Analyzing the Performance of an Anycast CDN

Matt Calder (USC), Ethan Katz-Bassett (USC), Ratul Mahajan (Microsoft Research), Jitu Padhye (Microsoft Research)

ABSTRACT

Content delivery networks must balance a number of trade-offs when deciding what method to use to direct a client to a CDN server. Whereas DNS-based redirection schemes require a complex global traffic manager, anycast depends on BGP routing to decide which CDN front-end serves a client. Anycast is simple to operate, scalable, and naturally resilient to DDoS attacks. The trade-off for this simplicity is lack of precise control of client redirection. In this paper, we examine the performance implications of using anycast in a global, latency-sensitive, production CDN. We analyze millions of client-side measurements from the Bing search service to capture anycast performance versus unicast performance to nearby front-ends. We find that anycast usually performs well despite the lack of centralized control but that it directs $\approx$ 20% of clients to a suboptimal front-end. We demonstrate that a simple prediction scheme allows DNS redirection to beat the suboptimal anycast performance for most of these clients.

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 focus on the Bing CDN, which delivers dynamic search result pages. The CDN 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 [28].

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 not only offers almost no control over client-front-end mapping, it is performance agnostic by design. However, it is very 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 [19].

In this paper, we aim to answer the question: for a given set of front-ends, how does the client performance compare for these two redirection approaches? In other words, should the CDN operator invest in the DNS approach, or is the performance of the simpler and cheaper anycast-based system "good enough?"

To study the question, we use data from Bing's anycast-based CDN. 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 suboptimal edge. When anycast does not direct a client to the best front-end, we find that it usually still lands on a nearby 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 as compared to anycast. Like any such study, our specific conclusions are closely tied to the Internet topology and current front-end deployment. However, as the first study of this kind that we are aware of, the results reveal important insights about CDN performance, demonstrating optimal anycast 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-conscious 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.
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. Adding more anycast front-ends locations can improve performance. Because anycast defers client redirection to Internet routing, it offers operational simplicity. In contrast to DNS-based strategies,

there is no need for a centralized global traffic manager, which can be challenging to build and maintain.

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 [19]. 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 [3, 19]. 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 server before the client initiates a request, the response from a server 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 [5]. 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. DATA SETS

Our study uses two data sets. The first set consists of Bing search query logs. The second set contains active client-side measurements collected by a JavaScript beacon. Both data sets contain measurements to around 50 CDN front-ends.

3.1 Search Logs

Bing search 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 focuses on the first week of April 2015, and represents many millions of queries.

3.2 Client-side Measurements

Client-side active measurements are run on a small fraction of page loads for Bing Search. These measurements were collected during March and April 2015 and represent many millions of search queries. Below, we explain in detail the mechanism used to collect these measurements. We aggregate measurements into /24 prefixes. This data was collected during March and April, 2015 and contains many millions requests from millions of /24s.

Javascript beacon: To measure CDN performance, we inject a JavaScript beacon into a small fraction of search results. After the results page has completed loading, the beacon instructs the client to fetch four test URLs. Each test URL has a globally unique identifier.

These URLs trigger a set of DNS queries to our authoritative DNS infrastructure. For one of the URLs (identified by the word "any" in its hostname), the DNS server returns the anycast address. For the other three URLs, the DNS server estimates client's geographic location using the LDNS address it sees, and returns unicast IP addresses to nearby front-ends based on this estimated location.

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 [27].

The JavaScript beacon implements several 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. 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 [25], whereas the W3C Resource Timing API [10] 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.

Regionally Announced Unicast: Client side measurements to front-ends are made using regional announcements in which each edge is assigned a unique /24 prefix, and that prefix is announced only from the peering point closest to that edge. Clients are handed out addresses from this prefix by our DNS server, based on the target edge. This *regionally announced unicast* setup lets us isolate the impact of Internet and ISP routing. Unicast addressing by itself ensures that the client reaches the target edge, but if those addresses were announced from all peering points in our backbone, we would not be able to control how the client reaches the edge. It could enter the backbone in one region and traverse to the target region along our backbone. The performance of such paths would be impacted not only by the Internet and ISP routing policies but also by the routing policies and congestion of our backbone.

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 introduce 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 only use the ten closest front-ends to the LDNS (based on geolocation data) as candidates to consider returning to the clients of

that LDNS. Our geolocation data is sufficiently accurate that the best front-ends for the clients are generally within that set. Second, to further reduce overhead, each beacon measurement 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 next nine closest, with the likelihood of a front-end being selected weighted by distance from the client LDNS (e.g. records for the 3rd closest front-end is returned with higher probability than the 8th 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.

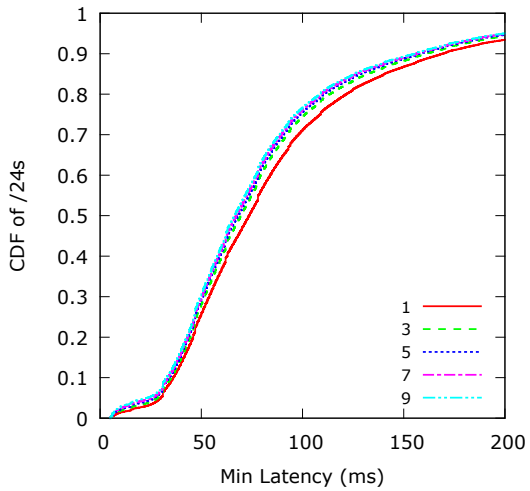

Figure 1: Diminishing returns of measuring to additional front-ends

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.

Having described our data sets and measurement methodology, we now turn to analysis of the collected data.

4. ANYCAST PERFORMANCE

We use measurements to estimate the performance penalty anycast pays in exchange for simple operation. Figure 2 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 2 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, worldwide, anycast is slower for 20% of clients, by as much as 60ms.

This graph suggests the possible performance boost of DNS-based redirection over anycast. 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 2 – 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. We began by investigating a few cases by hand, using tools such as traceroute. In one interesting example, a client was roughly the same distance from two border routers announcing the anycast route, leaving BGP little to distinguish between them. Routing picked to route towards router *A*. However, internally in our network, router *B* is very close to an front-end*C*, whereas traffic has a longer intradomain route to front-end*D* after ingressing at router *A*. With anycast, there is no way to communicate this internal topology information in a BGP announcement. Intrigued by these sort 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.

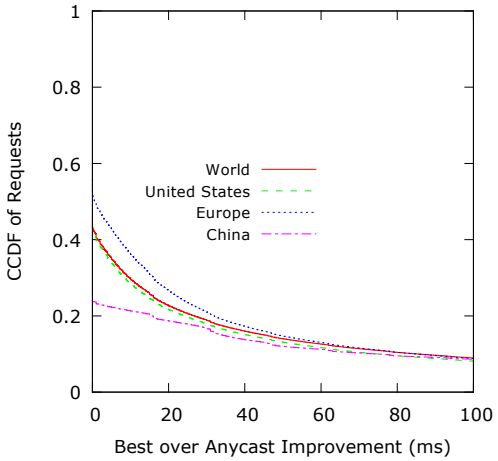

Figure 2: Regional differences in potential performance improvement over anycast.

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.

Using the search log data set, Figure 3, the lines labeled "to Front-end" shows the distribution of the distance from client to front-end for all clients in a day. One line weights clients by query volume. About 82% of clients are directed to a front-end with 2000 km while 87% of weighted clients are within 2000 km, suggesting that anycast performs better when accounting for more active clients.

The second pair of lines in Figure 3, labeled "Past Closest", show 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. An additional 37% of clients and 39% of weighted clients are directed to a front-end within 2000 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.

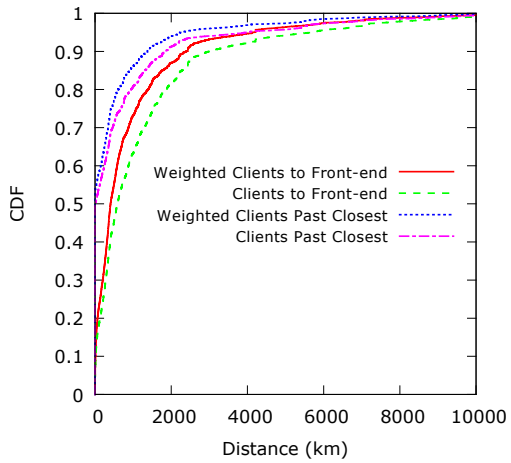

Figure 3: The distance in kilometers between clients and the front-ends they are directed to.

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

Compared to unicast, 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 build an Internet map of median latency between the prefix and all measured unicast front-ends and anycast.

Figure 4 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

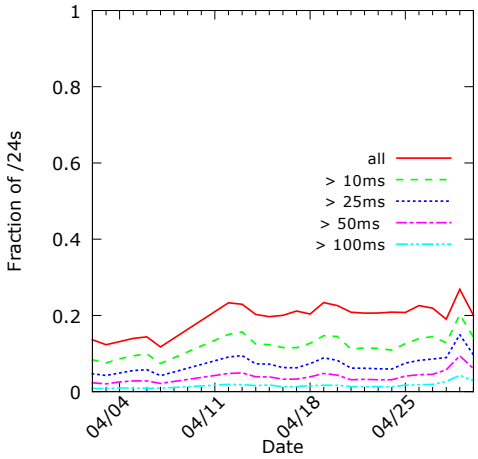

Figure 4: Daily poor-path prevalence during April 2015 showing what fraction of client /24s see different levels of latency improvement over anycast.

benefit from going to a specific unicast front-end instead of using anycast. There are 12% of clients seeing 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 4 always due to the same problematic clients?

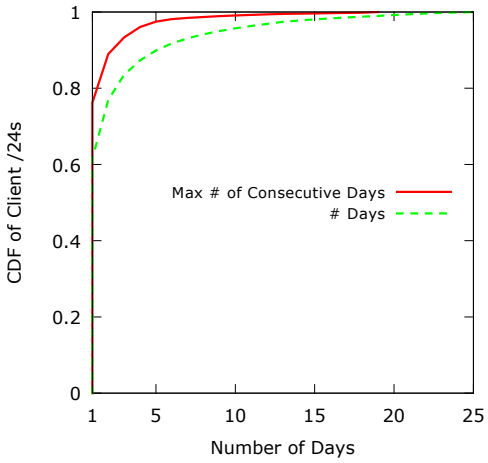

Figure 5: Poor path duration across April 2015

Figure 5 shows the duration of poor-performing paths during April 2015. The majority of /24s, 60%, categorized as poor-performing paths are short-lived, showing up for only one day over the month. 10% of /24s show poor performance for 5 days or more. Around 78% of /24s only see intermittent poor performance. Only 5% of /24s see continuous poor performance over 4 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 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: Front-end affinity refers to how "attached" particular clients are to a front-end. We examine how front-end selection changes for users over time because this could be an indicator for route stability issues which would lead to anycast performance problems.

Figure 6 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 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 [16, 26]. A likely contributing factor is that our anycast deployment is around 10 times larger than the number of instances present in K-root at the time of that work.

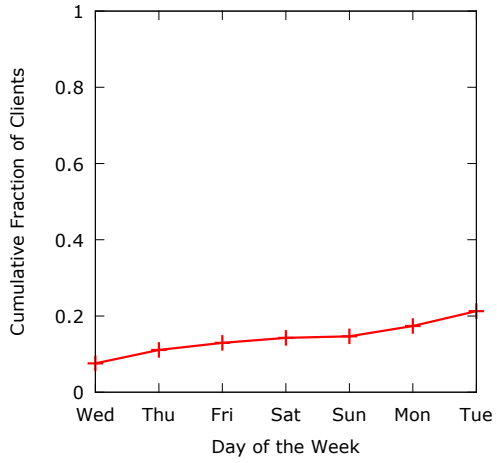

Figure 6: The cumulative fraction of clients that have changed front-ends at least once by different points in a week

Figure 7 shows the difference in distance between pairs of client-front-end switches. 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 edge-affinity, that is, they continue going to the same edge over time. For the clients that do switch edges, there is a long tail of distance between a client and switched pairs of edges.

5. 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

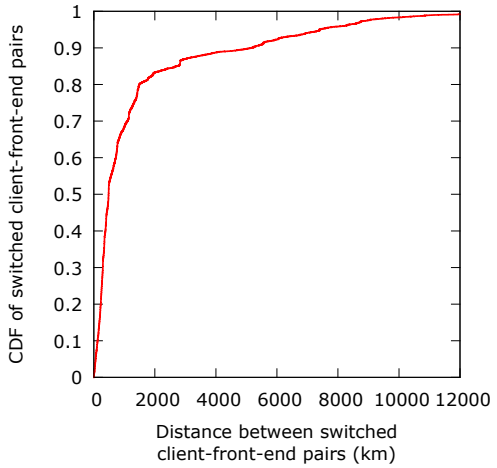

Figure 7: The difference in distance between all pairs of client-front-end switches for the 7% of clients that change front-end throughout a day

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 EDNS client-subnet-prefix). Can the system reliably determine front-ends that will perform well for the set of clients? In this section, we evaluate to what degree schemes using DNS and EDNS client-subnet-prefix can better serve clients with poor anycast performance.

We evaluate (in emulation) a prediction scheme that maps from a client group (clients of an LDNS or clients within an EDNS prefix) to its predicted best front-end. It updates its mapping every *prediction interval*, set to one day in our experiment.[1] In addition to mapping clients to their best front-ends, the system uses a beacon to measure to alternate front-ends. At the end of the day, these measurements determine the prediction for the next day. 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 an 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

[1]We use this interval because the low sampling rate at which we were permitted to inject beacon measurements into search results does not permit confidence at shorter intervals.

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.[2] Then, for a given prediction scheme, for a given client group, we pick the set of beacon measurements from the next day that probed the front-end predicted as best by the scheme. 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.

Prediction using EDNS client-subnet-prefix: The EDNS0 client-subnet-prefix extension [17] 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 8 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 front-end. 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. In aggregate, 20% of weighted clients see benefit when redirected to unicast front-ends.

LDNS-based prediction: Traditionally, DNS-based redirection can only make decisions based on a client's LDNS. In this section, we estimate the amount of performance that can be recovered with LDNS granularity. From our DNS logs, we have a map of which clients use which LDNS resolvers and for what fraction of the time. The volume for each client is split proportionally across the best front-end selected by the client's LDNS servers.

The "LDNS" lines in figure 8 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 40% of weighted /24s, we also pay a penalty where our prediction did poorly for around 25% of /24s. The results show that overall our LDNS-based prediction improved performance for 15% of /24s weighted by query volume.

Our results have shown it is possible to recover significant fractions of poor anycast performance using traditional and recent DNS techniques. 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. Potential advantages of hybrid approach include better scalability.

[2]For a given client group, we select among the front-ends with 20+ measurements from the clients.

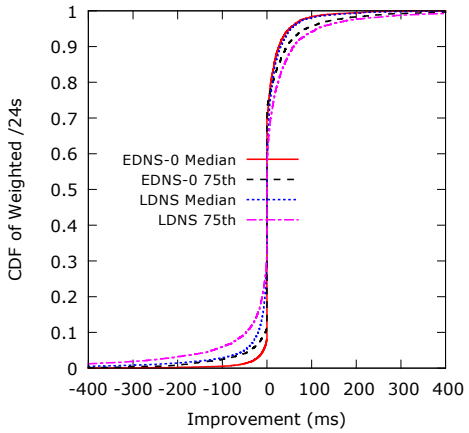

Figure 8: Improvement over anycast from making LDNS or EDNS-based decisions with prediction using 25th percentile prediction metric. Negative x-axis values show where anycast was better than our prediction. Values at 0 show when we predicted anycast was the best performing. Positive x-axis values show our improvement.

6. RELATED WORK

Most closely related to our work is from Alzoubi et al. [8, 9]. 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 and anycast. Follow up work [7] focuses on handling anycast TCP session disruption due to BGP path changes. Our work is also closely related to FastRoute [19], a system for load balancing within an anycast CDN, but doesn't address performance issues around redirection. There has been a good deal of work on improving and evaluating general CDN performance [30, 20, 29, 6, 28, 21]. 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, 23, 2, 4, 16] but less so for TCP and anycast[3]. Sarat et al. examined the performance impact of anycast on DNS across different anycast configurations [31]. Fan [18] present new methods to identify and characterize anycast nodes. There are several pieces of work describing deployment of anycast services [24, 11, 12, 22].

7. 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 centralized control but that it directs $\approx$ 20% of clients to a suboptimal front-end. We demonstrate that a simple prediction scheme allows DNS redirection to beat the suboptimal anycast performance for most of these clients.

8. REFERENCES

[1] Cloudflare. https://www.cloudflare.com/.

[2] Effects of anycast on k-root performance. NANOG 37.

[3] Operation experience with tcp and anycast. NANOG 37.

[4] Traffic source analysis of the j root anycast instances. NANOG 39.

[5] V. K. Adhikari, Y. Guo, F. Hao, V. Hilt, and Z.-L. Zhang. A tale of three cdns: An active measurement study of hulu and its cdns. In *INFOCOM WORKSHOPS '12*.

[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. Anycast cdns revisited. In *WWW '08*.

[9] 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) 2011*.

[10] Z. W. A. Q. Arvind Jain, Jatinder Mann. Resource Timing, Apr. 2015.

[11] H. Ballani and P. Francis. Towards a global ip anycast service. In *SIGCOMM '05*.

[12] H. Ballani, P. Francis, and S. Ratnasamy. A measurement-based deployment proposal for ip anycast. In *IMC '06*.

[13] P. Barber, M. Larson, M. Kosters, and P. Toscano. Life and times of j-root. In *NANOG 32*, 2004.

[14] P. Boothe and R. Bush. Anycast measurements used to highlight routing instabilities. NANOG 35.

[15] P. Boothe and R. Bush. Dns anycast stability. *19th APNIC, '05*.

[16] L. Colitti, E. Romijn, H. Uijterwaal, and A. Robachevsky. Evaluating the effects of anycast on dns root name servers. *RIPE document RIPE-393, '06*.

[17] C. Contavalli, W. van der Gaast, S. Leach, and E. Lewis. Client subnet in DNS requests, Apr. 2012. Work in progress (Internet draft draft-vandergaast-edns-client-subnet-01).

[18] X. Fan, J. Heidemann, and R. Govindan. Evaluating anycast in the domain name system. In *INFOCOM '13*.

[19] 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*.

[20] 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 2014*.

[21] M. J. Freedman, E. Freudenthal, and D. Mazieres. Democratizing content publication with coral. In *NSDI 04*.

[22] M. J. Freedman, K. Lakshminarayanan, and D. Mazières. Oasis: Anycast for any service. In *NSDI '06*.

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

[24] D. Katabi and J. Wroclawski. A framework for scalable global ip-anycast (gia). *SIGCOMM CCR 2000*.

[25] W. Li, R. K. Mok, R. K. Chang, and W. W. Fok. Appraising the delay accuracy in browser-based network measurement. In *IMC '13*.

[26] Z. Liu, B. Huffaker, M. Fomenkov, N. Brownlee, et al. Two days in the life of the dns anycast root servers. In *PAM '07*.

[27] 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*.

[28] E. Nygren, R. K. Sitaraman, and J. Sun. The akamai network: a platform for high-performance internet applications. *SIGOPS '10*.

[29] 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*.

[30] I. Poese, B. Frank, B. Ager, G. Smaragdakis, S. Uhlig, and A. Feldmann. Improving content delivery with padis. *Internet Computing, IEEE 2012*.

[31] S. Sarat, V. Pappas, and A. Terzis. On the use of anycast in dns. In *ICCCN '06*.