Analyzing the Performance of an Anycast CDN

Matt Calder[*,†], Ashley Flavel[†], Ethan Katz-Bassett[*], Ratul Mahajan[†], and Jitendra Padhye[†]

[†]Microsoft [*]University of Southern California

ABSTRACT

Content delivery networks must balance a number of trade-offs when deciding how to direct a client to a CDN server. Whereas DNS-based redirection requires a complex global traffic manager, anycast depends on BGP to direct a client to a CDN front-end. Anycast is simple to operate, scalable, and naturally resilient to DDoS attacks. This simplicity, however, comes at the cost of precise control of client redirection. We examine the performance implications of using anycast in a global, latency-sensitive, CDN. We analyze millions of client-side measurements from the Bing search service to capture anycast versus unicast performance to nearby front-ends. We find that anycast usually performs well despite the lack of precise control but that it directs roughly 20% of clients to a suboptimal front-end. We also show that the performance of these clients can be improved through a simple history-based prediction scheme.

Categories and Subject Descriptors

C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks—Internet; C.4 [**Performance of Systems**]: Measurement techniques

Keywords

Anycast; CDN; Measurement;

1. INTRODUCTION

Content delivery networks are a critical part of Internet infrastructure. CDNs deploy front-end servers around the world and direct clients to nearby, available front-ends to reduce bandwidth, improve performance, and maintain reliability. We will focus on a CDN architecture which directs the client to a nearby front-end, which terminates the client's TCP connection and relays requests to a backend server in a data center. The key challenge for a CDN is to map each client to the right front-end. For latency-sensitive services such as search results, CDNs try to reduce the client-perceived latency by mapping the client to a nearby front-end.

CDNs can use several mechanisms to direct the client to a front-end. The two most popular mechanisms are DNS and anycast. DNS-based redirection was pioneered by Akamai. It offers fine-grained and near-real time control over client-front-end mapping, but requires considerable investment in infrastructure and operations [35].

IMC'15, October 28–30, 2015, Tokyo, Japan.

ACM 978-1-4503-3848-6/15/10 ...$15.00.
DOI: http://dx.doi.org/10.1145/2815675.2815717.

Some newer CDNs like CloudFlare rely on anycast [1], announcing the same IP address(es) from multiple locations, leaving the client-front-end mapping at the mercy of Internet routing protocols. Anycast offers only minimal control over client-front-end mapping and is performance agnostic by design. However, it is easy and cheap to deploy an anycast-based CDN – it requires no infrastructure investment, beyond deploying the front-ends themselves. The anycast approach has been shown to be quite robust in practice [23].

In this paper, we aim to answer the questions: Does anycast direct clients to nearby front-ends? What is the performance impact of poor redirection, if any? To study these questions, we use data from Bing's anycast-based CDN [23]. We instrumented the search stack so that a small fraction of search response pages carry a JavaScript beacon. After the search results display, the JavaScript measures latency to four front-ends– one selected by anycast, and three nearby ones that the JavaScript targets. We compare these latencies to understand anycast performance and determine potential gains from deploying a DNS solution.

Our results paint a mixed picture of anycast performance. For most clients, anycast performs well despite the lack of centralized control. However, anycast directs around 20% of clients to a sub-optimal front-end. When anycast does not direct a client to the best front-end, we find that the client usually still lands on a nearby alternative front-end. We demonstrate that the anycast inefficiencies are stable enough that we can use a simple prediction scheme to drive DNS redirection for clients underserved by anycast, improving performance of 15%-20% of clients. Like any such study, our specific conclusions are closely tied to the current front-end deployment of the CDN we measure. However, as the first study of this kind that we are aware of, the results reveal important insights about CDN performance, demonstrating that anycast delivers optimal performance for most clients.

2. CLIENT REDIRECTION

A CDN can direct a client to a front-end in multiple ways.

DNS: The client will fetch a CDN-hosted resource via a hostname that belongs to the CDN. The client's local DNS resolver (LDNS), typically configured by the client's ISP, will receive the DNS request to resolve the hostname and forward it to the CDN's authoritative nameserver. The CDN makes a performance-based decision about what IP address to return based on which LDNS forwarded the request. DNS redirection allows relatively precise control to redirect clients on small timescales by using small DNS cache TTL values.

Since a CDN must make decisions at the granularity of LDNS rather than client, DNS-based redirection faces some challenges. An LDNS may be distant from the clients that it serves or may serve clients distributed over a large geographic region, such that there is no good single redirection choice an authoritative resolver can make. This situation is very common with public DNS resolvers such as Google Public DNS and OpenDNS, which serve large, geographically disparate sets of clients [17]. A proposed solution to this issue is the EDNS client-subnet-prefix standard (ECS) which allows a portion of the client's actual IP address to be forwarded

to the authoritative resolver, allowing per-prefix redirection decisions [21].

Anycast: Anycast is a routing strategy where the same IP address is announced from many locations throughout the world. Then BGP routes clients to one front-end location based on BGP's notion of best path. Because anycast defers client redirection to Internet routing, it offers operational simplicity. Anycast has an advantage over DNS-based redirection in that each client redirection is handled independently – avoiding the LDNS problems described above.

Anycast has some well-known challenges. First, anycast is unaware of network performance, just as BGP is, so it does not react to changes in network quality along a path. Second, anycast is unaware of server load. If a particular front-end becomes overloaded, it is difficult to gradually direct traffic away from that front-end, although there has been recent progress in this area [23]. Simply withdrawing the route to take that front-end offline can lead to cascading overloading of nearby front-ends. Third, anycast routing changes can cause ongoing TCP sessions to terminate and need to be restarted. In the context of the Web, which is dominated by short flows, this does not appear to be an issue in practice [31, 23]. Many companies, including Cloudflare, CacheFly, Edgecast, and Microsoft, run successful anycast-based CDNs.

Other Redirection Mechanisms: Whereas anycast and DNS direct a client to a front-end before the client initiates a request, the response from a front-end can also direct the client to a different server for other resources, using, for example, HTTP status code 3xx or manifest-based redirection common for video [4]. These schemes add extra RTTs, and thus are not suitable for latency-sensitive Web services such as search. We do not consider them further in this paper.

3. METHODOLOGY

Our goal is to answer two questions: **1)** How effective is anycast in directing clients to nearby front-ends? And **2)** How does anycast performance compare against the more traditional DNS-based unicast redirection scheme? We experiment with Bing's anycast-based CDN to answer these questions. The CDN has dozens of front end locations around the world, all within the same Microsoft-operated autonomous system. We use measurements from real clients to Bing CDN front-ends using anycast and unicast. In § 4, we compare the size of this CDN to others and show how close clients are to the front ends.

3.1 Routing Configuration

All test front-ends locations have both anycast and unicast IP addresses.

Anycast: Bing is currently an anycast CDN. All production search traffic is current served using anycast from all front-ends.

Unicast: We also assign each front-end location a unique /24 prefix which does not serve production traffic. Only the routers at the closest peering point to that front-end announce the prefix, forcing traffic to the prefix to ingress near the front-end rather than entering Microsoft's backbone at a different location and traversing the backbone to reach the front-end. This routing configuration allows the best head-to-head comparison between unicast and anycast redirection, as anycast traffic ingressing at a particular peering point will also go to the closest front-end.

3.2 Data Sets

We use both passive and active measurements in our study, as discussed below.

3.2.1 Passive Measurements

Bing server logs provide detailed information about client requests for each search query. For our analysis we use the client IP address, location, and what front-end was used during a particular request. This data set was collected on the first week of April 2015 and represents many millions of queries.

3.2.2 Active Measurements

To actively measure CDN performance from the client, we inject a JavaScript beacon into a small fraction of Bing Search results. After the results page has completely loaded, the beacon instructs the client to fetch four test URLs. These URLs trigger a set of DNS queries to our authoritative DNS infrastructure. The DNS query results are randomized front-end IPs for measurement diversity, which we discuss more in § 3.3.

The beacon measures the latency to these front-ends by downloading the resources pointed to by the URLs, and reports the results to a backend infrastructure. Our authoritative DNS servers also push their query logs to the backend storage. Each test URL has a globally unique identifier, allowing us to join HTTP results from the client side with DNS results from the server side [34].

The JavaScript beacon implements two techniques to improve quality of measurements. First, to remove the impact of DNS lookup from our measurements, we first issue a warm-up request so that the subsequent test will use the cached DNS response. While DNS latency may be responsible for some aspects of poor Web-browsing performance [5], in this work we are focusing on the performance of paths between client and front-ends. We set TTLs longer than the duration of the beacon. Second, using JavaScript to measure the elapsed time between the start and end of a fetch is known to not be a precise measurement of performance [32], whereas the W3C Resource Timing API [29] provides access to accurate resource download timing information from compliant Web browsers. The beacon first records latency using the primitive timings. Upon completion, if the browser supports the resource timing API, then the beacon substitutes the more accurate values.

We study measurements collected from many millions of search queries over March and April 2015. We aggregated client IP addresses from measurements into /24 prefixes because they tend to be localized [27]. To reflect that the number of queries per /24 is heavily skewed across prefixes [35], for both the passive and active measurements, we present some of our results weighting the /24s by the number of queries from the prefix in our corresponding measurements.

3.3 Choice of Front-ends to Measure

The main goal of our measurements is to compare the performance achieved by anycast with the performance achieved by directing clients to their best performing front-end. Measuring from each client to every front-end would introduce too much overhead, but we cannot know a priori which front-end is the best choice for a given client at a given point in time.

We use three mechanisms to balance measurement overhead with measurement accuracy in terms of uncovering the best performing choices and obtaining sufficient measurements to them. First, for each LDNS, we consider only the ten closest front-ends to the LDNS (based on geolocation data) as candidates to consider returning to the clients of that LDNS. Recent work has show that LDNS is a good approximation of client location: excluding 8% of demand from public resolvers, only 11-12% of demand comes from clients who are further than 500km from their LDNS [17]. In Figure 1, we will show that our geolocation data is sufficiently accurate that the best front-ends for the clients are generally within that set. Second,

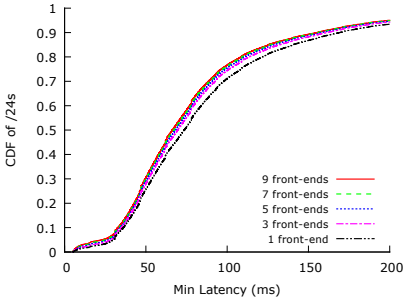

Figure 1: Diminishing returns of measuring to additional front-ends. The close grouping of lines for the 5th+ closest front-ends suggests that measuring to additional front-ends provides negligible benefit.

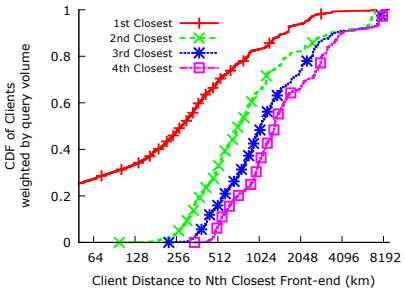

Figure 2: Distances in kilometers (log scale) from volume-weighted clients to nearest front-ends.

to further reduce overhead, each beacon only makes four measurements to front-ends: (a) a measurement to the front-end selected by anycast routing; (b) a measurement to the front-end judged to be geographically closest to the LDNS; and (c-d) measurements to two front-ends randomly selected from the other nine candidates, with the likelihood of a front-end being selected weighted by distance from the client LDNS (e.g. we return the 3rd closest front-end with higher probability than the 4th closest front-end). Third, for most of our analysis, we aggregate measurements by /24 and consider distributions of performance to a front-end, so our analysis is robust even if not every client measures to the best front-end every time.

To partially validate our approach, Figure 1 shows the distribution of minimum observed latency from a client /24 to a front-end. The labeled *N*th line includes latency measurements from the nearest *N* front-ends to the LDNS. The results show decreasing latency as we initially include more front-ends, but we see little decrease after adding five front-ends per prefix, for example. So, we do not expect that minimum latencies would improve for many prefixes if we measured to more than the nearest ten front-ends that we include in our beacon measurements.

4. CDN SIZE AND GEO-DISTRIBUTION

The results in this paper are specific to Bing's anycast CDN deployment. In this section we characterize the size of the deployment, showing that our deployment is of a similar scale–a few dozens of front-end server locations–to most other CDNs and in particular most anycast CDNs, although it is one of the largest deployments within that rough scale. We then measure what the distribution of these dozens of front-end locations yields in terms of the distance from clients to the nearest front-ends. Our characterization of the performance of this CDN is an important first step towards understanding anycast performance. An interesting direction for future work is to understand how to extend these performance results to CDNs with different numbers and locations of servers and with different interdomain connectivity [18].

We compare our CDN to others based on the number of server locations, which is one factor impacting CDN and anycast performance. We examine 21 CDNs and content providers for which there is publicly available data [3]. Four CDNs are extreme outliers. ChinaNetCenter and ChinaCache each have over 100 locations in China. Previous research found Google to have over 1000 locations worldwide [16], and Akamai is generally known to have over 1000 as well [17]. While this scale of deployment is often the popular image of a CDN, it is in fact the exception. Ignoring the large Chinese deployments, the next largest CDNs we found public data for are CDNetworks (161 locations) and SkyparkCDN (119 locations). The remaining 17 CDNs we examined (including ChinaNetCenter's and ChinaCache's deployments outside of China) have between 17 locations (CDNify) and 62 locations (Level3). In terms of number of locations and regional coverage, the Bing CDN is most similar to Level3 and MaxCDN. Well-known CDNs with smaller deployments include Amazon CloudFront (37 locations), CacheFly (41 locations), CloudFlare (43 locations) and EdgeCast (31 locations). CloudFlare, CacheFly, and EdgeCast are anycast CDNs.

To give some perspective on the density of front-end distribution, Figure 2 shows the distance from clients to nearest front-ends, weighted by client Bing query volumes. The median distance of the nearest front-end is 280 km, of the second nearest is 700 km, and of fourth nearest is 1300 km.

5. ANYCAST PERFORMANCE

We use measurements to estimate the performance penalty anycast pays in exchange for simple operation. Figure 3 is based on millions of measurements, collected over a period of a few days, and inspired us to take on this project.

As explained in § 3, each execution of the JavaScript beacon yields four measurements, one to the front-end that anycast selects, and three to nearby unicast front-ends. For each request, we find the latency difference between anycast and the lowest-latency unicast front-end. Figure 3 shows the fraction of requests where anycast performance is slower than the best of the three unicast front-ends. Most of the time, in most regions, anycast does well, performing as well as the best of the three nearby unicast front-ends. However, anycast is at least 25ms slower for 20% of requests, and just below 10% of anycast measurements are 100ms or more slower than the best unicast for the client.

This graph suggests possible benefits in using DNS-based redirection for some clients, with anycast for the rest. Note that this is not an upper bound: to derive that, we would have to poll all front-ends in each beacon execution, which is too much overhead. There is also no guarantee that a deployed DNS-based redirection system will be able to achieve the performance improvement seen in Figure 3 – to do so the DNS-based redirection system would have to be practically clairvoyant. Nonetheless, this result was sufficiently tantalizing for us to study anycast performance in more detail, and seek ways to improve it.

Examples of poor anycast routes: A challenge in understanding anycast performance is figuring out why clients are being directed to distant or poor performing edges front-ends. To troubleshoot, we used the RIPE Atlas [2] testbed, a network of over 8000 probes predominantly hosted in home networks. We issued traceroutes from Atlas probes hosted within the same ISP-metro area pairs where we have observed clients with poor performance. We observe in our analysis that many instances fall into one of two cases. 1) BGP's lack of insight into the underlying topology causes anycast

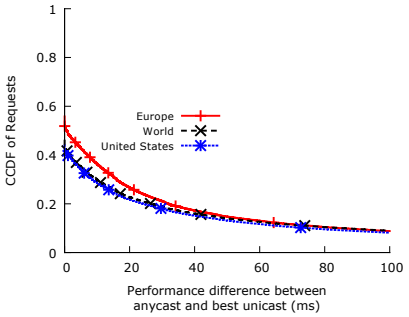

Figure 3: The fraction of requests where the best of three different unicast front-ends out-performed anycast.

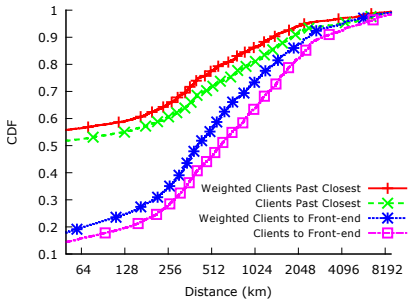

Figure 4: The distance in kilometers (log scale) between clients and the anycast front-ends they are directed to.

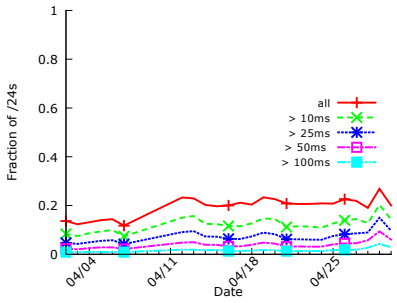

Figure 5: Daily poor-path prevalence during April 2015 showing what fraction of client /24s see different levels of latency improvement over anycast when directed to their best performing unicast front-end.

to make suboptimal choices and 2) intradomain routing policies of ISPs select remote peering points with our network.

In one interesting example, a client was roughly the same distance from two border routers announcing the anycast route. Anycast chose to route towards router *A*. However, internally in our network, router *B* is very close to a front-end *C*, whereas router *A* has a longer intradomain route to the nearest front-end, front-end *D*. With anycast, there is no way to communicate [39] this internal topology information in a BGP announcement.

Several other examples included cases where a client is nearby a front-end but the ISP's internal policy chooses to hand off traffic at a distant peering point. Microsoft intradomain policy then directs the client's request to the front-end nearest to the peering point, not to the client. Some examples we observed of this was an ISP carrying traffic from a client in Denver to Phoenix and another carrying traffic from Moscow to Stockholm. In both cases, direct peering was present at each source city.

Intrigued by these sorts of case studies, we sought to understand anycast performance quantitatively. The first question we ask is whether anycast performance is poor simply because it occasionally directs clients to front-ends that are geographically far away, as was the case when clients in Moscow went to Stockholm.

Does anycast direct clients to nearby front-ends? In a large CDN with presence in major metro areas around the world, most ISPs will see BGP announcements for front-ends from a number of different locations. If peering among these points is uniform, then the ISP's least cost path from a client to a front-end will often be the geographically closest. Since anycast is not load or latency aware, geographic proximity is a good indicator of expected performance.

Figure 4 shows the distribution of the distance from client to anycast front-end for all clients in one day of production Bing traffic. One line weights clients by query volume. Anycast is shown to perform 5-10% better at all percentiles when accounting for more active clients. We see that about 82% of clients are directed to a front-end within 2000 km while 87% of client volume is within 2000 km.

The second pair of lines in Figure 4, labeled "Past Closest", shows the distribution of the difference between the distance from a client to its closest front-end and the distance from the client to the front-end anycast directs to. About 55% of clients and weighted clients have distance 0, meaning they are directed to the nearest front-end. Further, 75% of clients are directed to a front-end within around 400 km and 90% are within 1375 km of their closest. This supports the idea that, with a dense front-end deployment such as is achievable in North America and Europe, anycast directs most clients to a relatively nearby front-end that should be expected to deliver good performance, even if it is not the closest.

From a geographic view, we found that around 10-15% of /24s are directed to distant front-ends, a likely explanation for poor performance.[1] Next we examine how common these issues are from day-to-day and how long issues with individual networks persist.

Is anycast performance consistently poor? We first consider whether significant fractions of clients see consistently poor performance with anycast. At the end of each day, we analyzed all collected client measurements to find prefixes with room for improvement over anycast performance. For each client /24, we calculate the median latency between the prefix and each measured unicast front-end and anycast.

Figure 5 shows the prevalence of poor anycast performance each day during April 2015. Each line specifies a particular minimum latency improvement, and the figure shows the fraction of client /24s each day for which some unicast front-end yields at least that improvement over anycast. On average, we find that 19% of prefixes see some performance benefit from going to a specific unicast front-end instead of using anycast. We see 12% of clients with 10ms or more improvement, but only 4% see 50ms or more.

Poor performance is not limited to a few days–it is a daily concern. We next examine whether the same client networks experience recurring poor performance. How long does poor performance persist? Are the problems seen in Figure 5 always due to the same problematic clients?

Figure 6 shows the duration of poor anycast performance during April 2015. For the majority of /24s categorized as having poor-performing paths, those poor-performing paths are short-lived. Around 60% appear for only one day over the month. Around 10% of /24s show poor performance for 5 days or more. These days are not necessarily consecutive. We see that only 5% of /24s see continuous poor performance over 5 days or more.

These results show that while there is a persistent amount of poor anycast performance over time, the majority of problems only last

[1]No geolocation database is perfect. A fraction of very long client-to-front-end distances may be attributable to bad client geolocation data.

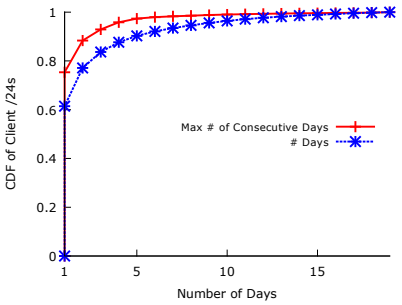

Figure 6: Poor path duration across April 2015. We consider poor anycast paths to be those with any latency inflation over a unicast front-end.

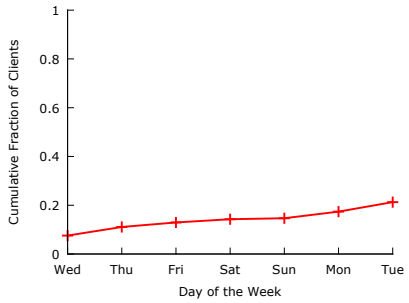

Figure 7: The cumulative fraction of clients that have changed front-ends at least once by different points in a week

for a single day. Next we look at how much of poor performance can be attributed to clients frequently switching between good and poor performing front-ends.

Front-end Affinity: Recurrent front-end selection changes for user over time may indicate route stability issues which can lead to anycast performance problems. We refer to how "attached" particular clients are to a front-end as *front-end affinity*. In this section, we analyze our passive logs.

Figure 7 shows the cumulative fraction of clients that have switched front-ends at least once by that time of the week. Within the first day, 7% of clients landed on multiple front-ends. An additional 2-4% clients see a front-end change each day until the weekend, where there is very little churn, less than .5%. This could be from network operators not pushing out changes during the weekend unless they have to. From the weekend to the beginning of the week, the amount of churn increases again to 2-4% each day. Across the entire week, 21% of clients landed on multiple front-ends, but the vast majority of clients were stable. We discuss potential solutions to this more at the end of §6. We observe that the number of client front-end switches is slightly higher in a one day snapshot compared to the 1.1-4.7% reported in previous work on DNS instance-switches in anycast root nameservers [20, 33]. A likely contributing factor is that our anycast deployment is around 10 times larger than the number of instances present in K root name server at the time of that work.

Figure 8 shows the change in the client-to-front-end distance when the front-end changes. This shows that when the majority of clients switch front-ends, it is to a nearby front-end. This makes sense given the CDN front-end density in North America and Europe. The median change in distance from front-end switches is 483 km while 83% are within 2000 km.

We saw in this section that most clients show high front-end-affinity, that is, they continue going to the same front-end over time. For the clients that do switch front-ends, there is a long tail of distance between a client and switched pairs of front-ends.

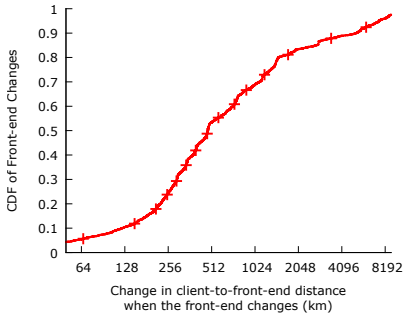

Figure 8: The distribution of change in client-to-front-end distance (log scale) when when the front-end changes, for the 7% of clients that change front-end throughout a day.

6. ADDRESSING POOR PERFORMANCE

The previous section showed that anycast often achieves good performance, but sometimes suffers significantly compared to unicast beacon measurements. However, the ability for unicast to beat anycast in a single measurement does not guarantee that this performance is predictable enough to be achievable if a system has to return a single unicast front-end to a DNS query. If a particular front-end outperformed anycast in the past for a client, will it still if the system returns that front-end next time? Additionally, because of DNS's design, the system does not know which client it is responding to, and so its response applies either to all clients of an LDNS or all clients in a prefix (if using ECS). Can the system reliably determine front-ends that will perform well for the set of clients?

We evaluate to what degree schemes using DNS and ECS can improve performance for clients with poor anycast performance. We evaluate (in emulation based on our real user measurements) a prediction scheme that maps from a client group (clients of an LDNS or clients within an ECS prefix) to its predicted best front-end. It updates its mapping every *prediction interval*, set to one day in our experiment.[2] The scheme chooses to map a client group to the lowest latency front-end across the measurements for that group, picking either the anycast address or one of the unicast front-ends. We evaluate two *prediction metrics* to determine the latency of a front-end, 25th percentile and median latency from that client group to that front-end. We choose lower percentiles, as analysis of client data showed that higher percentiles of latency distributions are very noisy (we omit detailed results due to lack of space). This noise makes prediction difficult, as it can result in overlapping performance between two front-ends. The 25th percentile and median have lower coefficient of variation, indicating less variation and more stability. Our initial evaluation showed that both 25th percentile and median show very similar performance as *prediction metrics*, so we only present results for 25th percentile.

We emulate the performance of such a prediction scheme using our existing beacon measurements. We base the predictions on one day's beacon measurements. For a given client group, we select among the front-ends with 20+ measurements from the clients.

We evaluate the performance of the prediction scheme by comparing against the performance observed in next day's beacon measurements. We compare 50th and 75th anycast performance for the group to 50th and 75th performance for the predicted front-end. The Bing team routinely uses 75% percentile latency as an internal benchmarks for a variety of comparisons. Next, we evaluate prediction using both ECS and LDNS client grouping.

[2]We cannot make predictions at finer timescales, as our sampling rate was limited due to engineering issues.

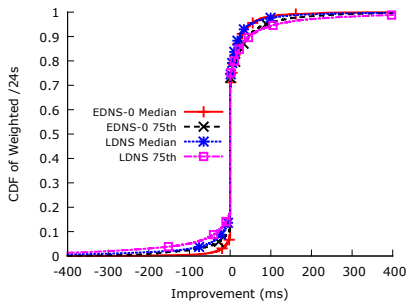

Figure 9: Improvement over anycast from making LDNS or ECS-based decisions with prediction using 25th percentile prediction metric. Negative x-axis values show where anycast was better than our prediction. Values at 0 show when we predicted anycast was the best performing. Positive x-axis values show our improvement.

Prediction using EDNS client-subnet-prefix: The ECS extension [21] enables precise client redirection by including the client's prefix in a DNS request. Our prediction scheme is straightforward: we consider all beacon measurements for a /24 client network and choose the front-end according to the *prediction metrics*.

The "EDNS-0" lines in Figure 9 depict, as a distribution across clients weighted by query volume, the difference between performance to the predicted front-end (at the 50th and 75th percentile) and the performance to the anycast-routed front-end (at the same percentiles). Most clients see no difference in performance, in most cases because prediction selected the anycast address. For the nearly 40% of queries-weighted prefixes we predict to see improvement over anycast, only 30% see a performance improvement over anycast, while 10% of weighted prefixes see worse performance than they would with anycast.

LDNS-based prediction: Traditionally, DNS-based redirection can only make decisions based on a client's LDNS. In this section, we estimate to what degree LDNS granularity can achieve optimal performance when anycast routing sends clients to suboptimal servers. We construct a latency mapping from LDNS to each measured edge by assigning each front-end measurement made by a client to the client's LDNS, which we can identify by joining our DNS and HTTP logs based on the unique hostname for the measurement. We then consider all beacon measurements assigned to an LDNS and select the LDNS's best front-end using the *prediction metrics*. In the page loads in our experiment, public DNS resolvers made up a negligible fraction of total LDNS traffic so their wide user base have an insignificant impact on results.

The "LDNS" lines in Figure 9 show the fraction of /24 client networks that can be improved from using prediction of performance based on an LDNS-based mapping. While we see improvement for around 27% of weighted /24s, we also pay a penalty where our prediction did poorly for around 17% of /24s.

Our results demonstrate that traditional and recent DNS techniques can improve performance for many of the clients who experience suboptimal anycast routing. We are also considering a hybrid approach that combines anycast with DNS-based redirection. The key idea is to use DNS-based redirection for a small subset of poor performing clients, while leaving others to anycast. Such a hybrid approach may outperform DNS redirection for clients not well represented by their LDNS, and it may be more scalable.

7. RELATED WORK

Most closely related to our work is from Alzoubi et al. [9, 8]. They describe a load-aware anycast CDN architecture where ingress routes from a CDN to a large ISP are managed by an ISP's centralized route controller. Unlike our work, they do not examine the end-to-end application performance comparison between DNS redirection and anycast. Follow up work focuses on handling anycast TCP session disruption due to BGP path changes [7]. Our work is also closely related to FastRoute [23], a system for load balancing within an anycast CDN, but it does not address performance issues around redirection. There has been a good deal of work on improving and evaluating general CDN performance [37, 24, 36, 6, 35, 25]. The majority of previous work on anycast performance has focused on DNS. There has been significant attention to anycast DNS from the network operations community [13, 15, 14, 28, 19, 12, 20] but less so for TCP and anycast [31]. Sarat et al. examined the performance impact of anycast on DNS across different anycast configurations [38]. Fan et al. [22] present new methods to identify and characterize anycast nodes. There are several pieces of work describing deployment of anycast services [30, 10, 11, 26].

Akamai recently published a study on DNS-based redirection [17]. The authors showed that the majority of clients are nearby their LDNS, enabling DNS-based redirection to perform well. However, they also show that a significant number of clients are far from their LDNS, and that some LDNS serve clients spread over large geographic regions. The paper describes Akamai's adoption of ECS-based redirection for clients of public DNS resolvers, showing impressive performance improvements for these clients versus LDNS-based redirection. However, public resolvers only make up a small fraction of global DNS traffic. Clients using their ISPs' LDNS cannot benefit unless the ISPs enable ECS and the CDN supports ECS requests from the LDNS. Since anycast works well for many clients, we see benefit in a hybrid approach that chooses whether to use DNS redirection or anycast based on measurements of which works better for the LDNS and whether the LDNS supports ECS.

8. CONCLUSION

In this paper we studied the performance of a large anycast-based CDN, and evaluated whether it could be improved by using a centralized, DNS-based solution. We found that anycast usually performs well despite the lack of precise control, but that it directs $\approx$ 20% of clients to a suboptimal front-end. We demonstrated that a simple prediction scheme may allow DNS redirection to improve performance for some of the clients that see poor anycast performance.

Acknowledgements

We gratefully acknowledge Nick Holt and Daniel Gicklhorn for their support of this work. Matt Calder and Ethan Katz-Bassett were partially supported by the U.S. National Science Foundation grant numbers CNS-1351100 and CNS-1413978.

9. REFERENCES

[1] CloudFlare. `https://www.cloudflare.com/`.
[2] RIPE Atlas. `https://atlas.ripe.net/`.
[3] USC CDN Coverage. `http://usc-nsl.github.io/cdn-coverage`.
[4] V. K. Adhikari, Y. Guo, F. Hao, V. Hilt, and Z.-L. Zhang. Tale of Three CDNs: An Active Measurement Study of Hulu and its CDNs. In *IEEE Global Internet Symposium '12*.
[5] B. Ager, W. Mühlbauer, G. Smaragdakis, and S. Uhlig. Comparing DNS Resolvers in the Wild. In *IMC '10*.
[6] B. Ager, W. Mühlbauer, G. Smaragdakis, and S. Uhlig. Web Content Cartography. In *IMC '11*.

[7] Z. Al-Qudah, S. Lee, M. Rabinovich, O. Spatscheck, and J. Van der Merwe. Anycast-aware Transport for Content Delivery Networks. In *WWW '09*.

[8] H. A. Alzoubi, S. Lee, M. Rabinovich, O. Spatscheck, and J. Van Der Merwe. A Practical Architecture for an Anycast CDN. *ACM Transactions on the Web (TWEB) '11*.

[9] H. A. Alzoubi, S. Lee, M. Rabinovich, O. Spatscheck, and J. Van der Merwe. Anycast CDNs Revisited. In *WWW '08*.

[10] H. Ballani and P. Francis. Towards a Global IP Anycast Service. In *SIGCOMM '05*.

[11] H. Ballani, P. Francis, and S. Ratnasamy. A Measurement-based Deployment Proposal for IP Anycast. In *IMC '06*.

[12] P. Barber, M. Larson, and M. Kosters. Traffic Source Analysis of the J Root Anycast Instances. NANOG 39. February, '07.

[13] P. Barber, M. Larson, M. Kosters, and P. Toscano. Life and Times of J-ROOT. NANOG 32. October, '04.

[14] P. Boothe and R. Bush. Anycast Measurements Used To Highlight Routing Instabilities. NANOG 35. October, '05.

[15] P. Boothe and R. Bush. DNS Anycast Stability. *19th APNIC, '05*.

[16] M. Calder, X. Fan, Z. Hu, E. Katz-Bassett, J. Heidemann, and R. Govindan. Mapping the Expansion of Google's Serving Infrastructure. In *IMC '13*.

[17] F. Cheng, R. K. Sitaraman, and M. Torres. End-user mapping: Next Generation Request Routing for Content Delivery. In *SIGCOMM '15*.

[18] Y. Chiu, B. Schlinker, A. B. Radhakrishnan, E. Katz-Bassett, and R. Govindan. Are We One Hop Away from a Better Internet? In *IMC '15*.

[19] L. Coletti. Effects of Anycast on K-root Performance. NANOG 37. June, '06.

[20] L. Colitti, E. Romijn, H. Uijterwaal, and A. Robachevsky. Evaluating the Effects of Anycast on DNS Root Name Servers. *RIPE document RIPE-393, '06*.

[21] C. Contavalli, W. van der Gaast, D. Lawrence, and W. Kumari. Client Subnet in DNS Requests. IETF Draft draft-vandergaast-edns-client-subnet-02, July 2015.

[22] X. Fan, J. Heidemann, and R. Govindan. Evaluating Anycast in the Domain Name System. In *INFOCOM '13*.

[23] A. Flavel, P. Mani, D. Maltz, N. Holt, J. Liu, Y. Chen, and O. Surmachev. FastRoute: A Scalable Load-Aware Anycast Routing Architecture for Modern CDNs. In *NSDI '15*.

[24] B. Frank, I. Poese, Y. Lin, G. Smaragdakis, A. Feldmann, B. Maggs, J. Rake, S. Uhlig, and R. Weber. Pushing CDN-ISP Collaboration to the Limit. *SIGCOMM CCR '14*.

[25] M. J. Freedman, E. Freudenthal, and D. Mazieres. Democratizing Content Publication with Coral. In *NSDI '04*.

[26] M. J. Freedman, K. Lakshminarayanan, and D. Mazières. OASIS: Anycast for Any Service. In *NSDI '06*.

[27] M. J. Freedman, M. Vutukuru, N. Feamster, and H. Balakrishnan. Geographic Locality of IP Prefixes. In *IMC '05*.

[28] J. Hiebert, P. Boothe, R. Bush, and L. Lynch. Determining the Cause and Frequency of Routing Instability with Anycast. In *AINTEC '06*.

[29] A. Jain, J. Mann, Z. Wang, and A. Quach. W3C Resource Timing Working Draft. http://www.w3.org/TR/resource-timing/, July 2015.

[30] D. Katabi and J. Wroclawski. A Framework For Scalable Global IP-anycast (GIA). *SIGCOMM CCR '00*.

[31] M. Levine, B. Lyon, and T. Underwood. Operation Experience with TCP and Anycast. NANOG 37. June, '06.

[32] W. Li, R. K. Mok, R. K. Chang, and W. W. Fok. Appraising the Delay Accuracy In Browser-based Network Measurement. In *IMC '13*.

[33] Z. Liu, B. Huffaker, M. Fomenkov, N. Brownlee, et al. Two Days in the Life of the DNS Anycast Root Servers. In *PAM '07*.

[34] Z. M. Mao, C. D. Cranor, F. Douglis, M. Rabinovich, O. Spatscheck, and J. Wang. A Precise and Efficient Evaluation of the Proximity Between Web Clients and Their Local DNS Servers. In *USENIX ATC '02*.

[35] E. Nygren, R. K. Sitaraman, and J. Sun. The Akamai Network: A Platform for High-performance Internet Applications. *SIGOPS '10*.

[36] J. S. Otto, M. A. Sánchez, J. P. Rula, and F. E. Bustamante. Content Delivery and the Natural Evolution of DNS: Remote DNS Trends, Performance Issues and Alternative Solutions. In *IMC '12*.

[37] I. Poese, B. Frank, B. Ager, G. Smaragdakis, S. Uhlig, and A. Feldmann. Improving Content Delivery with PaDIS. *Internet Computing, IEEE '12*.

[38] S. Sarat, V. Pappas, and A. Terzis. On the Use of Anycast in DNS. In *ICCCN '06*.

[39] N. Spring, R. Mahajan, and T. Anderson. The Causes of Path Inflation. In *SIGCOMM '03*.