

Proactive and Passive Performance Optimization of IP Anycast

by

Xiao Zhang

Department of Computer Science
Duke University

Date: _____

Approved:

Bruce M. Maggs, Supervisor

Xiaowei Yang, Supervisor

Maria Gorlatova

Danyang Zhuo

Dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in the Department of Computer Science
in the Graduate School of Duke University
2023

ABSTRACT

Proactive and Passive Performance Optimization of IP
Anycast

by

Xiao Zhang

Department of Computer Science
Duke University

Date: _____

Approved:

Bruce M. Maggs, Supervisor

Xiaowei Yang, Supervisor

Maria Gorlatova

Danyang Zhuo

An abstract of a dissertation submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy in the Department of Computer Science
in the Graduate School of Duke University
2023

Copyright © 2023 by Xiao Zhang
All rights reserved

Abstract

IP Anycast, as a vital routing technique, can distribute user requests to different servers with the same IP worldwide. It can improve large-scale distributed systems performance and load balance. Nonetheless, all the sites in the anycast-based system have identical IP addresses, which makes it challenging to control the system's catchment (which site the user should go to) and results in anycast performance inefficiency.

In this thesis, we introduce two approaches to optimize the performance of IP anycast, proactively and passively. The first approach-AnyOpt, managed to build a prediction model to predict the catchment site of the user with controlled experiments and measurements with the sites. Using AnyOpt, a network operator can find a subset of anycast sites that minimizes client latency. In an experiment using 15 sites, each peering with one of six transit providers, AnyOpt predicted site catchments of 15 300 clients with 94.7% accuracy and client RTTs with a mean error of 4.6%. AnyOpt identified a subset of 12 sites, announcing to which lowers the mean RTT to clients by 33 ms compared to a greedy approach that enables the same number of sites with the lowest average unicast latency.

The second approach-regional anycast, is an approach that we found to have already been implemented by two large CDNs (Edgio and Imperva). In regional anycast, a CDN divides its content-hosting sites into different geographic regions, announces a distinct IP anycast prefix from each region, and uses DNS and IP-

geolocation to direct a client to a CDN site in the same geographic area. We aim to understand how a regional anycast CDN partitions its sites and maps its customers' clients, and how a regional anycast CDN performs compared to its global anycast counterpart. We study the deployment strategies and the performance of two CDNs (Edgio and Imperva) that currently deploy regional IP anycast. We find that both Edgio and Imperva partition their sites and clients following continent or country borders. In addition, we compare the client latency distribution in Imperva's regional anycast CDN with that in its similar-scale DNS global anycast network, after discounting the relevant deployment differences between the two networks. We find that regional anycast can effectively mitigate the pathology in global IP anycast where BGP routes a client's traffic to a distant CDN site (*e.g.*, a site in a different continent). However, DNS mapping inefficiencies, where DNS returns a sub-optimal regional IP anycast address that does not cover a client's low-latency CDN sites, can harm regional anycast's performance. Finally, using the Tangled testbed, we show what performance benefit regional IP anycast can achieve if we discount DNS mapping sub-optimality.

We also include a measurement work about the ever-increasing anycast flipping. We observe an increase in flipping over the past several years, reaching 4.4% of RIPE Atlas vantage points in 2023. We present evidence that the prevalence of anycast flipping is increasing, and for a small but not negligible portion of clients, the impact on web performance is significant.

Acknowledgements

I want to express my deepest gratitude and appreciation to Professor Bruce M. Maggs and Professor Xiaowei Yang. As my advisors, they provided unbounded support and efforts toward the completion of my research projects with high quality.

Professor Bruce provides the collaboration opportunity with Akamai Technologies, which has the most extensive network footprint in the world so far, and that inspires our excellent research motivations.

Great motivations are not enough. Professor Xiaowei Yang's engaging and approachable writing finally made our papers formed. And I also learned how vital persistence and consistency can be.

My parents, Change Zhou and Deyuan Zhang, supported my curiosity and interest in information technology with their professional experience. They taught me how to be a great engineer by their word and deed. My mother taught me how to use RJ23/RJ45 ethernet crimper, and explained to me the difference between 568A, and 568B cables, also the difference between hub, switch, and router when I was 10. My father taught me the functionality of each component in an AMD K6-2 computer and how to assemble one when I was 7.

I also want to thank my collaborative colleagues: Tim April, Aaron Atac, Danny Cooper, Tingshan Huang, Kathryn T. Kun, Kyle Schomp, Zewen Wang. It is with your help that we have done so many fantastic projects.

Contents

Abstract	iv
Acknowledgements	vi
List of Tables	xii
List of Figures	xiii
1 Introduction	1
1.1 Organization	2
2 Proactive Performance Optimization	4
2.1 Introduction	4
2.2 Background	9
2.2.1 Architecture of an Anycast Network	9
2.2.2 Motivating Examples	9
2.2.3 Anycast Configuration	11
2.3 Overview	12
2.3.1 Anycast Testbed	12
2.3.2 Measuring catchments.	14
2.3.3 Measuring RTTs.	14
2.3.4 Choosing Ping Targets	15
2.3.5 Pairwise Preference Discovery	16
2.3.6 Prediction and Optimization	16

2.3.7	Practical Challenges	17
2.3.8	No total order.	17
2.3.9	Too many experiments.	18
2.4	Design	18
2.4.1	Sufficient Conditions for Total Orders	18
2.4.2	BGP routing model.	19
2.4.3	Why a total order might not exist?	19
2.4.4	Why do we observe total orders in practice?	20
2.4.5	Practical BGP Implementation Issues	21
2.4.6	Arrival orders of BGP advertisements.	22
2.4.7	Multi-path routing.	23
2.4.8	Two-level Preference Discovery	23
2.4.9	Provider-level preference discovery.	24
2.4.10	Site-level preference discovery.	24
2.4.11	Incorporating Peers	25
2.4.12	Putting it Together	27
2.4.13	Analysis.	28
2.5	Performance Evaluation	29
2.5.1	Pairwise Preference Discovery	30
2.5.2	Inter-AS experiments & impact of announcement order.	30
2.5.3	Intra-AS experiments.	33
2.5.4	Catchment Prediction	34
2.5.5	Takeaways.	36
2.5.6	Performance Optimization	37
2.5.7	Incorporating Peering Links	40

2.6	Limitations and Future Work	42
2.7	Related Work	44
2.8	Conclusion	46
2.9	ACKNOWLEDGMENTS	47
2.10	Appendix	47
2.10.1	No Consistent Linear Order	48
2.10.2	Can Preserve a Linear Order, but Not Predictive from the Linear Order	48
3	Passive Performance Optimization	52
3.1	Introduction	52
3.2	Background and Motivation	56
3.2.1	The Catchment Inefficiency Problem	56
3.2.2	Challenges	57
3.2.3	Regional IP Anycast	58
3.2.4	Motivation	58
3.3	Measurement Infrastructure	59
3.3.1	RIPE Atlas	59
3.3.2	The Tangled Testbed	61
3.4	Deployments	61
3.4.1	Identifying Regional IP Anycast CDNs	61
3.4.2	Customers of Regional IP Anycast CDNs	63
3.4.3	Client Partitions	65
3.4.4	CDN Site Partitions	67
3.4.5	Reachability of Regional IP addresses	70
3.5	Performance	71
3.5.1	DNS Mapping Efficiency	71

3.5.2	Client Latency	75
3.5.3	Regional vs. Global Anycast	78
3.6	Insights	84
3.7	Potentials	87
3.7.1	Latency-based Regional Partition	87
3.7.2	Global vs. Regional on Tangled	88
3.8	Related Work	89
3.9	Conclusion	91
3.10	Appendix	91
3.10.1	IP Geolocating p-hops	91
3.10.2	Collection of CDN's Technical documents	93
3.10.3	Same-Site Latency Comparison	94
4	Anycast Flipping Detection	96
4.1	Introduction	96
4.2	Background	100
4.3	Prevalance & Impact: Root DNS	102
4.3.1	RIPE Atlas's DNS CHAOS Queries	102
4.3.2	Root DNS Site Discovery	104
4.3.3	Geo-Locate a Catchment Site	105
4.3.4	Estimate Anycast Flipping	106
4.3.5	Longitudinal View of Anycast Flipping	106
4.3.6	Does Flipping Break TCP Connections?	108
4.3.7	Impact on RTTs	109
4.4	Prevalance & Impact: Anycast CDN	110
4.4.1	Infrastructure	111

4.4.2	Methodology	112
4.4.3	Prevalence	116
4.4.4	Impact on RTTs	117
4.5	Impact on Web Performance	118
4.5.1	Infrastructure	119
4.5.2	Mock Webpage Results	121
4.5.3	Popular Website Results	124
4.6	Related Work	127
4.7	Conclusions and Future work	130
4.8	Appendix	132
4.8.1	Geocodes Used by Different Roots	132
5	Conclusion	135
	Bibliography	136
	Biography	163

List of Tables

2.1	Locations of the 15 anycast sites along with the transit providers and counts of peers at each location.	12
2.2	Path and Anycast Site Finally Chosen in the Counter Case I	49
2.3	Path and Anycast Site Finally Chosen in the Counter Case II	49
2.4	Path and Anycast Site Finally Chosen in the Counter Case III	50
2.5	Path and Anycast Site Finally Chosen in the Counter Case IV	51
3.1	The number of sites in each geographic area of different networks (<i>EG/IM-Pub: Edgio and Imperva's Published sites</i>).	68
3.2	DNS Mapping Performance	72
3.3	Tail latency comparison	78
3.4	Performance Categorization	81
4.1	The percentage of RIPE Atlas probes that experience anycast flipping over the past seven years.	107
4.2	The percentage of RIPE Atlas probes that experience anycast flipping over the past seven years (continue).	108
4.3	Anycast Flipping among Roots	114
4.4	Naming schemes used by different operators	133
4.5	Naming schemes used by different operators (continue)	134

List of Figures

2.1	The architecture of an anycast network.	10
2.2	This figure illustrates the testbed we use for RTT and catchment measurements.	13
2.3	Counter example of preference order	20
2.4	A significant fraction of ping targets switch their preferences based on the order in which they receive the anycast prefix announcements. . .	30
2.5	Transit provider preference order fraction	31
2.6	Ping target fraction	32
2.7	Catchment prediction accuracy	34
2.8	Absolute RTT estimation error	35
2.9	Relative RTT estimation error	36
2.10	AnyOpt-optimized configuration substantially outperforms other approaches in terms of RTT.	38
2.11	CDF of a peer AS’s catchment when adding a peering link to AnyOpt’s transit-only configuration.	39
2.12	Mean RTT changes when adding a peering link to AnyOpt’s transit-only configuration.	40
2.13	CDF of client RTTs after incorporating peering links.	41
2.14	The Case that do not Preserve Linear Preference	48
2.15	The Counter Example I	49
2.16	The Counter Example II	50
2.17	The Counter Example III	51

3.1	Regional benefit example	56
3.2	Edgio-3 (www.straitstimes.com) divides its sites into three regions . . .	62
3.3	Edgio-4 (www.asus.com) divides its sites into four regions	63
3.4	Imperva-6 (www.stamps.com) divides its sites into six regions	64
3.5	P-hop location resolution	68
3.6	Edgio-3 vs. Edgio-4	75
3.7	Imperva-6	76
3.8	Imperva-6 vs. Imperva-NS with same config	77
3.9	CDFs of RTT and distance differences between regional and global anycast.	80
3.10	ReOpt’s site and client partitions	82
3.11	Regional w/ and w/o DNS configuration	82
3.12	Regional w/ ReOpt partitions vs. Global	83
3.13	Route server and public peering	85
3.14	Lack of performance hints	86
3.15	CDFs of RTTs of the same-config probes	94
4.1	An IP anycast deployment.	100
4.2	Flipping at the border between two ASes	101
4.3	Flipping internal to an AS	103
4.4	The percentage of RIPE Atlas probes that experience anycast flipping in the past seven years.	109
4.5	Longest Detour	110
4.6	A,B,C,D,E Roots	111
4.7	F,G,H,I,J Roots	112
4.8	K,L,M Roots	113
4.9	Fraction of requests reaching the mode site	116

4.10	# sites reached per proxy node	117
4.11	ΔRTT of proxy nodes that flip between sites	118
4.12	HTTP/1.1 metrics	120
4.13	HTTP/2 metrics	121
4.14	HTTP/1.1 Slowdown Time (FCP median)	122
4.15	HTTP/2 Slowdown Time (FCP median)	123
4.16	HTTP/2 Slowdown Ratio (FCP median)	124
4.17	HTTP/1.1 Slowdown Time (FCP 90th Percentile)	125
4.18	HTTP/2 Slowdown Time (FCP 90th Percentile)	126
4.19	HTTP/2 Slowdown Ratio (FCP 90th Percentile)	127
4.20	HTTP/1.1 Slowdown Time (PLT median)	128
4.21	HTTP/2 Slowdown Time (PLT median)	129
4.22	HTTP/2 Slowdown Ratio (PLT median)	130
4.23	HTTP/1.1 Slowdown Time (PLT 90th Percentile)	131
4.24	HTTP/2 Slowdown Time (PLT 90th Percentile)	132
4.25	HTTP/2 Slowdown Ratio (PLT 90th Percentile)	133

1

Introduction

IP anycast is an essential routing technique many large organizations and corporations advocate for building large-scale distributed systems. Content delivery networks, Domain name systems use IP anycast to distribute user requests globally to different sites deployed at different locations.

In an anycast system, multiple sites could be deployed at different locations. Then, the same IP is configured to those servers/replicas of sites and announced to different peers at that site. Then, the client's request to that anycast IP, will be routed to the 'nearest' site based on the route selection result along the path. Therefore, different sites in this anycast system will split the global network into different catchments. In the client networks that are covered by the same anycast catchment, all the requests will be forwarded to the site that creates this catchment.

IP anycast is expected to direct users to the nearby and performant site to improve user experience and balance the load. However, the unpredictability of the catchment made it a challenging issue.

In this thesis, we shed light on the approaches to optimizing the performance of anycast system with proposed and existing methods and with the measurement of

the ever-increasing anycast flipping.

1.1 Organization

The rest of this thesis is organized as follows:

- In Chapter 2, we propose an approach to model and predict the user’s catchment site with a linear preference model. We can predict any user’s most preferred site for arbitrary anycast configuration with controlled pair-wise BGP experiments and measurements. Thus we could select the best anycast configuration (e.g., a set of anycast sites) to provide the anycast service.
- In Chapter 3, we show another existing approach that passively optimizes the anycast performance. Different from our proactive approach proposed in Chapter 3, the regional anycast divides its content-hosting sites into different geographic regions, announces a distinct IP anycast prefix from each region, and uses DNS and IP geolocation to direct a client to a CDN site in the same geographic area. It keeps global anycast’s ease of use and provides finer-grained control over the region. Our work shows how regional anycast can outperform its counter-global anycast network with the help of DNS.
- In Chapter 4, we measure the prevalence of anycast flipping, which means multiple anycast sites could be reached by the same user under the existing anycast network. Seven years ago, Wei and Heidemann work first found it rarely happened. With our longitudinal study, however, we observe an increase in flipping over the past several years, reaching 4.4% of RIPE Atlas vantage points in 2023. Besides the measurement of DNS, we also incorporated residential proxy measurement to an existing anycast CDN, the result shows an even larger amount of residential users could be affected. We also use MAHIMAHl to emulate

the web browsing performance with typical anycast flipping settings, the result shows severe impact to the user.

Proactive Performance Optimization

2.1 Introduction

IP anycast [PMM93, Met02] is the practice of announcing the same IP address prefix from multiple network locations, and it is commonly used for load balancing and latency reduction. In part due to its inherent support in the routing system and potential for improving performance, anycast is used in large and popular services such as DNS (to distribute DNS query load [LHF⁺07a, SBK⁺20a]), content delivery networks (CDNs) (to reduce latency between servers and clients [CFKB⁺15a, dWARD20]), and distributed denial of service (DDoS) mitigation services (to distribute and scrub attack traffic loads [MSH⁺16a, Tec20]). A key challenge for deploying anycast services effectively is that mappings between client networks and anycast sites (i.e., anycast catchments) are determined by BGP’s policy-based routing decisions rather than service providers’ goals such as minimizing latency and balancing load. In fact, several measurement studies have revealed that some anycast catchments exhibit unexpectedly inflated latency [BFR06b], and increasing the number of anycast sites in a deployment (in an attempt to reduce the distance between clients and sites)

counter-intuitively increases the average latency for clients and disrupts attempts to balance load [LLSB18a].

As a result, managing anycast deployments is a challenging task that requires expert knowledge and continuous intervention in response to BGP path changes, regular maintenance [dVARD20], or DDoS attacks [MSH⁺16a]. Network operators lack tools that can accurately predict the system performance under different anycast configurations (i.e., the set of sites making announcements and the next-hop neighbors to whom the prefix is announced). Since BGP paths are determined by non-public policy information, such tools will require measurements or inferences for prediction. A naïve approach to measuring the impact of all potential announcements would require probing that scales exponentially with the number of sites under consideration. Using inferred topologies [SK19] to predict catchments can limit this cost, but it may introduce imprecision because of missing information in topology models and how BGP routers break ties among equally preferred paths.

In this paper, we address the above problems by using theoretical foundations to develop efficient measurement, prediction, and optimization techniques that allow an anycast operator to optimize a deployment for low latency while balancing load. This problem is important because latency is critical to the revenue generation of many Internet services [Yoa19]. Specifically, we present an experimental approach, AnyOpt, for predicting anycast catchments. A service operator can use AnyOpt to optimize an anycast network’s deployment or dynamically reconfigure the network.

The key, empirically informed, insight that enables efficient catchment prediction is that most client networks, when given an option between any two (of potentially many) anycast sites, will consistently prefer one or the other. Further, we find that when considering *all pairs* of anycast sites, the set of pairwise preferences for a client network often forms a *total order*. This total order makes it straightforward to predict a site’s catchment when we enable any subset of the anycast sites, as

most client networks will consistently pick their most preferred sites in the subset. Furthermore, we observe that if a client network has a consistent total order among the anycast sites, we can map the anycast optimization problem to the Simple Plant Location with Preference Orderings [HP87] problem and solve it offline to find the subset of anycast sites that achieve the lowest overall latency.

Making AnyOpt accurate and efficient, however, requires addressing two key challenges. First, we find that not all client networks exhibit the consistent preferences that enable our approach. We use both theoretical analyses and measurements to understand why this problem occurs and whether consistent preference orders can predict a site’s catchment. Our analyses reveal that a client network may not have a total order among preferences for anycast sites when autonomous systems (ASes) on the path of a BGP advertisement assign different local preferences to the advertisement. We prove sufficient conditions under which pairwise measurements yield a consistent total order and the total order is predictive of a client network’s catchment. One example that meets these conditions is that we only announce an anycast prefix to tier-1 ISPs, and we adopt this setup for our real-world anycast testbed. Our experiments show that another cause of inconsistent preference orderings is a BGP implementation choice where ties between equally preferred paths are broken by the order in which a router receives BGP advertisements, which is not part of the BGP standard [RLH06] but is implemented by most deployed routers (e.g. Cisco [cis16] and Jupiter [jun20]). Once we take into account both the policy-induced and implementation-induced inconsistent preference orders in AnyOpt, we show that we can expect consistent pairwise preferences. Then, we use the total orders constructed from those pairwise preferences to predict anycast catchments.

Second, we alleviate the issue of scaling AnyOpt measurements to large anycast deployments, e.g., those with hundreds or more sites [CAJ⁺15a]. For a deployment of this size, pairwise preference discovery experiments become impractical. For ex-

ample, for an anycast network of 100 sites, if we space each pairwise experiment by two hours, which is necessary to avoid BGP instability, it would take years to finish all pairwise experiments. To address this challenge, we design AnyOpt to take a two-level approach to predict anycast catchments. The routing structure of the Internet makes the inter-AS and intra-AS anycast catchments two separate processes, where BGP determines the inter-AS catchments and the (interior) routing inside an AS determines the intra-AS catchments. Our experiments show that a site’s catchment at the AS level remains stable when an anycast site is enabled or disabled within the same AS. Therefore, we can use pairwise experiments to discover client networks’ AS-level preferences by choosing one representative site in each tier-1 AS that the anycast network connects to and run site-level pairwise experiments for sites within the same AS. If the latter is still prohibitive for a large network, we discuss a heuristic approach that might further reduce the number of BGP experiments needed for catchment prediction.

Our experiments on a real-world testbed show that AnyOpt can accurately predict anycast catchments and optimize client latency distribution, when announcing an anycast prefix to only tier-1 providers. We start with tier-1 network providers because they act as the backbone network that delivers the majority of the traffic for the testbed anycast network. To extend beyond the tier-1 providers, we adopt a heuristic to determine the impact of announcing via a peering link while simultaneously announcing to the tier-1 providers (§ 2.4.11). Our anycast testbed has 15 sites and connects with six tier-1 ASes. In the evaluation of transit-only configuration, we randomly choose an anycast configuration, predict its catchments and average latency to client networks, then deploy the configuration, and measure its actual catchments and average latency. Then we repeated this for 38 times. We find that AnyOpt predicts catchments with 94.7% accuracy and average RTTs with a mean error of 4.6%. In the offline configuration searching, AnyOpt identifies a 12-site low-

est latency configuration that reduces the average client latency by more than 30 ms compared to configurations that greedily include sites with the lowest average unicast latency or randomly chosen sites. For the peering links, we also iterated through 104 peering links in the testbed and identified 47 peering links that can improve performance; more specifically, we find that including peering links in the 12-site lowest-latency configuration can further reduce the mean latency by 5 ms to 7 ms.

AnyOpt represents a crucial first step towards predicting and optimizing the performance of an anycast network. Specifically, this work makes the following contributions:

- (1) We propose AnyOpt, an empirical approach that uses BGP measurements to reveal a client network’s preferences between any two anycast sites, and then uses these to predict and optimize anycast network performance. We report for the first time the extent to which BGP announcement arrival orders affect anycast catchments at scale and develop a technique to incorporate them into the catchment prediction. We use two-level prediction techniques to reduce the number of required experiments.
- (2) We analyze the theoretical underpinnings for the heuristic approach and prove sufficient conditions for this approach to work.
- (3) We use a real-world anycast testbed of a large content delivery network to evaluate AnyOpt. Our experiments show that AnyOpt can predict anycast performance accurately and can reduce the average latency to client networks by as much as 33 ms (32%) compared to greedy approaches.

Ethical considerations. Active measurements such as issuing pings and BGP announcements can cause extra load on the Internet infrastructure. As discussed later

in the paper, we mitigate these concerns by gathering our measurements at reasonably low rates, and target our measurements at routers (not end hosts). Our BGP announcements use only prefixes that we control and only our AS number in our announcements. The anycast prefixes we use do not serve any clients. This work raises no other ethical issues.

2.2 Background

In this section, we briefly describe the architecture of an anycast network, define the terms we use, and use real-world anycast systems to motivate AnyOpt’s design.

2.2.1 Architecture of an Anycast Network

Figure 4.1 illustrates the architecture of an anycast network. A service provider such as a CDN or a DDoS mitigation provider has servers that receive anycast traffic deployed at multiple locations. These servers offer services such as traffic scrubbing or caching. We refer to each location where these servers are deployed as an *anycast site*. A site has an onsite router that connects to one or more ASes. We refer to each BGP connection to an outside AS as an *ingress point*. In Figure 4.1, the simple anycast network has three sites, each having two or three ingress points. We refer to the set of end systems reaching one site as the *catchment* of the site. Figure 4.1 groups each site’s catchment within an oval shape and marks the catchment’s ingress AS/connection with the same line type.

2.2.2 Motivating Examples

A key motivating application of our work is the configuration management of systems such as Akamai DNS [SBK⁺20a] or an anycast-based CDN [CFKB⁺15a, dVARD20]. Akamai DNS is one of the world’s largest authoritative DNS systems, serving millions of queries per second from a few hundred sites that host 24 distinct anycast prefixes.

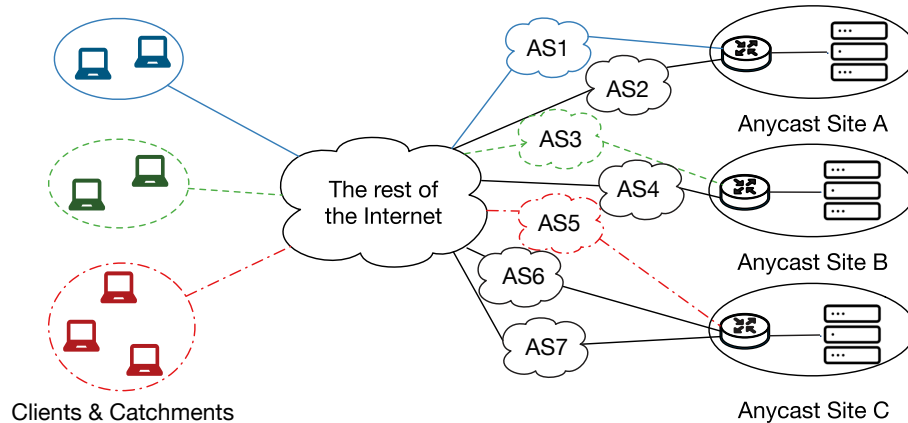


FIGURE 2.1: The architecture of an anycast network.

Each anycast prefix is hosted from a subset of about 30 sites that form an “anycast cloud”. Each domain name hosted on Akamai DNS is assigned to a delegation set of about 6 anycast prefixes. When a recursive DNS resolver (i.e., client) requests an authoritative translation of a domain name, it sends the request to an anycast prefix that is in the delegation set of that domain name. The request is then routed to a site within that prefix’s anycast cloud by BGP. That site then responds with the answer.

The key challenge in configuring Akamai DNS is assigning each of the 24 anycast prefixes to a subset of sites such that the average query response latency experienced by the resolvers is minimized. As network conditions (e.g., routing policy or load) change, the subset of sites that host each anycast prefix must be recomputed to maintain optimal anycast performance. Since the number of ways to configure an anycast cloud is exponentially large, it is infeasible to predict the catchments of sites accurately, and, consequently, impractical to estimate the query latency achieved in each configuration. The state-of-the-art for configuring large anycast networks such as Akamai DNS is based on Monte Carlo simulations [SBK⁺20a]. Our work is focused on improving the state-of-the-art in configuring anycast networks. AnyOpt assists the problem of optimally configuring an anycast cloud using a principled measure-

model-and-optimize approach that is directly applicable to real-world systems such as Akamai DNS.

An anycast-based CDN faces a similar configuration challenge. For a CDN service provider, the latency between a client and an edge server can have a multiplicative effect on page-load times, given the many round-trips typically required to download various resources. Therefore, reducing latency by even tens of milliseconds can result in a substantial reduction of page-load times [KLA⁺21]. Simply adding more anycast sites, however, does not necessarily improve the mean latency between the clients and an anycast network [LLSB18a]. Even if some sites offer poor performance for clients, BGP may prefer these sites to others for policy reasons. In such cases, AnyOpt can reliably identify which anycast sites improve performance, obviating manual interventions.

2.2.3 Anycast Configuration

Although an anycast network cannot control the catchment of a site, it can “shape” the catchment with three control knobs: (1) the sites from which it announces an anycast prefix, (2) the ASes to which it announces the prefix at a particular site, and (3) the BGP path attributes it uses when announcing the prefix. Specifically, if we use S to denote the set of sites an anycast network has (or considers to open), the network can choose to announce an anycast prefix from any subset of S . For each anycast site s_i , let’s denote the set of ASes it connects to as P_{s_i} . The service provider can choose any subset of P_{s_i} to announce the prefix. For each BGP announcement, the network can vary the parameters associated with the announcement, including the Multiple Exit Discriminator (MED) and the AS path length.

There are, hence, more than $2^{\sum |P_{s_i}|}$ possible ways to configure an anycast network. As a first step, in this work, we explore how an anycast network can optimize its performance by finding (a) from which subset of sites to announce the anycast

Table 2.1: Locations of the 15 anycast sites along with the transit providers and counts of peers at each location.

Site	Location	Transit	#peers
1	Atlanta	<i>Telia</i>	4
2	Amsterdam	<i>Telia</i>	1
3	Los Angeles	<i>Zayo</i>	6
4	Singapore	<i>TATA</i>	15
5	London	<i>GTT</i>	14
6	Tokyo	<i>NTT</i>	3
7	Osaka	<i>NTT</i>	4
8	Los Angeles	<i>Zayo</i>	4
9	Miami	<i>NTT</i>	7
10	London	<i>Sparkle</i>	2
11	Newark	<i>NTT</i>	7
12	Stockholm	<i>Telia</i>	14
13	Toronto	<i>TATA</i>	9
14	São Paulo	<i>Sparkle</i>	9
15	Chicago	<i>GTT</i>	5

prefix and (b) to which ASes at each site to announce the anycast prefix. We assume that an anycast network sets the path attributes to default values when it announces an anycast prefix. We call a site or an AS as “enabled,” when it is chosen to announce an anycast prefix. We refer to the combination of the chosen subset of sites and the chosen ASes at each site as an anycast configuration.

2.3 Overview

In this section, we describe the anycast testbed used in this work, the experiments for discovering a client network’s pairwise preferences, and the high-level idea of using the preference orders to predict and optimize the performance of an anycast network.

2.3.1 Anycast Testbed

Our testbed consists of an instance of GoBGP (version 2.14.0) [NTT20], an open-source BGP implementation, running on an Ubuntu (18.04.3 LTS) server with 4

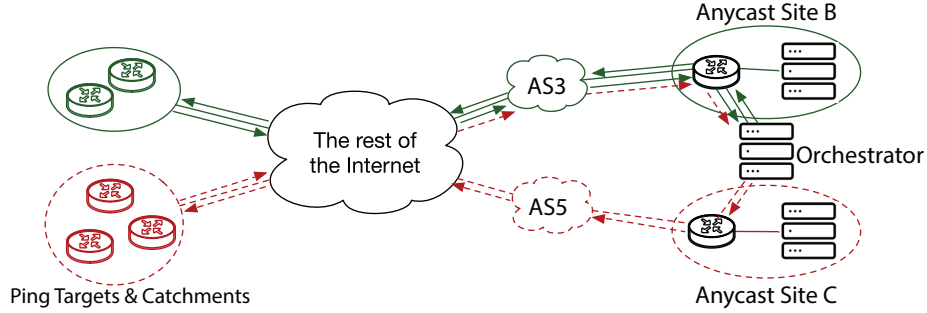


FIGURE 2.2: This figure illustrates the testbed we use for RTT and catchment measurements.

cores and 16 GB of RAM. The GoBGP instance uses generic routing encapsulation (GRE) tunnels [FLH⁺00] to peer with a large CDN’s routers at different locations as described in Table 2.1. The routers at different locations serve as anycast sites. Each anycast site has a tier-1 transit provider to ensure global reachability, i.e., any client or end-user in the Internet can reach the anycast site. In addition to the transit provider, each site peers with a few other ASes (Table 2.1), including some under a settlement-free policy, i.e., where neither network pays the other for transit. We launch all active experiments and collect our measurements from the Ubuntu server, which we henceforth refer to as the *orchestrator*.

We deploy an anycast configuration as follows. First, we establish BGP sessions between the orchestrator and the routers at chosen anycast sites. Then, we program GoBGP [NTT20] to announce an anycast prefix assigned to us via the BGP sessions between the orchestrator and the site routers. The router at an anycast site likely peers with multiple other neighbors. We use, hence, BGP’s community attribute to control to which next-hop neighbors the router should advertise our anycast prefix. We can choose, therefore, to advertise to a site’s transit provider or any chosen peer by appropriately setting the community attribute.

We develop a measurement tool that is similar to Verfploeter [dVdOSH⁺17a]. We run this tool at the orchestrator to measure the catchments. We also improve

the tool to measure the RTT between any client and any anycast site as we soon describe.

2.3.2 *Measuring catchments.*

For a given anycast configuration, we measure each site’s catchment. We use these measurements in two ways. First, we use them to determine a client network’s preference between two anycast sites, henceforth referred to as *pairwise preference*. Second, we compare the predicted catchments with the empirically observed catchments. To gather the measurements, we send ICMP requests [Pos81] from the orchestrator to a large number of ping targets, which are routers in different client networks chosen by the CDN hosting our experiments to evaluate its network’s global performance. We set the source address of each ICMP request to an IP anycast address that we advertise and its destination address to a target’s IP address. When a target responds to this request, i.e., to the anycast address, it will be routed to an enabled anycast site. The router at the site will then forward that reply to the orchestrator, via a preconfigured GRE tunnel. The GRE tunnel carrying the reply, thus, identifies the target’s catchment site.

2.3.3 *Measuring RTTs.*

We can measure the RTT from the orchestrator to any target, but, for predicting and optimizing anycast performance, we must measure the RTT between the anycast site and the target. Rather than indirectly estimating the RTT between a site and a target from the physical distance between the two or approximating the RTT through appropriate proxies as in King [GSG02a], we use the following approach. First, we announce an anycast prefix from only one anycast site and send ICMP requests via the GRE tunnel connected to that site. We include a timestamp in the request for RTT measurements. Second, when the orchestrator receives a reply from

a target, we subtract the echoed timestamp from the current time to calculate the RTT between the orchestrator and the target. Third, we periodically measure the “tunnel” RTT between the orchestrator and each anycast site. Finally, we subtract from the measured RTT between the orchestrator and a target, the corresponding tunnel RTT, i.e., the RTT between the orchestrator and the site through which the orchestrator received the target’s ICMP responses. For each RTT measurement, we repeat the ICMP requests seven times and use the median value (to filter outliers) as the RTT between the concerned site and the target. If the link experience high packet loss rates, we can still sample a median RTT from at least three valid responses.

As an example, in Figure 2.2, suppose we only announce our prefix to AS3. Even though we send out the ICMP requests to the targets from both anycast site B and anycast site C, the ICMP replies will only return to anycast site B. Therefore, we can measure the RTT between any target and the orchestrator. By subtracting the tunnel RTT from the orchestrator to site B, we obtain the RTT between a target and site B.

2.3.4 Choosing Ping Targets

To understand the impact of anycast configurations on client networks’ performances, we conduct active measurements (with ICMP probes). The targets of these ICMP measurements are routers in or near the targeted client networks.

To select our targets for measurements, we follow an approach used by the CDN hosting our testbed [LSLS07]. Specifically, we merge the network paths from the end-users to a CDN’s edge server into a tree, rooted at the edge server. We then pick a common ancestor that is closest to the end-users. In our active measurements, we ping such targets from diverse networks to obtain a reasonable approximation of the global performance of end-users. Additionally, the targets also help us avoid sending ping probes to real end-users. Our target set contains 15 300 IP addresses

from 12 143 /24 network prefixes or 5317 ASes. Each target is representative of one or more client networks.

2.3.5 Pairwise Preference Discovery

We conduct pairwise BGP experiments to elicit a client network’s preference orders. For each experiment, we choose two sites s_i and s_j from the available anycast sites for comparison. We announce an IP anycast prefix from these two sites to *only* their corresponding transit providers, for reasons we soon describe in §2.4.1. If a ping target’s response reaches site s_i instead of site s_j , we record that the client network prefers s_i to s_j . By pinging all targets in one experiment, we obtain all client networks’ preferences between s_i and s_j .

2.3.6 Prediction and Optimization

With the RTT measurements and pairwise preference experiment results, we can predict an anycast configuration’s performance and choose an optimal configuration. If a client network’s set of pairwise preferences has no cycles, we can construct a total preference order for the network. For any subset of anycast sites enabled, we predict the client network’s catchment site as its most preferred site within that subset. A client network’s pairwise preferences may, however, not form a total order, and we discuss why this situation may happen in §2.4.

Given the RTTs and preference predictions, we can map an anycast optimization problem to the Simple Plant Location Problem with clients’ Preference Orderings (SPLPO) [HP87] for optimization. SPLPO is an extension to the well-known (un-capacitated) plant location problem [CNW83]. It considers the problem of how to open facilities that have the overall lowest cost when each client has a preference order among the set of possible facility locations. If we consider a “facility location” as an anycast site, and use the *RTT* as the cost, then anycast performance

optimization problem becomes exactly the SPLPO problem. The SPLPO problem is NP-hard [ACG⁺12], and we show in Appendix ?? that even approximating the minimum cost of SPLPO is NP-hard.

A network operator can, however, solve or approximate the optimization problem using offline simulations. When the number of anycast sites is small, he or she can solve it exhaustively. When the number of sites is large, he or she may not find the (theoretically) optimal configuration, but he or she can find a configuration that has the best performance among all configurations she simulates.

If an anycast network has a total of $|S|$ sites, each having one transit provider as in our testbed, then to optimize or predict an anycast network’s performance, a network operator needs to run $O(|S|^2)$ pairwise experiments to obtain each client network’s total preference order and $O(|S|)$ experiments to obtain a client network’s RTT to each site. In contrast, if we do not employ the prediction or optimization technique, the operator needs to deploy $O(2^{|S|})$ anycast configurations to measure and compare the performance of each configuration. We formally describe the optimization model in Appendix ?? and show that the model can optimize for latency while meeting the load constraints of a site.

2.3.7 Practical Challenges

We have outlined the high-level idea behind AnyOpt’s design. However, to make it useful, we must address the following challenges.

2.3.8 No total order.

When we perform the pairwise preference discovery experiments, a client network may not exhibit consistent pairwise preferences that form a total order. Without a total order, we cannot predict a site’s catchment for an arbitrary anycast configuration.

To address this challenge, we formally analyze the sufficient routing conditions under which a client network has a total order over a set of anycast sites and the total order predicts a site’s catchment (§2.4.1). We modify the pairwise experiments described in this section to induce and discover a client network’s total order.

2.3.9 *Too many experiments.*

A naïve approach for pairwise preference discovery requires at least $O(|S|^2)$ BGP experiments. As it takes on the order of minutes for BGP to converge and routers implement route damping for frequently changing prefixes, conducting such experiments at scale may become impractical. Solving the optimization problem using an exhaustive search also becomes infeasible.

We reduce the number of experiments by separating AS-level catchment prediction from intra-AS catchment prediction (§ 2.4.8). This technique reduces the number of pairwise BGP experiments from $O(|S|^2)$ to $O(|I|^2) + O(\text{avgSite}^2 \times |I|)$, where I is the set of transit ISPs an anycast network connects to and avgSite is the average number of sites connecting to a transit provider. For a large anycast network, this number of experiments may still be infeasible. We, hence, describe a heuristic method to approximate a client network’s intra-AS site preferences. This heuristic can eliminate the need for intra-AS pairwise preference discovery experiments.

2.4 Design

Below, we discuss how we address the practical challenges that AnyOptfaces.

2.4.1 *Sufficient Conditions for Total Orders*

First, we investigate why client networks exhibit a total preference order and why this order can be used to predict anycast catchments. With this understanding, we can determine whether our experimental approach is generally applicable to other

networks.

2.4.2 BGP routing model.

We analyze anycast routing using the Gao-Rexford BGP routing model [GR01]. For simplicity, we consider two kinds of contractual relationships: provider-customer and peer-to-peer. In the former, a provider AS advertises a route received from a customer AS to all its other neighbors, while in the latter, a peer only advertises another peer's routes to its customers.

When a BGP router receives different route advertisements to the same prefix from its neighbors, it chooses the “best” path for reaching the prefix and advertises only the best path to its neighbors based on its export policies. The algorithm for determining the best path works as follows [RLH06]. When a router compares two paths, it lexicographically compares two tuples, each consisting of an ordered list of attributes of the corresponding paths. The first element in the tuple of path attributes is local preference (`LOC_PREF`). An AS would generally prefer an economically profitable route. Therefore, under common BGP policies, an AS would prefer a customer route to a peer route and prefer a peer route to a provider route. When the routes have the same `LOC_PREF`, BGP breaks ties using the following path attributes, in the given order: AS path length, origin of prefix, `MED`, type of BGP session, interior cost, router Id, and neighbor address.

2.4.3 Why a total order might not exist?

As a BGP path passes along from one AS to another, each AS may rank the paths differently. An AS may prefer a path with a longer AS path length, while a downstream AS may prefer one with a shorter AS path length (all due to differences in `LOC_PREF` values at the two ASes). Suppose that A , B , and C are three anycast sites, and dst is a client network that receives anycast announcements (Figure 2.17).

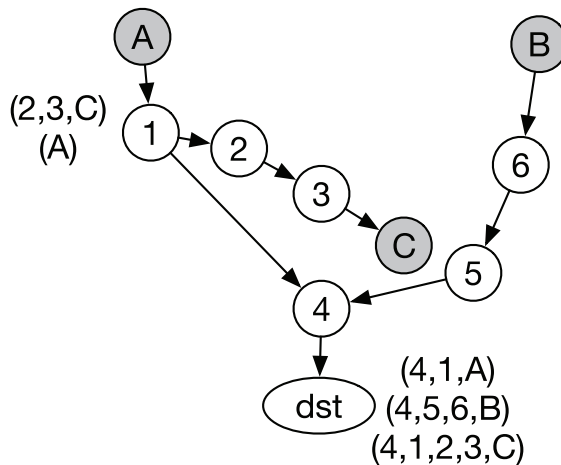


FIGURE 2.3: This example explains why a client network (*dst*) may not exhibit a total preference order among anycast sites *A*, *B*, and *C*. Arrows point from providers to customers. AS 1 prefers the path originated at site *C*, as it is a customer router, while AS 4 prefers the path from *A*, as it has a shorter AS path.

Each circle represents an AS and an arrowed line points from a provider AS to a customer AS. To elicit the preferences between the sites *A* and *B*, we use *A* and *B* to announce our anycast prefix. The client network *dst* will choose the path originated from site *A*, as both paths from *A* and *B* are provider routes, and the path from *A* has a shorter AS path. So we observe $A >_{dst} B$, where the operator $>_{dst}$ denotes “preferred by *dst*.” When we compare the preferences between *A* and *C*, *dst* will choose *C* (i.e., $C >_{dst} A$), since AS 1 prefers a customer route to a provider route and will advertise only the path from *C* to AS 4. Finally, when we compare *B* and *C*, *dst* will choose *B* (i.e., $B >_{dst} C$), as it has the same LOC_PREF as the path from *C* but with a shorter AS path. This scenario leads to cyclic pairwise preferences—no total order.

2.4.4 Why do we observe total orders in practice?

If a client network might not exhibit a total preference order under the common BGP model, why do our experiments observe so many instances of a consistent total order? To answer this question, we focus on the case where anycast sites

peer only with tier-1 networks and make two assumptions: (a) any network that has settlement-free peering with a tier-1 network has settlement-free peering with all tier-1 networks; and (b) valley-free routing [GR01] holds. Then, if a network receives one or more announcements for an anycast prefix, then all of them should come from either peers (if the receiving network is a tier-1 network) or from providers (if the receiving network is not a tier-1 network). In selecting a path, a non-tier-1 network will, hence, only choose among paths advertised by providers. The available paths will, therefore, have the same `LOC_PREF` under the common BGP policy for any non-tier-1 network receiving the anycast prefix announcement, and the AS path length will be the most significant route selection criterion.

Furthermore, except for `AS_PATH`, the rest of BGP route attributes are all based on AS-local or router-local identifiers. We can view them collectively as one combined `neighbor_ID`. These local identifiers are numerical and therefore have a total order. Under these conditions, we prove in Appendix ?? that the following theorem holds.

Theorem 1. *If in a network a BGP speaker selects its best paths by comparing (`AS_PATH`, `neighbor_ID`), then the paths from a client network to all available anycast sites form a total order. Pairwise preference comparison experiments are able to discover this total order, and this total order is predictive of a client network’s catchment site when any subset of anycast sites are enabled.*

This sufficient condition suggests that if an anycast network announces an anycast prefix from only tier-1 transit providers, then under the common BGP routing policy, a client network will exhibit a total preference order among the anycast network’s sites.

2.4.5 Practical BGP Implementation Issues

Below, we discuss two major challenges stemming from BGP implementations and how we address them.

2.4.6 *Arrival orders of BGP advertisements.*

In our experiments, we observe that when we compare the same two sites, client networks may sometimes prefer different sites. This behavior is inconsistent with the BGP specification and introduces cyclic preferences in our experiments. Upon investigating this issue, we found that real-world BGP implementations use another attribute—the arrival time of a route advertisement—as a tie-breaker after the “interior cost” attribute. Both Cisco [cis16] and Juniper [jun20] describe this tie-breaking algorithm in their online documents, albeit the attribute is not part of the BGP specification [RLH06]. Our empirical result shows that, after resolving the arrival order problem, the ratio of clients that have a consistent total order increases significantly. This result suggests that tie-breaking based on the arrival-order is a widespread implementation, and it is frequently triggered in a router’s route selection process. This is in contrast to findings from Anwar et al. [ANC⁺15a], where arrival order affected only 1.6-2.5% of measured paths.

To cope with this implementation issue, we take the arrival orders of route advertisements into account in our experiment design. In our pairwise experiments, we explicitly discover the client networks that are affected by the arrival orders of a route advertisement and incorporate the arrival orders in anycast catchment prediction. Specifically, we space the route advertisements from two different sites by an interval T such that the first advertisement arrives earlier than the second at a client network globally. We measure each client network’s catchment twice, with the route-advertisement order in the second experiment being the reverse of that in the first. If a network’s preference stays the same across the two experiments, we conclude that the network has a strict preference order between these two sites; otherwise, we conclude that it has equivalent preferences.

Later, when predicting the catchments for an anycast configuration, we consider

how the order of announcements would affect a site’s catchment and use the corresponding pairwise comparison results to predict the catchments. For instance, if we choose a configuration of three sites A, B, and C, and we announce an anycast prefix in the order of first A, then B, and last C, we will use a client network’s preference orders obtained from the measurements when A is announced before B, and B is announced before C for predicting the catchments.

Our experiments reveal that the order of BGP announcements primarily affects a network’s preference at the AS-level. It does not have any effect on a network’s preference orders when the prefix announcements are from different sites within the same AS.

2.4.7 Multi-path routing.

Some routers may split traffic to the same destination prefix among multiple paths. A network’s traffic may, as a consequence, reach different anycast sites, leading to inconsistent total orders. This practice of multi-path routing complicates the catchment prediction and could explain why the inconsistent total orders exhibited by some networks.

Most networks exhibit, however, consistent total orders after we take into account the arrival orders of route advertisements. We ignore the networks that continue to exhibit inconsistent total orders (i.e., even after taking route-announcement orders into account) from catchment prediction and optimization, but still include them when identifying catchments and measuring client RTTs under a given configuration.

2.4.8 Two-level Preference Discovery

A real-world anycast network, such as Akamai DNS [SBK⁺20a], may have a few hundred sites. It is impractical to run pairwise measurements for a network of this size. To reduce the number of preference measurements, we exploit the two-level

structure—inter-AS and intra-AS—of routing in the Internet.

When one or more sites that connect to the same AS advertise an anycast prefix, the site-level differences disappear (i.e., cannot be observed) once a neighboring AS re-advertises the prefix to its neighbors. Suppose that a client network is not directly connected to an anycast site. Then, if we discover the client network’s total preference order among the ASes that interconnect it to the anycast network, we can predict which ingress AS the network will use. Within that AS, the interior routing metrics determine which site the network will use. This routing structure allows us to separate the discovery of a client network’s preference order at the AS-level from that at the site-level. More concretely, our two-level approach first predicts the AS-level (or provider-level) preferences of a client network, and then proceeds to discover the client network’s site-level preferences, across available sites within an AS.

2.4.9 Provider-level preference discovery.

To elicit a client network’s pairwise preferences at the AS (or provider) level, we choose two transit providers and use one representative site from each provider for announcing the anycast prefix, as described in §2.3; we repeat the exercise across all pairs of transit providers. Recall that I denotes the set of transit providers of an anycast network. We need $O(|I|^2)$ pairwise experiments to discover a client network’s total order. The experiments in our testbed show that when we vary the representative site or the number of representative sites for each transit provider, 94.2% of the client networks on average do not change their pairwise preferences.

2.4.10 Site-level preference discovery.

To discover a client network’s site-level preferences among anycast sites within each transit provider, we proceed as follows. We choose two sites for each transit provider,

announce the anycast prefix and measure the client network’s preference order; as before, we repeat this experiment with other site pairs. We found that the announcement order has no impact on the client network’s site-level preferences.

Site-level preference discovery might still be prohibitively costly for a large anycast network. We could, however, use the following heuristic to eliminate this step. Once a client network’s traffic enters an AS that hosts multiple anycast sites, that AS’s interior routing protocol determines the network’s catchment site, typically based on shortest path routing metrics. We can, therefore, approximate a client network’s site-level preference order in a given AS using the shortest path distances from the client network’s ingress point to the anycast sites inside the AS. Per our experiments in some tier-1 networks, the shortest-path distance is closely correlated with a client network’s RTT to an anycast site. We use, therefore, the RTT from a client network to an anycast site to predict the site-level preference: the shorter the RTT, the more preferable the site.

Once both the provider-level and site-level preference-discovery steps are completed, a network operator can determine a network’s preference order among its transit providers as well as across the sites within each transit provider. Armed with the preference orders and the RTT measurements, they can predict the catchments of a given anycast configuration for a specific announcement order, thereby predicting the latency and load distribution. They can furthermore simulate the performance of various anycast configurations and deploy the ones that best fit their performance requirements.

2.4.11 Incorporating Peers

With the steps above, AnyOpt can generate an optimal transit-only anycast configuration. However, an anycast network may include peering connections. As peering connections can be settlement-free and deliver traffic to client networks via shorter

AS paths, incorporating them in an anycast configuration may improve performance and reduce transit cost. However, it is not straightforward how to incorporate peers in an anycast configuration, as previous work [GNK⁺21, CCGF14a] suggests that peer connections can worsen the performance of a transit-only anycast configuration.

While it is our future work to study how to incorporate peering connections in an anycast configuration in more depth, we develop a heuristic technique to conservatively include only the beneficial peers in an anycast configuration. We refer to this heuristic as the “one-pass” method. Again, it is based on measurements and offline optimization.

In the one-pass method, we first measure whether enabling a peer will reduce the average latency of the baseline configuration where only transit providers are enabled. If so, we consider the peer as a beneficial peer. From the optimal transit-only configuration found by AnyOpt, we enable one peer at a time, measure the peer’s catchment after the peer is enabled, and measure how the average latency has changed. If the average latency is reduced, we mark this peer as a beneficial peer. We then disable this peer, measure another peer, and so on until we measure all peering connections of an anycast network. If an anycast network has a total of M peers, this step requires M BGP measurements.

After we identify the beneficial peers, we use a greedy offline algorithm to choose the set of peers to add to the transit-only configuration. We rank the beneficial peers by the size of their catchments measured during the one-pass experiments. We start with including the beneficial peer with the largest catchment, and then examine the beneficial peer with the second largest catchment, and so on. For each peer we consider, we estimate whether including this peer will reduce the average latency. If it will reduce the average latency, we include it in the configuration. Otherwise, we skip it. We note that the one-pass experiments we conduct only include one peer

at a time. Thus we cannot predict the catchment of a peer and hence, the average latency, accurately when multiple peers are enabled simultaneously. To overcome this challenge, we conservatively assume that when we add a peer with a smaller catchment size, all client networks in the peer’s catchment discovered in the one-pass experiments will switch to that peer. Only if the average latency is reduced in this case, we will include the peer. Otherwise, we will skip the peer.

Experimentally, we find that the one-pass method can further reduce the average latency of a transit-only configuration, but not by much. It is around 5 ms for our testbed. It is our future study to investigate why the reduction is this small. One plausible explanation is that the beneficial peers in total only attract a small fraction of the overall traffic in our testbed, as we show in § 2.5.7. This result may not hold for other anycast networks where peering connections attract larger amounts of traffic, but Schlinker et al. also observed that peer routes and provider routes had similar performance in terms of latency for the Facebook network [SCC⁺19].

2.4.12 Putting it Together

Finally, we summarize the steps it takes to predict an anycast network’s catchments and to optimize its performance.

- (1) For each site, we announce a test anycast prefix to a transit provider the site connects to. We use this experiment to measure the RTT from a client network to this site (refer §2.3).
- (2) We run pairwise preference measurements among all transit providers of the anycast network. We consider the impact of the arrival orders of BGP advertisements by announcing each pair twice with a reversed order in the second measurement to get enough information for simulating all possible announcement orders. We choose one representative site from each transit provider to

perform these experiments.

For each transit provider, we use pairwise experiments to discover a client network’s preferences among the sites within this transit provider. For a large anycast network with many sites where this approach is infeasible, we approximate a client network’s preferences by its RTTs to the sites within the transit provider.

- (3) Using the above experiments, we compute offline the total preference order of each client network for the announcement order that maximizes the number of client networks with a consistent total order. We exclude a client network in this computation if its pairwise preferences do not exhibit a total order. We use the total order to predict the catchment site of a client network given a site-level anycast configuration. We can enumerate through as many configurations as required offline and choose the best ones to deploy. After the deployment, we can include the beneficial peering links discovered using the one-pass method described in §2.4.11.

2.4.13 Analysis.

We now estimate the number of BGP measurements needed as well as the time it takes to finish them for optimizing the Akamai DNS anycast network [SBK⁺20a] with a transit-only configuration. Akamai DNS has a few hundred sites. We use 500 sites and 20 transit providers to approximate the Akamai DNS network. For a network of this size, site-level pair-wise experiments are infeasible. We instead use a client network’s RTTs to the anycast sites to approximate their intra-AS site-level preferences. In total, we require 500 singleton experiments for measuring a client network’s RTT to each site and 380 pair-wise measurements for discovering the network’s pair-wise preferences between any two transit providers. We can, however,

run the BGP measurements in parallel with different anycast prefixes. Suppose that we use four test anycast prefixes (the number of prefixes we use in our testbed), and we separate each BGP experiment by two hours. Then the 500 singleton measurements will take $500 * 2/4 = 250$ hours or about 10 days to finish. The 380 pair-wise experiments will take $380 * 2/4 = 190$ hours or around eight days to finish. So for an anycast network of size as large as the Akamai DNS system, a network operator can perform these measurements once a month and use the results to adjust their network configurations. If the topological features of the Internet such as a client network's average RTT to a site remains stable over the course of a month, then AnyOptis suitable for such large networks.

2.5 Performance Evaluation

In this section, we use real-world experiments on the anycast testbed described in §2.3.1 to evaluate AnyOpt. In particular, we answer the following questions:

- (1) How does the order in which we announce an anycast prefix from different sites affects the catchment of each site?
- (2) How effective are the pairwise preference elicitation experiments in discovering the total ordering of AS-level and site-level preferences of client networks in a provider-only anycast configuration?
- (3) How accurately can we predict the catchments in a provider-only anycast configuration?
- (4) Can AnyOpt's catchment prediction help in optimizing anycast deployment for performance (e.g., in terms of latency reduction)?

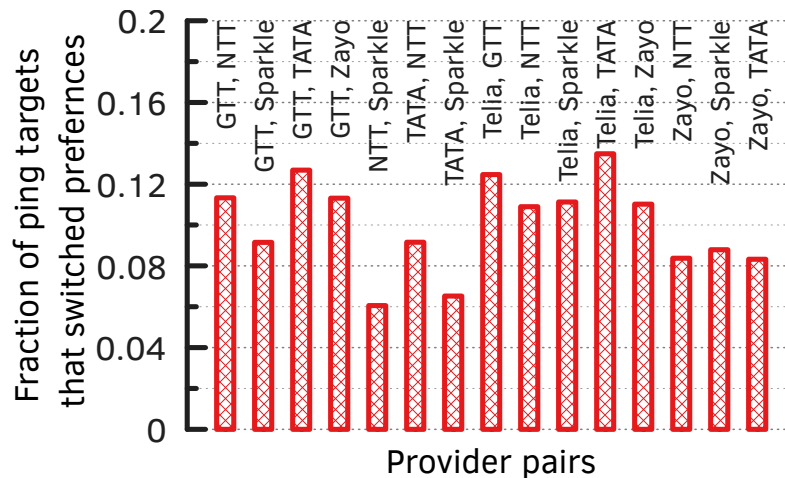


FIGURE 2.4: A significant fraction of ping targets switch their preferences based on the order in which they receive the anycast prefix announcements.

2.5.1 Pairwise Preference Discovery

In this section, we answer the first two questions regarding the impact of BGP announcement order and our ability to observe a total preference order. We begin with experiments to assess inter-AS preferences and then repeat the same for intra-AS preferences.

2.5.2 Inter-AS experiments & impact of announcement order.

AnyOptuses pairwise experiments to discover a client network’s preference order between two anycast sites. If a network has a total ordering among all sites, AnyOptuses it to predict its catchment for a given anycast configuration. We take the two-level approach described in §2.4.8 to discover a network’s preference orders for anycast sites on our testbed. For the AS-level preference discovery, we pick two transit providers and run two pairwise comparison experiments. In each experiment, we announce an anycast prefix to a representative site from each provider AS respectively. After BGP stabilizes, we measure each site’s catchment and the RTT from

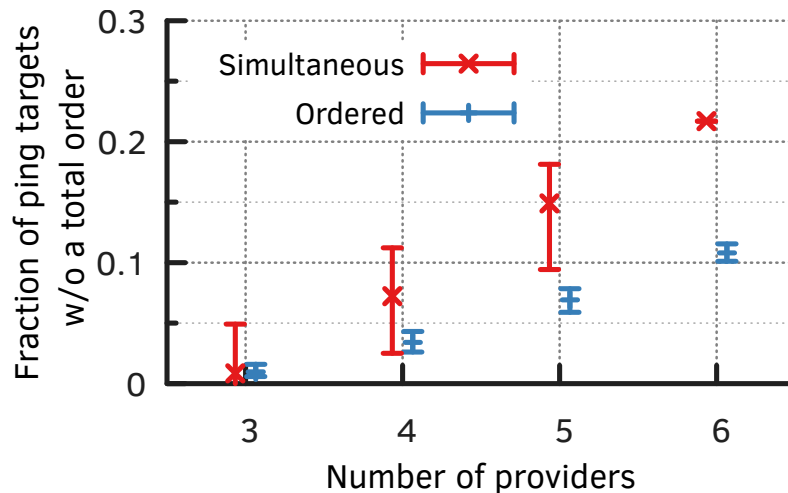


FIGURE 2.5: Announcing an anycast prefix simultaneously (in red) from two transit provider ASes lead to more clients without a total preference ordering compared to separating the announcements from the two ASes by six minutes (in blue).

a target network to each catchment site as described in §2.3. We separate the two announcements by six minutes in each experiment, and in the second experiment, we reverse the order of the announcements in the first experiment. As we describe in §2.4.5, BGP implementations break ties using the arrival order of route advertisements. Therefore, a network’s preference may differ across the two experiments.

Naturally, we first investigate how catchments change when we switch the prefix announcement order between two transit providers. Figure 2.4 shows that around 6% to 14% of ping targets change their catchment sites, suggesting that the arrival order of the BGP announcements breaks the tie between two equally preferred paths. Note that the change of a client network’s preference is not due to transient path changes, as 1) we wait long enough for BGP to converge to measure the catchments, and 2) we separate the first and second experiments by two hours and withdraw the prefix announced in the first experiment before we announce it in the second experiment.

Next, we check whether a client network’s pairwise preferences can form a total

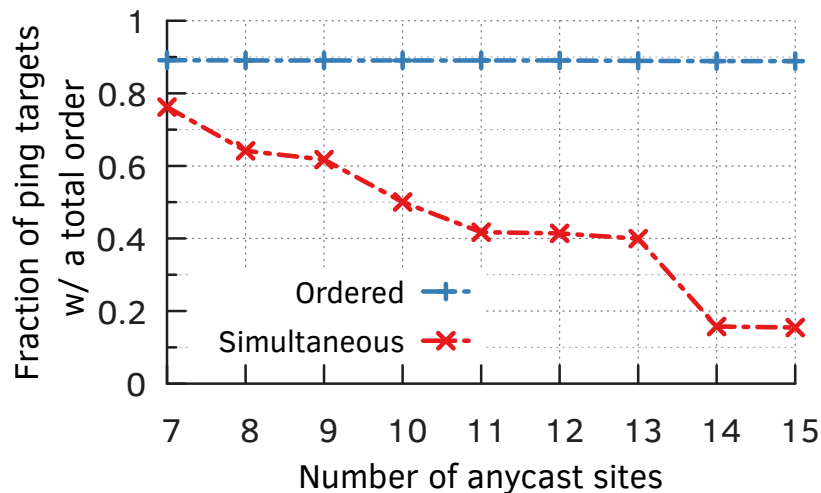


FIGURE 2.6: Fraction of ping targets with a total ordering remains steady at 85% as the number of sites increases and the announcement order is controlled, but falls drastically otherwise.

order among the set of transit providers. As a comparison, we also run pairwise experiments without considering the order of BGP announcements. That is, we announce an anycast prefix simultaneously from a representative site in each of the two providers. Our testbed has a total of six transit providers. We use it to emulate an anycast network with three to six providers respectively. For each emulation, we choose a random $X \in [3, 6]$ number of transit providers and run pairwise preference discovery experiments among those providers.

Figure 2.5 shows that as the number of providers increases the fraction of networks that do not have a total order increases. The error bars in Figure 2.5 show the variance among different measurements. Incorporating the order of BGP announcements reduces the fraction of networks without a total ordering by half. When there are six transit providers, if we do not consider the announcement order, 21.7% of networks do not have a total preference order among the providers. In contrast, if we consider the announcement order, this number decreases to 10.8%.

2.5.3 *Intra-AS experiments.*

After AS-level preference discovery, we determine each network’s preference orders among the anycast sites within the same transit provider AS. To do so, we choose a transit provider and announce an anycast prefix from any two sites within the transit provider. The site-level catchment is determined by an AS’s interior routing mechanism. Therefore, the BGP announcement order should not affect site-level catchment. For a transit provider with N sites, we run $N * (N - 1)/2$ pairwise site-level experiments for discovering a network’s preference order. For our testbed, each provider has two to four sites. So it takes one to six pairwise experiments to discover a network’s preferences per provider.

After the two-level preference measurements, we calculate a network’s total preference order among all sites by first ranking the transit providers based on the network’s preference for a specific announcement order and then ranking the sites within each provider. We then calculate the fraction of networks that have a total order. Similarly, as a comparison, we also run pairwise site-level preference discovery experiments without considering the BGP announcement order. To do so, we pick two random sites and announce an anycast prefix from these two sites simultaneously. We measure a network’s pairwise preferences and compute its total preference order. We vary the number of sites in the experiments to emulate an anycast network of varying sizes.

We start with an anycast network with one site in each transit provider connected to our testbed. When we add more sites, the fraction of networks that have a total preference order sharply decreases (as shown in Figure 2.6) if we do not consider the impact of BGP announcement orders on route selection. When the number of sites reaches 15, only 15.5% of networks have a total preference order. In contrast, when we consider BGP announcement orders and use the two-level pairwise preference

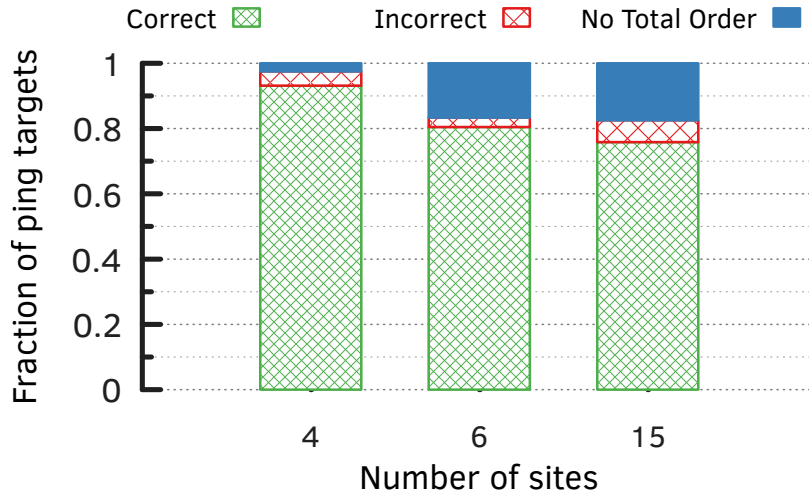


FIGURE 2.7: Catchment prediction accuracy

discovery mechanism, 88.9% of networks still exhibit a total preference order, which enables AnyOpt to accurately predict the catchments of an anycast configuration.

2.5.4 Catchment Prediction

Next, we evaluate whether AnyOpt can accurately predict catchments and the overall latency of an anycast configuration, which answers question (3). We first choose a random subset R from all sites in our anycast testbed. We then use each client network’s total preference order (among this subset) under a BGP announcement order for predicting the client network’s most preferred site among R and its RTT to its catchment site. We do not predict catchments for client networks that do not exhibit a total order. We then deploy the configuration R under the same BGP announcement order and measure the resulting catchment of each site in R and each target’s RTT to its catchment site. We compare the predicted catchments and RTTs with the measured ones to gauge the accuracy of AnyOpt’s predictions. We then vary the subset R and repeat the above steps.

Figure 2.9 summarizes the results. In Figure 2.7, we show the results from three

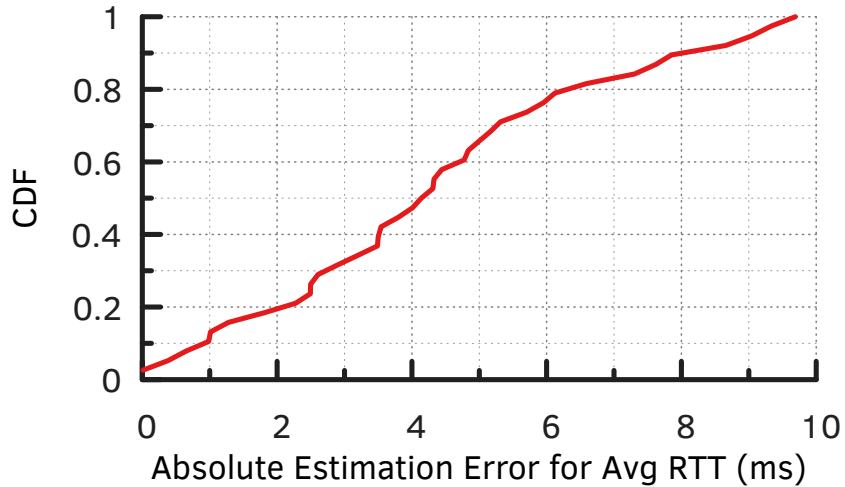


FIGURE 2.8: Absolute RTT estimation error

experiments. In the first two experiments, we choose four and six transit providers in our anycast testbed and enable a representative site in each provider. In the third experiment, we enable all 15 sites. We measure the fraction of client networks for which we correctly/incorrectly predict their catchment sites and the fraction of networks that do not have a total order. The figure shows that when the number of sites increases, the number of networks that have a total preference order decreases. However, within those networks that have a total order, we can correctly predict the catchment sites more than 93% of the time. There are several reasons that might lead to no total orders, such as multipath routing, uncommon BGP policies, and routing configurations that violate the sufficient conditions for a total order (§2.4.1).

We plot the CDF of the absolute values of the differences between the predicted average RTT (of all targets) of an anycast configuration and the measured average RTT of the same anycast configuration in Figure 2.8. We compute the CDF from 38 random anycast configurations with the number of sites ranging from 1 to 14. Per this figure, the predicted average RTT is within 6 ms of the measured RTT for more than 80% of anycast configurations.

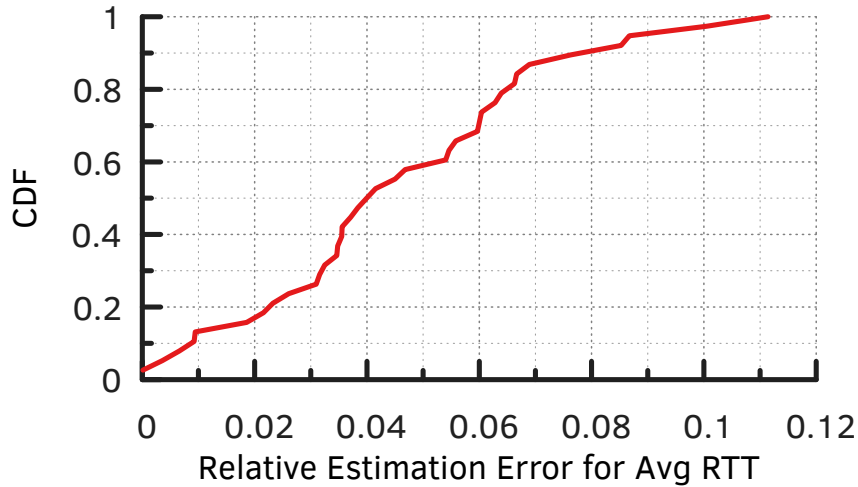


FIGURE 2.9: Relative RTT estimation error

Figure 2.9 shows the relative errors of the predicted average RTT when compared with the measured average RTT for each anycast configuration we choose. For all configurations we tested, the mean predicted average RTT error is less than 4.6%.

2.5.5 Takeaways.

These results are encouraging. They suggest that once we obtain a client network’s total preference order and the RTTs from each site to each target, we can accurately predict the catchments and the overall RTT of an anycast configuration for our testbed.

Inferring the linear ordering of network preferences of clients is crucial for predicting the catchment. Regardless of which subset of networks we choose for announcing the anycast prefix, we demonstrate that we can predict the catchment of clients with high accuracy. We exclude clients with inconsistent linear ordering of network preferences, unless otherwise mentioned, for this evaluation.

We varied the number of sites as well as the tier-1 transit providers for announcing the prefix, and checked for each scenario how accurately we can predict the catch-

ment for all clients. In Fig. 2.7 we show the fraction of correct predictions in each of the different scenarios; we also show, for completeness, the fraction of clients with inconsistent linear ordering of network preferences. For each testing scenario, we also indicate how we inferred the linear ordering of clients’ network preferences: “N” and “A” indicate that we used PPE tests without and with activation sequences, respectively. “N” and “A” indicate that we used PPE tests *without* and *with* activation sequences, respectively.

When we rely on the “simple” PPE test (i.e., with no activation sequences), the fraction of clients with inconsistent linear ordering of network preferences increases drastically with increasing number of sites (Fig. 2.7). When doubling the number of sites from 8 to 16, for instance, the fraction of clients with inconsistent linear ordering of network preferences nearly triples from 30% to approximately 90%. When using the PPE test with activation sequences, the fraction of such clients is relatively small—nearly 20%. Among the clients with a consistent linear ordering of network preferences, however, our prediction accuracy is quite high (Fig. 2.7). In terms of the available clients on which we could test the accuracy, in the worst case our accuracy drops to 89% for the scenario “4;N” scenario. When using the PPE test with activation sequences, the inferred linear order of network preferences helps us achieve at least 95% accuracy. Shall inter AS level preference order also work for different anycast sites in that AS? We also conduct a pair evaluate experiment to verify this, we picked different anycast sites but belong to the same two ASes to announce our prefix in the same activation sequences, the inter AS preference of all clients agree with each other for 94.5% on average.

2.5.6 Performance Optimization

In this section, we answer question (4). In addition to predicting the catchments of an anycast configuration, AnyOptassists in finding a configuration (i.e., the set

of sites enabled to announce an anycast prefix) that results in the lowest average RTTs between the anycast sites and the targets. To estimate the extent of potential RTT reductions, we conduct the following experiments. We use offline computations to iterate over as many anycast configurations for our testbed as we could possibly compute within a time bound, which we currently set to six hours. For each configuration, we choose a prefix announcement order that yields the largest fraction of client networks with a total preference order. The computation returns a 12-site configuration out of 15-site testbed. We deploy this AnyOpt-optimized configuration and measure the catchments of each site and the average client latency. We then compare the average RTT of the AnyOpt configuration with two other types of configurations and the default configuration of enabling all 15 sites.

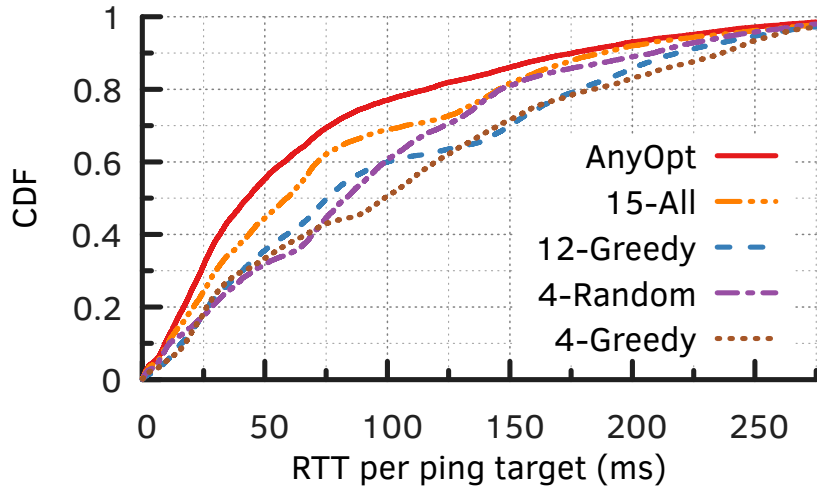


FIGURE 2.10: AnyOpt-optimized configuration substantially outperforms other approaches in terms of RTT.

N-Greedy. In these configurations, we enable N sites using a greedy algorithm. We enable the sites in a configuration according to their average unicast RTTs to all client networks. Recall that we measured those RTTs by announcing an anycast

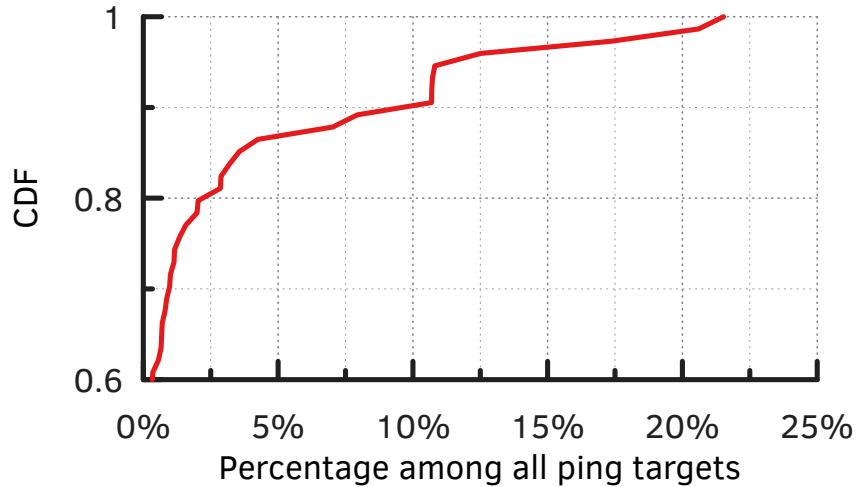


FIGURE 2.11: CDF of a peer AS’s catchment when adding a peering link to AnyOpt’s transit-only configuration.

prefix from only one site. We choose the top N sites with the lowest average RTTs to the measurement targets, deploy the configuration, and measure its catchments and RTTs. The 12-Greedy configuration has the same number of sites as in the AnyOpt-optimized configuration.

4-Random. To simplify management, network operators may choose to use only a small number of providers and sites. For this scenario, we assume a network chooses two providers and two sites in each provider. We randomly generate three such configurations, deploy them, and measure their catchments and RTTs.

We show CDF of the resulting RTTs to each target network under each scenario in Figure 2.10. The 4-Random line is the result from the best random four-site configurations we generate. The median RTT for the AnyOpt-optimized configuration (the “AnyOpt” line) is 43 ms, while that of the greedy configuration of the same number of sites (the “12-Greedy” line) is 76 ms. Put another way, AnyOpt improves the median RTT by 43.4% for the same number of sites. Compared with other configurations, AnyOpt improves the median RTT by 27-59.8%. Although not shown in

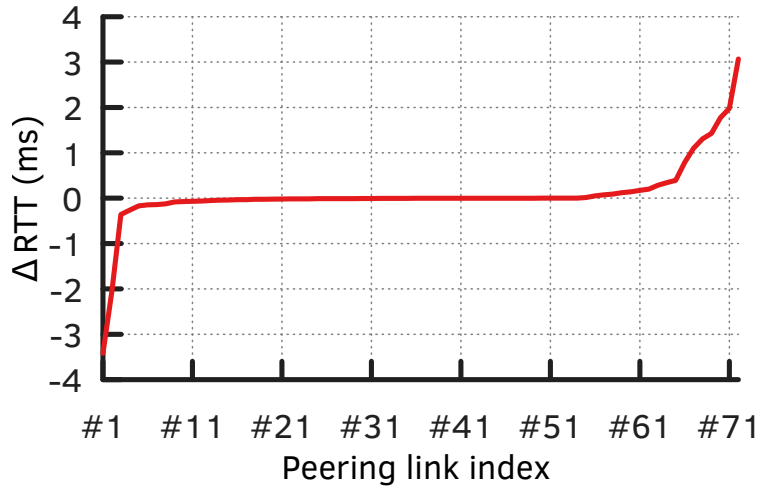


FIGURE 2.12: Mean RTT changes when adding a peering link to AnyOpt’s transit-only configuration.

the figure, the AnyOptconfiguration also has a 33 ms lower average RTT compared to the greedy configuration with the same number of sites. It has 14 – 35 ms lower average RTT than the other configurations.

Consistent with observations from prior work, we find that the configuration with all 15 sites enabled (the “15-all” line in Figure 2.10) exhibits worse performance than a smaller AnyOpt configuration with 12 sites. However, the 12-site AnyOptconfiguration substantially outperforms the other configurations with fewer or the same number of sites. *This result shows that **more sites** can lead to **substantially better performance** when using the right measurements and optimization approach.*

2.5.7 Incorporating Peering Links

Next, we show how incorporating peering links to the transit-only AnyOptconfiguration can impact the average client latency. The AnyOpttestbed includes 104 non-transit peering links. Among them, only 72 peering links can reach some of our ping targets, which could be due to routing configurations, e.g., a peer may filter our testbed traffic, or a peer’s catchment is too small to include any ping target.

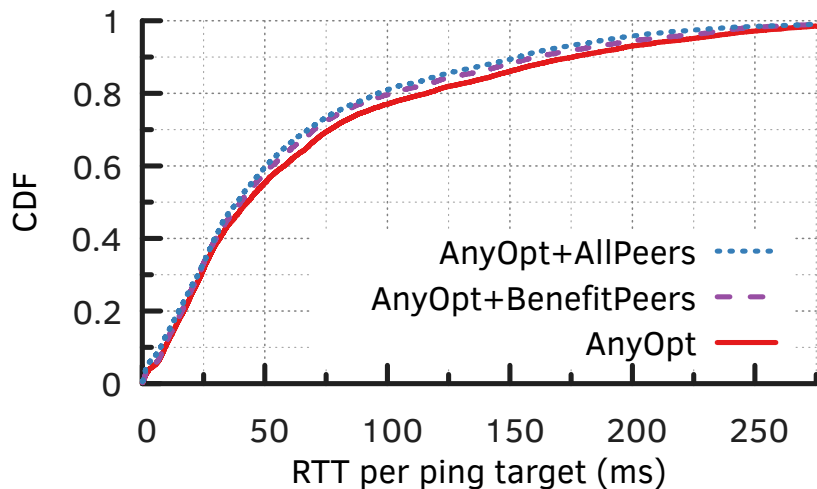


FIGURE 2.13: CDF of client RTTs after incorporating peering links.

To estimate a peer’s impact on the average client latency, we enable each peer separately on the AnyOpt-optimized transit-only configuration (as described in §2.4.11). We then measure the peer’s catchment size under this configuration and how the average client RTT has changed. If enabling the peer reduces the average RTT, we deem it a beneficial peer. Figure 2.11 shows the CDF of a peer’s catchment size distribution, and Figure 2.12 shows how enabling a peer changes the RTT averaged over all ping targets. We rank the peers by the value of average RTT changes they introduce. As can be seen, more than 80% of the peer links on our testbed have a catchment size consisting of fewer than 2.5% ping targets. Only a few peers have noticeable impact on the average RTTs.

We then use the one-pass heuristic to enable the beneficial peers that are likely to reduce the average RTT of the AnyOpt-optimized configuration. We refer to this configuration as AnyOpt+BenefitPeers. We measure the RTT distribution of each ping target under this configuration, and compare it with the configuration of enabling all peers (AnyOpt+AllPeers) and AnyOpt. Per Figure 2.13, AnyOpt+AllPeers and AnyOpt+BenefitPeers have similar performance. Both perform slightly better than

AnyOpt, but not significantly. Specifically, AnyOpt+BenefitPeers reduces AnyOpt’s average RTT from 68 ms to 63 ms, while AnyOpt+AllPeers reduces it to 61 ms. We note that in our testbed, enabling all peers leads to a configuration with a slightly lower mean RTT than the configuration identified by the one-pass heuristic. This result may not be generally applicable to other anycast networks, and a conservative approach such as the one-pass heuristic, which includes beneficial peers one-by-one, will be useful in situations where enabling all peers worsens the performance of a transit-only configuration.

2.6 Limitations and Future Work

This work has a few key limitations and leaves open a number of interesting directions for future work. We discuss them below.

Testbed We obtain all experimental results on the anycast testbed we use. Although from our theoretical analysis, we expect that other networks would obtain similar results, this hypothesis is yet to be validated by real-world experiments on other anycast networks.

Large anycast networks. Although we outlined a heuristic approach in §2.4.8 that uses a client network’s RTTs to anycast sites of a large anycast network to discover the network’s intra-AS site preferences, we have not yet to test the effectiveness of this approach on other anycast networks.

Settlement-free peers. We have performed experiments to fine-tune a baseline anycast configuration consisting of only advertisements to tier-1 transit providers. We also used a one-pass method to evaluate the peering links we had in our infrastructure. Our evaluation showed that the RTT reduction brought by these peerings is

small. While interesting, our findings might be impacted by limited connectivity in our testbed (15 sites and 104 peering links). An open question is how much performance might change if we advertised to settlement-free peers for other larger anycast networks.

Stability analysis. Deploying AnyOptas a production system would require a longitudinal study to determine how often client total orders change, and by how much. This information would then govern the frequency of the pairwise experiments that would be necessary to keep the total orders up to date. We have only conducted a few experiments in January 2021 to gauge whether the performance of an anycast configuration remains stable. That is, if we deploy the optimal configuration given by AnyOpt, will it remain optimal? We deployed a configuration and measured it weekly in the first three weeks of January 2021. The results are promising, more than 90% of the catchments remain unchanged and the average RTT is also very stable in the three-week duration. It is our future work to study the stability of an optimal configuration in detail.

Other control knobs. A network operator can modify the BGP attributes of an anycast prefix advertisement (e.g., prepend its own AS numbers) to influence catchments. She can also use the BGP poisoning technique [KBSC⁺12] to avoid a specific AS hop along the path. It is in our future work to explore how to use these “knobs” for catchment prediction and performance optimization.

Reducing the number of experiments. When the number of transit networks that an anycast network connects to grows larger (e.g., 20 or more), performing a quadratic number of experiments becomes burdensome. It is natural to ask whether the total orders could be learned, or learned approximately, using fewer experiments. One

possible future direction to reducing the number of experiments would be to rely on publicly available BGP routing tables to infer as much about catchments as possible, and then to supplement the information gleaned from these tables with active measurements.

2.7 Related Work

Measuring IP anycast performance. IP anycast has long been used by Internet services to provide automatic load balancing and latency reduction among service replicas. Previous work focuses on measuring the performance of deployed IP anycast systems, including DNS root servers [LLC⁺13a, LHF⁺07a, MSH⁺16a, LJD⁺13a, GCF⁺16, KLA⁺21, BFR06b, KLA⁺21, dOSHK17b] and CDNs [CFKB⁺15a, dVARD20, KLA⁺21]. Most of the studies on root DNS anycast systems show that global IP anycast often fails to route clients to the replicas that provide the lowest latency or to evenly distribute the workload among the replicas. As an example, Li et al. [LLSB18a] showed that for the D-root name server, only one third of its clients' queries were routed to their geographically closest anycast sites. Differently, Calder et al. [CFKB⁺15a] and Koch et al. [KLA⁺21] show that for Microsoft CDN, only 35% of users experience anycast latency inflation.

Explaining and improving poor performance. Sarat et al. [SPT05] proposed to limit the radius of an anycast prefix announcement to prevent a client from reaching a topologically distant site. However, as a global ISP often has a network spanning a large geographic area, limiting the radius of a BGP announcement cannot prevent a client from reaching a distant site. Ballani et al. [BFR06b] hypothesized that the sub-optimal performance of IP anycast is due to BGP's routing behavior. BGP is performance oblivious, and ASes configure their BGP routers to find the "cheapest" rather than the best performing routes. They proposed to host all anycast sites

through one tier-1 provider. Li et al. [LLSB18a] proposed to embed the origin router’s geographic location in a BGP announcement to make BGP latency aware. Alzoubi et al. [ALR⁺11] proposed to use a central route controller and MPLS tunnels to direct anycast traffic to specific anycast sites within one ISP, but it is difficult to generalize this approach to large anycast networks that span multiple ISPs, as across-domain MPLS engineering is not well supported by ISPs. Fastroute [FMM⁺15a] describes an anycast architecture that combines DNS redirection and anycast routing to manage the workload of a large CDN.

Different from this body of work, AnyOptaims to predict anycast catchment and enable a service provider to choose an optimal anycast configuration. It can reduce the overall client latency without modifying BGP announcements to embed geographic information. Although we do not explicitly address load-balancing in this work, as we explain in §2.3 and Appendix ??, a network operator can add a load constraint to the optimization problem or predict how load will change by accurately predicting anycast catchment.

Measuring and inferring anycast catchment. Cloudflare’s Verfploeter [dVdOSH⁺17a, dVARD20] measures the catchment of an anycast site without using a large number of active probes. Verfploeter sends out ICMP requests to hosts with the source addresses of the requests set to an anycast address. A host’s ICMP reply will reach its corresponding catchment site. Vries [dVdOSH⁺17a] et al. have shown that Verfploeter can accurately map out the catchment of an anycast site, overcoming the limitation of previous work that uses RIPE Atlas [Sta15a] to measure anycast catchment. Another body of work [dOSHK17b, LLSB18a, WH17a] uses RIPE Atlas to send active probes to an anycast address, but RIPE Atlas has a skewed geographic distribution. AnyOptborrows Verfploeter’s architecture to map an anycast site’s catchment, but enhances the architecture to measure a client’s RTT to an anycast

site (§2.3).

Sermpezis and Kotronis [SK19] proposed to use the inferred AS-level Internet topology for predicting anycast catchment. Their approach cannot, however, accurately predict how ASes break ties among equally preferred routes. In addition, any incorrect inference in the AS topology will exacerbate the inaccuracy of its prediction. As shown in their simulations, when the number of anycast sites increases from two to four sites, the number of nodes with certain inference decreases from 15000 to 6000 and will keep decreasing as the number of anycast sites increases.

In contrast, AnyOpt takes a measure-model-and-optimize approach. It uses carefully designed BGP experiments to discover how ASes choose paths and combine the experimental results with offline computation for anycast performance prediction and optimization. In the future, we plan to investigate whether we can combine AnyOpt with an inference-based method to further reduce the number of BGP experiments required for making accurate predictions.

2.8 Conclusion

In this paper, we introduced AnyOpt and showed how it can be used to minimize the latency and balance the load of an anycast network. The key idea is that, in certain circumstances, the site preferences of each client network exhibit a total order and we can discover the total orders of all client networks using pairwise preference-elicitation experiments that announce an anycast prefix from any two available sites. We prove the sufficient conditions under which a client network exhibits a total order and use a two-level approach consisting of inter-AS experiments followed by intra-AS experiments to reduce the total number of experiments. With the total order in hand for each client network, we can predict the catchment of each anycast site for any particular subset of sites that might advertise. Then by formalizing the problem as an instance of the SPLPO problem, we can find a set of anycast sites that minimize

latency while balancing load (i.e., satisfying capacity constraints). Our evaluation using a testbed that has 15 global sites demonstrates the feasibility of our system. AnyOpt can predict catchment areas with small errors using only a quadratic number of experiments, and solving the resulting optimization problem yields tangible reductions in latency. To the best of our knowledge, AnyOpt is the first work that systematically tackles the anycast performance prediction and optimization problem.

2.9 ACKNOWLEDGMENTS

We thank the anonymous reviewers and our shepherd Jennifer Rexford for their helpful comments, and Haoyu Wang and Shen Zhu for helping with an early draft of this paper. We sincerely thank the network engineering team at Akamai Technologies, especially Aaron Block and Aaron Atac, whose help made this work possible. We thank Kamesh Munagala for help in proving that even approximating the minimum cost of SPLPO is NP-hard. This work was supported in part by the National Science Foundation under awards 1910867, 1763617, 1763742, 1822965, and 1827674, and in part by subcontracts from Akamai Technologies in support of DARPA prime contract HR0011-17-C-0030. Additional support was provided by Microsoft Research Faculty Fellowship 8300751 and AWS Machine Learning Research awards.

2.10 Appendix

In this simplified network model, one can prove the following lemma:

Lemma 1. *Any node n in this network has a total preference order over the set of anycast nodes A .*

This lemma holds because we can rank n 's shortest path distances to the nodes in A numerically, and the preference order is the same as the shortest path ranking order.

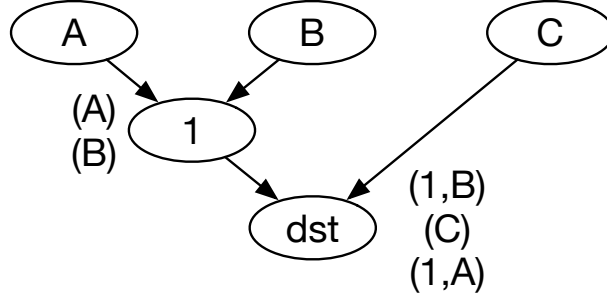


FIGURE 2.14: The Case that do not Preserve Linear Preference

Furthermore, we show that under this model, if we know the total preference order of a node, we can predict a node’s catchment:

Lemma 2. *If we enable any subset of nodes (S) in A to announce an anycast prefix, then a node n will be “caught” by its most preferred node in the subset (S).*

This lemma holds because all nodes use shortest path routing. Node n , by definition, has the shortest path reaching its most preferred node in S . Therefore, n ’s traffic will be “caught” by its most preferred node in S .

Next, we show that given a set of clients C , we are able to choose a subset of nodes in A to optimize for latency and load balancing.

2.10.1 No Consistent Linear Order

As shown in figure 2.14, the anycast sites are A, B, and C. D and E are the downstream networks of the anycast sites, the list beside each network is the preference order as described in Gao&Rexford assumption.

The full combinations and path selected are shown in the table 2.2

The pair-wise comparison in this case result in a loop.

2.10.2 Can Preserve a Linear Order, but Not Predictive from the Linear Order

Here, we present a counter example that comply the Gao&Rexford assumption, but the preference order can not be inferred by the pair-wise comparisons.

Table 2.2: Path and Anycast Site Finally Chosen in the Counter Case I

Configuration	Path Selected	Site Selected
A	(1,A)	A
B	(1,B)	B
C	(C)	C
A, B	(1,A)	A
A, C	(C)	C
B, C	(1,B)	B
A, B, C	(C)	C

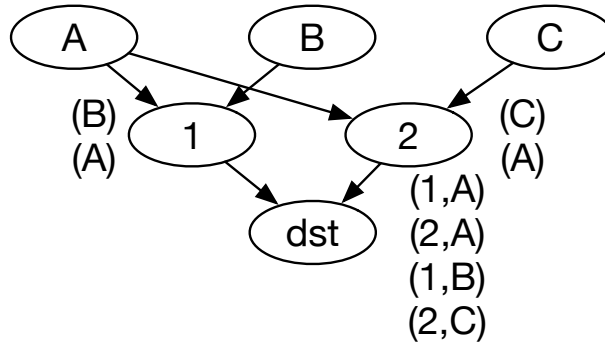


FIGURE 2.15: The Counter Example I

As shown in figure 2.16, the anycast sites are A, B, and C. D and E are the downstream networks of the anycast sites, the list beside each network is the preference order as described in Gao&Rexford assumption.

The full combinations and path selected are shown in the table 2.5

From the pair-wise comparison result, we can get $A > B > C$. However, when

Table 2.3: Path and Anycast Site Finally Chosen in the Counter Case II

Configuration	Path Selected	Site Selected
A	(1,A)	A
B	(1,B)	B
C	(2,C)	C
A, B	(2,A)	A
A, C	(1,A)	A
B, C	(1,B)	B
A, B, C	(1,B)	B

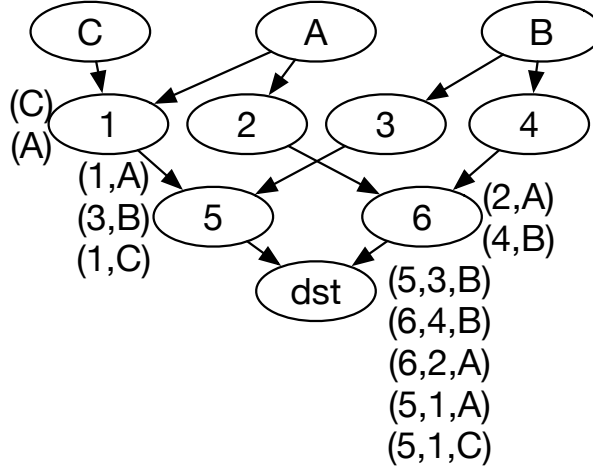


FIGURE 2.16: The Counter Example II

Table 2.4: Path and Anycast Site Finally Chosen in the Counter Case III

Configuration	Path Selected	Site Selected
A	(6,2,A)	A
B	(5,3,B)	B
C	(5,1,C)	C
A, B	(6,2,A)	A
A, C	(6,2,A)	A
B, C	(5,3,B)	B
A, B, C	(5,3,B)	B

we turn on all three anycast sites, site B is chosen based on the preference order of each network.

And we also observe that, the winning path in the pair-wise comparison between A and B is different from the unicast path of site A alone.

The full combinations and path selected are shown in the table:

The full combinations and path selected are shown in the table:

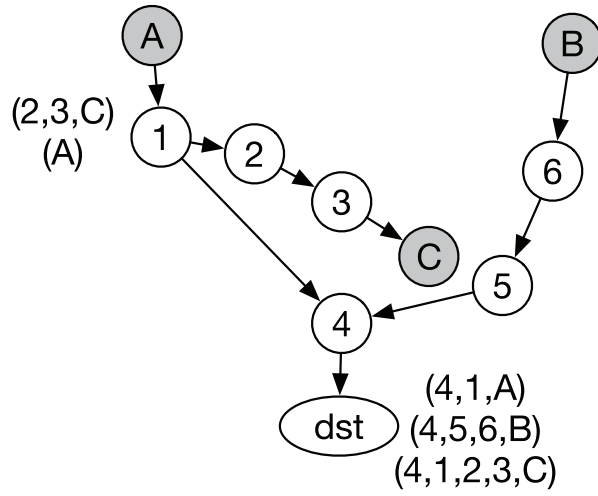


FIGURE 2.17: The Counter Example III

Table 2.5: Path and Anycast Site Finally Chosen in the Counter Case IV

Configuration	Path Selected	Site Selected
A	(5,1,A)	A
B	(4,5,6,B)	B
C	(4,1,2,3,C)	C
A, B	(4,1,A)	A
A, C	(4,1,2,3,C)	A
B, C	(4,5,6,B)	B
A, B, C	(4,5,6,B)	B

Passive Performance Optimization

3.1 Introduction

IP anycast [PMM93] refers to the routing practice where a network announces the same IP prefix from multiple geographically-distributed locations. It is widely used by distributed systems such as root Domain Name Service (DNS) servers and Content Distribution Networks (CDNs) to reduce client latency and balance load. Unlike conventional DNS-based redirection services [CST15a], IP anycast can direct client traffic to nearby CDN sites without a separate load-balancing system [CFKB⁺15b]. Partly due to its simplicity, several large CDNs, including Cloudflare [Clo22], Google Cloud CDN [Clo], and Microsoft Azure [Mic], have all adopted IP anycast.

The simplicity of IP anycast comes with a downside: it gives a CDN operator little control over which sites its clients' traffic reaches. The Border Gateway Protocol (BGP) [RL94a] routes a client's traffic to a CDN site based on its policies and network topology dynamics. Since BGP is a policy-routing protocol, its best-path selection algorithm does not consider any performance metrics directly tied to path latency. As a result, previous studies show that BGP often routes a client to an anycast

site that is geographically distant from the closest anycast site [BF05a, BFR06a, SPT06, CRUR, LHF⁺07b, Kui, LLSB18b, DVvRDDBP19], resulting in prolonged client latency. We refer to this pathology as catchment inefficiency. Interactive applications such as gaming and web browsing demand low latency [Aka, MCZ⁺20]. In a competitive market, providing low-latency access to clients worldwide is crucial for CDN providers to gain competitive advantages.

Recently, several private and public CDNs adopted regional IP anycast as a promising approach to address some of the limitations of IP anycast [Mah15, CSG⁺18, HZWS18]. A CDN that employs regional IP anycast partitions its sites into different regions and announces a distinct IP anycast prefix from the sites in each region. When a client makes a DNS query to one of the CDN’s customers, the CDN’s DNS returns a regional IP anycast address based on the client’s location. For clarity, we hereafter refer to the anycast configuration where a network announces the same IP prefix from multiple sites without regional partitions as global IP anycast.

Regional IP anycast retains the simplicity of IP anycast. Yet it gives a CDN operator some degree of control over which sites a client’s traffic reaches. However, regional IP anycast has not been thoroughly studied. Questions such as *how a regional IP anycast CDN is deployed* and *whether regional IP anycast can effectively address the catchment inefficiency problem of global IP anycast* are unanswered. Understanding these questions can deepen our understanding of how to deploy a performant large-scale IP anycast system.

This work aims to answer the above questions. We first conduct an in-depth study on the deployment strategies and the performance of two global-scale regional anycast CDNs: Edgio (formerly Edgecast and acquired by Limelight in 2022) and Imperva (formerly Incapsula and now part of Imperva). According to a prior study [HZWS18] and our recent survey (§ 3.4.1), these two CDNs are among the top-15 largest CDNs that currently deploy regional IP anycast. Furthermore, we discover that Imperva’s

own authoritative DNS server system uses global IP anycast and its sites and network configurations overlap significantly with its CDN. Therefore, we choose Imperva’s authoritative DNS server system as its global anycast counterpart and compare their performance differences. We use RIPE Atlas [Sta15b], a globally distributed set of probes, to send DNS queries to the domains hosted by Edgio and Imperva. We then send traceroute queries from RIPE Atlas probes to the IP addresses the probes receive and infer the geographic locations of the CDN sites the probes’ traffic reaches, which we refer to as the catchment sites. From these steps, we are able to infer the regional site partition and the DNS mapping strategies of the two CDNs (§ 3.4.4). Similarly, we can map the sites of Imperva’s authoritative server system.

We find that Edgio and Imperva use different regional partition strategies (§ 3.4.3). Edgio divides its customers’ clients into three or four regions, while Imperva divides its customers’ clients into six regions. The region boundaries in both CDNs largely follow country or continent borders. Most clients receive IP prefixes originating from the CDN sites in the same geographical areas from DNS, but sometimes DNS returns a remote regional IP address to a client. We reached out to both Edgio and Imperva to discuss our findings and one responded and confirmed parts of our findings.

The performance study of regional IP anycast reveals both the advantages and limitations of regional IP anycast (§ 3.5). We find that regional IP anycast can effectively limit the worst-case catchment inefficiencies experienced by global IP anycast by directing clients to regional IPs. For instance, regional anycast reduces the 90th percentile client latency for Imperva in North America from ms110 to ms38. However, compared to global IP anycast, regional IP anycast suffers DNS mapping sub-optimality. DNS may map a client to a sub-optimal regional IP that does not include a client’s low-latency CDN sites, offsetting regional IP anycast’s advantages of reducing catchment inefficiencies.

Finally, we study how much benefit regional anycast can have over global anycast

without DNS mapping sub-optimality using the Tangled testbed [BCdV⁺21] (§ 3.7). We use a latency-based scheme to partition the Tangled testbed into regions and assign each RIPE Atlas probe to the region that includes its lowest-latency site. We then deploy both global IP anycast and regional IP anycast on the Tangled testbed. In this case, regional IP anycast can achieve lower client latency than global anycast in all geographical regions. This result highlights the performance potentials of regional IP anycast.

An inherent limitation of this work is that we measure client latency using RIPE Atlas, like many previous studies [LLSB18b, MUF19a, KLA⁺21]. Using a different set of clients, one may observe different latency values. Despite this limitation, we believe this work makes the following general contributions:

- We study in detail the deployments and performance of two regional IP anycast CDNs and compare one CDN’s performance with its comparable global IP anycast system. To the best of our knowledge, this work is the first extensive study of regional IP anycast CDNs.
- We validate the performance advantages of regional anycast experimentally and discover its drawbacks in certain circumstances. We show that regional IP anycast can effectively mitigate the worst-case catchment inefficiency problem experienced by global IP anycast by directing clients to regional IP anycast addresses, but DNS mapping sub-optimality may offset some of this effect.
- We experiment with a latency-based region partition and client mapping scheme that addresses DNS mapping inefficiencies using the Tangled testbed. We find that this method reduces the latency for RIPE Atlas probes in all geographic areas compared to global anycast. This experiment shows the performance potentials of regional IP anycast.

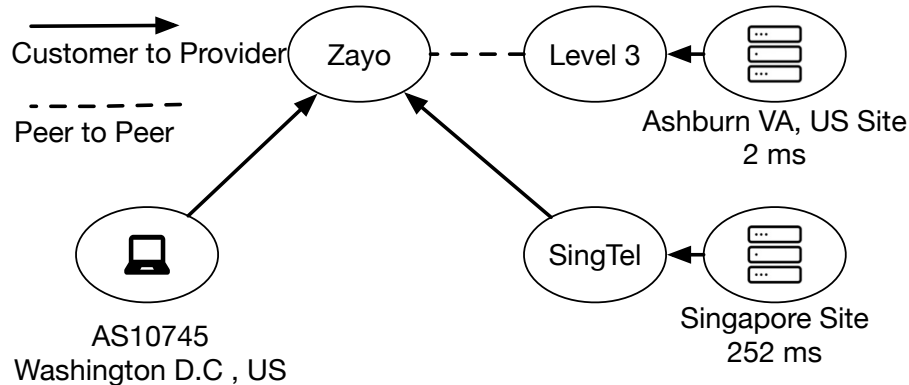


FIGURE 3.1: With a global anycast configuration, the probe in Washington D.C. reaches the CDN site in Singapore; with regional anycast, the probe reaches the site in Ashburn, Virginia.

Ethical Considerations Active measurements such as issuing pings and BGP announcements can cause extra load on the Internet infrastructure. We mitigate these concerns by conducting our measurements at reasonably low rates (i.e., only one round of ping or traceroute for each anycast IP address) with publicly accessible infrastructure. Our BGP announcements use only prefixes that Tangled controls and Tangled’s AS number. The prefixes we use do not serve any clients. We only measured the CDN providers with ping and traceroute, and we did not retrieve webpages which could incur extra costs for their customers. This work raises no other ethical issues.

3.2 Background and Motivation

In this section, we discuss the catchment inefficiency problem of IP anycast, the challenges in addressing it, and our motivation to study regional IP anycast.

3.2.1 The Catchment Inefficiency Problem

IP anycast is a popular technique used by CDNs to direct client traffic to their sites [HZWS18]. With this technique, a CDN operator relies on the inter-domain routing protocol BGP to select the site a client reaches. However, being a policy-

routing protocol, BGP often fails to route a client to a low-latency site. Figure 3.1 shows an example we observe in our measurements. The CDN we study has two involved sites: one connected to Level 3 in Ashburn, Virginia, and the other connected to SingTel in Singapore. When both sites announce the same global IP anycast prefix, the probe located in Washington D.C. reaches the Singapore site, as SigTel is the customer of the probe’s provider Zayo [fAIDACc], while Level 3 is Zayo’s peer. Under common BGP policies, ISPs prefer customer routes to peer routes. So the latency from this probe to the CDN is inflated by ms250. Furthermore, for routes with the same policy preference, BGP uses other non-performance-related metrics, such as AS path lengths, to select routes. Since a large AS may span multiple continents, routes with shorter (or the same) AS path lengths may have longer latencies than routes with longer (or the same) AS path lengths. A recent study [KLA+21] shows that for Microsoft CDN (a global IP anycast system), nearly 30% of users experience more than 30 ms latency inflation.

3.2.2 Challenges

There exist several proposals to improve client latency in a global anycast system. Ballani *et al.* [BFR06a] proposed to deploy anycast sites within a single provider. This proposal effectively limits BGP’s policy routing, but it is sometimes necessary to connect an anycast system to many ISPs for scalability and robustness. Li *et al.* [LLSB18b] proposed to introduce a new BGP attribute that encodes an anycast prefix’s geographical origin. However, introducing changes to BGP is difficult in practice.

Alternatively, McQuistin *et al.* [MUF19a] proposed DailyCatch, a system that uses routine measurement to choose between a transit-provider-only and an all-peer configuration for an anycast system. This approach can effectively choose the better configuration between the two measured configurations, but can not opti-

mize beyond that. Catchment inefficiencies can exist under either configuration. Zhang *et al.* [ZSZ⁺21] proposed AnyOpt, which uses pair-wise BGP experiments to choose an optimal site configuration for an anycast system among all possible site deployments. But pair-wise experiments do not scale to large networks.

3.2.3 Regional IP Anycast

Regional IP anycast emerged as a promising approach to address the catchment inefficiency problem [Mah15, CSG⁺18, HZWS18]. With regional IP anycast, a CDN divides its sites into multiple distinct geographic regions. It then assigns a distinct IP anycast prefix to each region. Sites in the same region will announce the same IP anycast prefix. We refer to such an IP prefix or address as a *regional IP prefix/address* or *regional IP* for short. The CDN then configures its DNS servers to assign a regional IP to a client based on the client’s location. In the example shown in Figure 3.1, with the regional anycast configuration, the CDN announces different IP anycast prefixes in U.S. and Asia. The probe in the U.S. receives the U.S. IP prefix, and consequently, it reaches the Virginia site and enjoys a ms2 RTT.

Compared to other proposals, regional IP anycast does not require changes to BGP, nor does it restrict how an anycast network connects to its providers. In addition, it scales to large networks. It is complementary to DailyCatch, as it can limit catchment inefficiencies under various provider configurations.

3.2.4 Motivation

Despite its potential advantages, regional IP anycast is not well studied or widely deployed. In a blog article [Mah15], LinkedIn described how they migrated their private CDN to a “prototype” regional anycast system and measured its client latency distribution. But their study is limited to LinkedIn’s private CDN, which is primarily located in North and South America. Hao *et al.* [HZWS18] reported that two out of

the top 20 CDNs (Edgio and Imperva) employ regional IP anycast without further performance analysis. Calder *et al.* [CSG⁺18] discussed the performance difference for the Microsoft CDN when it employs a regional vs. a global anycast configuration.

This work aims to understand how regional anycast is deployed in practice and experimentally examine its performance benefit compared to global anycast. This study can provide new insight into how to design an anycast-based system that achieves low client latency worldwide. In addition, if this study experimentally validates the performance benefit of regional anycast, it may motivate more CDNs to adopt regional anycast.

3.3 Measurement Infrastructure

We use two publicly available platforms: RIPE Atlas [Sta15b] and the Tangled testbed [BCdV⁺21] to conduct our experiments. Our experiments were conducted in August 2022.

3.3.1 RIPE Atlas

RIPE Atlas is a measurement infrastructure that has more than ten thousand probes distributed around the world, each with the ability to execute pre-defined measurements periodically. Each probe’s geographic location is publicly available. We leverage RIPE Atlas’s user-defined measurement feature to send DNS queries, ping packets, and traceroute queries. We take the following steps to address the limitations of RIPE Atlas to suit our measurement purposes.

First, RIPE Atlas probes use user-reported geo-locations, which may contain errors, but we use the probes’ built-in geocodes as ground truth to calculate the distance between a probe and its catchment site. To mitigate this issue and obtain stable measurement results, we discard probes (1) with unreliable geocodes using the methods described in [GSH⁺17] and (2) probes that do not have a built-in stability

tag (e.g., "system-ipv4-stable-1d"). After this step, we retain 9,700+ probes out of 11,000+ RIPE Atlas probes.

Second, RIPE Atlas probes are unevenly distributed across different geographic areas and Autonomous Systems (ASes). An uneven distribution may lead to under- or over-estimation of performance in certain geographic areas or ASes. To address this limitation, we group the probes by $\langle \text{city}, \text{AS}_i \rangle$ pairs and present statistics based on probe groups as in [MUF19a]. We obtain the city code of a probe by mapping the probe to its closest airport within the same country and using the airport's International Air Transport Association (IATA) code $[(\text{IA})]$ as the probe's city code. We use the probe's built-in AS number to identify its AS. Then, we use the median value measured from each $\langle \text{city}, \text{AS}_i \rangle$ probe group to represent the performance of a client residing in the same city and AS. Without specific mention, all CDFs, percentage, and percentile values we present here are computed based on probe groups, rather than individual probes. We obtain 6100+ unique probe groups.

Finally, RIPE Atlas has much more probes in Europe and North America than in other continents. Such bias may lead to results that overestimate (or underestimate) regional IP anycast performance in different regions. To address this limitation, we separately present the performance results of the probes in different geographic areas. We define the four areas as follows:

- **EMEA**: Europe, Middle East, and Africa. This area has 3,859 unique probe groups and 6,917 unique probes.
- **NA**: North America, excluding countries in Central America. This area has 1,154 unique probe groups and 1,716 unique probes.
- **LatAm**: South America and countries in Central America. This area has 141 unique probe groups and 177 unique probes.

- **APAC**: the rest of the globe. This area has 613 unique probe groups and 950 unique probes.

We note that this area definition is based on the location of a RIPE Atlas probe and is independent of the CDN region partition schemes we soon discuss.

3.3.2 The Tangled Testbed

Tangled is a worldwide open-access IP anycast testbed. It has 12 sites distributed around the world. We use Tangled to experience a latency-based regional anycast scheme and to compare the performance of global anycast with latency-based regional anycast (§ 3.7.2). We also considered using the PEERING testbed [SACKB19], but it has no site in Asia and the Pacific area. So we only used the Tangled testbed. We list the site distribution of Tangled by geographic area in Table 3.1.

3.4 Deployments

In this section, we dissect the deployments of two regional IP anycast CDNs: Edgio and Imperva. We describe how we conduct measurements to answer the following questions:

- How do these regional IP anycast CDNs assign their clients to regional IP addresses?
- How do these regional IP anycast CDNs partition their sites and announce regional IP prefixes?

3.4.1 Identifying Regional IP Anycast CDNs

First, we identify the set of public CDNs that deploy regional IP anycast. To do so, we acquire the top apex domain list from Tranco [LVT⁺19] in April 2022. An apex domain is a two-level domain [HSF19], e.g., `example.com`. We then use the

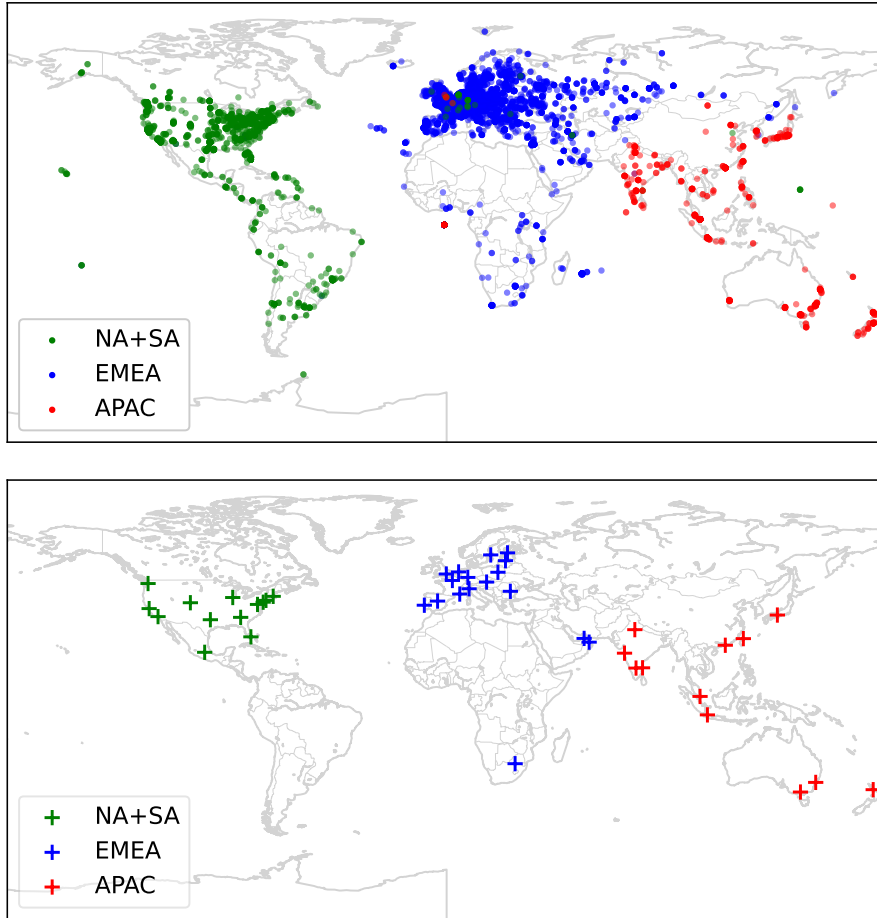


FIGURE 3.2: Edgio-3 (www.straitstimes.com) divides its sites into three regions

CDNFinder’s API [Peta] to identify the CDN providers of each domain. Specifically, we feed CDNFinder the www-prefixed hostname (i.e., the website) of each apex domain, e.g., `www.example.com`. CDNFinder returns the CDN providers a website uses by examining the response header of each resource on the landing page of the website. Since each resource identifier is possessed by a hostname, we can count the number of hostnames a CDN provider serves. We choose the top-15 CDN providers ranked by the number of hostnames they serve. The top-15 CDN providers cover 65.7% of all Tranco’s top-10k domains. By manually examining their official technical articles or configuration documents (see Appendix 3.10.2), we identify that Edgio and

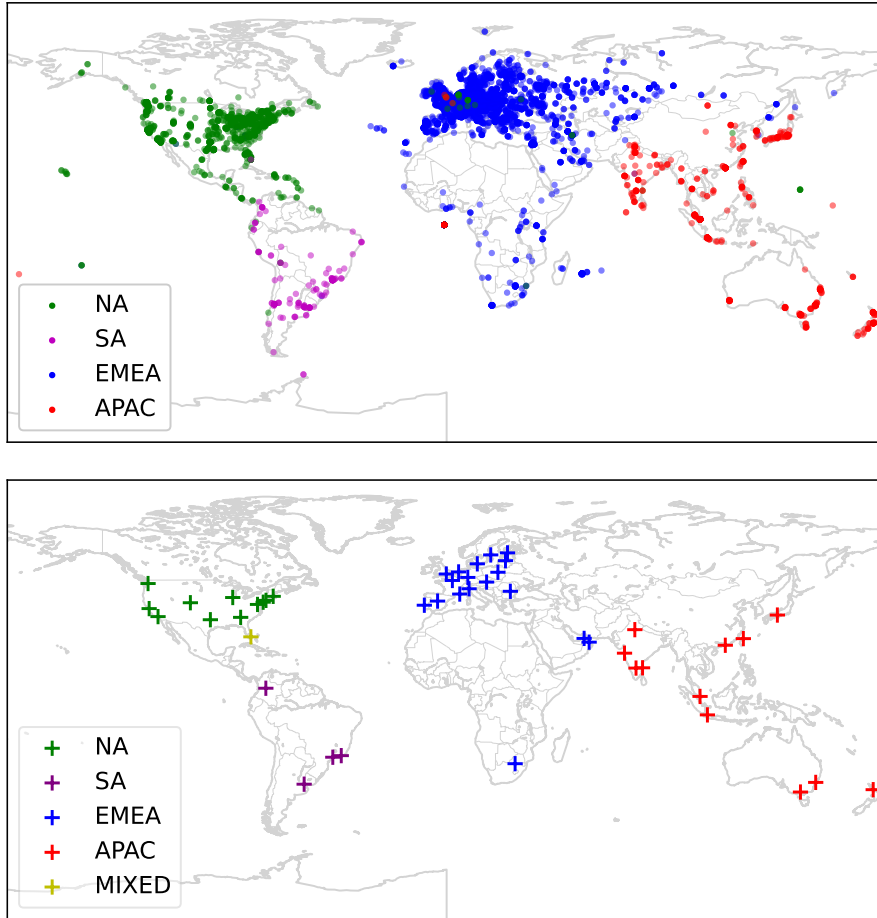


FIGURE 3.3: Edgio-4 (www.asus.com) divides its sites into four regions

Imperva are the only two CDNs that deploy regional anycast among the top-15 CDN providers, consistent with a prior study [HZWS18]. Therefore, we focus our study on these two CDNs.

3.4.2 Customers of Regional IP Anycast CDNs

Second, we select the representative customers of a regional IP anycast CDN. A CDN provider may negotiate different service packages with its customers and use different system configurations to implement different service packages. As this work does not aim to unveil various service packages of a CDN, we select and measure representative CDN customers to understand how the CDNs enable regional IP anycast for their

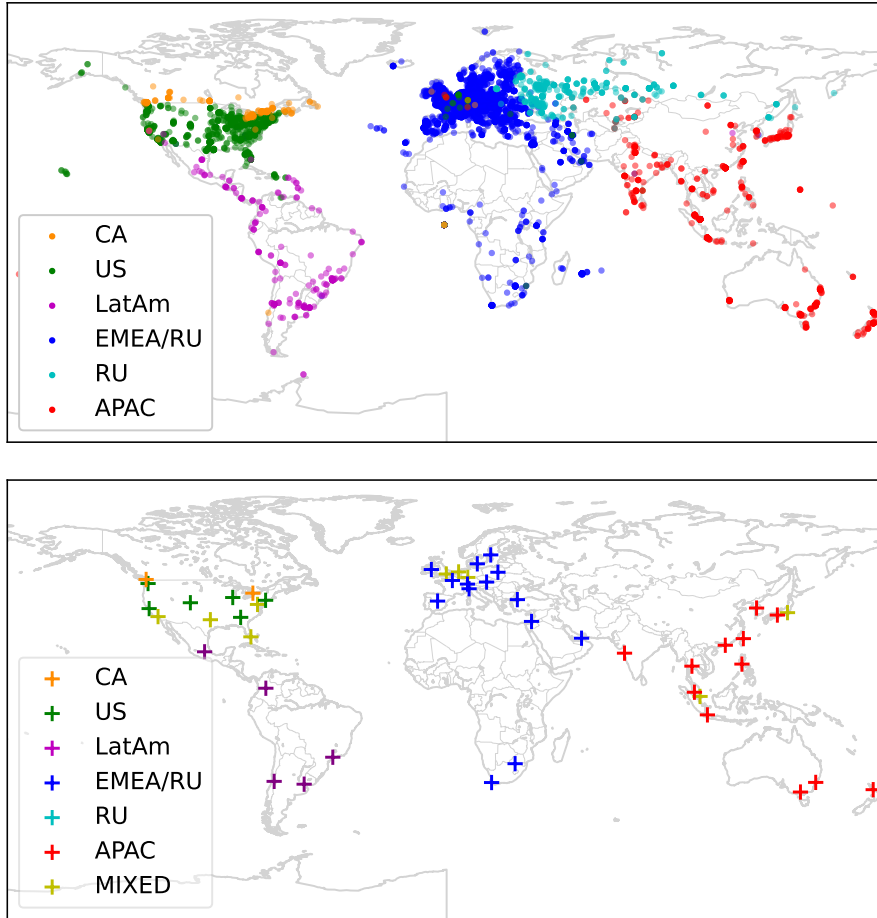


FIGURE 3.4: Imperva-6 (www.stamps.com) divides its sites into six regions

customers in commercial platforms.

For this step, we first resolve all hostnames uncovered by CDNFinder to use Edgio or Imperva as their CDN providers to IP addresses. The results from CDNFinder in the previous step show that 2.98% of the top-10K apex domains are using Edgio or Imperva, in which Edgio serves 209 domains and Imperva serves 89 domains. We extract 187 (96 and 91, respectively) distinct hostnames that point to Edgio or Imperva from these domains. To emulate a worldwide clientele, we compile a list of /24 client IP prefixes that cover the IP address span of the entire RIPE Atlas. We then use Google DNS [Mul] with the EDNS Client Subnet Extension (ECS) [CvdGLK16]

to resolve all hostnames, an approach used in [CFH⁺13b, JLK⁺21].

We discover the number of unique IP prefixes each hostname resolves to and filter those hostnames that are not served by Edgio or Imperva’s regional IP anycast networks. We record the IP address(es) (the A record) in each DNS response we receive. For 52.1% (50 out of 96) Edgio’s hostnames, we receive three different IP addresses for each of these hostnames for the emulated world-wide clientele. We refer to this set of hostnames as *Edgio-3*. For 35.4% (34 out of 96) Edgio’s hostnames, we receive four different IP addresses for each of the hostnames. We refer to this set of hostnames as *Edgio-4*. In contrast, the majority of Imperva’s hostnames resolve to the same number of IP addresses. For 85.7% (78 out of 91) Imperva’s hostnames, we receive six different IP addresses for each hostname. We refer to this set of hostnames as *Imperva-6*.

For the rest of the hostnames CDNFinder uncovers to use Edgio or Imperva, they either resolve to one IP address, or numerous IP addresses that match the number of published CDN sites, or IP addresses that belong to other CDNs. We conclude that these hostnames are not served by Edgio’s or Imperva’s regional IP anycast CDN and exclude them from this study.

3.4.3 Client Partitions

To characterize which IP address a client in a geographic region receives, we use RIPE Atlas probes to resolve a hostname that belongs to a customer of Edgio’s or Imperva’s regional IP anycast CDN. We then cluster RIPE Atlas probes based on the IP addresses they receive. For each hostname, we group the probes that receive the same IP address together. We run the experiments for all hostnames in the three sets: *Edgio-3*, *Edgio-4*, and *Imperva-6*. We find that the clustering results for each hostname in the same set remain the same.

For an in-depth study, we pick one hostname which illustrates stable and consis-

tent deployments while remaining top-ranked in Tranco’s list from each hostname set. We eventually choose `www.straitstimes.com`, `www.asus.com`, and `www.stamps.com` from the Edgio-3, Edgio-4, and Imperva-6 hostname set, respectively. The first row in Figure 3.4 shows how the probes that receive the same regional IP addresses for the three hostnames are distributed globally. We depict the probes that receive the same regional IP with the same color.

For both Edgio hostnames and Imperva hostnames, the client partition appears to happen at the continent or large-country level. For example, for Edgio-3 hostnames, the probes in North America and South America receive the same regional IP, and the probes in Europe, Africa, and Middle East receive the same regional IP. For Imperva hostnames, the probes in Russia receive different regional IPs from those in Europe; and the probes in the U.S. and Canada are also separated from each other, each receiving a distinct IP.

We use the country code of each RIPE Atlas probe to evaluate whether the probes in the same country always receive the same regional IPs for the same hostname and find that the majority of countries receive only one regional IP. For the probes from all 172 countries, 81.7%, 84.7% and 79.3% of the countries only receive one regional IP for the representative Edgio-3, Edgio-4, and Imperva-6 hostnames, respectively. For countries that receive two or more regional IPs, we find two cases. In the first case, the IP addresses of the probes are geo-located to different countries. For instance, the probes whose IPs belong to international transit providers are often geo-located to their home countries, not the countries they reside in. Second, the countries are either at the border of two regions or across two different continents. For instance, 10 out of 547 probes in Russia receive the EMEA regional IPs in Imperva-6.

Takeaways: Both Edgio’s and Imperva’s regional anycast CDNs map clients to regional IPs largely by geographic locations, *i.e.*, the continents or the countries they

reside in.

3.4.4 *CDN Site Partitions*

Next, we aim to understand how Edgio and Imperva partition their hosting sites into different regional IP anycast networks. To do so, we need to locate the CDN sites that announce a regional IP anycast prefix. We describe this process as follows.

First, we obtain the locations of Edgio and Imperva’s CDN sites. Both Edgio and Imperva publish their Points of Presence (PoPs) on their websites [Imp, Edg]. We aggregate those locations at the city level and combine multiple PoPs in the same city as one site. We use these published site locations at the city level as the ground truth locations of their sites.

Second, to discover the location of the CDN site announcing a regional IP, we traceroute from each RIPE Atlas probe to the regional IP the probe receives from DNS and geo-locate the penultimate hop (referred to as *p-hop* hereafter)’s IP address from the traceroute output. We use the ground truth CDN site location to map each p-hop to its closest CDN site, as in [MUF19a]. This step also reveals the location of the catchment site of a probe, which we use to compute the distance between a probe and its catchment site in § 3.5.

IP geo-location is a task performed by many previous studies [MUF19a, AGE⁺20, LLSB18b, SMA03, SMW02, LHM⁺21, LHC19, GSH⁺17]. This step involves substantial work, but it is not our main contribution. We summarize the work here and leave the detailed description in Appendix 3.10.1. We first use the geo-hints in the reverse DNS (rDNS) name of a p-hop’s IP to infer its location [LHM⁺21, LHC19]. If the geo-hints are unavailable, we use the location of a RIPE Atlas probe whose RTT is within $\text{ms}1.5$ to the p-hop [GSH⁺17] to infer its location. The threshold is chosen according to the typical size of a metropolitan area, as the speed-of-light latency in fiber is roughly 100 km per 1 ms RTT (referred to as RTT Range). Third, if we

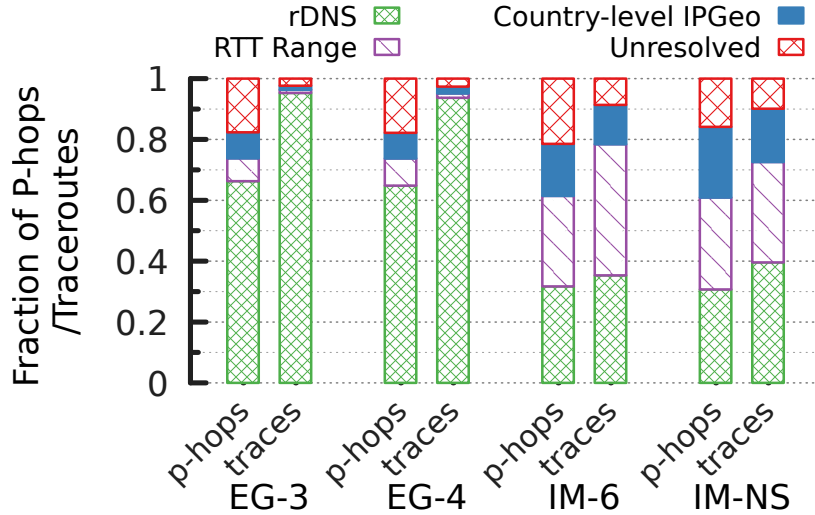


FIGURE 3.5: The *p-hops* bars show the fraction of p-hops successfully IP-geolocated by a technique and the fraction of unresolved p-hops for each network. The *traces* bars show the fraction of traceroutes whose p-hops are successfully IP-geolocated by a technique and the fraction of traceroutes whose p-hops are unresolved. EG: Edgio; IM: Imperva.

Table 3.1: The number of sites in each geographic area of different networks (*EG/IM-Pub*: *Edgio and Imperva's Published sites*).

	EG-3	EG-4	EG-Pub	IM-6	IM-NS	IM-Pub	Tangled
APAC	14	15	19	16	17	17	2
EMEA	15	16	26	15	15	15	5
NA	13	12	24	12	12	12	3
LatAm	1	4	10	5	5	6	2
Total	43	47	79	48	49	50	12

cannot resolve a p-hop's geo-location after the previous two steps, we use the country information from three IP-geolocation databases (MaxMind [Max], ipinfo [IPI], and EdgeScape [Tec]) to infer the location of the p-hop. If all IP-geolocation databases return the same country location for the p-hop, and the CDN provider only lists one site in the country, we use the listed site location as the p-hop's location (referred to as country-level IPGeo).

Figure 3.5 summarizes the fraction of p-hops we successfully geo-locate for the representative Edgio-3, Edgio-4, and Imperva-6 hostnames by each technique. We also show the fraction of traceroutes whose p-hops are successfully ip-geolocated by

each technique. As we can see, we are able to resolve the majority of the p-hops observed in the traceroute outputs.

Table 3.1 shows the CDN sites we uncover after mapping p-hops to their CDN sites. We uncover 49 out of 50 Imperva’s published sites. Edgio publishes 79 sites, while we only uncover 43 for the Edgio-3 hostname and 47 for the Edgio-4 hostname. This result indicates Edgio uses different network configurations for different customers.

CDN Site Partitions After the site-numeration step, we are able to depict the location of a CDN site that announces a regional IP prefix. Figure 3.4 shows the result. We represent a CDN site with a colored cross symbol. The color corresponds to the regional IP it announces. We use the same color-coding scheme as in client partitions. If a site announces more than one regional IPs, we color the site yellow.

We make the following observations. First, in both Edgio and Imperva, client and site partitions are mostly consistent. Edgio partitions its sites into four regions, while Imperva into six regions. Clients in the same geographical regions receive the regional IPs originating from the CDN sites in the same regions except two cases. For the Edgio-3 hostname, the probes in South America receive North America’s regional IP. This finding shows that different customers received different types of services from a CDN. Some customers’ content may not be hosted by Edgio’s South America sites. So clients will not be directed to those sites. For Imperva, although most probes in Russia receive distinct regional IPs as shown in Figure 3.4, these IPs are announced by three sites in Europe (Amsterdam, Frankfurt, and London), which also announce the EMEA regional IPs. We do not observe any Imperva sites in Russia.

Second, most sites only announce one regional IP prefix, but there exist sites that announce multiple regional IP addresses. We refer to this behavior as cross-region

announcement. A closer examination of Figure 3.3 and Figure 3.4 show that the sites which announce multiple regional IPs usually locate near the border of two different regions. Cross-region announcement can help shorten client latency without adding additional site. For instance, Edgio-4’s “mixed” site in Florida, U.S. can serve both clients in North America and in South America. However, we later discover that cross-region announcements can lead to inefficient catchments, increasing the RTTs of some probes (§ 3.5.2).

Do CDN sites change the regional IP prefixes they announce over time? We enumerated the CDN sites that announce the regional IP prefixes for five hostnames in Edgio-3, five hostnames in Edgio-4, and three hostnames in Imperva-6 weekly for two months. We find that for the same hostnames, the sites that announce their regional IP prefixes in this two-month period remain the same.

Takeaways: Edgio and Imperva’s site and client regional partitions are largely consistent, but there exist regions in which clients are assigned to regional IPs originating from sites in different geographic regions.

3.4.5 Reachability of Regional IP addresses

Are regional IP prefixes globally reachable? We are interested in understanding whether a CDN restricts the BGP announcements of a regional IP anycast prefix to a geographic area. If it does, a client outside a geographic area cannot reach the regional IP prefix announced from that area. To do so, we send ping packets from RIPE Atlas probes to the regional IP addresses they do not receive from DNS. We run this experiment for each representative Edgio-3, Edgio-4, and Imperva-6 hostname. The results show that all probes can reach the regional IP addresses DNS returns to the probes in other regions. Global reachability provides robustness to regional anycast: even if DNS returns a regional IP unintended for a client’s geographic area,

the client can still reach the CDN site announcing the unintended regional IP.

3.5 Performance

In this section, we study the performance of Edgio’s and Imperva’s regional IP anycast CDNs. We aim to answer the following questions:

- How effectively does DNS map a client to a lowest-latency or close-to-lowest-latency regional IP?
- What are the performance benefits of regional IP anycast compared to global IP anycast?

We use two metrics for this study: network latency and geographic distance.

Network latency or latency We measure a probe’s round trip time (RTT) to its catchment site to quantify the client latency distribution achieved by an anycast system. The lower the latency, the better the performance.

Geographic distance We also measure the geographic distance between a probe and a CDN site as in previous work [dOSHK17a, LLSB18b, KLA⁺21]. Since we have inferred the location of a probe’s catchment site in § 3.4.4, we can compute the geographic distance between a probe to its catchment CDN site.

3.5.1 DNS Mapping Efficiency

To study DNS mapping efficiency, we measure how often DNS returns the lowest- or a close-to-lowest-latency regional IP to a client. We consider any regional IP with less than 5 ms RTT difference to a probe’s lowest-latency regional IP as close-to-lowest regional IP. We instruct RIPE Atlas probes to send DNS queries to the representative hostnames served by Edgio and Imperva and record the returned IP addresses. Then

Table 3.2: \checkmark/\times Region indicates whether a probe receives a regional IP intended for its geographic location or vice versa; ΔRTT is the difference between a probe’s RTT to the regional IP returned by DNS and the lowest one among its RTTs to all regional IPs.

Condition	CDN	Local DNS			Authoritative DNS				
		APAC	EMEA	NA	LatAm	APAC	EMEA	NA	LatAm
$\Delta RTT < 5ms$	Edgio-3	94.2%	96.7%	98.6%	99.1%	94.2%	98.7%	98.8%	99.1%
	Edgio-4	91.0%	97.3%	97.4%	92.9%	94.8%	98.8%	98.2%	94.7%
	Imperva-6	86.3%	78.3%	88.0%	85.6%	87.1%	78.0%	89.3%	86.5%
\checkmark Region, $\Delta RTT \geq 5ms$	Edgio-3	3.5%	1.3%	0.2%	0.0%	4.0%	0.8%	0.1%	0.0%
	Edgio-4	3.6%	0.7%	0.7%	3.5%	3.4%	0.7%	0.6%	3.5%
	Imperva-6	10.5%	19.4%	8.2%	11.7%	10.8%	21.0%	9.0%	12.6%
\times Region, $\Delta RTT \geq 5ms$	Edgio-3	2.4%	2.0%	1.2%	0.9%	1.8%	0.5%	1.1%	0.9%
	Edgio-4	5.4%	2.0%	2.0%	3.5%	1.8%	0.4%	1.2%	1.8%
	Imperva-6	3.2%	2.3%	3.8%	2.7%	2.1%	1.0%	1.7%	0.9%

we instruct each probe to ping all regional IPs associated with a hostname. Because DNS mapping results depend on whether a local resolver implements EDNS Client Subnet Extension [CvdGLK16], we run these experiments with two different DNS configurations:

Local DNS (LDNS) we configure each RIPE Atlas probe to use its local DNS resolver to send DNS queries when resolving a hostname.

Authoritative DNS (ADNS) As a comparison, we configure each RIPE Atlas probe to send DNS queries directly to CDN’s authoritative name servers that are responsible for resolving customer domains to CDN’s IPs. By doing this, the authoritative name servers of the CDN can determine the IP addresses they return based on the querying clients’ IP addresses.

We divide the experimental results into three groups and tabulate the results in Table 3.2. In the first group, DNS effectively returns a regional IP with less than 5 ms RTT difference to a client’s lowest-latency regional IP ($\Delta RTT < 5 \text{ ms}$). We consider 5 ms a reasonable threshold to differentiate the performance of two CDN sites. We consider DNS mapping as efficient in this case. In the second and third groups, DNS returns a regional IP whose RTT exceeds a client’s minimum RTT among all regional IPs by 5 ms.

We sub-divide the second and third groups by what causes DNS mapping inefficiencies. According to the study in § 3.4.3, regional anycast maps clients based on their geographic locations; clients residing in the same geographic regions often receive the same regional IPs. If DNS maps a client to a regional IP outside its geographic area, we refer to it as incorrect region mapping (\times Region). If DNS maps a client to a regional IP intended for clients in its geographic area (\checkmark Region), but the RTT to this regional IP exceeds a client’s minimum RTT to a regional IP, we

consider this case as sub-optimal region mapping.

From Table 3.2, we can see that for Edgio’s two representative hostnames, DNS maps more than 90% of the probes in all regions to regional IPs within 5 ms difference to their lowest-RTT regional IPs. Both incorrect region mapping and sub-optimal region mapping contribute to DNS inefficiencies. DNS mapping inefficiencies are more dominant in APAC and LatAm regions.

Imperva-6’s DNS mapping is less efficient than Edgio’s. The majority of the DNS inefficiencies are caused by inefficient regional mappings as shown in the (\checkmark Region, $\Delta RTT \geq 5ms$) row in Table 3.2, due to Imperva-6’s six-region partition scheme. Imperva-6 partitions the probes and sites in NA into two regions: Canada and the U.S. Around 10% probes in Canada and the U.S. are located near the Canada and U.S. border. Some of these probes have shorter RTTs to regional IPs in the other country. As a result, DNS maps only 88.0% of probes in NA to their low-latency regional IPs.

Similarly, Imperva-6 partitions Russia into a separate region. Probes in Russia receive distinct regional IPs which originate from three sites in Europe (Amsterdam, Frankfurt, and London). In this case, some probes in Russia have shorter RTT to EMEA (exclude Russia) regional IPs than to its own regional IPs and vice versa. For example, a probe in Russia reaches the site in Amsterdam, but its lowest-latency regional IP originates from a site in Copenhagen, Denmark in the EMEA region. Its latency to the EMEA regional IP is 30 ms less than the latency to the Russian regional IP. As a result, for the EMEA region, only 78.3% of the probes receive regional IPs within 5 ms to their lowest-latency regional IPs in the Imperva-6 CDN.

Takeaways: This study reveals that regional anycast experiences DNS mapping inefficiencies. Both IP-geolocation errors and DNS mapping based on a rigid regional partition of a client’s geographic location can lead to sub-optimal performance.

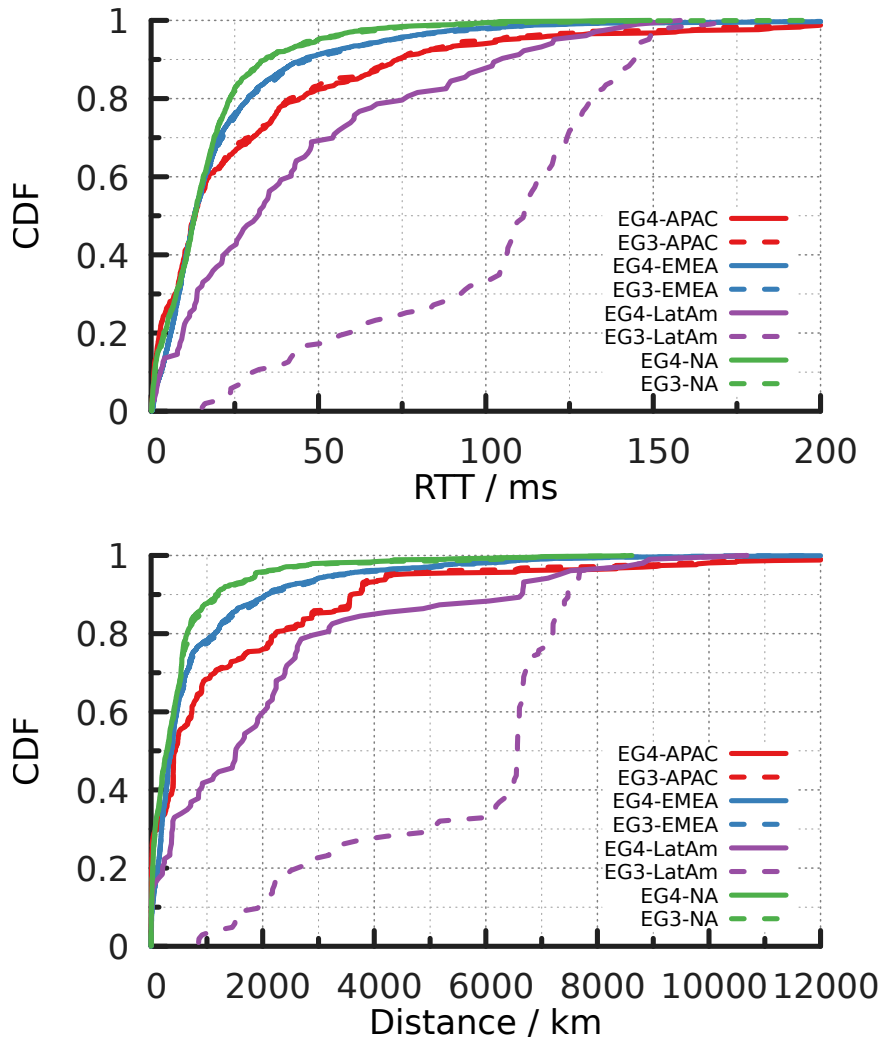


FIGURE 3.6: Edgio-3 vs. Edgio-4

3.5.2 Client Latency

In this section, we measure the performance of Edgio’s and Imperva’s regional anycast CDNs using the metrics we describe above: network latency and geographic distance. To measure client latency, we instruct a RIPE Atlas probe to ping the regional IP address it receives from DNS and record the RTT for each of the representative hostname of Edgio-3, Edgio-4, and Imperva-6 customers. In the meantime, we also plot the distance between a probe and the regional anycast site it reaches, as the

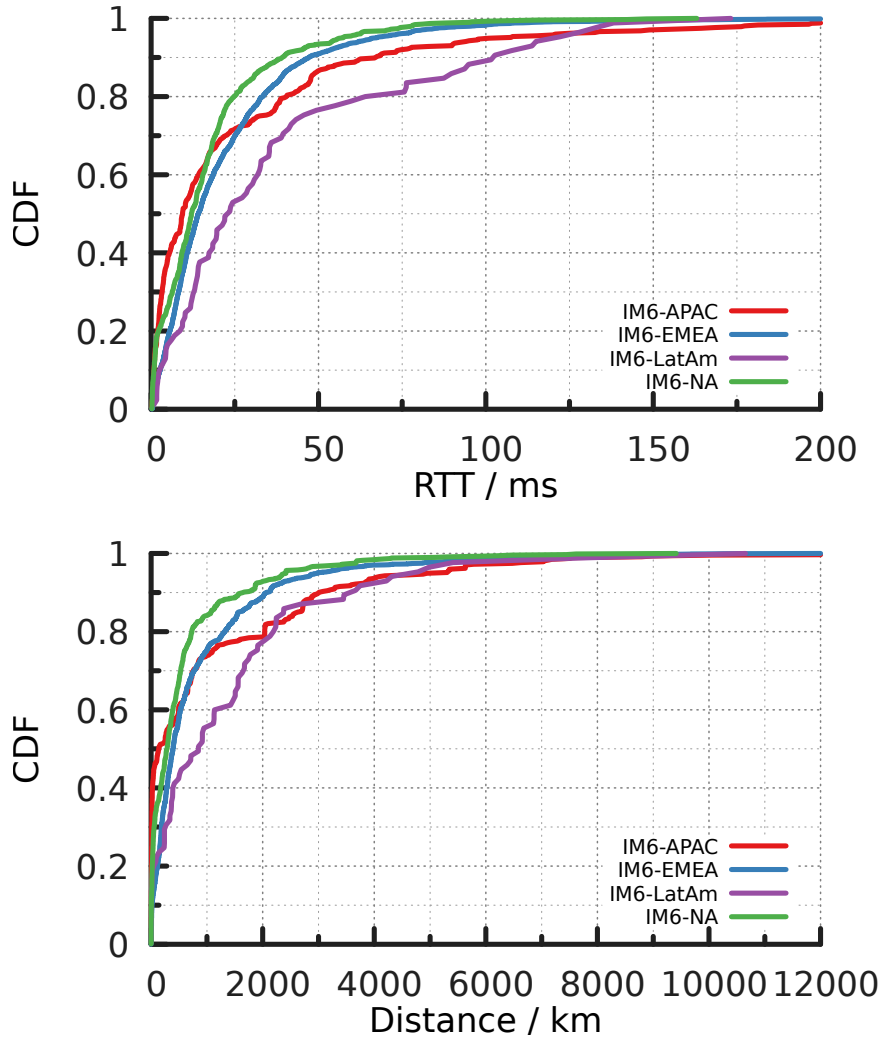


FIGURE 3.7: Imperva-6

speed-of-light latency lower-bounds the network latency. Figure 3.8 (a) and (b) show the cumulative distributions of the RTTs and the distance values of RIPE Atlas probes to Edgio-3, Edgio-4, and Imperva-6 regional anycast CDNs. We combine the measurement results for Edgio-3 and Edgio-4 for comparison.

We highlight two observations. First, the latency and distance values for probes in LatAm improve significantly in Edgio-4 compared to Edgio-3. The 80th percentile client latency decreases from 132 ms to 76 ms. Recall that Edgio maps the probes in South America to sites in North America in the Edgio-3 configuration. This result

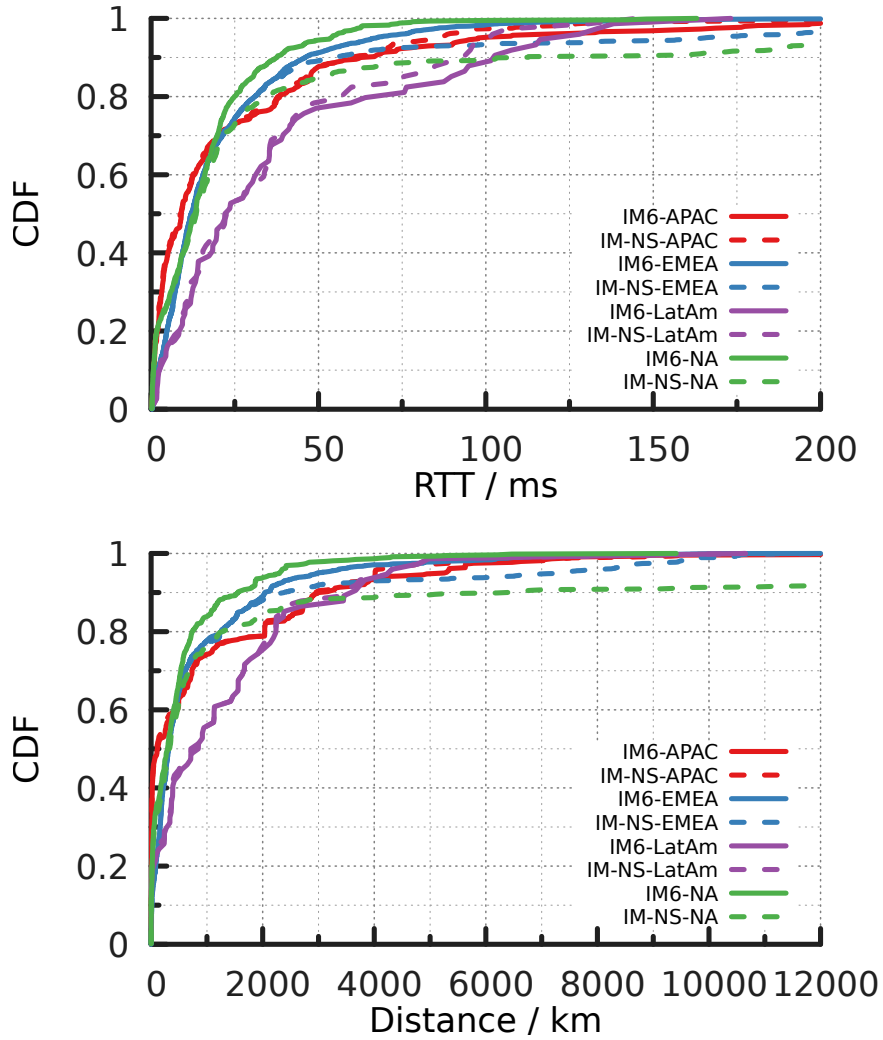


FIGURE 3.8: Imperva-6 vs. Imperva-NS with same config

shows that mapping clients to the regional IPs of nearby CDN sites improves client performance.

Second, for both Edgio and Imperva, the 98th-percentile client latency in the NA and EMEA regions is less than 100 ms, the human interaction application threshold [MCZ⁺20]. In the APAC and LatAm regions, however, there are more than 8.5% of probe groups in Edgio-4 and 5.2% of probe groups in Imperva-6 that have RTTs exceeding 100 ms. We examine why probes in those regions have latency values exceeding 100 ms and find two root causes for each region. For Imperva-6, 55.2% of

Table 3.3: The tail latency comparison between Imperva-6 and its DNS global anycast network (Imperva-NS). RTTs are in the unit of millisecond. Green indicates a 5+ ms latency reduction in Imperva-6 and red indicates a 5+ ms increase.

Percentile	Imperva-6 (Imperva-NS)			
	APAC	EMEA	NA	LatAm
80-th	38 (38)	31 (31)	25 (35)	68 (57)
90-th	63 (59)	45 (53)	38 (110)	102 (93)
95-th	98 (87)	67 (165)	54 (221)	120 (101)

the probe groups whose RTTs exceed 100 ms in the APAC region suffer from sub-optimal region mapping that DNS does not map them to the lowest-latency regional IP. And 27.6% of them are far away from their catchment sites. For example, we find 4 probe groups in China reached the site in California, US, as the site also announces Imperva-6’s APAC regional IP. For probe groups whose RTTs exceed 100 ms in the LatAm region, 71.4% of them have long paths to reach their CDN sites. One example is a group of probes in Argentina that reach their catchment site in Brazil detoured through Italy. Besides, 14.3% of them receive a regional IP originating from sites in different continents.

3.5.3 Regional vs. Global Anycast

Next, we aim to study the performance impact of regional anycast when compared to global anycast.

A Comparable Global Anycast Counterpart Ideally, we would like to study the performance of a regional anycast CDN in the context of what performance the CDN can achieve if it uses a global anycast configuration, *i.e.*, announcing one global IP anycast prefix from all sites to which it announces regional anycast IP prefixes to the same set of peers as in regional anycast with the same policy configurations. Such a study can unambiguously reveal the benefits or drawbacks of regional IP anycast.

Lacking access to a real-world CDN, we cannot conduct such a study. Instead,

we use the DNS global anycast network of a regional anycast CDN to emulate its comparable global anycast counterpart. A common practice among CDN providers is to deploy their authoritative name servers using global anycast at the same sites where they deploy hosting servers [SBK⁺20b, FMM⁺15b, CST15a]. Using the anycast site enumeration method we describe in § 3.4.4, we find that all 48 sites we uncover for Imperva-6 overlap with the sites of its DNS global anycast network. We refer to Imperva’s DNS global anycast network as Imperva-NS. Edgio’s CDN sites do not overlap significantly with its authoritative name server network so we exclude Edgio from this study.

Furthermore, we uncover the set of peering ASes to which a CDN announces both its regional IP anycast prefixes and its DNS global IP anycast prefixes. Even if a CDN deploys its hosting servers and its authoritative domain name servers at the same site, it may announce its regional CDN IP anycast prefixes and its global DNS IP anycast prefixes to different peers with different policy configurations. Uncovering the common set of peering ASes enables us to emulate a global anycast network that shares the same sites and peers with a regional anycast CDN. We map the valid penultimate hop (p-hop) in each RIPE Atlas probe’s traceroute to a regional anycast IP or to the global DNS anycast IP to the AS or the Internet Exchange Point (IXP) that owns the p-hop’s IP address. We use RouteViews’ BGP archive [Roub] of the same day when we collect the probes’ traceroutes and the published IP prefixes from PeeringDB [fAIDACa] to construct the IP-to-AS or IP-to-IXP mapping. In 49.0% of the traceroutes, the p-hops’ IPs belong to IXPs and are not visible in BGP. After this step, for each overlapping site between Imperva-6 and Imperva-NS, we construct the set of common peers (ASes or IXPs) at that site.

Moreover, we assume that Imperva does not apply different latency-impacting policies when it announces a regional anycast IP prefix or a DNS global anycast IP prefix to the same peer at the same site. We validate this assumption by comparing

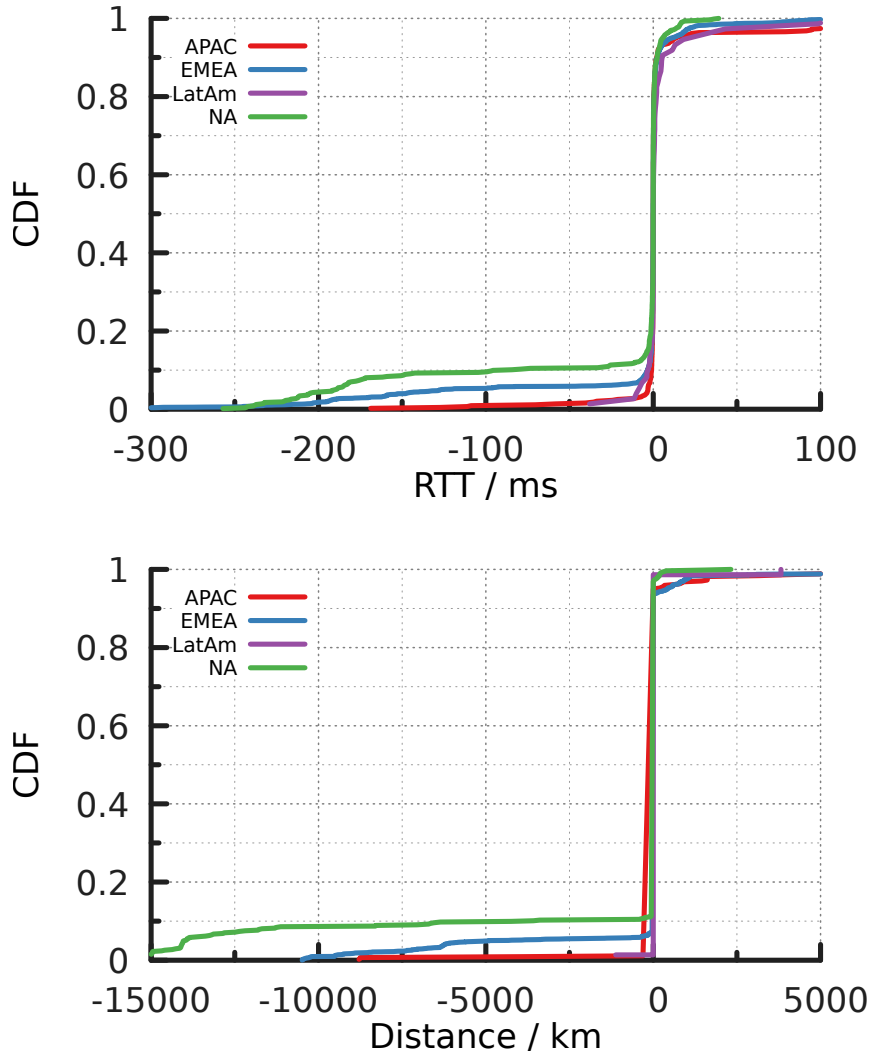


FIGURE 3.9: CDFs of RTT and distance differences between regional and global anycast.

the RTTs from the same probe reaching the same site via a regional IP or via a DNS global anycast IP. We find that the RTT differences are negligible (as shown in Figure 3.15 in Appendix 3.10.3).

After these steps, we consider that the part of Imperva-NS that includes the 48 overlapping sites with Imperva-6 and the overlapping set of peers with Imperva-6 at each site emulates the comparable global anycast counterpart of Imperva-6. This is because any site a probe reaches in the emulated global anycast network also an-

Table 3.4: We examine the number of probe groups that have better/similar/worse RTTs (5 ms chosen as the threshold) in regional anycast than in global anycast. In each performance group, we examine the percentage of probe groups that reach closer, same, or further sites.

Region (# probe groups)	$\Delta RTT < -5ms$	Closer Site	Same Site	Further Site
	$ \Delta RTT \leq 5ms$			
	$\Delta RTT > 5ms$			
APAC (440)	15	26.7%	60.0%	13.3%
	395	0.0%	99.7%	0.3 %
	30	3.3%	33.3%	63.3 %
EMEA (2529)	219	69.9%	26.0%	4.1 %
	2130	1.0%	98.1%	0.8 %
	180	1.7%	22.8%	75.6%
LatAm (74)	5	0.0%	100.0%	0.0%
	60	0.0%	100.0%	0.0%
	9	11.1%	77.8%	11.1%
NA (584)	79	79.7%	19.0%	1.3%
	473	0.6%	97.9%	1.5%
	32	0.0%	68.8%	31.3%

nounces a regional IP prefix to the same set of peers, and the RTT differences between reaching the site via a global anycast IP or a regional anycast IP are negligible.

Finally, to perform the comparison between regional anycast and global anycast, we instruct each RIPE Atlas probe to traceroute to its regional IP received via DNS when querying the representative hostname served by Imperva-6 and a global anycast IP of Imperva-NS. We filter the probes which do not have a valid p-hop in their traceroute outputs, or do not reach a common site in Imperva-6 and Imperva-NS, or do not reach their catchment sites via a common peer in Imperva-6 and Imperva-NS. In total, there are 4,417 probe groups with successfully resolved p-hops and we retain 82.1% (3,627 out of 4,417) of them after this step.

Comparison of Imperva-6 and Imperva-NS Figure 3.8 plots the CDFs of the probe groups’ RTTs to regional IPs in Imperva-6 and to global anycast IPs in Imperva-NS after excluding two networks’ non-overlapping peers and sites, respectively. For probes in EMEA and NA, we see improved tail latency for regional anycast. For probes in NA, the 90th percentile RTT decreases from 110 ms to 38 ms. For probes

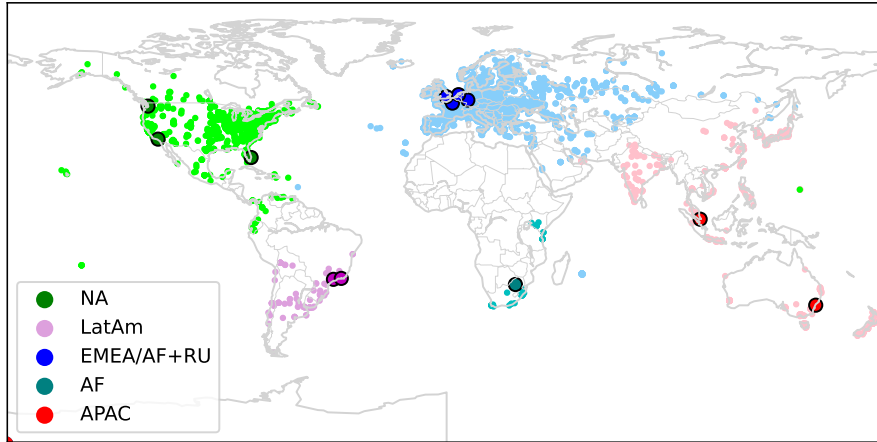


FIGURE 3.10: ReOpt’s site and client partitions

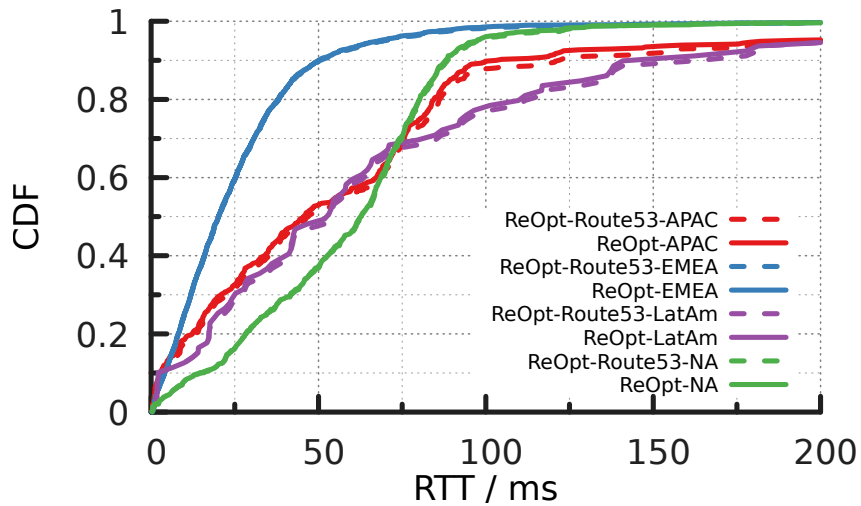


FIGURE 3.11: Regional w/ and w/o DNS configuration

in APAC and LatAm, regional anycast performs slightly worse than global anycast. Figure 3.9 plots the RTT difference and the distance difference between a probe’s catchment site in Imperva-6 and in Imperva-NS. We observe that the percentage of probes with a distance reduction in regional anycast correlates well with the percentage of probes with latency reduction.

Further, we examine in detail the probes whose RTTs differ by 5 ms in Imperva-6 and Imperva-NS and how the RTT differences correlate with a probe’s distance dif-

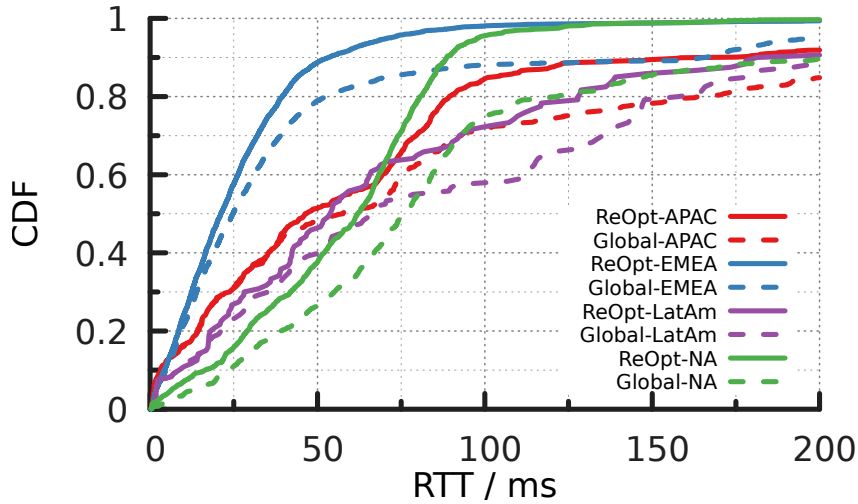


FIGURE 3.12: Regional w/ ReOpt partitions vs. Global

ference between its global anycast catchment site and its regional anycast catchment site. Table 3.4 summarizes the results. For the EMEA and NA regions, we find that when the probes achieve better performance in regional anycast, the majority of them reach closer sites in regional anycast than in global anycast. In the EMEA region, 69.9% (out of 219) probe groups that achieve more than 5 ms latency reduction, reach closer sites in regional anycast. In the NA region, 79.7% (out of 79) probe groups that achieve more than 5ms latency reduction reach closer sites.

For the EMEA region, 7.1% probe groups (180 out of 2529) experience more than 5 ms longer latency in regional anycast than in global anycast; the majority (75.6%) of them reach further away sites. This result is consistent with our observation in § 3.5.1 that DNS mappings in the EMEA region are inefficient. Overall, there are more probe groups in EMEA that improve their performance in regional anycast. So we observe that the client latency distribution has improved in regional anycast.

The probe groups with significant latency differences in other cases are few and we do not observe a consistent pattern.

For probe groups in each region that have similar performance in regional anycast

and global anycast, between 97.9% to 100% of them reach the same sites. For probe groups in each region that have better or worse performance, some of them also reach the same sites. We examine the traceroute outputs of those probes and find that some of them reach the same sites via different AS paths and others have the same AS paths but different RTTs. BGP’s route-selection uncertainty [ANC⁺15b] and route instability (*e.g.*, temporary congestion) can contribute to these cases.

Takeaways: In the EMEA and NA regions, Imperva’s regional anycast CDN effectively directs clients to nearby CDN sites and reduces client tail latency compared to global anycast. However, DNS mapping inefficiency reduces the performance of its regional anycast CDN in the EMEA region.

3.6 Insights

As we mentioned earlier, the performance benefits of regional anycast are mainly derived from the shorter distance to catchment sites. We want to further explain the benefits at the route level by examining the traceroute outputs of those probes which have shorter RTT in regional anycast. We divide those probes by whether their global anycast catchment sites also announce the regional anycast prefixes they receive.

For those probes whose global anycast catchment sites do not announce the regional anycast prefixes, they experience degraded performance in global anycast due to unexpected BGP routing policies. The first one is that ISPs always prefer customer routes to peer routes. As we illustrated in Section 3.2, this routing policy can make probes select routes from customer cones but to remote sites. The propagation of prefix announcements from the CDN to the non-tier-1 provider and subsequently to the tier-1 providers can easily lead to such situations occurring. We argue that it is common in global anycast since all sites announce the same prefixes.

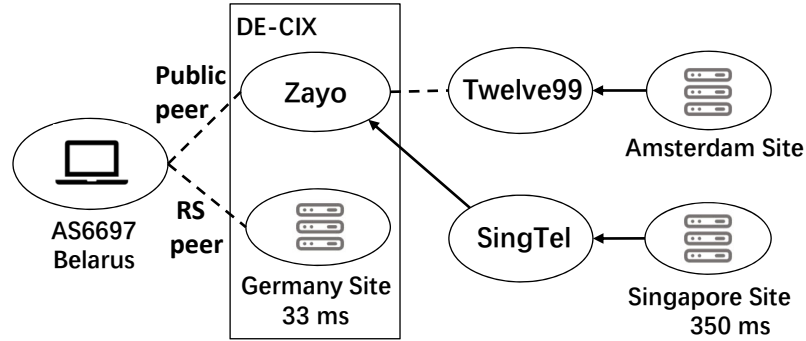


FIGURE 3.13: With a global anycast configuration, the probe in Belarus reaches the CDN site in Singapore; with regional anycast, the probe reaches the site in Germany.

The second relevant policy is that ISPs always prefer public peers to route server peers (in IXP). Public peers are network entities that directly exchange routes and traffic over the fabric of an Internet Exchange Point (IXP), without using a route server. In contrast, when routers receive peers' routes via a route server, they are known as route server peers. Routers generally prefer public peers over route server peers, as public peering allows for direct exchange of traffic and typically offers better performance compared to route server peering [SKC⁺17a].

As we show in Figure 3.13, Belarus has a public peer with Zayo and a router server peer with Imperva via DE-CIX. Zayo has routes to both the regional IP prefix and global IP prefix which are advertised by Twelve99 and SingTel, respectively. However, Zayo will not export its peers' routing information (*i.e.*, routes to the regional IP prefix) to Belarus. As a result, Belarus can only receive routes to the regional IP prefix from the route server peer and it will be directed to the site in Germany with regional anycast while to the CDN site in Singapore with global anycast.

When sites both announce the global IP prefix and regional IP prefix (the probes receive), we still find there are probes whose catchment sites are different. As depicted in Figure 3.14, probes in DINET have different routes to the global IP prefix

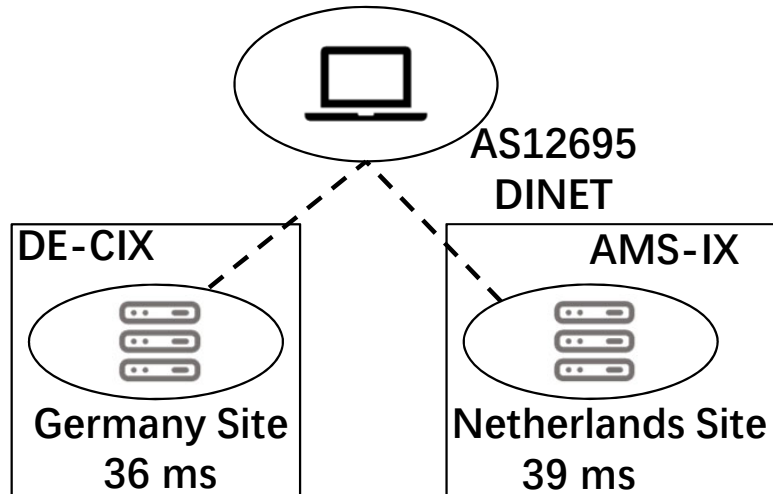


FIGURE 3.14: With a global anycast configuration, the probe in DINET reaches the CDN site in the Netherlands; with regional anycast, the probe reaches the site in Germany.

and regional IP prefix. One route traverses AMS-IX to the site in the Netherlands, while the other route traverses DE-CIX to the site in Germany. Despite both sites announcing the regional IP prefix and global IP prefix in the same way, there is still a difference between regional anycast and global anycast. We speculate that the routes differentiate due to other tie-breaking BGP policies, *e.g.*, arrival time. This factor does not have a significant impact on performance, as we can see that regional anycast has only a 3 ms reduction compared with global anycast.

Recommendation: Based on our previous findings, we give some recommendations about how to deploy a regional anycast:

- Instead of basing regional partition solely on geographical location, it is more effective to partition regions based on performance factors such as network latency. While geographical location can provide a rough approximation of network performance, it does not always accurately reflect the actual network conditions.

- To extend their connectivity, CDNs should prioritize peering over transit with their non-tier-1 neighbors. Because the announcements from peers are not propagated to providers, cases where routers prefer routes from customers but remote sites will no longer occur.

3.7 Potentials

The measurement study in previous sections shows that DNS mapping inefficiencies can impact regional anycast’s performance. In this section, using the Tangled testbed, we explore how much performance improvement regional anycast can achieve over global anycast if we improve its DNS mapping strategies

3.7.1 Latency-based Regional Partition

We develop a region partition and client mapping strategy, referred to as ReOpt, to address DNS mapping inefficiencies. First, we partition the Tangled testbed into geographic regions using the K-Means algorithm [Mac67]. This step partitions geographically-close sites into regions. Second, we measure the unicast latency from each probe to each Tangled site and assign the probe to the region where its lowest unicast latency site is. Finally, we generate the country-level client-to-region mapping. For each country, we map all probes in the same country to the region where the majority of the country’s probes are assigned to. This final step enables a network operator to use country-level IP geolocation to map a client to a regional IP.

To obtain the optimal number of regions, we vary the number of regions from three to six and calculate the average client latency under each regional partition. We find that a 5-region partition has the lowest average client latency on Tangled. Figure 3.10 shows the site partition and the client mapping of ReOpt’s regional anycast configuration.

We observe two main differences between ReOpt’s regional partitions and the regional partitions used by Edgio and Imperva. First, there is a separate region in Africa in ReOpt, while that region is part of the EMEA region in both Edgio and Imperva’s regional partitions. Second, some probes in Central America are separated from probes in South America and are mapped to the North America region, but they are grouped together with the probes in South America in Edgio-4 and Imperva-6.

3.7.2 Global vs. Regional on Tangled

Next, we configure the Tangled testbed to deploy both global IP anycast and regional IP anycast and study their performance differences. When deploying global IP anycast, we configure all 12 sites in the Tangled testbed to announce one IP prefix to all their peering ASes and measure the RTTs from RIPE Atlas. When deploying regional anycast, we configure the sites in each of the five regions, as determined by the ReOpt algorithm, to announce a distinct regional IP prefix to all their peering ASes.

We then conduct two regional anycast experiments. In the first experiment, we directly assign each probe a regional IP that contains its lowest-unicast-latency site and measure the RTT to each probe’s assigned regional IP. We use this step to study the optimal regional anycast performance without IP geo-location errors and without aggregating probes by country.

In the second experiment, we use a commercial DNS provider, Amazon Route 53 [Ama], to configure a country-level client-to-regional-IP mapping. Route 53 supports both country-level and continent-level DNS mappings. We create a test domain name and delegate it to Route 53 and use Route 53’s country-level mapping configuration tool to map clients from a country to a regional IP based on the mapping generated by the ReOpt algorithm. We then instruct the RIPE Atlas probes to ping the test domain name and measure their RTTs.

Figure 3.11 shows the CDFs of the probes’ RTTs when we directly assign a probe to a regional IP and the probes’ RTTs when we use Route 53. We can see that the performance of regional anycast is similar under these two configurations, while Route 53 mapping causes a slight performance degradation in the APAC and SA regions. This result suggests that performance-wise, it is acceptable to use the country-level DNS mapping service provided by a commercial DNS provider such as Route 53 for implementing regional anycast.

Then we compare the performance of regional anycast with that of global anycast on Tangled. Figure 3.12 compares the CDF of the probes’ RTTs under ReOpt’s regional anycast configuration with Route 53 with that under the global anycast configuration. We observe that in the regional IP anycast configuration, the client latency in all regions has improved compared to global anycast. For instance, ReOpt’s five-region configuration shortens the 90th percentile latency for the probe groups in NA from 232.6 ms to 88.6 ms. This result shows that with latency-based region partition and client mapping, regional anycast can outperform global anycast, even in the presence of DNS mapping inefficiencies.

We note that latency-based regional partition and client mapping requires that a CDN operator measures the client latency distribution to its regional IPs. This requirement increases the design complexity of regional anycast. However, managing client-to-regional-IP mapping is still simpler than managing client-to-site mapping, as done in DNS-based CDNs, because a regional IP covers multiple sites and an operator need not manage load-balancing and fault tolerance among those sites.

3.8 Related Work

There exists a large body of work measuring the IP-layer anycast adoption in the Internet [CAJ⁺15b] or characterizing the performance of global IP anycast systems, including root DNS servers [BF05a, BFR06a, SPT06, CRUR, LHF⁺07b, Kui,

LLSB18b, FHG13, WH17b, KLA⁺21, CR18] and global anycast CDNs [CFKB⁺15b, KLA⁺21, CR18].

The blog article [Mah15] discusses why LinkedIn adopted regional anycast for its private CDN and it reports that the client latency improved after the CDN switched from global anycast to regional anycast.

Differently, this work characterizes the deployments and study in-depth the performance of two global-scale public regional anycast CDNs. We experimentally validate regional anycast’s performance advantages and uncover its drawbacks. In addition, we use the Tangled testbed to explore how to improve the performance of regional anycast.

The performance challenges of global anycast systems motivated a few proposals to improve its performance [BFR06a, dOSHK17a, MUF19a, FMM⁺15b, LLSB18b, ZSZ⁺21, SAD20a]. Among the existing proposals, we consider regional anycast as the most promising approach as we discuss in § 3.2. Therefore, we aim to understand the deployments and performance of two real-world regional IP anycast CDNs in this work. We leave a comparison between regional anycast and other proposals as future work.

This work builds on McQuistin *et al.*’s work [MUF19a] of using the penultimate hops in traceroute outputs to infer the number of AS-level connections an anycast network has. Differently, in this work, we combine multiple sources, including rDNS, penultimate hops, IP-geolocation databases, RTT ranges, and RIPE Atlas probe locations to enumerate the CDN sites that announce an anycast IP prefix. iGreedy [CJR⁺15] identifies an IP anycast prefix and geo-locates the anycast instances using iterative latency measurements from known vantage points. We experimented with iGreedy for anycast site enumeration and found that it mapped fewer published CDN sites than the method we used in this work.

3.9 Conclusion

Regional anycast, combining IP anycast and DNS redirection, has been deployed in practice. Yet how well it performs and how a CDN deploys it remain unknown. In this paper, we perform a comprehensive study to characterize the deployments and performance of two real-world regional IP anycast CDNs. In particular, we explore the region division strategies of the CDNs and how they map clients to regions. We find that the CDNs divide their networks into regions by continents or large countries and similarly, the CDNs assign clients to regional IPs by the continents or the countries they reside in. Our measurements show that regional IP anycast in general can mitigate the catchment inefficiency problem experienced by global IP anycast, but poor DNS mapping, where DNS returns a regional IP originated from distant CDN sites to a client, could worsen client latency.

3.10 Appendix

3.10.1 *IP Geolocating p-hops*

We determine the catchment site of a RIPE Atlas probe by IP-geolocate the penultimate hop (p-hop) from its traceroute output to an IP anycast address. We then use the p-hop’s geo-location to locate a CDN site. If the CDN site has an on-site router, the p-hop is often the site router so that we can observe a co-located CDN site and infer that the CDN site announces the regional IP address we traceroute to. If a CDN site does not have a site router and connects to a remote IXP via a link-layer connection [ACF⁺12], we will not observe a CDN site co-located with a p-hop. In this case, we assume that the CDN site closest to the p-hop announces the regional IP we traceroute to.

Because IP-geolocation at the city-level is not reliable [PUK⁺11a, SZ11b, HFC11], we combine a number of sources, including IP-geolocation databases, reserve DNS

(rDNS) records, and RTT ranges to infer a p-hop’s location. We use three IP-geolocation databases: MaxMind [Max], ipinfo [IPI], and EdgeScape [Tec] to provide the possible locations of the p-hop. And we describe this process as follows.

Reverse DNS (rDNS): We first infer the location of a p-hop from its rDNS name [LHM⁺21, LHC19]. If a p-hop has an rDNS name and the name contains a geo-hint at the city level (*e.g.*, operator-defined codes, IATA/ICAO codes [(IA)], or CLLI code [ico]), we use the geo-hint to map the p-hop to a PoP location published by Edgio or Imperva. For instance, an rDNS name `ae-65.core1.amb.edgecastcdn.net.` indicates a p-hop is located in Amsterdam, Netherlands.

If the rDNS name does not contain any valid geo-hints at the city level, we will check the domain name’s country code Top-Level Domain (ccTLD). If the CDN provider (Edgio or Imperva) has only one anycast site deployed in the ccTLD’s country, we will map the p-hop to that site’s city-level location.

RTT range: If a p-hop does not have an rDNS name or we could not infer its location from the rDNS name, we then attempt to use the location of the RIPE Atlas probe that went through the p-hop [GSH⁺17] and has less than 1.5 ms RTT to the p-hop to infer its location. The threshold is chosen according to the typical size of a metropolitan area, as the speed-of-light latency in fiber is roughly 100 km per 1 ms RTT. First, we query three IP-geo databases to estimate the p-hop’s location, which may return different results. Then, we filter the invalid results based on the speed-of-light distance between the p-hop and the RIPE Atlas probe, and use the valid location closest to the RIPE Atlas probe as the p-hop’s location. By this method, the location of the p-hop can be resolved when both the position of the probe and the IP-geo databases are correct at the same time.

Country-level IPGeo: The two steps above could resolve the majority of Imperva and Edgio’s p-hops’ locations. If after the above two steps, the location of the p-hop is still unresolved, we will use the country information from the IP-geolocation databases to infer the location of the p-hop. If all IP-geolocation databases return the same country location for the p-hop, and the CDN provider only lists one site in the country, we will use that listed site location as the p-hop’s location, this is an effective way to discover the single site in one country.

Unresolved: We find that for the three sets of hostnames (Edgio-3, Edgio-4, and Imperva-6), we cannot resolve the locations of the p-hops of 2.3% to 9.9% valid traces. Increasing the RTT threshold will lead to an inaccurate inference. For those unresolved p-hops, even if we use IP geo-location databases to approximate their locations, we do not find more CDN sites. Therefore, we leave those p-hops unmapped.

3.10.2 Collection of CDN’s Technical documents

Existing work [HZWS18, Petb] has revealed that Edgio and Imperva are the two CDNs deploying regional anycast. To validate this observation and obtain a more recent view, we manually collect and examine the official technical documents to identify the studied CDNs’ redirection mechanisms. As illustrated in Section 3.4.1, we obtain a list of the top 15 CDN providers by retrieving CDNFinder’s API [Peta]. We note that we here focus on the CDN platform of each provider while global anycast could be available with other components to support specific services, *e.g.*, DNS or Cloud hosting. We also eliminate two other CDNs from our top list: (1) Facebook CDN, because it is deployed as a private CDN that supports Meta/Facebook’s services; (2) Automatic CDN, which works as a plugin to accelerate the static assets of sites that are tied to Wordpress.com.

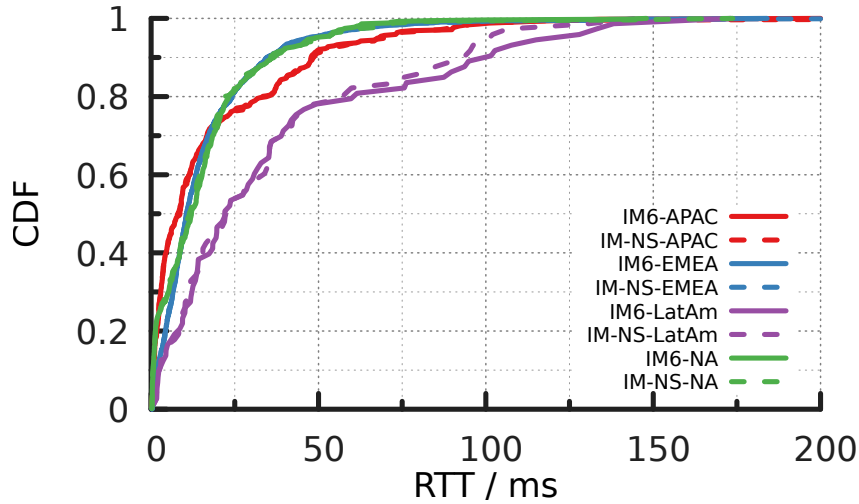


FIGURE 3.15: CDFs of the RTTs of the probes that reach the same CDN site via a regional IP anycast address and a global IP anycast address in Imperva-6 and Imperva-NS, after excluding the non-overlapping sites and peering ASes of the two networks.

3.10.3 Same-Site Latency Comparison

When we study the performance of Imperva-6, we compare it with Imperva’s DNS global anycast network (Imperva-NS) after excluding the non-overlapping sites and peering ASes of the two networks. We assume that when Imperva announces a regional IP anycast prefix and a DNS global IP anycast prefix at the same site to the same set of peers, it does not apply different latency-impact policies to these prefixes. To validate this assumption, we compare the RTT of a RIPE Atlas probe that reaches the same site via a regional IP anycast address with the RTT of the probe that reaches the site via a global IP anycast address. We include only the results from the probes that reach the CDN sites via the common set of peering ASes observed in Imperva-6 and Imperva-NS.

Figure 3.15 shows the result. As can be seen, the differences in the RTT distributions are negligible, which indirectly validate that Imperva does not apply different latency-impacting policies to its regional IP anycast prefixes and its DNS global IP

anycast prefixes. Because if it does, we should observe significantly different RTT distributions.

Anycast Flipping Detection

4.1 Introduction

IP anycast [PMM93, Met02] is a routing paradigm that splits traffic among a set of physical sites. Operators advertise the same IP prefix – via the Border Gateway Protocol (BGP) [EE06] – from each site and traffic from clients to the prefix is routed to any one of the sites, distributing the traffic load across the sites. Further, IP anycast presents the opportunity to improve performance by routing clients to a proximal site [CFKB⁺15a, dVdOSH⁺17a, ZSZ⁺21]. For these reasons, IP anycast is widely adopted by large-scale network services, such as the Domain Name System (DNS) [LHF⁺07a, SBK⁺20a], content delivery networks (CDNs) [dVARD20] and DDoS mitigation services [MSH⁺16a, Tec20]. Thus, the efficient operation and performance of anycast networks is critical to today’s Internet.

Clients of anycast services are commonly organized according to which sites they reach, and the set of clients reaching the same site is known as that site’s “catchment”. It is possible for a client to switch catchments. In the past, operators have raised concerns that such anycast “instability” could interrupt connection

flows [LLU06]. Because connection-oriented protocols typically store state at both ends of the connection, a change in site during a flow would break that flow. The existence of major services relying on anycast, however, belies the prevalence of broken flows. Indeed, previous work [WH17a] demonstrates that changes in site occur rarely in measurements from RIPE Atlas [Sta15a] probes to the DNS root nameservers.

The same study, however, observes that 1% of the vantage points measured did “flip” frequently between catchments, sometimes changing with *each* measurement. In follow-up work [WH18a], the authors show that flipping within TCP flows is rarer than flipping overall (impacting 0.15% of vantage points), and provide a plausible explanation that flipping occurs per-flow, rather than per-packet, so that all packets within a single flow reach the same site, preserving the connection. So, while flipping does occur, it is not observed to interrupt flows.

In our own measurements of anycast services, we observed high variability of round-trip-time (RTT). Intrigued, we set out to discover the source of the variability and found ourselves rereading the topic of flipping. While flipping is rare, using the same methodology as in [WH17a] we found that for RIPE Atlas probes querying root name servers, flipping has more than quadrupled from 1% in 2016 to 4.4% in 2023. In these experiments, a probe issues a query every four minutes, and a flip is said to occur if two consecutive queries by the same probe are routed to different root name server sites. A probe is said to be “flipping” if it averages at least one flip every ten minutes.

Investigating the cause of the increase through a longitudinal study, we find that increasing numbers of anycast sites – adding to the possibilities for flipping – contribute to the prevalence of flipping in 2023. With these changing network realities, we argue that anycast flipping is a more significant issue now, warranting further study of its impact on web performance.

Next, by downloading a mock web page with many embedded objects from a

major anycast CDN through the BrightData residential proxy service [bri23a], we confirm the finding that flipping rarely breaks connections and appears to occur per-flow. Furthermore, we find that the incidence of flipping from the BrightData vantage points to the CDN sites is perhaps even higher than from the RIPE Atlas probes to the root name servers. Because the CDN experiment differs from the DNS experiment (e.g., we didn't download the page every four minutes), we cannot adopt the same definition of flipping for both experiments. Instead we say that a vantage point is flipping if more than 10% of TCP connections are sent to a site other than the most common CDN site for that vantage point. In the CDN experiment we found that 2318 of 9965 (23.3%) BrightData vantage points were flipping.

The CDN experiments also revealed that the round-trip time from a vantage point to the sites its requests are sent to can vary widely. Hence, we set out to measure the impact of flipping and the corresponding variations in latency on the time to download and render popular web sites.

In summary, this paper makes the following contributions:

1. We show that when querying DNS root name servers from RIPE Atlas probes, the prevalence of anycast flipping has increased from only 1% of probes in 2016 to 4.4% in 2023. We confirm the previous finding that flipping rarely breaks connections, and that flipping commonly occurs at the granularity of an entire flow. Further, we find that flipping is also common when issuing requests from the BrightData residential proxy network [bri23a] to a major anycast CDN, with, according to a different measure (10% of TCP connections flip), 23.3% of these proxies flipping.
2. We show that flipping can significantly increase the RTT of client to a CDN site. For example, 20% of the flipping vantage points have a median increase in RTT larger than 32.5 ms, and 10% have a median RTT increase larger than

113.5 ms.

3. We demonstrate that for clients that experience very frequent flipping and large latency penalties, the impact on web download performance can be significant. For example, we performed trace-driven emulations of 20 popular web sites with MAHIMAHI [NSD⁺15], extending it so that we could randomly apply a flipping probability of 50% to entire TCP flows and a flipping latency penalty of 50ms. In this experiment, there was a median increase of more than 20.7% - 52.6% in the First Contentful Paint metric for HTTP/1.1 browsers and 18.3% - 46.6% for HTTP/2 browsers.

The current prevalence of anycast flipping, which is already significant and appears to be increasing, combined with the high latency penalties experienced by some clients when flipping occurs, calls for careful optimization of anycast deployments and perhaps even improved routing protocol design.

Ethical considerations Active measurements such as issuing DNS queries and downloading web pages cause load on the Internet infrastructure. As discussed later in the paper, we mitigated the impact of our experiments by issuing measurements at low rates or a small number of times. Our analysis of the frequency of DNS-query flipping is based on publicly available RIPE Atlas data, and did not require us to initiate any measurements. Our measurements of the anycast CDN were directed at a mock web site that was hosted using a free service provided by the CDN. This work raises no other ethical issues.

Data and code Upon publication, the authors will make all data collected in this research and all code used to collect or process that data publicly available. The authors will also provide such data and code to the program committee upon request.

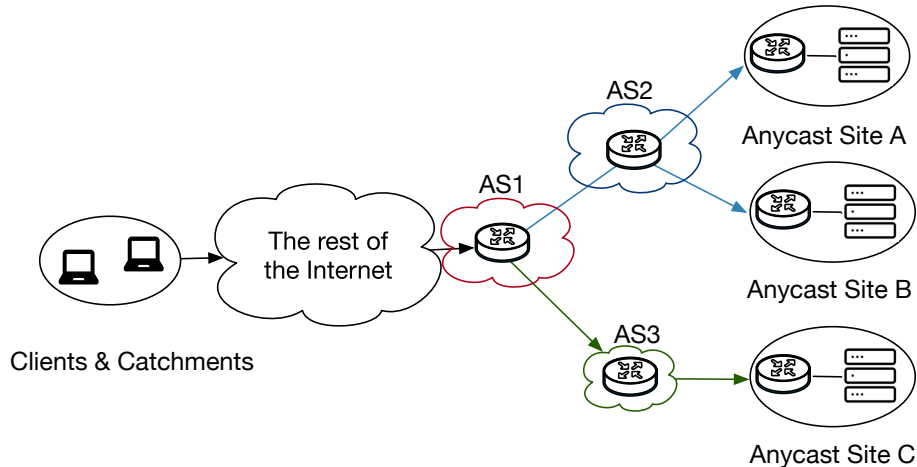


FIGURE 4.1: An IP anycast deployment.

4.2 Background

In this section, we use network traces to illustrate several examples of anycast flipping in the Internet.

Figure 4.1 shows a simple anycast network with three sites, each one serviced by a single upstream autonomous system (AS). Routers at each of the sites peer with routers in the upstream AS and advertise a route to the same IP prefix. Upon receiving the advertisements, the upstream ASes propagate them further to other ASes. If a router receives multiple routes to the IP prefix (e.g., the router in AS1), it applies BGP best path selection [EE06] to choose a route to reach that prefix. Best path selection is a cascading comparison between metrics of the available routes, including – among others – the AS path length. Traffic from clients of the service to the IP prefix is then routed to different sites via the emergent properties of the different path selection choices made by routers in the Internet. The set of clients all reaching the same site is the *catchment* of the site.

Over time, catchments can change for a variety of reasons. One of the most straightforward is route updates. Upon each route advertisement/withdrawal, routers recompute the best path, which may be different from the previous one and originate

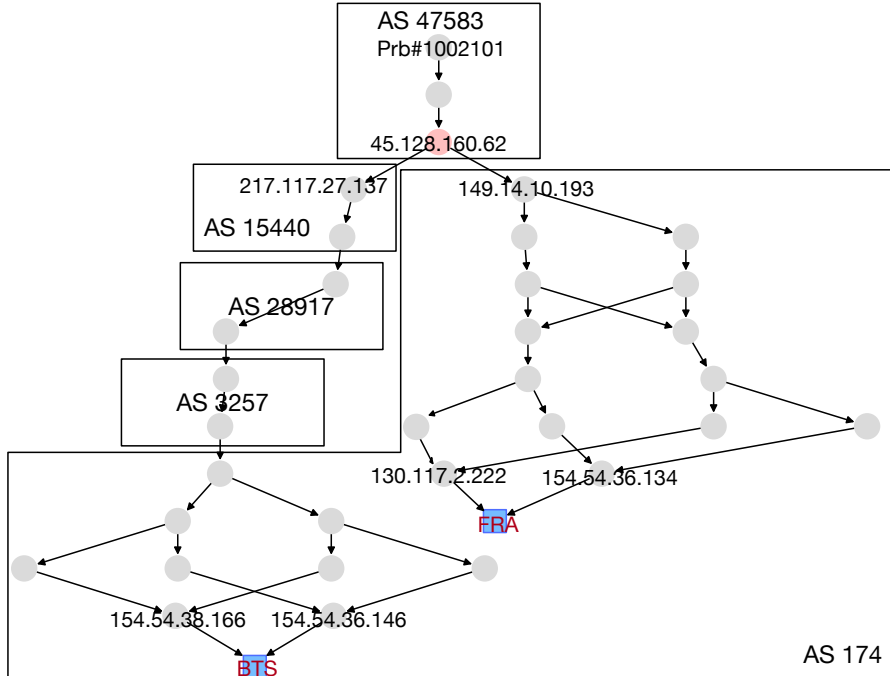


FIGURE 4.2: Flipping at the border between two ASes

at a different site. Wei and Heidemann [WH17a] argue that updates are unlikely to cause frequent catchment changes due to BGP route flap damping (RFD) [VCG98], although a sophisticated analysis by Gray et al. [GMB⁺20] provides a lower bound of only 9.1 on the percentage of ASes that implement RFD. But flipping can have other causes as well, including IGP Equal-Cost Multipath (ECMP), and BGP Multipath.

In this section we illustrate two examples that were collected from RIPE Atlas probes that we observed flipping. More details are provided in Section 4.3.

In the first example, shown in Figure 4.2, flipping takes place at the border of two ASes. The figure shows multiple traceroutes from RIPE Atlas probe #1002101 to the C-root on Jan-02-2023. The traceroutes have been merged so that hops appearing in multiple traceroutes are represented by a single node in the graph. The probe flips between anycast sites in Frankfurt (FRA) and Bratislava (BTS). While the traceroutes produce numerous paths to the target anycast IP address, the third hop – the last in AS47583 – visually splits the paths among the two sites. Interestingly,

the hops immediately after AS47583 are from two different ASes.

We are not sure what mechanism is responsible for flipping in this example. Originally, BGP Multipath required (among other things) the AS path of an alternate route to be strictly identical to the primary route in both the length and the sequence of AS numbers. As [LT23] notes, however, router configuration settings are often provided to relax the sequence requirement, e.g., Cisco’s `multipath-relax` setting [cis18] and Juniper’s `multiple-as` setting [jun23]. But in this example both the length and the sequence are different.

Figure 4.3 shows an example in which the flipping is internal to an AS. Probe #2121 flips between anycast sites of C-root in Frankfurt (FRA), and Paris (PAR) on Jan-02-2023. AS174 advertises routes to AS2914 in at least 3 places. Hop 81.25.197.1 forwards flows further within AS2914 that ultimately traverse different egress points to reach the different anycast sites. We speculate that AS174 applies ECMP internally to load balance traffic within the network, contributing to the flipping we observe.

4.3 Prevalance & Impact: Root DNS

In this section, we use RIPE Atlas’s built-in measurements [RIP23] to root DNS servers to characterize the prevalence of anycast flipping and its impact on a probe’s RTT to a root DNS server. We first describe the measurement infrastructure and datasets we use. Then we present how we infer anycast flipping from the datasets. Finally, we present the trend and degree of anycast flipping from RIPE Atlas probes to root DNS servers between 2016 and 2023.

4.3.1 RIPE Atlas’s DNS CHAOS Queries

RIPE Atlas [Sta15a] is a global measurement infrastructure that has more than 11,000 active probes world-wide. It conducts several periodic measurements to all

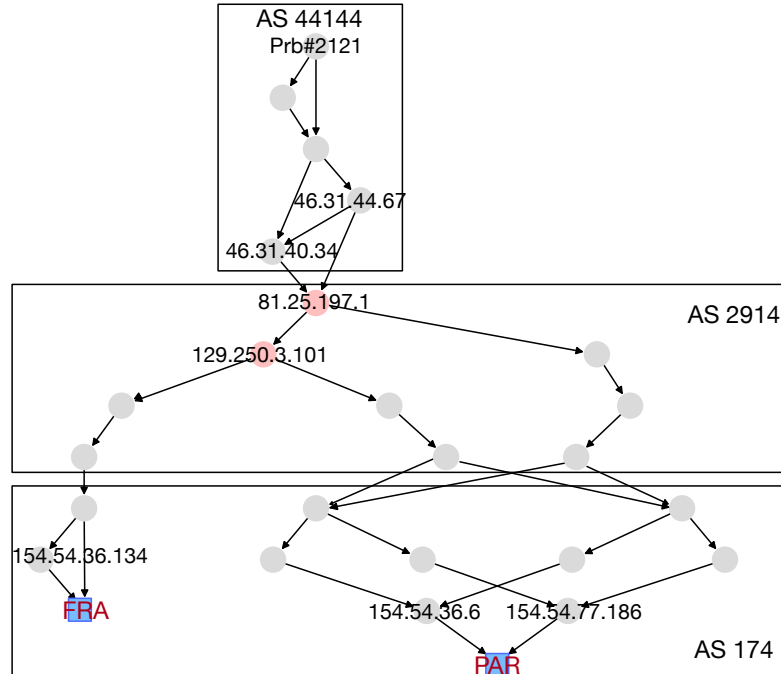


FIGURE 4.3: Flipping internal to an AS

13 root DNS servers and achieve the measurement results for researchers to study the properties of the Internet. Specifically, we use the DNS CHAOS [CW07a] measurement for this study. In this measurement, every probe sends a DNS CHAOS query for the TXT field of the domain name `hostname.bind` to every root DNS server every four minutes. Each probe records the response it receives. A response to a CHAOS query from a root server includes a site identifier (ID), which we identify the catchment site of the query.

Datasets To evaluate the trend of anycast flipping, we use the DNS CHAOS responses collected on the first Monday (24-hour) of every half-year from the last half year of 2016 to the first half year of 2023, a total of 182 datasets. Similar to [WH17a], we use all available probes at the measured time. For the first dataset collected on Jul, 04, 2016, there are 9,246 active probes; and for the most recent dataset collected on Jan, 01, 2023, the number of active probes increases to 11,568.

4.3.2 Root DNS Site Discovery

We parse the TXT record included in a server’s reponse to determine the catchment site of a probe’s CHAOS query and we use the change of the catchment site of a probe to determine anycast flipping (Section 4.3.4).

Each root DNS server uses a customized naming scheme to embed an anycast site ID in the TXT record of a CHAOS reponse. For example, a site ID in a response from A-root can be ‘nnn1-nlams-1a’ or ‘rootns-fra5’, each consisting of several sections. The first section (e.g., ‘nnn1’ or ‘rootns’) indicates which DNS software is running on the responding server. The second section encodes the location of the site. In the first example, ‘nlams’ is the UN/LOCODE [fE22] for Amsterdam, Netherlands, while in the second example, ‘fra’ is the IATA [(IA)] airport code for Frankfurt, Germany. The third section in the first example indicates the specific server/replica ID at the Amsterdam site. Similarly, the number ‘5’ in the second section of the second example specifies the ID of the server located at the ‘fra’ site.

We manually examine the CHAOS responses from each root server and construct regular expressions to extract the site ID field from those responses. In addition, the site naming scheme of each root has also changed over years. For example, L-root’s scheme changed three times in the last eight years. Therefore, we use different regular expressions per root and per time period to extract the site ID from the TXT field of each CHAOS response. We note that some CHAOS replies do not follow any naming convention and appear to come from record injection attacks [RLP⁺21]. An example is a TXT record that includes the string `byaazbknliphsiiy.vla.yip-c.yandex.net`. The regular expressions we construct automatically filter such responses. For the datasets we use, we filter 2.0% – 2.2% of such responses. We include the regular expressions we construct in Appendix 4.8.1.

4.3.3 *Geo-Locate a Catchment Site*

In addition to identifying a catchment site, we also aim to geo-locate where the site is at the city level. This geo-location information enables us to measure the distance between two catchment sites a probe flips between. Since most root servers include geo hints such as UN/LOCODE or IATA codes in their site IDs, we first extract those geo-codes from the site IDs we obtain in the previous step. Second, we use a geo-location method to confirm that 1) the geo-code included in a site's identifier is accurate, and 2) resolve the locations of the sites that either contain erroneous geo-codes or do not contain valid geo hints. The second step is important because a site ID may include a wrong code. For example, a site used by A-root has 'tko' in its site identifier, which is the IATA code for the Tlokoeng airport in South Africa. However, using the geo-location method, we find that the site is actually located in Tokyo.

Our geo-location method works as follows. First, we pick the top three probes that have the lowest RTTs to a site. If the RTT of each probe to the site is less than 5 ms, and the geo-code in the site's identifier specifies a location that is within a radius of 500 km of one of the probes, we consider the site's geo-code accurate and use the geo-code as the site's location. For the remaining site IDs, we check whether the site identifiers contain any geo-hint that matches one of the three closest probes' locations. For example, a site of A-root contains the string 'elpek' in its ID. Although it complies with the format of a UN/LOCODE, it is not a valid UN/LOCODE. But the sub-string, 'pek' is the IATA airport code for Beijing, China, and the three probes within 5 ms of this site confirm that the site is in Beijing. Finally, for the remaining unresolved sites, we use IP-geo information [IPI] of the penultimate hop in a traceroute output from each of the three-closest probes to locate the site. For example, a site ID 's1.org' from I-root contains no valid geo-hint and is resolved in

this final step.

We use the latest dataset obtained on Jan 02, 2023 as an example to show how many sites we are able to resolve at each step. Other datasets have similar results. For the latest dataset, we extract a total of 1,014 site IDs. Among them, we confirm 808 of them as valid geo-codes in the first step. We are able to resolve 198 sites' locations in the second step. The final step (traceroute) resolves the remaining eight sites.

4.3.4 Estimate Anycast Flipping

After we obtain the site ID and the geo-location of each catchment site a probe reaches, we are able to quantify anycast flipping. We use the same any flipping detection and counting mechanism as in [WH17a]. For each probe and for each dataset, we construct a time-series vector that consists of the catchment sites a probe reaches. If the catchment sites in two consecutive CHAOS responses are different, we count it as a flip. We then calculate the mean time between two flips across the whole day for each probe. If the mean flipping time is less than 10 minutes, then we consider this probe experiences anycast flipping. We compare the flipping results we obtain using the same dataset as used in [WH17a] (Aug-01-2016) and confirm the results are almost identical. For example, on that day, C-root had a flipping fraction of 1.2% in our measurement, and it is the same value in [WH17a], similar to the other roots. Therefore, we conclude that we are measuring the same phenomenon as in [WH17a].

4.3.5 Longitudinal View of Anycast Flipping

Now we describe our reproduction of the results from the 2016 study [WH17a] and our longitudinal study of anycast flipping since then. Longitudinally, we find that average percentage of RIPE Atlas probes that experience anycast flipping has increased

Table 4.1: The percentage of RIPE Atlas probes that experience anycast flipping over the past seven years.

Flipping Ratio	A	B	C	D	E	F	G
2016-07-04	0.02%	0.00%	1.38%	3.67%	3.75%	0.60%	0.46%
2017-01-02	1.08%	0.00%	1.07%	2.11%	1.74%	0.53%	0.57%
2017-07-03	0.84%	0.06%	1.00%	2.02%	1.74%	0.42%	0.19%
2018-01-01	2.39%	0.05%	1.05%	2.11%	1.42%	0.77%	0.16%
2018-07-02	0.63%	0.04%	0.83%	2.73%	2.03%	1.06%	1.19%
2019-01-07	1.87%	0.37%	0.63%	3.69%	1.08%	1.32%	1.18%
2019-07-01	1.10%	0.62%	0.42%	3.39%	0.91%	1.06%	0.03%
2020-01-06	2.27%	0.53%	0.75%	3.72%	1.68%	1.98%	1.52%
2020-07-06	2.07%	0.32%	0.44%	4.16%	1.93%	1.43%	1.15%
2021-01-04	9.83%	0.50%	0.64%	3.84%	1.78%	2.55%	0.06%
2021-07-05	10.88%	0.64%	0.64%	3.94%	1.90%	2.19%	1.18%
2022-01-03	9.95%	1.32%	0.68%	4.19%	1.92%	1.46%	0.07%
2022-07-04	11.72%	0.91%	0.88%	5.39%	2.80%	2.23%	1.15%
2023-01-02	11.87%	1.55%	1.47%	5.47%	2.96%	2.38%	1.59%

from about 1.2% to 4.4% within the past seven years. Figure 4.4 summarizes our findings. Nine of the 13 roots have a significant increase in anycast flipping during our measurement period. The most noticeable changes occur to A-root and J-root. Seven years ago, anycast flipping for both roots was minimal, but today, A-root incurs the largest amount of anycast flipping among RIPE Atlas probes, from 0.0% (only very few probes flipped at that time) to 11.87%. J-root sees a similar increase, although not as pronounced. Interestingly, both had a significant step between the second half year of 2020 and the first half year of 2021. We investigate A-root to better understand the causes.

A-root anycast flipping drastically increased from 2.07% to 9.83% between 2020 and 2021. On Jul-06-2020, RIPE Atlas probes received CHAOS TXT records that mapped to 21 sites, but half a year later the records mapped to two additional sites, ‘mnz’ and ‘wie’. Further, 88.9% of the probes that started flipping in 2021 but were not in 2020 reached at least one of the two new sites. We suspect that A-root deployed the two new sites at this time and the probes that flip between these two

Table 4.2: The percentage of RIPE Atlas probes that experience anycast flipping over the past seven years (continue).

Flipping Ratio	H	I	J	K	L	M
2016-07-04	0.00%	1.18%	1.20%	1.29%	1.15%	0.47%
2017-01-02	0.00%	1.07%	0.99%	0.94%	1.22%	1.04%
2017-07-03	0.00%	1.09%	0.51%	0.53%	0.97%	0.34%
2018-01-01	0.00%	1.30%	0.62%	0.61%	2.16%	0.20%
2018-07-02	0.00%	2.15%	1.31%	0.98%	2.23%	1.01%
2019-01-07	0.00%	2.51%	0.87%	1.45%	2.94%	1.07%
2019-07-01	0.05%	2.51%	1.09%	0.99%	1.66%	0.71%
2020-01-06	0.47%	3.56%	1.03%	1.43%	2.40%	0.57%
2020-07-06	0.09%	3.05%	1.25%	2.48%	1.21%	0.68%
2021-01-04	0.53%	2.98%	6.57%	3.37%	2.40%	1.99%
2021-07-05	0.44%	3.21%	8.73%	2.18%	2.87%	1.31%
2022-01-03	1.63%	3.00%	6.29%	2.30%	2.99%	1.46%
2022-07-04	3.04%	3.80%	5.94%	2.45%	4.57%	1.87%
2023-01-02	9.26%	3.18%	7.16%	2.53%	5.17%	2.76%

sites contribute to the increase in anycast flipping.

4.3.6 Does Flipping Break TCP Connections?

The prevalence of anycast CDNs suggests that anycast flipping does not tear down TCP connections, and the anycast flipping study [WH17a] in 2016 found that very few RIPE Atlas probes consistently failed to build a TCP session with the J root name server and a small number of those probes also suffer from UDP anycast flipping.

We use the RIPE Atlas built-in measurement that sends TCP DNS queries to obtain the SOA field of route DNS servers to further confirm this phenomenon. Using a dataset obtained on Jan-02-2023, we find that 97.4% - 98.0% of the probes can successfully establish TCP connections with a root DNS server. Only 220 probes cannot establish TCP connections to any root DNS server and among those probes, only 0-8 of them experience anycast flipping to some root DNS servers. The very few overlapping between probes that cannot establish a TCP connection and experience anycast flipping suggest that anycast flipping is unlikely to break TCP connections

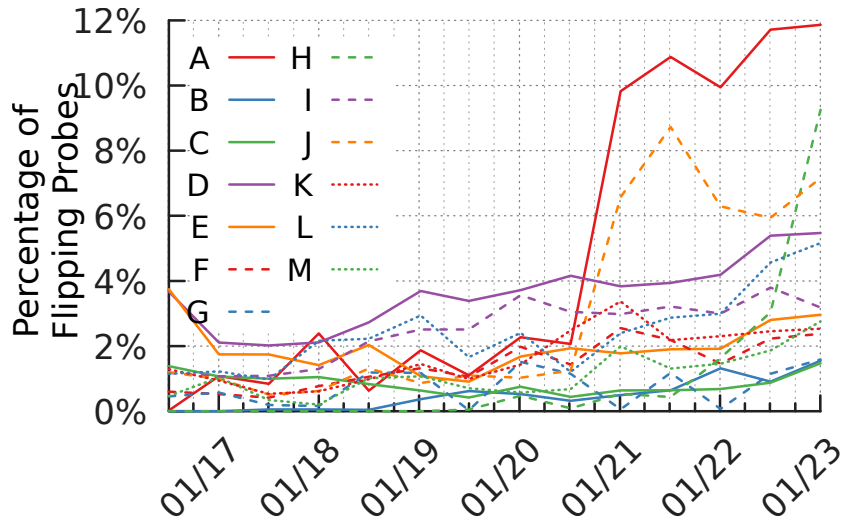


FIGURE 4.4: The percentage of RIPE Atlas probes that experience anycast flipping in the past seven years.

for the majority of Internet paths.

4.3.7 Impact on RTTs

For each of the RIPE Atlas probes (154 - 1243 for different roots) that we observed flipping on Jan-02-2023 to one of the root DNS servers, we compute the median RTT towards each of the anycast sites that they flip between. We then take the difference in the RTT to determine how much extra RTT flipping adds. Note that probes can flip between more than 2 sites – for A-root, 205 probes do. In that case, we take the difference between the minimum and maximum median RTTs to the sites, to determine the worst case extra RTT that flipping adds. Figure 4.8 shows the distribution per root of the extra RTT. In addition, we show the percentage of the probes whose worse case RTT exceeds 50 ms and that flip among three sites in table 4.3.

For B-root and G-root, the difference in RTT is extreme: nearly 80% of RIPE Atlas probes experience extra RTT of over 120ms/50ms to B-root/G-root, respectively. More probes flip reaching A-root than any other root, but the extra RTT in A-root



FIGURE 4.5: Longest Detour

is minimal: the extra RTT for 90% (1118 out of 1243 probes) of the probes less than 12ms, indicating that flipping in A-root occurs between nearby sites. For the other roots, the majority of probes experience modest extra RTT. Overall, though, 24 to 177 probes (0.2% - 1.7% of all probes) experience more than ≥ 50 ms extra RTT to at least one root.

We measure the distance between a probe and its catchment site. In Figure 4.8, we show the distribution of the extra distance corresponding to a probe’s flipping sites with which we compute the extra RTT. As we can see, the long extra RTTs correspond to long geographic distance between a probe’s catchment sites. In Figure 4.5, we show the example of the longest extra distance between a Singaporean probe’s flipping sites: one site in LAX and the other in SIN.

4.4 Prevalence & Impact: Anycast CDN

Next, we turn to our study of anycast flipping in a major anycast CDN. In this section, we use the BrightData to explore the prevalence of anycast flipping using different vantage points than RIPE Atlas probes and with a new target, a CDN that uses anycast.

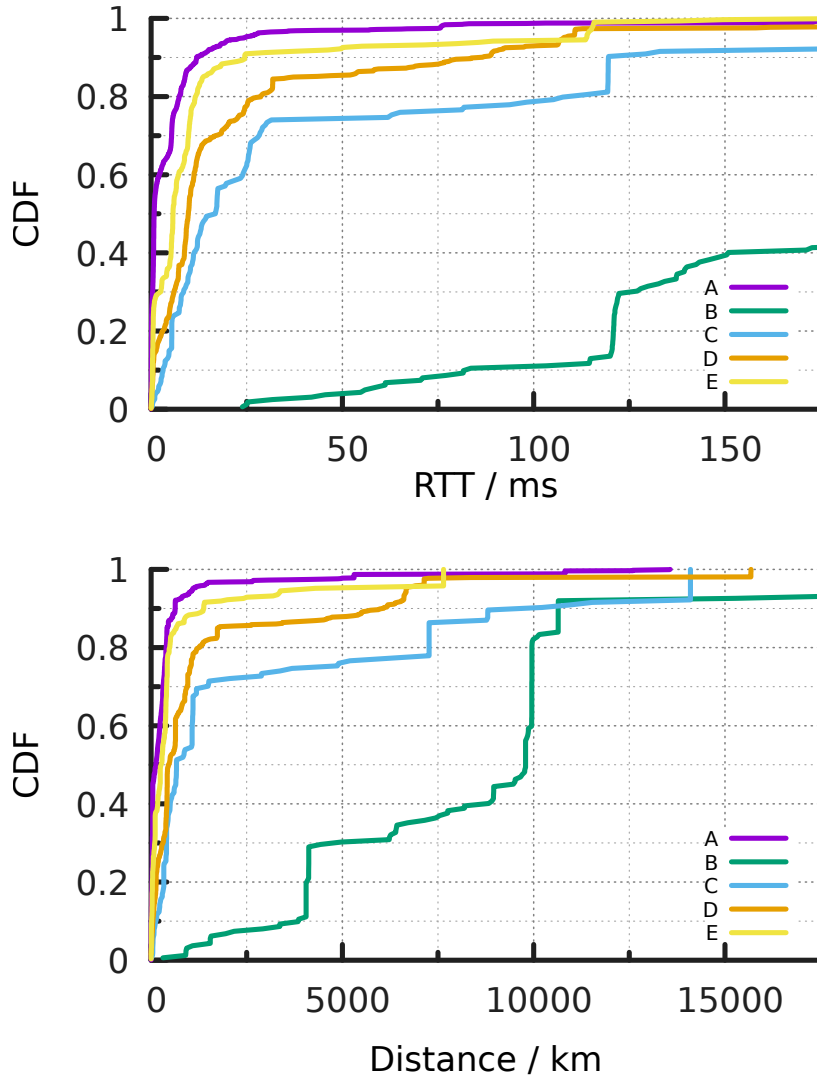


FIGURE 4.6: A,B,C,D,E Roots

4.4.1 Infrastructure

The RIPE-Atlas-based measurements can show us anycast flipping to the root name servers. To our knowledge, however, no existing measurements cover CDN servers. Therefore, we utilized a residential proxy service provider, BrightData [bri23a], to measure flipping of TCP connections to one of the largest anycast CDNs. This residential proxy service has previously been used in measurement studies, e.g. [CMK⁺21]. According to their website description, BrightData’s infrastructure con-

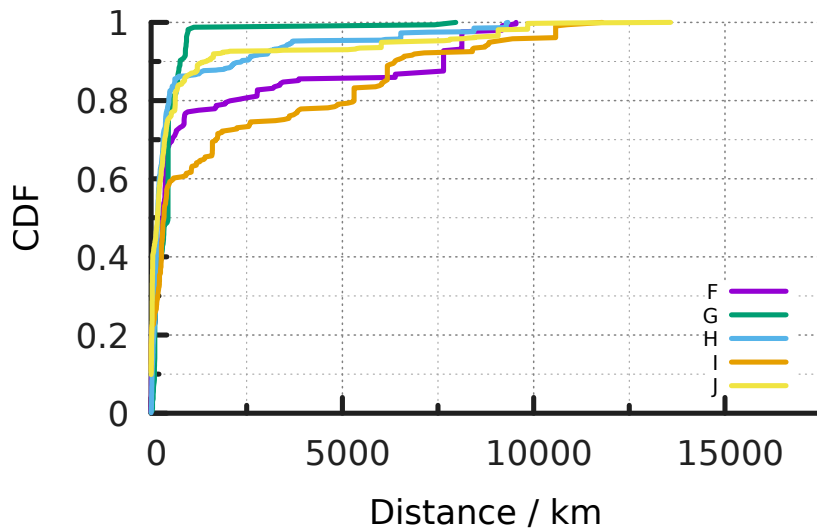
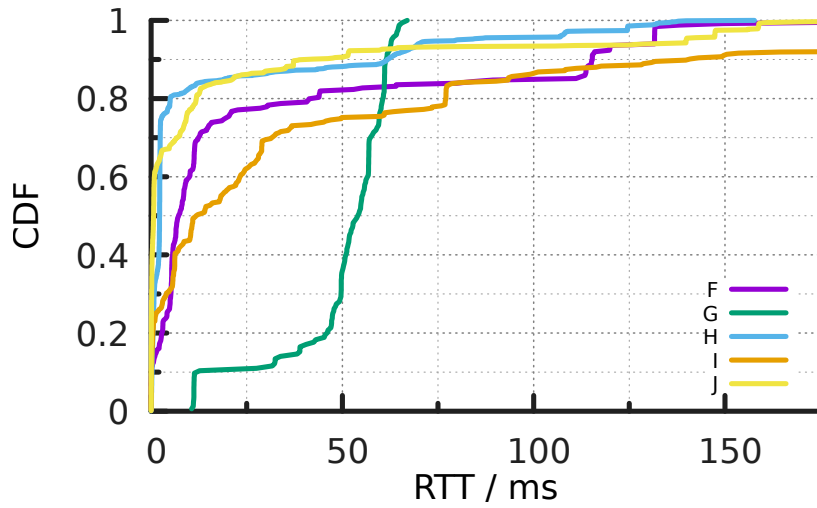


FIGURE 4.7: F,G,H,I,J Roots

sists of more than 72 million proxy nodes. In our experiments, we uncover around 10 thousand unique vantage points that span more than 150 different countries and 1600+ different ASes.

4.4.2 Methodology

To measure anycast flipping, we design a mock webpage, composed of 80 one-pixel images. To fetch the webpage, browsers must issue 81 total HTTP requests – the first being the root HTML object. Next, we configure this webpage and all embedded

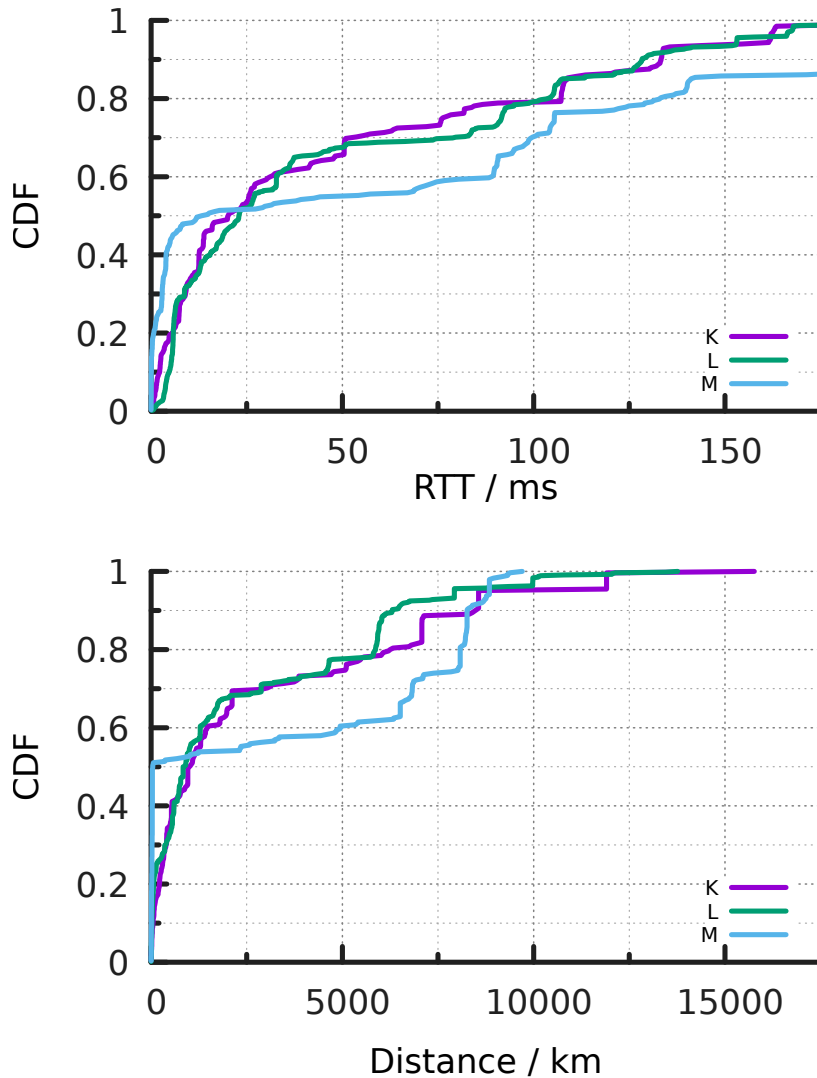


FIGURE 4.8: K,L,M Roots

images to be served by the CDN and make them cacheable so that they may be retrieved directly from the CDN's edge.

We instrument a headless Chrome browser to fetch the mock webpage through BrightData's residential proxy node pool. By default, BrightData may use a different proxy node per HTTP request. Since we wish to observe flipping per proxy node, this behavior is not desirable. Fortunately, we found a mechanism supported by BrightData that could enforce a consistent session via the same proxy node for the

Table 4.3: Anycast Flipping among Root-name Servers (*Long Flipping: Flipping that causes more than 50 ms round-trip-time difference. 3-Flipping: Flipping that reaches more than 2 different anycast sites*).

Root	Flipping Probe	Long Flipping	3-Flipping
A	1243	38 (3.06%)	205 (16.49%)
B	162	156 (96.30%)	2 (1.23%)
C	154	40 (25.97%)	0 (0.00%)
D	573	84 (14.66%)	121 (21.12%)
E	309	24 (7.77%)	40 (12.94%)
F	249	45 (18.07%)	22 (8.84%)
G	164	106 (64.63%)	0 (0.00%)
H	965	114 (11.81%)	0 (0.00%)
I	334	83 (24.85%)	16 (4.79%)
J	746	71 (9.52%)	76 (10.19%)
K	265	92 (34.72%)	30 (11.32%)
L	542	177 (32.66%)	92 (16.97%)
M	288	130 (45.14%)	0 (0.00%)

current webpage fetch [Bri23b]. To restrict all requests for fetching the mock webpage to use the same proxy node, we generate a random consistency token and concatenate it with our username registered with BrightData when we send the requests to the proxy service. With this mechanism, BrightData will use the same proxy node to send all the resource requests sharing the same consistency token. Further, BrightData allows the user to specify a geographic area to restrict the proxy selection. If an area is not provided, BrightData selects a proxy globally. Since we wish to study the impact of anycast flipping from as many vantage points as possible, we do not restrict proxy selection.

We repeatedly fetch the mock webpage through BrightData, each time with a chance to discover a new proxy. Between each fetch, we clear the browser state to ensure that no objects are cached at the client side. To measure anycast flipping from BrightData proxy nodes, we leverage a unique trait of the proxy service. We observe that for each request, a proxy node establishes a new TCP connection with a

target hosting site. Thus, for the 81 objects comprising the mock webpage, the proxy node establishes 81 TCP connections to the CDN, providing us 81 opportunities to observe flipping. We confirmed the behavior of BrightData using our own server. We first hosted our mock page on our own server and hence a proxy node requests directly reached our server rather than the CDN. Then, based on the HTTP log generated by our HTTP server, we observed that 81 different ports from a proxy node were used during the request.

In addition to fetching the mock webpage, we also fetch another URL: `http://lumtest.com/myip.json`. The `lumtest.com` website provides a service that echos back the source IP address of an HTTP request in its HTTP response, which enables us to obtain the information about the current proxy node used in a proxy session, including the proxy id, proxy IP address, and geographical and network details about the proxy node. From this information, we identify the proxy node assigned by BrightData to fetch the mock webpage.

BrightData's HTTP response to an HTTP request from our headless browser includes a performance timing profile that records the timing information of the download of each object. We use this information to measure the RTT from a proxy node to each of its catchment sites. We describe more detail in § 4.4.4.

In total, we fetch the mock webpage 23,651 times using BrightData and discover 10,634 proxies from 1,640 different ASes. However, not all BrightData proxies provide timing information. So we exclude 669 proxies, leaving 9,965 proxies in our study.

To study anycast flipping, we also need to detect which site responds to each HTTP request. Fortunately, the anycast CDN adds a header field in its responses that contains an IATA code to indicate which site responds to an HTTP request. Since the CDN we use is a global anycast CDN, if we observe that multiple sites respond to the request that fetches our mock webpage, we consider that the proxy node experiences anycast flipping.

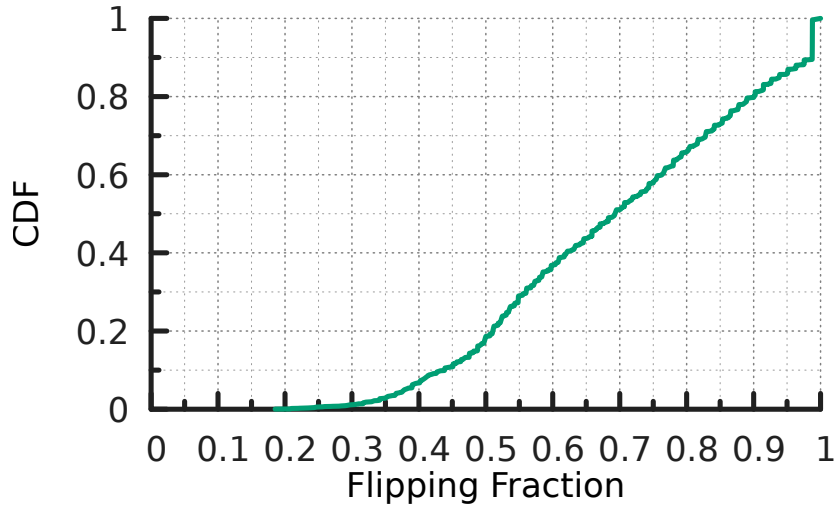


FIGURE 4.9: Fraction of requests reaching the mode site

4.4.3 Prevalence

We now present the results on how frequent we observe anycast flipping to the anycast CDN. Out of 9,965 proxy nodes, 2,907 (29%) of them reach multiple sites of the anycast CDN. Figure 4.9 shows a CDF of the fraction of the 81 requests that are responded to by the site that receives the largest number of requests (We define such a site as the “mode site”). A value close to 1 means that reaching a different site is very rare, which is not indicative of anycast flipping. So, we set a threshold ≥ 0.9 to focus on the proxy nodes that experience frequent anycast flipping. This threshold leaves out 589 proxies.

We show the CDF of the number of sites the remaining 2,318 proxies reach in Figure 4.10. As we can see, flipping between more than two sites is common, with 1,305 (56.3%) proxy nodes reaching three sites or more. In the extreme case, one proxy node flips between 11 different sites while downloading the 81 objects in our mock webpage.

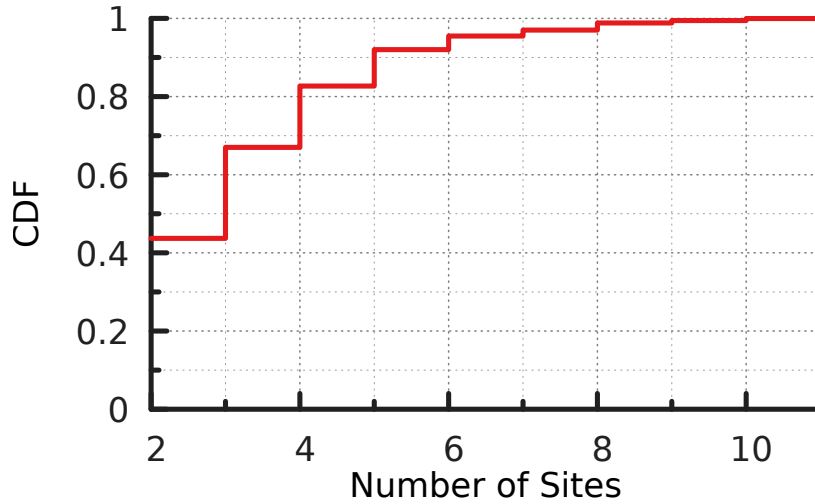


FIGURE 4.10: # sites reached per proxy node

4.4.4 Impact on RTTs

Next, we investigate the impact of anycast flipping on the RTT from a client to an anycast CDN site. From the timing profile included in a BrightData HTTP response, we extract the connect time – from when a proxy node sends a TCP SYN packet to the anycast CDN to when the proxy node receives the SYN+ACK reply. This metric estimates the RTT from the proxy node to its catchment site of the CDN, independent of the where we run the headless browser. A similar method is also used in [CMK⁺21].

We calculate the difference in RTT for each proxy node as follows. First, we calculate the median RTT, mRTT, per site. Next, we determine if the site that receives the largest number of requests, $site_x$, has the largest mRTT. If it does, then we find the minimum mRTT to any other site and subtract $site_x$ mRTT from it to determine the best case improvement in RTT that anycast flipping causes. If $site_x$ does not have the largest mRTT, then we find the maximum mRTT to any other site and subtract $site_x$ mRTT from it to determine the worst case additional RTT that anycast flipping causes. Figure 4.11 shows the CDF of these differences in RTT, the

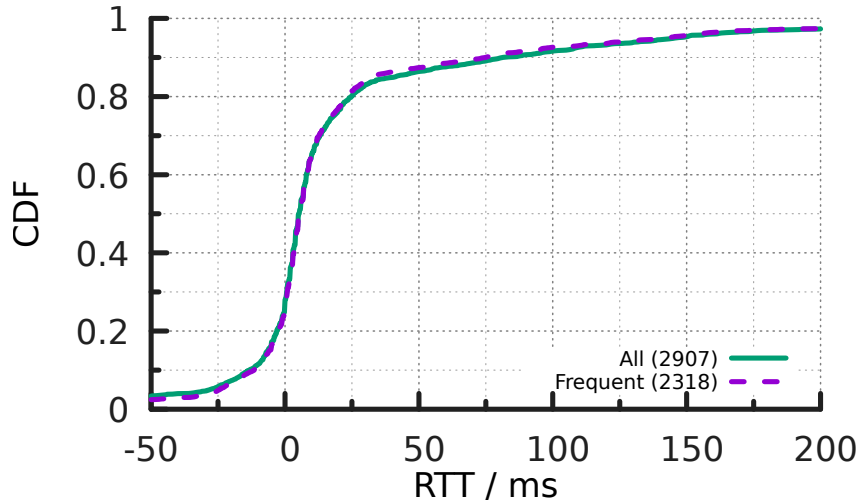


FIGURE 4.11: ΔRTT of proxy nodes that flip between sites

“Frequent” line includes the proxy nodes who reach their mode sites less than 90% of the time, while the “All” line includes all proxy nodes. . Anycast flipping reduces RTT for roughly 20% of proxies, although typically by small values. Similarly, other proxies see modest increases in median RTT due to flipping, and 20% of the proxies suffer from RTT differences that are larger than 23 ms, and around 10% suffer from differences larger than 75 ms. Combined with our observations on flipping rates, 132 (1.3% of all) proxies both flip more than 50% of the time and have an RTT difference ≥ 50 ms. In the next section, we investigate what implications the RTT increase has on web performance.

4.5 Impact on Web Performance

The previous section demonstrates the impact anycast flipping has on RTT between the proxies and the anycast CDN. We now turn to how the RTT increases due to flipping impact web performance.

4.5.1 Infrastructure

First, we must build an infrastructure to measure the impact. We extend the web emulator MAHIMAHI [NSD⁺15]. MAHIMAHI retrieves webpages, capturing all the resources – from different hosts – embedded in the webpage, and stores them locally. Then, it enables replaying of the webpage to a browser by serving the resources from Apache web servers, matching the resource and host association collected during the capture. Optionally, MAHIMAHI can also add a synthetic delay during the replay. Thus, MAHIMAHI can emulate the web performance of a page load under varying network conditions. To study anycast flipping, we make several extensions to MAHIMAHI.

Per-Host Delay While MAHIMAHI allows a configurable delay to be applied to fetching resources during the webpage replay, it applies the same delay to all fetches. This simplification is not faithful to the original webpage retrieval (many webpages typically include resources from diverse hosts), nor is it convenient for measuring the impact of anycast flipping. So, we add the ability to set different synthetic delays per Apache web server to MAHIMAHI.

Support for Anycast Flipping The previous extension enables setting a delay per host, but with anycast flipping the delay to a specific host may vary depending upon the site reached. So we further extend MAHIMAHI to probabilistically apply additional delay per TCP flow to emulate flows reaching either a near site (i.e., not additional delay) or a far site (i.e., additional delay). During the emulation step, we apply the probabilistic delay to TCP flows destined for hosts selected to flip (details below).

Support for HTTP/2 MAHIMAHI only supports HTTP/1.1 replay, yet HTTP/2 is frequently used today. Further, because anycast flipping operates on individual TCP

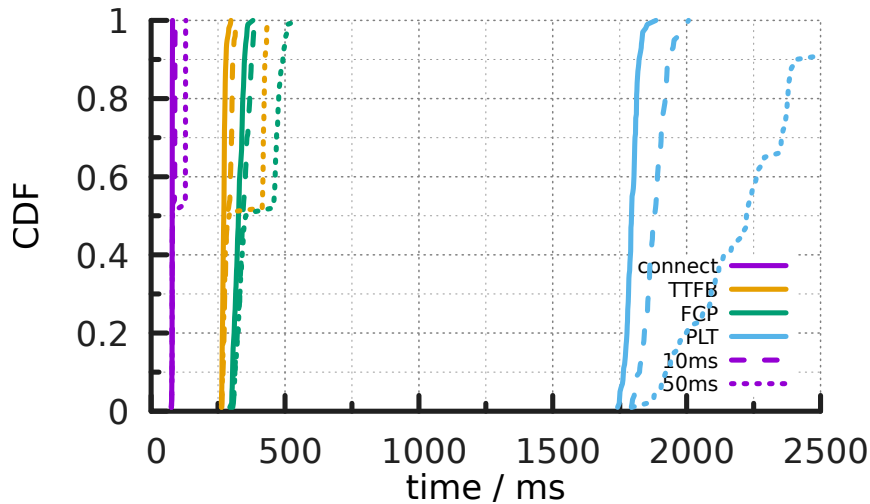


FIGURE 4.12: HTTP/1.1 metrics

flows and HTTP/2 handles multiplexing over the underlying transport differently than HTTP/1.1, we anticipate differences in the impact to each protocol. So, we update MAHIMAHI’s Apache web server to support HTTP/2 by replacing MPM prefork module (“mpm_prefork_module”), which does not support HTTP/2, with the MPM event module (“mpm_event_module”), and by adding HTTP/2 support with the “mod_http2” module.

Revised Retrieval Step Finally, we re-implement MAHIMAHI retrieval and share the code at [mit]. This was necessary for two reasons. First, MAHIMAHI’s retrieval does not support HTTP/2. Second, to collect the per-host delays used during emulation. For the latter part, we run ‘tshark’ to capture TCP packets on 443 port. After retrieval, we extract the RTT to each host from the packet capture and apply the RTTs to MAHIMAHI’s Apache web servers.

In emulation, our client is the headless Chrome browser in incognito mode, wrapped with Browsertime [bro23] to automate fetching webpages from MAHIMAHI and collect performance metrics.

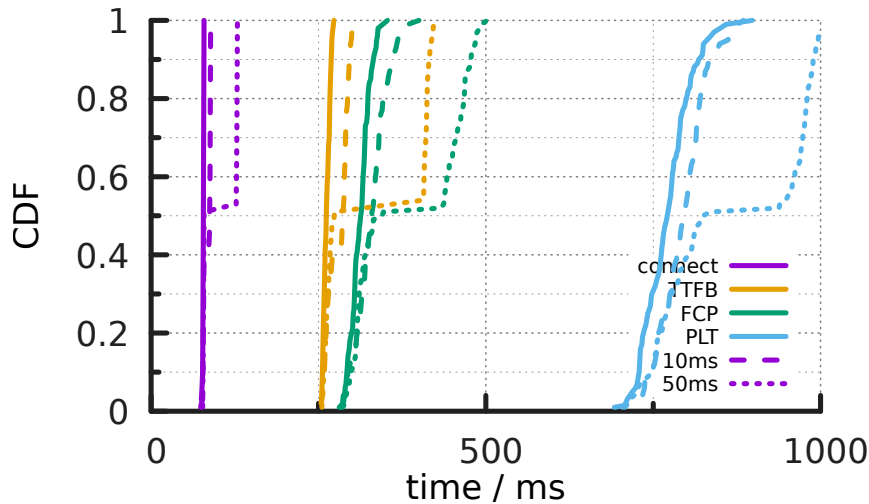


FIGURE 4.13: HTTP/2 metrics

4.5.2 Mock Webpage Results

To show the impact of anycast flipping, we emulate web browsing of our mock webpage using MAHIMAH. First, we retrieve the mock webpage from our vantage point at [Redacted for review] using our revised retrieval step. Because all of the one-pixel images in our mock webpage are served by the CDN, a single host serves the entire webpage. During retrieval, we calculate the RTT to the CDN from our vantage point and match the delay in emulation to the RTT. We then use our instrumented browser to fetch the mock webpage from MAHIMAH 100 times. Figure 4.13 in the solid lines shows the CDF across the 100 measurements of four key performance metrics: connect - the TCP connection time for the webpage root object, TTFB - time to first byte of the root object, FCP [fcp23] - first contentful paint is visible on the screen, and PLT - the start of the JavaScript load event. Smaller values in these metrics mean that page loads more quickly and the web performance is better. We show the results for both HTTP/1.1 (Figure 4.12) and HTTP/2 (Figure 4.13).

Using the solid lines as our baseline, we next investigate the impact of two anycast flipping scenarios by selecting the single host serving our entire mock webpage to flip.

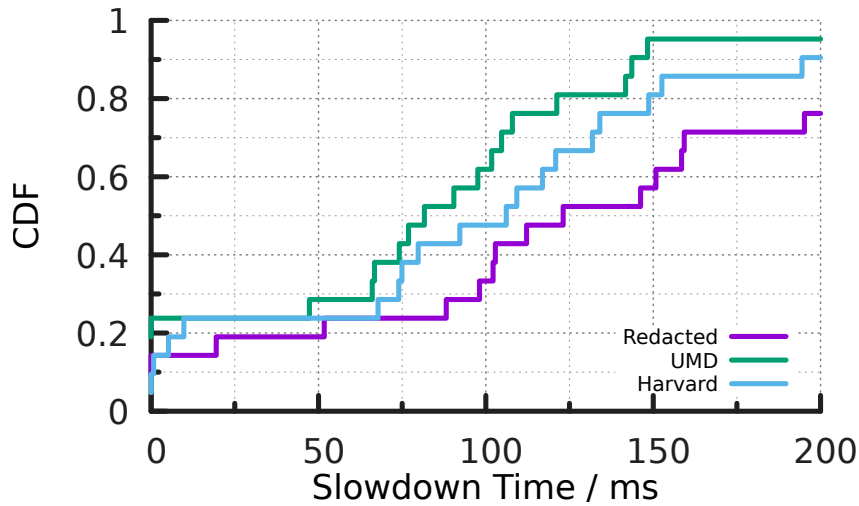


FIGURE 4.14: HTTP/1.1 Slowdown Time (FCP median)

To emulate anycast flipping observed in Section 4.4 that effects 1.3% of BrightData proxies, we set the probability of flipping to 50% and the additional RTT on a flip to 50ms — the dotted lines. We also run our emulations reducing the additional RTT to 10ms (which effects 2.5% of proxies) to see the impact of flipping when the RTT difference is less significant – the dashed lines. The metric values grow (become worse) as anycast flipping latency is added, regardless of protocol.

Consider the connect lines first. There is a noticeable step at roughly the middle of the distribution where flipping delay is added. Because flipping occurs per-flow and the flipping probability in our emulation is 50%, there is a 50% chance that the connection for the root object will reach the near site and the distributions in all three scenarios will be the same. However, if the connection for the root object is “unlucky”, then the connect time is increased by 10/50ms. This applies to both protocols.

However, unlike HTTP/1.1, Chrome’s implementation of HTTP/2 uses a single TCP connection per host and multiplexes all requests over the same connection. If the first connection for the root object is unlucky, then Chrome remains unlucky for

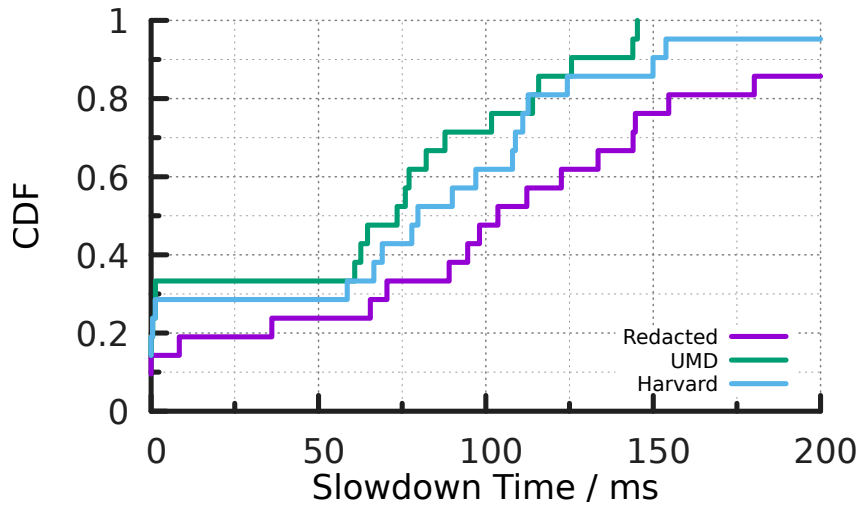


FIGURE 4.15: HTTP/2 Slowdown Time (FCP median)

the rest of the webpage fetch. This is evident in the steps visible in all other metrics. By comparison, Chrome’s HTTP/1.1 implementation uses 6 TCP connections total per host, giving it six opportunities to reach the near site. Moreover, we observe that Chrome dispatches HTTP requests on the first available connection, so the connections to the near site receive a disproportionate number of the 81 requests because each request completes faster than other requests on connections to the far site. As a result, HTTP/1.1 performance under anycast flipping degrades slower than HTTP/2. This is most visible in the PLT dotted lines where HTTP/2 performance becomes significantly worse when the single TCP connect flips to the far site, while HTTP/1.1 performance more gradually degrades in steps as each of the 6 TCP connections flips to the far site.

We note that HTTP/2 performance overall remains better than HTTP/1.1. Even though HTTP/1.1 establishes 6 TCP connections, the 80 HTTP requests for one-pixel images are still transmitted serially across the connections. Meanwhile, because HTTP/2 removes head-of-line blocking, the 80 HTTP requests occur in parallel..

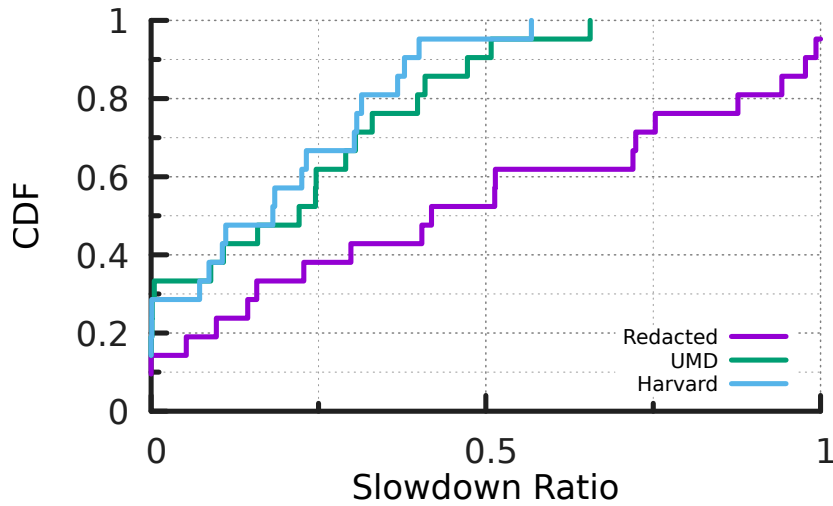


FIGURE 4.16: HTTP/2 Slowdown Ratio (FCP median)

4.5.3 Popular Website Results

The mock webpage provides us with a synthetic view of the impact of anycast flipping, but real webpages typically aren't composed of 80 one-pixel images. So, next we explore the impact of anycast flipping on real webpages. In this analysis, we use the landing webpages of the top 20 most popular websites (Table 4.8.1 in Appendix 4.8) from the Tranco list [PVG^T+19]. Also, in addition to our vantage point at [Redacted for review], we use two vantage points (Harvard University and the University of Wisconsin-Madison) provided by CloudLab [DRM⁺19] to retrieve the 20 webpages, giving us three distinct environments to measure the impact of anycast flipping.

As with the mock website, we apply a 50% flipping probability and add 50ms on flips to the far site. However, unlike the mock website, real webpages aren't typically served by a single host. Indeed, we observe that the 20 webpages we retrieve are served by a variety of hosts, including several CDNs. To study the impact of anycast flipping, we take the approach of assuming each of the CDNs uses anycast and their hosts to flip.

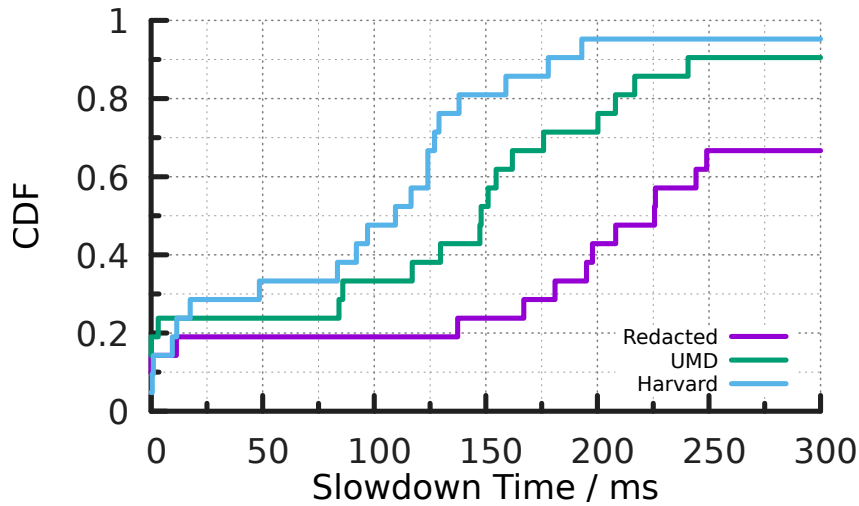


FIGURE 4.17: HTTP/1.1 Slowdown Time (FCP 90th Percentile)

For each webpage, we run the MAHIMAHIM emulation 100 times without flipping and then 100 times with flipping. We then compute the average and 90th percentile for FCP and PLT (We do not present connect and TTFB here as the impact of these two metrics on web performance are also reflected on FCP and PLT). To ease visualization and comparison, we introduce two new metrics. Slowdown time is the difference between the same metric (e.g., average FCP) with anycast flipping and without anycast flipping. Slowdown ratio is slowdown time divided by the metric without anycast flipping. A slowdown ratio of 1.0 means that anycast flipping doubled the value of the underlying metric.

Figures 4.14 and 4.15 show the slowdown time for average FCP as a CDF across the 20 webpages for HTTP/1.1 and HTTP/2, respectively. For half of the webpages we study, the anycast flipping we emulate increases average FCP by 81-111ms for HTTP/1.1 and 77-108ms for HTTP/2, depending upon vantage point. Similarly, Figures 4.17 and 4.18 show the equivalent results for the 90th percentile FCP and slowdown times over several hundred milliseconds are common. This logically follows from our results with the mock webpage, where we observe anycast flipping’s impact

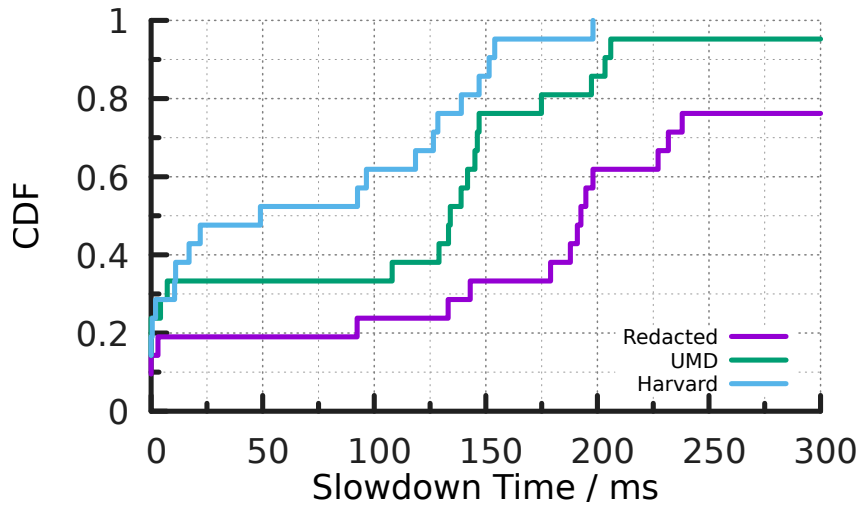


FIGURE 4.18: HTTP/2 Slowdown Time (FCP 90th Percentile)

is more significant in the tail of the distribution.

Interesting, the results are similar regardless of protocol used, while on the mock webpage there was a notable difference. Not surprisingly, we find that hosting many objects on the same host is rare in real webpages. In fact, domain sharding – the process of splitting resources among many domain names to work around HTTP/1.1 head-of-line blocking – is commonly used even with HTTP/2 [SBWR21]. Likely unintentionally, domain sharding currently mitigates that worst-case impact of anycast flipping on HTTP/2. Thus, website operators should consider the impact of anycast flipping on their websites before removing domain sharding.

Since there is no major difference in the results by protocol, we show the slowdown ratio of only HTTP/2 in Figures 4.16 and 4.19 for average and 90th percentile FCP, respectively. The results vary by vantage point, which is to be expected as the RTTs to the hosts serving the webpages differ among the sites. Thus, the additional 50ms from flipping will have more/less impact. For Harvard and UMD, most sites have small slowdown ratios, indicating that anycast flipping from those vantage points often does not significantly degrade perceived performance. [Redacted for review],

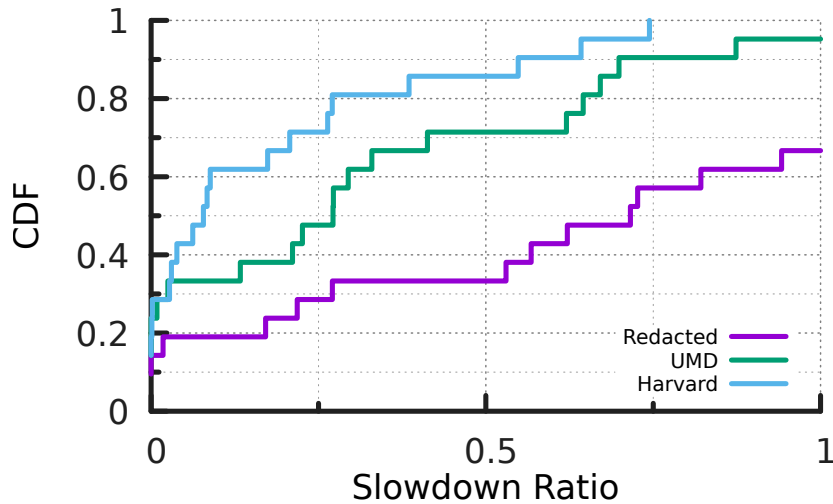


FIGURE 4.19: HTTP/2 Slowdown Ratio (FCP 90th Percentile)

on the other hand, has significantly larger slowdown ratios, showing the time added by anycast flipping is a larger portion of the overall FCP. However, turning to the 90th percentile for Harvard and UMD, 8 webpages have slowdown ratios of about 0.5 or more, So, again, tail performance can be dramatically worse due to anycast flipping.

In Figures 4.22 and 4.25, the results for PLT time are shown, and are similar. In summary, for several of the webpages studied, anycast flipping – at levels observed to impact 1.3% of BrightData residential proxies – can significantly impact web performance.

4.6 Related Work

Measuring IP anycast performance. IP anycast has long been used by Internet services to provide automatic load balancing and latency reduction with multiple anycast sites [CFKB⁺15a, BFR06b, dOSHK17b]. Many related works focus on measuring and analyzing the performance of existing IP anycast systems, including DNS root servers [LLC⁺13a, LHF⁺07a, MSH⁺16a, LJD⁺13a, GCF⁺16, KLA⁺21]

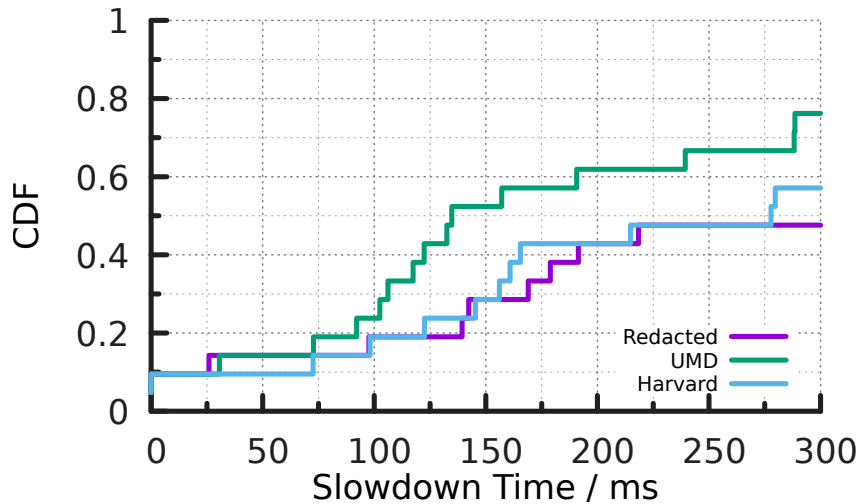


FIGURE 4.20: HTTP/1.1 Slowdown Time (PLT median)

and CDNs [CFKB⁺15a, dVARD20, KLA⁺21]. The main results from these studies are consistent: Global IP anycast does not always route clients to the sites that provide the lowest latency and does not always evenly distribute the workload among the sites [SPT05].

Many previous works assume that each client reaches only one anycast site, which we observe is not true in practice. In our work, we show that, as many as 10% of the clients reach multiple different anycast sites due to anycast flipping. Schomp and Al-Dalky [SAD20b] also observe that 17% of clients flip between sites in proprietary DNS server logs, and speculate that one cause may be load balancers.

Load-balancer detection. From the examples in Section 4.2, load-balancing within the network is one cause of anycast flipping. Load-balancers are a widely deployed technique to utilize multiple links in order to support large traffic volumes. Paxson’s measurement work on the diversity of routing behavior [Pax96] is the first work to show the impact on the performance due to the load balancing, but only limited to unicast routing. Almeida et al. [ACT⁺20] proposed MCA (Multipath Classification

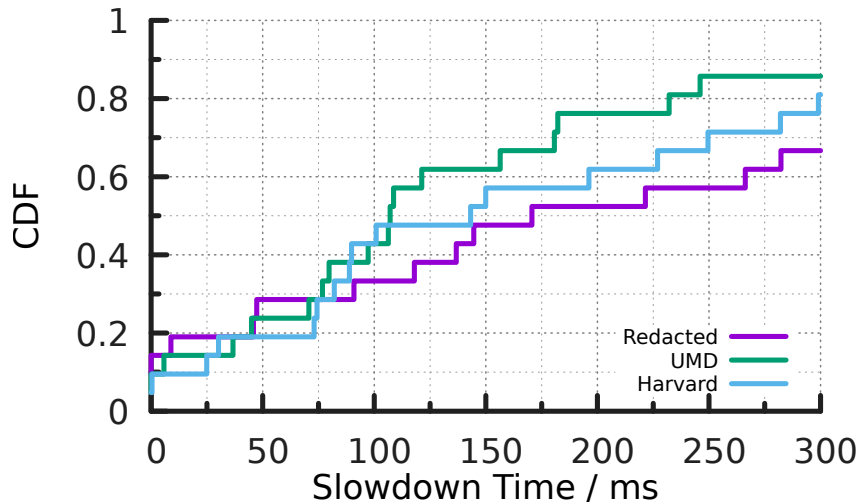


FIGURE 4.21: HTTP/2 Slowdown Time (PLT median)

Algorithm) to figure out the field in the packet used for load-balancing along a path, which in the presence of anycast can cause different flows – with different field values – to reach different sites. Paris traceroute [ACO⁺06] is a tool to perform path traces in the presence of load balancers. By keeping the fields load balancers might use constant, Paris traceroute tracks the complete path along one branch of the load balancer at a time. None of these works, however, consider the impact on anycast by load-balancers. Based on Paris traceroute, Augustin et al. proposed a multipath detection algorithm to probe multipath of load balance [AFT07]. Furthermore, Diamond-Miner [VRB⁺20] combines this multipath detection algorithm [AFT07] with high-speed randomized probing techniques [Bev16] to construct the Internet-scale topology with multipath of load balance.

Lan and Heidemann’s work [WH17a] is the first work that quantifies anycast flipping. In our work, we reproduced their results from 2016 with the same method, and update them to today. We find that anycast flipping is more common now, and expand to measure the performance impact of anycast flipping.

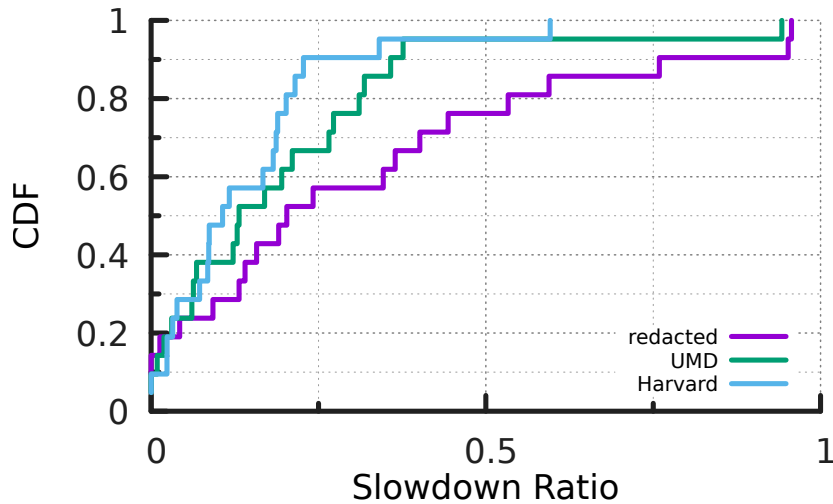


FIGURE 4.22: HTTP/2 Slowdown Ratio (PLT median)

4.7 Conclusions and Future work

In this paper, we revisited the anycast flipping problem and provided evidence that it is increasing in prevalence, and that for some clients it can significantly impact web-browsing performance.

This paper leaves open a number of interesting directions for future work.

Limitations. The web browsing performance study reported in this paper has several limitations. Most prominently, it’s likely that important applications other than web browsing are also impacted by anycast flipping, and should be evaluated. Putting aside other applications, the browsing analysis measures only the First Contentful Paint and Page Load Time metrics. Metrics such as Connect Time, Time to First Byte, and even customer conversion rate could also be considered. Perhaps more importantly, the clients from which the popular web sites were downloaded were not very diverse. For example, they did not include hosts on home networks or mobile hosts using cellular data connections. Finally, we only emulated two fixed flipping latency penalties (10ms and 50ms), both for the same flipping probability,

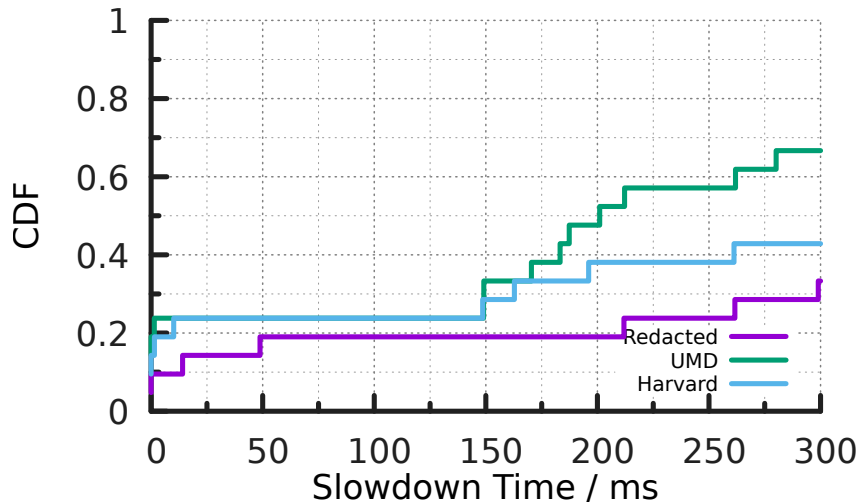


FIGURE 4.23: HTTP/1.1 Slowdown Time (PLT 90th Percentile)

50%, which is a large flipping probability experienced by a relatively small fraction of clients. An exhaustive study would evaluate a wider range of probabilities and penalties.

Impact on anycast CDN cache performance. If a browser requests the objects embedded on a web page from multiple anycast CDN sites chosen essentially at random, then in order to guarantee cache hits the CDN must store all of the objects at all of the sites. Alternatively, anycast flipping may reduce the cache hit rate. Our experiments with the anycast CDN were oblivious to this effect, as we measured only TCP connect time, and not object download time. But it would be interesting to further explore the effect of anycast flipping on caching.

Preventing flipping. If it is deemed desirable to reduce the occurrence of anycast flipping, a number of potential approaches come to mind. First, at present Internet routers are generally oblivious to the notion of anycast addresses. If designated prefixes were reserved for anycast use, routers could disable load balancing for any datagrams or flows to those prefixes. Alternatively, perhaps datagrams could in-

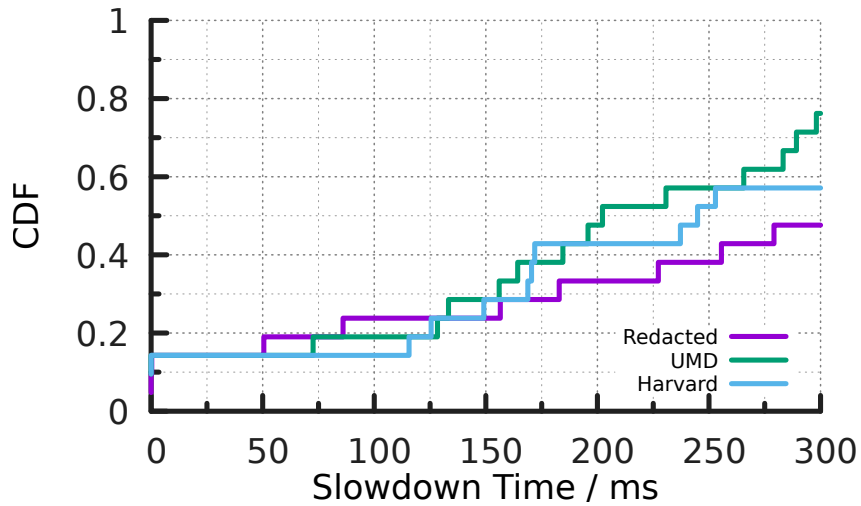


FIGURE 4.24: HTTP/2 Slowdown Time (PLT 90th Percentile)

clude “do-not-load-balance” tags. Second, anycast CDNs could employ a hybrid approach to delivering content, using an anycast address for the server delivering an HTML document, but then using a unicast address (for the same server site) for any embedded objects.

Mitigating flipping. We observed that in the default Chrome implementation, in HTTP/1.1 multiple TCP connections are established to download the necessary content, and the connections that perform best are used to download more objects. This adaptive adjustment to anycast flipping mitigates the impact of anycast sites with poor performance.

4.8 Appendix

4.8.1 Geocodes Used by Different Roots

Top 20 web sites

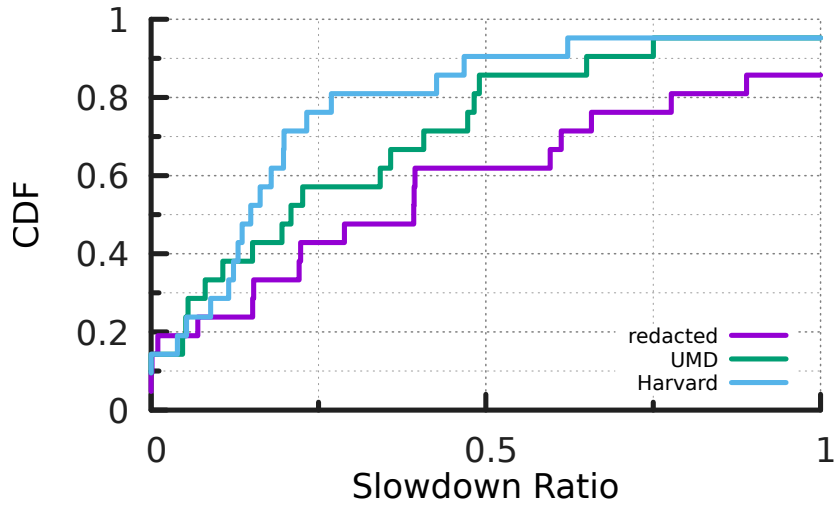


FIGURE 4.25: HTTP/2 Slowdown Ratio (PLT 90th Percentile)

Table 4.4: Naming schemes used by different operators

Root	Operator	Naming Scheme
A	Verisign	IATA, UN/LOCODE
B	USC-ISI	IATA
C	Cogent	IATA
D	UMD	City/Country Code
E	NASA Ames	IATA
F	ISC	IATA
G	DoD NIC	Other
H	ARL	IATA
I	Netnode	IATA, Other
J	Verisign	IATA, UN/LOCODE
K	RIPE NCC	City/Country Code
L	ICANN	City/Country Code
M	WIDE	IATA

Table 4.5: Naming schemes used by different operators (continue)

Root	Regular Expression
A	$(\text{rootns} \text{nnn1})-(\{\w{3}\}\backslash\text{d} \{\w{2}\}\{\w{3}\}).*$
B	$\text{b}\backslash\text{d}-\{\w{3}\}$
C	$\{\w{3}\}\backslash\text{d}\backslash\text{w}\backslash.\text{c}\backslash.\text{root-servers}\backslash.\text{org}$
D	$\{\w{4}\}\backslash\text{d}\backslash.\text{droot}\backslash.\text{maxgigapop}\backslash.\text{net}$
E	$\backslash\text{w}\backslash\text{d}\{\w{2}\}\backslash.(\backslash\text{w}*)\backslash.\text{eroot}$
F	$\backslash\text{w}\{\w{3}\}\backslash.\text{cf}\backslash.\text{f}\backslash.\text{root-servers}\backslash.\text{org}$
G	$\text{groot}-\{\w{4}\}-\backslash\text{d}$
H	$\backslash\text{d}\{\w{3}\}\backslash.(\backslash\text{w}\{\w{3}\})\backslash.\text{h}\backslash.\text{root-servers}\backslash.\text{org}$
I	$\text{s}\backslash\text{d}\backslash.(\backslash\text{w}\{\w{3}\})$
J	$(\text{rootns} \text{nnn1})-(\{\w{3}\}\backslash\text{d} \{\w{2}\}\{\w{3}\}).*$
K	$\text{ns}\backslash\text{d}.\{\w{2}\}-\{\w{3}\}\backslash.\text{k}\backslash.\text{ripe}\backslash.\text{net}$
L	$(\{\w{2}\}-\{\w{3}\})-\backslash\text{w}\{\w{2}\}$
M	$\text{M}-\{\w{3}\}-.*$

Conclusion

We present our works on IP anycast performance optimization in the previous chapters. However, all those approaches try to tackle this problem regarding the Internet as a black box due to the in-cooperative property of the intermediate networks.

This means ample space could be achieved by incorporating more tuning knobs. Therefore, we could expand the research in this field with the following improvements:

- SD-WAN is a new cloud network performance management framework that can conveniently select the desired path among available ones. This could help to restrict the preferred performant path in our linear preference order;
- In AnyOpt, only one transit link in each site is enabled due to the persevering of linear preference order because it is difficult to figure out the preference order among the co-located peers. To solve this problem, we could announce multiple different prefixes in the same site and announce the unique prefix from different peers. We could also use DNS to restrict the catchment site/peer for specific client prefixes;
- In all these systems, performance probing is still time-consuming due to the

pair-wise measurement between sites and client prefixes. A probabilistic probing method could be used here to reduce the number of necessary measurements.

Bibliography

- [ACC⁺19] Todd Arnold, Matt Calder, Italo Cunha, Arpit Gupta, Harsha V Madhyastha, Michael Schapira, and Ethan Katz-Bassett. Beating bgp is harder than we thought. In *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*, pages 9–16, 2019.
- [ACF⁺12] Bernhard Ager, Nikolaos Chatzis, Anja Feldmann, Nadi Sarrar, Steve Uhlig, and Walter Willinger. Anatomy of a large european xp. In *Proceedings of the Annual Conference of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'12)*, 2012.
- [ACG⁺12] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and approximation: Combinatorial optimization problems and their approximability properties*. Springer Science & Business Media, 2012.
- [ACL06] J. Abley, Afilias Canada, and K. Lindqvist. Operation of Anycast Services. RFC 4786, RFC Editor, Dec 2006.
- [ACO⁺06] Brice Augustin, Xavier Cuvellier, Benjamin Orgogozo, Fabien Viger, Timur Friedman, Matthieu Latapy, Clémence Magnien, and Renata Teixeira. Avoiding Traceroute Anomalies with Paris Traceroute. In *Proceedings of IMC*, pages 153–158. ACM, 2006.
- [ACT⁺20] Rafael Almeida, Ítalo Cunha, Renata Teixeira, Darryl Veitch, and Christophe Diot. Classification of Load Balancing in the Internet. In *Proceedings of INFOCOM*, pages 1987–1996, 2020.
- [AFT07] Brice Augustin, Timur Friedman, and Renata Teixeira. Multipath Tracing with Paris Traceroute. In *Workshop on End-to-End Monitoring Techniques and Services*, pages 1–8. IEEE, 2007.

- [AGE⁺20] Todd Arnold, Ege Gürmeriçliler, Georgia Essig, Arpit Gupta, Matt Calder, Vasileios Giotsas, and Ethan Katz-Bassett. (how much) does a private wan improve cloud performance? In *Proceedings of the Annual Conference of the IEEE International Conference on Computer Communications (INFOCOM'20)*, 2020.
- [Aka] Akamai Technologies. Akamai Online Retail Performance Report: Milliseconds are Critical. Retrieved in Sep, 2022 from <https://www.ir.akamai.com/news-releases/news-release-details/akamai-online-retail-performance-report-milliseconds-are>.
- [ale] Alexa Top Sites. Retrieved in April, 2022 from https://docs.aws.amazon.com/fr_fr/AlexaTopSites/latest/ApiReference_TopSitesAction.html.
- [ALR⁺08] Hussein A. Alzoubi, Seungjoon Lee, Michael Rabinovich, Oliver Spatscheck, and Jacobus Van der Merwe. Anycast CDNS Revisited. In *Proceedings of the 17th International Conference on World Wide Web (WWW'08)*, 2008.
- [ALR⁺11] Hussein A. Alzoubi, Seungjoon Lee, Michael Rabinovich, Oliver Spatscheck, and Jacobus Van Der Merwe. A practical architecture for an anycast cdn. *ACM Trans. Web*, 5(4), October 2011.
- [Ama] Amazon. Values Specific for Geolocation Records. Retrieved in Sep, 2022 from <https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resource-record-sets-values-geo.html#rrsets-values-geo-location>.
- [ANC⁺15a] Ruwaifa Anwar, Haseeb Niaz, David Choffnes, Ítalo Cunha, Phillipa Gill, and Ethan Katz-Bassett. Investigating interdomain routing policies in the wild. In *Proceedings of the 2015 Internet Measurement Conference, IMC '15*, page 71–77, New York, NY, USA, 2015. Association for Computing Machinery.
- [ANC⁺15b] Ruwaifa Anwar, Haseeb Niaz, David Choffnes, Ítalo Cunha, Phillipa Gill, and Ethan Katz-Bassett. Investigating interdomain routing policies in the wild. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'15)*, 2015.
- [ANC⁺15c] Ruwaifa Anwar, Haseeb Niaz, David Choffnes, Ítalo Cunha, Phillipa Gill, and Ethan Katz-Bassett. Investigating interdomain

- routing policies in the wild. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'15)*, 2015.
- [AS72] M. Abramowitz and I. A. (Eds.). Stegun. *Handbook of Mathematical Functions With Formulas, Graphs, and Mathematical Tables*. 1972.
- [ASBK17] Adnan Ahmed, Zubair Shafiq, Harkeerat Bedi, and Amir Khakpour. Peering vs. transit: Performance comparison of peering and transit interconnections. In *2017 IEEE 25th International Conference on Network Protocols (ICNP)*, pages 1–10, 2017.
- [BCdV⁺21] Leandro M Bertholdo, Joao M Ceron, Wouter B de Vries, Ricardo de Oliveira Schmidt, Lisandro Zambenedetti Granville, Roland van Rijswijk-Deij, and Aiko Pras. Tangled: A Cooperative Anycast Testbed. In *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM'21)*, 2021.
- [Bev16] Robert Beverly. Yarrp'ing the Internet: Randomized High-speed Active Topology Discovery. In *Proceedings of the 2016 Internet Measurement Conference*, pages 413–420. ACM, 2016.
- [BF05a] Hitesh Ballani and Paul Francis. Towards a Global IP Anycast Service. In *Proceedings of the Annual Conference of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'05)*, 2005.
- [BF05b] Hitesh Ballani and Paul Francis. Towards a global ip anycast service. In *ACM SIGCOMM Computer communication review*, volume 35, pages 301–312. ACM, 2005.
- [BFR06a] Hitesh Ballani, Paul Francis, and Sylvia Ratnasamy. A Measurement-Based Deployment Proposal for IP Anycast. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'06)*, 2006.
- [BFR06b] Hitesh Ballani, Paul Francis, and Sylvia Ratnasamy. A measurement-based deployment proposal for ip anycast. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement, IMC '06*, page 231–244, New York, NY, USA, 2006. Association for Computing Machinery.

- [BHW⁺19] Rui Bian, Shuai Hao, Haining Wang, Amogh Dhamdere, Alberto Dainotti, and Chase Cotton. Towards Passive Analysis of Anycast in Global Routing: Unintended Impact of Remote Peering. *ACM SIGCOMM Comput. Commun. Rev.*, 49(3):18–25, 2019.
- [bri23a] Brightdata, 2023.
- [Bri23b] BrightData. Session IP persistence. <https://help.brightdata.com/hc/en-us/articles/4413171447953-Session-IP-persistence>, 2023.
- [bro23] Browsertime, 2023.
- [CAJ⁺15a] Danilo Cicalese, Jordan Augé, Diana Jounblatt, Timur Friedman, and Dario Rossi. Characterizing ipv4 anycast adoption and deployment. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, CoNEXT '15, New York, NY, USA, 2015. Association for Computing Machinery.
- [CAJ⁺15b] Danilo Cicalese, Jordan Augé, Diana Jounblatt, Timur Friedman, and Dario Rossi. Characterizing IPv4 Anycast Adoption and Deployment. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT '15)*, 2015.
- [CCGF14a] Ignacio Castro, Juan Camilo Cardona, Sergey Gorinsky, and Pierre Francois. Remote peering: More peering without internet flattening. In *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, CoNEXT '14, page 185–198, New York, NY, USA, 2014. Association for Computing Machinery.
- [CCGF14b] Ignacio Castro, Juan Camilo Cardona, Sergey Gorinsky, and Pierre Francois. Remote Peering: More Peering without Internet Flattening. In *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies (CoNEXT'14)*, 2014.
- [CDNa] CDN Finder. Retrieved in Sep, 2022 from <https://www.cdnplanet.com/tools/cdnfinder/>.
- [cdnb] CDN Overview. Retrieved in Sep, 2022 from <https://www.cdnoverview.com/>.

- [CFH⁺13a] M. Calder, X. Fan, Z. Hu, E. Katz-Bassett, J. Heidemann, and R. Govindan. Mapping the Expansion of Google’s Serving Infrastructure. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC’13)*, 2013.
- [CFH⁺13b] Matt Calder, Xun Fan, Zi Hu, Ethan Katz-Bassett, John Heidemann, and Ramesh Govindan. Mapping the Expansion of Google’s Serving Infrastructure. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC’13)*, 2013.
- [CFKB⁺15a] Matt Calder, Ashley Flavel, Ethan Katz-Bassett, Ratul Mahajan, and Jitendra Padhye. Analyzing the performance of an anycast cdn. In *Proceedings of the 2015 Internet Measurement Conference, IMC ’15*, page 531–537, New York, NY, USA, 2015. Association for Computing Machinery.
- [CFKB⁺15b] Matt Calder, Ashley Flavel, Ethan Katz-Bassett, Ratul Mahajan, and Jitu Padhye. Analyzing the Performance of an Anycast CDN. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC’15)*, 2015.
- [CGLV19] Massimo Candela, Enrico Gregori, Valerio Luconi, and Alessio Vecchio. Using RIPE Atlas for Geolocating IP Infrastructure. *IEEE Access*, 7:48816–48829, 2019.
- [cis16] Bgp best path selection algorithm, sep 2016.
- [cis18] BGP Bestpath AS-Path Multipath-Relax, 2018.
- [CJR⁺15] Danilo Cicalese, Diana Joumblatt, Dario Rossi, Marc-Olivier Buob, Jordan Augé, and Timur Friedman. A Fistful of Pings: Accurate and Lightweight Anycast Enumeration and Geolocation. In *Proceedings of the Annual Conference of the IEEE International Conference on Computer Communications (INFOCOM’15)*, 2015.
- [Clo] Google Cloud. Cloud CDN. Retrieved in January, 2023 from <https://cloud.google.com/cdn>.
- [Clo22] Cloudflare. Cloudflare CDN Reference Architecture. https://cf-assets.www.cloudflare.com/s1t31c6tev37/18dA4NLfq8oXY8EVZxPlpY/b9cab82be79ebefa80f08c09eaa3d93e/Cloudflare_CDN_Reference_Architecture.pdf, 2022.

- [Clo23] Cloudflare. HTTP request headers. <https://developers.cloudflare.com/fundamentals/get-started/reference/http-request-headers/#cf-ray>, 2023.
- [CMK⁺21] Rishabh Chhabra, Paul Murley, Deepak Kumar, Michael Bailey, and Gang Wang. Measuring DNS-over-HTTPS Performance Around the World. In *Proceedings of IMC*, pages 351–365. ACM, 2021.
- [cna20] CNAME Reuse. <https://docs.imperva.com/bundle/cloud-application-security/page/more/cname-reuse.htm>, 2020.
- [CNW83] Gérard Cornuéjols, George Nemhauser, and Laurence Wolsey. The uncapacitated facility location problem. Technical report, Cornell University Operations Research and Industrial Engineering, 1983.
- [CR18] Danilo Cicalese and Dario Rossi. A Longitudinal Study of IP Anycast. *ACM SIGCOMM Comput. Commun. Rev.*, 48(1):10–18, 2018.
- [CRUR] Lorenzo Colitti, Erik Romijn, Henk Uijterwaal, and Andrei Robachevsky. Evaluating the Effects of Anycast on DNS Root Name Servers. Retrieved in Sep, 2022 from <https://www.ripe.net/publications/docs/ripe-393>.
- [CSG⁺18] Matt Calder, Manuel Schroder, Ryan Gao, Ryan Stewart, Jitu Padhye, Ratul Mahajan, Ganesh Ananthanarayanan, and Ethan Katz-Bassett. Odin: Microsoft’s Scalable Fault-Tolerant CDN Measurement System. In *Proceedings of 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI’18)*, 2018.
- [CST15a] Fangfei Chen, Ramesh K. Sitaraman, and Marcelo Torres. End-User Mapping: Next Generation Request Routing for Content Delivery. In *Proceedings of the Annual Conference of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM’15)*, 2015.
- [CST15b] Fangfei Chen, Ramesh K Sitaraman, and Marcelo Torres. End-user mapping: Next generation request routing for content delivery. *ACM SIGCOMM Computer Communication Review*, 45(4):167–181, 2015.

- [CvdGLK16] Carlo Contavalli, Wilmer van der Gaast, David C Lawrence, and Warren "Ace" Kumari. Client Subnet in DNS Queries. Technical report, RFC Editor, Dec 2016.
- [CW07a] David R. Conrad and Suzanne Woolf. Requirements for a Mechanism Identifying a Name Server Instance. RFC 4892, June 2007.
- [CW07b] David R. Conrad and Suzanne Woolf. Requirements for a Mechanism Identifying a Name Server Instance. Technical Report 4892, RFC Editor, June 2007.
- [DFB21] Trinh Viet Doan, Justus Fries, and Vaibhav Bajpai. Evaluating Public DNS Services in the Wake of Increasing Centralization of DNS. In *Proceedings of the IFIP Networking Conference*, 2021.
- [dns] TCP over IP Anycast - Pipe Dream or Reality? Retrieved in Sep, 2022 from <https://blog.catchpoint.com/2015/09/24/tcp-over-ip-anycast/>.
- [dOSHK17a] Ricardo de O. Schmidt, John Heidemann, and Jan Kuipers. Anycast Latency: How Many Sites Are Enough? In *Proceedings of Passive and Active Measurement (PAM'17)*, 2017.
- [dOSHK17b] Ricardo de Oliveira Schmidt, John Heidemann, and Jan Harm Kuipers. Anycast latency: How many sites are enough? In Mohamed Ali Kaafar, Steve Uhlig, and Johanna Amann, editors, *Passive and Active Measurement*, pages 188–200, Cham, 2017. Springer International Publishing.
- [DRM⁺19] Dmitry Duplyakin, Robert Ricci, Aleksander Maricq, Gary Wong, Jonathon Duerig, Eric Eide, Leigh Stoller, Mike Hibler, David Johnson, Kirk Webb, Aditya Akella, Kuangching Wang, Glenn Ricart, Larry Landweber, Chip Elliott, Michael Zink, Emmanuel Cecchet, Snigdhaswin Kar, and Prabodh Mishra. The design and operation of CloudLab. In *Proceedings of the USENIX Annual Technical Conference (ATC)*, pages 1–14, 2019.
- [dVARD20] Wouter B. de Vries, Salmān Aljammāz, and Roland van Rijswijk-Deij. Global-scale anycast network management with verfploeter. In *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, pages 1–9, 2020.

- [dVAvRD20] Wouter B de Vries, Salmān Aljammāz, and Roland van Rijswijk-Deij. Global-Scale Anycast Network Management with Verfploeter. In *Proceedings of 2020 IEEE/IFIP Network Operations and Management Symposium (NOMS'20)*, 2020.
- [dVdOSH⁺17a] Wouter B. de Vries, Ricardo de O. Schmidt, Wes Hardaker, John Heidemann, Pieter-Tjerk de Boer, and Aiko Pras. Broad and load-aware anycast mapping with verfploeter. In *Proceedings of the 2017 Internet Measurement Conference, IMC '17*, page 477–488, New York, NY, USA, 2017. Association for Computing Machinery.
- [DVdOSH⁺17b] Wouter B. De Vries, Ricardo de Oliveira Schmidt, Wes Hardaker, John Heidemann, Pieter-Tjerk de Boer, and Aiko Pras. Broad and Load-Aware Anycast Mapping with Verfploeter. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'17)*, 2017.
- [DVvRDDBP19] Wouter B De Vries, Roland van Rijswijk-Deij, Pieter-Tjerk De Boer, and Aiko Pras. Passive Observations of a Large DNS Service: 2.5 Years in the Life of Google. *IEEE transactions on network and service management*, 17(1):190–200, 2019.
- [EC13] B. Eriksson and M. Crovella. Understanding Geolocation Accuracy Using Network Geometry. In *Proceedings of the Annual Conference of the IEEE International Conference on Computer Communications (INFOCOM'13)*, 2013.
- [Edg] EdgeCast. EdgeCast POPs. Retrieved in Sep, 2022 from https://docs.edgecast.com/cdn/Content/Reference/POP_Listing.htm.
- [EE06] Rekhter Y. Ed. Li T. Ed. and S. Hares Ed. A Border Gateway Protocol 4 (BGP-4). RFC 4271, RFC Editor, 2006.
- [fAIDACa] Center for Applied Internet Data Analysis (CAIDA). Caida data server-peeringdb archive jaug-30-2022j. Retrieved in Feb, 2023 from https://publicdata.caida.org/datasets/peeringdb/2022/08/peeringdb_2_dump_2022_08_30.json.
- [fAIDACb] Center for Applied Internet Data Analysis (CAIDA). Routeviews prefix to AS mappings dataset for IPv4 and IPv6. <http://www.caida.org/data/routing/routeviews-prefix2as.xml>.

- [fAIDACc] Center for Applied Internet Data Analysis (CAIDA). The CAIDA AS Relationships Dataset, [30, Aug, 2022]. Retrieved in Feb, 2023 from <https://www.caida.org/catalog/datasets/as-relationships/>.
- [FBG⁺21] Marwan Fayed, Lorenz Bauer, Vasileios Giotsas, Sami Kerola, Marek Majkowski, Pavel Odintsov, Jakub Sitnicki, Taejoong Chung, Dave Levin, Alan Mislove, et al. The Ties that Un-Bind: Decoupling IP from web services and sockets for robust addressing agility at CDN-scale. In *Proceedings of the Annual Conference of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'21)*, 2021.
- [fcp23] First contentful paint (fcp), 2023.
- [fE22] United Nations Economic Commission for Europe. UN/LOCODE Code List by Country and Territory — UNECE, 2022.
- [FH10] Xun Fan and John Heidemann. Selecting Representative IP Addresses for Internet Topology Studies. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, IMC '10*, pages 411–423, New York, NY, USA, 2010. Association for Computing Machinery.
- [FHG13] Xun Fan, John Heidemann, and Ramesh Govindan. Evaluating Anycast in the Domain Name System. In *Proceedings of the Annual Conference of the IEEE International Conference on Computer Communications (INFOCOM'13)*, 2013.
- [FHLT94a] Dino Farinacci, Stanley P. Hanks, Dr. Tony Li, and Paul S. Traina. Generic Routing Encapsulation over IPv4 networks. RFC 1702, October 1994.
- [FHLT94b] Dino Farinacci, Stanley P. Hanks, Dr. Tony Li, and Paul S. Traina. Generic Routing Encapsulation over IPv4 Networks. RFC 1702, RFC Editor, 1994.
- [FLH⁺00] Dino Farinacci, Tony Li, Stanley P. Hanks, David Meyer, and Paul S. Traina. Generic Routing Encapsulation (GRE). RFC 2784, RFC Editor, March 2000.

- [FMM⁺15a] Ashley Flavel, Pradeepkumar Mani, David Maltz, Nick Holt, Jie Liu, Yingying Chen, and Oleg Surmachev. Fastroute: A scalable load-aware anycast routing architecture for modern cdns. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 381–394, Oakland, CA, May 2015. USENIX Association.
- [FMM⁺15b] Ashley Flavel, Pradeepkumar Mani, David Maltz, Nick Holt, Jie Liu, Yingying Chen, and Oleg Surmachev. FastRoute: A Scalable Load-Aware Anycast Routing Architecture for Modern CDNs. In *Proceedings of 4th USENIX Symposium on Networked Systems Design & Implementation (NSDI’15)*, 2015.
- [Gao01] Lixin Gao. On inferring autonomous system relationships in the internet. *IEEE/ACM Transactions on networking*, 9(6):733–745, 2001.
- [GCF⁺16] Danilo Giordano, Danilo Cicalese, A. Finamore, M. Mellia, M. Munafò, Diana Joumblatt, and D. Rossi. A First Characterization of Anycast Traffic from Passive Traces. In *IFIP workshop on Traffic Monitoring and Analysis (TMA)*, pages 30–38, ouvain La Neuve, Belgium, April 2016.
- [GDC16] Vasileios Giotsas, Amogh Dhamdhere, and Kimberly C Claffy. Periscope: Unifying looking glass querying. In *International Conference on Passive and Active Network Measurement*, pages 177–189, March 2016.
- [GMB⁺20] Caitlin Gray, Clemens Mosig, Randy Bush, Cristel Pelsser, Matthew Roughan, Thomas C Schmidt, and Matthias Wahlisch. Bgp beacons, network tomography, and bayesian computation to locate route flap damping. In *Proceedings of the ACM Internet Measurement Conference*, pages 492–505, 2020.
- [GNK⁺21] Vasileios Giotsas, George Nomikos, Vasileios Kotronis, Pavlos Sermpezis, Petros Gigis, Lefteris Manassakis, Christoph Dietzel, Stavros Konstantaras, and Xenofontas Dimitropoulos. O peer, where art thou? uncovering remote peering interconnections at ixps. *IEEE/ACM Transactions on Networking*, 29(1):1–16, 2021.
- [GO20] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2020.

- [goo] Google Geolocation API. Retrieved in Sep, 2022 from <https://developers.google.com/maps/documentation/geolocation/overview>.
- [GR01] Lixin Gao and Jennifer Rexford. Stable internet routing without global coordination. *IEEE/ACM Trans. Netw.*, 9(6):681–692, December 2001.
- [GSG02a] Krishna P. Gummadi, Stefan Saroiu, and Steven D. Gribble. King: Estimating latency between arbitrary internet end hosts. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement*, IMW '02, page 5–18, New York, NY, USA, 2002. Association for Computing Machinery.
- [GSG02b] Krishna P Gummadi, Stefan Saroiu, and Steven D Gribble. King: Estimating Latency between Arbitrary Internet end Hosts. In *Proceedings of SIGCOMM*, pages 5–18. ACM, 2002.
- [GSH⁺17] Manaf Gharaibeh, Anant Shah, Bradley Huffaker, Han Zhang, Roya Ensafi, and Christos Papadopoulos. A look at router geolocation in public and commercial databases. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'17)*, 2017.
- [GSW02a] Timothy G Griffin, F Bruce Shepherd, and Gordon Wilfong. The stable paths problem and interdomain routing. *IEEE/ACM Transactions On Networking*, 10(2):232–243, 2002.
- [GSW02b] Timothy G Griffin, F Bruce Shepherd, and Gordon Wilfong. The stable paths problem and interdomain routing. *IEEE/ACM Transactions On Networking*, 10(2):232–243, 2002.
- [GUF07] Bamba Gueye, Steve Uhlig, and Serge Fdida. Investigating the Imprecision of IP Block-Based Geolocation. In *Proceedings of Passive and Active Measurement (PAM'07)*, 2007.
- [GZCF04] B. Gueye, A. Ziviani, M. Crovella, and S. Fdida. Constraint-based Geolocation of Internet Hosts. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'04)*, 2004.
- [Had] Hadi Asghari. pyasn · pypi. Retrieved in Feb, 2023 from <https://pypi.org/project/pyasn/>.

- [HFC11] Bradley Huffaker, Marina Fomenkov, and Kc Claffy. Geocompare: A Comparison of Public and Commercial Geolocation Databases. Technical report, Cooperative Association For Internet Data Analysis (CAIDA), 2011.
- [Hop00] C. Hopps. Analysis of an Equal-Cost Multi-Path Algorithm. RFC 2992, RFC Editor, 2000.
- [How17] How Anycast Works to Bring Content Closer to Your Visitors. <https://www.imperva.com/blog/how-anycast-works/>, 2017.
- [HP87] Pierre Hanjoul and Dominique Peeters. A facility location problem with clients' preference orderings. *Regional Science and Urban Economics*, 17(3):451–473, 1987.
- [HSF19] Paul Hoffman, Andrew Sullivan, and Kazunori Fujiwara. DNS Terminology. RFC 8499, RFC Editor, Jan 2019.
- [HWLR08] Cheng Huang, Angela Wang, Jin Li, and Keith W. Ross. Measuring and Evaluating Large-scale CDNs. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'08)*, 2008.
- [HZWS18] Shuai Hao, Yubao Zhang, Haining Wang, and Angelos Stavrou. End-Users Get Maneuvered: Empirical Analysis of Redirection Hijacking in Content Delivery Networks. In *Proceedings of 27th USENIX Security Symposium (USENIX Security'18)*, 2018.
- [(IA)] The International Air Transport Association (IATA). IATA Airport Code. Retrieved in Sep, 2022 from <https://www.iata.org/en/publications/directories/code-search/>.
- [ico] iconectiv. CLI code. Retrieved in Sep, 2022 from <https://www.commonlanguage.com/resources/commonlang/productshowroom/product/cli/>.
- [Imp] Imperva. Imperva PoPs. Retrieved in Sep, 2022 from https://status.imperva.com/?_ga=2.95445268.988020403.1637323716-57897548.1602317115.
- [imp19] Route Optimization. <https://www.imperva.com/learn/performance/route-optimization-anycast/>, 2019.

- [IPI] IPInfo. IPinfo. Retrieved in Sep, 2022 from <https://ipinfo.io/>.
- [isl] PeeringDB. Retrieved in Sep, 2022 from <https://www.peeringdb.com/net/6639>.
- [JBP⁺02] S. V. Jones, S. Brown, T. Parker, V. Stubblefield, E. Kössler, C. Simmons, and J. Wittner. Anomalous indeterminate amorphous transitory gradations observed in certain specific nongeneralizable locii attendant to unique fleeting unpredictable phenomena. *Journal of Nonspecific Research*, 88:665–703, 2002.
- [JHWC18] Lin Jin, Shuai Hao, Haining Wang, and Chase Cotton. Your Remnant Tells Secret: Residual Resolution in DDoS Protection Services. In *Proceedings of the 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'18)*, 2018.
- [JLK⁺21] Weifan Jiang, Tao Luo, Thomas Koch, Yunfan Zhang, Ethan Katz-Bassett, and Matt Calder. Towards Identifying Networks with Internet Clients Using Public Data. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'21)*, 2021.
- [jun20] Understanding bgp path selection, dec 2020.
- [jun23] BGP User Guide: multipath (Protocols BGP), 2023.
- [k-r23] root-servers.org-k-root, 2023.
- [KBSC⁺12] Ethan Katz-Bassett, Colin Scott, David R. Choffnes, Ítalo Cunha, Vytautas Valancius, Nick Feamster, Harsha V. Madhyastha, Thomas Anderson, and Arvind Krishnamurthy. Lifeguard: Practical repair of persistent route failures. In *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '12, page 395–406, New York, NY, USA, 2012. Association for Computing Machinery.
- [KLA⁺21] Thomas Koch, Ke Li, Calvin Ardi, Ethan Katz-Bassett, Matt Calder, and John Heidemann. Anycast in context: A tale of two systems. In *Proceedings of the 2021 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '21, New York, NY, USA, 2021. Association for Computing Machinery.

- [Kui] JH Kuipers. Analysing the K-root Anycast Infrastructure. Retrieved in Sep, 2022 from https://labs.ripe.net/Members/jh_kuipers/analyzing-the-k-root-anycast-infrastructure.
- [l-r23] root-servers.org-l-root, 2023.
- [LBM⁺20] Shucheng Liu, Zachary S. Bischof, Ishaan Madan, Peter K. Chan, and Fabián E. Bustamante. Out of sight, not out of mind: A user-view on the criticality of the submarine cable network. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'20)*, 2020.
- [LHC19] Matthew Luckie, Bradley Huffaker, and K Claffy. Learning Regexes to Extract Router Names from Hostnames. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'19)*, 2019.
- [LHF⁺07a] Ziqian Liu, Bradley Huffaker, Marina Fomenkov, Nevil Brownlee, and kc claffy. Two days in the life of the dns anycast root servers. In Steve Uhlig, Konstantina Papagiannaki, and Olivier Bonaventure, editors, *Passive and Active Network Measurement*, pages 125–134, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.
- [LHF⁺07b] Ziqian Liu, Bradley Huffaker, Marina Fomenkov, Nevil Brownlee, and Kimberly Claffy. Two Days in the Life of the DNS Anycast Root Servers. In *Proceedings of Passive and Active Measurement (PAM'07)*, 2007.
- [LHM⁺21] Matthew Luckie, Bradley Huffaker, Alexander Marder, Zachary Bischof, Marianne Fletcher, and K Claffy. Learning to Extract Geographic Information from Internet Router Hostnames. In *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT'21)*, 2021.
- [LJD⁺13a] Jinjin Liang, Jian Jiang, Haixin Duan, Kang Li, and Jianping Wu. Measuring query latency of top level dns servers. In Matthew Roughan and Rocky Chang, editors, *Passive and Active Measurement*, pages 145–154, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
- [LJD⁺13b] Jinjin Liang, Jian Jiang, Haixin Duan, Kang Li, and Jianping Wu. Measuring Query Latency of Top Level DNS Servers. In *Proceedings of Passive and Active Measurement (PAM'13)*, 2013.

- [LLC⁺13a] Matthew Lentz, Dave Levin, Jason Castonguay, Neil Spring, and Bobby Bhattacharjee. D-mystifying the d-root address change. In *Proceedings of the 2013 Conference on Internet Measurement Conference, IMC '13*, page 57–62, New York, NY, USA, 2013. Association for Computing Machinery.
- [LLC⁺13b] Matthew Lentz, Dave Levin, Jason Castonguay, Neil Spring, and Bobby Bhattacharjee. D-Mystifying the D-Root Address Change. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'13)*, 2013.
- [LLSB18a] Zhihao Li, Dave Levin, Neil Spring, and Bobby Bhattacharjee. Internet anycast: Performance, problems, & potential. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, SIGCOMM '18*, page 59–73, New York, NY, USA, 2018. Association for Computing Machinery.
- [LLSB18b] Zhihao Li, Dave Levin, Neil Spring, and Bobby Bhattacharjee. Internet Anycast: Performance, Problems, & Potential. In *Proceedings of the Annual Conference of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'18)*, 2018.
- [LLU06] Matt Levine, Barrett Lyon, and Todd Underwood. TCP Anycast: Don't Believe the FUD - Operational experience with TCP and Anycast. NANOG 37, 2006.
- [LMH⁺11] S. Laki, P. Mátray, P. HÁga, T. Sebők, I. Csabai, and G. Vattay. Spotter: A model based active geolocation service. In *Proceedings of the Annual Conference of the IEEE International Conference on Computer Communications (INFOCOM'11)*, 2011.
- [LSLS07] F Thomson Leighton, Ravi Sundaram, Matthew Levine, and Adrian Soviani. Method for generating a network map, July 31 2007. US Patent 7,251,688.
- [LT23] P. Lapukhov and J. Tantsura. Equal-Cost Multipath Considerations for BGP. Internet Draft, 2023.
- [LVT⁺19] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation. In *Pro-*

ceedings of the 26th Annual Network and Distributed System Security Symposium (NDSS), 2019.

- [Mac67] J MacQueen. Classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symp. Math. Statist. Probability*, 1967.
- [Mah15] Ritesh Maheshwari. TCP over IP Anycast - Pipe Dream or Reality? Retrieved in Sep, 2022 from <https://engineering.linkedin.com/network-performance/tcp-over-ip-anycast-pipe-dream-or-reality>, 2015.
- [Max] Inc. MaxMind. MaxMind. Retrieved in Sep, 2022 from <https://www.maxmind.com/en/home>.
- [MBGR03] Z Morley Mao, Randy Bush, Timothy G Griffin, and Matthew Roughan. Bgp beacons. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 1–14, 2003.
- [MCD⁺02] Zhuoqing Mao, Charles Cranor, Fred Douglass, Michael Rabinovich, Oliver Spatscheck, and Jia Wang. A Precise and Efficient Evaluation of the Proximity Between Web Clients and Their Local DNS Servers. In *Proceedings of the USENIX Annual Technical Conference (USENIX ATC'02)*, 2002.
- [MCH⁺20] Giovane C. M. Moura, Sebastian Castro, Wes Hardaker, Maarten Wullink, and Cristian Hesselman. Clouding Up the Internet: How Centralized is DNS Traffic Becoming? In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'20)*, 2020.
- [MCM13] Doug Madory, Chris Cook, and Kevin Miao. Who are the anycasters. *Proceedings of NANOG59*, 10, 2013.
- [MCZ⁺20] Nitinder Mohan, Lorenzo Corneo, Aleksandr Zavodovski, Suzan Bayhan, Walter Wong, and Jussi Kangasharju. Pruning edge research with latency shears. In *Proceedings of the 19th ACM Workshop on Hot Topics in Networks*, 2020.
- [Met02] Chris Metz. Ip anycast point-to-(any) point communication. *IEEE Internet Computing*, 6(2):94–98, 2002.

- [Mey05] David Meyer. University of oregon route views project. <http://www.routeviews.org/>, 2005.
- [MHH⁺20] Giovane CM Moura, John Heidemann, Wes Hardaker, Jeroen Bulten, Joao Ceron, and Cristian Hesselman. Old but gold: Prospecting tcp to engineer dns anycast (extended). Technical report, Tech. Rep. ISI-TR-740, USC/Information Sciences Institute, 2020.
- [Mic] Microsoft. What is a content delivery network on Azure? Retrieved in January, 2023 from <https://docs.microsoft.com/en-us/azure/cdn/cdn-overview>.
- [mit] Redacted for review.
- [MMdOSH17] Moritz Müller, Giovane CM Moura, Ricardo de O. Schmidt, and John Heidemann. Recursives in the Wild: Engineering Authoritative DNS Servers. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'17)*, 2017.
- [Moc87] P. Mockapetris. Domain Names - Implementation and Specification. RFC 1035, RFC Editor, Nov 1987.
- [MS15] Bruce M Maggs and Ramesh K Sitaraman. Algorithmic Nuggets in Content Delivery. *ACM SIGCOMM Comput. Commun. Rev.*, 45(3):52–66, 2015.
- [MSH⁺16a] Giovane C.M. Moura, Ricardo de O. Schmidt, John Heidemann, Wouter B. de Vries, Moritz Muller, Lan Wei, and Cristian Hesselman. Anycast vs. ddos: Evaluating the november 2015 root dns event. In *Proceedings of the 2016 Internet Measurement Conference*, IMC '16, page 255–270, New York, NY, USA, 2016. Association for Computing Machinery.
- [MSH⁺16b] Giovane C.M. Moura, Ricardo de O. Schmidt, John Heidemann, Wouter B. de Vries, Moritz Muller, Lan Wei, and Cristian Hesselman. Anycast vs. DDoS: Evaluating the November 2015 Root DNS Event. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'16)*, 2016.
- [MUF19a] Stephen McQuistin, Sree Priyanka Uppu, and Marcel Flores. Taming Anycast in the Wild Internet. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'19)*, 2019.

- [MUF19b] Stephen McQuistin, Sree Priyanka Uppu, and Marcel Flores. Taming anycast in the wild internet. In *Proceedings of the Internet Measurement Conference, IMC '19*, page 165–178, New York, NY, USA, 2019. Association for Computing Machinery.
- [Mul] Mullaned. Simplify Traffic Steering with Cloud DNS Routing Policies. Retrieved in Sep, 2022 from <https://cloud.google.com/blog/products/networking/dns-routing-policies-for-geo-location--weighted-round-robin>.
- [Naw28] B. Nawahi. Acoustic production in liquid filled ceramic environments. *J. Chem. Phys.*, 108:9893–9904, 1928.
- [NKS⁺18] George Nomikos, Vasileios Kotronis, Pavlos Sermpezis, Petros Gigis, Lefteris Manassakis, Christoph Dietzel, Stavros Konstantaras, Xenofontas Dimitropoulos, and Vasileios Giotsas. O Peer, Where Art Thou?: Uncovering Remote Peering Interconnections at IXPs. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'18)*, 2018.
- [NSD⁺15] Ravi Netravali, Anirudh Sivaraman, Somak Das, Ameesh Goyal, Keith Winstein, James Mickens, and Hari Balakrishnan. Mahimahi: Accurate Record-and-Replay for HTTP. In *Proceedings of ATC*, pages 417–429. USENIX Association, 2015.
- [NSS10a] Erik Nygren, Ramesh K. Sitaraman, and Jennifer Sun. The akamai network: A platform for high-performance internet applications. *SIGOPS Oper. Syst. Rev.*, 2010.
- [NSS10b] Erik Nygren, Ramesh K Sitaraman, and Jennifer Sun. The Akamai Network: A Platform for High-Performance Internet Applications. *ACM SIGOPS Oper. Syst. Rev.*, 44(3):2–19, 2010.
- [NTT20] NTT Labs. BGP implemented in the Go Programming Language. <https://github.com/osrg/gobgp>, 2020.
- [OKG⁺16] Chiara Orsini, Alistair King, Danilo Giordano, Vasileios Giotsas, and Alberto Dainotti. Bgpstream: a software framework for live and historical bgp data analysis. In *Proceedings of the 2016 Internet Measurement Conference*, pages 429–444, 2016.
- [OSRB12] John S. Otto, Mario A. Sánchez, John P. Rula, and Fabián E. Bustamante. Content delivery and the natural evolution of dns:

- Remote dns trends, performance issues and alternative solutions. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'12)*, 2012.
- [Pax96] Vern Paxson. End-to-end Routing Behavior in the Internet. *SIGCOMM Computer Communication Review*, 26(4):25–38, 1996.
- [Peta] Aaron Peters. CDN Finder - CDN Planet. Retrieved in Sep, 2022 from <https://www.cdnplanet.com/tools/cdnfinder/>.
- [Petb] Aaron Peters. CDNPlanet - EDNS Client Subnet Checker. Retrieved in Sep, 2022 from <https://www.cdnplanet.com/tools/edns-client-subnet-checker/>.
- [Pla] CDN Planet. Overview of Content Delivery Networks. Retrieved in Sep, 2022 from <https://www.cdnplanet.com/cdns/>.
- [PMM93] Craig Partridge, Trevor Mendez, and Walter Milliken. Host any-casting service. RFC 1546, RFC Editor, 11 1993.
- [Pos81] Jon Postel. Internet Control Message Protocol. RFC 777, RFC Editor, April 1981.
- [Pro] ProxyRack. Retrieved in Sep, 2022 from <https://www.proxyrack.com>.
- [PS01] Venkata N. Padmanabhan and Lakshminarayanan Subramanian. An Investigation of Geographic Mapping Techniques for Internet Hosts. In *Proceedings of the Annual Conference of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'01)*, 2001.
- [PUK⁺11a] I. Poese, S. Uhlig, M. A. Kaafar, B. Donnet, and B. Gueye. IP Geolocation Databases: Unreliable? *ACM SIGCOMM Comput. Commun. Rev.*, 41(2):53–56, 2011.
- [PUK⁺11b] Ingmar Poese, Steve Uhlig, Mohamed Ali Kaafar, Benoit Donnet, and Bamba Gueye. IP Geolocation Databases: Unreliable? *ACM SIGCOMM Comput. Commun. Rev.*, 41(2):53–56, 2011.
- [PVGT⁺19] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In *Proceedings of NDSS*. ISOC, 2019.

- [RBCH22] A S M Rizvi, Leandro Bertholdo, João Ceron, and John Heidemann. Anycast Agility: Network Playbooks to Fight DDoS. In *Proceedings of 31st USENIX Security Symposium (USENIX Security'22)*, 2022.
- [RFP+12] Robert Raszuk, Rex Fernando, Keyur Patel, Danny R. McPherson, and Kenji Kumaki. Distribution of Diverse BGP Paths. RFC 6774, RFC Editor, 2012.
- [(RI] Reseaux IP Europeens (RIPE). API Reference. Retrieved in Sep, 2022 from <https://atlas.ripe.net/docs/api/v2/reference/#!/probes/Type>.
- [RIP23] RIPE. RIPE Built-in Measurements. <https://atlas.ripe.net/docs/built-in-measurements/>, 2023.
- [RL94a] Yakov Rekhter and Tony Li. A Border Gateway Protocol 4 (BGP-4). RFC 4271, RFC Editor, Jan 1994.
- [RL94b] Yakov Rekhter and Tony Li. A Border Gateway Protocol 4 (BGP-4). RFC 4271, RFC Editor, Jan 1994.
- [RLH06] Yakov Rekhter, Tony Li, and Susan Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271, RFC Editor, January 2006.
- [RLM+05] Scott Rose, Matt Larson, Dan Massey, Rob Austein, and Roy Arends. DNS Security Introduction and Requirements. RFC 4033, RFC Editor, Mar 2005.
- [RLP+21] Audrey Randall, Enze Liu, Ramakrishna Padmanabhan, Gautam Akiwate, Geoffrey M. Voelker, Stefan Savage, and Aaron Schulman. Home is Where the Hijacking is: Understanding DNS Interception by Residential Routers. In *Proceedings of IMC*, pages 390–397. ACM, 2021.
- [roo] Root Server Technical Operations Association. Retrieved in Sep, 2022 from <https://root-servers.org/>.
- [roo23] root-servers.org, 2023.
- [Roua] RouteViews. Retrieved in Sep, 2022 from <http://www.routeviews.org/routeviews/>.

- [Roub] RouteView. BGP RIB Archive [Aug-30-2022]. Retrieved in Feb, 2023 from <ftp://archive.routeviews.org/bgpdata/2022.08/RIBS/rib.20220830.0000.bz2>.
- [RWXZ02] Jennifer Rexford, Jia Wang, Zhen Xiao, and Yin Zhang. BGP Routing Stability of Popular Destinations. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement*, pages 197–202. ACM, 2002.
- [SACKB19] Brandon Schlinker, Todd Arnold, Italo Cunha, and Ethan Katz-Bassett. PEERING: Virtualizing BGP at the Edge for Research. In *Proceedings of the 15th ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT '19)*, 2019.
- [SAD20a] Kyle Schomp and Rami Al-Dalky. Partitioning the Internet Using Anycast Catchments. *ACM SIGCOMM Comput. Commun. Rev.*, 50(4):3–9, 2020.
- [SAD20b] Kyle Schomp and Rami Al-Dalky. Partitioning the internet using anycast catchments. *SIGCOMM Comput. Commun. Rev.*, 50(4):3–9, oct 2020.
- [SAD20c] Kyle Schomp and Rami Al-Dalky. Partitioning the Internet Using Anycast Catchments. *ACM SIGCOMM Comput. Commun. Rev.*, 50(4):3–9, 2020.
- [San05] F. Sanford. Oxidized ferrous materials. *Journal of Junk*, 5(4):324–345, 2005.
- [San18] Conor Sanchez. Does Establishing More IXPs Keep Data Local? Brazil and Mexico Might Offer Answers, 2018.
- [SBA⁺20a] R. Sommesse, L. Bertholdo, G. Akiwate, M. Jonker, R. van Rijswijk-Deij, A. Dainotti, k. claffy, and A. Sperotto. MAnycast² - Using Anycast to Measure Anycast. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'20)*, 2020.
- [SBA⁺20b] Raffaele Sommesse, Leandro Bertholdo, Gautam Akiwate, Mattijs Jonker, Roland van Rijswijk-Deij, Alberto Dainotti, KC Claffy, and Anna Sperotto. Manycast2: Using anycast to measure anycast. In *Proceedings of the ACM Internet Measurement Conference*, pages 456–463, 2020.

- [SBA⁺20c] Raffaele Sommesse, Leandro Bertholdo, Gautam Akiwate, Mattijs Jonker, Roland van Rijswijk-Deij, Alberto Dainotti, KC Claffy, and Anna Sperotto. MANycast2: Using Anycast to Measure Anycast. In *Proceedings of the Annual Conference of the ACM Internet Measurement Conference (IMC'20)*, 2020.
- [SBK⁺20a] Kyle Schomp, Onkar Bhardwaj, Eymen Kurdoglu, Mashooq Muhaimen, and Ramesh K. Sitaraman. Akamai dns: Providing authoritative answers to the world's queries. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication, SIGCOMM '20*, page 465–478, New York, NY, USA, 2020. Association for Computing Machinery.
- [SBK⁺20b] Kyle Schomp, Onkar Bhardwaj, Eymen Kurdoglu, Mashooq Muhaimen, and Ramesh K. Sitaraman. Akamai DNS: Providing Authoritative Answers to the World's Queries. In *Proceedings of the Annual Conference of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'20)*, 2020.
- [SBWR21] Constantin Sander, Leo Blöcher, Klaus Wehrle, and Jan Rüth. Sharding and HTTP/2 Connection Reuse Revisited: Why Are There Still Redundant Connections? In *Proceedings of IMC*, pages 292–301, 2021.
- [SCC⁺19] Brandon Schlinker, Italo Cunha, Yi-Ching Chiu, Srikanth Sundaresan, and Ethan Katz-Bassett. Internet performance from facebook's edge. In *Proceedings of the Internet Measurement Conference, IMC '19*, page 179–194, New York, NY, USA, 2019. Association for Computing Machinery.
- [SGU08] S. S. Siwpersad, Bamba Gueye, and Steve Uhlig. Assessing the Geographic Resolution of Exhaustive Tabulation for Geolocating Internet Hosts. In *Proceedings of Passive and Active Measurement (PAM'08)*, 2008.
- [SH70] M. Smith and I. A. Hall. *Handbook of Interstellar Travel*. Dover, New York, 7th edition, 1970.
- [SK19] Pavlos Sermpezis and Vasileios Kotronis. Inferring catchment in internet routing. *Proc. ACM Meas. Anal. Comput. Syst.*, 3(2), June 2019.

- [SKC⁺17a] Brandon Schlinker, Hyojeong Kim, Timothy Cui, Ethan Katz-Bassett, Harsha V. Madhyastha, Italo Cunha, James Quinn, Saif Hasan, Petr Lapukhov, and Hongyi Zeng. Engineering egress with edge fabric: Steering oceans of content to the world. In *Proceedings of the Annual Conference of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'17)*, 2017.
- [SKC⁺17b] Brandon Schlinker, Hyojeong Kim, Timothy Cui, Ethan Katz-Bassett, Harsha V Madhyastha, Italo Cunha, James Quinn, Saif Hasan, Petr Lapukhov, and Hongyi Zeng. Engineering egress with edge fabric: Steering oceans of content to the world. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 418–431, 2017.
- [SLD11] Eric Sven-Johan Swildens, Zaide Liu, and Richard David Day. Global traffic management system using ip anycast routing and dynamic load-balancing. US Patent 7,904,541, 2011.
- [SMA03] Neil Spring, Ratul Mahajan, and Thomas Anderson. The causes of path inflation. In *Proceedings of the Annual Conference of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'03)*, 2003.
- [SMW02] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring isp topologies with rocketfuel. In *Proceedings of the Annual Conference of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'02)*, 2002.
- [SPT05] Sandeep Sarat, Vasileios Pappas, and Andreas Terzis. On the use of anycast in dns. In *Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '05, page 394–395, New York, NY, USA, 2005. Association for Computing Machinery.
- [SPT06] Sandeep Sarat, Vasileios Pappas, and Andreas Terzis. On the Use of Anycast in DNS. In *Proceedings of the 2005 ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'06)*, 2006.
- [STA01] A. Shaikh, R. Tewari, and M. Agrawal. On the effectiveness of dns-based server selection. In *Proceedings of the Annual Conference of the IEEE International Conference on Computer Communications (INFOCOM'01)*, 2001.

- [Sta15a] RIPE NCC Staff. Ripe atlas: A global internet measurement network. *Internet Protocol Journal*, 18(3), 2015.
- [Sta15b] RIPE NCC Staff. RIPE Atlas: A Global Internet Measurement Network. *Internet Protocol Journal*, 18, 2015.
- [SZ11a] Y. Shavitt and N. Zilberman. A Geolocation Databases Study. *IEEE Journal on Selected Areas in Communications*, 29(10):2044–2056, 2011.
- [SZ11b] Yuval Shavitt and Noa Zilberman. A Geolocation Databases Study. *IEEE Journal on Selected Areas in Communications*, 29(10):2044–2056, 2011.
- [SZC⁺14] Brandon Schlinker, Kyriakos Zarifis, Italo Cunha, Nick Feamster, and Ethan Katz-Bassett. Peering: An AS for US. In *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*, 2014.
- [Tec] Akamai Technologies. EdgeScape. Retrieved in Sep, 2022 from <https://developer.akamai.com/edgescape>.
- [Tec20] Akamai Technologies. Prolexic Routed, 2020.
- [Tel] TeleGeography. Submarine Cable Map. Retrieved in Feb, 2023 from <https://www.submarinecablemap.com/>.
- [VCG98] Curtis Villamizar, Ravi Chandra, and Dr. Ramesh Govindan. BGP Route Flap Damping. RFC 2439, November 1998.
- [vdM] Achiel van der Mandele. Introducing Regional Services. Retrieved in Sep, 2022 from <https://blog.cloudflare.com/introducing-regional-services/>.
- [Ver] Interconnection Policy for Internet Networks. <https://www.verizon.com/business/terms/peering/>.
- [VK10] IL Vasilyev and KB Klimentova. The branch and cut method for the facility location problem with client’s preferences. *Journal of Applied and Industrial Mathematics*, 4(3):441–454, 2010.
- [VKK09] IL Vasil’ev, KB Klimentova, and Yu A Kochetov. New lower bounds for the facility location problem with clients’ preferences. *Computational Mathematics and Mathematical Physics*, 49(6):1010–1020, 2009.

- [VRB⁺20] Kevin Vermeulen, Justin P. Rohrer, Robert Beverly, Olivier Fourmaux, and Timur Friedman. Diamond-Miner: Comprehensive Discovery of the Internet’s Topology Diamonds. In *Proceedings of NSDI*, pages 479–493. USENIX Association, 2020.
- [WCD⁺18] Florian Wohlfart, Nikolaos Chatzis, Caglar Dabanoglu, Georg Carle, and Walter Willinger. Leveraging interconnections for performance: The serving infrastructure of a large cdn. In *Proceedings of the Annual Conference of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM’18)*, 2018.
- [WH17a] Lan Wei and John Heidemann. Does anycast hang up on you? In *2017 Network Traffic Measurement and Analysis Conference (TMA)*, pages 1–9, 2017.
- [WH17b] Lan Wei and John Heidemann. Does Anycast Hang Up on You? In *Proceedings of the Network Traffic Measurement and Analysis Conference (TMA’17)*, 2017.
- [WH18a] Lan Wei and John Heidemann. Does Anycast Hang up on You (UDP and TCP)? *IEEE Transactions on Network and Service Management (TNSM)*, 15(2):707–717, 2018.
- [WH18b] Lan Wei and John Heidemann. Does Anycast Hang Up on You (UDP and TCP)? *IEEE Transactions on Network and Service Management*, 15(2):707–717, 2018.
- [WS07] Bernard Wong and Ivan Stoyanov. Octant: A Comprehensive Framework for the Geolocalization of Internet Hosts. In *Proceedings of 4th USENIX Symposium on Networked Systems Design & Implementation (NSDI’07)*, 2007.
- [Yoa19] Einav Yoav. Amazon found every 100ms of latency cost them 1% in sales, jan 2019.
- [ZSZ⁺21] Xiao Zhang, Tanmoy Sen, Zheyuan Zhang, Tim April, Balakrishnan Chandrasekaran, David Choffnes, Bruce M. Maggs, Haiying Shen, Ramesh K. Sitaraman, and Xiaowei Yang. AnyOpt: Predicting and Optimizing IP Anycast Performance. In *Proceedings of the Annual Conference of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM’21)*, 2021.

- [ZZH⁺23] Minyuan Zhou, Xiao Zhang, Shuai Hao, Xiaowei Yang, Jiaqi Zheng, Guihai Chen, and Wanchun Dou. Regional IP Anycast: Deployments, Performance, and Potentials. In *Proceedings of SIGCOMM*. ACM, 2023.

Biography

Xiao (Shane) ZHANG, is a Ph.D. candidate in Computer Science at Duke University since August 2018. He is co-advised by Professor Bruce M. Maggs and Professor Xiaowei Yang. He also works with Akamai Technologies on anycast optimization. Before that, he received an M.Eng. degree in computer science at Xi'an Jiaotong University in 2017 and a B.S. degree in computer science at Xi'an Jiaotong University in 2013. He visited The Chinese University of Hong Kong as an exchange student from Jan 2012 to Jun 2012. He received the 2016-2017 IBM Fellowship award in Jan 2016 and visited IBM Research - China as a research assistant intern from July 2015 to July 2018.