Anycast Agility: Network Playbooks to Fight DDoS

A S M Rizvi∗ Leandro Bertholdo∗ João Ceron John Heidemann
USC/ISI University of Twente SIDN Labs USC/ISI

Abstract

IP anycast is used for services such as DNS and Content Delivery Networks (CDN) to provide the capacity to handle Distributed Denial-of-Service (DDoS) attacks. During a DDoS attack service operators redistribute traffic between anycast sites to take advantage of sites with unused or greater capacity. Depending on site traffic and attack size, operators may instead concentrate attackers in a few sites to preserve operation in others. Operators use these actions during attacks, but how to do so has not been described systematically or publicly. This paper describes several methods to use BGP to shift traffic when under DDoS, and shows that a response playbook can provide a menu of responses that are options during an attack. To choose an appropriate response from this playbook, we also describe a new method to estimate true attack size, even though the operator’s view during the attack is incomplete. Finally, operator choices are constrained by distributed routing policies, and not all are helpful. We explore how specific anycast deployment can constrain options in this playbook, and are the first to measure how generally applicable they are across multiple anycast networks.

1 Introduction

Anycast routing is used by services like DNS or CDN where multiple sites announce the same prefix from geographically distributed locations. Defined in 1993 [49] anycast was widely deployed by DNS roots in the early-2000s [4, 29, 64], and today it is used by many DNS providers and Content Delivery Networks [16, 17, 24, 26, 80].

In IP anycast, BGP routes each network to a particular anycast site, dividing the world into catchments. BGP usually associates networks with nearby anycast sites, providing generally good latency [62]. Anycast also helps during Distributed-Denial-of-Services (DDoS) attacks, with each site adds to the aggregate capacity at lower cost than a single very large site. Each site is independent, so should DDoS overwhelm one site, sites that are not overloaded are unaffected.

DDoS attacks are getting larger and more common. Different root servers and anycast services frequently report DDoS events [18,42,47,48]. Different automated tools make it easier to generate attacks [81], and some offer DDoS-as-a-Service, allowing attacks from unsophisticated users for as little as US$10 [68]. DDoS intensity is still growing, with the 2020 CLDAP attack exceeding 2.3 Tb/s in size [66], and the 2021 VoIP.ms attack lasting for over 5 days [50, 65]. The reservoir of attack sources grow with millions of Internet-of-Things devices whose vulnerabilities fuel botnets [35].

Operators depend on anycast during DDoS attacks to provide capacity to handle the attack and to isolate attackers in catchments. Service operators would like to adapt to an ongoing attack, perhaps shifting load from overloaded sites to other sites with excess capacity. Prior studies of DDoS events have shown that operators take these actions but suggested that the best action to take depends on attack size and location compared to anycast site capacity [45]. While prior work suggested countermeasures, and we know that operators alter routing during attacks, to date there has been only limited evaluation of how routing choices change traffic [4,27,36,52]. Only very recent work examined path poisoning to avoid congested paths [70]; there is no specific public guidance on how to use routing during an attack.

The goal of this paper is to guide defenders in traffic engineering (TE) to balance traffic across anycast during DDoS.

Our first contribution is a system with novel mechanism to estimate true attack rate and plan responses. First, we propose a new mechanism to estimate the true offered load, even when loss happens upstream of the defender. Estimating the relative load on each site (§3.3) is the first step of defense, so that the defender can match load to the capacities of different sites, or decide that some sites should absorb as much of the attack as possible. Second, we develop a BGP playbook: a guide that allows operators to anticipate how TE actions rebalance load across a multi-site anycast system. Together, these two elements provide a system that can automate response to DDoS attacks by adjusting anycast routing according to the playbook, or recommend actions to a human operator.

The second contribution is to understand how well routing options for multi-hop TE work: AS prepending, community strings and path poisoning. While well known, it is not widely understood how available and effective these mechanisms are. In §6 we show that while AS prepending is available almost anywhere, community strings and path poisoning support varies widely. We also show that their effectiveness varies greatly, in part because today’s “flatter Internet” [15] means AS prepending often shifts either nearly all or nearly no traffic. Community strings provide finer granularity control, but we

∗Shared first author
show their support is uneven. Path poisoning may provide control multiple hops away, but like community strings it is often filtered, particularly for Tier-1 ASes. When these factors combine with the interplay between multiple sites and an anycast system, a BGP playbook is important to guide defenders. Since the effects of TE are often specific to the peers and locations of a particular anycast deployment, we explore how sensitive our results are to different locations and numbers of anycast sites (§7).

Our final contribution is to demonstrate successful defenses in practice. We replay real-world attacks in a testbed and show TE can defend (§8). Of course no single defense can protect against all attacks, these examples show our approach provides a successful defense to many volumetric and polymorphic DDoS attacks. They show that our algorithm and process contributions (attack size estimation and playbook construction) have practical application.

Our work uses publicly available datasets. Datasets for the input and results from our experiments are available at no charge. Because our data concerns services but not individuals, we see no privacy concerns.

2 Related Work

Anycast routing has been studied for a long time from the perspective of routing, performance, and DDoS-prevention.

**BGP to steer traffic:** Prior work showed BGP is effective to steer traffic to balance load on links [8,27,53]. However, Ballani et al. showed that anycast requires planning and care for effective load balancing [4]. Others proposed to manipulate BGP based on packet loss, latency and jitter [46,52]. We build on Ballani’s recommendation to plan anycast, proposing a BGP playbook, and studying how well it can work.

Chang et al. [14] suggested using BGP Communities for traffic engineering [10,13,74]. Recent work has examined BGP communities for blackhole routing in IXPs and ISPs [21,28]. Smith and Glenn examined path poisoning to address link congestion [70]. While each of these are important options in routing for defense, we show a system that guides the operator to select between them. A system with multiple choices is necessary because no single method works against all attacks. For example, we show path poisoning does not work when we poison a Tier-1 AS.

**Anycast performance:** Most anycast research focused on efficient delivery and stability [11,39,40,59,79]. Later studies explicitly investigate the proximity of the clients [4,11,39].

Some studies try to improve anycast through topology changes [44,62]. Anycast services for DDoS is already used in commercial solutions e.g., Amazon [63], Akamai [75] and AT&T [72]. However, none of them address how to use routing manipulations as a DDoS defense mechanism.

**Anycast catchment control as a DDoS mitigation tool:** To our knowledge, the idea of handling DDoS attacks by absorbing or shifting load across anycast sites was first published in 2016 [45]. Kuipers et al. [36] refined that work, defining the traffic shifting approaches that we review in §3.4 and explore through experiment. We develop the idea of a BGP playbook to guide responses, and describe a new approach to estimate attack size, and finally show that responses can be effective with real-world events.

**Commercial and automated solutions:** Most published commercial anti-DDoS solutions use routing to steer traffic towards a mitigation infrastructure [22]. Sometimes there is a requirement for all the sites to be connected through a private backbone to support traffic analysis [63]. Another defense uses BGP to divert all traffic to a scrubbing center, then tunnels good traffic to the destination [69]. Other methods use DNS manipulation [12], or anycast proxies [30] which cannot be used in DNS anycast deployments itself. Rather than outsourcing the problem, we explore how one can defend it. Other automated defenses include responsive resource management [25], client-server reassignment [34], and filtering approaches [58]. Our method uses TE approaches to efficiently use available resources in anycast.

3 Mechanisms to Defend Against DDoS

In this section we describe our BGP mitigation process; how we pre-compute a BGP playbook, estimate the attack size and select a TE response.

3.1 Overview and Decision Support

In Figure 1 we show how defense against DDoS works. Defense against a DDoS begins with detection (2), then defenders plan a defense (4), carry it out (5), and repeat this process until the attack is mitigated or it ends (bottom cycle in Figure 1). Detecting the attack is straightforward, since large attacks affect system performance. The challenge is selecting the best response and quickly iterating.

We bring two new components to attack response (colored light green in Figure 1): mapping before the attack, and estimating attack size when the attack begins. Mapping (1) (discussed in §3.2) provides the defender with a playbook of planned responses and the information about how they will change the traffic mix across their anycast system. Size estimation (3) (discussed in §3.3) allows the defender to determine how much traffic should be moved and select a promising response from the playbook. Together, these tools help to

![Figure 1: Overview of the decision process.](image-url)
understand not only how to reduce traffic at a given site, but also the sites where that traffic will go.

These components come together in our automated response system (§3.4) that iterates between measurement and attack size estimation, defense selection, then deployment. Defense uses the playbook built during mapping; we provide an example playbook in §6.4. We show how these defenses operate in testbed experiments in §8.

Our system is designed for services that operate with a fixed amount of infrastructure on specific anycast IP addresses and do not employ a third-party scrubbing service. Operators of CDNs with multiple anycast services, DNS redirection, or scrubbing services may use our approach, but also have those other tools. However, many operators cannot or prefer not to use scrubbing and DNS redirection: all operators of single-IP services (all DNS root servers), many ccTLDs who value national autonomy, and scrubbing services themselves. Our approach defends against volumetric attacks where we have spare capacities in other sites. Since DDoS causes unavailability of services, suboptimal site selection during an attack is not a concern.

3.2 Measurement: Mapping Anycast

We map the catchments of anycast service before an attack so that the defender can make an informed choice quickly during an attack, building a BGP playbook (§6.4).

To map anycast catchments we used Verfploeter [20]. As an active prober (ICMP echo request), Verfploeter observes the responses of all ping-responsive IPv4 /24s and maps which anycast site receives the responses. We provide a detailed description of anycast and Verfploeter in Appendix A. Since mapping happens before the attack, mapping speed is not an issue.

Alternatively, we can map traffic by observing which customers are seen at each site over time, or measuring from distributed vantage points such as RIPE Atlas [3, 73]. (Operators may already collect this information for optimization.)

Mapping should consider not only the current catchments but also potential shifts we might make during the attack. This full mapping is easy to do with Verfploeter, which can be continuously running in an adjacent BGP prefix to map the possible shifts. This mapping process is important to anticipate how traffic may shift. We will show later that BGP control is limited by the granularity of routing policy (§6) and by the deployment of the anycast sites (§7).

A challenge in pre-computed maps with routing alternatives is that routing is influenced by all ASes. Thus, the maps may shift over time due to changes in the routing policies of other ASes. Fortunately, prior work shows that anycast catchments are relatively slow to change [79]. We also show that our BGP playbook is stable over time (§6.4 and Appendix E).

3.3 Estimation of the Attack Size

After the detection of an attack, the first step in DDoS defense is to estimate the attack size, so we can then select a defense strategy of how much traffic to shift. Our goal is to measure offered load, the traffic that is sent to (offered to) each site. During DDoS offered load balloons with a mix of attack and legitimate traffic, and loss upstream of the service means we cannot directly observe true offered load. We later evaluate our approach with real-world DDoS events (§4).

Idea: Our insight is that we can estimate true offered load based on changes in some known traffic that actually does arrive at the service, even when there is upstream loss.

To know how much offered load actually arrives at the service, we need to estimate some fraction of legitimate traffic. We can then observe how much this traffic drops during the attack, inferring upstream loss. Unfortunately, there is no general way to determine all legitimate traffic, since legitimate senders change their traffic rates, and attackers often make their traffic legitimate-appearing. Our goal is to reliably estimate some specific legitimate traffic: we describe several sources next.

Traffic sources: There are several possible sources of known legitimate traffic—we consider known measurement traffic and regular traffic sources that are heavy hitters [5].

For DNS, our demonstration application, RIPE Atlas provides a regular source of known-good traffic, sent from many places. RIPE makes continuous traffic from around 10k publicly available vantage points [55]. Each RIPE vantage point queries every 240 s, and there is enough traffic (about 2500 queries/minute) to provide a good estimate of offered load. (Although RIPE Atlas is specific to DNS, other commercial services often have similar types of known monitoring traffic.)

To find the known-good traffic at each site, we use the catchments of RIPE vantage points with pre-deployed RIPE DNS CHAOS queries (one exists for each root DNS IP, such as measurement ID 11309 for A-root). We can also use Verfploeter or captured traces in the anycast sites. An advantage of using RIPE traffic is that it does not place any new load on the service.

Heavy hitters can provide an additional source of known-good traffic. Many services have a few consistently large-volume users with regular traffic patterns, and while they vary over time, many are often stable. For DNS, we find that most heavy hitters have a strong diurnal variation in rate; we model them with TBATS (Trigonometric seasonality, Box-Cox transformation, ARMA errors, Trend and Seasonal) [19] to factor out such known variation. While an adversary could spoof heavy hitters, that requires a large and ongoing investment to succeed.

Estimation: Our goal is to estimate offered load, $T_{offered}$. We can measure the observed traffic rate, $T_{observed}$, at the access link. We define $\alpha$ as the access fraction—the fraction of traffic that is not dropped. Therefore $T_{observed} = \alpha \cdot T_{offered}$.

To estimate the access fraction ($\alpha$), we observe that known good traffic has the same loss on incoming links as does other good traffic and attack traffic. We estimate the known
traffic rate (from RIPE Atlas measurement traffic, or from heavy hitters, or both), as $T_{\text{known}}$. Then $\alpha \cdot T_{\text{known, offered}} = T_{\text{known, observed}}$ and our estimate of offered load is $\hat{T}_{\text{offered}} = T_{\text{observed}} \cdot T_{\text{known, offered}} / T_{\text{known, observed}}$.

### 3.4 Traffic Engineering as a Defense Strategy

With knowledge of the offered load, the defender can select an overall defense strategy that will drive traffic engineering decisions. The defender first must determine if the attack exceeds overall capacity or not.

For attacks that exceed overall capacity, the defender’s goal is to preserve successful service at some sites, while allowing other sites to operate in degraded mode as absorbers [45]. The defender may also choose to shift traffic away from some degraded sites to ease their pain. Unloading the overloaded sites is recognized as breakwaters [36].

For moderate-size attacks, the defender should try to serve all traffic, rebalancing to shift traffic from overloaded sites to less busy sites. In heterogeneous anycast networks, where some sites have more capacity than others, the defense approach can be different. In these cases, larger, “super”-sites can attract traffic from smaller sites. For moderate-size attacks, it may even be best for smaller sites to shut down if the super-sites can handle the traffic.

Regardless of attack size, traffic engineering allows the defender to shift attack traffic to absorber or breakwater sites. We next describe traffic engineering options, and then how one can automate response. For operators unwilling to fully automate response, our system can still provide recommendations for possible actions and their consequences.

#### 3.4.1 Traffic Engineering to Manage an Attack

Given an overall defense strategy (absorb or rebalance), the defender will use traffic engineering to shift traffic, either automatically (§3.4.2) or as advice under operator supervision. For anycast deployments connected by the public Internet, BGP [8] will be the tool of choice to control routing and influence anycast catchments. Organizations that operate their own wide-area networks may also be able to use SDN to manage traffic on their internal WAN [31, 61]. Fortunately, BGP has well established mechanisms to manage routing policy. We use three BGP mechanisms in the paper: AS-Path prepending, BGP communities and Path Poisoning.

**AS-Path Prepending** is a way to de-prefer a routing path, send traffic to other catchments. BGP’s AS-Path is the list of ASes back to the route originator. The AS-Path both prevents routing loops and also serves as a rough estimate for distance, with BGP preferring routes with shorter AS-Paths. By artificially inserting extra ASes into the AS-Path, the route originator can de-prefer one site in favor of others. Path prepending is known to be a coarse routing technique for traffic engineering. We measure how fine the control AS-Path prepending provides to anycast in §6.1.

We define **Negative Prepending** as the use of AS-Path prepending to draw traffic towards a site, preferring one site over others. Prepending can only increase path lengths, but an anycast operator in control of all anycast sites can prepend at all sites except one, in effect giving that site a shorter AS-Path (relative to the other sites) than it had before. “Negative prepending by one at site S” is, therefore, shorthand for prepending by one at all sites other than S.

Long AS-Paths due to prepending can make prefixes more vulnerable to route hijacking [41]. However, this issue has a small impact on anycast prefix, as always there is a site announcing without any prepend, keeping the path length limited. We suggest that formal defenses to hijacking such as RPKI are needed even without prepending, and when they are in place, prepending can be an even more valuable tool for TE.

**BGP Communities** (or community strings) label specific BGP routes with 32 or 64 bits of information. How this information is interpreted is up to the ASes. While not officially standardized, a number of conventions exist where part of the information identifies an AS and the other part a policy such as blackholing, prepend, or set local-preference. Community strings are widely supported to allow ISPs to delegate some control over routing policy to their customers [1, 74].

**Path Poisoning** is another way to control the incoming traffic. This technique consists of adding the AS of another carrier to the AS PATH. Paths that repeat ASes in different parts of the AS PATH indicate routing loops and must be discarded by BGP.

When using path poisoning we announce a path with both the poisoned AS and own AS (otherwise neighbors may filter our announcement as not from us). We must therefore also prepend twice at all other anycast sites, otherwise poisoning also results in a longer AS path.

Figure 2 shows how traffic engineering can be applied to anycast systems in order to modify the catchment. In this example, site-1 is overwhelmed by an attack. Aiming to shift bins of traffic to site-2 with spare capacity, we can make BGP announcements. site-1 poisons AS3, prepends (only showing to AS4), and prevents announcement to AS5 using *not-export* BGP community. These changes decrease load in site-1, shifting the traffic to site-2.

#### 3.4.2 Automatic Defense Selection

To automate defense we use a centralized controller. The controller collects observations for all sites (from external measurements, or assuming the site is saturated if it cannot
We discussed our approach with operators of root DNS and were enthusiastic about automated defenses to deal with common DDoS events, and to handle events during non-business hours, some operators prefer human-supervised (non-automated) response, and all expected human supervision of response during initial deployment to build trust before full automation.

To support human-supervised response, we design an operator assistance system as an alternative (or precursor) to automation. This system provides a web-based interface that activates route changes, coupled with playbook lookup that recommends good options based on current sensor status (Figure 3). Details are described in Appendix B.

4 Evaluation of Offered Load Estimation

We next evaluate estimating offered load with real-world events; a testbed evaluation is in Appendix C.

4.1 Case Studies

We test our approach with two large DNS DDoS events from 2015-11-30 and 2016-06-25. The November 2015 event was a DNS flood, and the June 2016 was a SYN and ICMP flood attack. B-root exhibited significant upstream loss in both these events, so we estimate true offered load to B-root and compare to observations at other roots for ground truth.

To apply our system we measure the access fraction ($\alpha$) using the known-good traffic. Table 1 shows the expected typical known-good traffic (“normal”), the observed rate under attack (“observed”) and the computed $\alpha$. Here we use RIPE Atlas as known good [54]. We see similar results (omitted due to space) when using the top 100 heavy hitters.

Figure 4 compares the observed load (the bottom blue line) with the estimated offered load (the middle, varying, orange line) from our system, as compared to the attack rate reported from other roots (the dashed purple line). The offered load columns of Table 1 give numeric values.

Even though the attack was large, we see that the estimated attack size of the 2015 event of 4–6.5 Mq/s is close to the reported 5.1 Mq/s [45,47]. We also see similar results from the 2016 event [48], where we estimate 8–11 Mq/s of total traffic, compared to the 10 Mq/s reported rate (details with figure in §C.2). We also add the result from Testbed experiment which also shows a good accuracy (details in §C.1).

These two events show that even with high rates of upstream loss we are able to get reasonable estimates of total offered load. Our results provide good accuracy when the known-good traffic has 2500 queries/minute with RIPE, and additional known-good traffic can improve accuracy. Use of additional known-good traffic (such as heavy hitters) improves accuracy in these cases by providing a larger signal. However, in practice, even a rough estimation allows a far better response than using directly observed load.

We conclude that attack size estimation is close enough to help plan response to DDoS events.
5 Evaluation Approach

We next describe how we will evaluate the effectiveness of TE (§6) and that results generalize to different deployments (§7). Traffic engineering in response to DDoS depends on the anycast deployment—where sites are and with whom they peer. We evaluate on two different testbeds. Our approach (estimation, TE, and playbook construction) can be applied anywhere with different anycast setups. We expect network operators will execute our approaches on a test prefix (in parallel with their operational network) prior to an event so that no service interruption happens.

5.1 Anycast Testbeds

We evaluate our ideas on testbeds to see the constraints of real-world peering and deployments. We use two independent testbeds: Peering [60] and Tangled [7]. Table 2 summarizes information about each testbed with their own set of geographically distributed sites along with their locations (Peering supports more sites but we used 8 sites). These sites show different connectivity, and have one or more transits and IXP peers. Most Peering sites have academic transits while Tangled has more commercial providers. Our testbed is about the same size as many operational networks, since nearly half of real-world networks have five or fewer sites [16].

<table>
<thead>
<tr>
<th>Testbed</th>
<th>Used Sites</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peering</td>
<td>Amsterdam*† (AMS), Boston* (BOS), Belo Horizonte*† (CNP), Seattle* (SEA), Athens* (ATH), Atlanta* (ATL), Salt Lake City* (SLC), Wisconsin* (MSN)</td>
<td>8</td>
</tr>
<tr>
<td>Tangled</td>
<td>Miami (MIA)<em>, London (LHR)</em>, Sydney (SYD)<em>, Paris (CDG)</em>, Los Angeles (LAX)<em>, Enschede (ENS)</em>, Washington (IAD)<em>, Porto Alegre (POA)</em>†</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 2: Testbed and respective sites used in our experiments. Transit providers (*) and IXP (†).

5.2 Measuring Routing Changes

To measure the effect of a BGP change, we first change the routing announcement at a site, give some time to propagate, confirm that the announcement is accepted, and finally start the anycast measurement.

Route convergence: After a change, we allow some time for BGP route propagation. We know that routing and forwarding tables can be inconsistent (resulting in loops or black holes) while prefix is updating [37, 67, 76]. Although routing updates are usually stable within 5 minutes [67], we wait 15 minutes for routing to settle when building our playbook since it is a non-attack period. When the attack is not mitigated after deploying a routing policy, our system moves to a different approach after 5 minutes.

Propagation of BGP policies: Policy filtering could limit the acceptance of announced routes, although in practice these limits do not affect our traffic engineering. Best practices for networks at the edge to filter out AS-Paths longer than 10 hops, and ASes in the middle often accept up to 50 hops, both more prepends than we need. Based on routing observations from multiple global locations using RIPE RIS, we confirm that configurations in our experiments are never blocked due to route filtering in multi-hops away from our anycast sites.

6 Traffic Engineering Coverage and Control

From an estimate of attack load, operators use BGP to shift traffic. We next evaluate three TE mechanisms: AS-Path prepending, community strings and path poisoning. For each we consider when it works and what degree of control it provides. Table 3 summarizes our key results from tests on two testbeds (§5.1); in §7 we evaluate generalizability.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Key Takeaways</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path prepending</td>
<td>Works everywhere to effectively de-prefer a site (§6.1.2), but shifts traffic in large amounts (§6.1.3), and has few traffic levels (Figure 6).</td>
</tr>
<tr>
<td>Neg. Prepending</td>
<td>Works everywhere to prefer a site (§6.1.2).</td>
</tr>
<tr>
<td>BGP community-</td>
<td>Although widely implemented, well-known communities are not universal (§6.2.1).</td>
</tr>
<tr>
<td>poison-</td>
<td></td>
</tr>
<tr>
<td>BGP path poisoning</td>
<td>When supported, they provide finer-granularity control than prepending (§6.2.2).</td>
</tr>
<tr>
<td></td>
<td>Many Tier-1 ASes drop the announcements when it sees Tier-1 ASes in the paths. (§6.3.1)</td>
</tr>
<tr>
<td></td>
<td>Control over traffic is limited by the filters from other ASes. (§6.3.2).</td>
</tr>
</tbody>
</table>

Table 3: Experiment summarization and findings.
when preends are in place, and we do not observe changes in the routing propagation for both testbeds. Otherwise, this might reveal the existence of AS path length filters [32, 33].

6.1.2 Does prepending work?

Since AS-Path prepending is widely supported, we next evaluate this attractive TE method.

We explore this question for a representative scenario using Peering using three sites from three continents—Europe (Amsterdam-AMS), North America (Boston-BOS) and South America (Brazil-CNF). In §7 we generalize to other configurations. We estimate load by counting /24 blocks in catchments, then compare the baseline with TE options. (We also explored traffic weighted by traffic loads instead of blocks, getting the same qualitative results and shapes with different constants, Appendix F.)

Figure 5 shows the traffic from each site under different conditions. The middle bar in each graph is the baseline, the default condition with no prepending. We then add prepending at each site, with one, two or three preends in each bar going to the right of center. We also consider negative prepending (§3.4.1) in one to three steps, with bars going left of center.

We first consider the baseline (the middle bar) of all three graphs in Figure 5. Amsterdam (AMS, the bottom, maroon part of each bar) gets about 68% of the traffic. AMS receives more traffic than BOS and CNF because that site has two transit providers and several peers, and Amsterdam is very well connected with the rest of the world.

We next consider prepending at each site (the bars to the right of center). In each case, prepending succeeds at pushing traffic away from the site, as expected. For AMS, each prepend shifts more traffic away, with the first prepend cutting traffic from 68% to 37%, then to 29%, then to about 16%. BOS and CNF start with less traffic and prepending has a stronger effect, with one prepend sending most traffic away (at BOS, from 15% to 7%) and additional preends showing little further change. These non-linear changes are because changing BGP routing with prepending is based on path length, and the Internet’s AS-graph is relatively flat [2, 15].

The bar graphs also show that when prepending pushed traffic away from a site, it all goes to some other site. Where it goes depends on routing and is not necessarily proportional to the split in other configurations. For example, after one prepend to AMS, more traffic goes to CNF (the top sky blue bar) than to BOS (the middle yellowish bar). These unexpected shifts are why we suggest pre-computing a “playbook” of routing options before an attack (§3.2) to guide decisions during an attack and anticipate the consequences of a change.

We also see that negative prepending succeeds at drawing traffic towards the site—in each case the bars to the left of center see more traffic in the site that is not prepending while the others prepend. AMS sees relatively little change (68% to 89%) since it already has most traffic, while BOS and CNF each gain up to 68% of traffic.

All three sites show some networks that are “stuck” on that site, regardless of prepending. One reason for this stickiness is when some networks are only routable through one site because they are downstream of that exchange. We confirm this by taking traceroute to two randomly chosen blocks that are stuck at BOS. Traceroutes and geolocation (with Maxmind) confirm they are in Boston, at MIT and a Comcast network (based on the penultimate traceroute hop). We have used the local-preference BGP attribute to move such stuck blocks, but a systematic exploration of that option is future work.

In summary, the experiment shows that AS prepend does work and can shift traffic among sites, however, this traffic shift is not uniform.

6.1.3 What granularity does prepending provide?

Having established that prepending can shift traffic, we next ask: how much control does it provide? This question has two facets: how much traffic can we push away from a site or attract to it, and how many different levels are there between minimum and maximum.

Limits: Figure 5 suggested that in Peering, with those three sites, there is a limit to the traffic that can shift. AMS, BOS, and CNF always get about 16%, 7% and 3% of blocks, regardless of prepending.

Figure 6 confirms this result with a 5-site deployment (two from Europe, one from North America, one from South America and one from Australia) in our other testbed (Tangled). X axis is presented with the number of preends applied to each site. The number zero (0) represents the baseline, the positive numbers (1-5) are the number of prepending applied and the negative numbers represent negative preends. As depicted, each site can capture at most 55–65% of blocks, and can shed at most 95% of blocks, even with up to 5 preends. We can also see that we do not get a granular control as only three points are between the minimum and maximum.

We conclude that while prepending can be a useful tool to shift traffic, it provides relatively limited control.

6.2 Control with BGP Communities

We next show that BGP community strings have the opposite trade-off: what options they support vary from site to site, but when available, they provide more granular control over traffic. We use whatever community strings that can be supported at each site. Specific values for the same concept often vary.

6.2.1 Community string coverage

ASes must opt-in to exchange community strings with peers, as opposed to prepending’s near-universal support (since AS paths are used for loop detection, prepending works unless it is explicitly filtered out). Explicit support is required because communities are only a tagging mechanism; the actions they trigger are at the discretion of peering AS. Prior work has studied the diverse options supported by community strings [28].

To evaluate coverage, we review support for BGP communities in the testbeds we use. The testbeds provide information...
We considered selective announcement options at AMS, with 12 peers compared to the transit-1 in Figure 7. The AMS site has 854 peers (106 bilateral peers including 2 route servers with 748 peers), and 2 transit providers. CNF has one transit provider and 129 peers (with only 6 bilateral peers, other peers are connected through 2 route servers). For our Verfploeter measurement, we consider the peers and route servers with bilateral BGP sessions. A single peer covers a small fraction of the address space in our Verfploeter measurement. For some peers, we observed no coverage at all which requires further investigation with the peers to confirm our observation. Hence, all the selective announcement options do not make difference in the catchment distribution (see the catchment in AMS with 12 peers compared to the transit-1 in Figure 7). The options column of Table 4 summarizes these results, showing how many routing options we have using community strings.

In Table 4 we describe path prepending and poisoning support and what types of community strings are supported at each site. We group communities by class: advertisement options (no-peer, no-export to customers, and no export to anyone), selective prepending, and peers and transits that support selective advertisement. We also show the number of non-transit peers and transits.

Peering allows selective announcement to the transits and peers at each site, although the number of peers and transits varies. Many sites with one transit provide no alternatives. We considered selective announcement options at AMS, with 854 peers (106 bilateral peers including 2 route servers with 748 peers), and 2 transit providers. CNF has one transit provider and 129 peers (with only 6 bilateral peers, other peers are connected through 2 route servers). For our Verfploeter measurement, we consider the peers and route servers with bilateral BGP sessions. A single peer covers a small fraction of the address space in our Verfploeter measurement. For some peers, we observed no coverage at all which requires further investigation with the peers to confirm our observation. Hence, all the selective announcement options do not make difference in the catchment distribution (see the catchment in AMS with 12 peers compared to the transit-1 in Figure 7). The options column of Table 4 summarizes these results, showing how many routing options we have using community strings.

We evaluate Tangled to provide a second deployment with different peers. Tangled built its anycast network over cloud providers, crowd-sourced transit providers and IXP sites. All transit providers and IXP sites support communities as described in Table 4. With Tangled, the POA site has 250 peers and most of them support communities strings.

We conclude that the number of options at each anycast site may vary depending on the number of connections with peers and transits. This uncertainty shows the need for a playbook that shows the possible options.

6.2.2 At what granularity do community strings work?

We next examine how well community strings work and what granularity of control they provide. We use community strings to make BGP selective announcements, where we propagate our route only to specific transit providers or IXP peers.

For our experiment, we use Peering, varying announcements at AMS and observing traffic when anycast is provided from AMS, BOS and CNF (the same topology as §6.1.2). As described in §6.2.1 selective announcement community strings are provided only at AMS and CNF; and they affect our Verfploeter measurement only at AMS with several peers together, two transit ones by one, and route servers.

To select the target ASes for selective announcement, we sort all the working peers of AMS site, based on the size of their customer cone using CAIDA’s AS rank list. We then choose the 6 largest IXP peers and the 12 largest, as the left two bars in Figure 7. We then examine the route server, announced separately (the next bar), and then all IXP peers including route servers. Finally, we see the coverage with each of the two transit providers, announced separately.

First, we see that selective announcement provides more control than prepending, as AMS shifts from baseline 68% of blocks to other configurations from 53 to 6% of blocks.

Second, we see that there is some overlap in some com-
Table 4: Traffic engineering options on each testbed sites. ✓: supported, -: not supported, △: not tested.

<table>
<thead>
<tr>
<th>Site</th>
<th>Routing policy</th>
<th>Peering</th>
<th>Tangled</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AS-path prepend</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AMS</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>BOS</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CNF</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SEA</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ATH</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ATL</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SLC</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MSN</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MIA</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>LHR</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>IAD</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CDG</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>LAX</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ENS</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SYD</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>POA</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td># non-transit peers</td>
<td>854 0 129 0 0 0 0 0 0 0 0 0 250</td>
<td></td>
<td></td>
</tr>
<tr>
<td># transits</td>
<td>2 1 1 1 1 1 1 1 1 1 1 1 1 1 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td># options</td>
<td>856 1 130 1 1 1 1 1 1 1 1 1 1 1 252</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6.3 Control with Path Poisoning

We next turn to path poisoning, and show that like community strings, coverage and granularity are limited by routing filters deployed in upstream peers.

6.3.1 Poisoning coverage

Support for path poisoning is dependent on the ASes we are poisoning and on route filters deployed by our upstream ASes. We find that many ISPs, especially Tier-1 ASes, filter out AS paths that poison any Tier-1 AS. Tier-1 ASes deploy these filters to block BGP announcements from customers that contain other Tier-1 ASes in the path to prevent route leaks [43, 71]. This filtering often makes path poisoning ineffective to control traffic.

To verify that poisoning Tier-1 ASes is often ineffective from filtering, we poison Tier-1 ASes announcing only from AMS in Peering, a unicast set-up blocking the impacts of other sites, and make traceroutes from 1000 RIPE vantage points to our prefix. Our measurement shows the evidence of filters when we poison Tier-1 ASes——AS7018 (AT&T), AS6453 (Tata Communications America), and AS1299 (Telia Company). We observe many vantage points fail to reach our prefix as they are dependent on Tier-1 ASes for their routes. Some others change their paths avoiding Tier-1 ASes. We also validate route disappearance via most Tier-1 ASes using RouteViews telescopes [77].

Although poisoning Tier-1 ASes is often ineffective, poisoning is effective with most non-Tier-1 ASes. Unfortunately, these ASes carry little traffic when they are not immediate upstreams. Poisoning these small ASes only has little impact on traffic. We again traceroute after poisoning a non-Tier-1 AS (AS57866), and observe that Tier-1 ASes propagate the poisoned path. This proves poisoned paths with Tier-1 and non-Tier-1 ASes are treated differently by other ASes.

6.3.2 What granularity does poisoning provide?

Path poisoning coverage is limited because one cannot usually poison a Tier-1 AS. This same filtering limits the granularity that poisoning allows: poisoning Tier-1 ASes is not allowed, poisoning non Tier-1 ASes has little impact when they are multiple hops away because they represent little traffic. Poisoning immediate neighbors may shift traffic, but is more complex than just not announcing to them. We confirm these
AMS is overloaded, the operator could select a TE configuration higher in the table (perhaps ‘e’, ‘g’, or ‘j’). The operator can then see the implications of that TE choice on other site traffic compared to the baseline distribution. We group different levels of prepending (positive or negative) at each site, and show selected community strings and poisoning affect their specific deployment. We show in §7 that the process generalizes. Using a playbook, an operator does not need a single “best” approach, rather a combination of approaches in the playbook ensures a greater control over traffic distribution.

A playbook is a list of variations of routing policy and the resulting traffic distributions. Table 5 shows the playbook for our testbed, with the baseline of 65% blocks to a site shown in white. We group different levels of prepending (positive or negative) at each site, and show selected community string and poisoning configurations.

To summarize the many configurations from Table 5, Table 6 identifies which combinations result in specific traffic ratios at each site. Each letter in this table refers back to a specific configuration from Table 5. During an attack, if the anycast system begins at the baseline configuration (q), if AMS is overloaded, the operator could select a TE configuration higher in the table (perhaps ‘e’, ‘g’, or ‘j’). The operator can then see the implications of that TE choice on other site traffic (for example, ‘e’ increases load on both other sites, with ‘g’ increases load on BOS but decreases it at CNF).

An operator may also use a playbook with traffic load for two reasons. First, loads in most interesting services have diurnal pattern. Second, loads from each /24 prefix may vary because of the number of clients behind each prefix (more on Appendix F). Building the playbook with load is computationally simple; an operator can just use the same catchment mapping along with the per prefix load.

Even with attack size estimation, attacks are accompanied by uncertainty, and attacker locations may be uneven. However, the playbook provides a much better response than “just relying on informal prior experience” in two ways: the defender can anticipate the consequences of the TE action (that traffic will go somewhere!), and the defender can choose between different possible outcomes if the first is incomplete.

Playbook flexibility and completeness: Table 6 helps quantify the “flexibility” that traffic engineering allows us in this anycast deployment. Using these 10% traffic bins, we see that AMS has 9 options, CNF 7, and BOS only 6. Because AMS and CNF mostly swap traffic after TE changes, and because BOS is less well connected, no configuration with three sites allows BOS to take traffic within 50-60% range, and no 3-site configuration can drive BOS or CNF over 70%.

This analysis shows more central sites like AMS, and it may suggest the need for topology changes (perhaps adding another site in Europe or Asia to share AMS’ load).

### 7 Deployment Stability and Constraints

In §6 we showed BGP-based TE provides considerable flexibility. Building playbooks supports defenders by allowing them to explore how transit providers, prepending, community strings, and poisoning affect their specific deployment. We next look at how stable the results are depending on choice of sites and the number of sites. While the details of the playbook vary for each deployment, and we do not claim our testbeds represent all possible deployments, we show our approach is flexible and can respond to attacks in different deployments—our approach generalizes.

#### 7.1 Effects of Choice of Anycast Sites

First we see how sites affect our playbook. New sites change catchments because they depend on location and peering,
In §6.1, we studied catchments with three specific Peering sites on three continents: AMS, at a large, commercial IXP in Europe; CNF with an academic backbone transit in Brazil; and BOS, an academic site in the U.S. We now switch to three educational sites all in the United States: SEA, at University of Washington on the west coast; SLC, at the University of Utah in the Rockies; and BOS, at Northeastern University in Boston on the east coast.

More important than just geographic location, site connectivity is the most important factor in choosing sites. Multiple transit providers increase the chance of having more BGP options to affect traffic control and granularity. While a poorly connected site inside a university network tends to provide less traffic control options.

**Prepending baseline:** Figure 9 shows catchment sizes for the three North American sites with positive and negative prepending. Now the baseline distribution is unbalanced, but less so than before, with SEA capturing 50% of blocks. We discussed SEA’s heavy traffic with the Peering operators. They suspect that SEA is near to the Seattle IXP, making its paths one hop from many commercial providers. Which site has the greatest visibility depends on its peering and will vary from deployment to deployment.

**Prepending coverage and granularity:** As with our prior experiments, we can adjust prepending to see how traffic shifts. With these three sites, traffic shifts very quickly for BOS and SEA after one positive or negative prepend. SLC has more flexibility, perhaps because it has the smallest catchment at the baseline, and gains more coverage with each step of negative prepending, to 42%, 63%, and 91% of blocks. Often (but not always), we see that academic sites exhibit less granularity because either they have few peers, or their peers are academic networks with similar connectivity. As a result, minor changes in AS-Path length place one site further from the others. In addition, this less granular control shows the importance of building a playbook that is specific to a given deployment, or when the anycast topology changes.

**Community coverage:** While communities are common at IXPs and transit providers, academic networks (NRENs) have a more simple set of communities. None of those academic sites provide community strings.

This observation confirms our prior coverage observation: community string support is not uniformly available. We also looked at other combinations of sites in Peering and found similar results (in the extended version of the paper [57]).

**Path poisoning:** We repeated our path poisoning experiments with three sites in Boston, Salt Lake City and Seattle. We confirm that Tier-1 ASes typically cannot be poisoned (§6.3.1). We also see filters designed to prevent route leaks [71] also interfere with poisoning.

Our experiments confirm that while catchments are deployment-specific, our qualitative results hold—prepending works but is coarse, and community strings and poisoning are not supported everywhere.

### 7.2 Effects of Number of Anycast Sites

Next, we vary the number of sites and see how that changes control traffic. We select 3, 5 and 7 sites from each testbed, and build a playbook to evaluate defense options. Figure 10 shows selected configurations, grouped by number of sites.

**Baseline:** With more sites, overall capacity increases and baseline load at each site falls. For example, in Figure 10, the baselines (with an asterisk*) at the largest site (AMS) shifts from 70% of blocks with three sites to 61% and 56% with 5 and 7 sites. Smaller sites shift less (BOS goes from 14% to 6% and 6%, and CNF from 15% to 8% and 6%). Greater capacity and distribution requires a larger and distributed attacker to exhaust the overall service. We see similar results on our alternate testbed Tangled, as described in the extended version of the paper [57].

**Traffic flexibility:** With more sites, the largest site usually shows the largest changes and has the fewest catchment sizes. Comparing baseline to one prepping in Figure 10, AMS shifts from 70% to 37% with three sites, from 61% to 29%
We replay the traffic in simulation, assigning traffic to each server operator, the Dutch National Scrubbing Center, and the traffic in our system. We show that we can successfully respond to a different types of attacks in different ways.

Methodology: We use real-world attacks from B-root server operator, the Dutch National Scrubbing Center, and from an anonymized enterprise network. These events include polymorphic, adversarial, and a volumetric attack.

We evaluate these events by simulating traffic rates against a three-site anycast network. The first two events use Peering with our AMS, BOS, CNF configuration from §6. We vary this topology, using BOS, SEA, SLC from §7.1 in the last event. We replay the traffic in simulation, assigning traffic to each anycast site based on catchments measured in our experiments. We do not simulate the gradual route propagation, but instead have routing take effect 300 s after a change (a conservative bound, most routing changes happen in half that time). We then evaluate traffic levels at each site and compare that to a target capacity.

For each attack we run our system in defense, estimating the attack size and selecting a pre-computed playbook response. Since our playbook allows different responses: when we have choices we select different methods of defense: prepending, negative prepending, or community strings (Figure 11).

A 2017 polymorphic attack: Our first event is a DNS flood from 2017-03-06 in B-root [51] (Figure 11a). This event was a volumetric polymorphic attack where the attack queries have common formats like RANDOM.qycl520.com032 (from 0 s) and RANDOM.calling168.com032032 (changed at 4750 s, so polymorphic in nature). We assume 60k packets/s (30 Mb/s) capacity at each anycast site. The event was small enough that B-root was able to fully capture it across all active anycast sites at the time. The event lasted about 5 hours, but we show only the first 2.25 hours. Services and attacks capacity today will both be much larger; we use a small attack, scaling the attack and capacity up would show similar results.

In Figure 11a we can identity AMS site receives 100k packets/s traffic that is more than the capacity (shown as the maroon striped area). Our system notices the attack from bitrate alerts. It then estimates the AMS overload by computing the offered load using observed load and access fraction. The system maps networks to number of packets to each site using the pre-computed playbook (Table 6). Using this mapping our system/operator can then select a response. From Figure 11a, we can see the impact of the selected routing approach—announcing only to Transit-1 using community string. After 300 s, we can see no striped area which indicates the attack is mitigated.

The attacker changes the query names at 4750 s, making this attack polymorphic. Filtering on query names would need to react, but our routing changes can still mitigate the attack regardless of this type of change.

A 2021 variable-length polymorphic attack: We next examine an HTTP-attack launched on an enterprise network on 2021-09-05 in Figure 11b. This polymorphic attack changes after each of three pauses. The initial attack consists of millions of HTTP GETs (15k packets/s) launched from an IoT botnet; it terminates when the enterprise’s operator deploys IP-based filtering. About 1000 s later, a different botnet launched a multi-vector attack combining HTTP GETs using random paths (to avoid caching) and spoofed TCP ACKs. We then see a lull, brief burst, another lull, and a burst to the end.

The initial attack at time 0 overloads one site (AMS), prompting our routing response. After the estimation, we begin a route shift away from AMS, but the attack ends quickly (after 90 s), while routes are still changing.

Since the normal traffic sources originate from Europe,
most traffic went to AMS even after three prepends. At 1020 s the attack botnet changes, with more attack traffic from Asia and South America (based on IP geolocation from MaxMind) Our route changes in response to the initial attack are still in place, and the renewed attack is successfully spread over all three sites, allowing AMS to tolerate the new attack.

Routing convergence time implies that routing changes will overwhelm SEA, a smaller site, but can be handled at the super-site. We respond with negative prepend, with the traffic shift to BOS visible at 300 s. This response mitigates the attack (no striped area).

**Other attacks:** We have assessed additional attacks, and describe them in Appendix G. The additional polymorphic and volumetric attacks show that routing can successfully address attacks after routes propagate.

### 9 Limitations and Future Work

Our playbook of routing options (§3) is effective against many attacks (§8 and Appendix G). However, like any defense, it is not impervious. We next describe known limitations and areas of future work.

First, Internet routing is distributed, requiring time to converge. The effects of routing defenses cannot be seen until convergence. We do not make changes faster than 5 minutes.

Routing convergence time implies that routing changes will have limited applicability to short-lived attacks (less than 5 minutes). Although routing changes will not hurt the service, their benefits may not occur until routing shifts.

In addition, routing convergence means that polymorphic attacks that shift traffic sources quickly will be more effective. Routing changes are robust to polymorphic attacks that change method but take effect by traffic volume, they will spread load regardless of what it is, as we show in events in §8. However, when defending an attack where traffic shifts locations faster than routing converges, one must provision for the worst case volume to any site under the heaviest traffic it sees. Rapid shifts make defense harder, but not impossible.

Finally, we assume the anycast catchments of the underlying service change slowly (over days). We showed in §7.3 that this assumption generally holds.

Although we change routing during an attack to balance load across catchments, we do not explicitly attempt to locate attack origins. As future work, we could use such information to improve defense selection.

Attack response depends on human factors in service operators and attackers. Explicitly studying such human factors is potential future research. Our current work focused on the technical feasibility of our defenses.

### 10 Conclusions

This paper provides the first public evaluation of multiple anycast methods for DDoS defense. Our system estimates attack size, selects a strategy from a pre-computed playbook, and automatically performs traffic engineering (TE) to rebalance load or to advise the operator. Our contributions are attack-size estimation and playbook construction. We experimentally evaluate TE mechanisms, showing that prepending is widely available but offers limited control, while BGP communities and path poisoning are the opposite.

**Acknowledgments:** ASM Rizvi and John Heidemann’s work on this paper is supported, in part, by the DHS HSARPA Cyber Security Division via contract number HSHQDC-17-R-B0004-TTA.02-0006-I.

Joao Ceron and Leandro Bertholdo’s work on this paper is supported, by Netherlands Organisation for scientific research (4019020199), and European Union’s Horizon 2020 research and innovation program (830927). We would like to thank our anonymous reviewers for their valuable feedback. We are also grateful to the Peering and
Tangled admins who allowed us to run measurements. We thank Dutch National Scrubbing Center for sharing DDoS data with us.

References


[31] Chi-Yao Hong, Subhasree Mandal, Mohammad Al-Fares, Min Zhu, Richard Alimi, Kondapa Naidu B., Chandan Bhagat,


Matt Levine, Barrett Lyon, and Todd Underwood. TCP anycast—don’t believe the FUD. Presentation at NANOG 37, June 2006.


RIPE. Root DNS observations. Measurement ID 1009 (A-Root), 1010 (B-Root), etc., 2021.


RIPE. Root dns observations. Measurement ID 1009 (A-Root), 1010 (B-Root), etc., 2021.


A. Shaikh, R. Tewari, and M. Agrawal. On the effectiveness of
<table>
<thead>
<tr>
<th>Duration (seconds) relative to attack start</th>
<th>Normal duration</th>
<th>Attack duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimated rate</td>
<td>Reported rate</td>
<td>Observed rate</td>
</tr>
<tr>
<td>Normal duration attack duration</td>
<td>-150</td>
<td>-125</td>
</tr>
<tr>
<td>Query rate (million query/s)</td>
<td>12.0</td>
<td>2.0</td>
</tr>
<tr>
<td>12.0</td>
<td>2.0</td>
<td>4.0</td>
</tr>
<tr>
<td>4.0</td>
<td>6.0</td>
<td>8.0</td>
</tr>
<tr>
<td>6.0</td>
<td>8.0</td>
<td>10.0</td>
</tr>
<tr>
<td>8.0</td>
<td>10.0</td>
<td>12.0</td>
</tr>
</tbody>
</table>

**Figure 12:** Estimating real-world attack events: estimating June 2016 event with 0.91% access fraction.

## Appendix A Anycast and Verfploeter

IP anycast is a routing method used to route incoming requests to different locations (sites). Each site uses the same IP address, but at different geographic locations. Anycast then uses Internet routing with BGP to determine how to associate users to different sites—that is known as site’s anycast catchment. BGP has a standard path selection algorithm that considers routing policy and approximate distance [8].

Operators can influence the routing decisions process using different traffic engineering techniques (TE) to manipulate BGP. We describe TE techniques in §3.4.1 and how they can be used to rebalance the load during a DDoS attack.

We use Verfploeter [20] to find out the client to anycast site mapping. Using Verfploeter we build our BGP playbook with various BGP changes (§6.4). The main intuition behind Verfploeter is to send pings to millions of address blocks [23, 78], using an anycast prefix as source address. The replies to these pings will be routed to the nearest anycast site by the inter-domain routing system from which we can map address blocks to the anycast sites.

## Appendix B Operator Assistance System

To assist operators (§3.4.3), we provide an interface for defense. To react and reconfigure the anycast network, the operators can use a web interface similar to an equalizer, choosing the percentage of load to be increased or dropped at an anycast site. The possible ranges of slider positions are based on the playbook alternatives or presets of routing policies. This process hides the playbook complexity from the operator, making the process less error-prone and more intuitive, but still giving the operator a full control of the BGP routing.

In Figure 3 we can visualize a snapshot of this interface. Each slider represents an anycast site and each site has predetermined settings indicated by “notches”. The positions of the “notches” are the results of all the measurements obtained to create our playbook. The bar graph shows the results of the measurement process, indicating how many networks will be attracted to each anycast site. The operators can visualize the forecasted traffic to each position and then apply the configuration on the production network.
With poisoning coverage limited by filters (§6.3.1), we next examine what granularity control it provides. We expect to see limited range since we cannot poison Tier-1 ASes, and small ASes carry little traffic.
A 2020 volumetric attack at B-root: We observed an ephemeral volumetric event at B-root on 2020-02-14 where the attackers used a single query name—peacecorps.gov. This event lasted very briefly for 3 minutes. In practice, no routing approach can work against such short-lived attacks due to the propagation delay of BGP. We stretched the event with similar traffic rate so that we can see the impact if the attack continues for more time.

In this event also, AMS is overloaded with 60k packets/s when the assumed capacity is 40k packets/s (Figure 16a). We prepend AMS by 1 so that the traffic shifts away from AMS. After 300 s, we can see no overloaded striped area in AMS.

These volumetric attacks are common at root servers. Routing based approaches can defend against such attacks.

A 2021 B-root event where our system iterates: We evaluate another event at B-root occurred on 2021-05-28. In this event, the queries were IP fragmented (large packet size), and the common query name was pizzaseo.com (we stretched the event since it was short-lived). When the attack started, our system finds AMS site overloaded (Figure 16b). Our system finds prepending from AMS is the best approach to reduce traffic from AMS. However, after prepending AMS by 1, CNF site gets the most redirected traffic, and becomes overloaded. Redirected attack sources prefer CNF over BOS. When our system finds CNF site overloaded, it deploys an approach that will reduce traffic from CNF since it is now overloaded. Our system deploys negative prepending to push more traffic towards BOS site. After 900 s, we can see there is no overloaded site. This event shows how our system can gradually find out the best routing approach.

Defending with community strings: We next consider an attack observed at the Dutch National Scrubbing Center on 2021-08-27. This attack was a volumetric DNS amplification.

In this attack, AMS is overloaded. Consulting the playbook, we select a response using community strings to shift traffic, retaining six IXP peers at AMS, while dropping all other peers and transits. The impact of this change is visible at 300 s in Figure 16c, as the attack is successfully spread across all sites.

This example shows how different community strings provide control over traffic distribution. We show more events in the extended version of this paper [57].

We evaluate more attack events captured at B-root and at the Dutch National Scrubbing Center. We follow the same methodology mentioned in §8. We use the same playbook built with AMS, BOS and CNF sites ($6$) from Peering.