A Measurement-based Deployment Proposal for IP Anycast

Hitesh Ballani
Cornell University
Ithaca, NY
hitesh@cs.cornell.edu

Paul Francis
Cornell University
Ithaca, NY
francis@cs.cornell.edu

Sylvia Ratnasamy
Intel Research
Berkeley, CA
sylvia@intel-research.net

ABSTRACT

Despite its growing use in critical infrastructure services, the performance of IP(v4) Anycast and its interaction with IP routing practices is not well understood. In this paper, we present the results of a detailed measurement study of IP Anycast. Our study uses a two-pronged approach. First, using a variant of known latency estimation techniques, we measure the performance of current commercially operational IP Anycast deployments from a large number (>20,000) of vantage points. Second, we deploy our own small-scale anycast service that allows us to perform controlled tests under different deployment and failure scenarios. To the best of our knowledge, our study represents the first large-scale evaluation of existing anycast services and the first evaluation of the behavior of IP Anycast under failure.

We find that: (1) IP Anycast, if deployed in an ad-hoc manner, does not offer good latency-based proximity, (2) IP Anycast, if deployed in an ad-hoc manner, does not provide fast failover to clients, (3) IP Anycast typically offers good affinity to all clients with the exception of those that explicitly load balance traffic across multiple providers, (4) IP Anycast, by itself, is not effective in balancing client load across multiple sites. We thus propose and evaluate practical means by which anycast deployments can achieve good proximity, fast failover and control over the distribution of client load. Overall, our results suggest that an IP Anycast service, *if deployed carefully*, can offer good proximity, load balance, and failover behavior.

Categories and Subject Descriptors: C.4 [Performance of Systems]: Measurement techniques, Performance attributes.

General Terms: Measurement, Performance.

Keywords: IP Anycast, BGP, Proximity, Affinity, Failover, Load Distribution.

1. INTRODUCTION

IP Anycast [28] is an addressing mode in which the same IP address is assigned to multiple hosts. Together, these hosts form an IP Anycast *group* and each host is referred to as an anycast *server*. Packets from a client destined to the group address are automatically routed to the anycast server closest to the client, where "closest" is in terms of the metrics used by the underlying routing protocol. Since Internet routing does not differentiate between multiple routes to multiple hosts (as in IP Anycast) and multiple routes to the same host (as in multihoming), IP Anycast is completely backward compatible requiring no changes to (IPv4 or IPv6) routers and routing protocols.

IP Anycast offers an attractive primitive for service discovery – the route-to-closest-server abstraction offers reduced access latency for clients, load-balancing across servers, and network-level resilience to DDoS attacks while its implementation at the network layer allows these advantages to be realized with no special configuration at clients or servers and with no dependence on higher-layer services such as the DNS. While this potential has long been recognized [7,26,28], it is mostly in recent years that IP Anycast has gained in importance. This is in large part due to its use in the critical DNS root-server deployment – six of the thirteen DNS root-servers have been transparently replicated using IP Anycast and this deployment continues to grow [18]. In addition however, IP Anycast is being used in a growing variety of infrastructure services. For example, IP Anycast is used to improve the performance of caching DNS servers [27], for drawing in private address space DNS queries as part of the AS-112 project [39], to discover rendezvous points for multicast groups [21], as a transition mechanism from IPv4 to IPv6 [19], for sinkholing DoS attacks [16] and for redirection in commercial CDNs [40]. Finally, recent research efforts have used IP Anycast to build a proxy-based anycast service [4] and a next-generation architecture deployment service [3,30].

However, despite its growing use in critical infrastructure services, IP Anycast and its interaction with IP routing practices is not well understood. For example, the impact of anycasting of DNS root-servers on clients that should, in theory, access the closest server has not been analyzed in any detail. Similarly, there has been no exploration of whether root-server operators can control the load on individual servers by manipulating their routing advertisements, nor of the behavior of IP Anycast under server failure. Moreover, the various applications of IP Anycast make different assumptions about the underlying service. For example, the use of IP Anycast in CDNs assumes client packets are routed to a proximal CDN server and that the impact of a server failure

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

IMC'06, October 25–27, 2006, Rio de Janeiro, Brazil.
Copyright 2006 ACM 1-59593-561-4/06/0010 ...$5.00.

on clients is shortlived (*i.e.*, clients are quickly routed to a different server). To gauge the effectiveness of IP Anycast in existing deployments as also the feasibility of future usage scenarios, it is imperative to evaluate the performance of IP Anycast.

A couple of root-server operators [6,12] have offered valuable reports on the performance of their anycast deployments. These are probably the first reports on the performance of IP Anycast and represent the best source of data on operational IP Anycast deployments from the point of view of the anycast servers. However, the analysis is preliminary (as the authors themselves state) and details of the study are not published, nor is the data publicly available. Drawing from these, this paper presents a detailed study of inter-domain IP Anycast as measured from a large number of vantage points. Specifically, our study seeks to answer the following questions:

1. What kind *failover* properties does a typical IP Anycast deployment offer?

2. What kind of *load distribution* do existing IP Anycast deployments see? Also, can the operator of an anycast deployment control this distribution of client load?

3. Past studies [4,32] have reported that existing IP Anycast deployments may offer poor latency-based *proximity*; *i.e.*, many clients may not routed to the server closest in terms of latency. Using a larger number of clients and anycast groups, we aim to confirm and understand the reasons for this poor latency as well as explore possible remedial measures.

4. Past studies [4,6,8,12,31] have presented conflicting reports regarding the *affinity*[1] offered by IP Anycast and consequently, the ability to run stateful services of top of anycast. We seek to measure, at scale, the affinity offered by IP Anycast.

To explore these questions, we study four existing IP Anycast deployments including two anycasted DNS root-servers. In terms of methodology, our study differs from previous efforts on two fronts:

1. We use a variant of the King [17] measurement technique to observe and evaluate IP Anycast deployments from a very large (>20,000) number of vantage points. To the best of our knowledge, this represents a two order of magnitude increase in the number of vantage points from which IP Anycast deployments have been actively probed for evaluation.

2. We deploy our own small scale IP Anycast service for controlled evaluation of anycast under different deployment and failure scenarios. Performing such experiments would be difficult using commercial IP Anycast deployments such as the DNS root-servers.

The main results of this study are as follows:

- We corroborate evidence from past studies indicating that IP Anycast, by itself, does not offer good latency-based *proximity*. For example, for the 13 server J-root deployment, we find 8903 (≈40%) of the 22,281 measured clients are directed to a root-server that is more than 100msec farther away from the closest server. While the impact of inter-domain routing on end-to-end path length has been well documented [34], we find that inter-domain routing metrics have an even more severe impact on the selection of paths to anycast destinations.

- We propose and evaluate a practical deployment scheme designed to alleviate the *proximity* concerns surrounding IP Anycast. Specifically, ensuring that an ISP that provides transit to an anycast server has global presence and is (geographically) well covered by such servers improves the latency-based proximity offered by the anycast deployment.

- We find that IP Anycast is affected by delayed routing convergence and hence, clients using anycast services may experience slow *failover*. However, our proposed deployment scheme addresses this by reducing the scope of routing convergence that follows a server failure and hence, can ensure that clients failover at a fast rate. For instance, we find that in case of a server failure in an IP Anycast deployment conforming to our proposal, a vast majority (>95%) of clients can be re-routed to other operational servers in less than 20 seconds.

- Through a much larger scale study as compared to past efforts, we find that the anycasting of an IP prefix does not have any unfavorable interactions with inter-domain routing. Hence, IP Anycast offers very good *affinity* for all but a very small fraction of clients. Using temporal clustering, we show that the poor affinity observed by this small fraction of clients can be attributed to dynamic load-balancing mechanisms near them.

- We find that a naive IP Anycast deployment does not lead to an even distribution of *client load* across servers. However, we also propose and evaluate the impact of operators manipulating BGP advertisements at individual anycast servers to control their load. Our results show that such mechanisms can achieve coarse-grained load balancing across anycast servers.

Overall, our measurements show that an IP Anycast service can be deployed so as to provide a robust substrate offering good proximity and fast failover while allowing for coarse-grained control over server load. In what follows, Section 2 reviews related measurement studies, Section 3 details the IP Anycast deployments we measure in this paper while Section 4 describes our measurement methodology. We describe our proximity measurements in Section 5, failover measurements in Section 6, affinity measurements in Section 7 and load distribution measurements in Section 8. Finally, we discuss related issues in Section 9, and conclude with Section 10.

2. RELATED MEASUREMENT STUDIES

An invaluable vantage point for measuring anycast deployments is at the anycast servers themselves. In recent presentations [6,12], the operators of the J and K root-servers report on their analysis of client logs collected at their anycast servers. They present the observed distribution of client load and affinity. In terms of load distribution, both studies

[1] Affinity measures the extent to which consecutive anycast packets from a client are delivered to the same anycast server.

report a skewed distribution of client load across their respective deployments. With regard to affinity, the J-root operators report instances of clients that exhibit poor affinity and conjecture that anycast may not be suitable for stateful services. By contrast, the K-root operators find that most of their clients experience very high affinity.

Our study builds on these earlier reports. Using *active* measurements from over 20,000 clients, we measure the affinity and load-distribution for our own small-scale anycast deployment, explore the reasons behind the observed load and affinity, and evaluate techniques to control server load. In addition to load and affinity, we use active measurements to evaluate the (latency) proximity seen at clients to four different anycast deployments. Finally, using our own deployment, we study the behavior of IP Anycast under server failure. As we describe in Section 6, our desire to use a large number of client vantage points prevents us from measuring the affinity and load to the DNS root-server deployments. However, for completeness, we did perform such measurements from a smaller set of clients that we have direct access to (*i.e.,* PlanetLab nodes and approximately 200 publicly-available traceroute servers). Since the results were consistent with the larger-scale measurements over our own deployment, we only present the latter here. The details of the PlanetLab-based study can be found in [5].

We are aware of two recent efforts that measure the performance of IP Anycast using active probing from clients. Sarat et. al. [31,32] use PlanetLab nodes [11] as vantage points for evaluating the K-root, F-root and .ORG TLD deployments. They measure proximity and affinity and report poor proximity and moderate-to-poor affinity. Similarly, using PlanetLab and approximately 200 volunteer nodes, Boothe *et al.* [8] measure the affinity offered by the anycasted DNS root-servers and report poor affinity. In direct contrast to Boothe *et al.*, PIAS [4] uses PlanetLab-based measurements to claim that anycast flaps are relatively rare for the same anycasted DNS root-servers. Relative to the above, our study in this paper uses a significantly larger number of client vantage points, performs a more detailed analysis of affinity and proximity and evaluates deployment strategies to improve the proximity offered by IP Anycast. In addition, we explore load-distribution due to IP Anycast and its behavior under failure.

3. DEPLOYMENTS MEASURED

An IP Anycast group is associated with an IP address (hereon referred to as the anycast address for the group) and servers join the group by just advertising this address into the routing infrastructure. For an *intra-domain anycast group* with servers restricted to a single administrative domain, this advertisement is into the intra-domain routing protocol for the domain in question. For *inter-domain anycast groups*, each server advertises the anycast address[2] into BGP. Despite the simplicity of the basic idea, the interaction with BGP, the involvement of multiple administrative domains *etc.* raise several interesting questions regarding the behavior of inter-domain IP Anycast. This paper restricts itself to studying issues related to inter-domain IP Anycast. Also, while we focus on IPv4 Anycast, our results should apply equally to IPv6 Anycast deployments.

Name	Anycast prefix	AS#	No. of servers
F root-server [45]	192.5.5.0/24	3557	27
J root-server [48]	192.58.128.0/24	26415	13
AS 112 [39]	192.175.48.0/24	112	20

Table 1: The three external IP Anycast deployments that this paper evaluates.

Server Unicast address	Host-Site	Host-site AS#	Upstream provider
128.84.154.99	Cornell University	26	WCG
12.155.161.153	IR Berkeley	2386	ATT
195.212.206.142	IR Cambridge	65476	ATT-World
12.108.127.148	IR Pittsburgh	2386	ATT
12.17.136.150	IR Seattle	2386	ATT

Table 2: The internal IP Anycast deployment comprising of five servers. Each of these advertise the anycast prefix (204.9.168.0/22) through a BGP peering with their host-site onto the upstream provider. Note that IR stands for "Intel-Research".

Since clients can access an anycast group simply by sending packets to the IP address associated with the group, IP Anycast has been used for transparent replication of many services including the DNS root-servers. For example, the F root-server deployment is currently comprised of 37 servers that form an IP Anycast group and each server advertises the F root-server anycast prefix. The deployment is also associated with its own AS number (AS#) which serves as the origin AS for the anycast prefix. Thus, clients can access a F root-server by sending packets to an address in the prefix. In this paper, we evaluate three such currently operational deployments - two anycasted DNS root-servers and the AS112 anycast deployment.[3] The anycasted AS112 servers are used to draw in reverse DNS queries to and for the link local address space (RFC1918 addresses–10.0.0.0/8, 172.16.0.0/12 and 192.168.0.0/16). Since we have no control over these deployments, we refer to these as *external deployments.* Table 1 gives details for the external deployments including the number of the servers in each deployment at the time of our experiments.[4]

In practice, each "server" in these deployments is a cluster of hosts located behind some form of load-balancing device. For example, for the F root-server, hosts in each cluster form an intra-domain anycast group and it is the server site's gateway routers that balance incoming traffic between them [2]. However, our focus on inter-domain anycast makes our measurements oblivious to this cluster-based deployment. Thus, for the purpose of this paper, each server site can be thought of as a single host.

The production-mode nature of these external deployments makes it difficult, if not impossible, to conduct controlled experiments such as injecting server failure, or manipulating the advertisements at individual servers. For such experiments we deployed our own IP Anycast service that we call the *internal deployment.* For this, we obtained a

[2] In practice, each server must advertise a prefix for a block of addresses into BGP. This is the *anycast prefix* for the group and servers are accessible through all addresses in the prefix.

[3] The choice of the anycast deployment to measure was limited by the need to know the *unicast* addresses of the individual anycast servers for our experiments.

[4] The F root-server and the AS112 deployments have since grown to 37 servers each.

Region	No. of clients	% of Total
North America	12931	54.827
Central America	317	1.344
South America	461	1.954
Europe	5585	23.680
Asia	2402	10.184
S.E. Asia	566	2.400
Oceania	1196	5.071
Africa	187	0.792
Arctic Region	9	0.038
Unknown	204	0.864
Total	23858	100.000

Table 3: Geographic distribution of the clients used in our study.

/22 prefix (204.9.168.0/22) and an AS# (33207) from ARIN and deployed anycast servers at the five sites listed in table 2. Each server advertises this prefix and AS# into BGP through their host site and on to their upstream provider and hence they form an IP Anycast group.

Note that an anycast server's "host-site" refers to the server's immediately upstream AS while the "upstream provider" refers to the closest major ISP (tier-1 or tier-2) that provides transit for the server's traffic. For example, the internal deployment anycast server at Cornell has Cornell (AS# 26) as its host-site and Williams Communication (WCG – AS#7911) as its upstream provider. This difference between the host-site and upstream provider is a quirk of the internal deployment; for most commercial IP Anycast deployments, the host-site is also the upstream provider.

The internal deployment is as yet very small in scale. The biggest hurdle in growing the deployment has been the site-by-site negotiation with upstream providers (ATT, WCG) to clear the advertisement of our anycast prefix. Note that we do not require these upstream providers to actively inject our prefix in the BGP but only propagate the advertisement from the anycast servers onwards. Instead, the approval from the ISPs is only due to the access control ISPs often enforce on the AS numbers and network prefixes they expect to see advertised from their customers. To accelerate this site-by-site deployment, we are currently in the process of deploying anycast servers over NLR [46]. Further details about the internal deployment as well as the requirements for much welcome volunteer sites are available at [44]. While the small size of the internal deployment certainly raises questions regarding the generality of our results, the fact that (in retrospect) most of our results follow intuitive reasoning supports the applicability of our study.

4. METHODOLOGY

A key observation guiding our measurement methodology is that all three external deployments are DNS services. Hence, the anycast servers that are part of these deployments are all DNS nameservers and we can probe them using DNS queries. However, since packets from a client to the anycast address of any given anycast deployment are delivered to one of the servers, a large number of clients are needed to ensure that we are able to reach all the anycast servers. To achieve this we used the King measurement technique [17]. King allows for measurement of latency between any two arbitrary hosts on the Internet by utilizing recursive nameservers near the hosts in question.

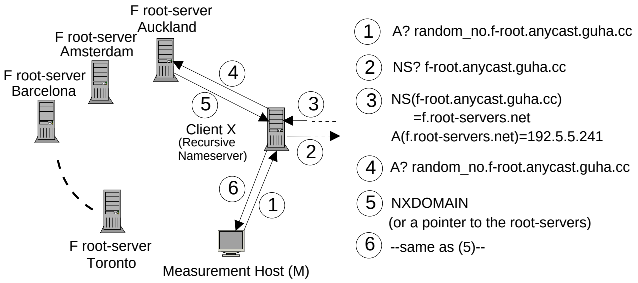

F root-server
Auckland
F root-server
Amsterdam
F root-server
Barcelona
F root-server
Toronto
Client X
(Recursive
Nameserver)
Measurement Host (M)
4
5
3
2
6
1
1 A? random_no.f-root.anycast.guha.cc
2 NS? f-root.anycast.guha.cc
3 NS(f-root.anycast.guha.cc)
=f.root-servers.net
A(f.root-servers.net)=192.5.5.241
4 A? random_no.f-root.anycast.guha.cc
5 NXDOMAIN
(or a pointer to the root-servers)
6 --same as (5)--

Figure 1: Measurement Host M uses DNS queries to direct client X to send packets to the anycast address associated with the F root-server deployment. In the scenario depicted here, packets from X are routed to the anycasted F root-server located at Auckland, Australia. Note that we use a domain under our control (`anycast.guha.cc`) to trick client X into assuming that 192.5.5.241 is the authoritative nameserver for the domain `f-root.anycast.guha.cc`.

Using the same basic idea, we used recursive DNS nameservers in the Internet as *clients* in our study. To this effect, we took an address in each routable prefix in a BGP routing table obtained from Route-Views [49] and determined if any of the nameservers authoritative for the `in-addr.arpa` name associated with the address had recursion enabled. For example, for the prefix 128.84.223.0/24, we determined if any of the nameservers authoritative for the name `1.223.84.128.in-addr.arpa.` had recursion enabled. This approach yielded a list of 23,858 unique recursive nameservers. Table 3 details the geographic distribution of these nameservers. The nameservers belong to 7,566 different ASs. Hence, they provide us a view of the anycast deployments from 7,566 of the 18,391 routeable ASs on the Internet (based on a BGP routing table obtained from Route-Views). The quantity and the spread of these nameservers makes us confident that our measurements closely reflect the behavior of IP Anycast as seen from hosts in the Internet in general.

Note that using a large number of clients yields a more representative picture for all the metrics we measure in this paper. For example, the scale of our study makes our arguments regarding client load distribution across the anycast deployment significantly more representative than (say) just using PlanetLab for measurements. Moreover, our use of widely-dispersed Internet nameservers as clients avoids the bias that would be introduced were we to use PlanetLab hosts as clients. This bias could severely impact the affinity, proximity and load measurements. For example, one of the J-root servers is connected to the GEANT network and hence, can be accessed through Internet2. Consequently, a large fraction of PlanetLab nodes are routed to this server when they probe the J-root anycast address.

Evaluating an IP Anycast deployment from the perspective of any client (say X) in the list requires that we be able to direct X to send packets to the anycast address of the deployment. As suggested by [17], we leveraged the fact that X is a recursive nameserver and hence, is willing to resolve DNS queries on behalf of other hosts. We can thus "trick" client X into sending DNS queries to an anycast address by making the NS record for a domain point to the address in question and querying X for any record in that domain. We used anycast.guha.cc, a domain we own, for this purpose. For example, in case of the F root-server deployment with anycast address 192.5.5.241, we created a domain f-root.anycast.guha.cc with its NS record pointing to 192.5.5.241. As illustrated in figure 1, querying client X for any record in this domain (for example, random_no.f-root.anycast.guha.cc) causes X to resolve the NS record for f-root.anycast.guha.cc (packets (2) and (3)), and then send a query to the anycast address for the F root-server deployment (packet (4)). A minor practical problem with this approach is that a client may be configured to resend query (4) a number of times on seeing that the response (5) is an error. As suggested by [17], we weeded out clients that may resend queries on receiving an error as follows: we directed each client to the anycast address of the internal anycast deployment and logged the DNS queries at each of the servers of the internal deployment to determine the number of times query (4) is sent.

The experiments presented in the paper use this basic technique for various tasks such as determining the particular anycast server accessed by each client and the latency of doing so. For example, to determine the latency from a client X to the anycast address of the F root-server, we:

- Send a recursive DNS query to client X for the NS record for f-root.anycast.guha.cc: this primes client X's cache with the fact that the F root-server anycast address corresponds to the authoritative nameserver for the domain f-root.anycast.guha.cc

- Send an iterative DNS query to client X: since an iterative query is answered by client X based on local information, this provides us with an estimate of the latency for packets {(1),(6)} in figure 1.

- Send a recursive DNS query to client X for the A record for random_no.f-root.anycast.guha.cc: as shown in figure 1, this causes client X to send packets to the F root-server anycast address and provides us with an estimate of the latency for packets {(1), (4), (5), (6)}.

This process is repeated eight times and the difference between the minimum measured latency for packets {(1), (4), (5), (6)} and {(1), (6)} is used as an estimate of the round-trip anycast latency from client X to the F root-server.

5. PROXIMITY

The value of anycast as a server selection primitive is in part due to its ability to find close servers. With IP Anycast, packets destined to an anycast address are routed to the server closest to the client in terms of the metrics used by the underlying routing protocol. For inter-domain IP Anycast, it is the BGP decision process (including routing policies) at the various ASs that governs the anycast server accessed by each client. This implies that anycast packets from clients may not be delivered to servers that are close in terms of *latency* and recent studies [4,32] have, in fact, indicated that IP Anycast offers poor latency-based *proximity*. In this section we use latency measurements from our $\approx$ 20,000 clients to show that this is indeed the case for existing anycast deployments. However, we also argue that poor latency can be avoided through a planned deployment.

Methodology: To determine the quality of proximity offered by an IP Anycast deployment to a given client, we need to determine the following latencies:

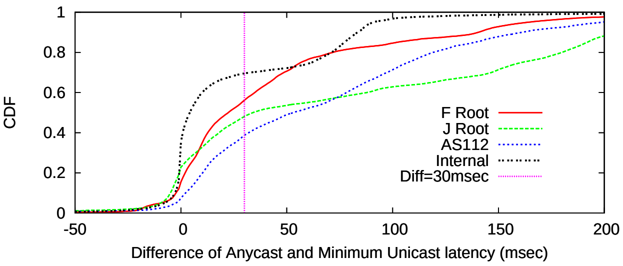


Figure 2: CDF for the difference between the anycast and minimum unicast latency for the external and the internal deployments.

- *Unicast Latency* to all anycast servers - here, unicast latency to an anycast server is the latency from the client to the unicast address of the server. Given that each client is a recursive nameserver, the King approach for determining latencies between two hosts applies here directly.

- *Anycast Latency* for the client or the latency from the client to the anycast address of the deployment. The procedure for determining the anycast latency for a client was described in the previous section.

We define *stretch-factor* as the difference between the anycast latency and the minimum unicast latency for a client. The stretch factor thus represents the quality of latency-based proximity an anycast deployment offers a client. We determined the stretch factor for each client in our list for the external and internal anycast deployments.

Results: Figure 2 shows the CDF for the stretch factor for all the clients. We see that for all four anycast deployments, a fair fraction of clients are not routed to the server closest to them.[5] For example, the number of clients that are routed to a server that is more than 30msec farther away from the closest server ranges from 31% (for the internal deployment) to 61% (for the AS112 deployment). Similarly, in case of the J root-server deployment, 40% of the clients incur a stretch-factor of more than 100msec.

The internal and the external anycast deployments have been deployed such that most of the anycast servers have different upstream providers. For example, in case of the internal deployment, the anycast servers have three different upstream providers. We believe that it is this deployment model that makes these services vulnerable to the fact that Internet routing is not based on latency as reflected by the inefficacy of IP Anycast in selecting closeby anycast servers for these deployments.

The following anecdotal scenario serves to illustrate the causes for the poor proximity. We use the example of a publicly available traceroute-server at UC-Berkeley (net.berkeley.edu) acting as a client trying to access the internal anycast deployment. Anycast packets from this client are routed to the server at Cornell (latency=87msec) instead of the close-by

[5] The poor selection in case of F root-server deployment can be attributed to their use of hierarchical anycast [1,45] In effect, only 2 of the 27 F root-servers advertise the F-root anycast prefix globally and so, it is not fair to compare the anycast latency to the minimum unicast latency across all the 27 servers. However, this is not the case for the other deployments we measured.

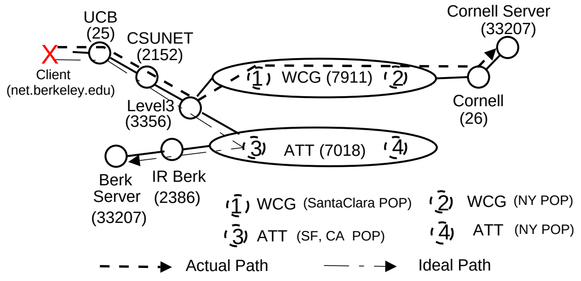


Figure 3: The relevant AS-level connectivity (with corresponding AS numbers in parenthesis) between the client (`net.berkeley.edu`) in UC-Berkeley and the internal anycast deployment servers at Cornell and Berkeley. Level3 receives at least two different advertisements for the internal deployment anycast prefix with the following AS-PATHs: [7911, 26, 33207] and [7018, 2386, 33207].

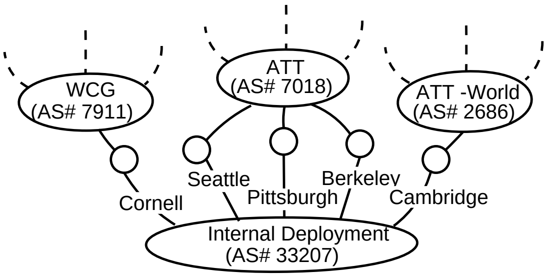


Figure 4: AS-level connectivity for the anycasted internal deployment – from the point of inter-domain routing, AS# 33207 is a multihomed stub AS.

server at Berkeley (latency=9msec). Figure 3 shows the relevant AS-level connectivity with the relevant POPs of ATT (upstream provider for the server at Berkeley) and WCG (upstream provider for the server at Cornell). We used BGP looking-glass servers to determine that Level3 has at least two paths for the internal deployment's anycast prefix: one through WCG's Santa Clara POP with AS-PATH=[7911, 26, 33207] and the other through ATT's San Francisco POP with AS-PATH=[7018, 2386, 33207]. The Level3 routers in the area choose the first path as the best path and hence, anycast packets from the client are delivered to the anycast server at Cornell. This path is labeled as "Actual Path" in the figure.

This example points to the crux of why Internet routing yields poor proximity for the measured anycast deployments. From the point of view of inter-domain routing, an anycasted AS is equivalent to a multi-homed stub AS (see figure 4). However, anycasting introduces multi-homing scenarios which differ significantly from normal multi-homing scenarios. In typical multihoming, multiple peerings of the multihomed stub AS are in the same geographical area. As a consequence, selection of paths from clients to the multihomed AS based on ISP policies and AS-PATH length leads to acceptable performance. On the other hand, for an anycasted AS, the multiple peerings are geographically dispersed and this is not accounted for in the existing inter-domain routing set-up. In the example above, Level3 receives two paths to the anycasted internal deployment of equal AS-PATH length and is not aware of the actual physical difference between the length of these paths. Consequently, there is a good chance that Level3 may choose a route that causes the anycast packets to be routed to a distant anycast server. In other words, current route selection mechanisms have a much higher chance of making an unsuitable choice when selecting paths to an anycasted AS.

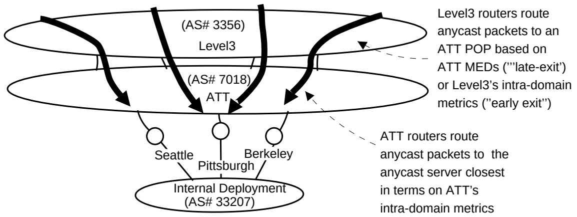


Figure 5: Shown here is a deployment with ATT as the common upstream provider – ASs beyond ATT route the anycast packets to ATT's network. Here, Level3's routers route anycast packets to a close-by ATT POP which then routes the packets to the closest anycast server.

Although negative, the importance of this observation cannot be overemphasized. It brings to light that while the routing protocols used to choose paths to unicast destinations work naturally for anycast destinations too, the metrics used for routing decisions can lead to a poor choice of anycast server in terms of latency. While changing routing protocols to differentiate between anycast and unicast prefixes would be one possible approach to address this problem, a more practical approach would be to plan the deployment of the anycast servers so as to account for inter-domain route selection.

We hypothesize that deploying the anycast servers such that all of them have the same upstream provider and the servers are spread across the provider is one such deployment approach – this approach was briefly mentioned in [4] but was not explored in any detail. This approach is based on two key observations. First, an ISP that is an upstream provider for some of the anycast servers routes incoming anycast traffic to the server closest among them – this is a consequence of the fact intra-domain traffic engineering is mostly consistent with latency sensitive routing [34]. For example, in case of the internal deployment, ATT is an upstream provider for three of the five anycast servers. Routers in ATT's network receive routes of equal AS-PATH length and equal preference from each of these anycast servers. Hence, the BGP decision process causes incoming anycast packets at any ATT POP to be routed to the server that is closest in terms of the intra-domain metrics used by ATT. As the measurements presented next show, this server is also closest in terms of latency in a large majority of cases.

Second, such a deployment de-couples route selection at the common upstream provider from the selection at ASs beyond it. Due to reasons detailed above, the common upstream provider delivers incoming anycast packets to the closest server. Any ASs farther away from the anycast server sites than the upstream provider only need to select among the possible routes to the upstream provider. Figure 5 illustrates this. Of these ASs, the ones that use an "early-

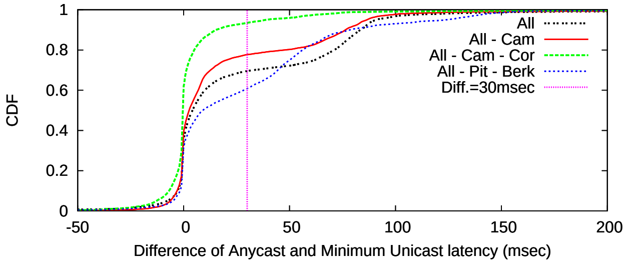


Figure 6: CDF for the difference between the anycast and minimum unicast latency for various subsets of the internal deployment. Here, All - x implies measurements with server x switched off.

exit" (or hot-potato routing) routing policy would route anycast packets to the closest POP of the common upstream provider. Alternatively, ASs that use a "late-exit" (or cold-potato routing) routing policy would honor the MEDs of the upstream provider and route anycast packets to the upstream provider POP that is most suitable for delivery to the closest of the deployed anycast servers. While it is possible that an AS may choose overly long paths to the upstream provider, the prevalence of the two aforementioned policies amongst ASs leads us to believe that this would not be the norm and our measurements confirm this. Its also important to note that any overhead arising due to the choices made at this step of the selection process also apply to the inter-domain routing in general and are not specific to anycast per se.

Hence, in case of the internal deployment, we would like to ensure that instead of five servers with three different upstream providers, all the servers should have the same provider. As a matter of fact, the subset of the internal deployment comprising of the three servers at Berkeley, Pittsburgh and Seattle conforms to our deployment proposal. These servers have the same upstream provider (ATT) and are geographically spread out. With this three server deployment, all anycast packets would be routed to the ATT network and then delivered by ATT to the (closest) anycast server. For instance, if the sample client (`net.berkeley.edu`) presented above accessed this deployment, it would be routed to the server at Berkeley. This is because, in figure 3, Level3 would not receive an advertisement for the anycast prefix from WCG. Instead, Level3's routers would route the client's packets to the ATT POP at San Francisco. The routers in ATT's San Francisco POP would in turn route these packets to the nearest anycast server. Thus, the anycast packets would be routed to the server at Berkeley and not at Seattle or Pittsburgh. We validated this by stopping the advertisement of the anycast prefix from the Cornell and the Cambridge server to remove them from the anycast deployment and observing that the anycasted packets from the sample client were delivered to the anycast server at Berkeley. This path labeled as "Ideal Path" in figure 3.

To validate this hypothesis, we repeated the stretch factor measurements for our list of clients with only subsets of the internal deployment operational. Figure 6 shows the results from these measurements. As can be seen, an anycast deployment comprising of just the three servers at Berkeley, Pittsburgh and Seattle (labeled as "All - Cam - Cor" in the figure) yields good proximity with just 5% of the clients incurring a stretch factor of more than 30msec. Note that it may seem that this improvement in the stretch factor for clients is due to the fact that the choice of servers has reduced from five to three. To account for this we also measured the proximity offered by a deployment with the three servers at Seattle, Cornell and Cambridge. All these servers have a different upstream provider and as can be seen from the figure (curve labeled as "All - Pit - Berk"), this yields poor proximity too. These results show that an anycast deployment with all servers having the same upstream provider does provide good latency-based proximity to clients.

As a matter of fact, this result can be generalized. It is possible to have anycast deployments with multiple upstream providers for the anycast servers. However, all the upstream providers should have reasonable global geographic spread (in essence, tier-1 ISPs with a global network) and for each upstream provider, there must be a sufficient number of servers to cover the geographical spread of the provider. In such a set-up, whenever any of the upstream providers receives an anycast packet, there is a close-by anycast server that the packet can be routed to. ASs beyond these upstream providers would choose to route the anycast packets to one of the upstream providers and since all the providers are well covered by the anycast servers, this choice does not have a lot of bearing on the anycast path length. Due to reasons described earlier, each AS beyond the upstream providers for the anycast deployment would most likely route the packets to a suitable POP of the upstream provider it chooses. Thus, a deployment according to this model would offer good latency based proximity to clients. The small size of the existing internal deployment does not allow us to validate this claim; however, this is something that we aim to address in future work.

6. FAILOVER TIME

Inter-domain IP Anycast involves each site advertising the anycast prefix into BGP. Consequently, when an anycast server fails, the process by which clients using the failed server are re-routed to other operational servers is tied to BGP convergence. Past studies have shown that failures at multi-homed end sites can lead to a convergence process that may last for minutes [23]. Such a slow failover, if it applies to IP Anycast to, would not bode well for a number proposed uses of IP Anycast. Hence we decided to measure the failover rate for the internal anycast deployment. To the best of our knowledge, this represents the first attempt to study the failure-mode behavior of IP Anycast.

Methodology: In order to determine the rate at which clients failover when an anycast server fails, we need to be able to determine the specific anycast server that each client is routed to. To this effect, we configured the anycast servers in the internal deployment to act as authoritative nameservers for a domain under our control (`internal.anycast.guha.cc`) and to respond to TXT-type DNS queries for this domain with a location-specific string. So for example, a TXT-type DNS query for this domain from a client whose anycast packets are routed to the anycast server at Cornell will receive "Cornell" as the response. This, when combined with the ability to direct clients to send DNS queries to the anycast address of the internal deployment, allows us to determine the anycast server being accessed by all clients in our list. Figure 7 illustrates this process for one client.

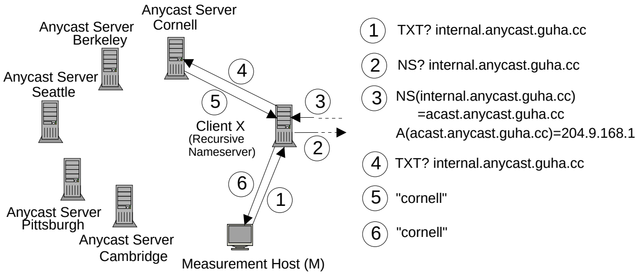


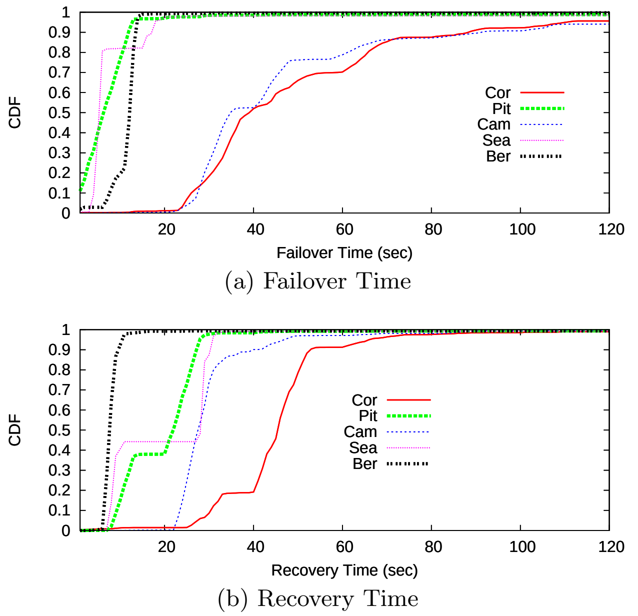
Figure 7: TXT-type DNS queries from client X to the internal deployment anycast address (204.9.168.1) are routed to the server at Cornell which responds with a location-specific string ("cornell")

The internal deployment servers have been configured to respond to TXT-type queries with a TTL value of 0. This implies that clients cannot cache the response from the anycast server and hence, need to send packets to the deployment's anycast address each time they are queried. Also, note that since we don't have any way to determine the anycast server accessed by clients for the external deployments[6], the measurements in this and the following sections are restricted to the internal deployment.

Given this, we determined the impact of the failure of each server in the internal deployment on clients being routed to that server. We induced failures at individual servers by tearing down their BGP peerings leading to a BGP withdrawal being generated for the anycast prefix. Concurrently, we sent the aforementioned TXT queries to the anycast address through the clients that were being routed to the failed server at a rate of once every five seconds for three minutes and at a rate of once per minute for the next fifty-seven minutes leading to a total probe period of one hour. For each such client, we determined the time it takes for the client to failover and "settle" at a different anycast server. This is referred to as the *failover time* for the client. Note that during the convergence process, a client may be temporarily routed to a server different from the server it is finally settles on. For example, a client accessing the Cornell server before it failed may be temporarily routed to the server at Cambridge before being routed to the server at Pittsburgh for good. The time between the failure and the first query that is routed to the Pittsburgh is the client's failover time.

We would also like to clarify that unlike affinity (as discussed in the next section), fast failover is not very relevant with regards to the feasibility of running connection oriented services on top of IP Anycast. In a vast majority of the scenarios, the failure of a server would also break all client connections irrespective of how fast the failover process is. Instead, the failover time characterizes the time after a failure for which clients using the failed server cannot utilize the anycast service.

Similarly, we also restarted the failed servers by re-establishing their BGP peerings and determined the time it takes for clients that were originally using the server in question to be routed back to it again. This time between the re-establishment of the peering and the re-routing of a client to the server is referred to as the *recovery time* for the client. For each server in the internal deployment, we induced failures and restarts and measured the failover and the recovery time for the clients using the server. These experiments were repeated 5 times each – we avoided more runs because the experiments impose a heavy query load on nameservers that we don't own.



(a) Failover Time

(b) Recovery Time

Figure 8: The failover and recovery times for the servers in the internal deployment.

Results: Figure 8 shows the CDF for the failover and recovery times corresponding to each server in the deployment (the time axis has been shortened in the interest of clarity). The servers at Cornell and Cambridge have similar failure mode behavior with a median failover time of ≈35 seconds and a 95^{th} percentile of ≈120 seconds. The peaks around 30 and 60 seconds in these curves can be attributed to the BGP MinRouteAdverTimer which governs the time between updates for the same prefix to the BGP same peer. The value of this timer defaults to 30 seconds on many routers [25]. On the other hand, the servers at Berkeley, Pittsburgh and Seattle have a median failover time ranging from 7 to 12 seconds with a 95^{th} percentile of 14 to 18 seconds. The trends in the recovery times are less obvious and we don't completely understand these results.

In case of the server at Cornell, there is no other anycast server with WCG as an upstream provider. Consequently, the failure of this server causes clients using the server to be re-routed to servers with other upstream providers. This also implies that the BGP updates resulting from the server failure need to be propagated beyond WCG's network. As a result, the failover process involves a number of ASs, is impacted by the various BGP timers and overall, is affected by delayed routing convergence resulting in the large failover time. A similar explanation applies to the failure of the

[6] Some of the anycasted DNS root-server deployments do allow users to query them for the particular server the user is being routed to [45]. However, these queries have been chosen such that they cannot be generated through recursive nameservers, probably to avoid the possibility of this being used for a DNS-amplification attack on the root-servers [47].

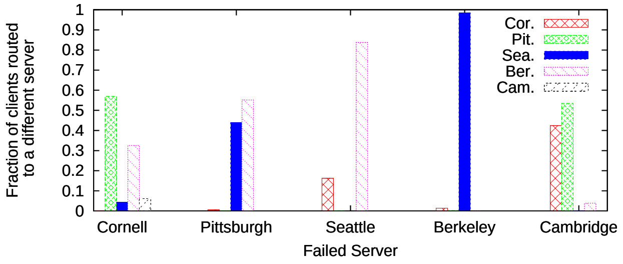


Figure 9: As a server fails, clients that were being routed to the server are now routed to other operational servers. The Y-axis shows the fraction of clients that failover to each other operational server when the particular server on the X-axis fails.

server at Cambridge.

On the other hand, the other three servers at Berkeley, Pittsburgh and Seattle offer much faster failover. This is an outcome of the fact that these servers have the same upstream provider (ATT). Thus, when one of the servers fails, clients accessing that server are routed to the one of the other two servers. To verify this, we determined the fraction of clients that go to other operational servers when a particular server fails – this is plotted in figure 9. As can be seen, in case of a failure at Berkeley, Seattle and Pittsburgh, most of the clients are re-routed to one of the other two operational servers from the same group. This implies that when one of these servers fails, the BGP convergence process is restricted mostly to ATT's network resulting in faster failover.

The distribution of clients when a server fails (figure 9) can also be used to explain some of the trends in figure 8(a). For example, almost all the clients accessing the Berkeley server when it fails are routed to the server at Seattle. As a result, the failover time for the Berkeley server is almost the same for all clients accessing it. On the other hand, a small fraction of clients accessing the Seattle server when it fails are routed to the server at Cornell.[7] This explains the inflection point in the failover time for the Seattle server showing that a small fraction of the clients take much more time to failover than the rest.

These results suggest that in an IP Anycast deployment with a number of anycast servers per upstream provider (as proposed in the previous section), the failure of an anycast server would cause clients to be routed to one of the other servers with the same upstream provider. Hence, the proposed deployment model is conducive to fast failover with a large majority of clients being rerouted to another server within 20 seconds. Reports that many commercial DNS-based anycast deployments aim for similar sub-minute failover times (by using a TTL values between 10-20 seconds [22,33]) lead us to conclude the failover rate of a planned IP Anycast deployment should suffice for almost all anycast applications.

On a broader note, our study shows that while results from previous studies reporting slow BGP convergence [23] do apply to IP Anycast deployments in general, an IP Anycast deployment can be planned so as to decouple anycast failover from delayed routing convergence. In effect, this addresses the long held belief that IP Anycast is bound to provide very slow failover, for example, the possibility of server failures causing outages of five or more minutes [42]. As a matter of fact, the clustered deployment model (as described in section 3) used by most commercial IP Anycast deployments was primarily motivated by the need to decouple host failure from BGP events. While we agree that using clustered hosts at each anycast server site is a necessary part of the IP Anycast deployment picture, an IP Anycast deployment conforming to our deployment proposal ensures that even when an entire server site fails, clients do not have to wait an inordinate amount of time for failover.

7. AFFINITY

The fact that IP Anycast is a network layer service implies that two consecutive anycast packets from a single client need not be routed to the same server. For example, in case of the internal deployment, it is possible that a client whose anycast packets were being routed to the server at Cornell suddenly gets routed to the server at Cambridge. Such an occurrence, hereon referred to as a *flap*, would break any higher-layer connections (for example, TCP connections) that may exist between the client and the anycast service. Hence, determining the affinity offered by IP Anycast is important for characterizing its the impact on stateful services being run on top.

As mentioned earlier, past studies have painted a contradictory picture of IP Anycast affinity. A study using a few (<200) PlanetLab nodes as vantage points [4] claimed that flaps are relatively rare; for example, they reported a median inter-flap duration when probing the F root-servers to be more than 10 days. Similarly, operators of the anycasted K root-server [12] found that their IP Anycast deployment offers very good affinity. On the other hand, a study based on a few (<200) volunteer and PlanetLab nodes [8,31] and anecdotal evidence from the anycasted J root-server [6] support claims to the contrary. For example, Boothe et. al. [8] found the median inter-flap duration to be 1.4 hours for PlanetLab nodes and 3 hours for their volunteer nodes. As a matter of fact, IP Anycast affinity and its suitability for stateful services has been passionately debated on many mailing lists; a summary of some these discussions can be found at [43]. However, none of the aforementioned studies have attempted to delve into the reasons behind the routing flaps (few or many) observed by them. In this section, we present a detailed analysis of the affinity offered by the internal anycast deployment as determined through active probing from our clients.

Methodology: The TXT-record based querying described in section 6 allows us to determine the particular anycast server that anycast packets from a given client are routed to. Using this, we can periodically query a client in order to capture the anycast flaps experienced by it. For these experiments, we randomly chose 5200 clients from our list of clients and determined the number of flaps they experience by querying them at a rate of once per minute for a period of 17 days.

Results: Figure 10 shows a CDF for the number of clients observing the respective number of flaps. The figure shows

[7] The failure of the Seattle server probably causes ATT to modify the MEDs on the anycast prefix advertisements it propagates to its peers, some of whom then choose to route anycast packets to other servers that don't have ATT as an upstream provider. This explains why some clients are routed to Cornell when the server at Seattle fails.

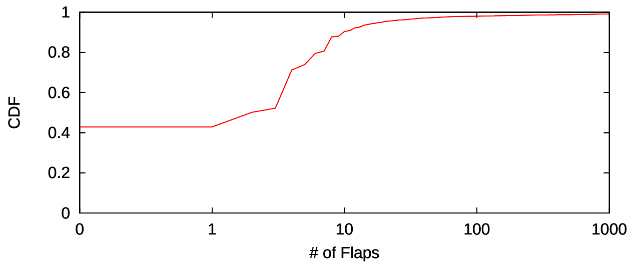


Figure 10: Affinity measurements for our anycast deployment – the measurements involve 5277 nameservers as vantage points and span a period of 17 days.

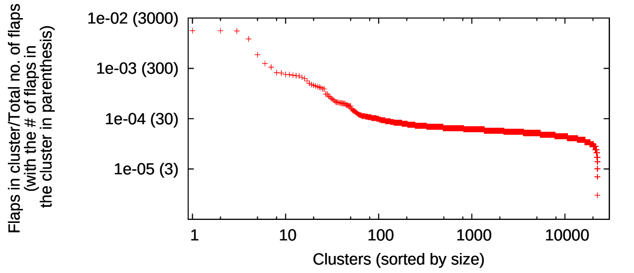


Figure 11: Clustered flaps and their contribution towards the total number of flaps – there are a small number of large clusters but a majority of the flaps belong to very small sized clusters.

that ∼40% of the clients do not observe any flap, ∼95% of the nodes observe less than a flap per day (less than 17 flaps in total) and ∼99% of the clients observe less than 10 flaps per day. Overall, the trace comprises of 290,814 flaps of which 266,967 (∼92% of the total) were observed by just 58 (∼1%) clients. Hence, apart from this very small fraction of nodes, the internal anycast deployment seems to provide very good affinity.

These anycast flaps can occur due to a variety of events ranging from link, peering or server failures to ASs load-balancing the anycast traffic across multiple links. Typically, the impact of a given event depends on its location: events near an anycast server or in a core ISP would cause a number of clients to flap while the impact of an event close to a client site would be restricted to a small number of clients. Given this observation, we proceeded to use temporal clustering to construct events out of the flaps seen in our trace. The idea here is to cluster flaps that occur close by in time on the assumption that they are probably due to the same routing event. Hence, we clustered the flaps observed at all the clients such that flaps within 10 seconds of each other were in the same cluster. Since flaps in the cluster are assumed to be due to the same routing event, we limited the maximum cluster size to 180 seconds, i.e. no two flaps more than 180 seconds apart can be in the same cluster. This is similar to the BGP update clustering used in [14]. The clustering results presented here are not very sensitive to the maximum cluster size – clustering with a maximum cluster size of 120 and 240 seconds yielded similar results. Using this approach, the 290,814 flaps yielded 22,319 clusters with the largest cluster containing 1,634 flaps.

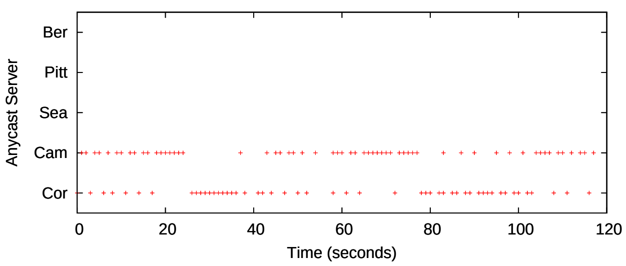
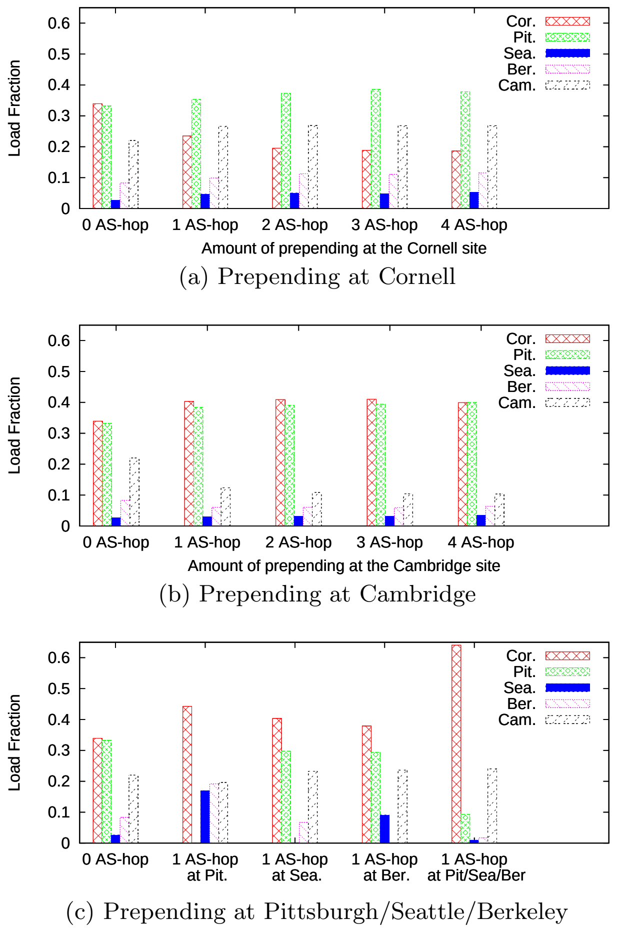


Figure 12: Probes at a rate of once per second from an unstable client. Each plotted point in the figure represents a probe and shows the server it is routed to – as can be seen, the client flaps very frequently between the anycast servers at Cornell and Cambridge.

Figure 11 shows the distribution of the fraction of the total number of flaps that occur in the cluster. The figure has a small number of moderate-to-large clusters and these correspond to infrequent BGP events near the servers or in core ISPs. This further buttresses our argument that IP Anycast does not have any harmful interactions with inter-domain routing. More importantly, the figure has a long tail depicting a very large number of very small clusters. These correspond to events near clients. As a matter of fact, a large majority of these clusters comprise of flaps seen at the highly unstable clients. This points to the fact that even the very small fraction of clients that observe poor affinity do so due to events that are close to them.

Further, the frequency at which the unstable clients observe flaps leads us to believe that these clients are multi-homed and are using some kind of dynamic load-balancing across multiple upstream providers. For example, the client which observed the most flaps belongs to AS# 15710 which, in turn, has two upstream providers – AS# 3356 (Level3) and AS# 8928 (INTEROUTE). This client observed almost continuous flaps between the server at Cornell and Cambridge in our trace. To investigate further, we probed this client at a rate of once per second for a period of two hours. Figure 12 shows the server each probe is routed to (for clarity, we show the probes only for a two minute duration – the rest of the trace is similar). The figure shows that the client experiences very frequent flaps. In many cases, consecutive probes are routed to different servers. Given that BGP events occur at much coarser time scale, this high frequency of flaps suggests dynamic load balancing by the client.

To validate our conjecture, we used the view of inter-domain routing available through Route-Views [49] and CIDR-Report [41] to determine the AS-level connectivity of the unstable clients. The 58 unstable clients belong to 47 distinct ASs and at least 42 of these have multiple upstream ASs. Since we do not have control over these clients, and hence cannot determine if the client ASs are indeed load-balancing across their upstream ASs, we conducted an e-mail survey of the client ASs to determine if this is indeed the case. The survey yielded just five responses, though all the five ASs claimed to be using some form of load balancing across their provider. While the exact set-up of these clients begs further investigation, all the evidence at hand leads us to conclude that dynamic load-balancing by clients is the root-cause of the poor affinity observed by them.



(a) Prepending at Cornell

(b) Prepending at Cambridge

(c) Prepending at Pittsburgh/Seattle/Berkeley

Figure 13: Load on the anycast sites of anycast deployment in the default case and with various kinds of AS path prepending at the sites. Here, Load Fraction for a site is the ratio of the number of clients using the site to the total number of clients (≈20000).

8. CLIENT LOAD DISTRIBUTION

Clients access an IP Anycast deployment simply by sending packets to its anycast address and it is the routing infrastructure that is responsible for delivering the packets to the one of the servers. Consequently, anycast operators don't have any control over the number of clients that each server handles. In this section, we study the distribution of client load across the internal deployment and evaluate means by which this can be controlled.

We used the TXT-record based querying described in section 6 to determine the distribution of all clients in our list across the servers in the internal deployment. The set of bars corresponding to "0 AS-hop" in figure 13(a) shows this distribution. As can be seen, in the default set-up, most of the clients (≈90%) are routed to the server at Cornell, Cambridge or Pittsburgh. This result is consistent with the uneven load distribution of clients across the anycasted J root-servers [6] and K root-servers [12]. Hence, IP Anycast, by itself, does not balance client load across the anycast servers.

It is interesting to note that the distribution of clients is skewed even amongst the servers that have ATT as their upstream since a lot more clients are routed to Pittsburgh than to Seattle and Berkeley. On the other hand, the proximity measurements in section 5 showed that anycast packets coming into the ATT network are routed to the closest of these three anycast servers. This would imply that most of the clients in our list that are routed to ATT are closer to the server at Pittsburgh than to the other two servers. In effect, this brings out the implicit trade-off between proximity and load-balance. An anycast deployment which offers optimal latency-based selection of the anycast server is unlikely to achieve an even distribution of clients across the servers in the deployment.

Given the uneven load distribution, we would like to investigate if anycast operators can use routing advertisement manipulations at individual server sites to control the number of clients routed to them. In the rest of this section we evaluate the effectiveness of AS-PATH prepending as a means of controlling the client load on anycast servers. AS-PATH prepending at a server involves increasing the length of the AS-PATH in the BGP advertisement that the server propagates and hence, should wean some of the clients away from it. For example, when "1 AS-hop" prepending is used at the Cornell site, the AS-PATH seen by the Cornell server's peer is [33207 33207].[8] Similarly, "2 AS-hop" prepending implies that the Cornell server advertises the internal deployment anycast prefix with the AS-PATH as [33207 33207 33207].

Figure 13(a) shows the distribution of client load across the anycast servers when path prepending is used at the Cornell server. As can be seen, prepending 1 AS-hop causes the fraction of clients being routed to the Cornell server to reduce from 34% to 23%. However, the reduction in load tapers off beyond this with ≈18% of the clients being routed to the Cornell server irrespective of the amount of prepending used. This can be explained in terms of typical ISP policies and the BGP decision process: the ISP policy (expressed as weights and local preferences) has higher priority than the AS-PATH length in the BGP decision process. Thus, ASs that choose to use the Cornell server as dictated by their routing policies are oblivious to the amount of prepending being done and the impact of path prepending soon runs into diminishing returns. Figure 13(b) shows the variation of client load with path prepending at the Cambridge server. These results follow a pattern similar to what is described above with the client load on the Cambridge server reducing and then tapering off.

Figure 13(c) shows the distribution of client load with path prepending at Pittsburgh, Seattle and Berkeley. These are the three anycast sites with the same upstream provider (ATT). The figure shows that prepending the AS-PATH of the advertisement at the Pittsburgh server causes the number of clients routed to Pittsburgh to drop to zero. Since the servers at Seattle and Berkeley also have ATT as the upstream provider, using AS-PATH prepending at Pittsburgh causes routers in ATT's network to prefer both the Berkeley and the Seattle servers to the Pittsburgh server. Hence, all anycast traffic reaching ATT's network is split between the Seattle and Berkeley servers only. Results for 1-AS hop prepending at Seattle and Berkeley are analogous.

These results imply that if some servers in an anycast

[8] As mentioned earlier, 33207 is the origin AS used for the internal deployment.

deployment have the same upstream provider, all of them need to prepend the AS-PATH in their advertisements in order to divert clients away from them. In the internal deployment, diverting clients away from the three servers with ATT as their upstream ISP (Pittsburgh, Seattle and Berkeley) would require all of them to use path prepending. The final set of bars in figure 13(c) shows the load distribution across our deployment in such a scenario.

Hence, while AS-PATH prepending can be used to manipulate load across servers with different providers, it is not effective for manipulating load within the set of servers with the same provider. Alternatively, in an IP Anycast service deployed according to the model presented in section 5, AS-PATH prepending can only be used for balancing load across between groups of servers that have the same upstream ISP. For example, consider a deployment with a few servers having ATT as their upstream provider and the rest having WCG as their upstream provider. The servers with ATT as their upstream need to use AS-PATH prepending together to divert clients towards the servers with WCG as their upstream. For balancing the client load across servers with the same upstream provider, the servers need some sort of traffic engineering arrangement with their common provider. For instance, many ISPs allow their customers to manipulate incoming traffic through the use of specific BGP community attributes in their routing advertisements [10,50]. Anycast servers with the same provider can thus use such mechanisms to coarsely control the number of clients routed to them. We are currently in the process of talks with ISPs who would allow us to host anycast servers and experiment with such mechanisms.

Finally, note that these mechanisms provide operators with a coarse-grained control over the distribution of clients across server sites (or groups of sites). For instance, this could be used by anycast operators in the face of a DoS attack on the deployment to redistribute traffic away from server sites under strain. Beyond this, anycast operators can use load balancing devices at server sites for a fine grained control over the distribution of clients being served by the site across the hosts that are part of the site. As a matter of fact, current commercial IP Anycast deployments use such mechanisms for balancing the number of clients served by individual cluster hosts at each site.

9. DISCUSSION

Section 2 described previous IP Anycast measurement studies. Here we discuss other research efforts relevant to IP Anycast and relate our study to this broader context.

In addition to its implementation at the network layer, anycast can also be implemented at the application layer. *Application-layer anycast* provides a one-to-any service that maps a high-level name, such as a DNS name, into one of multiple servers, returning the selected server's IP address to the client. Such an approach offers a number of advantages over IP Anycast: it is easier to deploy, offers fine-grained control over the load on the servers and can provide very fast failover to clients. And indeed, these advantages have led to the widespread adoption of application layer anycast as a service discovery primitive. For example, commercial CDNs [38] use DNS-based redirection (in combination with URL-rewriting) to direct clients to an appropriate server. Related proposals in the academic community include, but are not restricted to, [13,15,35–37].

In spite of these advantages, application-layer anycast is not a panacea. The fact that IP Anycast operates at the network layer implies that it is the only form of anycast that can be used by low-level protocols; *e.g.*, the use of anycast in IPv4-to-IPv6 transition [19]. As importantly, operating at the network layer gives IP Anycast a "ground level" resilience not easily achieved by application-layer anycast – *e.g.*, using DNS-based redirection to achieve resilience across a group of web servers requires first that the DNS servers themselves be available. It is this that makes IP Anycast particularly well suited for replicating critical infrastructures such as the DNS. For applications that do use IP Anycast, our deployment proposal can be used to build an anycast service that offers good proximity, fast failover, and control over the distribution of client load.

While IP Anycast functionality is available even today, it scales poorly in the number of anycast groups. Recognizing its many advantages, GIA [20] and PIAS [4] seek to make IP Anycast more broadly usable and propose solutions to improve the scalability of IP Anycast. Our study focusses on the basic effectiveness of IP Anycast, not its scalability, and our results are relevant to the performance one might expect of an IP Anycast service whether implemented as a proxy-based service (PIAS) or a more scalable IP-layer implementation (GIA).

Past studies have analyzed the use of AS-PATH prepending as a traffic engineering tool for multi-homed stub sites [24,29] and have proposed automated mechanisms for this [9]. However, we are not aware of any studies analyzing the use of AS-PATH prepending as a mechanism for controlling the distribution of client load across anycasted servers. Similarly, we conjecture other traffic engineering techniques can also be used as a load distribution mechanism by anycast operators. For example, some of the F root-servers use the BGP `no-export` attribute as part of their anycast advertisements. This restricts the scope of the advertisement emanating from the server and hence reduces the number of clients served by it.

10. CONCLUSION

This paper presented the results of a detailed measurement study of IP Anycast. Our study differs from previous efforts on two fronts. First, we evaluate IP Anycast deployments from a large number (>20,000) number of client vantage points. Second, we deploy our own IP Anycast service to perform controlled experiments that, for example, allow us to study the failure-mode behavior of IP Anycast. Our findings include:

1. IP Anycast, by itself, does not route clients to servers that are close in terms of latency.

2. IP Anycast is affected by delayed routing convergence and may be slow in re-routing clients in the face of server failures.

3. IP Anycast offers good affinity to all clients with the exception of a a small fraction that explicitly load balance traffic across multiple upstream providers. *i.e.*, we find IP Anycast does not interact poorly with inter-domain routing and hence should not significantly impact stateful services.

4. IP Anycast services experience a skewed distribution of client load across the anycast servers.

Based on these measurements, we hypothesize that an IP Anycast deployment with a single upstream provider and with servers spread across this provider would offer good latency-based proximity. Our evaluation shows that this holds in our internal anycast deployment. Further, we generalize this model and argue that for good proximity in an IP Anycast deployment with multiple upstream providers, each major upstream provider should be geographically spread and well covered by anycast servers. Our evaluation further suggests that such a deployment model provides fast failover to clients. However, an evaluation of this approach over larger deployments and fully characterizing the proximity within such a model is a topic of future work. We also evaluate the effectiveness of AS-PATH prepending to manipulate the distribution of client load across servers and find that it can be used for controlling the number of clients routed to groups of anycast servers with the same upstream provider. Overall, we find that an IP Anycast service can be deployed to offer good proximity and fast failover to clients while allowing for coarse-grained control over the distribution of client load across the deployment.

Our study is limited in several aspects. First, the size of the internal deployment raises concerns regarding the generality of our results and benefits of our proposed deployment model for larger deployments. Second, our internal deployment setup does not allow us to evaluate the effectiveness of certain other traffic engineering techniques for controlling client load. Finally, the use of external DNS nameservers as clients in our study restricted the amount and rate of probing that could be done. For the same reason, our conjectures regarding load balancing at clients had to be verified using heuristics and survey data. Nonetheless, we hope the measurement techniques presented here can serve in the large-scale evaluation of experimental anycast deployments along the lines presented in the paper. We are currently pursuing the addition of anycast sites to our deployment which would allow us to address some of the above limitations.

Note

The data sets used in this paper are available at `http://pias.gforge.cis.cornell.edu/measure.php`.

Acknowledgements

We would like to thank Dan Eckstrom at CIT and James Gurganus and the rest of the IT team at Intel-Research for their efforts towards the internal anycast deployment. We are also grateful to Phil Buonadonna and Timothy Roscoe for their help with the deployment. This material is based upon work supported by the National Science Foundation under Grant No. 0338750, by the AFOSR under Award No. F49620-02-1-0233, FA8750-05-2-0128, PA8750-05-C-0268 and by a grant from Cisco systems.

11. REFERENCES

[1] Abley, J. Hierarchical Anycast for Global Service Distribution. ISC Technical Note ISC-TN-2003-1 `www.isc.org/tn/isc-tn-2003-1.html`.

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

[3] Ballani, H., Ermolinskiy, A., Ratnasamy, S., and Francis, P. An Experiment in Deploying Next Generation Network Protocols. Tech. rep., Cornell University Technical Report, 2006.

[4] Ballani, H., and Francis, P. Towards a global IP Anycast service. In *Proc. of ACM SIGCOMM* (August 2005).

[5] Ballani, H., and Francis, P. Understanding IP Anycast. Tech. rep., Cornell University Technical Report, 2006. `http://pias.gforge.cis.cornell.edu/unpub/any-measure.pdf`.

[6] Barber, P., Larson, M., Kosters, M., and Toscano, P. Life and Times of J-Root. NANOG 32 meeting, October 2004. `http://www.nanog.org/mtg-0410/kosters.html`.

[7] Basturk, E., Haas, R., Engel, R., Kandlur, D., Peris, V., and Saha, D. Using IP Anycast For Load Distribution And Server Location. In *Proc. of IEEE Globecom Global Internet Mini Conference* (1998).

[8] Boothe, P., and Bush, R. Anycast Measurements Used to Highlight Routing Instabilities. NANOG 34 meeting, May 2005. `http://www.nanog.org/mtg-0505/boothe.html`.

[9] Chang, R., and Lo, M. Inbound traffic engineering for multi-homed AS's using AS path prepending. *IEEE Network* (2005), 18–25.

[10] Chen, E., and Bates, T. An Application of the BGP Community Attribute in Multi-home Routing, August 1996.

[11] Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M., and Bowman, M. PlanetLab: An Overlay Testbed for Broad-Coverage Services. *ACM SIGCOMM Computer Communication Review 33*, 3 (July 2003).

[12] Colitti, L. Effect of anycast on K-root. DNS-OARC Workshop, July 2005. `http://www.ripe.net/info/ncc/presentations/anycast-kroot.pdf`.

[13] Fei, Z., Bhattacharjee, S., Zegura, E. W., and Ammar, M. H. A Novel Server Selection Technique for Improving the Response Time of a Replicated Service. In *Proc. of INFOCOM* (1998).

[14] Feldmann, A., Maennel, O., Mao, Z. M., Berger, A., and Maggs, B. Locating internet routing instabilities. In *Proc. of ACM SIGCOMM* (2004).

[15] Freedman, M. J., Lakshminarayanan, K., and Mazires, D. OASIS: Anycast for Any Service. In *Proc. of 3rd USENIX/ACM Symposium on Networked Systems Design and Implementation* (2006).

[16] Greene, B., and McPherson, D. ISP Security: Deploying and Using Sinkholes. NANOG meeting, June 2003. `www.nanog.org/mtg-0306/sink.html`.

[17] Gummadi, K. P., Saroiu, S., and Gribble, S. D. King: Estimating Latency between Arbitrary Internet End Hosts. In *Proc. of the SIGCOMM Internet Measurement Workshop* (2002).

[18] Hardy, T. RFC 3258 - Distributing Authoritative Name Servers via Shared Unicast Addresses, April 2002.

[19] Huitema, C. RFC 3068 - An Anycast Prefix for 6to4 Relay Routers, June 2001.

[20] Katabi, D., and Wroclawski, J. A framework for scalable global IP-anycast (GIA). In *Proc. of ACM*

SIGCOMM (2000).

[21] Kim, D., Meyer, D., Kilmer, H., and Farinacci, D. RFC 3446 - Anycast Rendevous Point (RP) mechanism using Protocol Independent Multicast (PIM) and Multicast Source Discovery Protocol (MSDP), January 2003.

[22] Krishnamurthy, B., Wills, C., and Zhang, Y. On the use and performance of content distribution networks. In *Proc. of ACM SIGCOMM Workshop on Internet Measurement* (2001).

[23] Labovitz, C., Ahuja, A., Bose, A., and Jahanian, F. Delayed Internet routing convergence. *IEEE/ACM Trans. Netw.* (2001).

[24] Lo, S. S. M., and Chang, R. K. C. Active Measurement of the AS Path Prepending Method. In *Proc. of ICNP (Poster)* (2005).

[25] Mao, Z. M., Bush, R., Griffin, T. G., and Roughan, M. BGP beacons. In *Proc. of the 3rd ACM SIGCOMM conference on Internet measurement* (2003).

[26] Matsunaga, S., Ata, S., Kitamura, H., and Murata, M. Applications of IPv6 Anycasting. draft-ata-ipv6-anycast-app-00, February 2005.

[27] Miller, K. Deploying IP Anycast. NANOG 29 meeting, 2003. http://www.net.cmu.edu/pres/anycast/.

[28] Partridge, C., Mendez, T., and Milliken, W. RFC 1546 - Host Anycasting Service, November 1993.

[29] Quoitin, B., Pelsser, C., Bonaventure, O., and Uhlig, S. A performance evaluation of BGP-based traffic engineering. *Intl. Journal of Network Management 15*, 3 (2005).

[30] Ratnasamy, S., Shenker, S., and McCanne, S. Towards an Evolvable Internet Architecture. In *Proceedings of SIGCOMM 2005* (Aug. 2005).

[31] Sarat, S., Pappas, V., and Terzis, A. On the use of Anycast in DNS. Tech. rep., HiNRG, Johns Hopkins University Technical Report, 2004.

[32] Sarat, S., Pappas, V., and Terzis, A. On the use of Anycast in DNS (Poster). In *Proc. of ACM SIGMETRICS* (2005).

[33] Shaikh, A., Tewari, R., and Agrawal, M. On the Effectiveness of DNS-based Server Selection. In *Proc. of IEEE INFOCOM 2001* (2001).

[34] Spring, N., Mahajan, R., and Anderson, T. Quantifying the Causes of Path Inflation. In *Proc. of ACM SIGCOMM* (2003).

[35] Stoica, I., Adkins, D., Zhuang, S., Shenker, S., and Surana, S. Internet Indirection Infrastructure. In *Proc. of ACM SIGCOMM* (2002).

[36] Wong, B., and Sirer, E. G. ClosestNode.com: An Open-Access, Scalable, Shared Geocast Service for Distributed Systems. *SIGOPS Operating Systems Review 40*, 1 (Jan 2006).

[37] Zegura, E. W., Ammar, M. H., Fei, Z., and Bhattacharjee, S. Application-layer anycasting: a server selection architecture and use in a replicated Web service. *IEEE/ACM Trans. Netw. 8*, 4 (2000), 455–466.

[38] Akamai, May 2006. http://www.akamai.com/.

[39] AS112 Project Home Page, May 2006. www.as112.net.

[40] CacheFly, May 2006. http://www.cachefly.com.

[41] CIDR Report, May 2006. http://www.cidr-report.org/.

[42] Global Server Load Balancing, May 2006. http://www.tenereillo.com/GSLBPageOfShame.htm.

[43] History of DNS Root Anycast controversy, May 2006. http://www.av8.net/IETF-watch/DNSRootAnycast/History.html.

[44] Internal Anycast Service deployment, May 2006. http://pias.gforge.cis.cornell.edu/deployment.php.

[45] ISC F Root-Server, May 2006. http://www.isc.org/index.pl?/ops/f-root/.

[46] National Lambda Rail, April 2006. www.nlr.net.

[47] Report on DNS Amplification attacks, May 2006. http://www.circleid.com/posts/report_on_dns_amplification_attacks/.

[48] Root-Server Technical Operations, May 2006. http://www.root-servers.org/.

[49] Route Views Project Page, May 2006. www.route-views.org.

[50] SprintLink's BGP Policy, May 2006. http://www.sprintlink.net/policy/bgp.html.