Dissertation Proposal

Thomas Koch
Columbia University

1 INTRODUCTION

As network applications become a more vital, integral part of our lives, it's increasingly important for those applications to be available and performant. For example, enterprises regularly use cloud services to manage product databases, collaborate in real time on product designs, and conduct meetings. However, as our demands on the network evolve, its protocols remain dangerously stagnant, and cloud/content providers (CPs) are realizing protocol limitations. A particular limitation CPs face is that they cannot choose the paths taken by users to services, hindering the speed/granularity at which they can respond to network problems, the range of solutions they can provide, and overall deployment resilience.

Specifically, BGP (the protocol that determines Internet paths) limits CPs since it may pick the wrong path (routes are not chosen for performance or reliability), since a BGP router typically only selects and shares one path, and since failover between paths is slow. BGP also spreads decisions about which routes are chosen among multiple third parties along the path, making modeling and debugging difficult.

Two common approaches that CPs use to route users to their deployments are anycast and DNS. Part of my research characterized different anycast deployments, demonstrating that a broad footprint and extensive peering helps most users achieve near-optimal latency with high availability [18]. Although these approaches work well for traditional content (*e.g.*, web pages) for most users, they fail to meet latency and reliability requirements that modern applications demand, since they can only coarsely/slowly control how users are routed to deployments. For example, in a large CP 10% of users are routed more than 1,000 km further than is necessary, leading to inflated latency for millions of customers [18]. Still, anycast offers a simple, reliable means of routing users to content, which is possibly a reason why at least 12 out of 22 of the largest CPs use it in their deployments in some way today [10]. Using DNS leaves CPs with similar limitations since users can only be redirected at course granularities (per recursive resolver) and users often do not respect DNS record TTLs.

Luckily, emerging deployment patterns and technologies give CPs more control at the edge. In particular, the increased presence of applications with network-stack control [1], Apple's private relay [38], the MASQUE effort [13], cloud-edge compute [49], CDN off-nets [10], and Networking-as-a-Service enterprise solutions [14] all provide CPs with flexible, deployable solutions to enact policy on user traffic. We refer to all these technologies as proxies, since we envision using these technologies to enact policy (tunnel, modify) on traffic at the edge. In addition to these technologies, vast global footprints with hundreds of PoPs, thousands of peerings, and thousands of off-nets give CPs a rich solution space over which to optimize routes from users to services.

To overcome the limitations of current approaches to routing users to CPs, I first investigated using these edge proxies as footholds for making routing decisions and enacting policy on user traffic, while also configuring prefix advertisements from the CP's deployment to expose additional routes that optimize for latency and resilience. This approach optimizes on two key dimensions that other approaches fail to address: deployability and precision. In using proxies at the edge, this approach allows CPs to deploy software on a relatively small number of devices/implementations yet affect a large number of clients, simplifying management and increasing iteration velocity. In using modern networking devices that can perform packet operations at line rate, we made precise, agile decisions, shifting traffic at per-flow granularities to the additional prefixes we expose and reacting to changing network conditions at RTT timescales, rather than waiting for traditional decision timelines such as BGP route convergence and DNS TTL expirations.

I used this enhanced functionality to break away from simple, reliable, yet limited methods of directing users to content by finding optimal ways to advertise CP reachability to the Internet, and using proxies to enhance deployment agility. The key idea is that CPs can advertise different prefixes to their many peers/providers to expose more paths to users, offering enhanced latency and resilience. CPs can optimize announcements over thousands of peerings and hundreds of PoPs, but cannot expose all paths since doing so would pollute global BGP routing tables, giving us a challenging optimization problem to solve.

My preliminary work (PAINTER, §3) addresses this challenge by computing cost-effective advertisement strategies to improve performance that advertise prefixes to peers/providers with non-overlapping customer cones (customer cones defined in prior work [25]). The key insight is that advertising in this way does not lead to path inflation (overcoming anycast limitations), since users in those respective cones would only have one policy-compliant path to the prefix. Proxies direct traffic over different tunnels to these prefixes, enhancing deployment agility (overcoming DNS' limitations). The algorithm computes advertisements greedily, advertising prefixes via peerings in a way that provides the most marginal improvement at each step of computation.

To *fully* demonstrate how moving away from simple, static advertisement schemes towards richer advertisement frameworks allows CPs to build faster, more resilient deployments, I next propose a richer model/optimization framework to find even better strategies of advertising destinations to the Internet (§4). The key insights that enable the proposed work are that (a) it can be much easier to predict latency than to predict paths, allowing us to optimize over unseen strategies, and (b) explicitly optimizing with respect to both advertisement cost *and* benefit can lead to more globally optimal solutions than greedily optimizing solely over benefit.

Our proposed framework alternatively explores (measures) and exploits (optimizes) over the space of advertisements, iteratively building a model of user routes to deployments and optimizing the CP's advertisement strategy. Preliminary results suggest that this framework outperforms both PAINTER and other strategies (§4.4).

The model of user paths to CPs that we refine during announcement computation allows us to answer "what-if" questions about the deployment; hence, our significantly more complex framework can provide better understanding about user routes to the deployment than simple anycast configurations. Both our preliminary work and proposed richer framework naturally point towards the following statement, which my work holistically demonstrates.

Thesis Statement. Using increasingly rich, complex Internet path exposure frameworks and increasingly fine-grained traffic control technology allows us to better optimize and understand routes to service provider deployments.

This proposal thus motivates and characterizes increasingly complex frameworks with richer spaces over which to expose destinations and increasingly realistic, useful routing models to build better deployments. Moving from a pure-anycast configuration with relatively static routes to a deployment with edge proxies dynamically measuring performance and directing user traffic along best paths, and then further to a global system that optimizes how destinations are advertised to improve performance, brings us far away from the simple "one and done" anycast solution.

Enhanced system complexity may make CPs hesitate, which we hope to combat with enhanced *understanding* of how users are routed to deployments. Regardless, in the face of increasingly essential, demanding uses of the Internet, the complexity of this proposal represents an inevitable eventuality that we should embrace should we wish to rely on the Internet to support increasingly essential, amazing applications.

2 RELATED WORK

Optimizing Interdomain Routing. Studies have investigated anycast's suboptimalities [4, 6, 22, 28, 47, 50] and DNS [5, 9, 24] deployments. Our preliminary work [18] builds on studies investigating anycast performance in particular, clearing up discrepancies in our understanding of anycast performance and why those discrepancies existed. Our preliminary and proposed work improves over both these methods of routing users to deployments.

As these technologys' suboptimalities are widely acknowledged, recent work investigated how they might be improved using a combination of DNS and anycast [4, 8], complex advertisement strategies that utilize BGP "tricks" [36, 54], overlapping anycast advertisements [52], and advertising different prefixes from CPs to expose more paths [2, 46]. Our work builds on these studies, using a richer framework to provide lower latency, more resilient deployments than the proposed solutions at lower cost to CPs.

Some studies have proposed frameworks for global end-to-end (as opposed to our focus—the path from users to the deployment) path optimization [24, 32]. Key differences between our framework and these deployments are that we (a) optimize over a richer space of solutions by considering modern deployment's thousands of peerings and global presence, and (b) iteratively optimize over this space and build a model of user routes to the deployment. Those studies optimized deployments by tailoring DNS records; our preliminary work considers how our approaches improve over DNS (via fine-grained traffic control).

A recent survey of key advancements and remaining challenges in CP traffic engineering identified a key area for research as using new edge proxies to facilitate more-optimal interdomain routing [44]. Both our preliminary and proposed work use this emerging deployment model to enable our solutions. Prior work uses this deployment model to optimize routing, but assumes no collaboration between the edge and the CP [53], which we leverage in our solution. Other work points to/uses other forms of collaboration with the edge to enable interdomain routing optimization solutions [19, 32, 34], but it is unclear whether those solutions are scalable. Our proposed optimization framework requires little/no cooperation with edge networks, and works in whatever network contains proxies.

Cloud providers have deployed systems that optimize routes from deployments to users (*i.e.,* egress) routes [11, 20, 40, 51]. Our proposed work optimizes routes from users to deployments (*i.e.,* focusing on the ingress), and so works alongside those systems.

Modeling Routes and Performance to CP Deployments. Verfploeter used an anycast deployment to measure anycast catchments and ingress latencies without instrumenting end users to issue measurements [7], and we adopt this method to measure user latencies to the PEERING testbed [39] to test our deployment strategies. Other work tries to predict catchments in unseen circumstances using both BGP models [42], pairwise preferences between ingresses [52], and historical data [27]. Our proposed framework similarly tries to determine pairwise preferences, but only when it is important for achieving deployment objectives (and so is much more scalable). Other work tried to shift traffic to a particular site using BGP "tricks" [36, 54]. Our framework offers many paths to proxies, which can dynamically move traffic to different ingresses.

WISE is a system that tries to predict user latencies when adding new PoPs and peers [45]. It is unclear how well WISE would translate to other settings since the study is old (deployments have grown considerably since then), and was specific to a particular deployment; our proposed framework to estimate latency works for all deployments.

Optimization in Next-Generation Networks. By "proxies" we refer to the increased number of applications with network-stack control [1], Apple's private relay [38], the MASQUE effort [13], cloud-edge compute [49], off-nets [10], and Networking-as-a-Service enterprise solutions [14] since all these emerging technologies provide CPs with flexible, deployable solutions to enact policy on user traffic. Using the emerging proxy-deployment-pattern as a tool to optimize deployments is similar to a collaborative push in 5G networks between mobile providers and CPs.

A recent measurement study characterized how widespread edge-cloud deployments already are, and how these deployments could be used to enhance application performance [49]. Part of my work argues that cloud services will inevitably use these deployments to enact policy on user traffic (as opposed to waiting for widespread deployment of other, standardized approaches like MPTCP [12, 35]).

DChannel [41] and Zhuge [29] demonstrate the potential utility of these edge proxies from an end-user perspective by showing that their compute can be used to improve user performance in wireless networks. Our framework would operate alongside such technologies, using the emerging technologies / 5G deployment patterns to significantly enhance network experience.

Tango [2] proposes a similar framework to ours (advertising different prefixes in different ways to expose more paths), but assumes network edges will cooperatively implement this mechanism in a well-understood protocol (we assume CPs operate proxies). Tango also does not propose how to optimally expose destinations, which is a key contribution of this proposal.

3 PRELIMINARY WORK

My preliminary work towards optimizing deployments consists of first characterizing their performance and understanding their inefficiencies (§3.1), and second proposing how to improve them (§3.2). My proposal to improve deployments involves (a) identifying a concrete optimization problem to solve, (b) solving the Internet routing/measurement/modeling challenges surrounding that optimization problem, and (c) developing an algorithm for (possibly approximately) solving the optimization problem. In solving the optimization problem, we arrive at an advertisement *strategy–i.e.,* a way of advertising Internet prefixes to peers/providers.

Given this clearer view of the specific problem to solve, my proposed work (§4) largely consists of an improved means of (b) and (c) above, modeling and solving the optimization problem.

3.1 Characterizing Anycast Deployment Performance and Inefficiency

Before improving upon routes from users to content, I first characterized what performance a popular advertisement strategy, anycast, gave to users. Performance and limitations when using DNS redirection (another popular routing technique) are already well understood [3, 5, 9, 26, 31].

The large body of prior work characterizing anycast left a mixed message, with some saying it provided great performance for users [4, 16, 21, 30] while others pointing out that it commonly inflated paths from users to deployments [22, 23, 37]. My work [18] elucidated how different deployment strategies and content offerings could lead to drastically different perceptions of anycast. By comparing two deployments—the root DNS and Microsoft's CDN—we painted a decent "overall" picture of anycast. Most users were not inflated at all and a small fraction experienced significant inflation, especially when accounting for local resource caching.

However, the work also revealed that anycast, despite its simplicity, can route users poorly since BGP (which determines interdomain paths to anycast addresses) is not performance-aware. Users are routed to root DNS deployments poorly since root DNS operators lack incentives to spend time optimizing user routes. For example, instead of carefully selecting who to peer with/advertise routes to (like Microsoft), root DNS deployments often have an "open" peering policy.

Figure 1 shows that Microsoft's highly engineered anycast deployment maps 90% of users to within 1,000 km (10 ms in fiber) of their closest PoP, whereas the same is true for only 70% of users when reaching the root DNS. Figure 1 shows distance over the shortest PoP users travel to reach different deployments scaled to the speed of light in fiber, where R28-110 refer to different sub-deployments of Microsoft, and root DNS refers to the average inflation experienced over all root DNS letter deployments. Microsoft spends more time fixing poor routes than the root DNS since latency matters much more *on average* for Microsoft services than for root DNS records which are highly cacheable.

Hence, this preliminary work demonstrates anycast provides decent latency, but increasingly strict application requirements demand better than "decent". Manual optimization (largely what Microsoft does today) is not a long-term

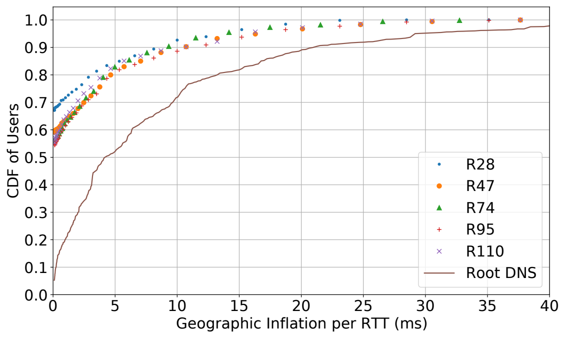

Figure 1: While anycast can lead to near-optimal performance for most users in some deployments such as Microsoft's, anycast routes can be poor if not optimized such as in the root DNS. Operators must have incentives to fix poor routes and optimize deployments for latency.

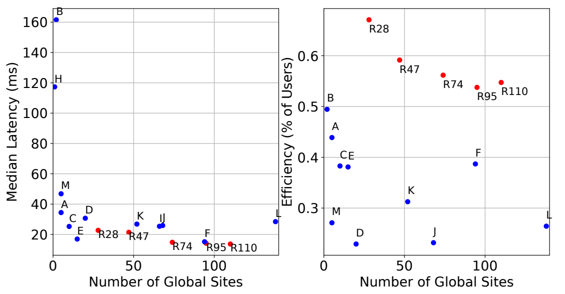

Figure 2: Adding PoPs offers more low-latency options for users, but resulting deployments are often less "efficient", meaning anycast leaves plenty of room to optimize further.

solution to performance problems since deployments continue to grow, and since problems often occur in the administrative domain of CPs. Hence CPs would find an automated, effective means of fixing "problems" invaluable.

A key source of the confusion this work cleared up is that offering richer spaces over which to optimize deployments (more PoPs, more peerings) makes anycast deployments less efficient (i.e., inflation is more common) but holistically improves deployments since user latency decreases and resilience/capacity increases. Figure 2 demonstrates that as deployment size increases, latency decreases and efficiency increases for both root DNS deployments (letters) and Microsoft deployments (R28-R110). From an optimization perspective, larger deployments can therefore be further from a globally optimal solution, but often provide a globally lower objective/cost function value than for smaller deployments. Hence, in addition to identifying the need to optimize deployments, this preliminary work identified extensive *opportunity* to optimize deployments over an increasingly rich space.

3.2 Exposing Paths to Optimize Deployments

Using the vast set of PoPs and peerings available to modern CPs, I then proposed an algorithm/system, PAINTER, that

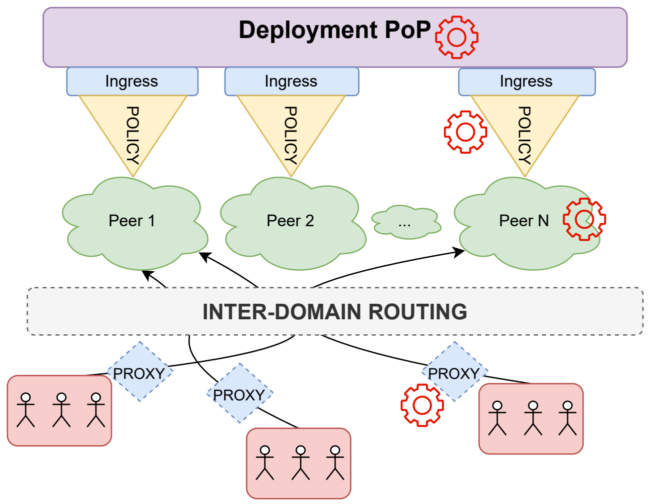

Figure 3: CPs have global footprints with hundreds of PoPs, thousands of peering sessions, and (increasingly) tens of thousands of edge proxies capable of enacting policy on user traffic. CPs have a powerful set of control knobs (gears) with which to optimize user routes.

uses the proxy deployment model to efficiently expose and direct traffic over new paths that improved user latency and deployment resilience. In designing PAINTER, I framed the problem of optimizing deployments as one of maximizing benefit over all possible "advertisement strategies", where a strategy broadly refers to how prefixes are advertised via PoPs/peerings. Figure 3 illustrates this optimization space. Gears indicate the deployment optimization "knobs' ' CPs have at their disposal.

The PoP knob in Figure 3 refers to flexibility for CPs to deploy CP entry-points in new cities/regions. The Peer knob refers to the vast set of networks CPs can establish BGP sessions with. The Proxy knob represents CPs decisions to deploy proxies in particular networks, and run particular path-selection/traffic-redirection/measurement methodologies on these flexible proxies. Finally, the Policy knob (which we concern ourselves with) refers to the fact that different prefixes can be advertised with different BGP policies to CP peers/providers.

PAINTER greedily (as opposed to exhaustively) optimizes over the space of advertisement strategies (Policy knob in Figure 3 since there are exponentially many advertisement strategies to optimize over, measuring effects of advertisement strategies takes several minutes per strategy, and it is difficult to predict/model how a strategy affects performance before measuring it. CPs then advertise computed strategies to peers/providers. A component of PAINTER partially situated in proxies then tunnels user traffic to their best-performing prefix; thus PAINTER improves user latency by exposing more low-latency paths, increases deployment resilience by exposing backup paths, and enhances deployment agility since edge proxies react to changing network conditions at RTT-timescales.

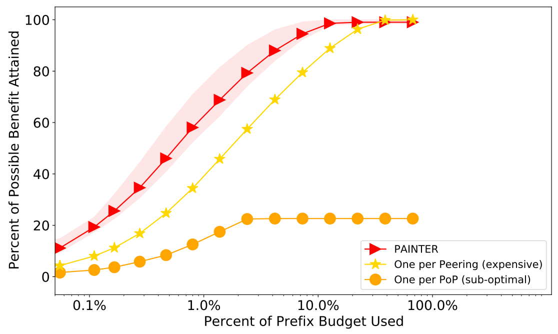

Figure 4: Our initial algorithm, PAINTER, greedily allocates prefixes to peerings gives users lower latency paths to the deployment at a lower prefix cost than alternate solutions. Error bars around PAINTER correspond to performance uncertainty.

PAINTER advertises the same prefix across many peerings across possibly different PoPs to simultaneously expose new paths for users while keeping BGP footprint/prefix costs small. Hence, the entire advertisement strategy can be viewed as a set of anycast advertisements. PAINTER advertises the same prefix to different peers/providers (*i.e.,* reuses it) essentially only when those peers/providers have minimally overlapping customer cones. The key insight here is that advertising in this way does not lead to path inflation, since users in those respective cones only have one policy-compliant path to the prefix and we only advertise prefixes when the resulting exposed paths are low-latency. (Cone overlap is allowed if the potential inflation penalty is small.)

Figure 4 demonstrates that PAINTER exposes paths more efficiently than other strategies such as an expensive, exhaustive approach (One per Peering which exposes all paths) and a simple approach used in prior work (One per PoP which leads to lots of inflation). Users see the same latency benefit over anycast (y-axis) as the exhaustive, expensive approach using a third of the prefix budget (cost).

Limitations PAINTER, however, missed out on a few key opportunities for optimizing over the space. First, the algorithm made naive predictions during optimization about routes users would take, missing out on chances to learn from incorrect assumptions during optimization. Specifically, if a user had more than one policy-compliant path to a prefix, PAINTER assumed the user's latency would be the average across policy-compliant options (in reality BGP would choose one of those paths). Second, PAINTER greedily optimized over the space of solutions, potentially sacrificing global optimality for short-term gains. Third, PAINTER provided no enhanced understanding of how users are routed to deployments—determining how component failures, withdrawn advertisements, or canceled peering contracts affect deployment metrics is just as difficult, if not more difficult, using PAINTER as it is for anycast.

4 PROPOSED FRAMEWORK / INNOVATIONS

We propose a richer framework for optimizing over advertisement strategies that overcomes the limitations of current approaches (PAINTER, other proposed work (§2)). The new framework computes how we should expose paths for our proxies so as to improve performance and reliability, and enhances our understanding of deployments in unseen situations. The framework comes with increased complexity, however, especially compared to simple schemes for exposing paths such as anycast. Hence, our evaluations explicitly identify performance and resilience benefits over these less-complex solutions (§5).

I overcome the aforementioned limitations of PAINTER (the next-best approach) by (a) reframing the optimization problem as an explicit, differentiable tradeoff between cost and benefits, (b) building a model of how different advertisement strategies translate to user performance, (c) using this model to optimize over strategies *without* explicitly measuring all strategies, and (d) improving this model over time by intelligently issuing a small number of advertisements that yield the most useful measurements.

4.1 Framework Overview

CPs expose paths to users by advertising prefixes via their many peerings, so we frame the problem of optimizing routes to deployments as optimizing over the space of possible advertisement strategies. Generally, exposing more paths yields more benefit for users, but comes at increased cost to the CP (so we cannot expose all paths), so the challenge is to find the best subset of paths to expose. Even this general problem only utilizes *one* of the control knobs in Figure 3—that of modifying policy. Other variables in Figure 3 we do not explicitly optimize are: where to establish PoPs, who to peer with, and where to place proxies/other edge infrastructure. Optimizing these knobs could be the subject of future work (§6).

To model an advertisement strategy, let the matrix a represent an advertisement from the CP with $a_{ij} = 1$ if we advertise prefix j to peer i and 0 otherwise. Modeling an advertisement strategy in this way is a simplification of potential policies—a key challenge I will address is extending the model to handle more complex schemes.

To find an optimal strategy, we aim to minimize the following function with respect to a

$$F(a) = \lambda C(a) - B(a) \tag{1}$$

where C represents advertisement cost, B represents advertisement benefit, and $\lambda > 0$ is a parameter modeling the tradeoff between our desire for cost effective, yet beneficial advertisements. In what follows we discuss how we model these functions and how we optimize over possible advertisement strategies.

4.2 Modeling Cost and Benefit

4.2.1 Modeling Cost (C)

We consider three costs associated with an advertisement: the cost of engaging in a peering agreement, the BGP routing table pollution cost of advertising a new prefix, and the cost of a prefix. Establishing peerings may cost the CP money and, although it may be a complicated function of traffic volume or type of data exchanged, we assign a static cost to each peering i $c_i^{peering}$. If we do not advertise any prefixes via peering i then our cost for that peering is essentially zero, hence the cost of of the i^{th} peering can be modeled as

$$c_i^{peer} \mathbb{1}_{\sum_j a_{ij} > 0} \tag{2}$$

and so the total cost over peerings is

$$\sum_i c_i^{peer} \mathbb{1}_{\sum_j a_{ij} > 0} \tag{3}$$

Similarly, we can model the cost to advertise a prefix as

$$c^{prefix} \mathbb{1}_{\sum_i a_{ij} > 0} \tag{4}$$

where c^{prefix} is the cost per prefix advertisement (buying an address block, and polluting BGP routing tables), which does not depend on j. The prefix cost may depend on i if advertisements to different peers/providers lead to different BGP routing table impacts (we do not consider this case). Therefore, the entire cost of an advertisement can be modeled as a sum

$$C(a) = \sum_i c_i^{peer} \mathbb{1}_{\sum_j a_{ij} > 0} + c^{prefix} \sum_j \mathbb{1}_{\sum_i a_{ij} > 0} \tag{5}$$

This objective function can be written as a sum of L0 norms (where the L0 norm is the number of non-zero entries in a vector). Specifically, Equation (5) is equivalent to

$$C(a) = \sum_i c_i^{peer} || \sum_j a_{ij} ||_0 + \sum_j c^{prefix} || \sum_i a_{ij} ||_0 \tag{6}$$

It is useful to identify cost as a sum of L0 norms, since minimizing such objective functions is a well-studied subject with theoretical convergence guarantees [48]. It is difficult to explicitly optimize objective functions with L0 norms as they are highly non-convex so a common approach is to instead optimize with respect to the L1 norm, since doing so also (in practice) enforces sparse solutions. Replacing L0 with L1 norms in Equation (6) and combining the summations yields the following cost function

$$C(a) = \sum_{ij} (c_i^{peer} + c^{prefix}) |a_{ij}| \tag{7}$$

Although Equation (7) essentially counts the total weighted number of advertisements, rather than total prefix and peering costs, preliminary results suggest that the good convergence properties of the (convex) L1 norm lead our framework to find low advertisement cost solutions. Experiments with non-convex, more accurate objective functions yielded poor results.

4.2.2 Modeling Benefit (B)

Assumptions. A single advertisement provides benefit to users since it gives users a path to the deployment, allowing users to access CP resources. Different paths may provide additional benefit if, for example, they are lower latency or offer some form of resilience to failure. Different paths can be exposed to users by advertising multiple prefixes via different peerings.

In what follows we assume that the benefit is the mean benefit across users, and that the benefit per user can be modeled as a function of the path that user takes to the CP and the advertisement strategy (*e.g.,* latency). We also assume the user has at most one path to the CP via a given peering, this path does not change over the time scale of interest, and that user paths to CPs via peerings are only functions of advertisements to *that peer/provider* (this final assumption may not be true in general [15, 52], we discuss further in Section 6). Finally, we assume that if a user can select from many prefixes to the CP, it will select the prefix that provides the most benefit (via traffic redirection at proxies).

Predicting Ingresses. Benefit is a function of paths from users to the CP, but measuring paths for all advertisement strategies is infeasible since there are exponentially many strategies. Instead, we predict the peering via which users *ingress* the CP's deployment for different advertisement strategies and model benefit as distribution of possible benefits over ingress likelihoods. Even given perfect knowledge, this model is a potentially coarse approximation since a user may have many paths to the CP via an ingress. We only consider policy-compliant ingresses for each user, meaning those corresponding peerings each user can reach according to routing policy.

Given an advertisement strategy, it is difficult to predict where users ingress the deployment since paths depend on BGP decisions in intermediate organizations which hide information, so we model a user as having preferences among ingresses (similar to prior work [52]). To see what we mean by preference, consider the simple case where a user group has a path through two peerings to the CP, and both peerings receive the same advertisement. The BGP decision process will decide one path from a user's edge network to the prefix corresponding to one ingress over which to direct the user's traffic. We call the selected ingress the *preferred* ingress and we say that one ingress is preferred by a user over another. If a user has N policy-compliant ingresses we can perform at most $\lceil log_2(N) \rceil$ comparisons to determine all pairwise preferences among ingresses.

Performing all such comparisons over many hundreds of thousands of user networks to determine winning ingresses is infeasible, so we model preferences as a random variable and focus on gathering information about preferences when it matters for optimization. Suppose we have no prior knowledge about user preferences for ingresses. Then we initialize a distribution on ordered preferences among ingresses for each user as uniform among orderings. For example, if a user can ingress via ingresses A,B, C we would set $p(ordering) = \frac{1}{6}$ for each possible ordering. An example ordering is $P_A > P_B > P_C$ where $A > B$ means ingress A is preferred. If we have prior belief about the orderings (for example, distant ingresses are less likely than nearby ones) we can incorporate it using Bayesian methods.

When we issue an advertisement we may learn about pairwise preferences. For example, if we advertise via three peerings A, B, and C and a user takes the path through ingress B, we know the only valid orderings for that user are $B > A > C$ and $B > C > A$. Each of these orderings is then equally likely unless we have prior information about the relative ordering of A and C. A key challenge I will address is handling cases where this pairwise preference assumption breaks down (§5). Since we only consider policy-compliant ingresses for users, there are a relatively small number of orderings to consider so a relatively small number of measurements can significantly refine ingress likelihood estimates.

To predict ingresses for each user, we then calculate the probability that the path is through a particular peer as the sum of probabilities over all orderings for which that peer is the most preferred out of all peers who receive an advertisement. (That is, the conditional probability that a peer is most preferred, given the set of available peers.)

4.2.3 Calculating Benefit

Given a user's ingress, we assume the benefit $B(u, \iota(u))$ for user u and ingress $\iota(u)$ is known (*e.g.,* latency measured using Verfploeter [7]). The ingress for a prefix is a random variable specified by the aforementioned ingress likelihoods and, if a user receives advertisements for multiple prefixes, then the user selects the prefix with the maximum benefit yielding a different random variable. Although here we write benefit as dependent on one ingress, more complex benefit functions that incorporate many ingresses are also possible (for example, having two low-latency ingresses may add resilience).

The total estimated benefit distribution associated with an advertisement is the mean benefit across users, *i.e.,* $\tilde{B}_d(a) = \sum_u w(u)B(u, \iota(u))$ where w is some optional weight function. Since each user benefit is a distribution, the total benefit is also a distribution given by the convolution of all the user benefit distributions (thus the subscript d). Given a distribution on benefits (*i.e.,* the convolution over all users of the maximum benefit distribution over all prefixes), we finally calculate *the* benefit for an advertisement scheme as the *expected* benefit (*i.e.,* $\tilde{B} = \mathbb{E}(\tilde{B}_d)$). The tilde emphasizes that $\tilde{B}$ is an estimate of the true benefit, B.

4.3 Exploring and Exploiting to Optimize Deployments

Even though the benefit, $\tilde{B}$, as we have defined it is not differentiable, we can approximate $\tilde{B}$ as a differentiable function, and apply gradient descent descent methods to minimize F (Eq. (1)). This gradient descent method intuitively uses information about all "adjacent" schemes to update strategies and so slowly converge towards a more globally optimal strategy (exploitation), unlike greedy algorithms which select the immediately-next-best adjacent scheme. $\tilde{B}$ as we have defined it is also an estimate of the true benefit (B), so during optimization we intelligently issue a small number of advertisements to refine this approximation (exploration).

Smoothing Benefit. To calculate gradients, we allow the advertisement strategy a to take on any real value during optimization and threshold it at 0.5 when interpreting it as an advertisement scheme. We approximate the gradient component of B with respect to a_{ij} as the gradient of a smoothed step function. The value before the jump is $\tilde{B}$ with $a_{ij} = 0$ (denoted $\tilde{B}_{off}$) and the value after as $\tilde{B}$ with $a_{ij} = 1$ (denoted $\tilde{B}_{on}$). Specifically, we use the sigmoid approximation of the step function to obtain a smoothed $\tilde{B}_s$

$$\tilde{B}_s(a_{ij}) = \tilde{B}_{off} + \frac{(\tilde{B}_{on} - \tilde{B}_{off})}{1 + e^{-k(a_{ij}-0.5)}} \tag{8}$$

where $k > 0$ is a parameter that controls the sharpness of $\tilde{B}_s$. Even though $\tilde{B}_{off}$ and $\tilde{B}_{on}$ are themselves functions of a, we treat them as constants. Calculating gradients thus requires calculating two expected benefits to obtain $\tilde{B}_{off}$ and $\tilde{B}_{on}$ and then calculating the derivative of Equation (8).

Algorithm. Our algorithm for finding optimal advertisements is summarized in Algorithm 1. We initialize a to be essentially normally distributed about the threshold, with a slight bias towards enabling some advertisements we (a-priori) believe would help user performance. At each iteration of optimization we take a gradient step which intuitively represents the direction we should "push" our advertisement scheme to maximize benefit while not raising the cost of the advertisement too much. We also incorporate a momentum term to speed convergence.

Although we optimize continuously over advertisement schemes, only the thresholded advertisement has physical meaning. At each iteration we therefore check to see if our real (*i.e.,* thresholded) advertisement has changed; if so, we issue the new advertisement. Upon issuing an advertisement we observe how users are routed to the CP, potentially providing us with new information about ordering distributions for users (§4.2.2). For issued advertisements, $\tilde{B} = B$ since we precisely measure all user ingresses.

We also issue another advertisement at each iteration solely to refine our estimate of path likelihoods (and hence user benefits). We do not change our current optimal advertisement strategy, we only measure for the purpose of

Algorithm 1 Algorithm for calculating optimal advertisement.

```
        Input Learning rate $\alpha$, momentum parameter $\beta$,
              cost-benefit tradeoff $\lambda$
  $a \leftarrow a_{init}$
  while !$converged$ do
      $w \leftarrow a - \alpha\nabla(\lambda C(a) - B(a))$
      $a \leftarrow \beta a + (1-\beta)w$
      $check\_issue\_advertisement(threshold(a))$
      $a_{explore} \leftarrow argmax_{neighbors(a)}(potential\_benefit(n_a))$
      $check\_issue\_advertisement(a_{explore})$
  end while
```

learning ordering preferences. Specifically, at each iteration we search over *neighbors* of our current advertisement and calculate the distribution of benefits at each neighbor. We then select the strategy whose benefit distribution has maximum *bimodality* since, intuitively, large bimodality indicates strategies that are either good or bad—understanding these scenarios better could help us avoid moving into unexpectedly poorly performing regions. We use a measure of bimodality from prior work [17] which we have empirically found helps Algorithm 1 find good advertisement strategies.

At each iteration we become more certain about the expected benefit a user will experience at unmeasured advertisements strategies even if we have not issued that particular advertisement before. Thus, our approximation of the benefit ($\tilde{B}$) approaches the true benefit (B) over time. Iteration speed is limited by route flap dampening (15 minutes - 2 hours) [33, 52], as in prior work so it is important for our approximation to converge quickly.

4.4 Key Remaining Challenges

Simulations suggest the proposed framework finds better advertisement strategies that give users lower latency with lower cost than current state-of-the-art (in terms of time to compute solutions and in terms of prefixes). For example, Figure 5 demonstrates that our solution does better than PAINTER (preliminary work), and other state-of-the art solutions including both academic (Anyopt) and simple, widely-used configurations (anycast).

Although these small-scale simulations show promise, there are key remaining challenges to solve to completely develop and evaluate the framework.

Assessing Model Utility The first remaining challenge is assessing the degree to which our model of how users are routed to deployments is sufficient to optimize advertisement strategies. Our model may be incorrect, since users may not exhibit pairwise preferences between ingresses, pairwise preferences may change over time, and users may be routed to multiple ingresses due to load balancing.

Prior work [52] proposed using this pairwise preference model in a slightly different setting—the authors only considered modeling pairwise preferences among roughly 15

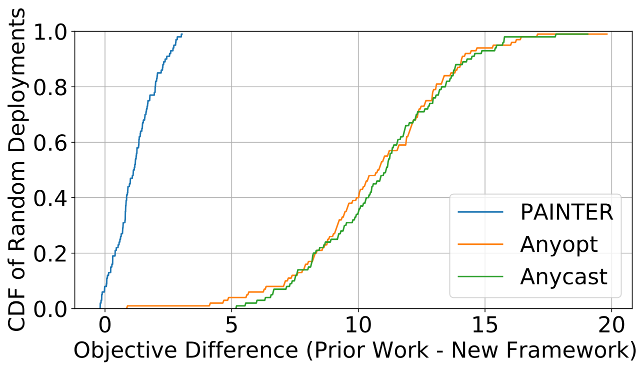

Figure 5: Difference between converged objective function value using prior methods and the proposed framework. The proposed framework shows promise—it arrives at better (lower objective) solutions over hundreds of small, simulated deployments compared to other solutions.

transit providers and found the model (a) held for more than 90% of cases and (b) was accurate enough to improve that deployment's performance, whereas our model encompasses thousands of peerings so it is unclear if their results generalize. We hope to understand the degree to which model inaccuracies affect algorithm convergence using both simulations and small scale deployments (§5). That is, we are less concerned with how often our model actually captures routing, and more concerned with whether our model is a useful tool for optimizing deployments. In simulation we will investigate, at scale, how random "violations" of the model affect convergence properties and in the actual Internet we will use the PEERING testbed [39] to test whether or not we can optimize over real Internet routes (albeit, with a small-scale, unrepresentative set of PoPs/peerings).

Scaling to Deployments Another remaining challenge is scaling the approach—CP deployments have hundreds of PoPs and thousands of peerings, and can use hundreds of prefixes to expose paths. Calculating gradients is especially computationally intensive since it requires estimating the benefit at many "adjacent" strategies, and each calculation involves estimating probability distributions over all user groups, prefixes, and ingresses. Computing advertisement strategies using our framework at this scale may require significant parallelization, or approximations/modifications—for example, we may only want to compute a subset of gradient components at each timestep, which is an approach taken in other computationally-intensive gradient descent problems [43].

Stability A third challenge is assessing framework stability. As optimization starting points are random, output advertisement strategies of our framework may be drastically different depending on initialization. It could be difficult for network operators to interpret our framework if different solutions under the same network conditions imply different values of different business (i.e., peering) agreements. Similarly, small modifications to the deployment (network performance, traffic volumes) should similarly not lead to drastically different optimal configurations. Ensuring our framework (at least empirically) converges to similar solutions may require framework modification.

Richer BGP Policies A final challenge we may address is incorporating more complex advertisement schemes that utilize advanced BGP features to expose more paths— community tagging, prepending, and/or path poisoning may give us even richer solution spaces. However, modeling and optimizing over these categorical attributes is significantly more challenging than an "on/off" advertisement model, since there is no natural continuous extension of such configurations (continuous extensions are what allow us to perform gradient descent). We hope to explore using strategies such as BGP path prepending to "shrink" advertisements into fewer prefixes, but directly incorporating complex schemes into an analytical framework is likely out of scope.

4.5 Timeline

A rough timeline for the proposed work is as follows

(1) Status to date: working simulated version of the framework at small (unrealistic) scales.
(2) End of 2022: working simulated version of the framework at a scale where it would be useful to modern CPs (thousands of peers, tens of thousands of user networks, hundreds of prefixes).
(3) Spring 2023: Key simulated evaluations, environment development for deployment on the PEERING testbed. Key challenges to address involve testing the framework in real scenarios where route flap dampening limits iteration velocity.
(4) Spring/Summer 2023: Key evaluations on the PEERING testbed. Key evaluations to complete before submission are utility of the model/framework to find good solutions, answering deployment "what-if" questions.
(5) Fall 2023: Submission to NSDI, continuing to develop more far-reaching goals: transferring to other deployments and incorporating more complicated policies.
(6) Winter 2023/Spring 2024: Plans contingent on acceptance into NSDI. Draft thesis combining all work and iterate on the proposed framework for a revise/resubmit if needed.

5 FRAMEWORK EVALUATIONS

As the purpose of our framework is to expose paths that enhance deployment latency and resilience subject to limited resources (i.e., prefixes, time), we will evaluate it on these dimensions. We will perform all evaluations using both simulation and the PEERING testbed [39], when possible. Key factors that may prevent us from evaluating in the wild are (BGP convergence) times and a limited number of prefixes (the testbed will allot us at most a few). We may additionally continue our collaboration with CPs giving us access to measurements from real users to ingresses (albeit not ground-truth ingresses for each advertisement).

Is Our Model Good (Enough)? We first will evaluate our framework's ability to provide lower latency, more resilient deployments than current approaches. We will compare our framework against other strategies including prior work [52], our preliminary algorithm (§3), and practical, static strategies such as anycast, per-PoP advertisements, and transit-only. We will consider resilience with respect to random failures in ingress links or PoPs, and random variation of user latencies to each ingress. We will assess whether computed strategies retain their benefits over long periods of time, as over time path latencies and pairwise preferences among ingresses may change.

We will assess whether our framework retains its favorable properties (offering low-latency, resilient deployments) compared to other solutions as our routing model (pairwise preference assumption) breaks down, or as the ground-truth model changes rapidly. We will evaluate whether our framework retains its benefits at scale—whether we can compute solutions in a reasonable amount of time at realistic deployment sizes, and/or whether some property (*e.g.,* resilience) is satisfied less often for larger deployments.

Can We Trust Solutions? We will then evaluate the framework's "trustworthiness", as computed solutions could have serious business implications. For example, if a solution dictates that it is too costly (*i.e.,* not worth it) to advertise prefixes to a particular peer/provider (even for resilience purposes), we might infer that the CP should cancel their peering agreement with that party. A signal that could be useful to evaluate the trustworthiness of these major decisions is whether we repeatedly arrive at this conclusion across several random initializations.

Transferring to Other Deployments? If CPs were confident that the proposed framework would improve performance for their users, adoption of and iteration on the framework would more readily occur. Developing and fine-tuning such a system could require significant engineering effort for hypergiants due to their scale and reliability requirements. We would like to see if there are ways of taking optimal advertisement strategies from one deployment and "porting" them to other deployments. One possible way we could approach this is to derive major performance drivers (*i.e.,* user latencies) from advertisement strategies for broad classes of simulated models, and invert (*i.e.,* decode) these representations on new deployments. We would train and evaluate the methodology for porting strategies on simulated deployments, allowing for rapid iteration.

Answering "What-If" Questions We finally hope to determine whether this framework can also be used to *understand* deployments better, despite the significant additional advertisement complexity. A key limitation of anycast is that it is difficult to predict properties of the deployment (user latencies, link volumes) in unmeasured scenarios. Some prior work [27] used historical information to roughly predict where users ingress after route withdrawal, but predictions were imprecise and modifications to the deployment were slight (*i.e.,* an announcement withdrawal over a single link). During convergence, our model intelligently issues advertisements to build a model of user ingress preferences, notably in cases where we are uncertain about performance or resilience. Therefore, answering questions in unseen situations about (a) where traffic will ingress, (b) what performance users will see, (c) how resilient the new deployment is, and (d) how *confident* we are in any of those answers is as simple as querying our model. We will evaluate our model's ability to provide predictions about user ingresses in unseen situations to compare to TIPSY [27], and its ability to predict user performance like WISE [45].

6 CONCLUSIONS AND FUTURE WORK

Towards overcoming the limitations of BGP, we propose optimizing deployments over several dimensions (shown in Fig. 3). Our proposed framework represents significant progress in optimizing over just one of these dimensions, but this proposal also highlights several future research directions.

Future Research Directions that our proposal does not encompass. The proposed framework considers optimizing over simple "on/off" advertisement strategies (§4.2) whereas other frameworks could incorporate richer BGP policies capable of exposing more paths. Moreover, we optimize over ingresses into the deployment, rather than paths over the public Internet to the deployment—a richer framework could incorporate path-level information to inform better benefit predictions.

Going beyond the proposed framework, future work could consider optimizing the *other* knobs in Figure 3. Finding optimal PoP locations, peering/provider contracts, and proxy locations are challenging but relevant problems. Adding more PoPs, for example, could reduce latency for users but comes at a deployment cost. Algorithmic challenges such as coordinating among proxies to determine jointly optimal forwarding decisions is another interesting area for future work.

Finally, while we envision small-scale deployments on research testbeds, deploying these systems in real CPs is an important direction for future work, as issues that are only fully visible in those domains (*e.g.,* scale) may arise.

Bringing it All Together Enhanced control over user traffic, better network application performance, and answering "what-if" questions are all significant advancements to get excited about, and the ideas my research explores could have even further-reaching implications. Exercising control over user traffic from a relatively small number of proxies could drastically speed up technological/framework rollout velocity, and enable new (unknown) applications or network architectures. The enhanced understanding of deployments from models of how users are routed to services could lead to new insights, network architectures, and peering strategies, and could even inform how deployments should evolve (new regions, product spaces). The enhanced performance that my framework delivers could enable next-generation applications (*e.g.,* URLLC). I see this proposed framework as the first in a wave of systems that exercise fine-grained control of client traffic that will drive the high-performance, Internet-scale systems of tomorrow.

REFERENCES

[1] Apple. 2021. Configuring Network Extensions. developer.apple.com/documentation/xcode/configuring-network-extensions

[2] Henry Birge-Lee, Maria Apostolaki, and Jennifer Rexford. 2022. It Takes Two to Tango: Cooperative Edge-to-Edge Routing. In *HOTNETS*.

[3] Matt Calder, Xun Fan, and Liang Zhu. 2019. A Cloud Provider's View of EDNS Client-Subnet Adoption. In *TMA*.

[4] Matt Calder, Ashley Flavel, Ethan Katz-Bassett, Ratul Mahajan, and Jitendra Padhye. 2015. Analyzing the Performance of an Anycast CDN. In *IMC*.

[5] Fangfei Chen, Ramesh K Sitaraman, and Marcelo Torres. 2015. End-user mapping: Next generation request routing for content delivery. In *SIGCOMM*.

[6] Ricardo de Oliveira Schmidt, John Heidemann, and Jan Harm Kuipers. 2017. Anycast latency: How many sites are enough? In *PAM*.

[7] Wouter B De Vries, Ricardo de O. Schmidt, Wes Hardaker, John Heidemann, Pieter-Tjerk de Boer, and Aiko Pras. 2017. Broad and load-aware anycast mapping with verfploeter. In *IMC*.

[8] Ashley Flavel, Pradeepkumar Mani, David Maltz, Nick Holt, Jie Liu, Yingying Chen, and Oleg Surmachev. 2015. Fastroute: A Scalable Load-Aware Anycast Routing Architecture for Modern CDNs. In *NSDI*.

[9] Ashley Flavel, Pradeepkumar Mani, and David A Maltz. 2014. Re-Evaluating the Responsiveness of DNS-Based Network Control. In *LANMAN*.

[10] Petros Gigis, Matt Calder, Lefteris Manassakis, George Nomikos, Vasileios Kotronis, Xenofontas Dimitropoulos, Ethan Katz-Bassett, and Georgios Smaragdakis. 2021. Seven years in the life of Hypergiants' off-nets. In *SIGCOMM*.

[11] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. 2013. Achieving High Utilization with Software-Driven WAN. In *SIGCOMM*.

[12] IETF. 2011. Architectural Guidelines for Multipath TCP Development. https://datatracker.ietf.org/doc/rfc6182/

[13] IETF. 2022. IP Proxying Support for HTTP. datatracker.ietf.org/doc/draft-ietf-masque-connect-ip/

[14] Mordor Intelligence. 2022. Network as a service market - growth, trends, COVID-19 impact, and forecasts. mordorintelligence.com/industry-reports/network-as-a-service-market-growth-trends-and-forecasts

[15] Umar Javed, Italo Cunha, David Choffnes, Ethan Katz-Bassett, Thomas Anderson, and Arvind Krishnamurthy. 2013. PoiRoot: Investigating the root cause of interdomain path changes. *SIGCOMM* (2013).

[16] Dina Katabi and John Wroclawski. 2000. A Framework for Global IP-Anycast (GIA). In *Proceedings of the 2000 ACM SIGCOMM Conference*. ACM.

[17] Thomas R Knapp. 2007. Bimodality revisited. *Journal of Modern Applied Statistical Methods* (2007).

[18] Thomas Koch, Ethan Katz-Bassett, John Heidemann, Matt Calder, Calvin Ardi, and Ke Li. 2021. Anycast in context: A tale of two systems. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*.

[19] Cyrill Krähenbühl, Seyedali Tabaeiaghdaei, Christelle Gloor, Jonghoon Kwon, Adrian Perrig, David Hausheer, and Dominik Roos. 2021. Deployment and scalability of an inter-domain multi-path routing infrastructure. In *CoNEXT*.

[20] Raul Landa, Lorenzo Saino, Lennert Buytenhek, and João Taveira Araújo. 2021. Staying Alive: Connection Path Reselection at the Edge.. In *NSDI*.

[21] Zhihao Li. 2019. *Diagnosing and Improving the Performance of Internet Anycast.* Ph.D. Dissertation. University of Maryland, College Park.

[22] Zhihao Li, Dave Levin, Neil Spring, and Bobby Bhattacharjee. 2018. Internet Anycast: Performance, Problems, & Potential. In *SIGCOMM*.

[23] Jinjin Liang, Jian Jiang, Haixin Duan, Kang Li, and Jianping Wu. 2013. Measuring Query Latency of Top Level DNS Servers. In *PAM*.

[24] Hongqiang Harry Liu, Raajay Viswanathan, Matt Calder, Aditya Akella, Ratul Mahajan, Jitendra Padhye, and Ming Zhang. 2016. Efficiently Delivering Online Services over Integrated Infrastructure. In *NSDI*.

[25] Matthew Luckie, Bradley Huffaker, Amogh Dhamdhere, Vasileios Giotsas, and KC Claffy. 2013. AS relationships, customer cones, and validation. In *Proceedings of the 2013 conference on Internet measurement conference*. 243–256.

[26] Zhuoqing Morley Mao, Charles D Cranor, Fred Douglis, Michael Rabinovich, Oliver Spatscheck, and Jia Wang. 2002. A Precise and Efficient Evaluation of the Proximity Between Web Clients and Their Local DNS Servers.. In *NSDI*.

[27] Michael Markovitch, Sharad Agarwal, Rodrigo Fonseca, Ryan Beckett, Chuanji Zhang, Irena Atov, and Somesh Chaturmohta. 2022. TIPSY: predicting where traffic will ingress a WAN. In *SIGCOMM*.

[28] Stephen McQuistin, Sree Priyanka Uppu, and Marcel Flores. 2019. Taming Anycast in the Wild Internet. In *IMC*.

[29] Zili Meng, Yaning Guo, Chen Sun, Bo Wang, Justine Sherry, Hongqiang Harry Liu, and Mingwei Xu. 2022. Achieving consistent low latency for wireless real-time communications with the shortest control loop. In *SIGCOMM*.

[30] Christopher Metz. 2002. IP Anycast Point-To-(Any) Point Communication. *IEEE Internet Computing* (2002).

[31] Giovane CM Moura, John Heidemann, Ricardo de O Schmidt, and Wes Hardaker. 2019. Cache me if you can: Effects of DNS Time-to-Live. In *IMC*.

[32] Srinivas Narayana, Wenjie Jiang, Jennifer Rexford, and Mung Chiang. 2013. Joint server selection and routing for geo-replicated services. In *International Conference on Utility and Cloud Computing*.

[33] Lars Prehn, Pawel Foremski, and Oliver Gasser. 2022. Kirin: Hitting the Internet with Millions of Distributed IPv6 Announcements. *arXiv preprint arXiv:2210.10676* (2022).

[34] Enric Pujol, Ingmar Poese, Johannes Zerwas, Georgios Smaragdakis, and Anja Feldmann. 2019. Steering hyper-giants' traffic at scale. In *CoNEXT*.

[35] Costin Raiciu, Christoph Paasch, Sebastien Barre, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure, and Mark Handley. 2012. How hard can it be? designing and implementing a deployable multipath TCP. In *NSDI*.

[36] ASM Rizvi, Joao Ceron, Leandro Bertholdo, and John Heidemann. 2022. Anycast agility: Adaptive routing to manage DDoS. *arXiv preprint arXiv:2006.14058* (2022).

[37] Sandeep Sarat, Vasileios Pappas, and Andreas Terzis. 2006. On the Use of Anycast in DNS. In *SIGMETRICS*.

[38] Patrick Sattler, Juliane Aulbach, Johannes Zirngibl, and Georg Carle. 2022. Towards a tectonic traffic shift? investigating Apple's new relay network. In *IMC*.

[39] Brandon Schlinker, Todd Arnold, Italo Cunha, and Ethan Katz-Bassett. 2019. PEERING: Virtualizing BGP at the Edge for Research. In *CoNEXT*.

[40] Brandon Schlinker, Hyojeong Kim, Timothy Cui, Ethan Katz-Bassett, Harsha V Madhyastha, Italo Cunha, James Quinn, Saif Hasan, Petr Lapukhov, and Hongyi Zeng. 2017. Engineering egress with edge fabric: Steering oceans of content to the world. In *SIGCOMM*.

[41] William Sentosa, Balakrishnan Chandrasekaran, P Brighten Godfrey, Haitham Hassanieh, and Bruce Maggs. 2023. DChannel: Accelerating Mobile Applications With Parallel High-bandwidth and Low-latency Channels. In *NSDI*.

[42] Pavlos Sermpezis and Vasileios Kotronis. 2019. Inferring catchment in Internet routing. *Measurement Analysis and Computing Systems* (2019).

[43] Yoav Shechtman, Amir Beck, and Yonina C Eldar. [n.d.]. GESPAR: Efficient phase retrieval of sparse signals. *IEEE transactions on signal processing* ([n. d.]).

[44] Rachee Singh, Nikolaj Bjørner, and Umesh Krishnaswamy. 2022. Traffic engineering: from ISP to cloud wide area networks. In *SOSR*.

[45] Mukarram Tariq, Amgad Zeitoun, Vytautas Valancius, Nick Feamster, and Mostafa Ammar. 2008. Answering what-if deployment and configuration questions with WISE. In *SIGCOMM*.

[46] Vytautas Valancius, Bharath Ravi, Nick Feamster, and Alex C Snoeren. 2013. Quantifying the benefits of joint content and network routing. In *SIGMETRICS*.

[47] Lan Wei, Marcel Flores, Harkeerat Bedi, and John Heidemann. 2020. Bidirectional Anycast/Unicast Probing (BAUP): Optimizing CDN Anycast. In *NOMS*.

[48] John Wright and Yi Ma. 2022. *High-dimensional data analysis with low-dimensional models: Principles, computation, and applications*. Cambridge University Press.

[49] Mengwei Xu, Zhe Fu, Xiao Ma, Li Zhang, Yanan Li, Feng Qian, Shangguang Wang, Ke Li, Jingyu Yang, and Xuanzhe Liu. 2021. From cloud to edge: a first look at public edge platforms. In *IMC*.

[50] Jing'an Xue, Weizhen Dang, Haibo Wang, Jilong Wang, and Hui Wang. 2019. Evaluating performance and inefficient routing of an anycast CDN. In *IWQoS*.

[51] Kok-Kiong Yap, Murtaza Motiwala, Jeremy Rahe, Steve Padgett, Matthew Holliman, Gary Baldus, Marcus Hines, Taeeun Kim, Ashok Narayanan, Ankur Jain, et al. 2017. Taking the Edge off with Espresso: Scale, Reliability and Programmability for Global Internet Peering. In *SIGCOMM*.

[52] Xiao Zhang, Tanmoy Sen, Zheyuan Zhang, Tim April, Balakrishnan Chandrasekaran, David Choffnes, Bruce M Maggs, Haiying Shen, Ramesh K Sitaraman, and Xiaowei Yang. 2021. AnyOpt: predicting and optimizing IP Anycast performance. In *SIGCOMM*.

[53] Yang Zhang, Jean Tourrilhes, Zhi-Li Zhang, and Puneet Sharma. 2021. Improving SD-WAN resilience: From vertical handoff to WAN-aware MPTCP. In *ToN*.

[54] Jiangchen Zhu, Kevin Vermeulen, Italo Cunha, Ethan Katz-Bassett, and Matt Calder. 2022. The best of both worlds: high availability CDN routing without compromising control. In *IMC*.