# BGP Anycast Tuner: Intuitive Route Management for Anycast Services

Leandro M. Bertholdo*, João M. Ceron†, Lisandro Z. Granville‡,
Giovane C. M. Moura†, Cristian Hesselman†*, Roland van Rijswijk-Deij*§
*University of Twente, Enschede, The Netherlands
{l.m.bertholdo, r.m.vanrijswijk}@utwente.nl
†SIDN Labs, Arnhem, The Netherlands
{joao.ceron, giovane.moura, cristian.hesselman}@sidn.nl
‡Federal University of Rio Grande do Sul, Porto Alegre, Brazil
granville@inf.ufrgs.br
§NLnet Labs, Amsterdam, The Netherlands

***Abstract*—IP anycast has become a vital technology for DNS and CDN operators alike. Yet, while big operators have their tools to monitor and configure anycast routing, most of anycast networks are still configured manually. In this paper, we introduce a new approach to anycast management. Our solution is based on active measurements combined with traffic engineering. We propose the concept of a "BGP Cookbook" that allows operators to forecast the effects of routing policy changes over their services. We also introduce a web-based interface, called *"BGP Anycast Tuner"*, that allows operators to gain insight into their service's performance and provides easy management through automation. We evaluate our approach by implementing a prototype running in a testbed composed of 12 anycast sites covering 5 continents. We demonstrate our tool in two different use cases: discovering and fixing a sub-optimal anycast routing issue, and shifting traffic between continents, which is useful during service disruptions.**

## I. Introduction

IP anycast is a technology that allows for the same IP address to be announced from multiple global locations. Each announcement is generated by one anycast site, and whole solutions relies on inter-domain routing to allocate traffic between sites [1] [2]. IP Anycast is used in Content Delivery [3] [4] [5], distributed denial-of-service (DDoS) mitigation [6] [7] [8], and by the Domain Name System (DNS) infrastructure [9] [10] [11]. Nowadays, it is possible to identify hundreds of anycast prefixes being used across the Internet [12].

Anycast routing divides the Internet into "catchments", where each site attracts a share of clients. To do that, IP anycast relies on BGP, the default inter-domain routing protocol on the Internet [13]. However, this division is not perfect, and clients sometimes wind up at faraway sites [14] [15]. To help operators map catchments, Verfploeter [16] is the state-of-the-art tool. Just mapping catchments, though, is not enough. Operators want to control how many clients, or even which clients, go to each site for performance and security reasons. This can be done by changing BGP routing at anycast sites.

While large anycast operators deal with catchment control by adding more sites, most of anycast operators needs to perform routing adjustments to better support their users. One example of small anycast operator are the DNS cc-TLD (country code top-level domain), who frequently owns less than twenty sites by anycast prefix. Small anycast networks adjust catchment by doing routing changes manually. This limits the ability of an operator to explore the best configurations. Moreover, such manual tasks are prone to human errors with potentially serious consequences [17] [18] [19] [20].

To help operators address this problem, we present BGP Anycast Tuner, a measurement-based tool that automates the configuration of anycast sites, the administration of catchments, and the analysis and visualization of anycast management. Through an intuitive web-based interface, operators can observe the predicted effects of routing policy changes, select an appropriate configuration, and deploy it automatically.

The main contributions of our work are: (*i*) we establish a systematic approach to measure the distribution of clients of anycast services under different routing conditions, (*ii*) we present and release an open source tool, BGP Anycast Tuner, to manage anycast services using a routing Cookbook; and (*iii*) we evaluate our prototype in a real-world anycast testbed.

The remainder of this paper is organized as follows. Related work is discussed in Section II. In Section III, we review anycast routing and we establish the necessary requirements for anycast network management. In Section IV, we present our solution, BGP Anycast Tuner, followed by Section V, where we explain details of our implementation and evaluation on a real-world anycast testbed. Finally, we draw conclusions and future work in Section VI.

## II. Related Work

Anycast has been extensively studied since it was first proposed back in 1993 [1]. Early studies focused on the first large-scale application of anycast to the DNS [9] [21] [22], and on architectures for creating a global anycast service [3] [4]. More recently, anycast was used as a security tool [23] [6] [24]. Hesselman *et al.* [25] study how to provide a control plane for DNS top-level domain (TLD) operators to increase security and stability of TLDs, while Rizvi *et al.* [8] describe

a method for building a "response playbook" for the use of BGP to shift anycast traffic when under DDoS attacks.

On anycast routing and performance, Ballani *et al*. were the first to study anycast performance metrics [26]. Wei *et al.* [27] show that while anycast routing is generally stable, a small number of routes sees persistent instability. De Vries *et al.* [28] developed VERFPLOETER to measure anycast catchments and later deployed it in one of the largest global anycast CDNs [29]. McQuistin *et al.* [30] analyze the impact of routing on anycast CDNs. Recently, Wei *et al.* [31] proposed a system to detect latency issues through an anycast/unicast probing.

Although there is no specialized management for anycast networks yet, Wiefferink [32] evaluated how current tools can help in the analysis of anycast, while Costella *et al.* [33] built a solution to visualize anycast catchments.

## III. REQUIREMENTS

In this section we discuss the challenges and requirements of managing anycast services. First, however, we present fundamental concepts that help understand the proposed solution.

By using anycast, services are hosted on multiple servers using the same IP address. Figure 1 depicts an anycast service where three sites ($S_n$) share the address `10.0.0.1`. Each client (bottom part) tries to reach the server IP via distinct paths according to AS (clouds) interconnect relationships (straight lines). Under the hood, BGP is responsible for selecting the routing path for each destination according to a set of metrics and policies. The color of each AS shows its preference in reaching the `10.0.0.0/24` prefix.

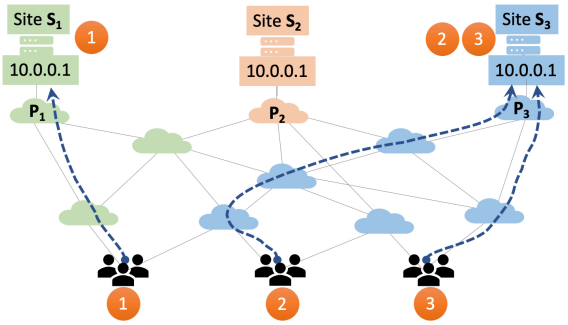


Fig. 1: Anycast catchment example.

Each site has a so-called *catchment*, which is the set of clients that reaches it. The catchment of the green site ($S_1$) consists of client 1, the red site ($S_2$) attracts no clients, and the blue site ($S_3$) attracts clients 2 and 3.

An operator observing the anycast network in Figure 1 may conclude that: (*i*) client 2 is likely better served by the red server ($S_2$) than the blue one ($S_3$), and (*ii*) in order to make the red server ($S_2$) attract client 2, routing must be reconfigured.

There are many cases where the catchment of an anycast service may be less than ideal for some clients [31]. To modify client catchment – *i.e.*, to make a client end up at a different anycast server – BGP Interdomain Traffic Engineering (BGP-TE) [34] would be required. While Figure 1 is just a toy example, to perform this for a real anycast network with tens, or even hundreds of sites, the use of management tools is essential. Anycast operators need to know, for example, how clients are distributed across the anycast service, what would be the optimal catchment for a particular client, and how traffic engineering affects the catchment distribution.

Our goal is to make operating an anycast service easier by introducing management tooling. In the remainder of this section, we introduce requirements for such management tooling. Table I provides an overview, and we discuss each requirement in detail next.

| Requirement | Anycast service management tooling… |
|---|---|
| R1 | needs to map site catchments |
| R2 | needs to control site catchments |
| R3 | support several metrics for clients and sites |
| R4 | automatically discover traffic engineering (TE) options |
| R5 | support fast, atomic configuration deployment |
| R6 | have a simple and scalable management interface |

TABLE I: Requirements for anycast traffic management.

**R1: Catchment Mapping —** To understand the traffic distribution and observe deviations from the baseline, it is essential to map the catchment of each anycast site. Anycast catchments can be measured both passively and actively. Passive approaches rely on analyzing incoming flows to anycast sites. Active approaches, in turn, generate traffic from vantage points (VPs) in typically two approaches: using application clients such as RIPE Atlas [35], or by probing the Internet and observing the answers [28].

**R2: Traffic Engineering(TE) for Catchment Control —** Although the anycast operator does not have control of the routing path that each client uses, it is possible to use BGP attributes to influence the routing preferences at the AS-Level. The most common used attributes are AS-Path prepending [36], and community strings [37]. Anycast management tooling must support easy manipulation of routing policies for all anycast sites and measure its impact on the service (R1).

**R3: Support for "Pluggable" Metrics —** While catchment control is a common goal of operators, the motivation for exerting such control varies. For example, a common go-to metric to measure *User-Experience* is delay, such as round-trip time [38]. However, other goals and metrics should also be supported. Consider for example an operator who needs to control in which geographic region clients end up, in order to comply with a country's laws. In this case, client metrics should include *IP Geo-location* [15] [39]. Another example is when the objective is to handle DDoS attacks; in this situation, metrics should consider *site metrics*, as interface loss rate and CPU loads [6] [8].

**R4: Automated Discovery of TE Options —** As previously discussed, anycast operators may use individual site TE (traffic engineering) options to modify the catchment distribution. Such options is tied to how the site is connected (*e.g.,* number of peers and policies). More sites and richer connectivity provide ranges from dozens to thousands policies. By combining all options, the operator has a complex problem to

solve: which BGP policy should be used and which are its effects on site catchments? To deal with this complexity, one requires an automated method to measure all the BGP policies available per site, and map their side effects in term of client distribution.

**R5: Fast, Atomic Configuration Deployment —** Manual routing configuration of each site does not scale for anycast networks. Problems, such as DDoS attacks, make it necessary to quickly act when deploying a chosen routing configuration. Therefore, a key requirement for an anycast management system is that a new routing configuration must be deployed in a fast and atomic way on all sites that together comprise the anycast service.

**R6: A Simple and Scalable Management Interface —** Most anycast services range in size from a few to tens of different sites. A requirement for a management system therefore is that its interface must be able to accommodate these differences in scale. Furthermore, a management interface should be intuitive for operators, both in visualizing the catchment distribution (helping operators understand how routing affect catchments) and in changing routing policies to achieve certain goals (*e.g.*, shift traffic away from certain sites, improve performance for certain clients, prioritize sites with spare capacity).

## IV. Approach

In this section, we describe our approach to manage anycast client distribution, how we test routing policies, optimize this process, and provide this information to operators.

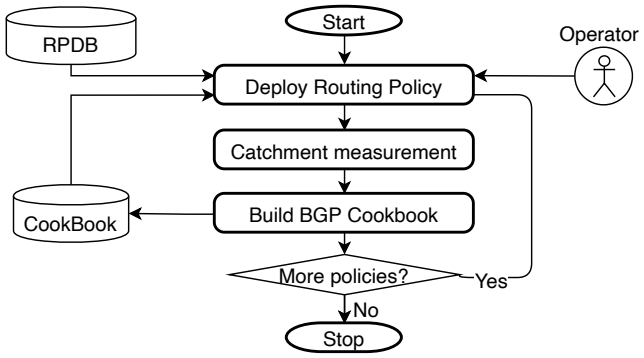


Fig. 2: Method used to map the effect of traffic engineering over the anycast client distribution.

### A. Overview

We propose a scalable methodology to map the effects of client distribution over anycast sites given a BGP routing policy. Figure 2 shows a schematic overview of our approach. To bootstrap the process, it is fundamental to determine all the available BGP policies per site. This information is stored in a *BGP Routing Policy Database*, and it is used to deploy the configuration. The process *Deploy Routing Policy* is responsible for applying a routing configuration on the sites of the anycast service. Next, we perform the *Catchment Measurement*. Data from this measurement describes the client distribution per site at each deployed BGP policy. We summarize this distribution in a COOKBOOK that describes the effect of a particular policy on the anycast service. Finally, an *Operator* (top-right) can choose which policies the system evaluates, and select an appropriate one to achieve certain operational goals.

### B. The Routing Policy Database (RPDB)

An anycast service typically consists of multiple sites, each with their own external connectivity and different options to influence routing. We map all these possible BGP routing scenarios in a centralized repository named Routing Policy Database. This database reflects the *BGP peering agreement* established between each site and the respective upstream provider and/or Internet eXchange Point (IXP).

Table II illustrates the concept for one site in our example network from Figure 1. This table shows six BGP routing policies that can be applied on site $S_3$. Column *Peer* describes the BGP neighbors of the site, *i.e.*, where the routing policy will be deployed. *Prefix* describes the routing prefix that is advertised in conjunction with the *Routing Policy*. For example, in the first row (*Baseline*), no routing policy is used. From rows 2 up to 3, AS-Path prepending is used, and in the last two rows the BGP community strings *noPeer* and *noExport* are used [40].

| Policy Id | Site | Peer | Prefix | Routing Policy |
|---|---|---|---|---|
| Baseline | $S_3$ | $P_3$ | 10.0.0.0/24 | – |
| 1xPrepend | $S_3$ | $P_3$ | 10.0.0.0/24 | $AS_n$ |
| 2xPrepend | $S_3$ | $P_3$ | 10.0.0.0/24 | $AS_n\_AS_n$ |
| 3xPrepend | $S_3$ | $P_3$ | 10.0.0.0/24 | $AS_n\_AS_n\_AS_n$ |
| noPeer | $S_3$ | $P_3$ | 10.0.0.0/24 | community 65535:65284 |
| noExport | $S_3$ | $P_3$ | 10.0.0.0/24 | community 65535:65281 |

TABLE II: Example of RPDB for Figure 1.

Operators must decide upon which policy will be added to RPDB. More combinations means more TE options and more fine-grained traffic control. Too many combinations, however, can lead to time being spent on low impact anycast-TE options. The operator is responsible for establishing a trade-off between policy combinations and useful results to store in the COOKBOOK.

### C. Deploying Routing Policy

With RPDB in place, the next step is to deploy one routing policy (one row from Table II). While deploying a policy, it is important to observe best practices on inter-domain routing, such as BGP convergence time and route flapping [41].

### D. Catchment Measurement

At this point, the anycast sites are configured with the selected BGP policy. Next, we need to map the client distribution and other metrics affected by the deployed configuration. Which other metrics to consider depends on the goals of the operator, but that can include, for example, geographic location of clients or traffic load on each site. In our approach, we choose the methodology of De Vries *et al.* [28] because it is an active measurement, has larger coverage, and delivers a more impartial view than the other options available.

### E. The BGP COOKBOOK

After deploying each routing policy and measuring its impact, we store the results in a database named the BGP COOKBOOK. This name derives from the idea that we consider each routing policy as an ingredient used to build different traffic engineering recipes. While in RPDB we store all possible routing policies available, in the COOKBOOK we just store those whose results are relevant in the eyes of the operator (manual), or following parameters established by the operator, such as the percentage gain (in terms of traffic shift) compared to previous stored recipes (automated).

Table III shows a possible COOKBOOK for our example from Figure 1. Let's assume that the operator wants to distribute the clients evenly over the sites. It is important to remember that, in the baseline situation shown in Figure 1, $S_3$ attracts two clients while $S_2$ does not attract any.

| Policy Id | Routing Policy | | | Metric $Qt\_Catchment$ | | |
|---|---|---|---|---|---|---|
| | $S_1P_1$ | $S_2P_2$ | $S_3P_3$ | $S_1P_1$ | $S_2P_2$ | $S_3P_3$ |
| Baseline | – | – | – | 1 | 0 | 2 |
| 1x$S_3$ | – | – | 1xPrep | 1 | 2 | 0 |
| noPeer$S_3$ | – | – | noPeer | 1 | 1 | 1 |
| -3x$S_2$ | 3xPrep | – | 3xPrep | 0 | 3 | 0 |

TABLE III: Example of COOKBOOK for Figure 1.

In this COOKBOOK, we find our normal state (Baseline) and three distinct anycast-TE strategies: **1x$S_3$**, saying one prepend on $S_3P_3$ will result in clients $C_1$ and $C_2$ shifting from the Blue Site to the Red one; **noPeer$S_3$**, forecasting the metric $Qt_{Catchment}$ will be equalized if we deploy a policy called noPeer (BGP community `no-peer`) on site $S_3$; and the **-3x$S_2$**, indicating reverse prepend or negative prepend as a composite policy to bring all clients to $S_2$ through prepending on all sites except $S_2$. As can be noticed, **noPeer$S_3$** is the policy aligned with the operator's goal.

### F. Process Optimization and Limitations

While the process described so far yields a useful COOKBOOK of relevant routing configurations, there are two important concerns regarding scalability: (*i*) the time required to deploy and collect metrics for each anycast-TE recipe; and (*ii*) the combinatorial explosion resulting from an increase in the number of anycast sites.

In our prototype (Section V), we can check around 60 distinct routing policies per day. If we build the entire COOKBOOK for our example at Figure 1–3 anycast sites and 5 policies–we would spend over 6 hours ($3 \times 5 \times 25$ min). However, if we test all possible policy combinations, such as the **-3x$S_2$** in Table III, we would spend around 52 hours ($5^3 \times 25$ min). If we expand our toy scenario to 10 sites and 7 policies per site, a naïve test would take thousands of years.

However, instead of testing all possible combinations, we can choose an explore-and-test approach to significantly reduce the number of tests. The policy selection can be optimized by considering one of the following approaches:

- **Customer Cone:** Based on number of customers of each AS [42], we were able to reduce nearly 99% of individual peer tests on IXPs. We selected just to apply policies on ASes with large cone sizes as provided by AS-Rank [43].
- **Sparse Test**: For each class of test, we can divide all combinations into smaller chunks and we try the best and worst cases on each chunk first. If results are not good, we stop (*e.g.*, we try 3-prepends and compare with site baseline before trying 1 or 2 prepends). We could reduce in 83% the test on individual routing policies.
- **Representativeness**: We do not try policies for peers that account for less than 1% of a site's catchment.
- **Human-in-the-Loop Factor:** An operator knows best when it is good enough to stop testing more policies.

Although with lower impact, there is also room to optimize the other time-consuming part of our process, measuring the catchment. Our chosen measurement system uses a hit list to probe at least one address in each routable /24 prefix on the Internet. We can reduce time by reducing the list in two ways:

- **Hitlist shrinking:** Probing just a single address per prefix. As most prefixes on full routing table are larger than /24 (*e.g.*, of size /16), we reduced the hit list in 87%.
- **Hitlist dimming:** Probing a single address per AS. While this reduces visibility into the largest ASes, which may not have homogeneous routing policies for all of their prefixes, the view on smaller ASes remains precise, and the number of measurements was vastly reduced by 98%.

### G. The BGP Anycast Tuner Interface

The BGP policies and metrics summarized in the COOKBOOK aim to help an operator chose the best routing policies to achieve a certain goal. However, choosing the best solution can be a complex task when hundreds of policies are available.

A closer examination of our approach shows that the results we stored in the COOKBOOK are mainly about attracting more traffic to a site, or reducing it. Also consider the fact that anycast is a "closed system", *i.e.*, if one site attracts more traffic, other sites will lose such traffic. With this in mind, it is possible to think of each individual site having a limited number of options within a band of possibilities ranging between positive (attract more traffic) and negative (reduce traffic) values, and if one site is changed, all other sites will change as well, compensating the difference. This concept reminds of an old-fashioned "sound tuner", leading us to chose this as a metaphor for our graphical user interface.

BGP ANYCAST TUNER is a prototype graphical interface that brings together all parts of our methodology. It provides a simple and intuitive interface for operators and presents the distribution of clients over the anycast sites for the different pre-determined BGP configurations from the BGP COOKBOOK. Figure 3 presents the BGP ANYCAST TUNER graphical user interface. The figure shows the client distribution for each site using a histogram (in blue), and sliders underneath the histogram. Using these sliders, operators can increase or decrease the catchment of a site using a set of predetermined settings indicated by "notches" on the slider. These notches correspond to specific BGP policies, the effect on catchment

we previously determined using measurements. As the example also shows, these notches are not evenly distributed across the slider, reflecting the fact that the flexibility of a site to shift clients to or from other anycast sites varies. In other words, it reflects the fact that some BGP policies result in a larger shift in catchment than others. In addition, some sites have more degrees of control because of their relationships with their BGP neighbors and traffic agreements. Also notice that the sliders for the sites are linked, *i.e.*, if the slider of one site is moved, all other sliders move as well to reflect the fact that a change in catchment for one site automatically means the catchments of all other sites change as well.

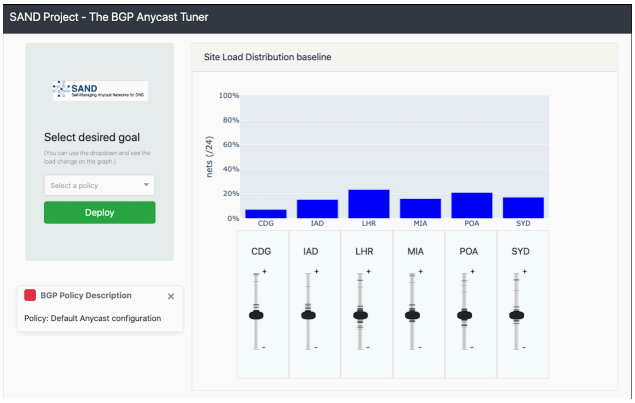


Fig. 3: BGP Anycast Tuner user interface.

## V. Prototype and Evaluation

In this section, we evaluate our approach against the requirements discussed earlier in Table I. For this evaluation, we prototyped our approach and implemented it on an anycast testbed infrastructure. In the subsequent subsections, we describe our testbed, the results we obtained in terms of traffic engineering, and show how the prototype behaves in two scenarios. The prototype code can be found in our repository[1].

### A. Prototype Environment and Implementation

We built our prototype on top of our anycast testbed Tangled [44]. This testbed consists of 12 anycast sites across the globe. It have a good diversity of peers, being connected to IXPs, cloud providers, commercial transit providers, and academic networks. Such similarly to industry environment supporting a full-range of routing agreements, make it a challenging testing ground for our approach.

We note that, although the prototype was implemented in the Tangled anycast testbed, the implementation method is not tied to its specific topology, sites, or network resources. Furthermore, our goal for the evaluation is not to quantify or measure results specific to the testbed, but rather to validate that our approach works and meets the requirements. Thus, we restricted the choice of routing policies in our evaluation process, avoiding combinations of policies and sending the same BGP announcements to all peers at each site.

[1]https://github.com/SIDN/BGP-Anycast-Tuner

Our prototype consists of four parts: (*i*) a routing policy deployment tool based on ExaBGP that is used to deploy the selected routing policy on all the sites of the anycast service, (*ii*) an anycast-TE orchestration tool that deploys a chosen recipe from the Cookbook on all sites and coordinates the catchment measurement with Verfploeter, (*iii*) a statistical analysis tool for the catchment measurement data that persists resulting metrics to the Cookbook, and finally (*iv*) two dashboards for the operator, one to visualize catchment data and one for load tuning (the latter forms the core of our approach and brings together all of its components).

As part of the deployment process, we wait for BGP to settle for 10 minutes. This amount of time is presented in the literature [45] [46] [47] as a necessary routing and forwarding convergence time.

### B. A Cookbook for Tangled

As the first step of our evaluation, we build a Cookbook for the Tangled testbed. To make the evaluation easier to explain, we show just a subset of 6 anycast sites and 13 distinct routing policies per site (the entire cookbook has 78 policies). Every line in Table IV represents a measurement in our testbed and the respective percentage of /24 networks handled by each site for each BGP policy. We highlight the row with our service baseline (regular BGP announcement). We also highlight the site LHR and sort the table in descending order, from maximum load (64.58%) to minimum load (0%).

Observing our baseline, we note that 7.22% of clients went to the site located in Paris (CDG); 15.28% went to Washington D.C. (IAD); 23.38% went to London (LHR); 15.97% to Miami (MIA); 20.98% went to Porto Alegre (POA), and 17.16% went to Sydney (SYD). In the Anycast-TE column, we use mnemonics (*e.g.*, *-2xLHR*) to describe two negative prepends for site LHR (the negative notation means two prepends on all sites except LHR). Other examples of mnemonics are *1xLHR* to describe a single prepend for the London site, and *LHRnoPeer* and *LHRnoExp* to express that the BGP community `noPeer` and `noExport` were used at London.

The measurement data in Table IV predicts the number of clients at each site for each BGP configuration. Operators can use this table to configure their anycast service depending on the desired load on sites. Ordering entries by LHR load, as we did, reflects an operator strategy to select how much traffic he/she wants to bring to or remove from LHR. In anycast services with many sites, however, this process may be confusing and error-prone. To address this challenge, we introduced BGP Anycast Tuner to automate this process and visualize it in an intuitive manner. Effectively, ordering the table illustrates how the slider for LHR in the BGP Anycast Tuner interface shown in Figure 3 is populated.

In the next subsections, we present two scenarios in which we applied our approach in the Tangled testbed.

### C. Scenario 1: Catchment Visualization and Troubleshooting

This scenario shows how our prototype meets requirements R1, R2, R3, R4, and R6. The scenario illustrates how an operator can detect and repair a case of sub-optimal routing.

| Anycast-TE | CDG | IAD | LHR | MIA | POA | SYD |
|---|---|---|---|---|---|---|
| -2xLHR | 2.04 | 15.17 | 64.58 | 8.89 | 8.05 | 1.26 |
| -1xLHR | 2.42 | 15.03 | 54.09 | 12.39 | 14.55 | 1.50 |
| 2xMIA | 7.68 | 16.15 | 30.33 | 2.35 | 20.39 | 23.09 |
| 2xIAD | 7.03 | 14.62 | 25.02 | 16.19 | 19.73 | 17.41 |
| Baseline | 7.22 | 15.28 | 23.38 | 15.97 | 20.98 | 17.16 |
| -1xIAD | 4.15 | 47.38 | 14.89 | 9.33 | 6.78 | 17.46 |
| -3xSYD | 1.47 | 15.70 | 10.61 | 4.10 | 9.41 | 58.71 |
| -3xCDG | 55.49 | 15.04 | 8.86 | 9.12 | 10.17 | 1.31 |
| 1xLHR | 15.14 | 17.18 | 7.94 | 17.45 | 22.63 | 19.65 |
| LHRnoPeer | 15.84 | 20.19 | 3.03 | 17.79 | 23.69 | 19.47 |
| 3xLHR | 15.92 | 21.08 | 1.97 | 17.81 | 23.62 | 19.60 |
| -5xIAD | 0.77 | 88.11 | 1.68 | 3.09 | 2.19 | 4.16 |
| LHRnoExp | 16.78 | 21.82 | 0.00 | 17.75 | 24.23 | 19.42 |

TABLE IV: COOKBOOK for the TANGLED testbed.

To better visualize catchment information for operators, we developed an interactive web interface (Figure 4). This interface allows operators to investigate site catchment and detect client distribution anomalies, such as identifying which anycast sites serve a specific country or AS.

In this case, Figure 4 shows which anycast sites are preferred by US clients (*R1*) in the baseline situation (Figure 3). We select US clients based on geolocation of the source IP addresses hitting each site (*R3*). The leftmost (green) box shows all TANGLED sites and the US being selected as catchment source. The middle box (blue) shows that 40.28% of US customers prefer the US-MIA site, 33.11% BR-POA, 16.02% US-IAD, 8.60% UK-LHR, and 1.98% are distributed over other sites. At this point, an operator may wonder why so many clients from the US get routed to a site in Brazil. The operator then selects BR-POA in the middle box, which triggers a list of autonomous systems to be shown in the rightmost box (red). This rightmost box shows the top 5 ASes from the US hitting BR-POA, showing that 38.3% of them are from AS7922 (Comcast), 9.49% from AS20115, 5.81% from AS10796, 5.34% from AS20001, and 4.54% from AS11427 (the last four ASes all belong to Charter Communications).

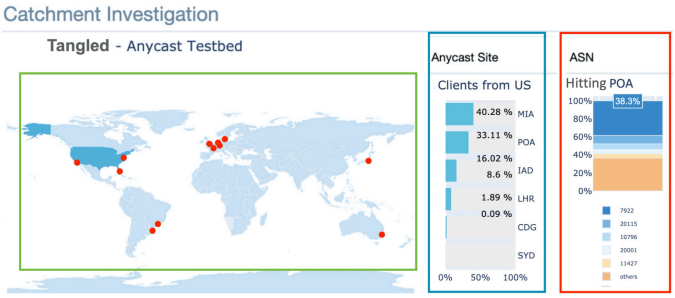


Fig. 4: Dashboard for catchment inspection and visualization.

An operator may then wish to remediate this issue and bring the US traffic hitting sites in Brazil back to the US. By using BGP ANYCAST TUNER, operators can experiment with different routing policies, something that is a tedious and time-consuming manual process without the intuitive interface of BGP ANYCAST TUNER on Figure 3 (*R2, R6*).

While further policy refinement may be needed to optimize the last fraction of clients getting misrouted, this scenario shows the how BGP ANYCAST TUNER can make the task of troubleshooting anycast performance easier for operators.

### D. Scenario 2: Moving Traffic

In this scenario, we demonstrate how we accomplish requirements *R2*, *R4*, *R5*, and *R6*, by showing how an operator can move as much traffic as possible to a specific location.

Here, we mapped a set of policies to deploy specific action. This mapping allows operators to move traffic just moving one slider in Figure 3. For example, if the slider for Paris (CDG) is moved all the way up, as much traffic as possible is attracted to this site. The sliders for the other sites also change and the histogram shows that the majority of clients will now move to the catchment of the site in Paris. Doing this, it fulfils requirements about mapping (*R1*), control (*R2*), and an easy-to-use interface (*R6*).

Notice that the prototype interface also provides a drop-down menu on the left-hand side of the interface in Figure 3. This drop-down menu allows operators to pick a goal, *e.g.*, "Bring traffic to Europe". These goals correspond to preset slider positions and thus to a preset combination of BGP policies among the ones presented. This feature alows a fast traffic engineering deployment, thus fulfilling (*R5*).

## VI. CONCLUSIONS

The use of anycast is becoming increasingly common and important; every year more service providers use it to improve the robustness of their services and the user's quality-of-experience. In this paper, we introduced a new approach to address these problems. In our approach, we register all routing policies available at each site in an anycast service. We then measure and map site catchments for each routing policies. This mapping can predict the most optimal traffic engineering scenarios to be implemented on anycast sites.

We then prototype a set of tools to assist operators to monitor and manage catchment of their anycast network in real-time. The core of our approach is an intuitive web interface that we call the BGP ANYCAST TUNER. This interface eases the burden of managing an anycast service for operators. We tested and showcased the prototype on a global anycast testbed spanning 12 sites on 5 continents. In two case studies, we show how the BGP ANYCAST TUNER can be used to identify sub-optimal routing of traffic (to a remote site) and how traffic on a live anycast service can be shifted to a desired location with a single mouse click.

We publish all tools under an open source license in a public repository[1]. As a next step, we are working with the operations team of SIDN, the operator of the `.nl` ccTLD, to deploy our approach on their production anycast service.

## VII. ACKNOWLEDGEMENTS

This work is funded by the Netherlands Organisation for Scientific Research NWO (628.001.029) and CONCORDIA (830927). This work was also supported by SIDN Labs and NLnet Labs through the project Self-managing Anycast Networks for the DNS (SAND). We also thank to John Heidemann for valuable feedback during the project.

## REFERENCES


[1] C. Partridge, T. Mendez, and W. Milliken, "Host anycasting service," Internet Requests for Comments, RFC Editor, Tech. Rep. 1546, 1993.

[2] D. McPherson, D. Oran, D. Thaler, and E. Osterweil, "Architectural Considerations of IP Anycast," IETF, RFC 7094, Jan. 2014.

[3] H. Ballani and P. Francis, "Towards a Global IP Anycast Service," in *Computer Communication Review*, vol. 35, 2005, Conference Proceedings, pp. 301–312.

[4] M. J. Freedman, K. Lakshminarayanan, and D. Mazières, "OASIS: Anycast for Any Service," in *NSDI*, vol. 6, 2006, Conference Proceedings.

[5] P. Gilmore, "The Akamai Blog," https://blogs.akamai.com/2013/04/serving-at-the-edge-good-for-performance-good-for-mitigating-ddos-part-ii.html, Apr 2013.

[6] G. C. M. Moura, R. de O. Schmidt, J. Heidemann, W. B. de Vries, M. Müller, L. Wei, and C. Hesselman, "Anycast vs DDoS: Evaluating the November 2015 root DNS event," in *Proceedings of the ACM Internet Measurement Conference*, Nov. 2016.

[7] J. Nazario, "DDoS Attack Evolution," *Network Security*, vol. 2008, no. 7, pp. 7–10, 2008.

[8] A. Rizvi, J. Ceron, L. Bertholdo, and J. Heidemann, "Anycast agility: Adaptive routing to manage DDoS," *arXiv preprint arXiv:2006.14058*, 2020.

[9] S. Sarat, V. Pappas, and A. Terzis, "On the Use of Anycast in DNS," in *Proceedings of 15th International Conference on Computer Communications and Networks*. IEEE, 2006, pp. 71–78.

[10] W. B. de Vries, "Improving Anycast with Measurements," Ph.D. dissertation, University of Twente, Netherlands, 2019.

[11] D. Cicalese and D. Rossi, "A Longitudinal Study of IP Anycast," *ACM SIGCOMM Computer Communication Review*, vol. 48, no. 1, 2018.

[12] D. Cicalese, J. Auge, D. Joumblatt, T. Friedman, and D. Rossi, "Characterizing IPv4 Anycast Adoption and Deployment," p. Article 16, 2015.

[13] Y. Rekhter, T. Li, and S. Hares, "A border gateway protocol 4 (bgp-4)," Internet Requests for Comments, RFC Editor, Tech. Rep. 4271, 2006.

[14] R. de Oliveira Schmidt, J. Heidemann, and J. H. Kuipers, "Anycast Latency: How Many Sites Are Enough?" in *International Conference on Passive and Active Network Measurement*. Springer, 2017.

[15] W. B. De Vries, R. Van Rijswijk-Deij, P.-T. de Boer, and A. Pras, "Passive Observations of a Large DNS Service: 2.5 Years in the Life of Google," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 190–200, 2020.

[16] W. B. de Vries, R. d. O. Schmidt, W. Hardaker, J. Heidemann, P.-T. de Boer, and A. Pras, "Verfploeter: Broad and load-aware anycast mapping," Technical Report ISI-TR-719, USC/Information Sciences Institute, Tech. Rep., 2017.

[17] A. B. Brown and J. L. Hellerstein, "An approach to benchmarking configuration complexity," in *Proceedings of the 11th Workshop on ACM SIGOPS European Workshop*, ser. EW 11. New York, NY, USA: Association for Computing Machinery, 2004, p. 18–es.

[18] A. B. Brown, A. Keller, and J. L. Hellerstein, "A model of configuration complexity and its application to a change management system," in *2005 9th IFIP/IEEE International Symposium on Integrated Network Management, 2005. IM 2005.*, 2005, pp. 631–644.

[19] D. Woods, *Behind Human Error*. Ashgate, 2010. [Online]. Available: https://books.google.nl/books?id=NqeTXB9XDsQC

[20] J. Graham-Cumming, "Cloudflare outage on july 17 2020," https://blog.cloudflare.com/cloudflare-outage-on-july-17-2020/, 07 2020, (Accessed on 07/20/2020).

[21] Z. Liu, B. Huffaker, M. Fomenkov, N. Brownlee *et al.*, "Two Days in the Life of the DNS Anycast Root Servers," in *International Conference on Passive and Active Network Measurement*. Springer, 2007.

[22] X. Fan, J. Heidemann, and R. Govindan, "Evaluating Anycast in the Domain Name System," in *2013 Proceedings IEEE INFOCOM*. IEEE, 2013, pp. 1681–1689.

[23] W. B. de Vries, R. d. O. Schmidt, and A. Pras, "Anycast and Its Potential for DDoS Mitigation," in *IFIP International Conference on Autonomous Infrastructure, Management and Security*. Springer, 2016, pp. 147–151.

[24] J. H. Kuipers, "Anycast for DDoS," https://essay.utwente.nl/73795/1/Kuipers_MA_EWI.pdf, 2017, [Online; accessed 5-May-2020].

[25] C. Hesselman, G. C. Moura, R. d. O. Schmidt, and C. Toet, "Increasing DNS Security and Stability Through a Control Plane for Top-level Domain Operators," *IEEE Communications Magazine*, vol. 55, no. 1, pp. 197–203, 2017.

[26] H. Ballani, P. Francis, and S. Ratnasamy, "A Measurement-based Deployment Proposal for IP Anycast," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, 2006, pp. 231–244.

[27] L. Wei and J. Heidemann, "Does Anycast Hang Up on You?" in *Network Traffic Measurement and Analysis Conference (TMA)*. IEEE, 2017.

[28] W. B. de Vries, R. de O. Schmidt, W. Hardaker, J. Heidemann, P.-T. de Boer, and A. Pras, "Broad and Load-Aware Anycast Mapping with Verfploeter," in *Proceedings of the 2017 Internet Measurement Conference*, ser. IMC '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 477–488.

[29] W. B. de Vries, S. Aljammāz, and R. van Rijswijk-Deij, "Global-Scale Anycast Network Management with Verfploeter," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*, 2020.

[30] S. McQuistin, S. P. Uppu, and M. Flores, "Taming Anycast in the Wild Internet," in *Proceedings of the Internet Measurement Conference*, ser. IMC '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 165–178.

[31] L. Wei, M. Flores, H. Bedi, and J. Heidemann, "Bidirectional Anycast/Unicast Probing (BAUP): Optimizing CDN Anycast," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*.

[32] C. Wiefferink, "Visualizing the performance of an anycast network," B.S. thesis, University of Twente, July 2019. [Online]. Available: http://essay.utwente.nl/78642/

[33] L. Costella, M. Trentin, and R. Schmidt, "CatchmentView: Uma Ferramenta para a Análise e Comparação de IPv4 e IPv6 Catchments," in *Anais Estendidos do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. Porto Alegre, RS, Brasil: SBC, 2019, pp. 49–56. [Online]. Available: https://sol.sbc.org.br/index.php/sbrc_estendido/article/view/7769

[34] N. Feamster, J. Borkenhagen, and J. Rexford, "Guidelines for interdomain traffic engineering," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 5, p. 19–30, 2003.

[35] Staff, RIPE NCC, "RIPE Atlas: A Global Internet Measurement Network," *Internet Protocol Journal*, vol. 18, no. 3, 2015.

[36] R. Gao, C. Dovrolis, and E. W. Zegura, "Interdomain Ingress Traffic Engineering through Optimized AS-path Prepending," in *International Conference on Research in Networking*. Springer, 2005, pp. 647–658.

[37] R. Chandra, P. Traina, and T. Li, "Bgp communities attribute," Internet Requests for Comments, RFC Editor, Tech. Rep. 1997, 1996. [Online]. Available: https://www.rfc-editor.org/rfc/rfc1997.txt

[38] G. C. Moura, J. Heidemann, W. Hardaker, J. Bulten, J. Ceron, and C. Hesselman, "Old but gold: Prospecting tcp to engineer dns anycast (extended)," Tech. Rep. ISI-TR-740, USC/Information Sciences Institute, Tech. Rep., 2020.

[39] Z. Li, D. Levin, N. Spring, and B. Bhattacharjee, "Internet anycast: Performance, problems, & potential," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*.

[40] J. Borkenhagen, R. Bush, R. Bonica, and S. Bayraktar, "Policy Behavior for Well-Known BGP Communities," Internet Requests for Comments, RFC Editor, Tech. Rep. 8642, 2019.

[41] G. Huston, "Commentary on Inter-Domain Routing in the Internet," Internet Requests for Comments, RFC Editor, Tech. Rep. 3221, 2001.

[42] M. Luckie, B. Huffaker, A. Dhamdhere, V. Giotsas, and K. Claffy, "AS Relationships, Customer Cones, and Validation," in *Proceedings of the 2013 Internet Measurement Conference*, 2013, pp. 243–256.

[43] CAIDA, "CAIDA AS Rank," http://as-rank.caida.org/, 10 2019, [Online: accessed 26-jun-2020).

[44] L. M. Bertholdo, J. M. Ceron, W. B. de Vries, R. d. O. Schmitt, L. Zambenedetti Granville, R. van Rijswijk-Deij, and A. Pras, "Tangled: A Cooperative Anycast Testbed," *arXiv e-prints*, p. arXiv:2008.12881, Aug. 2020.

[45] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, "Delayed Internet Routing Convergence," *ACM SIGCOMM Computer Communication Review*, vol. 30, no. 4, pp. 175–187, 2000.

[46] R. Teixeira, S. Uhlig, and C. Diot, "BGP Route Propagation between Neighboring Domains," in *International Conference on Passive and Active Network Measurement*. Springer, 2007, pp. 11–21.

[47] R. B. d. Silva and E. S. Mota, "A Survey on Approaches to Reduce BGP Interdomain Routing Convergence Delay on the Internet," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, 2017.