On the Use of Anycast in DNS

Sandeep Sarat
sarat@cs.jhu.edu
Johns Hopkins University

Vasileios Pappas
vpappas@cs.ucla.edu
UCLA

Andreas Terzis
terzis@cs.jhu.edu
Johns Hopkins University

Abstract

This paper attempts to fill a gap in our understanding of the Internet infrastructure: while anycast is deployed in many DNS zones, no studies exist that measure the performance impact of anycast in DNS. In this paper, we present results from our study of four top-level DNS zones, to evaluate the performance implications of different anycast configurations. Increased availability is one of the supposed advantages of anycast and we found that indeed the number of outages observed was small, suggesting that anycast provides a mostly stable service. On the other hand, outages can last up to multiple minutes, mainly due to slow BGP convergence. Furthermore, depending on the anycast configuration used, 37% to 80% of the queries are directed to the closest anycast instance. Our measurements reveal that there exists an inherent trade-off between increasing the percentage of queries answered by the closest server and the stability of the DNS zone, measured by the number of query failures and server switches.

We believe that our findings will help network providers to deploy anycast more effectively in the future.

***Index Terms*— Anycast, Routing, Internet, DNS, Root Servers, Top Level Domains.**

1 Introduction

Anycast is widely deployed in DNS today [12]. The IP addresses of many top level DNS nameservers correspond to anycast groups. Client requests sent to these addresses are delivered by the Internet routing infrastructure to the closest replica in the corresponding anycast group. DNS operators have deployed anycast for a number of reasons: reduced query latency, increased reliability and availability as well as resiliency to DDoS attacks. While it is generally agreed that the deployment of anycast in DNS has been a positive step, no studies have been done to evaluate the performance improvement offered by anycast. This paper presents the first comprehensive study in this area.

Specifically, we aim to answer the following questions: (1) Does anycast reduce query latencies? (2) Does anycast selection indeed lead clients to the closest DNS server? (3) Do zones deploying anycast experience smaller number of outages? To answer these questions, we performed a measurement study using clients deployed over PlanetLab [22], to measure the performance characteristics of four top-level zones using anycast and compared it to zones not using anycast. We monitored zones that represent different points on the anycast design space to compare the effects of these design choices. Specifically, we evaluate the effects of single vs. multiple anycast addresses for a zone and global vs. localized visibility of the servers in the anycast group. We also compared these zones against a hypothetical zone with the same number of nameservers but where all the nameservers are individually addressable. By doing so, we can directly compare anycast to the traditional zone configuration guidelines [7].

Our results can be summarized as follows: We found that, in general the deployment of anycast decreases average query latency. While the number of query failures is relatively small ($\leq$ 0.9%), they are long in duration (50% last more than 100 seconds), affected by long BGP routing convergence times. The type of anycast scheme used, whether servers have local or global visibility, determines the percentage of queries directed to the closest anycast instance. This value ranges from about 37% for zones with few global nodes to about 80% for zones, wherin all nodes are global. We also uncovered an inherent trade-off between increasing the effectiveness of anycast in directing the queries to the nearest server and stability of the zone itself. For zones that advertise all its anycast group members globally, clients choose the nearest server majority of the time. The negative effect though is that, in this case the zone becomes vulnerable to increased number of server switches and outages.

The rest of the paper is structured as follows: We give a brief introduction to anycast in Section 2 and explain our measurement methodology in Section 3. Section 4 presents in detail the zones used in this study. We present our results in Section 5 and devote Section 6 to comparing the benefits and drawbacks of the different anycast techniques. Finally, we present related work in Section 7 conclude in Section 8.

2 Background

Anycast, first described in [19], provides a service, whereby a host transmits a datagram to an anycast address and the internetwork is responsible for delivering the datagram to at least one, preferably the closest, of the servers in the anycast group. The motivation behind anycast is that it simplifies service discovery. A host does not have to choose from a list of replica servers, offloading the responsibility of forwarding the request to the "best" server, onto the network.

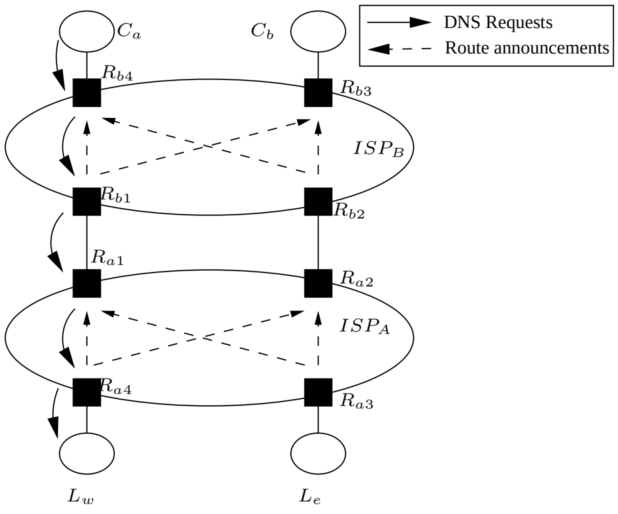


Figure 1: Sample Anycast configuration

Since the benefits of anycast are largely derived from its implementation, we briefly review how anycast is currently implemented in the Internet. In Fig. 1, the two servers L_e and L_w are members of the anycast group represented by address I. Each of these servers (or rather their first hop routers) advertise a prefix that covers I[1] using BGP [21] to R_{a4} and R_{a3} in ISP_A. Each of these routers in turn propagate the advertisements to their iBGP peers R_{a1} and R_{a2}. The process continues until the advertisements reach the egress routers R_{b4} and R_{b3} of ISP_B, where customers C_a and C_b are connected respectively. Router R_{b4} chooses the advertisement from iBGP peer R_{b1} because the IGP distance to R_{b1} is shorter than distance to R_{b2}. This selection is usually called *hot-potato* routing because it causes packets to exit the provider's network as early as possible. The final effect of these choices is that packets from C_a follow the path $C_a \rightarrow R_{b4} \rightarrow R_{b1} \rightarrow R_{a1} \rightarrow R_{a4} \rightarrow L_w$. Similarly, packets from C_b follow the right vertical path. It is evident from this description that the combination of BGP hot potato routing inside an autonomous system and shortest AS path routing across autonomous systems results in choosing the closest anycast server, closest being defined in terms of IGP metric and AS hop length.

1 The prefi x is usually a /20. This requirement emerges from the fact that advertisements for shorter prefi xes are not propagated by the routing infrastructure to reduce the size of the global routing table.

In practice, anycast deployment is accomplished using a variety of schemes. For example, a zone operator might use one or multiple nameserver addresses (NS records in DNS parlance) each with a different anycast address. Anycast prefixes can be globally advertised or their scope can be limited. Scoping is used to support servers with limited transaction and bandwidth resources and servers serving only local networks. Servers whose advertisements are scoped are called *local nodes* while nodes with no scoping are called *global nodes*. Local nodes limit the visibility of their advertisements by using the `no-export` BGP attribute. Peers receiving advertisements with this attribute should not forward the advertisement to their peers or providers. We have however found exceptions to this policy, where peers strip off the no-export attribute and advertise the local node to their peers.

In addition to the anycast address, each server in the anycast group has a unique unicast address. This address is mainly used for management purposes (e.g. zone transfers). To ensure that the management interface is reachable even in case the anycast prefix becomes unavailable (e.g. during a routing outage or DDoS attack on the anycast address) it is desirable to have a different path to the unicast address.

3 Measurement Methodology

Our goal is to investigate the implications of the use of anycast in DNS and to compare the performance benefits of different anycast configurations. To do so, we use the following four zones in our measurements, each representative of a different deployment scenario:

(1) A zone with one or more servers in a single geographic location: While this case does not use anycast, we use it as a base case to explore the potential performance improvements of anycast. We chose the B-root nameserver (192.228.79.201) as a representative of this category[2]. **(2)** A zone using a single anycast address for all its servers, with multiple servers in different locations: Since this anycast configuration is widely used, we chose two different examples, the F-root nameserver (192.5.5.241) and the K-root nameserver (193.0.14.129). Choosing two examples enables us to investigate the effects of the number and location of anycast group members on performance. **(3)** A zone using multiple anycast addresses for its servers, with multiple servers in different locations: We used UltraDNS, which is authoritative for the .org and .info top level domains, as the representative of this category. UltraDNS servers are members of two anycast groups TLD1 (204.74.112.1) and TLD2 (204.74.113.1). The perceived benefit of using multiple anycast addresses is that in the event of a network outage affecting one of the anycast addresses, clients can query the the other address for resolving names. **(4)** A zone with multiple geographically distributed servers, each individually accessible via unicast. To emulate this scenario, clients send requests to the unicast addresses of the F-root group members. Each client maintains an ordered list of servers based on their latency. This list is updated every hour. During each epoch all DNS queries are directed to the closest server according to this ranking. In case a server becomes unavailable, the client tries the servers down its list until it receives a response. We use this scenario to compare anycast to unicast when the number and location of name servers is the same.

Because DNS clients have no control of where their queries are directed, we need clients in multiple locations to cover all the servers in an anycast group. For this reason, we used the PlanetLab [22] testbed for our measurements. At the time of our measurements, there were approximately 400 nodes in PlanetLab contributed by universities and research labs around the globe. Table 1 shows the distribution of PlanetLab nodes based on their geographic location.

We ran a script on every PlanetLab node to send periodic DNS queries (the query interval is randomly selected between 25 and 35 seconds) to each of the DNS zones earlier mentioned. To ensure that the query time includes only a single round trip, we send queries that can always be answered locally by the server receiving the request. Our script records the query latency and unicast name corresponding to the anycast server answering the query. The scripts uses "special" DNS requests to retrieve the unicast name of the server replying to a request sent to the anycast address ([2] shows the request type for F-root)

We compare the selected anycast deployment schemes based on the following criteria:

[2] The reader will observe that while we are talking about zones, B-, F-,and K-root are actually *servers* of the root zone. This distinction is immaterial to our study since we use these examples *as if* they were actual zones.

Continent	**% of PL nodes**
South America	0.5
Australia	1.8
Asia	15.8
Europe	16.7
North America	65.2

Table 1: Distribution of PlanetLab nodes around the world.

Query latency Reduction of end-user delay is an oft quoted benefit of deploying anycast. To test whether this claim is true, we measure the latency of requests sent to the monitored zones and compare the results. Since anycast achieves this reduction using localization, we also calculate the percentage of DNS queries that are in fact routed to the nearest anycast instance.

Consistency Consistency measures the affinity of clients to a specific member of the anycast group. If a client switches from a server to another, we say a "flip" has occurred. We use the number of flips as a measure of consistency. A negligible number of flips implies that the particular zone offers a very consistent service. If a client flips frequently, it might encounter inconsistencies (e.g. when the servers are not synchronized). For services like the DNS, where transactions are very short (mostly limited to a single packet query and response) affinity is not critical. However, these results can be extrapolated to provide an indication whether longer transactions, such as bulk transfers over TCP, would be affected by server changes.

Availability For a global infrastructure service such as the DNS, availability is a key issue. To evaluate the impact of anycast on availability, we measure the number and duration of outage periods. An outage period is the time during which clients receive no replies to their requests. During these periods client name queries are not resolved. Since DNS requests and replies use datagrams, in case of a timeout, we resend the request twice to differentiate between dropped packets and real DNS outages. The beginning of an outage period is marked by the consecutive loss of all three requests. The end of the outage period is marked by the receipt of the first answer from a DNS server. The difference between the end and the start of an outage period gives the *length* of the outage period.

We collected data from the PlanetLab nodes from September 19, 2004 to October 8th, 2004. The results presented in this paper are based on the feedback from approximately 300 nodes.

4 Anycast Deployment Strategies

In this section, we present the configuration of the monitored zones and the distribution of requests from the PlanetLab clients to each of these zones.

4.1 Multiple Instances, One site: B-Root

The B-Root server has 3 nodes (b1/b2/b3.isi.edu), all of which are located in Los Angeles, CA. All the servers reside in the same network and therefore, this scenario is representative of a multiple instance, one site zone (case 1 in Sec. 3).

4.2 Multiple Instances, Multiple Heterogenous Sites: F,K-root

Table 2 gives a complete listing of all the F-root clusters at the time of the measurement. This list is publicly available at [14]. As can be seen from Table 2, a high percentage ($\sim 70\%$) of nodes are served by the F-root clusters PAO1 (Palo Alto) and SFO2 (San Fransisco). This is because these two clusters, which reside within ISC's home network, have been deployed as global nodes [1]. The rest of the clusters are visible locally and serve clients only within their peer communities. Out of the 26 listed F-root clusters, PlanetLab nodes contacted only 16 F-root clusters. The reason for this behavior is that the unreachable nodes have local scope and no PlanetLab nodes are located within their scope, because the AS path to the global node is shorter than the path to the local node. Therefore the PlanetLab site chooses to route requests to the global node instead of the local.

Like the F-root, the K-root hierarchy consists of global and local nodes, albeit much smaller in the group size. K-root consists of multiple clusters primarily concentrated in Europe. Table 3 gives a complete list of K-root clusters and their reachability from PlanetLab nodes. Clusters at Amsterdam and London have global visibility, while the remaining have local visibility [16]. This explains the high percentage ($\sim 97\%$) of PlanetLab nodes served by LINX (London) and AMS-IX (Amsterdam) as shown in Table 3.

4.3 Multiple Instances, Multiple Homogenous Sites: UltraDNS

Due to the unavailability of the complete listing of ultraDNS clusters, we only consider clusters that are reachable from PlanetLab nodes. Unicast names of the individual servers in the responses to specially constructed

Cluster	Indices	Location	%
PAO1	(c,d,e,f)	Palo Alto, CA, USA	38.5
SFO2	(a,b)	San Francisco, CA, USA	32.1
MUC1	(a,b)	Munich, Germany	4.9
HKG1	(a,b)	Hong Kong, China	3.5
LAX1	(a,b)	Los Angeles, CA, USA	3.4
YOW1	(a,b)	Ottawa, ON, Canada	3.1
LGA1	(a,b)	New York, NY, USA	2.8
SIN1	(a,b)	Singapore	2.0
TLV1	(a,b)	Tel Aviv, Israel	1.7
SEL1	(a,b)	Seoul, Korea	1.6
SJC1	(a,b)	San Jose, CA, USA	1.5
CDG1	(a,b)	Paris, France	1.1
YYZ1	(a,b)	Toronto, ON, Canada	1.1
GRU1	(a,b)	Sao Paulo, Brazil	1.0
SVO1	(a,b)	Moscow, Russia	0.8
ROM1	(a,b)	Rome, Italy	0.7
AKL1	(a,b)	Auckland, New Zealand	-
BNE1	(a,b)	Brisbane, Australia	-
DXB1	(a,b)	Dubai, UAE	-
JNB1	(a,b)	Johannesburg, South Africa	-
MAD1	(a)	Madrid, Spain	-
MTY1	(a,b)	Monterrey, Mexico	-
TPE1	(a,b)	Taipei, Taiwan	-
CGK1	(a,b)	Jakarta, Indonesia	-
LIS1	(a,b)	Lisboa, Portugal	-
PEK1	(a,b)	Beijing, China	-

Table 2: List of the 26 F-root sites. The last column shows the percentage of PlanetLab nodes served by each F-root cluster.

DNS queries provide a hint to the cluster name. For example, the unicast name `udns1abld.ultradns.net` suggests that the server belongs to the `abld` (London) cluster. The location of these clusters can then be extracted from the corresponding airport codes that show up in traceroute. Table 4 gives a list of all the ultraDNS clusters reachable from PlanetLab.

Unlike the F- and K-root, UltraDNS uses two anycast addresses for its nameservers, instead of one. All UltraDNS nodes respond to both anycast addresses, `tld1.ultradns.net` (TLD1) and `tld2.ultraDNS.net` (TLD2). Clusters at `abld` and `eqhk` are exceptions, which will be discussed later. Additionally, F- and K-root servers use a hierarchical setup of global and local nodes while UltraDNS uses a flat setup, where BGP advertisements from all nodes are globally visible throughout the Internet.

The distribution of DNS requests across UltraDNS clusters is more uniform compared to the F-, K-root servers. Servers in Europe (`abld`) and Asia (`eqhk`)

Cluster	Indices	Location	%
ams-ix	(k1,k2)	Amsterdam, Netherlands	51.6
linx	(k1,k2)	London, UK	46.7
denic	(k1,k2)	Frankfurt, Germany	0.9
grnet	(k1,k2)	Athens, Greece	0.7
mix	(k1,k2)	Milan, Italy	-
qtel	(k1,k2)	Doha, Qatar	-
isnic	(k1,k2)	Reykjavik, Iceland	-

Table 3: List of the 7 K-root sites.

Cluster	Location	Percentage	
		TLD1	TLD2
pxpa	Palo Alto, CA, USA	23.1	7.5
eqab	Ashburn, VA, USA	20.4	10.4
abld	London, UK	15.6	-
eqch	Chicago, IL, USA	15.1	7.1
pxvn	Maclean, VA, USA	8.8	37.8
isi	Los Angeles, CA, USA	8.3	18.6
eqsj	San Jose, CA, USA	4.5	18.6
eqhk	Tokyo, Japan	4.2	-

Table 4: The list of the 8 UltraDNS clusters reachable from PlanetLab.

serve a smaller percentage of nodes due to the fact that the concentration of PlanetLab nodes is smaller in those continents (cf. Table 1). Even though UltraDNS nodes respond to both the TLD1 and TLD2 anycast addresses, the anycast request distribution across TLD1 and TLD2 is totally different. For example, while `pxpa` receives 23% of the queries for TLD1 it receives only 7% of the queries for TLD2. To understand this behavior, we investigated whether DNS queries directed from the same client to the TLD1 and TLD2 anycast addresses, are indeed resolved by UltraDNS nodes belonging to the same ultraDNS cluster.

For a given PlanetLab node PL_n, we denote the list of TLD1 and TLD2 clusters PL_n contacts, by vectors l_1 and l_2 respectively. We define the correspondence (or similarity) between these lists of clusters as $l_1 \cdot l_2$, the inner product of l_1 and l_2 vectors. A correspondence of one implies that the lists are the same while a correspondence of zero implies that the two lists are completely different. Intermediate values imply a non-empty intersection. For example, assume that a given PlanetLab node contacts clusters `pxpa` and `abld` for TLD1 name resolution and clusters `pxpa` and `eqab` for TLD2 name resolution, then $l_1 = [1,0,0,1,0,0,0,0]$ and $l_2 = [1,1,0,0,0,0,0,0]$ (following the order of clusters used in Table 4). The correspondence between the l_1 and l_2 is then equal to:

$$\frac{1 \cdot 1 + 1 \cdot 0 + 0 \cdot 1 + 6 \cdot 0 \cdot 0}{\sqrt{2} \cdot \sqrt{2}} = \frac{1}{2}$$

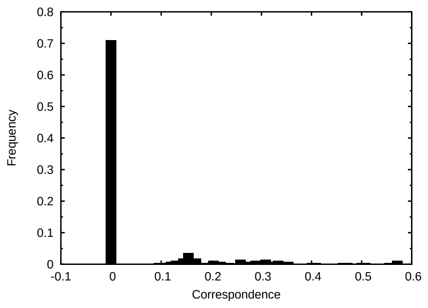


Figure 2: Histogram of correspondence between TLD1 vs TLD2 clusters contacted by PlanetLab nodes.

As Figure 2 depicts, the majority of the PL nodes ($> 71\ \%$) have a correspondence of zero, indicating that queries directed at TLD1 and TLD2 anycast addresses, originating from the same PlanetLab node are answered by different clusters. The benefit of such a configuration is that in the event of a network problem in one of the anycast addresses, the other address can be used, thus ensuring uninterrupted DNS service.

The reason why PlanetLab nodes often pick different clusters for TLD1 and TLD2 name resolution, is that UltraDNS uses two different carriers for TLD1 and TLD2 BGP advertisements. Data from Route-views [23] and traceroutes from PlanetLab nodes to tld1/tld2.ultradns.net reveal that traffic to TLD1 is mostly routed via ASN 2914 (Verio) while traffic to TLD2 is mostly routed via ASN 2828 (XO communications). This means that UltraDNS uses Verio for advertising TLD1 and XO for TLD2. The use of two different providers for TLD1 and TLD2 also explains why in Table 4 the clusters `abld` and `eqhk` receive zero queries. XO communications has no peering points outside North America and so queries from PlanetLab nodes in Europe and Asia are routed towards a US cluster.

5 Evaluation

This section examines (1) the query latencies for the monitored zones; (2) the availability of the monitored zones; (3) the affinity of clients to server they are directed to; (4) the percentage of clients not reaching the replica server that is closest to them and the additional delay incurred and (5) the effect a server's advertisement radius on the load received by that DNS server. Three primary factors affects these performance metrics (a) the relative distribution of anycast servers with respect to the placement of planet lab nodes; (b) the anycast scheme in use and (c) the load on the anycast server chosen.

5.1 Response times

Table 5 presents the mean and standard deviation of query latencies for the monitored zones over the whole measurement period. Figure 3 shows the response time CDFs of the various anycast schemes. The trends observed can be explained using the relative geographical placement of anycast servers argument. First, F-root has the smallest query latency among all the actual deployments. This is a direct consequence of the fact that F-root has the largest number of deployed sites and therefore queries travel shorter distances compared to other zones. Also, the average latency for TLD1 is marginally lesser than that of TLD2 as clusters `abld` and `eqhk` are not reachable for the TLD2 anycast address (Table 4) and therefore queries from clients in Europe and Asia have to travel to the US. K-root, however has a higher average response time than B-Root, due to the fact that the K-root servers are located in Europe and the Middle East, while most of the PlanetLab nodes are in North America. TLD1 and TLD2 exhibit increased variance in the response times due to two reasons viz. variability in the delay of the network paths and variability in the load on the anycast server. As we already explained, UltraDNS anycast addresses are globally announced. We will later show in Section 5.3 that this results in clients experiencing a higher number of flips, and consequently higher fluctuation in DNS response times. The other reason is evident from Figure 4[3]. The spikes in query times around Sep. 30 and Oct. 2, were due to intermittently higher query times of DNS requests sent to the `eqab` ultraDNS cluster. Since BGP ignores variations in traffic load, queries continue to be routed to the `eqab` cluster in spite of a slowdown in the responsiveness.

The last two rows of Table 5 represent synthetic results derived from actual measurements. The TLD1+TLD2 row represents the average query latency for clients that choose the closest server between TLD1 and TLD2, to direct their queries to. Remember that UltraDNS, which is authoritative for the .org and .info top level domains, uses two anycast addresses for these domains' nameservers. So this row represents the best case scenario where a client can measure the latency to each of the nameservers and subsequently direct its queries to the closest nameserver. Indeed, clients based on BIND 9 exhibit this behavior [26]. The last row of Table 5 shows the average latency of the hypothetical zone with all the F-root servers directly accessible by their unicast addresses. The reason why the latency of this zone is lower than F-root is that only two of the F-root servers are global and many clients don't have visibility to local nodes that are closer to them. The last two rows represent in some sense the optimal values of the response time that can be achieved using the F-Root and UltraDNS infrastructures respectively.

[3] Due to space constraints, we don't present the average daily response times for the other zones. Their response times did not exhibit the variability seen in TLD1 and TLD2.

Nameserver	**Mean(ms)**	**Std. Dev. (ms)**
B-Root	115	121
K-Root	140	104
F-Root	75	85
TLD1	96	207
TLD2	104	237
TLD1+TLD2	69	173
Hypothetical unicast	45	13

Table 5: Statistics of DNS response times for the cases we monitored.

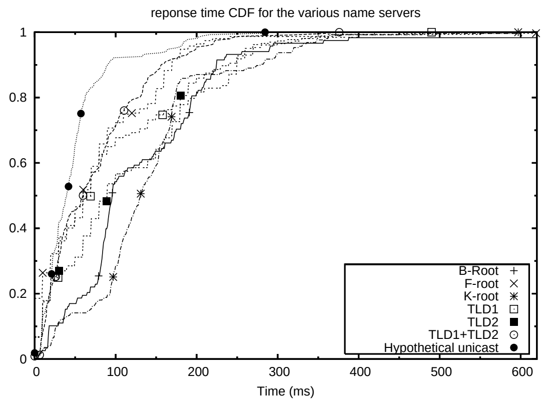


Figure 3: Response time CDF.

5.2 Availability

Considering the reliance of most Internet applications on DNS, ensuring continued availability is a prime requirement for top level name servers. Figure 5 shows the histogram of the percentage of queries which were unanswered by the various nameservers. For all the measured zones, the average percentage of unanswered queries is extremely low ($\leq 0.9\%$). The benefit of deploying servers in multiple locations is evident from the fact that F-Root and K-Root perform better than the B-Root. In fact, robustness increases with geographic diversity. This is the reason why F-Root has smaller number of outages compared to K-Root eventhough both of

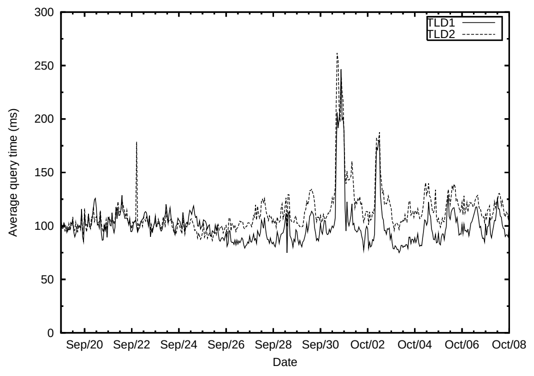


Figure 4: Average daily response time for TLD1 and TLD2.

these zones have similar anycast hierarchies. TLD1 and TLD2, however experience more failed queries while compared to F-root and K-root. While we don't fully understand why this is happening, our conjecture is that is related to the fact that all UltraDNS clusters are global. As a result, clients follow more different paths to reach their servers and are therefore more exposed to BGP dynamics when links fail.

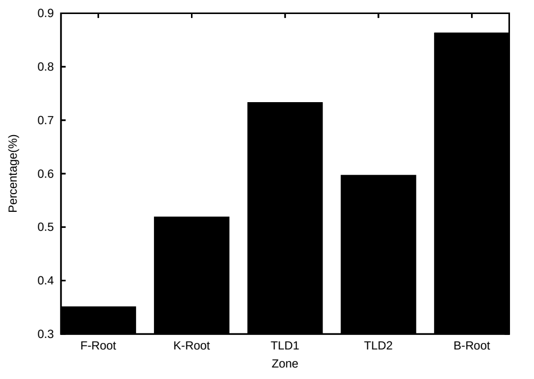


Figure 5: Percentage of unanswered queries by various zones.

We use the term "outage" to indicate a window of time when a node is unsuccessful in retrieving a record from the anycast server servicing it. Outages can be caused by unavailability of the DNS server or loss of network connectivity between the PlanetLab node and the anycast server. In a small number of cases, PlanetLab nodes lose connectivity to their local network and so our results only represent the higher bound of unavailability. Figure 6 plots the outage duration CDFs for all the nameservers. The first observation from the graph is that all outages last at least 30 seconds because of the time granularity with which we send dig requests. Second, outages for the hypothetical unicast zone have the shortest duration. Indeed some (20%-30%) of the PlanetLab experienced no outages. The maximum outage time is around 100 seconds, indicating that in the worst case a client will get a response after contacting at most three servers. At the same time, the mean outage duration is approximately 40 sec, two to three times shorter from the other zones. The combined zone TLD1+TLD2 experiences a larger number of outages but their duration is short. This is because we record an outage when a PlanetLab node switches from non-responding server in TLD1 to TLD2 (or vice versa). This result validates the intuition mentioned in Section 3 that having multiple anycast addresses reduces the effect of network outages on DNS clients. All the other services have considerable overlap and exhibit similar behavior. This reveals an interesting fact regarding anycast. Since anycast relies on Internet routing, once an outage has occurred the recovery time of anycast is governed by the recovery time of the network routing fabric. In fact, about 35% of the outages span a duration higher than 180 seconds. This is a direct consequence of the results presented by Labovitz et.al regarding delayed network convergence [17]. The outage recovery time is independent of the anycasting scheme used. What different anycast setups do achieve is varying reduction in severity of the outage i.e. the number of clients affected by an outage.

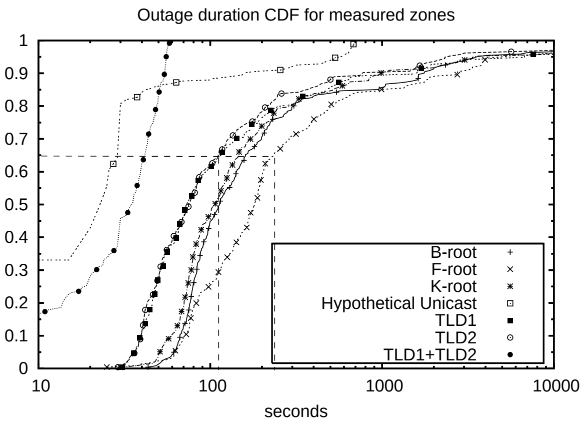


Figure 6: CDF of outage duration

For zones such as F-root and K-root, the number of unanswered queries spiked when a global cluster became unavailable. This is reasonable, since local nodes serve a comparatively smaller community and so the effect of their unavailability on the outage statistic is not so severe. E.g. We noticed a spike in the number (≈15 times the average) of unanswered queries on Sep 21. This was due to the temporary unavailability of the SFO cluster.

Similarly for the K-Root, a spike (≈20 times) on Sep 27 occurred due to an outage in the service of the AMS cluster. The spikes in TLD1 and TLD2 outages are much less drastic. E.g. the maximum spike in TLD1 and TLD2 was noticed on Sep 30, due to intermittent loss of availability to the `eqab` cluster. However, this only increased the number of unanswered queries by approximately 8 and 6 times the average for TLD1 and TLD2 respectively. Interestingly, comparison with the response time graphs of TLD1 and TLD2 during the same time period reveals that DNS response times too peaked simultaneously (cf. Fig. 4).

5.3 Affinity

Anycast does not guarantee that packets from a client will be consistently delivered to the same anycast group member. As a matter of fact, given the implementation of anycast outlined in Section 2, one expects that destinations will change over time. Protocols that keep state at the connection end-points (e.g. TCP) will be negatively affected by such changes. For this reason, RFC 1546 [19] suggests using the unicast address for TCP connections, once the anycast server is discovered, so clients are ensured of communicating with a single server. This doesn't mean that applications using short transactions (e.g. DNS) are not affected by such changes. If information among the anycast group members is not perfectly synchronized, DNS queries may encounter inconsistencies during "server flips".

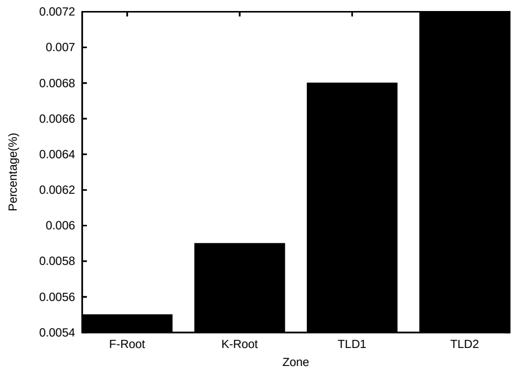


Figure 7: Percentage of flips observed.

In this section we present our findings on server switches (or flips) for the monitored anycast zones. We classify flips into two categories: *inter-cluster* and *intra-cluster*. An inter-cluster flip happens when consecutive client requests are directed to two different geographic clusters (e.g. `SFO2b`→`LAX1b`) and is due to BGP fluctuations. Intra-cluster flips happen when the same client is directed to different members located inside the same cluster (e.g. `SFO2a` →`SFO2b`) and are due to local load balancing at the anycast cluster. As seen in Sec. 5.1, the rate of flips affects the fluctuations in query times. Stability of service is more sensitive to inter-cluster flips than intra-cluster ones, because inter-cluster flips involve a change of transit route, and different routes may have widely different delay characteristics.

Figure 7 provides a histogram of the number of inter-cluster flips observed in various zones. Inter cluster flips in hierarchical anycast deployments mostly occur between the global nodes. The majority of the flips ($> 90\%$) for F-Root are between the `PAO` and `SFO` clusters, while for K-Root between `AMS` and `LINX`. Furthermore the total number of inter-cluster flips observed in the F-Root and K-Root nameservers is 20% lower compared to TLD1 and TLD2. This follows from the fact that ultraDNS anycast clusters are globally visible while a majority of F-Root and K-Root are local clusters. Therefore at a client gateway, BGP paths to a greater number of ultraDNS clusters are available compared to F-root clusters. Hence, ultraDNS server selection is more prone to BGP fluctuations (due to path failures). Since the hypothetical unicast zone optimizes the latency time by picking the closest server, it has the highest number of flips.

Nameserver	Flips following an outage (%)
F-Root	65
K-Root	63
TLD1	52
TLD2	51

Table 6: Percentage of flips due to outages.

Flips often correlate to outages. E.g, on Sep 21st (Fig. 9), a considerable number of PlanetLab nodes faced outages in the service from the `SFO` cluster, for the F-Root zone. After a brief outage of over a minute, service resumed with nodes contacting the `PAO` cluster for F-root name resolution instead. Similarly, on Sep 27th for the K-Root zone, all PlanetLab nodes using the `AMS` cluster experienced an outage. After a brief interval spanning over two minutes, all these PlanetLab nodes flipped their choice of the anycast server and instead contacted the `LINX` cluster. However, flips need not necessarily occur successively after outages. As Table 6 shows, in the case of TLD1 and TLD2 UltraDNS service, the occurrence of flips and outages are related to a lesser extent. Since UltraDNS clusters are all global nodes, flips are more frequent and half of the time occur independently of outages.

We are also interested in the time that PlanetLab nodes remain stable to the same server. We found that there is a range of 5 orders of magnitude in this metric! As Figure 8 illustrates while the median time a node remains stable to the same server is around 100 minutes, the lower 10% of the nodes change servers every 1 minute, while the most stable clients consistently choose the same server for days or weeks. This behavior is evidence that a small number of network paths are very stable while most other paths suffer from outages and a small percentage of paths have a pathological number of outages.

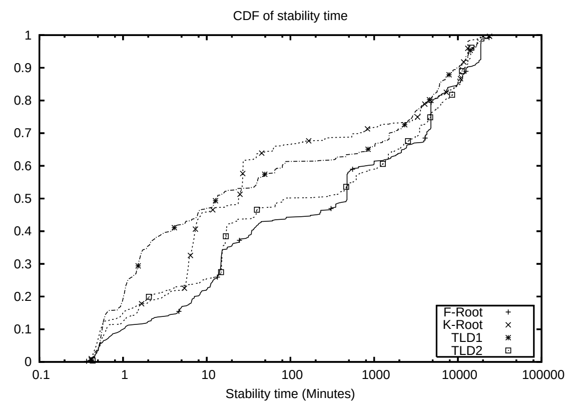
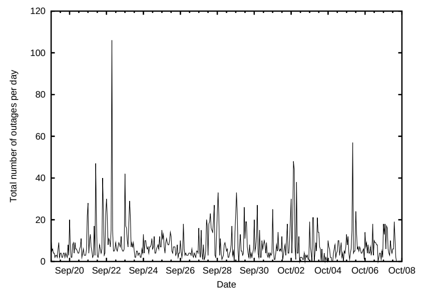
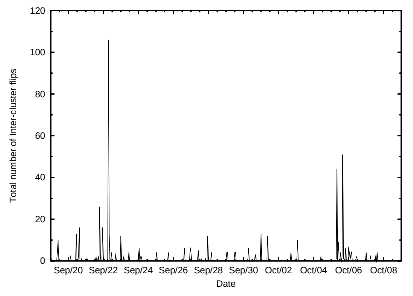


Figure 8: Period of time that PlanetLab nodes query the same server for the monitored zones.

Until now we have only discussed the geographical load balancing aspect of anycast and how it is affected by BGP route fluctuations. Load balancing is however also used internally by DNS clusters to distribute queries among the individual servers that make up the cluster. F-root uses IGP-based (OSPF) anycast for load balancing [2], but other configurations could use hardware based load balancers. Load balancers use either a per-packet or a per-flow mechanism. To discover the load balancing scheme in the nameservers, we use to our advantage the fact that each PlanetLab sites contains multiple nodes. These nodes can be expected to contact the same anycast cluster. The similarity between the anycast servers of single site nodes provides a hint to the type of load balancer used within each cluster. Large correlation between the servers contacted by the nodes of the same site, indicates a per-packet load balancer (given a round-robin load-balancing scheme we expect that packets from each client will be sent to all the servers inside the DNS cluster). On the other hand, low correlation indicates flow based load distribution (a technique often used by load balancer hashes the clients' source addresses). Using this technique we discovered that all the candidate nameservers used a flow based technique, except for the B-Root server which used a per packet load balancer. We observed that the B-root server faced a flip every half a



(a) Timeline of total daily F-root outages.



(b) Timeline of total daily F-root flips.

Figure 9: Correlation of outages and flips for the F-root server. A similar correlation was observed for the K-Root.

minute. This is typical of a per-packet load balancing technique, where successive data packets are sent to different servers without regard for individual hosts or user sessions. The other zones listed in the figure experience negligible number of intra cluster flips. Even in the hash-based flow sharing case, intra cluster flips may occur due to variations such as OSPF weight changes or equipment failures. In general, flow based hashing is preferred over per-packet load balancing as it consistently directs packets from a single client to the same cluster member.

5.4 Effectiveness of Localization

As our earlier results indicate, the use of anycast decreases query latencies by localizing client requests amongst the various DNS server replicas. However, the comparison of the F-root query latency to that of the hypothetical zone where all the servers are individually addressable (c.f Table 5) seems to suggest that anycast does not always pick the closest server. This raises the interesting question: Does anycast always lead clients to the closest instance among all the servers in the anycast group? If not, how much farther away is the selected server as compared to the closest? Anycast server selection depends on the path selected by BGP. These routing decisions are influenced by policies and sub-optimal heuristics such as using the path with the shortest AS hop count and can therefore lead to suboptimal choices. In fact, it is well known that in many cases the paths chosen by BGP are not the shortest [24, 25].

An Optimistic Estimate: A direct comparison between the ping times of all the clusters and the selected cluster is potentially flawed due to a subtle reason. As we pointed out earlier in Section 2, the unicast addresses of the DNS servers are selected from different address ranges and therefore the path from a client to the anycast address are different from the path to the unicast address of the server. We use a reasonable approximation to get around this difficulty. Note that if a traceroute to the last hop router and the anycast address follow the same path, then we can obtain a good approximation of the ping time to the different clusters by using the ping time to the last hop instead. Using traceroutes from the Planet Lab nodes, we found that this was indeed the case for the F-Root and TLD2 zones, but not so for TLD1 and the K-Root. Figure 10 presents the CDF of the additional network latency incurred by dig requests compared to the closest anycast cluster. As seen from the figure, majority of the anycast queries reach their nearest cluster. We see that about 60% of all the F-Root requests are sent to the nearest F-cluster and 80% of the TLD2 requests are sent to the nearest TLD2 cluster. It must be however be noted that this is an upper bound on the localization metric for the F-Root, as not all the anycast clusters are visible to the Planet Lab nodes (c.f. Table 2)

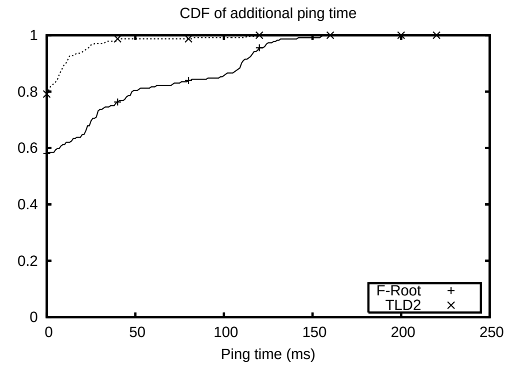


Figure 10: Additional ping time for the F-root and TLD2 zones.

A Pessimistic estimate: We also measured the effectiveness of localization using a different approach, which yields a pessimistic estimate. First, we calculate the geographic distance of each of the PlanetLab nodes to all the listed DNS clusters in a zone. We do so by calculating the length of a hypothetical straight line over the globe connecting the geographic locations of the PlanetLab node and the DNS server. The locations of PlanetLab nodes are available through the PlanetLab website. Then, we compare these geographic distances and determine whether the PlanetLab node contacts the geographically closest server in that zone. While it is known that Internet paths are longer than the direct geographic path connecting two end-points [10, 25], we assume that all paths exhibit the same *path inflation* factor. Based on this assumption, we can directly compare geographic distances to determine whether the best Internet path is selected for each client.

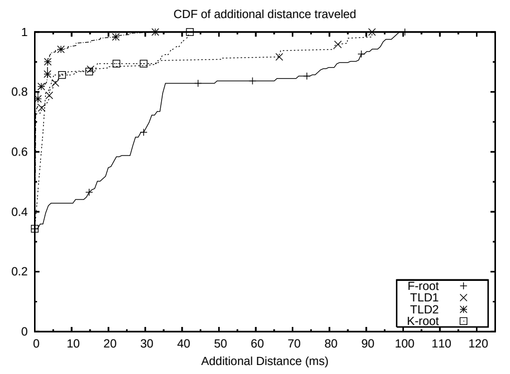


Figure 11: Additional distance over the optimal traveled by anycast queries to contact their F-root, K-root, TLD1 and TLD2 server.

Figure 11 presents the cumulative distribution of the additional distance across all PlanetLab nodes for each zone. We observe that about 37% of all the anycast requests are sent to the nearest F-root server while 35% of the anycast requests are sent to the nearest K-root server. Approximately 75% requests are served by the nearest TLD1 and TLD2 servers. Infact, the CDF for TLD2 closely matches with that in Figure 10. However, this is not the case with F-Root, because not all the clusters are visible from Planet Lab and consequently not accounted for in Figure 10.

Using these two estimates, we can conclude that the effectiveness of localization for the F-Root is between 37%-60%, $\geq$ 35% for the K-Root, while for TLD1, it is $\geq$ 75% and for TLD2 between 75%-80%. It is not surprising that TLD1 and TLD2 zones perform considerably better than other deployments. Not only a larger portion of nodes contact the closest server but the additional distances for those that don't, are also shorter. The reason is that UltraDNS clusters are not differentiated into global and local. Consequently, PlanetLab nodes have visibility to a greater number of BGP routes to UltraDNS clusters. Therefore, it is more likely that anycast chooses the nearest UltraDNS cluster. In a somewhat counter-intuitive way, the slowest 10% of TLD1 clients follow worse paths compared to TLD2 even-though TLD1 is advertised by two additional locates (London, Tokyo). We explain this behavior by an example. Consider a client in Asia. If it doesn't pick the HK site for TLD1, its requests are directed to the US. Thus the large additional distance. On the other hand, TLD2 is not advertised in HK and therefore clients correctly pick the US sites. A similar effect is visible for K-Root also. Clients don't traverse large additional distances compared to the closest cluster due to the fact that all their clusters are located within a relatively small geographical area.

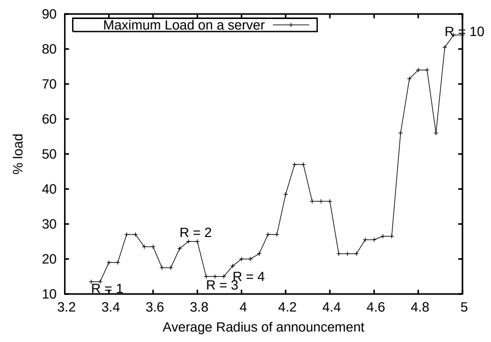


Figure 12: Variation of server load with varying server advertisement radius for a random distribution of 200 clients. Redundancy is denoted by R.

5.5 Effect of Advertisement Radius

Hierarchical anycast schemes claim that by changing the advertisement radius of a server the load on the server can be controlled. The goal of this section is to show how effective this technique really is in practice.

Algorithm 1: Radius adjustment algorithm

```
(1)   $Radius[1 \ldots Num\ of\ servers] \leftarrow 5.$
(2)   $Calculate$ *Redundancy*
(3)   $S \leftarrow server\ with\ max\ load.$
(4)   **while** *Redundancy* $>= 1$ **do**
(5)       $Radius[S] = Radius[S] - 1$
(6)       $Calculate\ load\ on\ each\ server.$
(7)       $Calculate$ *Redundancy*
(8)       $S \leftarrow server\ with\ max\ load.$
```

To measure the effect of radius advertisement on server load, we simulate the AS level topology of the Internet using the connectivity data available from Route Views [23]. Server placement is done based on the actual placement of F-root servers available from [14]. An initial advertisement radius is assigned to each of the servers. If a server has a radius of r then its prefix advertisement is visible r AS hops from the origin AS of this server. Finally we position 200 clients randomly across the set of all autonomous systems. While we understand that this setup is not a true representation of the distribution of DNS clients over the Internet, it nevertheless serves our goal of studying the effect of advertisement radius on the load experienced by the servers. Each client selects the server with the shortest AS path among all the visible paths.

We use the term "redundancy" to denote the minimum number of servers that is reachable by any client. At the beginning of the simulation, we fix the radius of all servers to be equal to a sufficiently large value (we used an initial radius of five). We then gradually reduce the radius of the server with the maximum load, thus confining the server to serve smaller communities using Algorithm 1. We iterate this process until there exists some client which is outside the sphere of influence of all the servers, i.e. it has redundancy of zero.

Figure 12 plots the load on the maximally loaded server as a function of the average radius. Initially, when each server has radius equal to 5, every client can reach at least ten servers, while the busiest server serves 80$\%$ of the traffic. As we decrease the radius of this server, its load decreases until another server becomes the maximally loaded server. Intuitively, as the average advertisement radius decreases the maximum number of clients served by a single host also decreases, thereby distributing excess load to other servers. At the limit, the highest loaded server receives about three times the optimal load (if clients were evenly distributed across servers). Based on this graph, we can see that an ideal operational region exists where the maximum server load is low while redundancy is greater than one. While this result is encouraging, it indicates that an adaptive mechanism is needed to minimize server load while keeping adequate redundancy levels. As far as we know, zones employing the global/local hierarchy don't use such a mechanism today.

We also calculated the average path length as a function of the radius presented in Figure 13. Initially clients have to travel a distance of approximately two ASes to reach their closest server. However, as the average radius decreases, the path length increases and consequently query latency also increases. The step-wise increase in path length shown in Figure 13 path length as is due to the nature of the Internet graph. A small number of ASes have extremely high degree and have very short distance d to the majority of the other ASes [9]. As long as the radius r of a DNS server located in one of these "hub"

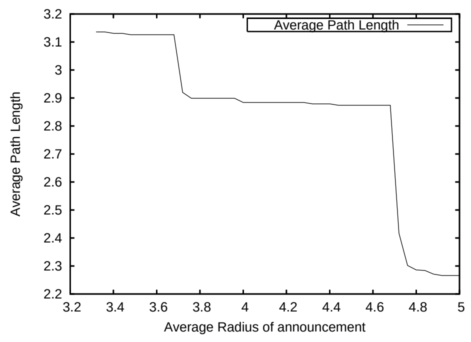


Figure 13: Variation of average AS path length with change in the radii of the server for a random distribution of 200 clients.

autonomous systems is higher than d most of the clients are directed to this server. When $r < d$ then clients are directed to a more distant server and thus the average path length increases.

6 Comparison of Strategies

We can broadly categorize existing anycast configurations into two schemes: hierarchical and flat. The hierarchical scheme distinguishes anycast nodes into local and global while in the flat scheme all the nodes are globally visible. Anycast servers in the flat configuration tend to have a uniform distribution of load. Also since, there is a greater diversity of choices of available anycast servers to a client, the distance between clients and DNS servers is generally shorter. However, Sections 5.1 and 5.3 show that having a large radius of advertisements has an adverse effect on the stability of the response times and increases the frequency of server changes (flips) of the anycast service. This is because the larger the radius of advertisements is, the greater is a server's sphere of influence. This consequently increases the number of choices of servers available at a client. In the hierarchical configuration, global nodes are generally under heavier load since they attract the bulk of the client queries. Section 5.5 shows that the load on a single server can be mitigated by decreasing the radius of advertisements at that server. Ideal operation would argue for a balance between sufficient redundancy and uniform load distribution. To do so, a large number of local nodes need to be deployed. This is why the F-Root server has better performance, higher stability, and lesser outages than the K-Root server.

On the critical issue of high availability, we have shown that deploying multiple anycast addresses pays off. Section 5.2 shows that even though TLD1 and TLD2 individually have high outage duration, the combined zone TLD1+TLD2 has outages that are smaller by an order of magnitude and also shorter than any other of the schemes that we measured.

7 Related Work

A number of existing studies have looked at the performance of the DNS infrastructure. Danzig et al. presented measurements related to DNS traffic at a root name server [6]. Their main result was that the majority of DNS traffic was caused by bugs and misconfigurations. Most of these problems have been fixed in recent DNS servers. Our work focuses on analyzing the effect of anycast in DNS performance, a feature that didn't exist at the time of Danzig's study. More recently, Brownlee et al. monitored the DNS traffic from a large campus network and measured the latency and loss rate of queries sent to the root nameservers [5]. The main goal of that paper was to create a model of DNS request/response distribution. Our results on average latencies and loss rates match those presented in that study. More interestingly, the authors of [5] observed that query times show clear evidence of multipathing behavior and conjectured that this is due to load balancing or changes in server load. Anycast at the BGP level and within a cluster, is a key cause of this observed multipathing. Jung et al. measured the performance of all DNS requests sent from the MIT campus and investigated the effect of caching on DNS performance [15]. Wessels et al. compared the effect of different caching techniques on the root nameservers [26]. Their results show that some caching servers favor nameservers with lower round-trip times while others don't. This indicates that the use of anycast benefits at least some resolvers since it transparently leads them to (approximately) the closest instance. On the other hand, resolvers that actively select the DNS server with the closest distance would see a performance benefit if the unicast addresses of the servers were exposed as we showed in Section 5.1.

The effectiveness of anycast in providing redundancy and load sharing has been exploited in a number of proposals. The AS112 project reduces unnecessary load on root nameservers by directing queries for local zones to a distributed black hole implemented via anycast [3]. The use of anycast has also been proposed for finding IPv6 to IPv4 gateways [13] and to implement sink holes for the detection and containment of worm activity [11]. Engel et al. provide results from their measurement of load characteristics on a set of mirror web sites using anycast [8]. Hitesh et al. present a scalable design for anycast and use a small subset of the PlanetLab nodes to measure the affinity of existing anycast deployments [4]. While this work has some similarity to ours, their focus is on the design of an anycast scheme while we are interested on the performance of existing anycast deployments in DNS.

Finally, a number of proposals have looked at alternatives to the existing DNS architecture with the goal of improving query performance [18, 20]. Our work is different. We are interested on exploring how anycast can improve the existing DNS rather than propose new naming systems.

8 Summary

In this paper we presented an analysis on the impact of anycast in DNS performance based on the measurement of four top-level zones. We found that in general the deployment of anycast decreases the average query latency. On the other hand, the majority of clients (55%-85%) are not directed towards the closest anycast server. The reason is twofold: First, clients may not be aware of the presence of nearby anycast servers because advertisements from these servers don't reach them. This is the case for deployments that use the global/local scheme (F-root and K-root). Second, because advertisements are carried via BGP, clients cannot base their decisions on the path latency. Using synthetic co-ordinates we found that clients may have to travel additional distance that is comparable to total query latency. The fact that advertisements from anycast servers are carried over BGP also has negative effects in the availability of DNS zones. While the number of outages is relatively small, they are longer in duration (50% last more than 100 seconds), affected by long routing convergence times. Finally, we uncovered an inherent trade-off between reduced query latency and stability. When clients have visibility to more anycast servers, they can choose the optimal server. The negative effect though is, that in this case the zone becomes vulnerable to increased number of outages.

While this trade-off is clear from the results presented here, we don't fully understand the underlying mechanisms that connect the scope of BGP advertisements, the rate of flips, and the duration of outages. To do so, would require access to the BGP advertisements at each monitoring point that were unfortunately unavailable. We are currently developing a theoretical model for the effect of link failures on service outages that we plan to validate via simulations. We believe that this model, coupled with access to the actual BGP advertisements, will explain the reasons behind some of the adverse effects observed in our study.

Acknowledgments

Joe Abley graciously responded to our queries regarding the implementation of anycast in the F-root servers. We would also like to thank Lixia Zhang, Claudiu Danilov and Alexandros Batsakis for their valuable comments that helped us improve this paper.

References

[1] J. Abley. Hierarchical Anycast for Global Service Distribution, 2003. http://www.isc.org/pubs/tn/?tn=isc-tn-2003-1.html.

[2] J. Abley. A Software Approach to Distributing Requests for DNS Service Using GNU Zebra, ISC BIND 9, and FreeBSD. In *Proceedings of USENIX 2004 Annual Technical Conference, FREENIX Track*, 2004.

[3] The AS112 Project. http://www.as112.net.

[4] H. Ballani and P. Francis. Towards a deployable IP Anycast Service. In *Proceedings of WORLDS*, Dec. 2004.

[5] N. Brownlee and I. Ziedins. Response time distributions for global name servers. In *Proceedings of PAM 2002 Workshop*, Mar. 2002.

[6] P. B. Danzig, K. Obraczka, and A. Kumar. An Analysis of Wide-Area Name Server Traffi c. In *ACM SIGCOMM'92*, 1992.

[7] R. Elz, R. Bush, S. Bradner, and M. Patton. Selection and Operation of Secondary DNS Servers. July 1997.

[8] R. Engel, V. Peris, and D. Saha. Using IP Anycast for Load distribution and Server Location. In *Proceedings of Global Internet*, Dec. 1998.

[9] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *SIGCOMM*, pages 251–262, 1999.

[10] L. Gao and F. Wang. The extent of AS path infl ation by routing policies. In *Proceedings of Global Internet Symposium, 2002*, 2002.

[11] B. R. Greene and D. Mcpherson. ISP Security: Deploying and Using Sinkholes. http://www.nanog.org/mtg-0306/sink.html.

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

[13] C. Huitema. An Anycast Prefi x for 6to4 Relay Routers. *RFC 3068*, June 2001.

[14] I. S. C. Inc. ISC F-Root. http://www.isc.org/ops/f-root/.

[15] J. Jung, E. Sit, H. Balakrishnan, and R. Morris. DNS Performance and the Effectiveness of Caching. *IEEE/ACM Trans. on Networking*, Oct. 2002.

[16] RIPE NCC K-Root. http://k.root-servers.org/.

[17] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed internet routing convergence. In *Proceedings of ACM SIGCOMM 2000*, pages 175–187, 2000.

[18] K. Park, V. S. Pai, L. Peterson, and Z. Wang. CoDNS Improving DNS Performance and Reliability via Cooperative Lookups. In *Proceedings of OSDI'04*, Dec. 2004.

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

[20] V. Ramasubramanian and E. G. Sirer. The Design and Implementation of a Next Generation Name Service for the Internet. In *Proceedings of ACM SIGCOMM 2004*, Aug. 2004.

[21] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). *RFC1771*, March 1995.

[22] I. Research. Planet Lab. http://www.planet-lab.org/, 2002.

[23] The Route Views Project. Available at http://www.antc.uoregon.edu/route-views/.

[24] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The End-to-End Effects of Internet Path Selection. In *Proceedings of SIGCOMM 1999*, Aug. 1999.

[25] N. Spring, R. Mahajan, and T. Anderson. Quantifying the Causes of Path Inflation. In *Proceedings of ACM SIGCOMM*, Aug. 2003.

[26] D. Wessels, M. Fomenkov, N. Brownlee, and K. Claffy. Measurements and Laboratory Simulations of the Upper DNS Hierarchy. In *Proceedings of PAM 2004*, Apr. 2004.