Data we generated for this paper is no cost [17].

## II. A Taxonomy of Anycast Configurations

IP anycast provides clients of a distributed service with a single-server abstraction [33]. Clients send traffic to a *specific IP address* identifying the *anycast service*. However, the service itself is implemented by a service provider using one or more *anycast nodes* that can be physically distributed around the Internet. Standard routing protocols such as BGP ensure that the user's packets are sent to a nearby anycast node. Each anycast node covers a *catchment* of nearby users, and, the whole Internet is divided into as many catchments as anycast nodes. Catchments may change when routing changes, so a user may switch between two or more catchments at any time. Successive packets from that client can therefore be routed to different anycast nodes, so anycast is usually used only for stateless, single-packet-exchange services like DNS [19] or datagram relay [21].

**Routing configurations:** We study both global and local anycast nodes; *global* nodes can be seen across multiple ASes, while *local* nodes advertise anycast prefix with the *no-export* BGP attribute, thus are visible only within the hosting or adjacent ASes. Anycast is used in both IPv4 and v6; this paper considers anycast in IPv4, although we believe the approaches also apply to anycast in IPv6.

**Node configurations:** While routing allows clients to access a nearby anycast node, a node itself can be structurally complex; each node may be implemented by one or more *anycast servers*. Figure 1 lists configurations that influence our design.

The top row (T1) shows the simplest case, where a single server provides service at a given anycast node. This server listens to traffic on the service anycast address.

Since anycast nodes are often placed in IXPs with complex local topologies, row T2 of Figure 1 shows a single server with links to multiple adjacent routers, either connected by a shared LAN or with multiple network interfaces.

For large services such as a top-level domain server, a service at an anycast node may be provided by multiple physical servers. Cases T3 and T4 show multiple servers behind one (T3) or two or more (T4) routers.

## III. Methods for Anycast Discovery

We wish to allow a third-party to *fully enumerate* all anycast nodes of a service. Our method sends special DNS queries from *vantage points* in many anycast catchments to elicit unique information from each anycast node. In addition, since one anycast node may have multiple anycast servers (cases T3 and T4 of Figure 1). We must therefore associate anycast servers with nodes to avoid node overcounts (Section IV-C).

We describe our two active probing methods below. First, we extend the existing *CHAOS queries* by adding traceroutes. Second, we develop a new proposal for a standardized type of *IN query*. These methods differ in what information they return and what vantage points they can use.

### *A. CHAOS Queries*

Anycast providers require methods to observe and debug their services. Their current methods use DNS records to identify individual anycast nodes and servers as documented in RFC-4892 [41]. Although not mandatory, we find these conventions used widely (Section V-C).

Since anycast is often used for DNS services, and DNS provides a convenient external query mechanism, RFC-4892 uses distinct DNS records to identify specific anycast servers. It re-purposes CHAOS-class network records from the now defunct Chaosnet to provide anycast diagnosis. Standard DNS records [30] with class CHAOS, type TXT, and name *hostname.bind* or *id.server* are defined to return a unique string per anycast server. The contents of the string are provider-defined and not formally standardized, although we have identified common conventions (see [15]).

In principle, presence of these records should make identifying anycast servers trivial. Standard DNS tools (such as dig or nslookup) can retrieve this information. Because CHAOS records are tied to individual servers, they correctly identify single-server nodes (cases T1 and T2 in Figure 1) and can also detect each server in multi-server nodes (cases T3 and T4).

In practice, CHAOS records are not always sufficient to identify anycast servers. They are specified in an informational RFC, and not in a mandated standard, so providers may choose to not to provide them. In addition, CHAOS records indicate anycast servers, but conventions to relate anycast servers to nodes are unspecified. Thus, the multi-server cases T3 and T4 in Figure 1, require additional information to determine the two servers located at the same anycast node.

These shortcomings motivate our design of a qualitatively different method based on IN queries (Section III-B). However, it is possible to overcome some of these limitations by augmenting CHAOS queries with traceroute information.

**Using Traceroute for Disambiguation:** Consider a traceroute from a vantage point to its nearest anycast node. We simplify the path and focus on the *penultimate router*, or *PR*. Ideally, each anycast node will have one PR, as exemplified by case type T1 in Figure 1. In practice, this is only partially correct, since anycast nodes with a rich local topology sometimes have multiple PRs (case T2 of Figure 1) or multiple servers per node (cases T3 and T4). These cases complicate our analysis as we will find multiple *PR* even in a single anycast node. If we count each *PR* as an anycast node, that leads to overcount. We will discuss overcount introduced by traceroute together with CHAOS query in section IV-C.

These ambiguities suggest that we cannot just use traceroutes, as they would cause significant overcounts. However, we find a combination of CHAOS query and traceroute helps disambiguate the cases of Figure 1; we evaluate our combined method in Section IV-C.

From Figure 1, we see that traceroute complements CHAOS queries. Sometimes both methods work (case T1), or one of the two works (cases T2, T3). In case T4, both methods fail with an overcount of the anycast node, and when no vantage point is in the node's catchment, they undercount. When possible we use them together: if either method results in a single identifier, we know there is a single anycast node, even if the other suggests multiple nodes. We take observations from all vantage points as input, separately merge records with duplicate CHAOS and PR identifiers, and finally merge these lists to get a combined estimate.

**Vantage Points:** As a result of the specific naming convention used for anycast identification (*hostname.bind*), these records cannot be retrieved using recursive DNS queries. As such, use of this method requires customized software in each catchment. One option is to use public research infrastructure like PlanetLab. In our experiments we generally use 238 PlanetLab nodes, about one per unique site. However, as a research platform, PlanetLab servers do not provide the geographic and topological diversity we need to cover all catchment areas. Even today, with "only" around 500 sites, PlanetLab cannot cover all ASes.

To overcome this limitation, we have also crowd-sourced anycast discovery. We requested the Netalyzr [24] developers to add our methods to their service. They implemented CHAOS queries, but omitted traceroute due to constraints of Java. In Section IV, we examine Netalyzr data obtained from about 62k vantage points.

### B. IN Queries

While the CHAOS query is current practice, its use requires diagnostic software at a vantage point in each anycast catchment. While Netalyzr's clients provide reasonable coverage, we consider an alternative that provides more convenient anycast discovery.

Regular Internet-class (IN) DNS records support *recursive DNS queries*, allowing the use of open recursive DNS-servers to serve as vantage points easily accessible from a centralized measurement site. We therefore propose a new approach using IN TXT records for anycast enumeration.

For IN queries, we propose that each anycast service define a designated subdomain `_ns-diagnostics` delegated to the anycast server. Inside that subdomain, dedicated TXT-type resource records identify anycast servers (label `_instance-id`) and nodes (`_node-id`) anycast instances. Thus, a node of the F-root could be identified by querying `_node-id._ns-diagnostics.f.root-servers.net`. The key advantage of IN records is that, unlike CHAOS records, they can be retrieved through recursive queries. We place them as a subdomain of the service domain so they require no new protocols; we use an unusual designated subdomain so their label is unlikely to conflict with existing domains. Our mechanism therefore requires that each anycast service create a separate zone for diagnostic information, and that each server populate that zone with server-specific resource records following our convention. Placing diagnostic information in the public namespace risks mixing user and operational information, it is consistent with existing usage such as in-addr.arpa reverse resolution, and allows use of one protocol for both puposes.

We have offered this proposal for standardization [16], but it is under consideration and not yet deployed. However, the AS112 anycast service uses a similar approach; it provides a proxy to evaluate our approach in Section IV-B.

**Vantage Points:** Our IN-class records can be queried using recursive DNS servers (rDNS), so they do not require custom diagnostic software (in PlanetLab or Netalyzr) at each vantage point. Many DNS servers offer recursive service, and a few hundred thousand of these support public queries. By sending queries indirectly through rDNS, each rDNS server effectively becomes a vantage point, potentially covering many more ASes and anycast catchments. We use open rDNS servers to quantify the performance of IN query based enumeration.

## IV. Validation

We next evaluate the accuracy of CHAOS and IN queries, and illustrate the role that traceroute plays in improving the accuracy of CHAOS queries.

### A. Methodology

We are interested in the efficacy of the anycast discovery methods discussed in the previous section. We evaluate this from many global vantage points to three large anycast services for which we have ground truth.

**Vantage points:** We use three different sets of vantage points: PlanetLab (238 VPs in 40 countries and 186 ASes) Netalyzr (61,914 VPs in 164 countries and 4,153 ASes), and recursive

DNS servers (318,988 VPs in 220 countries and 15,210 ASes). Our use of Netalyzr and rDNS allows us to evaluate the impact of an order of magnitude more VPs than prior work.

**Targets:** We study three anycast services in our experiments. In most cases we study the F-root DNS service run by ISC, and the Packet Clearing House (PCH) Anycast service that provides service for 56 TLDs. To evaluate IN queries, we use AS112, an anycast service providing reverse name lookups for private address space. ISC and PCH are professionally-run services, while AS112 servers are run by volunteers.

**Ground truth:** We consider two types of ground truth: oracle truth, and authority truth. By *oracle truth*, we mean the actual set of nodes serve on an anycast address in the Internet at any instant. We identify it as "oracle" truth because defining it requires a perfect snapshot of network routing from all parts of the Internet—an impossible goal. We define *authority truth* as the list of nodes that we get from the anycast service provider.

Oracle and authority truth can diverge for two reasons. First, third parties may operate anycast nodes for a given service with or without the provider's knowledge. Such third party nodes would not be part of authority truth. We discuss an example of a third-party node for F-root in Section V-A. Second, we derive authority truth from public web pages, which can sometimes lag the current operational system, as discussed in Section IV-C.

**Metrics:** Our active probing methods can result in several categories of results, with respect to authority and oracle truth. This influences our choice of metrics.

When our probes find an anycast node in authority truth, we have a *true positive* or *tp*. If we have no vantage points in a node's catchment, it's a *false negative* (an undercount, *fn*).

There are three cases that might be classified as false positives. If we are unable tell that two anycast servers belong to a same anycast node, then we would *overcount*. In an overcount, neither observation is completely wrong but they result in a mis-estimate of the number of anycast *nodes*. When we detect a node that we confirm is operated by the anycast provider but is not in authority truth, we have a *missing authority* ("missauth" for short). Finally, if a non-authorized anycast node appeared, we record an *extra* node. An extra node is a false positive when compared to authority truth, but it is a true positive when compared to oracle truth.

We define *precision* against authority and oracle truth:

$$precision_{authority} = \frac{tp}{tp{+}overcount} \tag{1}$$

$$precision_{oracle} = \frac{tp{+}missauth{+}extra}{tp{+}missauth{+}extra{+}overcount} \tag{2}$$

In general, we do not have false positives (because everything we find is an anycast server). Therefore authority precision reflects our level of accidental overcounts due to multi-server or multiple PR nodes.

*Recall* captures the coverage of authority truth:

$$recall = \frac{tp}{tp + fn} \tag{3}$$

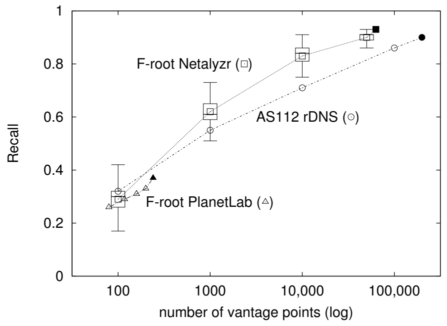


**Fig. 2:** *Recall as number of vantage points vary. F-root/PlanetLab uses CHAOS+traceroute (best case: 37%); F-root/Netalyzr uses CHAOS-only (best: 93%); AS112/rDNS uses IN-query only (best: 90%). Lines show means of subsamples, and rightmost point is our complete observation. Boxes on F-root/Netalyzr show median and quartiles, with the whiskers showing extrema.*

We do not define a recall for oracle truth because we do not have a complete set of the oracle population.

## B. Recall

Ultimately our recall is dominated by our ability to see different anycast nodes. At best, each vantage point is in a different catchment and sees a new node; at worst, they are all in the same catchment and we are uncertain if the target is using anycast at all. We next explore how query method and numbers of vantage points affect recall.

**Recall for CHAOS Queries:** We first consider recall for CHAOS queries. Figure 2 shows recall as a function of number of vantage points for F-root from PlanetLab and Netalyzr. For each line, the right-most point represents the complete observation. We also plot recall from smaller subsets of vantage points by taking 1000 random samples of them to estimate the effect of numbers of vantage points on recall. For Netalyzr, we show quartiles and extrema with box plots; other cases have similar distributions, omitted for clarity.

First, we see that with 62k vantage points, Netalyzr finds nearly all F-root anycast nodes at 93% recall (53 of the 57 official F-root nodes). By contrast, the 238 vantage points in PlanetLab provide a recall of only 37%.

We also see a roughly logarithmic relationship between recall and the number of vantage points: recall grows very slowly with increasing numbers of vantage points. On average, about 10,000 vantage points are required to achieve 80% recall; we note that, with the exception of [8] which used 20k rDNS servers, all prior anycast measurement studies have used far fewer vantage points.

**Recall for IN Queries:** Our proposal for standardizing IN queries for anycast identification is not yet widely deployed. Fortunately AS112's anycast service is ideal to test our IN queries approach because its providers have adopted the convention that each anycast node include a unique `hostname.as112.net` IN TXT DNS record; these records can serve as a proxy for our IN query based approach.

We query AS112 using over 300,000 rDNS servers, and find 65 servers; in contrast, issuing IN queries for AS112 from PlanetLab reveals only 14 of these servers. These statistics suggest the scale of vantage points required in order to enumerate anycast servers; almost 3 orders of magnitude more vantage points are required to quadruple the number of observed servers. This requirement is also evident in Figure 2, where 300k rDNS servers achieved a recall of 90%. Furthermore, our analysis of the recall achieved by subsets of rDNS servers reveals that almost 100k rDNS servers are required to achieve a recall of 80%. Intriguingly, rDNS exhibits lower recall than using Netalyzr clients, since the line for AS112 is consistently lower than the line for F-Root. We have left to future work an understanding of whether this difference results from differences in the two anycast services, or arises from the type or placement of vantage points.

To compute recall, we need to calculate the authority truth for AS112, a difficult task. The AS112 project maintains a voluntary list of known providers [4]. However, AS112 is run by volunteers, using public information to set up new nodes [2] with only loose coordination, so this list is both incomplete and out-of-date (as confirmed by AS112 coordinators). Each entry of the list identifies a provider by name and AS number. Some entries include one or more unicast IP addresses for an anycast node's DNS server.

Table I compares anycast nodes found by our IN queries approach to this list. We find that rDNS discovered 35 nodes that were *not* in the AS112 list, confirming that the voluntary list is incomplete and that *automatic diagnosis is important*. Moreover, rDNS discovers 42% (30) of the provider's list.

To build a more accurate "ground truth", we evaluate which entries on the list are actually alive or we can confirm are no longer operational. Besides rDNS probes, we confirm nodes are alive or down in two additional ways. First, when the list provides a unicast IP address for the node, we can confirm its presence with direct unicast DNS queries. Second, we use BGP only to confirm live nodes. BGP cannot to confirm down nodes because BGP also has limited recall. We probe the AS112 anycast prefix from 40 open BGP looking glasses and Merit's BgpTables service [29] which provides 38 BGP peers, and search for the provider's AS number in any AS paths.

Using these methods, we confirm that 18 nodes in the list (26%) are no longer operational, and there are 15 nodes (hard to judge) that we cannot decide if they are alive or not. These overstatements in the manual list again show the benefits of automatic diagnosis to improve accuracy.

From this analysis, there are two ways to define ground truth: (a) the nodes on the list that are alive and hard to judge (37+15), plus those found by rDNS but not on the list (30), or (b) the nodes on the list known to be definitely alive (37), plus those found by rDNS but not on the list (30). Recall defined by (a) is 75%, while that defined by (b) is 90%. We argue that (a) is a conservative choice, and that the true recall is likely to be closer to 90%, since we were able to determine that 18 of the nodes are no longer operational.

| CHAOS queries: | F-Root | | PCH |
|---|---|---|---|
| authority truth | 57 | | 53 |
| oracle truth | 58 | | 53 |
| estimated anycast nodes ($|\hat{A}|$) | 34 | | 26 |
| true positives | 21 | | 26 |
| overcounts | 12 | (0) | 0 |
| missing authority | 1 | | 0 |
| extra | 0 | | 0 |
| authority precision | 64% | (100%) | 100% |
| oracle precision | 65% | (100%) | 100% |

**TABLE II:** *Accuracy of CHAOS queries without traceroute.*

The AS112 community has recently recognized the need for automated methods of node discovery, and, prompted by our findings, have implemented an automated discovery method that also uses rDNS obtaining similar corrections to their public ground truth [5].

**Role of Vantage Point Diversity:** While we focus on the number of vantage points and their effect on recall, the role of additional VPs is their presence in multiple anycast cachments—the diversity of their network locations. There is clear redundancy in coverage, since there exist fewer than 100 anycast nodes in each of our systems, while we probe from thousands of VPs. While seemingly wasteful, such redundancy is essential to detect masquerading anycast nodes and to understand the scope of each cachement.

## C. Precision for CHAOS Queries

While determining the ground truth (and therefore recall) for AS112 was challenging, CHAOS queries face a different challenge: since CHAOS queries are not standardized, providers adopt different conventions for labeling servers and nodes, and this can affect precision. To evaluate precision, we use our PlanetLab experiments on F-Root and PCH; this is the only set of vantage points from which we were able to issue both CHAOS queries and traceroutes, and precision can be affected by the choice of whether to use traceroutes or not.

Table II describes the precision of using *CHAOS queries alone* (without traceroute). PCH precision is 100%. F-root precision falls to 64%, mostly because of 12 overcounts. These overcounts are due to T3 or T4 configurations where multiple servers provide service for a single node. Since ISC's CHAOS records are per-server (not per-node), multi-server configurations result in overcounts.

CHAOS records also reveal one case of incomplete authority truth for F-root. Although missing from the public web page, ISC confirmed that the one anycast node we found should have been listed. This missing authority makes our oracle precision slightly better than authority precision, from 64% to 65%.

Our CHAOS algorithm does not interpret the contents of the reply, because there is no formal standard. However, each anycast service provider has its own convention. As an experiment, we decoded ISC's convention, to extract identities of both the anycast node and the specific server. We show the results of this F-Root-aware CHAOS algorithm in parenthesis in Tables II and III. This provider-specific interpretation

| | | | | **authority** | **rDNS** |
|---|---|---|---|---|---|
| Found by rDNS, but not in ground truth | 35 | | | missing | **new** |
| Operator list (authority truth) | 70 | 100% | | both known | |
| node alive | 37 | 53% | | | |
| found by BGP information (and not rDNS) | 7 | | | known | *missing* |
| found by rDNS | 30 | 42% | | both known | |
| found by PlanetLab | 14 | 20% | | both known | |
| node down | 18 | 26% | | out-of-date | **corrected** |
| hard to judge | 15 | 21% | | interpretation uncertain | |
| Conservative ground truth (37 + 15 + 35) | 87 | | 100% | | |
| found by rDNS (30 + 35) | 65 | (Conservative recall) | 75% | | |
| Realistic ground truth (37 + 35) | 72 | | 100% | | |
| found by rDNS (30 + 35) | 65 | (Realistic recall) | 90% | | |

**TABLE I:** *Evaluation of IN queries coverage compared to the AS112 providers list as ground truth.*

| **Combined Method:** | **F-root** | | **PCH** |
|---|---|---|---|
| authority truth | 57 | | 53 |
| oracle truth | 58 | | 53 |
| estimated anycast nodes ($\|\hat{A}\|$) | 27 | | 26 |
| true positives | 21 | | 26 |
| overcounts | 3 | (0) | 0 |
| missing authority | 2 | | 0 |
| extra | 1 | | 0 |
| authority precision | 88% | (100%) | 100% |
| oracle precision | 89% | (100%) | 100% |

**TABLE III:** *Accuracy of CHAOS queries augmented with traceroute.*

makes our method completely correct, suggesting it would be beneficial to standardize reply contents.

Since we are unable to confirm provider-specific conventions, we next consider use of traceroute to improve precision. Table III measures how much our results improve by augmenting CHAOS queries with traceroute. Combining the two sources allows true positives to follow the larger of the two stand-alone methods for both targets. It reduces overcounts by 75% (3 instead of 12 or 13) for F-root, even without decoding F-root CHAOS replies, and eliminates overcounts for PCH.

These improvements translate into better precision for the combined method. For F-Root, precision rises to 88% (compared to 64% or 58% authority precision, with similar results for oracle precision), and PCH precision remains at 100%, the maximum of the single-source algorithms. Thus, because CHAOS conventions are not standardized, augmenting CHAOS queries with traceroute can improve precision significantly (from 65% to 89% for F-Root).

## V. Evaluation

Methods for identifying anycast server can help uncover anomalies in anycast configuration, characterize the level of deployment of anycast among root name servers and TLDs, and help us understand how anycast is managed as a service by providers for use by DNS root and TLD operators. This section demonstrates these uses of our approach.

### A. Anomalous Anycast Configurations

**Root Masquerading:** While validating CHAOS queries on F-Root, we encountered an anycast server that was not on ISC's list of F-Root anycast nodes, and which returned an empty CHAOS response. Discussions with ISC confirmed this site was *masquerading* as an F-Root anycast node—a non ISC server operating on the F-root IP address.

ISC described two general cases where third parties operate nodes at the F-Root anycast address. First, some organizations operate local copies of the root zone, and internally *masquerade* responses to `*.root-servers.org`. While ISC discourages this behavior, redirection of internal traffic is generally left up to the organization. Second, other organizations have attempted to hijack root DNS server from others, often to promote a modified root zone.

We observed this masquerading host from two vantage points inside CERNET, the China Education and Research Network. In both case the PR of the target is 202.112.36.246, at AS4538 in CERNET. The SOA record of the two zones are same, although we did not exhaustively compare the zones. ISC identified this non-ISC anycast node as a masquerading node, not hijacking, and we concur.

While this case represents a network provider choosing to handle requests from their own users using masquerading, nearly the same mechanisms can be used to detect hijacking. This potential illustrates the benefits of actively monitoring anycast services, at least until DNSSEC becomes pervasive.

**In-Path Proxies and Others:** Beyond masquerading, our methods can identify other abnormal configurations. We detected these anomalies when analyzing Netalyzr dataset.

While Netalyzr does not augment CHAOS queries with traceroute, it does include CHAOS queries to each root server, IP resolution requests for www.facebook.com and for a non-existent domain name RANDOM.com (where `RANDOM` is string longer than 40 characters, that triggers a non-existent domain error message). It also reports when the CHAOS queries timeout without response; we ignore these cases. We next use this information, to infer possible root causes of these abnormal responses for F-root.

In this dataset, we see two abnormal responses to CHAOS queries: incorrect CHAOS records and missing CHAOS records, making up about 1.8% of the observations (Table IV). We believe these observations detect *in-path proxies*. Usually end-systems are configured to use a local DNS resolver. An in-path proxy is a network middlebox that captures and redirects

| | | | | | |
|---|---|---|---|---|---|
| Total observations | 61,914 | 100% | | | |
| expected replies | 59,509 | 96.1% | | | |
| no reply | 1,289 | 2.1% | | | firewall discards or routing failure |
| abnormal replies | 1,116 | 1.8% | | | |
| observations have fake F-root CHAOS record | 355 | 0.6% | [100%] | | in-path proxies |
| Got facebook or non-existent-domain | 354 | | [99.7%] | | in-path proxies |
| neither facebook nor non-existent-domain | 1 | | [0.3%] | | in-path proxy or hijack/masquerade |
| observations got empty F-root CHAOS | 761 | 1.2% | | (100%) | |
| Got facebook or non-existent-domain | 550 | 0.8% | | (72%) | in-path proxies |
| no facebook and non-existent-domain | 211 | | | | |
| got empty CHAOS records for all roots | 93 | 0.15% | | | firewall or hijack/masquerade |
| got valid CHAOS records for some other roots | 117 | 0.2% | | | hijack/masquerade |
| got fake CHAOS records for some other roots | 1 | | | | in-path proxy |

**TABLE IV:** *Anomalies found for F-root CHAOS records in Netalyzr data.*

all DNS traffic directly. We believe that incorrect F-root CHAOS records (355 cases, 0.6%) indicate in-path proxies that modify CHAOS queries, since we know all F-root nodes provide correct CHAOS records. We believe these are in-path proxies because almost always (354 of the 355 cases) the client also gets a direct reply for Facebook from the supposed-F-root node. A true F-root server would not have directly responded with an entry for Facebook, but would have redirected to the .com DNS server. (The one case that omits Facebook is located in China, where Facebook is blocked.)

Empty CHAOS records occur more often (761 cases, 1.2%). In most of these cases (550, 72% of 761, 0.8% of all), we observe Facebook or non-existent domain replies, suggesting an in-path proxy for the same reasons as above. However, in some of these cases, we see an empty CHAOS record for F-Root, but also get neither a Facebook nor the non-existent domain reply. In some cases we get empty CHAOS records for all roots, in others we see some valid and other invalid CHAOS records. Without additional data (like traceroutes) we cannot diagnose these problems with certainty. We believe they are either firewalls or masqueraders.

To summarize, in about 62k unique IP addresses, our data suggests that 0.2% appear to be behind potentially masquerading F-root nodes, while 1.4% (0.6% + 0.8%) see in-path proxies, and about 0.15% see other unusual behavior. These observations suggest that DNS manipulation is not common, but does occur. They also suggest the need for external monitoring as our IN-queries, and for additional information to disambiguate these cases, as with our use of traceroute.

## B. Anycast Use in Other DNS Roots

We next consider anycast use in root TLD servers using Netalyzr, In Table V we compare the number of measured anycast nodes, from CHAOS queries with 62k Netalyzr vantage points, against the published number from root-servers.org. We expect 10 of the 13 to use anycast. In 3 of the 10 cases (F, H, and I) we detect anycast nodes not reported, one case (E) has now deployed anycast, suggesting public information is out-of-date, omitting up to 14 nodes. In 4 cases (A, J, K and L), we miss some nodes, either because recall with Netalyzr is not perfect, or because the published list is out-of-date. Finally, we fully enumerate three smaller cases (C, G, and M).

| DNS root servers | measured | | published | found |
|---|---|---|---|---|
| A (Verisign) | 2 | $<$ | **6** | 33% |
| B (ISI) | 1 | $=$ | 1 | 100% |
| C (Cogent) | 6 | $=$ | 6 | 100% |
| D (Univ. of Maryland) | 1 | $=$ | 1 | 100% |
| E (NASA) | 9 | $>$ | 1 | 900% |
| F (ISC) | **53** | $>$ | 49 | 108% |
| G (DISA) | 6 | $=$ | 6 | 100% |
| H (U.S. ARL) | **3** | $>$ | 2 | 150% |
| I (Automica) | **39** | $>$ | 38 | 103% |
| J (Verisign) | 59 | $<$ | **70** | 84% |
| K (RIPE) | 17 | $<$ | **18** | 94% |
| L (ICANN) | 78 | $<$ | **107** | 73% |
| M (WIDE) | 6 | $=$ | 6 | 100% |

**TABLE V:** *Comparing measured against published numbers of anycast nodes for all anycast root servers.*

## C. Anycast Use in Top-Level Domains

Anecdotal evidence suggests that anycast is widely used in DNS, but to our knowledge there has been no systematic study of *how extensively* it is used. In this section, we determine how many TLDs use anycast by using CHAOS queries (with traceroute) on PlanetLab. Although PlanetLab's recall is low, that should not affect the results discussed in the section since we are not trying to enumerate all of the anycast servers in each TLD. Rather, we try to determine if more than one server responds to a CHAOS query sent to a TLD nameserver.

**Target:** The targets for our study are the authoritative name servers for the country-code top-level domain names (ccTLDs), and the generic TLDs (gTLDs), as listed by IANA [22]. Together there were 1133 IP addresses providing TLD nameservice in April 2012.

**Methodology:** We use CHAOS queries and traceroute against each IP address for each name server, querying from 240 PlanetLab vantage points. (We omit IN queries because, until further standardization, only AS112 supports them.) We collected data on May 2011 and April 2012, and present the data collected on 2 April 2012. (We see similar results on other days in 2012, and fewer in 2011.)

In Table VI we interpret these results to identify definite and possible anycast services, since in this case there is no ground truth. Of these cases $CHAOS > 1 \wedge PR > 1$ is the strongest evidence for anycast, though our combined method still finds a few T4 unicast cases. The other cases where $CHAOS > 1$

| CHAOS (# recs.) | traceroute (number of PRs) > 1 | 1 | 0 |
|---|---|---|---|
| $> 1$ | **anycast**; T4 unicast | **anycast**; T3 unicast | **anycast**; T3 unicast |
| 1 | T2 unicast; **mis-config anycast** | unicast | unicast; mis-config anycast |
| 0 | **non-BIND anycast**; T2/T4 unicast | unicast | insufficient information |

**TABLE VI:** *Interpretation of CHAOS queries and traceroute on TLD nameservers.*

| 2012 April Results | | | | |
|---|---|---|---|---|
| CHAOS (# recs.) | traceroute (# PRs) $> 1$ | 1 | 0 | (definite, possible) anycast |
| $> 1$ | 255 (238, 0) | 14 (3, 0) | 159 (0, 159) | **(241, 159)** |
| 1 | 99 (2, 1) | 117 (0, 2) | 312 (0, 0) | **(2, 3)** |
| 0 | 44 (0, 44) | 32 (0, 0) | 101 (0, 0) | **(0, 44)** |
| | | | total TLD name servers and anycast: | 1133 **(243, 206)** |

**TABLE VII:** *Anycast discovered for TLD name servers. The first number in braces is definite anycast, the second number in braces is possible anycast.*

or *PR* $> 1$ are likely partially observed anycast addresses. We classify these results in two ways. *Definite anycast* means our method finds multiple nodes. *Possible anycast* means there are multiple records but we cannot guarantee anycast, such as when $CHAOS == 0 \wedge PR > 1$ or $CHAOS > 1 \wedge PR == 0$.

**Results:** Table VII shows our results of anycast deployment in TLD name servers. We report definite anycast as the first number in braces, and the second number is possible anycast. We observe that about 21% (243 of 1133) of TLDs nameservers use anycast, while another 18% (206 of 1133) are possibly anycasted. If we adopt definite anycast as a lower bound and definite plus possible as an upper bound, then at least 21% and perhaps 39% of TLDs nameservers use anycast.

A complementary view classifies use of anycast by the *name* of the TLD, rather than by IP address. As there are always several authoritative name servers for a top level domain name, we count a TLD name as definitely anycasted if at least one of its authoritative name servers is definitely using anycast. Table VIII shows anycast deployment in TLD names. When there are no definite anycasted name servers, but at least one is possible anycast, then we count the TLD name as a possible anycast. We see that at least 56% of the TLD names are definitely anycast, and 72% of TLD names possibly so. Thus, more than half and perhaps three-quarters of TLDs include anycast as part of their DNS service.

| Number of TLD names | definite anycast | possible anycast | higher bound |
|---|---|---|---|
| 314 (100%) | **177 (56%)** | 48 | **225 (72%)** |

**TABLE VIII:** *Anycast services discovered for TLD names*

The main implication of these findings is that anycast is an important component of the DNS, and needs to be continuously monitored for abnormal configurations, masquerading or, worse, hijacking (Section V-A).

## VI. Related Work

We briefly discuss related work; a more comprehensive discussion may be found in [15].

The DNS operational community has developed several techniques to support anycast diagnostics. RFC-4892 [41] documented CHAOS query; RFC-5001 [6] defines a NSID (name server identifier) option for DNS. However, these techniques do not support recursive DNS queries, nor do they standardize replies, and RFC-5001 is not yet widely supported.

Recent work [27] proposes unique-per-node AS numbers for anycast node identification. If the method is widely deployed, it can be used for anycast enumeration, and our analysis of recall will apply there.

The RIPE Atlas measurement platform has been used to evaluate anycast nodes from about 2k VPs [31]. This platform can potentially be used to host our traceroute-based enumeration method as well.

Complementary to our work is a rather large body of work on measuring the proximity (client-to-server latency), affinity (the stability of client-server association), and load balancing for DNS anycast. Methods to study proximity include comparing anycast query latency with unicast latency to anycast servers from several vantage points [8], [11], [36], measuring server-side accesses by clients, and geolocating clients to estimate latency [26]. Several researchers have explored affinity by periodically probing anycast servers to determine when routing changes cause anycast packets to be routed to a different node [7], [8], [10].

Our work is inspired by these works, but differs in several respects. While other work has explored the use of CHAOS records to study affinity [7], [9], [10], [37], we extensively validate CHAOS query use for anycast server enumeration and use it to characterize the use of anycast in TLDs. Most prior work listed above have used hundreds of vantage points, usually from PlanetLab; as our work shows, anycast recall is modest at the scale of PlanetLab implying that the conclusions drawn by prior work may need to be revisited. One exception is the work of Ballani et al. who have used 20,000 rDNS servers [8]; our evaluations contain an order of magnitude larger vantage points. Finally, IN-class records have been used in AS112, and Ballani et al. [8] use a similar mechanism to study anycast load balance in a controlled setting. Our primary contribution is a concrete proposal to standardize anycast identification using IN queries, and a careful characterization of its recall properties.

Prior work has explored methods to detect and/or prevent unicast route hijacking [20], [25], [32], [34], [40], [42], [43]. Detecting anycast hijacking is qualitatively harder than detecting unicast hijacking, since by definition, anycast packets can be sent to one of many destinations, one or more of which may be suspect while with unicast routing any examples of multiple destinations are illegitimate. To our knowledge, we are the first to monitor anycast hijacking.

While DNSSEC [3] provides origin authentication and data integrity, it does not address other risks of anycast hijacking. It prevents a hijacker from altering responses, but not from denying service or monitoring traffic [28]. Our work complements DNSSEC by providing tools to help detect and identify hijacking.

## VII. Conclusions

Through its wide use in DNS, anycast has become an indispensable part of the Internet. We developed new methods that combine CHAOS queries with traceroutes, or use new IN records to support tens of thousands of open recursive DNS servers as vantage points. We find our methods have generally good precision and high recall. In particular, we find that the topological dispersion of anycast requires a very large number of vantage points to enable high recall; on average, 10,000 vantage points are require for a recall of 80%. Finally, our studies of F-Root and PCH anycast infrastructure detect one third-party site masquerading as an anycast node, reveal several abnormal anycast configurations, and our evaluation of all country-code and generic top-level domain servers shows anycast is possibly used by 72% of the TLDs.

**Acknowledgments:** We thank P. Vixie, L. Bicknell, and S. Woolf (ISC) and B. Woodcock and R. Stapleton-Gray (PCH) for answers about their infrastructure. We thank PlanetLab, particularly hosts B. Karp and J. Andrews (UCL) and R. Teixeira (lip6). We thank Dan Massey (CSU) for comments about DNSsec. We thank N. Weaver, V. Paxson, and C. Kreibich for providing adding CHAOS support to Netalyzr and providing data. We thank D. Wessels (The Measurement Factory) for information about rDNS, and W. F. Sotomayor for help about AS112. Finally, this work was partially supported by the U.S. Dept. of Homeland Security, contracts N66001-10-C-0081 and NBCHC080035; All conclusions are those of the authors and do not necessarily reflect the views of DHS.

## References

[1] J. Abley and K. Lindqvist. Operation of anycast services. *RFC 4786*, 2006.

[2] J. Abley and W. Maton. AS112 nameserver operations. RFC 6304, Internet Request For Comments, July 2011.

[3] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS security introduction and requirements. *RFC 4033*, 2005.

[4] AS112 Project. AS112 Server Operators Listing. http://public.as112.net/node/10, 2012.

[5] AS112 Project. How many public AS112 nodes are there as of March 31, 2012? http://public.as112.net/node/30, Apr. 2012.

[6] R. Austein. DNS Name Server Identifier (NSID) Option. *RFC 5001*, 2007.

[7] H. Ballani and P. Francis. Towards a global IP anycast service. In *Proc. of ACM SIGCOMM*. ACM, Aug. 2005.

[8] H. Ballani, P. Francis, and S. Ratnasamy. A measurement-based deployment proposal for IP anycast. In *Proc. of ACM IMC*, pages 231–244, Rio de Janeiro, Brazil, Oct. 2006.

[9] P. Boothe and R. Bush. Anycast measurements used to highlight routing instabilities. In *NANOG 34*, May 2005.

[10] R. Bush. DNS Anycast Stability: Some Initial Results. CAIDA/WIDE workshop, http://www.caida.org/workshops/wide/0503/slides/050311.wide-anycast.pdf, Mar. 2005.

[11] L. Colitti. Effect of anycast on K-root. DNS-OARC Workshop, July 2005.

[12] E. Cronin, S. Jamin, C. Jin, A. R. Kurc, D. Raz, and Y. Shavitt. Constrained mirror placement on the Internet. *IEEE JSAC*, 20(7):1369–1383, Sept. 2002.

[13] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl. Globally distributed content delivery. *IEEE Internet Computing*, 6(5):50–58, Sept. 2002.

[14] R. Engel, V. Peris, D. Saha, E. Basturk, and R. Haas. Using IP anycast for load distribution and server location. In *Proc. of Global Internet*, Dec. 1998.

[15] X. Fan, J. Heidemann, and R. Govindan. Identifying and characterizing anycast in the domain name system. Technical report, June 2011.

[16] X. Fan, J. Heidemann, and R. Govindan. Improving diagnostics of name server anycast instances (draft-anycast-diagnostics-01.txt). discussed on dnsop@ietf.org mailing list and at http://www.isi.edu/~xunfan/research/draft-anycast-diagnostics.txt, Oct. 2011.

[17] X. Fan, J. Heidemann, and R. Govindan. LANDER anycast datasets. http://www.isi.edu/ant/traces/anycast, 2012.

[18] B. Greene and D. McPherson. ISP security: Deploying and Using Sinkholes. NANOG talk, http://www.nanog.org/mtg-0306/sink.html, June 2003.

[19] T. Hardie. Distributing authoritative name servers via shared unicast addresses. *RFC 3258*, 2002.

[20] X. Hu and Z. Mao. Accurate real-time identification of IP prefix hijacking. In *Proc. of IEEE Symposium on Security and Privacy*, 2007.

[21] C. Huitema. An anycast prefix for 6to4 relay routers. *RFC 3068*, 2001.

[22] Internet Assigned Numbers Authority. Root zone database. http://www.iana.org/domains/root/db/.

[23] D. Katabi and J. Wroclawski. A framework for scalable global IP-anycast (GIA). In *Proc. of ACM SIGCOMM*, pages 3–15, Aug. 2000.

[24] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson. Netalyzr: Illuminating the edge network. In *Proc. of ACM IMC*, pages 246–259, 2010.

[25] M. Lad, D. Massey, D. Pei, Y. Wu, B. Zhang, and L. Zhang. Phas: A prefix hijack alert system. In *Proc. of USENIX Security Symp.*, 2006.

[26] Z. Liu, B. Huffaker, M. Fomenkov, N. Brownlee, and k. Claffy. Two days in the life of the dns anycast root servers. In *Passive and Active Network Measurement*, pages 125–134, 2007.

[27] D. McPherson, R. Donnelly, and F. Scalzo. Unique origin autonomous system number (ASNs) per node for globally anycasted services. *RFC 6382*, Oct. 2011.

[28] D. McPherson, D. Oran, D. Thaler, and E. Osterweil. Architectural considerations of IP anycast. *Active Internet-Draft*, Nov. 2012.

[29] Merit Network, Inc. bgptables. http://bgptables.merit.edu/.

[30] P. Mockapetris. Domain names—concepts and facilities. RFC 1034, Internet Request For Comments, Nov. 1987.

[31] R. NCC. RIPE atlas. https://atlas.ripe.net/, 2010.

[32] O. Nordström and C. Dovrolis. Beware of BGP attacks. *ACM SIGCOMM Computer Communication Review*, 34(2):1–8, 2004.

[33] C. Partridge, T. Mendez, and W. Milliken. Host anycasting service. *RFC 1546*, 1993.

[34] T. Qiu, L. Ji, D. Pei, J. Wang, J. Xu, and H. Ballani. Locating prefix hijackers using lock. In *Proc. of USENIX Security Symp.*, 2009.

[35] ROOT-DNS. http://www.root-servers.org/.

[36] S. Sarat, V. Pappas, and A. Terzis. On the use of anycast in DNS. Technical report, Johns Hopkins University, 2004.

[37] Y. Sekiya. Passive and active DNS measurement: update, Mar. 2005.

[38] K. Sollins and L. Masinter. Functional requirements for uniform resource names. RFC 1737, Dec. 1994.

[39] The Domain Name System Operations Analysis and Research Center. The AS112 Project. http://www.as112.net.

[40] L. Wang, X. Zhao, D. Pei, R. Bush, D. Massey, and L. Zhang. Protecting BGP routes to top-level DNS servers. *IEEE Transaction on Parallel and Distributed Systems*, 14(9):851–860, Sept. 2003.

[41] S. Woolf and D. Conrad. Requirements for a mechanism identifying a name server instance. *RFC 4892*, 2007.

[42] Z. Zhang, Y. Zhang, Y. C. Hu, Z. M. Mao, and R. Bush. iSPY: Detecting IP Prefix Hijacking on My Own. In *Proc. of ACM SIGCOMM*, pages 327–338, Aug. 2008.

[43] C. Zheng, L. Ji, D. Pei, J. Wang, and P. Francis. A light-weight distributed scheme for detecting IP prefix hijacks in real-time. In *Proc. of ACM SIGCOMM*, Aug. 2007.