Improving Anycast with Measurements

Wouter de Vries

IMPROVING ANYCAST WITH MEASUREMENTS

DISSERTATION

to obtain
the degree of doctor at the University of Twente,
on the authority of the rector magnificus,
prof dr. T.T.M. Palstra,
on account of the decision of the graduation committee,
to be publicly defended
on Wednesday 18 December 2019 at 14:45

by

Wouter Bastiaan de Vries

born on November 3, 1990
in Hengelo, the Netherlands.

This dissertation has been approved by:

Supervisor:
Prof. dr. ir. A. Pras

Co-supervisors:
Dr. ir. P.T. de Boer
Dr. ir. R. van Rijswijk-Deij

© 2019 Wouter Bastiaan de Vries, The Netherlands. All rights reserved. No parts of this thesis may be reproduced, stored in a retrieval system or transmitted in any form or by any means without permission of the author. Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd, in enige vorm of op enige wijze, zonder voorafgaande schriftelijke toestemming van de auteur.

Graduation Committee

Chairman:	Prof. dr. J.N. Kok	
Supervisor:	Prof. dr. ir. A. Pras	
Co-supervisor:	Dr. ir. P.T. de Boer	
Co-supervisor:	Dr. ir. R. van Rijswijk-Deij	
Members:		
Prof. dr.	J.S. Heidemann	University of Southern California, United States
Prof. dr. rer. nat.	O. Hohlfeld	Brandenburg University of Technology, Germany
Prof. dr.	C. Pelsser	University of Strasbourg, France
Prof. dr. ir.	G.J. Heijenk	University of Twente, The Netherlands
Prof. dr. ir.	L.J.M. Nieuwenhuis	University of Twente, The Netherlands

Funding sources

SURFnet Research on Networks
EU H2020 CONCORDIA – #830927

DSI Ph.D. thesis series no. 19-020
Digital Society Institute
P.O. Box 217
7500 AE Enschede, The Netherlands

ISBN: 978-90-365-4897-7
ISSN: 2589-7721
DOI: 10.3990/1.9789036548977

Acknowledgements

During the years of a PhD you accumulate a fairly long list of people that deserve to be thanked in the acknowledgments of your thesis. As some people can attest my memory for names is not my best feature, so I would also like to issue a blanket *thank you* to everyone: consider yourself acknowledged.

I would like to thank my supervisor Aiko as well as my co-supervisors Roland and Pieter-Tjerk for their help, without them this thesis would not have been approved. Aiko is also the one who found some pieces of funding here and there and put them together into a PhD position for me, and I'm glad that I took that opportunity. Roland pushed me to greater heights than I would have managed to achieve by myself, even going so far as to make me move to a different country, all in the name of science, thank you. Pieter-Tjerk, there was a time during my PhD when I had very little supervision and you were volunteered to take the role of daily supervisor, for me it turned out to be a great success and I really enjoyed your eagerness to have small-talk about the French Minitel, among other random things. During the years I have come to rely on your great attention to detail, thank you!

My paranymphs, Luuk and Erik. Luuk, people say that I can be a little sarcastic, but I think that we achieved some type of synergy that brought my level of sarcasm to really great heights. We like to complain about the research group but I think we have to admit that we also had a lot of fun over the years, especially on one of our (many) trips together. I hope we will continue our friendship when you move back to that place where they, mistakenly, call fries *friet* instead of the obviously correct *patat*. Erik, I've known you since I started my Bachelor's degree, it's hard to imagine that more than ten years have passed since then. Clearly time flies when you are having fun, and drinking beer. I hope that we will keep brewing our own beer for a long time to come. Thank you both for being my paranymphs!

A huge thank you to my parents, Alfred (dad) and Hermi (mom). Without them this thesis would certainly not have happened. You have always gently pushed me into what I consider the right direction. As a kid I used to say that I wanted the same job as my dad, and while that did not quite happen, I did end up working at the University of Twente!

Albert and Bernadien, thank you for the many relaxing weekends that we have spent at your house, especially when we were still living in our fantastic flat in Enschede (even though I had not yet started on my PhD at that point, I surely would not have survived to start it otherwise). Also thank you for Fleur, she is the best part of my life!

All of my colleagues and former colleagues: Aashik, Anna, Anne, Baver, Bernd, Boudewijn, Christian, Cristian, Geert, Hans, Jair, Jeanette, Jeroen, Joao, Leandro, Mattijs, Moritz, Morteza, Mozhdeh, Nils, Olivier, Raffaele, Ramin, Ricardo, Rick, Roland, Sarwar, Suzan and Wouter. Thank you for making *DACS* what it is, and making my time there really enjoyable!

My friends, to avoid accidentally leaving someone out I will keep this short: thank you!

Finally, and most importantly, my wife Fleur. We have made many jokes about what I would say here, for example that I managed to finish this thesis despite your best efforts to distract me, or that I would refer to you as the old ball and chain. We both know that I would never dare to say such things. You have stood by me throughout this journey. You even joined me in moving to London so I could play with computers, without a second thought. Fleur, I love you.

Wouter de Vries

Abstract

Since the first Distributed Denial-of-Service (DDoS) attacks were launched, the strength of such attacks has been steadily increasing, from a few megabits per second to well into the terabit/s range. The damage that these attacks cause, mostly in terms of financial cost, has prompted researchers and operators alike to investigate and implement mitigation strategies. Examples of such strategies include local filtering appliances, Border Gateway Protocol (BGP)-based blackholing and outsourced mitigation in the form of cloud-based DDoS protection providers.

Some of these strategies are more suited towards high bandwidth DDoS attacks than others. For example, using a local filtering appliance means that all the attack traffic will still pass through the owner's network. This inherently limits the maximum capacity of such a device to the bandwidth that is available. BGP Blackholing does not have such limitations, but can, as a side-effect, cause service disruptions to end-users. A different strategy, that has not attracted much attention in academia, is based on anycast.

Anycast is a technique that allows operators to replicate their service across different physical locations, while keeping that service addressable with just a single IP-address. It relies on the BGP to effectively *load balance* users. In practice, it is combined with other mitigation strategies to allow those to scale up. Operators can use anycast to scale their mitigation capacity horizontally.

Because anycast relies on BGP, and therefore in essence on the Internet itself, it can be difficult for network engineers to fine tune this balancing behavior. In this thesis, we show that that is indeed the case through two different case studies. In the first, we focus on an anycast service *during normal operations*, namely the Google Public Domain Name System (DNS), and show that the routing within this service is far from optimal, for example in terms of distance between the client and the server. In the second case study, we observe the root DNS, *while it is under attack*, and show that even though in aggregate the bandwidth available to this service exceeds the attack we observed, clients still experienced service degradation. This degradation was caused due to the fact that some sites of the anycast service received a much higher share of traffic than others.

In order for operators to improve their anycast networks, and optimize it in terms of resilience against DDoS attacks, a method to assess the actual state of such a network is required. Existing methodologies typically rely on external vantage points, such as those provided by RIPE Atlas, and are therefore limited in scale, and inherently biased in terms of distribution. We propose a new measurement methodology, named Verfploeter, to assess the characteristics of anycast networks in terms of client to Point-of-Presence (PoP) mapping, i.e. the anycast catchment. This method does not rely on external vantage points, is free of bias and offers a much higher resolution than any previous method. We validated this methodology by deploying it on a testbed that was locally developed, as well as on the B root DNS. We showed that the increased *resolution* of this methodology improved our ability to assess the impact of changes in the network configuration, when compared to previous methodologies.

As final validation we implement Verfploeter on Cloudflare's global-scale anycast Content Delivery Network (CDN), which has almost 200 global Points-of-Presence and an aggregate bandwidth of 30 Tbit/s. Through three real-world use cases, we demonstrate the benefits of our methodology: Firstly, we show that changes that occur when withdrawing routes from certain PoPs can be accurately mapped, and that in certain cases the effect of taking down a combination of PoPs can be calculated from individual measurements. Secondly, we show that Verfploeter largely reinstates the `ping` to its former glory, showing how it can be used to troubleshoot network connectivity issues in an anycast context. Thirdly, we demonstrate how accurate anycast catchment maps offer operators a new and highly accurate tool to identify and filter spoofed traffic.

Where possible, we make datasets collected over the course of the research in this thesis available as open access data. The two best (open) dataset awards that were awarded for these datasets confirm that they are a valued contribution.

In summary, we have investigated two large anycast services and have shown that their deployments are not optimal. We developed a novel measurement methodology, that is free of bias and is able to obtain highly accurate anycast catchment mappings. By implementing this methodology and deploying it on a global-scale anycast network we show that our method adds significant value to the fast-growing anycast CDN industry and enables new ways of detecting, filtering and mitigating DDoS attacks.

Samenvatting

De kracht van gedistribueerde Denial-of-Service (DDoS) aanvallen neemt voortdurend toe. Waar dergelijke aanvallen aanvankelijk uit enkele megabits per seconde bestonden, hebben we het inmiddels over terabits per seconde. De schade die dit soort aanvallen, hoofdzakelijk vanuit een financieel oogpunt, veroorzaken heeft ervoor gezorgd dat zowel onderzoekers als netwerkbeheerders onderzoek doen naar strategieën om deze schade in te perken. Voorbeelden van dergelijke strategieën zijn het gebruik van filterapparatuur in het lokale netwerk, op Border Gateway Protocol (BGP)-blackholing gebaseerde technieken en het uit handen geven van de verdediging aan gespecialiseerde providers in *de cloud*.

Een aantal van deze strategieën is geschikt voor DDoS-aanvallen met een hoge bandbreedte, een ander deel voor lage bandbreedtes. Het gebruik van lokale filterapparatuur heeft bijvoorbeeld als nadeel dat al het verkeer alsnog door het netwerk van de eigenaar loopt, wat grenzen stelt aan de maximale omvang van de DDoS-aanvallen die afgeweerd kunnen worden. Bij het gebruik van BGP-blackholing gelden deze grenzen niet, maar hierbij kan de te beschermen dienst (tijdelijk) onbereikbaar worden voor eindgebruikers. Een andere strategie, die nog niet veel aandacht heeft gekregen in de academische wereld, is er een gebaseerd op anycast.

Anycast is een techniek die netwerkbeheerders in staat stelt om hun dienst te dupliceren op meerdere fysieke locaties, terwijl die dienst bereikbaar blijft op een enkel IP-adres. We spreken in zo'n geval van verschillende instanties van eenzelfde dienst. Anycast is afhankelijk van BGP om gebruikers te verdelen over de instanties van de dienst. In de praktijk wordt anycast gebruikt in combinatie met andere DDoS-verdedigingstechnieken om die zo verder te laten schalen, en dus grotere aanvallen af te kunnen weren. Netwerkbeheerders kunnen met behulp van anycast hun capaciteit om aanvallen af te weren horizontaal, dus door meer servers te plaatsen op meer locaties, uitbreiden.

Omdat anycast afhankelijk is van BGP, en daarmee van de combinatie van netwerken waar het internet uit bestaat, is het potentieel lastig voor netwerkbeheerders om de verdeling van het netwerkverkeer over de verschillende instanties van hun dienst te optimaliseren. In dit proefschrift laten we aan de hand van twee casussen zien dat de verdeling van verkeer inderdaad lastig te optimaliseren is. In de eerste casus focussen we op een anycastdienst die regulier in gebruik is, namelijk de Google Public Domain Name System (DNS). We laten zien dat de routering voor die dienst verre van optimaal is, gelet op de fysieke afstand tussen de gebruiker van de dienst en de dienst zelf. In de tweede casus kijken we naar de Root DNS terwijl deze een aanval ondervindt, en laten we zien dat hoewel er geaggregeerd voldoende bandbreedte beschikbaar is, gebruikers toch hinder ondervinden. Deze hinder wordt veroorzaakt doordat sommige instanties van de dienst een veel groter gedeelte van het verkeer ontvangen dan anderen.

Om netwerkbeheerders in staat te stellen om hun anycastnetwerken te verbeteren, en te optimaliseren tegen Distributed Denial-of-Service (DDoS) aanvallen, is er een methode nodig voor beheerders om de huidige toestand (wat is de huidige verdeling van het verkeer) van hun netwerk te kunnen bepalen. Bestaande methodes zijn veelal afhankelijk van externe observatiepunten, bijvoorbeeld zoals die worden aangeboden door RIPE Atlas, en zijn daardoor beperkt in schaal. Verder zijn dergelijke observatiepunten nooit volledig gelijkmatig verdeeld over de wereld. Wij stellen een nieuwe methodologie voor, genaamd Verfploeter, om de eigenschappen van een anycastnetwerk in termen van de verdeling tussen gebruikers en de instanties van de anycastdienst te kunnen meten. Deze methode is niet afhankelijk van externe observatiepunten, is daardoor vrij van het verdelingsprobleem en biedt daarnaast een veel hogere meetdichtheid. Deze nieuwe methode hebben we gevalideerd door hem te implementeren op een door ons ontwikkeld testbed, evenals op de B Root (onderdeel van de Root DNS). We hebben aangetoond dat de verhoogde meetdichtheid van de methode ons in staat stelt om de impact van veranderingen aan het netwerk beter vast te stellen.

Als laatste validatie hebben we de Verfploeter-methode geïmplementeerd op Cloudflare's anycastnetwerk, dat bestaat uit bijna 200 locaties wereldwijd, met een totale bandbreedte van meer dan 30 terabit per seconde. Door middel van drie casussen demonstreren we de voordelen van de methodologie. We laten zien dat veranderingen die ontstaan door het uitschakelen van specifieke anycastlocaties nauwkeurig gemeten kunnen worden. Daarnaast laten we zien dat in sommige gevallen metingen waarin individuele anycastlocaties uitgeschakeld zijn, gecombineerd kunnen worden om vast te stellen wat er zou gebeuren als beide locaties tegelijk uitgeschakeld worden. We laten ook zien dat Verfploeter gebruikt kan worden om verbindingsproblemen te troubleshooten. Als laatst demonstreren we hoe de nauwkeurige metingen gebruikt kunnen worden om verkeer waarbij het bronadres vervalst is te identificeren en te filteren, en zo DDoS-aanvallen af te weren.

Waar mogelijk hebben we de datasets die we gedurende het onderzoek verzameld hebben, openbaar beschikbaar gemaakt. Hiermee zijn ook twee *Best (open) dataset awards* gewonnen, wat aantoont dat deze bijdrage door de netwerkgemeenschap gewaardeerd wordt.

Samenvattend, we hebben twee grootschalige anycastdiensten onderzocht en aangetoond dat deze niet optimaal functioneren. We hebben een nieuwe meetmethode ontwikkeld die vrij is van de verdelingsproblemen die gebruikelijk zijn bij het gebruik van externe observatiepunten, en die in staat is om zeer nauwkeurig het gedrag van een anycastnetwerk te meten. Door deze methodologie te implementeren en uit te rollen op een wereldwijd anycastnetwerk hebben we aangetoond dat de methode een significante toegevoegde waarde heeft voor de snelgroeiende anycast Content Delivery Network (CDN) industrie en nieuwe mogelijkheden geeft om DDoS-aanvallen te detecteren, te filteren en af te weren.

Contents

CHAPTER 1

Introduction

1.1 Motivation

In 1965 development started of what would later be considered the beginning of the Internet at the National Physical Laboratory in the United Kingdom. Later, this knowledge was used as input for ARPANET, in the United States, mainly by the Defense Advanced Research Projects Agency (DARPA) [121]. This system was also influenced by the French CYCLADES research network. While a number of institutions and people were working simultaneously on establishing what became the Internet, there was little regard for security, or really any need for it. Nowadays, the Internet is all around us. From lighting and heating in houses to air traffic control, banking and even critical infrastructures such as energy and water. Everything is now connected, all the time.

Unfortunately, the need for security quickly became apparent. While it is not exactly clear when the first Denial-of-Service (DoS) attack took place, there are reports that it happened in the 70s [122], with the first real Distributed Denial-of-Service (DDoS) attacks following in the 90s [123]. Lately, DDoS attacks have started to rapidly increase in both quantity and severity, as shown in Figure 1.1.

It is interesting to note that the relatively stable attack volume from 2008, through to 2012, was broken by a Dutch company called Cyberbunker (in co-operation with the German company CB3ROB), which offered hosting services for everything up to, but not including, child pornography and terrorism. When this company was put on an anti-spam blacklist maintained by Spamhaus, they initiated a massive DDoS attack in excess of 300 Gbit/s, which at that time was by far the largest attack ever seen. For comparison, a typical single server nowadays has a bandwidth of 10 Gbit/s, and sometimes just 1 Gbit/s. The effects of this attack were noticeable across the Internet, and not just at the target, but also elsewhere due to congestion in intermediate networks. Spamhaus itself remained unreachable for five days.

Services, digital as well as physical, that we interact with daily are dependent on the Internet, think for example about the ubiquitous presence of contactless payments, and electronic payment as a whole. These systems can only work if they have a connection to the underlying banking infrastructure. Somewhat

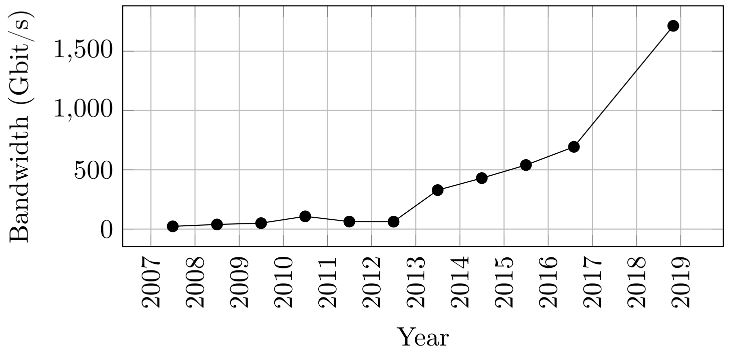


Figure 1.1: Increase of DDoS peak attack bandwidth over the years [124].

less obvious are the scheduling of public transport, airplanes, ticketing, entry to physical infrastructure such as parking lots or buildings, alarm systems, phones, or even the lights and temperature in our own houses. All of these can be affected by Internet outages to varying degrees, some systems can still be controlled locally, and some tasks can still be performed manually, for others we are completely dependent on an operational Internet.

Of course, a DoS attack, and most commonly the distributed variety, is only one of many possible attacks on an Internet connected service. Others include malware, phishing, spear phishing, SQL injection, etc. Of these though, the DDoS is one which has the lowest barrier of entry. Recent publications [1] show that high school students regularly purchase DDoS attacks, for example to delay or prevent an online exam from taking place, indicating that practically no technical knowledge is required for this type of attack.

There are various solutions to counter DDoS attacks against a service [2]. Some are more effective in particular cases than others, and in practice multiple solutions need to be deployed in a layered approach. We describe a few solutions here, also see Figure 1.2.

First, there are mitigation techniques that can be deployed locally, for example by rate-limiting or completely filtering traffic from specific addresses, or traffic matching specific signatures or behavior. These solutions generally work well if the attack volume is within the limits of both the processing capability of the machine that performs the filtering, as well as within the available upstream bandwidth. Another advantage is that this solution can be deployed without any dependency on an upstream party. However, some DDoS attacks now reach volumes in excess of 1 TBit/s, making it prohibitively expensive to provision sufficient resources to handle this traffic locally, both in terms of processing and bandwidth.

Complementary, there are the collaborative methods, where operators collaborate with upstream providers to block or limit incoming traffic. Such strategies are typically better suited towards higher attack volumes. These include

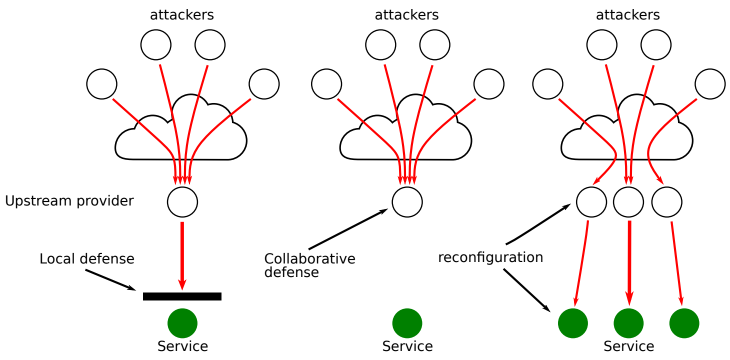


Figure 1.2: Schematic overview of three typical DDoS defense strategies

firewalls where rules are exchanged via some out-of-band channel, as well as Border Gateway Protocol (BGP) blackholing, where traffic is blocked via the BGP protocol by adding a specific attribute (a so called Community) to an announced route. In some cases significant side-effects are incurred where the targeted service becomes (partially) unreachable, but collateral damage is prevented. Positive aspects of these methods include cost efficiency, as well as the potential ability to handle much higher volumes of traffic.

The last category of methods we will describe are ones where the network, and/or the service itself, is changed to provide better resilience against DDoS attacks. One example of this is a solution where traffic is routed through a so called *scrubbing center*, where the DDoS traffic is blocked and only clean traffic is routed through to the service. Nowadays this is offered as a commercial service by companies such as Akamai [125] or, in the Netherlands, NBIP [126]. Another option is to replicate the service across multiple networks, and geographical locations, in order to achieve resilience against a DDoS attack. With the total traffic in a single location being much lower, it is easier to apply more local solutions.

This last option, service replication, is interesting in that it potentially allows a service to scale (almost) indefinitely by replicating to more and more locations. A logical question is how a client can know which replica of a service to contact. There are two basic ways of achieving this, the first one is by leveraging the Domain Name System (DNS). For example by applying a round-robin load balancing strategy traffic can be directed to many different replicas of the same service, thus lowering the bandwidth and processing requirements at each location. However, this method is not particularly suited towards improved resilience against DDoS as servers can still be specifically targeted by ignoring the DNS load-balancer.

The second option is to use anycast, which is a routing strategy that essentially means that the replicas of the service each use the exact same Internet Protocol (IP) address, and letting the Internet routing protocol take care of

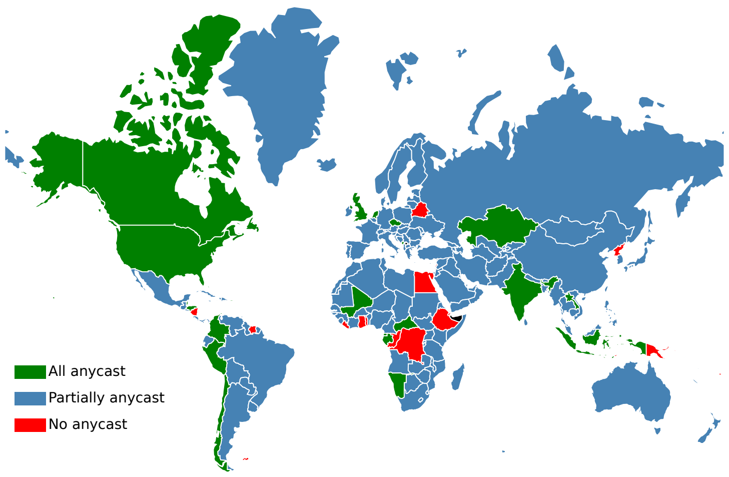


Figure 1.3: Use of Anycast across all ccTLDs. Interactive map available at `https://wbdv.nl/anycast/`

the load balancing. Due to the way BGP works, this causes the clients which are nearest to a specific location, topologically speaking, to be routed there. With this method, it is much harder for an attacker to concentrate an attack on a single location, as the BGP routing is within the collective control of the networks making up the Internet. In other words, to target a single location an attacker would have to cause its attack to originate in only those networks that happen to be routed to a specific location. Depending on the number of anycast instances this can be infeasible for an attacker to achieve.

Taking a step back from solutions against DDoS attacks, and taking a closer look at the Internet itself, we can see that the DNS is a major component of it. The DNS is that part which is responsible for translating between human understandable names, and the IP addresses that computers understand. If the DNS were to suffer a (partial) outage, much of the Internet would stop functioning, as computers would no longer be able to make sense of any domain names. The fact that this component is so critical, also makes it an interesting target for DDoS attacks. Many DNS operators, for example those operating Top Level Domain for their country, use anycast to improve the latency for clients, as well as the resilience against DDoS attacks. Figure 1.3 shows which countries use anycast for one or more of their authoritative DNS servers.

Interestingly, of the described DDoS mitigation solutions, anycast has not seen much research in that context, even though it is in widespread use. Therefore, in this thesis, we will focus on the use of anycast, particularly by DNS operators. We will investigate how anycast networks are operated, how they succeed or fail in the face of DDoS and ultimately we aim to provide measurement based methodologies to help improve the operation of such networks.

1.2 Objective, Research Questions & Approach

1.2.1 Objective

An anycasted service has no single point of failure, it also has the added advantage that when being subject to a DDoS attack it is more resilient, and can thus handle a higher volume of attack traffic. Even when overwhelmed the service might become unavailable to only a fraction of its client base.

Theoretically, anycast has interesting properties, however, research into anycast deployments, both in and out of the context of DDoS attacks is limited. For example, the attack on the DNS root on November 2015 [127], [3], which is largely anycasted, shows the need for research in this area: Some servers received so many requests that it saturated the network connections to some of them, even though the total available bandwidth across the entire service was higher than the total attack traffic.

In this thesis we aim to investigate how operators run their networks, to see how the Internet behaves with regard to anycast in the real world. Then, to develop methods to help improve the current state of the art in managing real world anycast deployments. Concretely, the objective is:

> *To measure anycast deployments and develop methods to optimize anycast deployments in order to improve service resilience against DDoS attacks*

In order to achieve our objective we aim to perform measurements on large-scale services on the Internet, such as the DNS Root and large Content Delivery Networks (CDNs). The methods that we develop will be tested on a testbed that, while not in production use, is active on the Internet. Wherever possible we will prove their use in a production environment.

1.2.2 Research Questions & Approach

Given that in the modern Internet anycast is already widely deployed, mainly for network performance reasons, a logical first step towards improvement is to observe what the current state is. Operators can deploy anycast in a variety of ways, for example by combining upstream providers, by peering at different Internet Exchange Points (IXPs) or, more fundamentally, by the choice of physical locations to deploy the hardware.

We aim to investigate large-scale anycast networks, both in normal operation, as well as under stress from a DDoS attack, in the real world. Ultimately, our goal is to learn how we can improve these deployments to achieve better resilience against such attacks. We thus formulate our first research question as follows:

> **RQ1** – *What can we learn from analyzing the behaviour of a large-scale anycast service?*

We approach RQ1 as follows: we will perform a longitudinal analysis of a large scale anycasted recursive DNS resolver during two and a half years. This will allow us to see how such a network evolves over time, as well as to see if there are any performance issues.

RQ2 – *How does anycast perform in the face of a DDoS attack?*

We approach RQ2 by looking at a recent large attack on a piece of vital Internet infrastructure: the Root DNS. We argue that the Root DNS is an excellent subject for three reasons: **a)** it is a critical system under high load, which is depended upon by practically every Internet user. **b)** it is diverse in network configuration, as it is hosted by multiple (13) parties, each with their own independent setup. **c)** this particular system is under continuous monitoring, data sets of which are made publicly available.

Now that we have learned how current deployments of anycast perform, both in normal operation, as well as against a large DDoS attack we shift our focus to the second part of our objective: *developing methods to optimize deployments*. Without methodologies that allow operators to accurately measure the current performance of their network, it is hard to effectively make changes to the network. Before moving on to developing such methods we first investigate what difficulties can arise when performing measurements on the Internet, especially concerning Internet paths, we therefore formulate our third research question as follows:

RQ3 – *What challenges are there in measuring anycast networks?*

Our approach to answering this question is to perform measurements of Internet paths, using standard tools, and investigate what the characteristics of such paths are. Then, depending on those characteristics, we can develop a measurement methodology, which leads to the following research question:

RQ4 – *How can we accurately measure anycast performance?*

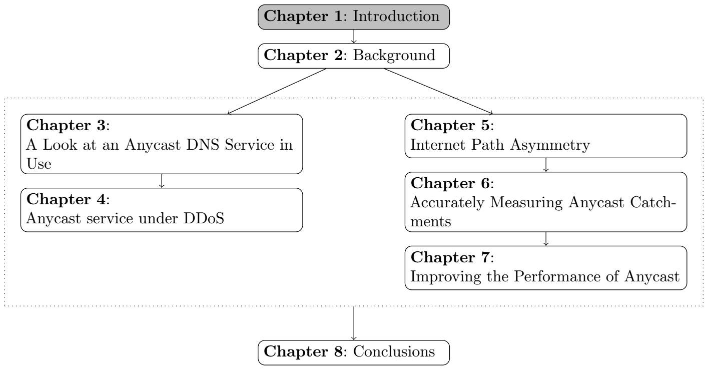
To address this research question we set out to develop a methodology that allows operators to quickly, comprehensively and accurately measure the performance of an anycast network. Currently, operators have two main choices, they can **a)** observe how their network performs by putting operational traffic on it, and then process the log files or **b)** use some external probing system, such as RIPE Atlas, or Thousand Eyes, to measure their network externally. Unfortunately, both have significant drawbacks. **a** requires operators to put production traffic onto the service prior to knowing its performance, and **b** relies on an external system, that is not necessarily representative of the client base of the service.

We then apply the developed methodology to assist in the improvement of the performance of anycast networks. This leads to the following, final, research question:

RQ5 – *How can we improve anycast performance and operations?*

We approach this research question by implementing our methodology on a large-scale global anycast network. We then demonstrate several use-cases on this network, and show how our methodology can be applied to improve the operations of anycast.

1.3 Thesis Organization & Contributions


Chapter 1: Introduction
Chapter 2: Background
Chapter 3: A Look at an Anycast DNS Service in Use
Chapter 4: Anycast service under DDoS
Chapter 5: Internet Path Asymmetry
Chapter 6: Accurately Measuring Anycast Catchments
Chapter 7: Improving the Performance of Anycast
Chapter 8: Conclusions

Chapter 2 – Background

In this chapter we provide background information on the three main topics of this thesis, namely the Border Gateway Protocol (BGP), the Domain Name System (DNS) and Distributed Denial-of-Service (DDoS) attacks.

Chapter 3 – A Look at an Anycast DNS Service in Use

This chapter provides a deep-dive into the anycast network that Google utilizes for their global recursive DNS service, 8.8.8.8. We look at passive logs that were collected over a time span of 2.5 years. Using this data we provide a look at the efficiency (in terms of latency, distance and load distribution) of Google's anycast network.

We highlight the following contributions from this chapter:

- We show that while anycast routing is generally considered relatively stable [4], performance of Google's anycast network varies over time. Importantly, we show that traffic is frequently routed to out-of-country Points-of-Presence (PoPs), even if a local, in-country

PoP is available. This potentially exposes DNS traffic to state-level surveillance.

- We show that, based on geolocation of IP addresses, there is often a PoP available that is closer to the end-user, than the one that is being used.
- We show that end-users switch away from their Internet Service Provider (ISP)'s resolver if it is severely underperforming, and more importantly, that these users will not switch back.
- We show that surprisingly large numbers of Simple Mail Transfer Protocol (SMTP) servers are configured to perform lookups through Google Public DNS (GPDNS). This is a potential privacy leak, as it allows the public resolver and any of the authoritative name servers involved in DNS lookups to infer that there is likely communication between two parties. As an additional validation, we verify that a number of common SMTP daemons perform DNS lookups in their default configurations.
- We quantify the adoption of QNAME minimization, as a privacy enhancing technique to counteract the previously found issue.
- We make our passive DNS dataset covering 2.5 years and 3.7 billion queries, as well as the dataset used to investigate QNAME minimization available as open data to the research community at `https://traces.simpleweb.org`.

This chapter is based on the following publications:

- W.B. de Vries, R. van Rijswijk-Deij, P.T. de Boer, A. Pras. *Passive observations of a large DNS service: 2.5 years in the life of Google.* In: IEEE Transactions on Network and Service Management (TNSM), 2019. Extended version, based on previous conference paper.
- W.B. de Vries, R. van Rijswijk-Deij, P.T. de Boer, A. Pras. *Passive observations of a large DNS service: 2.5 years in the life of Google.* In: Network Traffic Measurement and Analysis Conference, TMA 2018, 26-29 June 2018, Vienna, Austra – **Best Open Dataset Award**
- W.B. de Vries, Q. Scheitle, M. Muller, W. Toorop, R. Dolmans, R. van Rijswijk-Deij. *A First Look at QNAME Minimization in the Domain Name System.* In: Passive and Active Network Measurement Conference, PAM 2019, 27-29 March 2019, Puerto Varas, Chile – **Best Dataset Award**

Chapter 4 – Anycast service under DDoS

This chapter provides an *evaluation of several IP anycast services under stress with public data.* Our subject is the Internet's Root Domain Name

Service, made up of 13 independently designed services ("letters", 11 with IP anycast) running at more than 500 sites. Many of these services were stressed by sustained traffic at 100× normal load on Nov. 30 and Dec. 1, 2015. We use public data for most of our analysis to examine how different services respond to stress.

We highlight the following contributions from this chapter:

- We show the first evaluation of anycast services under a DDoS attack, under many different architectures
- We identify different policies of dealing with attack traffic, e.g. absorb the traffic, or withdraw the anycast site.
- We show the need to understand anycast design to improve service resilience

This chapter is based following publication:

- G. Moura, R. de O. Schmidt, J. Heidemann, W.B. de Vries, M. Muller, L. Wei, C. Hesselman. *Anycast vs. DDoS: Evaluating the November 2015 root DNS event.* In: Internet Measurement Conference, IMC 2016, 14-16 November 2016, Santa Monica, USA

The work in this chapter was a significant collaborative effort. To highlight a number of specific contributions by the author of this thesis we point at Figure 4.1, Figure 4.2, Figure 4.8 and Figure 4.11, along with their accompanying analyses. More minor textual contributions are spread throughout the chapter.

Chapter 5 – Internet Path Asymmetry

Anycast is fully dependent on Internet routing, and the paths that exist on the Internet. In this chapter we take a closer look at these paths. Specifically, we focus on the presence of routing asymmetry in the Internet, where the forward path (from server to client) is different from the reverse path (from client to server). Routing asymmetry is an important reason why tools such as traceroute can only capture part of the path between clients and servers, and it is these paths that are fundamental to anycast routing.

We highlight the following contributions from this chapter:

- We provide a conclusive overview on the partial asymmetry of Internet routing.
- We have confirmed the presence of asymmetry in the majority of Internet routes.
- We provide our measurements as open data.

This chapter is based on the following publication:

- W. de Vries, J.J. Santanna, A. Sperotto, A. Pras. *How Asymmetric Is the Internet?* In: Intelligent Mechanisms for Network Configuration and Security, AIMS 2015, 22-25 June 2015, Ghent, Belgium – **Best Paper Award**

Chapter 6 – Accurately Measuring Anycast Catchments

Given the importance for operators to understand, and predict, changes to anycast catchments, and given that determining said catchment is not trivial, we propose a novel methodology in an effort to solve this issue. The method introduced in this chapter is able to accurately and quickly determine the catchments of an anycast network from the inside.

We highlight the following contributions from this chapter:

- We provide a novel methodology to determine catchments.
- We improve upon existing methodologies by offering 430× the number of vantage points than the well-known probing system RIPE Atlas.
- We validated the approach using both a real-world test bed as well as by using it in the deployment of a new anycast site for the DNS B-root.
- We provide open source implementations for deploying the methodology.

This chapter is based on the following publication:

- W.B. de Vries, R. de O. Schmidt, W. Hardaker, J. Heidemann, P.T. de Boer, A. Pras. *Broad and load-aware anycast mapping with Verfploeter.* In: Internet Measurement Conference, IMC 2017, 1-3 November 2017, London, United Kingdom

Chapter 7 – Improving the Performance of Anycast

Now that we have a methodology to measure anycast catchments accurately, we show, by presenting three use-cases, how these can be applied to an operational anycast network. We also present a deployment of Verfploeter on one of the largest anycast CDNs in the world.

We highlight the following contributions from this chapter:

- We show how Verfploeter can be implemented on a large-scale network.
- We show how Verfploeter can be used to plan network changes.
- We show how accurate anycast catchment mappings can be used to detect spoofed IP traffic.

This chapter is based on the following paper, which has been accepted for publication:

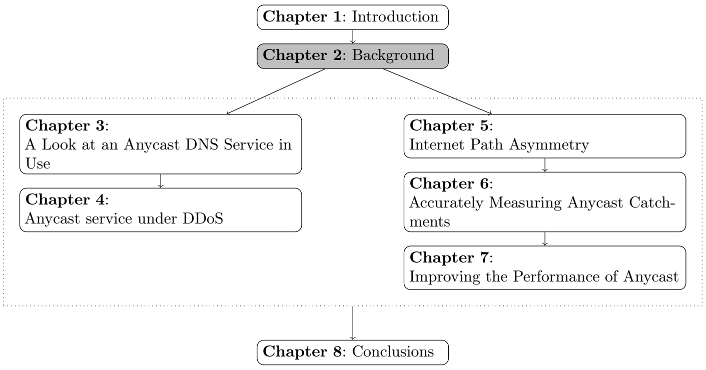
- W.B. de Vries, S. Aljammaz, R. van Rijswijk-Deij. *Global-scale Anycast Network Management with Verfploeter.* In: IEEE/IFIP Network Operations and Management Symposium (NOMS), 20-24 April 2020, Budapest, Hungary.

Chapter 8 – Conclusions

In this final chapter of the thesis we draw conclusions from the preceding chapters and reflect on our objective. We will also take a look forward and see what remains te be done in this research area.

CHAPTER 2

Background


Chapter 1: Introduction
Chapter 2: Background
Chapter 3: A Look at an Anycast DNS Service in Use
Chapter 4: Anycast service under DDoS
Chapter 5: Internet Path Asymmetry
Chapter 6: Accurately Measuring Anycast Catchments
Chapter 7: Improving the Performance of Anycast
Chapter 8: Conclusions

This chapter introduces a number of key background topics that are of importance in understanding the chapters to come. The intention is not to provide a full background on each of the topics, but instead to focus on those parts that are important for understanding the remainder of this thesis. We will introduce in Section 2.2 what the Border Gateway Protocol (BGP) is, its history, and how it relates to the Internet. Then, what anycast is, and how it fits into BGP. We will introduce key concepts of the Domain Name System (DNS) in Section 2.3, on which we base many of the measurements in this thesis. Finally, in Section 2.4 we will briefly describe what Distributed Denial-of-Service (DDoS) attacks are, how they work, and what categories of DDoS there are.

2.1 Reading guide

This chapter provides background information about BGP, DNS and DDoS. Section 2.2 introduces the history of BGP, as well as a basic explanation of how it is used on the Internet. Readers that already have a basic understanding of BGP might want to limit themselves to reading Section 2.2.6, which introduces Anycast. Section 2.3 introduces the concept of the DNS, the underlying protocol, as well as Extension mechanisms for DNS (EDNS) and EDNS Client Subnet (ECS) (Section 2.3.2). Understanding of ECS is important for Chapter 3, which relies on ECS extensively. The final section, Section 2.4, describes what DDoS attacks are and what enables them. This last section is particularly focused on DDoS in the context of Anycast.

2.2 BGP: The Border Gateway Protocol

2.2.1 History

The Border Gateway Protocol (BGP) is a protocol underlying most of the Internet in terms of routing. Fundamentally, it is a path-vector protocol that allows routers belonging to different Autonomous Systems to connect to each other and exchange routes. *Path-vector* means that the protocol makes decisions based on a *path* metric, such as how many routers are crossed, or in the case of BGP, how many Autonomous Systems (ASes). BGP also includes other properties in the route selection process (see Section 2.2.5).

BGP was first introduced in 1989 in Request for Comments (RFC) 1105 [5], building on its predecessor, the Exterior Gateway Protocol, as defined in RFC904 [6]. It was quickly superseded by BGP version 2 in 1990 [7]. The main difference between these two versions are: **1)** the removal of the 8 bit Direction field in the `update` message, which allowed routers to specify the direction of the route with respect to the *graph* of the network and **2)** the addition of the option to support multiple path attributes in an `update` message.

In 1991, BGP version 3 was standardized [8]. The most important change was the addition of a method to prevent two BGP speakers from simultaneously and successfully setting up two connections to each other (one initiated from each side), where only one active connection between two BGP speakers should exist. This version remained current for 3 years, when version 4 was introduced in 1994 [9]–[11], which remains the current version.

The main and most important difference between version 3 and 4 is the introduction of Classless Inter-Domain Routing (CIDR), itself introduced in RFC1519 [12], meaning that the IP(v4) space can be divided into far more parts than before.

Before BGP version 4, and before CIDR, the total IP space, from the perspective of BGP, was divided into 3 classes, A, B and C, corresponding respectively to a /8, /16 and /24. While IP space was still abundantly available in the early days of the Internet, this changed when entities started to require more and more space.

Specifically, the problem was that a B class IP block was much larger than a C class, and there was nothing in between. Thus, if an entity required more than 256 (2^8) addresses, they were assigned 65,536 (2^{16}) addresses, 256$\times$ the number of addresses. Of these B sized blocks there are only 16,384 (2^{14}) available, 16 bits minus 2 bits for the required prefix that identifies the class. CIDR was thus introduced as it became evident that this would lead to a rapid exhaustion of address space, which was then integrated in BGP version 4.

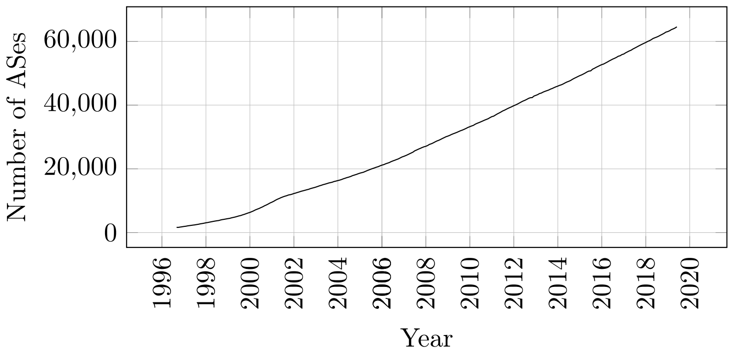


Figure 2.1: Number of Autonomous Systems on the Internet [128].

2.2.2 ASes: Autonomous Systems

An AS is a connected group of one or more IP prefixes run by one or more network operators which has a SINGLE and CLEARLY DEFINED routing policy. – RFC1930

An Autonomous System is essentially an entity that comprises a part of the Internet. Each AS on the Internet is identified by a uniquely assigned number, the Autonomous System Number (ASN). These numbers are assigned by the Regional Internet Registries (RIRs), which in turn depend on the Internet Assigned Numbers Authority (IANA) for allocations. Initially the ASN was a 16-bit number, but in 2007 with the standardization of RFC4893 [13] these were extended to 32-bit, allowing for far more ASes. As of writing, the number of unique ASes that have been assigned an ASN is approaching 100,000 [1]. The number of ASes that are announcing at least a single prefix in the routing system can be seen in Figure 2.1.

ASes are one important part of the Internet, another part are the connections between them. In the coming subsections we will describe how ASes connect to each other, and how route selection decisions are made.

2.2.3 AS Interconnection

Consider the scenario in Figure 2.2, here we see 6 autonomous systems (numbered 1 through 6), some of which are connected to each other. This can be considered a small version of the Internet. In this example, the only AS that announces any IP-space is AS1: 1.1.1.0/24.

Internally, the ASes can use any means to distribute routes between their own routers. Externally, however, it is typically required to use BGP, which

[1]`https://www-public.imtbs-tsp.eu/~maigron/RIR_Stats/RIR_Delegations/World/ASN-ByNb.html`

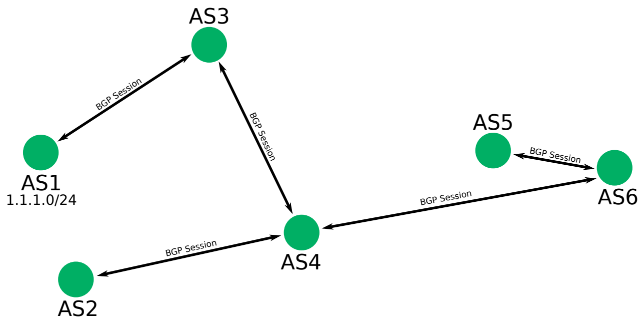


Figure 2.2: Example of a BGP connected network.

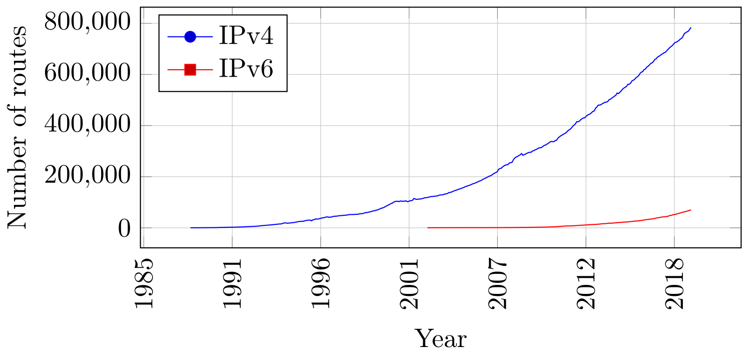


Figure 2.3: Increase of BGP routing table size over time [128].

is the common language spoken between ASes. BGP allows ASes to announce routes that they have learned towards certain IP prefixes to other ASes, so that those other ASes can learn where to route traffic destined for those IP-prefixes. If BGP is also used internally, it is referred to as iBGP, otherwise as eBGP. If a route is accepted then it is added into the routers routing table. Currently, a full route table consists of approximately 800,000 entries. As can be seen in Figure 2.3, this number keeps increasing as the Internet grows.

In the given scenario, traffic from AS6 trying to reach a system in 1.1.1.0/24 will take the path AS6-AS4-AS3-AS1. In the real Internet there are specific ASes that have the role of providing “Internet connectivity” to the other ASes in exchange for money. These are known as transit providers. Examples of these include CenturyLink, AT&T, NTT Communications, and, in the Netherlands, KPN.

2.2.4 IXPs: Internet Exchange Points

As mentioned in the previous subsection, there are ASes that have the sole goal of providing connectivity to other ASes, in exchange for money. However, in some cases ASes also exchange traffic directly, this can be implemented by interconnecting the two ASes physically, or by peering via a so called Internet Exchange Point. The idea is that the Internet Exchange Point (IXP) provides a switch to which an AS can connect, and through which it can then peer with (many) other ASes. Compared to directly linking this saves manual effort as well as saving resources in terms of physical network ports.

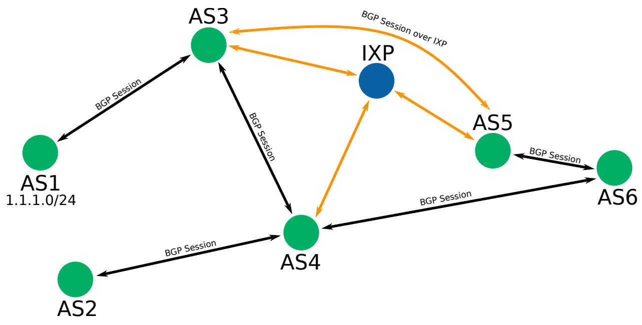


Figure 2.4: Example of a BGP connected network, including an IXP.

Compared to the previous scenario in Figure 2.2, Figure 2.4 also includes an IXP. There are three ASes connected to it, and two of the ASes actually "peer" with each other. Theoretically, there is no requirement for any AS connected to an IXP to actually peer with any other ASes, in which case no traffic will be exchanged over the IXP. This demonstrates how two ASes can connect directly through each other through the use of an IXP. In practice, IXPs typically facilitate the peering process by providing a so-called route server, which allows ASes to exchange routes with other ASes without having to negotiate with each AS that peers at that IXP separately, at the cost of some flexibility.

2.2.5 Route selection

It is quite common to have multiple possible routes to the same destination. For example when an IP prefix can be reached both via an IXP, as well as via a transit provider, or if it can be reached via two or more different transit providers. In such cases the selection process works as follows according to the standard (RFC4271, section 9.1.2):

1. Select the route which has the highest local preference (this can be manually determined by the operator).
2. Select the route which has the shortest AS_PATH, in case of a tie remove all routes with a longer path from consideration.

3. Select the route with the lowest origin number, in case of a tie: remove all routes with a higher origin number.
4. Select the route with the highest preferred MULTI_EXIT_DISC attribute, but only between routes learned from the same neighbor AS. In case of a tie: remove all routes that are less-preferred.
5. Select the route that was learned via external BGP, in case of a tie: remove all routes that were learned via internal BGP.
6. Select the route with the lowest interior cost, in case of a tie: remove all those with higher interior costs.
7. Select the route which was learned from the BGP neighbor with the lowest BGP Identifier value, in case of a tie: remove all those with a higher value.
8. Select the route which was learned from the lowest peer address.

Most real ASes on the Internet are opaque, i.e. they give no insight into their decision processes or the data that those decisions are made on. However, in some cases a so called looking-glass is provided [14]. This allows external users to view the routing table on routers within an AS, depending on the specific looking-glass in some cases also routes that have not been selected as the best are shown. These are provided for debugging purposes, for example in the case a sub-optimal route is selected, or to verify that a specific prefix is visible at an AS.

2.2.6 Anycast

IP anycast is an addressing and routing strategy in which multiple physical servers in the Internet are configured with the same logical IP address. This strategy is widely used to achieve high availability and redundancy of services over the Internet, such as DNS and Content Delivery Networks (CDNs).

IP anycast takes advantage of the route selection mechanism of BGP. Users are routed to the anycast instance that has the highest preference according to the route selection algorithm (see Section 2.2.5). The term anycast catchment refers to the distribution of clients between the anycast sites, i.e. the mapping of which client is routed to which anycast instance.

In this thesis we talk about the *catchment* of an anycast service *as a whole*, which means the complete mapping of each user and the anycast instance that it reaches. In contrast, the *catchment* of a specific anycast instance means that we are referring to *just* those users that are routed to that instance.

Anycast catchments can be hard to predict mainly due to a large variety of routing policies that are applied within and between Autonomous Systems (ASes) [15], [16], see also Section 2.2.5.

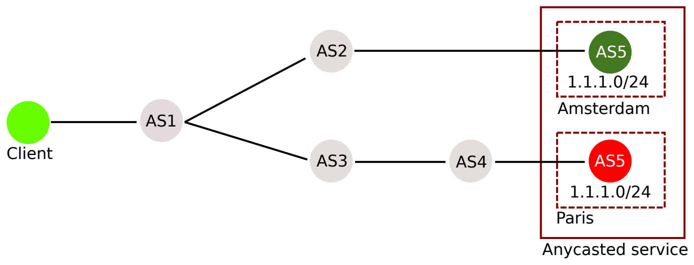


Figure 2.5: Client connecting to 1.1.1.1, an anycasted service.

Examples of services that are using anycast are the DNS (e.g. the Root DNS as well as many others, see also Section 2.3), DDoS mitigation providers (e.g. Akamai, Cloudflare) and CDNs

In Figure 2.5 we show a simple routing graph containing a client connecting to an IP address in the IP prefix 1.1.1.0/24. In this case, AS2 has two possible routes towards this destination prefix, one leading to a location in Amsterdam, and the other leading to a location in Paris. BGP Route selection determines which route will be selected as the best and, barring local preference settings, will pick Amsterdam as the closest, due to it having a shorter AS Path (AS2, AS5 (length 2), versus AS3, AS4, AS5 (length 3)).

2.3 DNS: The Domain Name System

In this thesis we structure many of our experiments around the DNS [17]. In this section we will briefly explain what the DNS is, and explain in more detail a number of aspects of it that are particularly relevant for the rest of this thesis.

In essence the DNS is what provides a mapping between a domain name (e.g. *utwente.nl*), and its corresponding IP address (e.g. *130.89.3.249*). This allows humans to use meaningful names, which computers can automatically translate to an address usable for IP routing protocols. The DNS stores this data in so-called A-records (for IPv4) and AAAA-records (for IPv6) [17]. Translating from a domain name to an IP address is referred to as "resolving", a service that performs this function is called a "resolver". While that is the main function of the DNS, it also allows for resolving different types of data, for example where e-mail should be delivered for a specific domain in a so called MX-record [18].

The DNS itself is structured as a tree, where each part of a domain, separated by a dot, potentially falls under a different authority. In Figure 2.6 we show a number of domain names, and how they are represented in the DNS. Note that domain names actually have a dot at the end which is typically not displayed to the user, e.g. "utwente.nl" actually means "utwente.nl.", where the final dot represents the root.

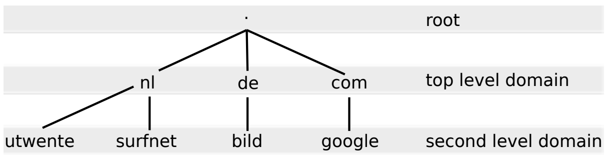


Figure 2.6: The Domain Name System is structured like a tree. Showing the domains *utwente.nl*, *surfnet.nl*, *bild.de* and *google.com.*

For a resolver to translate a domain name to an address, it always has to start at the root authoritative DNS server, for which the addresses must be hard coded (bootstrapped). From there, the root will indicate to which authoritative server the authority over a specific Top-Level Domain (TLD) has been delegated. In Figure 2.7 we show the complete process for *www.utwente.nl*. Aside from the record type (A vs AAAA) the process is identical for IPv4 and IPv6.

2.3.1 Protocol

The DNS is a Query/Response protocol, using a single message structure for both, which is shown in Figure 2.8. Both queries and responses are typically transmitted using the User Datagram Protocol (UDP), but may fall back to the Transmission Control Protocol (TCP), both use port 53. A client sending

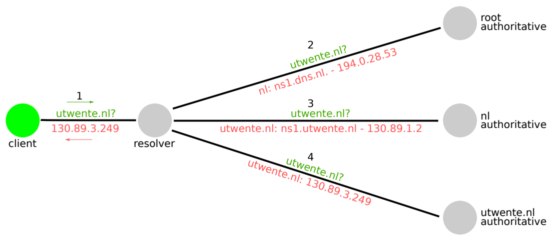


Figure 2.7: Steps taken to resolve *utwente.nl* recursively.

```
+---------------------+
|        Header       |
+---------------------+
|       Question      | the question for the name server
+---------------------+
|        Answer       | RRs answering the question
+---------------------+
|      Authority      | RRs pointing toward an authority
+---------------------+
|      Additional     | RRs holding additional information
+---------------------+
```

Figure 2.8: Structure of DNS Message (RFC1035)

a query leaves the *Answer*, *Authority* and *Additional* fields empty, which can be filled in by the server in the response. The server copies the message in the query when it answers, filling the fields as required, this means that the server also sends the question, verbatim, back to the client. The *Answer* field is reserved for data that contains an actual answer to the questions asked by the client, as opposed to a delegation to a different authoritative name server, for which the *Authority* field is to be used. Additional answer data, which is neither an answer to the question nor a delegation to a different authoritative name server can be included in the *Additional* field. This is typically used to provide the client with the IP addresses (A and/or AAAA records) corresponding to the authorities specified in the *Authority* field, for efficiency reasons.

The *Header* field is filled according to Figure 2.9. It contains a random *ID*, to be able to match responses to queries. The *QR* field indicates whether the message is a query or a response. It also contains several flags, for example to request the resolver to perform recursion (RD), or for the resolver to indicate whether recursion is available (RA). It also contains several fields to indicate the number of question records, answer records, authority records and additional records the message contains.

Each question record is structured as shown in Figure 2.10. The *qname* field indicates the query name, in other words, the domain name (e.g. utwente.nl, see Figure 2.7). The *qtype* indicates the type of record that the query is for,

```
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      ID                       |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|QR|   Opcode  |AA|TC|RD|RA|   Z    |   RCODE   |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    QDCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    ANCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    NSCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    ARCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Figure 2.9: Structure of DNS Message Header [17].

```
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                                               |
/                     QNAME                     /
/                                               /
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                     QTYPE                     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                     QCLASS                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Figure 2.10: Structure of DNS Message Question [17].

see Table 2.1 for a non-exhaustive list of possible query types. For example to retrieve the A record of a domain this would be set to 1. Finally the *qclass* indicates the query class, there are only two classes in widespread use, the first is Internet (IN, code 1), which is the normal class. The other is Chaos (CH, code 3), which is typically used to retrieve metadata associated with the name server. The name Chaos referred to Chaosnet, which was conceived as an alternative to the Internet, the use of the class nowadays is unrelated to its origin. Note that while it is technically possible to include multiple questions in a single query, this is not supported by any current implementation. The main reason for this is that the flags in the header of the query are not repeatable, and are only applicable to a single question record. Having multiple questions and answers would therefore result in ambiguity.

2.3.2 EDNS and EDNS Client Subnet

EDNS: The original DNS protocol [17] puts limits on both the size of DNS responses (512 bytes in a UDP datagram) and what options and flags a DNS message can have. However, many modern applications of the DNS have requirements that exceed these limits. For that reason, RFC6891 [19] introduced

Type	Type id.	Description
A	1	Address record
NS	2	Name server record
CNAME	5	Canonical name record
SOA	6	Start of authority record
PTR	12	Pointer record
MX	15	Mail exchange record
TXT	16	Text record
AAAA	28	IPv6 Address record
SRV	33	Service locator

Table 2.1: Common DNS query types [17].

the Extension mechanisms for DNS (EDNS). EDNS uses a special pseudo-record in the additional section of a DNS query or response. This so-called `OPT` record specifies EDNS parameters, it can be used to specify additional flags and can be used to specify new DNS options (*e.g.,*to increase the maximum message size). The options, are encoded in the form of <*tag,value*> pairs and can be used to convey arbitrary metadata about a DNS message.

Client Subnet: Many CDNs and other applications make use of the IP address from which queries are made to their authoritative DNS servers. The primary reason for this is to geolocate the user by performing an IP lookup, which provide the operator with a rough approximation of the location of the user. This information can then be used to direct this user to the nearest server, to redirect them to locale specific content or provide location-based access control. However, with the rise of public DNS resolvers, such as Google Public DNS (GPDNS), the accuracy of this method has strongly declined, as all queries originated from such a resolver appear to be coming from the physical location of that resolver. Even if the geo IP database is accurate enough to pinpoint the resolver location, they might not reside in the same country as the user, or might still not be accurate enough even if the country is correctly identified.

To solve the geolocation problem, the ECS option was introduced in RFC 7871 [20]. This option can be used by DNS resolvers to provide information about where a query originated. Specifically, DNS resolver includes two fields in the ECS option: the *IP prefix* from which the query originated and a *source prefix length* field that specifies the size of the provided prefix (e.g. `/24`). For privacy reasons, DNS resolvers typically limit how specific the scope is that they send in a request. The ECS standard [20] recommends using a maximum scope of `/24` for IPv4 and `/56` for IPv6. An authoritative name server can then use this information to decide which region-specific response to return to a query. To ensure that responses from the authoritative name servers are only cached for users in the correct prefix, the authoritative name server also includes its own *scope prefix length* field in the response. This field must be used by the DNS resolver when caching the response.

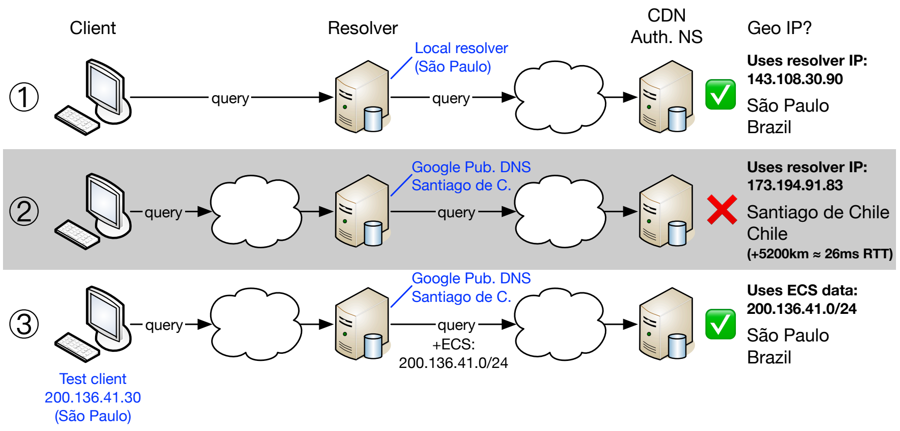

Client
Resolver
CDN Auth. NS
Geo IP?
①
query
Local resolver (São Paulo)
query
Uses resolver IP: 143.108.30.90
São Paulo Brazil
②
query
Google Pub. DNS Santiago de C.
query
Uses resolver IP: 173.194.91.83
Santiago de Chile Chile
(+5200km ≈ 26ms RTT)
③
query
Google Pub. DNS Santiago de C.
query +ECS: 200.136.41.0/24
Uses ECS data: 200.136.41.0/24
São Paulo Brazil
Test client 200.136.41.30 (São Paulo)

Figure 2.11: Explanation of EDNS0 Client Subnet

Figure 2.11 shows an example of 1) a local resolver, 2) a public resolver without ECS and 3) a public resolver with ECS. The figure shows the potential impact of not using ECS for a public resolver. The example is based on a client we control, located in São Paulo, Brazil. Without ECS, a CDN using Geo IP will assume this client is in Santiago de Chile, 2600km away as the crow flies, adding a potential 26ms to each network round-trip.

2.3.3 QNAME Minimization

When DNS was first introduced in the 1980s, there was no consideration for security and privacy. These topics have now gained considerable importance, leading to a plethora of RFCs that add security and privacy to the DNS. For example, DNSSEC [21]–[23] introduces end-to-end authenticity and integrity, but no privacy. More recently, DNS-over-TLS [24] and DNS-over-HTTPS [25] added transport security. "Aggressive Use of DNSSEC-Validated Cache" [26], reduces unnecessary leaks of non-existing domain names. Furthermore, running a local copy of the root zone at a resolver avoids sending queries to root servers completely [27].

Typically, resolvers send the full *qname* to each authoritative name server involved in a lookup. Consequently, root servers receive the same query as the final authoritative name server. Since the Internet Engineering Task Force (IETF) states that Internet protocols should minimize the data used to what is necessary to perform a task (see RFC6973 [28]), *qname minimization* (qmin) was introduced to bring an end to this. Resolvers that implement *qmin* only query name servers with a name stripped to one label more than what that name server is known to be authoritative for. E.g., when querying for *a.b.example.com*, the resolver will first query the root for *.com*, instead of *a.b.example.com*. The reference algorithm for *qmin* also hides the original query type by using the

Standard DNS resolution		
a.b.example.com.	`A`	$\rightarrow$.
com.	*`NS`*	$\leftarrow$.
a.b.example.com	`A`	$\rightarrow$ com.
example.com	*`NS`*	$\leftarrow$ *com.*
a.b.example.com	`A`	$\rightarrow$ example.com.
a.b.example.com	*`A`*	$\leftarrow$ *example.com.*
***qmin* Reference (RFC 7816)**		
com.	`NS`	$\rightarrow$.
com.	*`NS`*	$\leftarrow$.
example.com	`NS`	$\rightarrow$ com.
example.com	*`NS`*	$\leftarrow$ *com.*
b.example.com	`NS`	$\rightarrow$ example.com.
b.example.com	*`NS`*	$\leftarrow$ *example.com*
a.b.example.com	`NS`	$\rightarrow$ example.com.
a.b.example.com	*`NS`*	$\leftarrow$ *example.com*
a.b.example.com	`A`	$\rightarrow$ example.com.
a.b.example.com	*`A`*	$\leftarrow$ *example.com*

Table 2.2: DNS queries and responses without (top) and with (bottom) *qmin*.

`NS` type instead of the original until the last query. In Table 2.2 we show what queries are performed for both standard DNS and the *qmin* reference implementation.

This reference algorithm, however, faces two challenges on the real Internet: First, it does not handle configuration errors in the DNS well [29]. E.g., in case *b.domain.example* does not have any RRs but *a.b.domain.example* does, a name server should respond with `NOERROR` for a query to *b.domain.example*[30], but in fact often responds with `NXDOMAIN`, or another invalid `RCODE`. This would force resolvers that conform to the standard to stop querying and thereby not successfully resolve the query. Also, operators report other issues, such as name servers that do not respond to `NS` queries, which would break *qmin* as well [129].

Second, *qmin* can lead to a large number of queries. For example, a name with 20 labels would make the resolver issue 21 queries to authoritative name servers, causing excessive load at the resolver and authoritative. Attackers can abuse this for Denial-of-Service (DoS) attacks by querying excessively long names for victim domains.
Both of these issues led resolver implementors to modify their *qmin* implementations, as well as adding so called "strict" and "relaxed" modes, which we investigate in Section 3.4.

As of October 2018, three major DNS resolvers support *qmin*. Unbound supports *qmin* since late 2015 and turned relaxed *qmin* on by default in May 2018 [129]. Knot resolver uses relaxed *qmin* since its initial release in May 2016[130], and the recursive resolver of BIND supports *qmin* and turned the

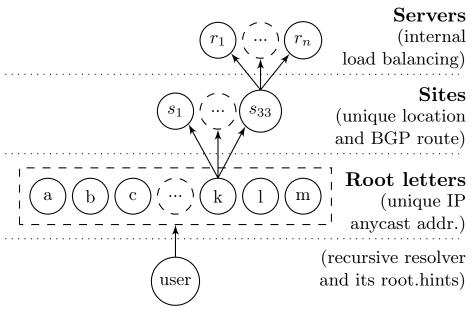

Servers
(internal load balancing)
Sites
(unique location and BGP route)
Root letters
(unique IP anycast addr.)
(recursive resolver and its root.hints)
r_1
...
r_n
s_1
s_{33}
a
b
c
k
l
m
user

Figure 2.12: Root DNS structure, terminology, and mechanisms in use at each level.

Letter	Operator	Sites
A	Verisign, Inc.	28/0
B	Information Sciences Institute	3/0
C	Cogent Communications	10/0
D	University of Maryland	154/0
E	NASA Ames Research Center	247/7
F	Internet Systems Consortium, Inc.	235/1
G	U.S. DOD Network Information Center	6/0
H	U.S. Army Research Lab	2/0
I	Netnod	49/8
J	Verisign, Inc.	164/0
K	RIPE NCC	69/0
L	ICANN	168/35
M	WIDE Project	9/0

Table 2.3: The 13 Root Letters, each operating a separate DNS service, with their reported architecture (number of sites with global/local sites [133], Date: 2019-08-26.

relaxed mode on by default in July 2018 [131]. Another frequently used resolver, PowerDNS Recursor, does not support *qmin* yet [132].

2.3.4 Root DNS

The Root DNS service is implemented with several mechanisms operating at different levels (Figure 2.12): a `root.hints` file which bootstraps the IP addresses of the root services in the recursive resolver. Multiple instances of the root (the root letters) operating on different IP addresses. Each of these IP addresses may be anycast using BGP. Then, at each of the anycast sites, one or multiple (using load-balancing) servers handle the actual requests.

The Root DNS is implemented by 13 separate DNS services (Table 2.3), each running on a different IP address, but sharing a common master data source.

These are called the 13 *DNS Root Letter Services* (or just the "Root Letters" for short), since each is assigned a letter from A to M and identified as `<letter>.root-servers.net`. The letters are operated by 12 independent organizations (Verisign operates both A and J), and each letter has a different architecture, an intentional diversity designed to provide robustness, both against DDoS attacks, as well as software and/or hardware vulnerabilities.

Root Letters have different policies, architectures, and sizes, as shown in Table 2.3. Some letters constrain routing to some sites to be *local*, using BGP policies (such as `NOPEER` and `NO_EXPORT`) to limit routing to that site to only itself neighboring ASes. Routing for *global* sites, by contrast, is not constrained.

2.4 DDoS: Distributed Denial-of-Service

In essence, a Distributed Denial of Service attack is a type of attack that aims to prevent legitimate users from accessing a service, i.e. denying service. A DDoS attack can take many shapes, and have various sources. For example, an attack can be:

1. **Volumetric:** the goal is to overwhelm the target with traffic.
2. **Protocol based (layer 3/4):** e.g. SYN floods, the goal is to make the receiving server consume more resources than it has.
3. **Application based (layer 7):** the goal is to overwhelm an application such as Apache, or Nginx, or underlying databases by performing many requests.

Aside from the type of attack, the source can also be varied:

1. **Servers hosting UDP-based services:** for example many DNS servers can be used to amplify attack traffic, by requesting information from these servers while spoofing the IP address of the target of the attack.
2. **Botnets:** a group consisting of hacked systems.
3. **Direct:** uncommonly, an attacker can choose to perform an attack directly from its own systems, either alone or in coordination with others.

There appear to be three main factors that enable today's DDoS attacks: First, source-address spoofing allows a single machine to masquerade as many machines, making filtering difficult, while also allowing attackers to make requests apparently on behalf of their victim. Secondly, some protocols, such as DNS and Network Time Protocol (NTP), have the property that they allow queries using UDP, which is connection-less and can thus be used with a spoofed source-address. Also, perhaps more importantly, the responses to these queries can be much larger than the requests, essentially amplifying the attack traffic [31]. Third, botnets of thousands of machines are widespread [32], making vast attacks possible even without spoofing and amplification. A botnet typically consists of hacked devices, historically personal systems or servers, but with the rise of the Internet-of-Things such devices are also becoming popular. Large attacks ranged from 50–540 Gb/s [134] in 2016, and have increased since then to 1.3 Tb/s and higher.

Many protocol-level defenses against DDoS attacks have been proposed. Source-address validation prevents spoofing [33]. Response-rate limiting [135] reduces the effect of amplification. Protocol changes such as DNS cookies [34] or broader use of TCP [35] can blunt the risks of UDP. While these approaches reduce the effects of a DoS attack, they cannot eliminate it. Moreover, deployment rates of these approaches have been slow [36], in part because there is a mismatch of incentives between who must deploy these tools (all Internet Service Providers (ISPs)) and the victims of attacks.

There are commercial services that promise to defend against DDoS, either by offering DDoS-filtering as a service (as provided by Verizon, NTT, and many others), or by providing a service that adapts to DDoS attacks (such as Akamai [37], Cloudflare [136], and others).

2.4.1 Anycast and DDoS: Design Options

In this thesis we are primarily interested in how anycast can be employed to mitigate DDoS attacks. So how should an anycast service react to the stress of a DDoS attack? Here we provide a theoretical evaluation of options, which is based on a research paper [3]. In Chapter 4 we will supplement these with empirical observations.

A site under stress, overloaded with incoming traffic, has two options. It can *withdraw* routes to some or all of its neighbors, shrinking its catchment and shifting both legitimate and attack traffic to other anycast sites. Possibly those sites will have greater capacity and service the queries. Alternatively, it can become a *degraded absorber*, continuing to operate, but with overloaded ingress routers, dropping some incoming legitimate requests due to buffers overflowing. However, continued operation will also absorb traffic from attackers in its catchment, protecting other anycast sites [137].

These options represent different uses of an anycast deployment. A withdrawal strategy causes anycast to respond as a waterbed, with stress displacing queries from one site to others. The absorption strategy behaves as a conventional mattress, “compressing” under load, with queries getting delayed or dropped. We investigate these behaviors in Chapter 4.

Although described as strategies, these options are the *result* of several factors: the combination of operator and host ISP routing policy, routing implementations withdrawing under load [38], the nature of the attack, and the locations of the sites and attackers. Some policies are explicit, such as the choice of local-only anycast sites, or operators removing a site for maintenance or modifying routing to manage load. However, under stress, the choices of

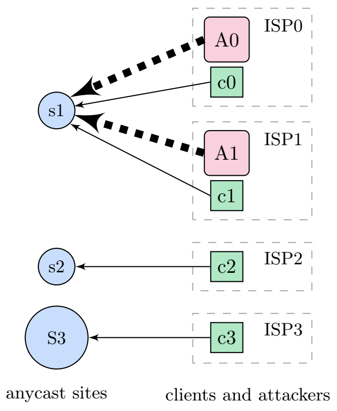


Figure 2.13: An example anycast deployment under stress.

withdrawal and absorption can also be a result that *emerges* from a mix of explicit choices and implementation details, such as BGP timeout values.

We can illustrate the options an operator has in terms of mitigation strategies with the following thought experiment. Consider the anycast system in Figure 2.13, it has three anycast sites: s_1, s_2, S_3, four clients c_0 and c_1 in s_1's catchment, with c_2 in s_2 and c_3 in S_3's. Let A_0 represent both the identity of the attacker and the volume of its attack traffic, and s_1 represent the site and its capacity.

The best choice of defense depends on the relative sizes of attack traffic reaching each site. For simplicity, we can ignore legitimate traffic (c_*), since DNS deployments are greatly overprovisioned ($c_* \ll A_*$). Overprovisioning by 3× peak traffic is expected [39].

To consider alternative responses to attack we evaluate a deployment where $s_1 = s_2$ and $S_3 = 10s_1$, as attack strength $A_0 = A_1$ increases. We measure the effects of the attack by the total number of served clients (H, "happiness").

1. If $A_0 + A_1 < s_1$, then the attack does not hurt users of the service, $H = 4$.

2. If $A_0 + A_1 > s_1$ and $A_0 < s_1$, then s_1 is overwhelmed ($H = 2$) but can shed load. If it withdraws its route to ISP1, A_1 and c_1 shift to s_2 and if $A_1 < s_2$, then and all clients are served: $H = 4$.

3. If $A_0 > s_1$ and $A_0 + A_1 < S_3$, then the attackers can overwhelm the smaller site, but not the bigger site. *Both* s_1 and s_2 should withdraw all routes and let the large site S_3 handle all traffic, for $H = 4$.

4. If $A_0 > s_1$, $A_0 + A_1 > S_3$, but $A_1 < S_3$, one can re-route ISP1 (with A_1 and c_1) to S_3, for $H = 3$.

5. If $A_0 > S_3$, the attack can overwhelm *any* site; making no change is optimal. s_1 becomes a degraded absorber and protects the other sites from the attack, at the cost of clients c_0 and c_1. $H = 2$.

(Of course, withdrawing routes in response to attacks may also increase latency as catchments change. The definition of H we chose here ignores latency as a secondary factor, focusing only on the ability to respond.)

Implications of this model: This model has several important implications, both about the range of *possible* strategies, which are *practical* today, and directions to explore in the *future*.

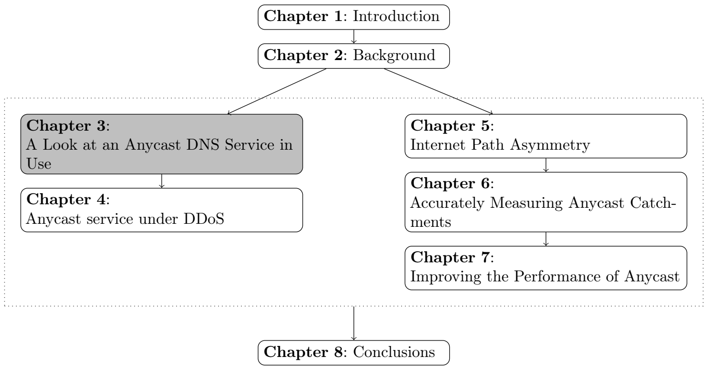
This thought experiment shows that for small attacks, the *withdraw* strategy can improve service by spreading the attack (although perhaps counter-intuitive, less can be more). For large attacks, *degraded absorbers* are necessary to protect some clients, at the cost of others. We cannot directly apply these rules in this thesis, specifically Chapter 4, since we know neither site capacity (something generally kept private by operators as a defensive measure), nor how much attack traffic reaches each site (a function of how attackers align with catchment, again, both unknown to us).

A second implication is that choice of optimal strategy is very sensitive to actual conditions—which of the five cases applies depend on attack rate, location, and site capacity. The practical corollary is that choosing the optimal strategy is not easy for operators, either. Attack traffic volumes are unknown to operators when the attack exceeds capacity; attack locations are unknown, due to source address spoofing; the effects of route changes are difficult to predict, due to unknown attack locations; and route changes are difficult to implement, since routing involves multiple parties. In the face of uncertainty about attack size and location, absorption is a good default strategy. However, route withdrawals may occur due to BGP session failure, caused by the DDoS, so they may occur regardless of operator intervention.

Finally, a key implication of this model is that there *can be* better possible strategies than just absorbing attacks. As described above, they require information about attack volume and location that is not available today, but their development is promising future work.

CHAPTER 3

A Look at an Anycast DNS Service in Use


Chapter 1: Introduction
Chapter 2: Background
Chapter 3: A Look at an Anycast DNS Service in Use
Chapter 4: Anycast service under DDoS
Chapter 5: Internet Path Asymmetry
Chapter 6: Accurately Measuring Anycast Catchments
Chapter 7: Improving the Performance of Anycast
Chapter 8: Conclusions

In this chapter we aim to investigate a large scale anycast network, and find out what we can learn from it. We achieve that by looking at the Google Public DNS (GPDNS), which was launched in 2009. Since that time it has grown to become the largest Domain Name System (DNS) server in existence, running on an anycast network with many Points-of-Presence (PoPs). We leverage EDNS Client Subnet (ECS), which is explained in the previous chapter (Section 2.3.2), to observe where clients of GPDNS are routed, from the perspective of a large authoritative DNS server. Using ECS we can precisely determine which PoP was used to handle a specific query. In this chapter we show that while GPDNS has PoPs in many countries, traffic is frequently routed out of country. This routing behavior can reduce performance, and expose DNS requests to state level surveillance. We also show that end users are often served by a suboptimal PoP. Second, we show that end users switch to GPDNS en masse *when their Internet Service Provider (ISP)'s resolver is unresponsive, and do not switch back. Third, we find that many e-mail providers configure GPDNS as resolver on their servers, causing serious privacy concerns. Finally, we perform an investigation the privacy enhancing technique named qname minimization that partially counteracts the previously found privacy issue. The key takeaway from this chapter is that operating an anycast network requires careful design, and network engineering in the context of anycast is difficult to get right. The work presented in this chapter was published as two research papers [40], [41].*

3.1 Introduction

The DNS is an important part of what the Internet is today. It *resolves* domain names to IP addresses. As described in Section 2.3, the DNS is a hierarchical system where different *name servers* are responsible for different parts of a domain name. In order to resolve a domain name, a so-called recursive resolver queries each name server responsible for part of a domain name in turn, until it has the final answer. A customer of an ISP, typically uses a recursive resolver that is provided by the ISP, and that is usually automatically configured. While most ISPs provide their own recursive resolver for their customers, it is usually not mandatory to use these. Customers can either run their own resolver, or use a third party resolver. Examples of organizations offering such third-party services include OpenDNS, Quad9 and Google. There are many reasons why an end-user might use a different resolver than the one operated by their ISP, such as stability, performance, privacy or to circumvent censorship.

In this chapter we mainly focus on one particular public resolver, GPDNS, which was launched in 2009. Since its inception, GPDNS has grown to be the largest public resolver in existence, serving hundreds of billions of requests per day [138]. While GPDNS uses only a few public IP addresses, its servers have a global presence. Google uses anycast to ensure traffic to GPDNS is routed to a nearby PoP. This increases resilience and reduces latency for clients.

Now while services such as GPDNS have a global presence, they typically do not have PoPs in every country. In early 2018, GPDNS had 21 active locations on 5 continents. It turns out that this poses challenges for Content Delivery Networks (CDNs). CDNs frequently rely on the geo-location of the IP address of recursive resolvers as a proxy for the location of end customers. This information is then used to route requests to content caches near the end customer and to serve local content. The underlying assumption is that DNS queries are typically handled by a resolver 'near' the end customer, *e.g.,* their ISP's resolver. In the case of a public resolver such as GPDNS, this assumption breaks down as requests appear to come from the PoP that handled the user request.

To address this problem an extension to the DNS called *ECS* [20] was introduced. This extension allows recursive resolvers to include part of the IP address of the client that sent a query in requests to authoritative name servers. This can assist CDNs in more accurately determining where clients using public resolvers come from.

Interestingly, the use of ECS by name servers has unintended side effects. By collecting ECS-enabled queries from CDNs that support the extension, it becomes possible to study the geographic distribution of their services, and how clients are mapped to certain services, as a number of existing studies have shown [42]–[46]. ECS, however, can also be used to examine how a public resolver works and is used. For more information about ECS refer to Section 2.3.2.

Continent	PoPs	Prefixes IPv4	IPv6	# of queries
Asia	4	13	4	391 523 557
Europe	6	19	6	1 800 743 147
North America	8	40	8	1 450 006 164
Oceania	1	3	1	2 633 248
South America	2	3	2	29 143 338
Total	*21*	*78*	*21*	*3 711 406 022*

Table 3.1: Distribution of Google PoPs (state of 8 November 2017) and received queries at our authoritative name server

In this chapter, we are the first to study a large scale public DNS resolver (GPDNS) over a 2.5-year period using passive observations of ECS data in DNS queries collected at a major authoritative name server.

3.2 Methodology

As discussed in the introduction, ECS provides a unique opportunity to observe the behaviour of large public DNS resolvers. In this section, we outline how we collected our data, and how we will use this to study one particular public DNS resolver operator: Google.

3.2.1 Data collection

Collection point: We used one of the authoritative name servers of SURFnet, the National Research and Education Network (NREN) in The Netherlands, to passively collect DNS queries from GPDNS. Data was collected from the end of June 2015 to the end of December 2017. Only DNS queries that include an ECS option are collected, and for these queries, we record the *origin IP* of the query (i.e. the IP of the Google resolver that sent the query), and the *IP prefix* and *source prefix length* included in the ECS option. In addition to this, we use CAIDA's IP prefix to Autonomous System (AS) dataset [139] to map the ECS IP prefix to an AS and we use the free IP2Location [140] dataset to map the ECS IP prefix to a country.

The SURFnet name server we used is authoritative for approximately 10,000 DNS zones, including a number of popular public suffixes[1] such as `.ac.uk`, `.gov.uk` and `.ac.be`. As a result of this, this name server sees a wide spread of queries from all over the Internet and world. Note though, as we discuss in more detail below in Section 3.2.3, we do expect bias in which Google PoPs send traffic to this server, due to resolver-to-authoritative Round Trip Time (RTT) optimisation. Table 3.1 shows an overview of the data we collected for this study, broken down per continent from which queries originated.

[1]For an explanation of public suffixes, see `https://publicsuffix.org`.

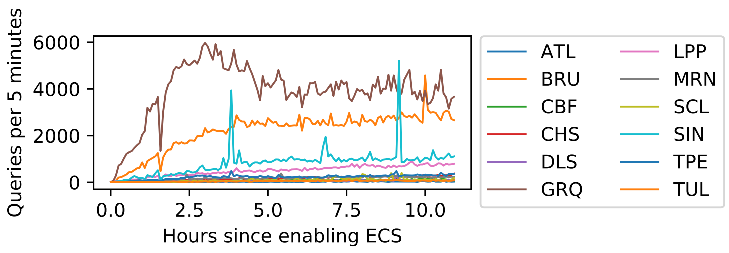


Figure 3.1: Ramp-up of GPDNS detecting and enabling ECS

BIND patch: GPDNS automatically detects support for ECS on authoritative name servers. In order to do this, Google regularly sends probing queries that include an ECS option. If an authoritative name server includes an ECS option in the response, this is interpreted as an indication of support. To ensure that Google would detect our name server as ECS-capable, we implemented a patch for the popular BIND DNS implementation. After this patch, BIND will accept the ECS option, and will include an ECS option in the response that mirrors the source address and prefix length in the scope prefix length field. Figure 3.1 shows the number of ECS-enabled queries per 5 minutes from Google PoPs increasing after we have enabled ECS on our patched name server as resolvers at these PoPs detect support for ECS.

Ethical Considerations: As the ECS standard [20] already specifies, there are inherent privacy concerns in the protocol, as a resolver that supports ECS includes a (sometimes significant) part of the client's IP address in queries. We are interested in how clients are routed to GPDNS and in general terms how GPDNS is used at the network operator level. Therefore, to protect user privacy, we take two measures: 1) we do not store query names, we only record query types and 2) we aggregate ECS client IP prefixes at the AS level when analysing the data, with one exception; if we believe the prefix contains servers that use GPDNS (rather than individuals), we analyze if certain types of hosts (specifically, e-mail servers) exist in these prefixes (Section 3.3.5). See Section 3.6 for further discussion on this topic.

A secondary concern is the effect on query and cache efficiency. While we implement ECS on the authoritative name server where we collect data, we do not differentiate DNS responses based on ECS. Since DNS resolvers that implement ECS should cache responses based on the ECS information, this may impact caching efficiency. Consequently, GPDNS may have to cache responses from our patched name server for every client prefix they send in ECS-enabled queries. The standard [20], however, provides clear guidelines for resolver implementers to avoid cache pollution. In addition to these guidelines, the impact of us implementing ECS will be limited as the other authoritative name servers for domains for which our patched name server is authoritative do not implement ECS. In many cases this means only one in four queries sent from Google will result in an ECS-enabled response (of course depending on how Google's resolvers distribute queries over the set of authoritative name servers for a domain).

3.2.2 Resolver IP to Point-of-Presence mapping

While end-users query GPDNS via the front-end IP addresses `8.8.8.8` and `8.8.4.4` or their IPv6 counterparts, an individual Google DNS resolver then uses different IP addresses to actually resolve the query. The IP-prefixes that are used are published and are frequently updated [141]. Google identifies their PoPs using the three-letter International Air Transport Association (IATA) code of the nearest airport. During the first part of the measurement period we did not store the mapping of prefixes to Google PoPs, but we recovered it using The Wayback Machine (TWM) [142]. Specifically, we collected 25 mappings between the start of our measurement period and the 3rd of August 2017 through TWM. From that point onward, we collected the mapping on a daily basis directly from Google through a DNS query, as described in Google's FAQ [141].

Figure 3.2 shows the number of prefixes associated with each PoP and how this varies over time. As the figure shows, several new PoPs were added over the period covered by our dataset, for example, approximately halfway through 2017 the Sydney PoP was added. We also observe, over the total duration, 4 instances where a prefix was reassigned to a different PoP. It is likely that due to the significant delay in mappings which we obtained via TWM we mismapped a portion of the traffic when such a reassignment occured. However, considering the large amount of total prefixes, the vast majority of which were not reassigned at any point, the impact of this is likely to be small. For our study, we use the prefix-to-PoP mapping we recorded to map queries to Google PoPs based on the prefix in which the source IP of a query is contained. In total we were unable to map 26 548 020 queries to their corresponding PoP (0.7% of all queries in our dataset).

3.2.3 Distribution of queries over authoritative name servers

As we discussed in Section 3.2.1, we collect data on a single authoritative name server. The median number of name servers configured per DNS zone on that name server, however, is 4, which means that not all queries from Google for domains hosted on that name server will be sent to that particular name server. Typical resolver implementations will distribute queries over all authoritative name servers for a domain, usually favoring servers with shorter RTTs [47].

While it is infeasible to exhaustively determine RTTs from all Google PoPs to all authoritative name servers for domains for which our test server is also authoritative, we did want to get an idea how GPDNS resolvers factor in

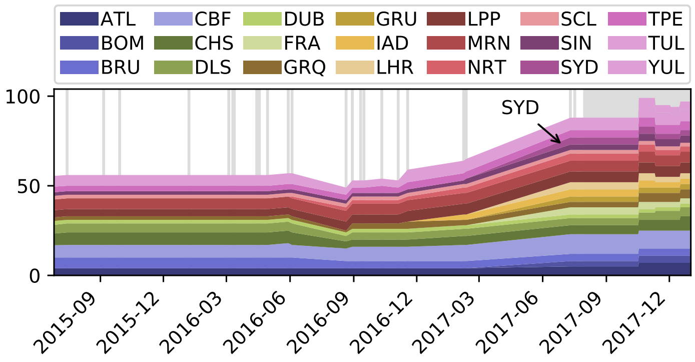
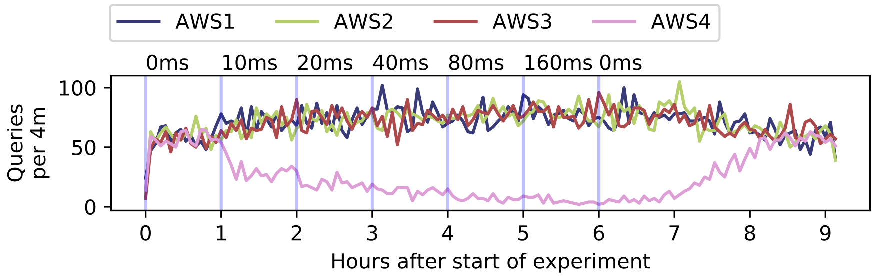


Figure 3.2: Number of Google Prefixes associated with each of the GPDNS PoPs. The gray background indicates where we have data.



.

Figure 3.3: Division of queries from Google to our 4 name servers. Every vertical line indicates a change in latency of AWS4 (e.g. +10ms, +20ms).

RTT when selecting an authoritative name server. Therefore, we conducted an experiment in which we set up four authoritative name servers for a single domain, each with a different public IPv4 address, but hosted on a single machine. This ensures uniform performance characteristics from an external viewpoint. We then measured the distribution of queries by GPDNS over several hours, where we artificially increase the RTT for one of the name servers every hour.

Figure 3.3 shows that a server which has an increased latency compared to the others, receives fewer queries. The ratio appears to be constant given a certain RTT distribution. In other words, as long as the latency remains constant, so does the distribution of queries over the authoritative name servers.

Based on this experiment, it is clear that by counting queries to our single vantage point, we cannot make claims about the total number of queries from GPDNS for domains for which our vantage point is authoritative. Since, how-

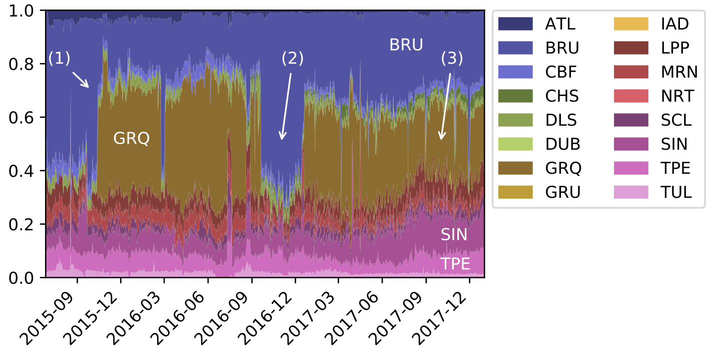


Figure 3.4: Ratio of DNS queries per GPDNS PoP seen at our authoritative servers

ever, the distribution of queries appears directly linked to RTT, and given that our vantage point is well-connected (a single hop away from major Internet Exchange Points (IXPs), including AMS-IX and LINX), we can measure trends in traffic coming from GPDNS over time.

3.3 Results

3.3.1 Query distribution

The front-end IP address of GPDNS is `8.8.8.8`, however, as the service is anycasted, this means that an end-user can potentially reach any of the PoPs, as determined by Border Gateway Protocol (BGP) routing [48]. In this section, we look at the actual distribution of queries over the PoPs that are available. Figure 3.4 shows the relative distribution of the traffic over PoPs. The three letter acronyms in the legend indicate IATA airport codes. Since the authoritative DNS server where we captured the traffic is located in the Netherlands, and is authoritative for mostly Dutch domain names, we expect the PoPs near or in The Netherlands to handle most of the load.

Prior to October 2015 most traffic was handled in the BRU PoP (Brussels, Belgium). Then, when GRQ (Groningen, The Netherlands) was brought online, marked by (1) in the plot, there was a major shift. The traffic to the BRU PoP was significantly reduced at the same time, with all the other PoPs showing a reasonably constant amount of traffic. This is likely due to the fact that the majority of users in The Netherlands have a shorter path to GRQ than to BRU. In the period marked by (2), the situation temporarily reverted to its previous state as, for an unknown reason, the GRQ PoP was deactivated. We see that, as with the previous change, the amount of traffic handled by BRU PoP increases significantly.

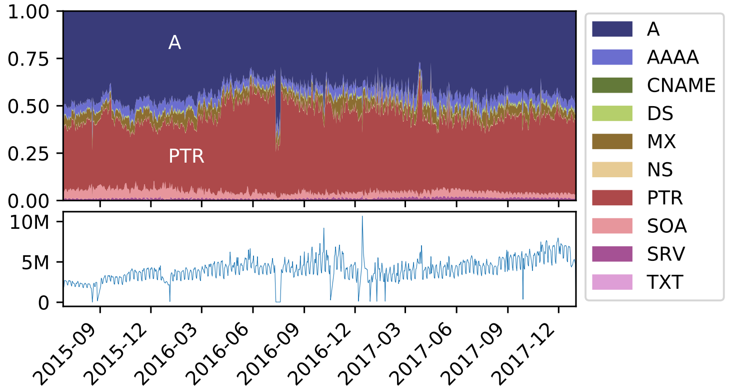


Figure 3.5: Ratio of DNS query types and absolute number of queries per day.

After the GRQ PoP was re-enabled, in the period marked by (3), the distribution of traffic is largely stable with no significant changes in almost a year, other than a slow increase in the share of traffic from PoPs in Asia (TPE and SIN).

Figure 3.5 shows the distribution of the top-10 query types. The fraction of PTR records is surprisingly high, almost equivalent to the fraction of A queries. This can be explained by the fact that the authoritative name server on which we collected queries, is also responsible for over 2,100 reverse DNS zones (including many at the /16 level in the IPv4 hierarchy). We suspect that most of these PTR queries are sent by mail servers, and examine this in more detail in Section 3.3.5.

Lastly, we see no significant increase in AAAA queries (IPv6) over the full measurement period, indicating a surprising lack of uptake of IPv6 among clients of GPDNS.

3.3.2 Out-of-country answers

Earlier work [49] showed that for public DNS services such as GPDNS, the distance between a resolver and a client varies greatly. This can lead to performance penalties if the anycast PoP serving the client is geographically remote from the client. Another question to ask in this context is: *are clients in a certain country always served by a PoP in that country?* This is especially relevant in an age of ubiquitous surveillance by intelligence services, because if traffic is routed through or to another country, this exposes that traffic to potential prying eyes. With an anycasted service such as GPDNS, one might expect that if there is a PoP available in country X, while making a request from that same country, that queries are answered from that PoP. However, depending on various parameters influencing BGP routing this is not necessarily the case.

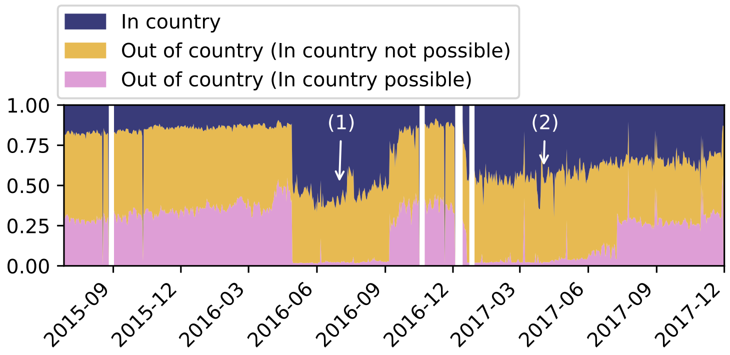


Figure 3.6: Ratio of out of country answers

As mentioned before, we map client prefixes to countries by using the IP2Location [140] database. Figure 3.6 shows the fraction of queries answered from outside the country of the end user, while a resolver inside was available. This type of out-of-country answers occur for approximately 30% of queries. We observe two deviations from the overall trend, marked by (1) and (2), from late April 2016 to early September 2016 and from December 2016 to July 2017 respectively.

In Table 3.2 we show the top 5 queries that are answered out of country, while an in-country resolver was available. Queries are grouped by resolver country and client country. We compiled this top 5 over five time periods, A through E, each representing a month of data, either before or after the beginning or end of one of the deviation periods (1) and (2), as labeled at the top of the table.

In period A, before the start of event (1), the top 5 are all answered from the US, while at the same time the Brussels datacenter is clearly active (see Figure 3.4). Then, in period B, during event (1), the total number of OOC queries drops dramatically, and the ones that do still occur are significantly closer in terms of geographic distance. The relative amount of OOC queries returns to its previous level in period C, between events (1) and (2), although the distribution has changed significantly, arguably for the better (i.e., less geographical distance between resolver and client).

The changes that occur at the event marked as 2 appear less dramatic, the number of clients who receive answers from a resolver in Belgium while located in Great Britain does increase significantly at that time, according to Table 3.2. The fact that these countries are relatively close to each other means that the performance impact is limited, although there is still a privacy impact. The situation for clients located in the Netherlands improved, as the number of queries served from Belgium decreased.

	A – before (1) 1 month before 2016-04-29			**B** – during (1) 1 month after 2016-04-29			**C** – after (1), before (2) 1 month after 2016-09-07		
#	R	C	Count	R	C	Count	R	C	Count
1	US	NL	55 258 986	NL	BE	825 902	BE	NL	33 545 772
2	US	SG	1 633 927	BE	IE	733 032	US	NL	14 007 550
3	US	BE	895 542	BE	NL	394 837	JP	US	966 666
4	US	IE	709 332	TW	SG	184 409	BE	IE	922 468
5	US	TW	520 368	TW	US	173 502	TW	JP	822 037

	D – during (2) 1 month before 2017-07-10			**E** – after (2) 1 month after 2017-07-10		
#	R	C	Count	R	C	Count
1	BE	NL	8 246 080	BE	GB	18 155 553
2	US	SG	2 179 035	BE	DE	4 011 011
3	BE	IE	1 035 833	US	BR	3 092 432
4	BE	GB	613 909	SG	IN	2 573 892
5	TW	JP	476 647	BE	NL	2 509 740

Table 3.2: Queries answered out of country, while an in-country resolver exists. R = Resolver country, C = Client country

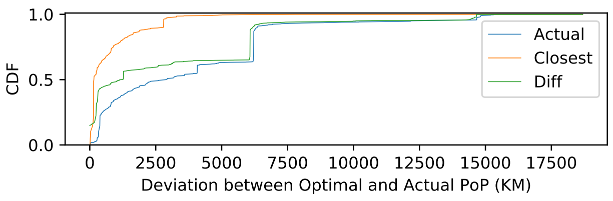


Figure 3.7: Deviation between optimal and actual PoP on 2016-04-29

3.3.3 Physical distance between end-user and PoP

While traffic being routed in or out of the user's country is particularly relevant for privacy, in terms of performance the physical distance between the end-user and the PoP is a better indicator of network distance. Consider that an end-user might reside close to a country's border, and as such a PoP in the neighboring country is potentially closer.

In this section we investigate the physical distance between the end-user, the PoP that that end-user *was served by*, and the closest PoP that that end-user *could have been served by*. We calculate the distance between the end user and each possible PoP, using the Haversine formula. This formula calculates the great circle distance between two coordinates, and it works under the assumption that the earth is a perfectly round sphere. We subtract the distance between the end-user and the used PoP and between the end-user and the PoP

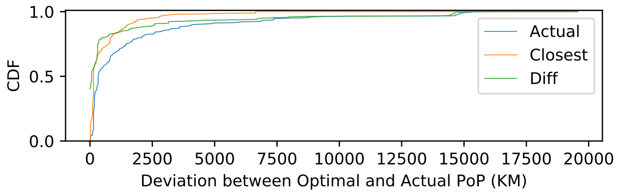


Figure 3.8: Deviation between optimal and actual PoP on 2017-07-10

that had the shortest distance. We refer to this value as the deviation between optimal and actual PoP.

Considering Table 3.2, we expect the Cumulative Distribution Functions (CDFs) of the deviation between the actual and optimal PoP on 2016-04-29 and on 2017-07-10 to show a significant difference, considering that the distance between The Netherlands and the United States is much larger than between Belgium and the Netherlands. As shown in Figure 3.7 and Figure 3.8, this is indeed the case.

To show the development of the deviation between optimal and actual PoP, we plotted this deviation as a colour plot. Figure 3.9 shows the result; in essence, each vertical line in the plot can be viewed as if looking down on a CDF as plotted in e.g. Figure 3.7 or Figure 3.8. Because the maximum deviation can be quite large, we also include a zoomed in version of this plot, which provides a view of the first 1000 km.

Interestingly, these graphs do not (completely) align with those in the previous analyses (e.g. Figure 3.6). The reason is that queries being answered in-country or out-of-country does not necessarily involve a big change in distance. Likewise, the GRQ PoP going down saw most clients moving to BRU, which is a relatively short distance. The large blocks marked by (1) and (2) are caused by a small amount of *heavy hitters*, doing reverse lookups, originating in The Netherlands, now reaching the IAD PoP in Washington DC.

3.3.4 Events leading to Google DNS adoption

There are various reasons why an end-user might switch from their ISP's DNS resolvers to GPDNS, such as performance, security (in the form of DNSSEC) or resilience. In this section we take a closer look at an event that led to a drastic increase in the use of GPDNS for a particular ISP.

The example we analyze in this section involves Ziggo, one of the largest ISPs in The Netherlands. Around August 2015 the DNS servers of this ISP suffered a Distributed Denial-of-Service (DDoS) attack, causing serious service disruption for its customers. Major national news services (*e.g.,* [143]) reported that configuring a third-party DNS service could help users. We asked ourselves:

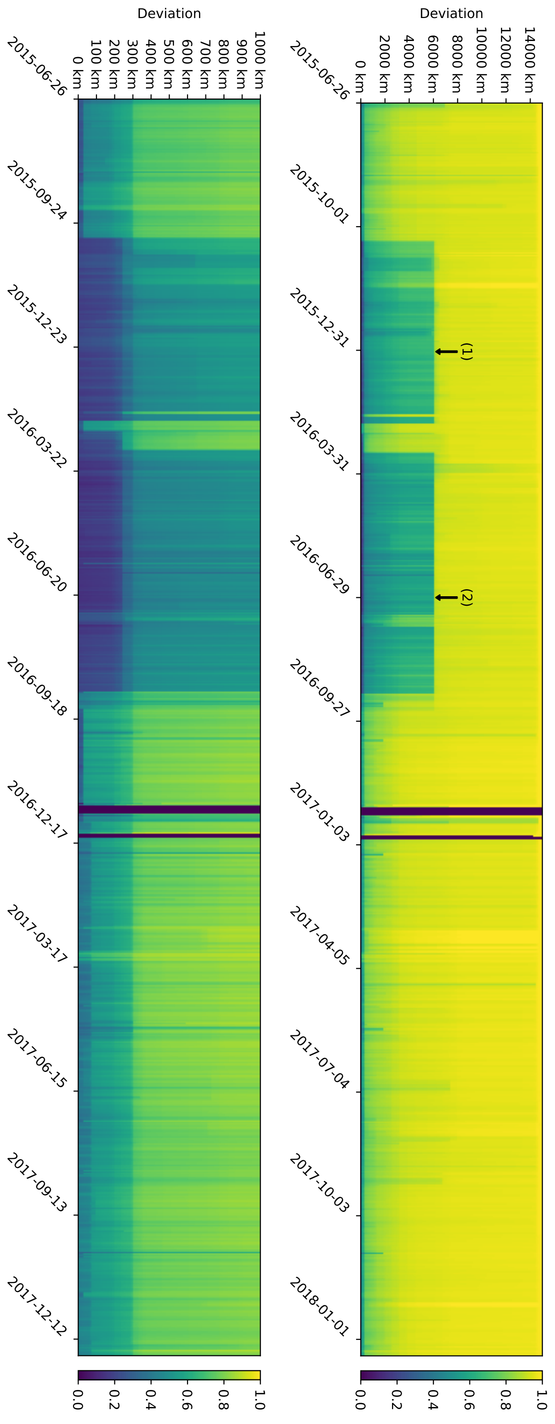
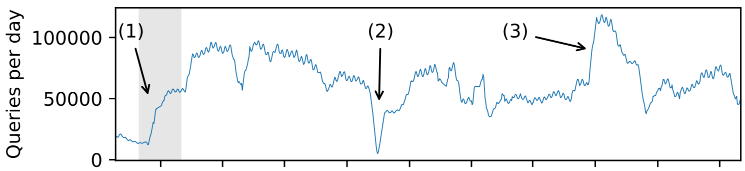
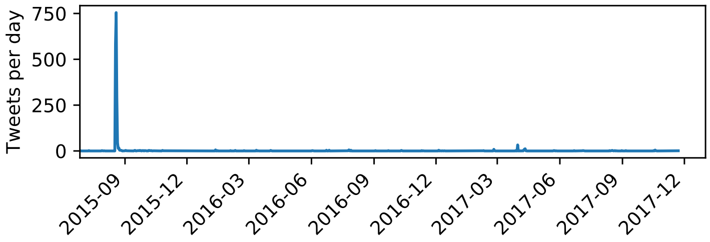


Figure 3.9: CDF of Deviation between Optimal and Actual PoP



(a) Queries from Ziggo's AS, moving average over -5 and +5 days



(b) Tweets mentioning DNS and Ziggo

Figure 3.10: Ziggo (AS9143)

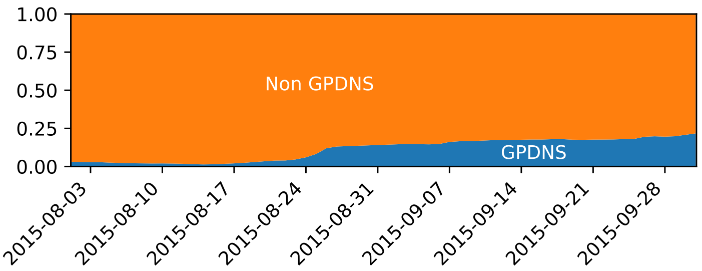


Figure 3.11: Total queries (including Non-Google) from Ziggo (AS9143)

does an attack on a major provider and subsequent media coverage suggesting the use of e.g., *GPDNS lead to increased adoption of GPDNS?*

As it turns out, an attack does lead to increased adoption. In Figure 3.10a we show the number of queries that originate from Ziggo's AS (9143) per day, using a moving average of -5 to +5 days. The uptake around the date of the attack can be clearly seen, marked by (1). The event indicated by (2) indicates a gap in our data collection, while (3) is caused by a single /24 temporarily issuing tens of thousands of requests. Figure 3.10b shows the number of tweets that used the words DNS and Ziggo over the entire measurement period. The tweets were collected using Twitter's web Application Programming Interface (API). The clear spike coincides with the DDoS attack, and marks the beginning of the increase in GPDNS use.

Another takeaway from Figure 3.10a is that uptake remains high, even after the attack has passed and Ziggo's DNS servers return to normal operation. Interestingly, while DNS is a fairly technical concept, in case of a major outage such as this people will switch away from their ISP's DNS servers, and more importantly: never switch back. How dramatic this effect really is, is illustrated by Figure 3.11. This graph (which zooms in on the two-month period around the attack) shows what fraction of queries to our vantage point arrives directly from Ziggo's AS, and what fraction arrives through Google.

3.3.5 SMTP, Google, and EDNS0 Client Subnet

As we hinted at in Section 3.3.1, the distribution of query types shows a large percentage of `PTR` queries. Pointer (`PTR`) records are used to define reverse DNS names for IP addresses. For example, given IP address `10.0.0.1` there might be a `PTR` entry for `1.0.0.10.in-addr.arpa` which points to a hostname, for example `my.host.com`. For a complete configuration there should then also be an `A` record for `my.host.com` that resolves to `10.0.0.1`. Using this methodology an IP address can be converted to its corresponding hostname and vice versa.

Reverse DNS (rDNS) is commonly used by mail servers to authenticate a sending host. Upon an incoming connection, an Simple Mail Transfer Protocol (SMTP) server typically performs a lookup of the reverse hostname of the connecting IP. If a hostname is returned, it will subsequently attempt to resolve this hostname back to the corresponding IP address. In typical configurations SMTP servers may not accept e-mail from other SMTP servers if they do not have a correctly configured reverse hostname [50].

The fact that we see large numbers of `PTR` queries coming via Google, suggests that there may be SMTP servers that have configured GPDNS as their resolver. This is a potential privacy issue, as an SMTP server then discloses to Google which servers are connecting to it to deliver e-mail. Even worse, though, GPDNS may in turn disclose this information to authoritative name servers through the ECS extension. While intuitively one might think this disclosure is not a serious issue, consider that these queries are not only sent to the authoritative name server for a domain or rDNS zone, but also to the name servers of their parent domain. Concretely, this disclosure exposes information about e-mail traffic to Top-Level Domain (TLD) and Root operators, as well as reverse DNS operators higher up the DNS hierarchy.

Figure 3.12 illustrates this scenario. In step (1) a sending SMTP server A connects to a receiving SMTP server B. In step (2), B then performs a reverse DNS lookup for the IP address of A via resolver C (which could be GPDNS). Resolver C resolves the actual IP address to a hostname by contacting the authoritative nameserver D in step (3). Once the reverse hostname is known usually the reverse hostname is again resolved to an IP address, following a similar pattern as before, in steps (4) and (5). If GPDNS is used, it may include ECS information in the queries to name servers D and E (if they support ECS), exposing information about which host is sending mail to this mail server to the authoritative name server operators. This becomes worse if GPDNS does not have sufficient information in its cache to contact D and E directly, and first needs to perform a full DNS recursion, hitting servers further up the DNS hierarchy (TLD, root, ...). If these also support ECS, information also leaks to these parties. Table 3.3 describes for different scenarios what information is leaked.

While we cannot be certain that it is in fact an SMTP daemon that performs the lookups as opposed to, for example, the firewall, this makes no difference to the privacy risks.

In order to verify this scenario in practice, we first extracted the subnets responsible for the bulk (80%) of the `PTR` queries, resulting in approximately 2,000 /24 subnets. We then used a standard scanning tool (nmap) to find systems which had port 25 (most likely SMTP) open. The scan found that a little over 50% of the subnets contain at least one system listening on port 25, for a total of roughly 15k systems.

We connect to each of the systems that have port 25 open, with a timeout of 3 seconds. We immediately transmit our identity in the client initiation phase of the SMTP session. We then read data for 6 seconds, checking for SMTP status codes as specified in [51], or until we receive a `250` message, indicating that our `HELO` message has been accepted. To determine if the SMTP daemon uses GPDNS to lookup the reverse hostname of our connecting system, we monitor the incoming DNS queries on system D in Figure 3.12. If we see an incoming DNS query for our domain between the time of connecting and the time of disconnecting we assume that this DNS query is a result of our connection.

Table 3.4 summarizes our results. Of the approximately 10k SMTP servers that we found that transmitted a valid status (a 3 digit number at the beginning of a line), we saw an incoming DNS query from roughly two thirds, and half of those came through GPDNS. We repeated the experiment without sending an initial `HELO` message from our side, with similar results.

For comparison, we also scanned the top 2 000 /24s responsible for `MX` queries. In contrast to `PTR` queries, these are likely to originate from "sending"

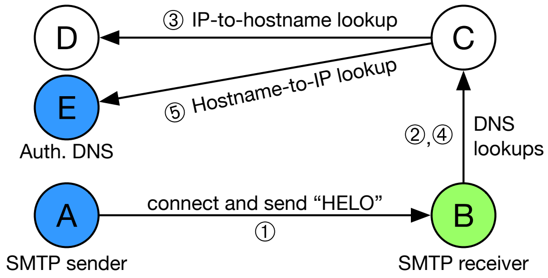


Figure 3.12: SMTP DNS Lookup process

	Google DNS at Sender	
Operator of service below can see	About sender	About receiver
Google DNS*	IP (Query source)	IP (DNS record)
Authoritative DNS for Reverse**	n/a	n/a
Intermediate DNS**	n/a	n/a
Authoritative DNS for Forward**	/24 prefix (ECS)	IP (DNS record)
Intermediate DNS**	/24 prefix (ECS)	IP (DNS record)
	Google DNS at Receiver	
Operator of service below can see	About sender	About receiver
Google DNS*	IP (DNS record)	IP (Query source)
Authoritative DNS for Reverse**	IP (DNS record)	/24 prefix (ECS)
Intermediate DNS**	IP (DNS record)	/24 prefix (ECS)
Authoritative DNS for Forward**	IP (DNS record)	/24 prefix (ECS)
Intermediate DNS**	IP (DNS record)	/24 prefix (ECS)

* Unless already in local cache (subject to TTL),
** Unless already cached by Google (subject to TTL)

Table 3.3: Information leak schema

	IPs (`PTR`)	IPs (`MX`)
Connectable	15 374	42 693
Valid SMTP Status code	9 681	32 391
DNS query in timeframe	6 503	20 107
From Google AS	3 188	11 208
Total unique	**14 204**	
Total DNS queries received	11 076	27 723

Table 3.4: Results of connecting to each of the SMTP servers

SMTP servers. We find approximately 43k systems with this scan. Similar to our previous results, approximately two thirds of these 43k systems perform a DNS query on connection, and half do this via GPDNS.

Summarizing, SMTP servers that (indirectly) use GPDNS as a resolver are common. Worryingly, we find 14,204 SMTP servers that, upon connection, leak our IP address or our prefix to GPDNS and any DNS servers that are hit during recursion.

There is a continuous effort to improve privacy in the DNS. A lot of attention is given to securing the connection between the end-user and the public resolver, such as DNS-over-TLS and DNS-over-HTTPS. However, one standard in particular focuses on reducing the amount of information leaked through query names from the public resolver to the authoritative name servers, namely QNAME Minimization (*qmin*), which is defined in RFC7816 [52]. In Section 3.4 we investigate this standard, and assess whether it has achieved traction in the Internet at large.

	Qmail v1.06	Sendmail v8.15.2	Postfix v3.1.9	Exim v4.89
1. Performs reverse lookup on connecting IP	Yes	Yes	No	Yes
2. Performs forward lookup on result of 1	No	Yes	No	Yes
3. Performs lookup on domain of from email address	No	Yes	No	No

Table 3.5: SMTP Daemons

3.3.6 Verification of Lookup Behaviour of SMTP daemons

To verify that SMTP daemons do indeed perform DNS lookups as part of their default configuration we performed a lab experiment testing four common open source SMTP daemons. These are Qmail, Sendmail, Postfix and Exim, which appear to be in wide spread use across the Internet, although real statistics of their market share are lacking.

We setup a Docker environment with a container for each of the daemons, and an additional container running the BIND DNS daemon. BIND is configured with a reverse `PTR` record for the IP address that we will use to connect to each of the SMTP daemons, as well as a forward record for the hostname.

The containers in which the SMTP daemons run are configured to resolve their DNS queries via the BIND daemon. They also run `tcpdump` to capture any DNS queries going to or coming from this container. Once configured we attempt to send a single e-mail through the SMTP daemon, to a local user (i.e. `root@dns-smtp-${daemon}.localhost`). The resulting DNS lookups are listed in Table 3.5.

Notably, all daemons but Postfix perform a DNS lookup in the standard configuration (as provided by the Debian maintainers). Sendmail and Exim both also perform a forward lookup, and the former also performs a lookup on the domain in the “From” address in the message. We note that Postfix can also trivially be configured to perform these DNS lookups, for example by setting the *reject_ unknown_ client_ hostname* or one of the variations thereof.

3.4 DNS Privacy: QName minimization

In the previous section we found that exposure of privacy sensitive information due to the combination of SMTP and ECS is common place. In this section we take a step back from the main topic of this chapter and investigate a privacy enhancing technique called *qmin* that partially counteracts this problem.

For a more detailed description of *qmin* refer to Section 2.3.3, In the standard DNS resolution process, described in Section 2.3, the recursive resolver, unaware of zone cuts in which different parts of the domain are under control of different authorities, sends the full *qname* to each of the authoritative name servers in the chain of responsible name servers.

Since the first two (the root and TLD) name servers in the recursion are very unlikely to be authoritative for the requested *qname*, this particular aspect causes unnecessary exposure of potentially private information [53], as we have shown in the previous sections. Exposing the *qname* of a website that is illegal in some countries to more parties than necessary might put the querying end-user at serious risk. A solution for this issue is proposed in RFC7816 [52], which introduces query name minimization (*qmin*), preventing recursive resolvers from sending the full *qname* until the authoritative name server for that *qname* is reached.

End-users typically do not run a recursive resolver, but instead depend on others, such as their ISP, to enable this privacy-preserving feature. From a user's perspective, *qmin* is difficult to detect except by querying a domain that is specifically setup to show whether or not it is enabled, making it hard to judge adoption.

In this section we study the adoption and performance of *qmin*. Specifically, we will quantify the adoption of *qmin* over time, both with active measurements from the end-user perspective, and with passive measurements from the authoritative name server perspective, at the root and a TLD. We will also investigate three resolver implementations of *qmin* and provide insight into its impact on performance and result quality.

3.4.1 Active Internet-Wide Measurements

We conduct active Internet-wide measurements to assess the adoption of *qmin* using two methods. First, we use RIPE Atlas probes to query a domain under our control. Second, we query open resolvers for the same domain. RIPE Atlas is a global measurement network with over 10,000 small devices called probes, and 370 larger probes, called anchors. In this section, we measure *qmin* adoption over time, classify the various *qmin* implementations in use, and shed light on *qmin* use by open resolvers.

Resolver Adoption over Time: We detect *qmin* support by relying on the fact that a non-*qmin* resolver will miss any delegation that happens in one of the labels before the terminal label. So, if we delegate to a different name server, with a different record for the terminal label in one of the labels *before* the terminal label, *qmin* resolvers will find a different answer than non-*qmin* resolvers.

We scheduled a RIPE Atlas measurement for all probes to perform a lookup with all the probe's resolvers for *"a.b.qnamemin-test.domain.example"* with type `TXT` [144], repeating every hour. Each probe uses its own list of resolvers, typically obtained via Dynamic Host Configuration Protocol (DHCP), and assumed typical for the network that hosts the probe.

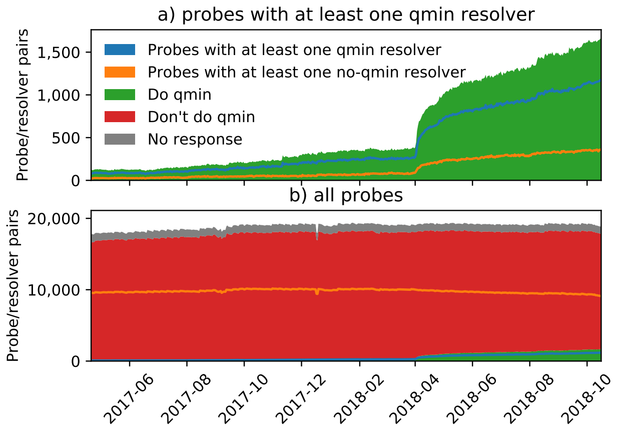


Figure 3.13: Adoption over Time

A non-*qmin* resolver will send a query for the full qname to the authoritative name server for "qnamemin-test.domain.example", and will end up with a `TXT` reply containing the text: "`qmin NOT enabled`." A *qmin* resolver will send a query for just the second-to-last label, "b.qnamemin-test.domain.example", to the authoritative name server for "qnamemin-test.domain.example". For this minimized query, it will receive a delegation to a different name server, which will return a `TXT` record containing the text: "`qmin enabled`."

This measurement runs since April 2017, and allows us to see the long term adoption of *qmin*. Figure 3.13b shows the overall adoption of *qmin* as seen from all RIPE Atlas probes. We count both probes and probe/resolver combinations, as a significant number of probes uses multiple resolvers. Adoption grew from 0.7% (116 of 17,663) of probe/resolver pairs in April 2017 to 8.8% (1,662 of 18,885) in October 2018. Also in April 2017, 0.9% (82 of 9,611) of RIPE Atlas probes had at least one *qmin* resolver, growing to 11.7% (1,175 of 10,020) in October 2018.

In Figure 3.13a only probe/resolver pairs supporting *qmin* are shown. We see a steep rise of *qmin* resolvers in April 2018. Figure 3.13a also shows probes that have at least one *qmin* resolver as well as at least one resolver that does not do *qmin*. It is noteworthy that at the last measurement (October 15, 2018) at least 31% of probes that have a *qmin* resolver, also have at least one non-*qmin* resolver.

Alongside the *qmin* measurement, we run measurements that return the IP address of the resolver as seen from an authoritative name server [145]–[147]. By identifying the Autonomous System Numbers (ASNs) associated with the IP addresses seen at the authoritative name server we gain insight in the organizations providing the *qmin* resolvers. From this we learn that

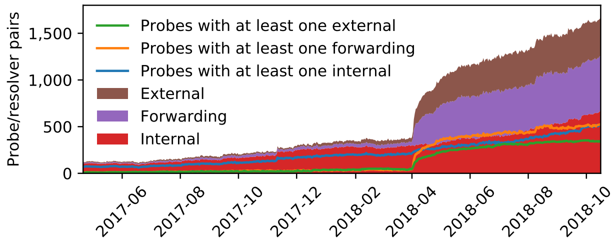


Figure 3.14: Internal, Forwarding and External resolvers supporting *qmin*

the adoption of Cloudflare (`1.1.1.1`) is responsible for the fast rise of *qmin* resolvers in April 2018.

We also found some public resolvers, such as GPDNS, that in some cases appear to support *qmin* according to our test, but in fact do not. This is likely caused by a *qmin*-enabled *forwarding* resolver, which forwards to, in Google's case, `8.8.8.8`. Additionally, the non-*qmin* resolver successively caches the authoritative for the second-last label and will appear to support *qmin* for the Time to Live (TTL) of the delegation (10 seconds in our test). We have developed an improved test without these issues in the course of this research, but this corrected test did not yet exist during scheduling of the RIPE Atlas measurement in April 2017.

The improved test, "a.b.*random-element*.domain.example. `TXT`", uses a random pattern as the third-last label which is uniquely chosen for each query, preventing other measurement queries to find a cached delegation for the second-last label. This improved test is used in measuring the adoption by open resolvers in "Adoption by Open Resolvers" in this section, removing false positives from that measurement.

We argue that this flaw had little impact on our results, as *i)*, RIPE Atlas measurements are spread out over an hour, whereas our test record has a small TTL, reducing this risk and *ii)* the overall trend over time is still indicative.

The ASNs seen at the authoritative were further used to classify resolvers in three categories: **1)** *Internal* resolvers have the same ASN for the probe and the observed resolver IP, **2)** *External* resolvers for which the ASN of the resolver IP configured on the probe matches the ASN for the IP observed on the authoritative, but differs from the ASN in which the probe resides, **3)** *Forwarding* resolvers, for which the ASN seen on the authoritative differs from both the ASN associated with the resolver IP configured on the probe and the ASN the probe resides in.

Figure 3.14 shows that both *External* and *Forwarding* probe/resolver pairs supporting *qmin* are on the rise, which is mainly due to adoption of the Cloudflare resolver in April 2018. We can also see that *qmin* support is steadily growing with *Internal* resolvers, which do not include the larger public resolvers.

Looking more closely at the *Internal* resolvers we have identified, we see that several ISPs started supporting *qmin* over the past 1.5 years. Most notably *Versatel Deutschland GmbH* started supporting *qmin* on November 9th, 2017; *Init Seven AG* on August 2nd, 2017; *OVH Systems* on February 1st, 2018; and *M-Net Telekommunikations GmbH, Germany* on May 1st, 2018. Note that these do not necessarily cause a visible change in Figure 3.14.

Adoption by Open Resolvers: Aside from resolvers that can be reached from inside networks, such as those offered by ISPs, there are also a large number of open resolvers on the Internet. These can range from unsecured corporate DNS resolvers, to large scale public DNS services, such as those run by Google, OpenDNS, Quad9 and Cloudflare.

Rapid7 provides a list of servers that are responsive on User Datagram Protocol (UDP) port 53, which are typically DNS servers. We query each such server using the method outlined in “Resolver adoption over Time”. The list contains a total of 8M IPv4 addresses, we receive a response from 64% of these. Of those responding, 32% respond with a NOERROR reply, of which only 72% (≈1.2M) provide a correct reply.

Of those 1.2M, only 19 717 (1.6%) resolvers support *qmin*. On the authoritative side, we only observe 110k unique source IPs, which suggests that many of the queried resolvers are in fact forwarders. Of the resolvers that implement *qmin*, 10 338 send queries from a Cloudflare IP, 2 147 from an OVH IP, and 1 616 from a TV Cabo Angola IP address. This shows that most *qmin*-supporting open resolvers simply forward to larger public DNS resolvers that implement *qmin*.

For *qmin*-enabled resolvers, we compare the ASN of the IP we send our query to with the ASN of the IP seen at the authoritative name server for that same query. We find 11.5k resolvers that resolve externally, and 8.2k resolvers that resolve internally.

The key takeaway is that many open resolvers on the Internet actually forward to centralized public DNS services. Thus, efforts to drive adoption of *qmin* should focus on large public DNS providers (e.g. Google, which does not support *qmin* yet).

3.4.2 Passive Measurements at Authoritative Name Servers

As *qmin* limits the visible information of a query at authoritative name servers, adoption of *qmin* likely changes the query profile of resolvers as observed on the authoritative side. We measure the impact and adoption of *qmin* with query data collected at the authoritative name servers of the Country Code Top-Level Domain (ccTLD) *.nl* and of K-Root.

Name servers of *.nl* are authoritative for the delegation of 5.8 million domain names. If they receive queries for a *.nl* domain name with 2 or more labels then they almost always (except for DS records) respond with a set of name servers that are actually responsible for the queried domain name. Thus, a query for

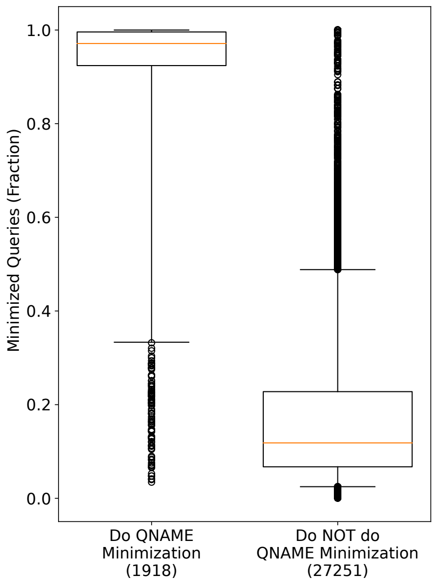


Figure 3.15: Minimized queries to *.nl*. Whiskers at 0.05 and 0.95.

the NS record of a second level domain name is sufficient for the *.nl* name servers to answer the query. Similarly, the root servers are authoritative for the 1.5k TLDs as of October 9, 2018, and a query for just the TLD is sufficient in most cases.

We cannot be certain whether resolvers send minimized queries to the authoritative name servers, but we can count the queries that follow the expected patterns if resolvers were to send minimized queries. For the rest of this section, and following the observations made in Section 3.4.1, we count queries as minimized if the query contains only 2 labels (at *.nl*) or 1 label (at K-Root). With increasing *qmin* adoption, we expect to see an increase in queries that follow these criteria.

Identifying qmin: First, we measure how query patterns seen at the authoritative name servers differ when resolvers implement *qmin*. We use the list of open resolvers from "Adoption by Open Resolvers" in Section 3.4.1 of which we know whether they have *qmin* enabled. Then, we count how many queries these resolvers send to the authoritative name servers of *.nl* for names with just two labels on 2018-10-11. In total, we observe 1 918 resolvers that *do* and 27 251 resolvers that *do not* support *qmin*.

In Figure 3.15 we see that *qmin*-enabled resolvers send a median of 97% of queries classified as minimized, whereas resolvers that have not enabled this feature send only 12% of their queries classified as minimized. This confirms that *qmin* has an observable impact at authoritative name servers.

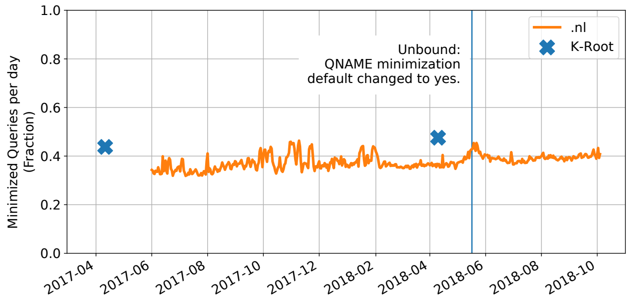


Figure 3.16: Share of minimized queries sent to *.nl* and K-Root

Resolver Adoption over Time: Based on the results of Section 3.4.1 we expect a visible impact from increasing adoption of *qmin* at authoritative name servers. To verify this expectation we count how many queries overall are sent for 2nd level domain names and TLDs respectively. We analyze *.nl* data collected from 2017-06-01 to 2018-09-30 at 2 of the 4 authoritative name servers [54] and rely on the Day in the Life of the Internet (DITL) data sets of K-Root on 2017-04-11 and 2018-04-10 collected by DNS-OARC [148]. We observe more than 400B queries from 2017-06-01 to 2018-09-30 at *.nl* and 12B queries on the two days of the DITL data sets. Figure 3.16 shows the fraction of minimized queries.

In the beginning of our measurement, roughly 33% of the queries to *.nl* where minimized. A year later, at least 40% of queries were minimized. A peak around May 2018 correlates with the date on which Unbound enabled *qmin* by default. This peak, however, is followed by a steep decline shortly after, which means we cannot confirm if Unbound enabling *qmin* by default caused this peak.

At K-Root we also observe an increase from 44% to 48% in queries for domain names with only one label. Note that query patterns at the root may strongly vary from one day to another and that many queries are sent to non-existing domain names which can influence our results [55].

3.4.3 Controlled Experiments: Impact on Resolver Performance and Result Quality

As *qmin* is deployed at the recursive resolver, we explore how it impacts the *performance* and the *result quality* of such a recursive resolver. We compare three popular *qmin*-enabled resolvers in their most recent version: Unbound 1.8.0, Knot 3.0.0, and BIND 9.13.3. We use all three resolvers with their default options, only adjusting to an equal cache size of 4GB and turning DNSSEC

	Unbound 1.8.0			Knot 3.0.0	Bind 13.3.2		
qmin mode	off	relaxed	strict	relaxed	off	relaxed	strict
# packets	5.70M	6.82M	6.71M	5.94M	5.07M	6.39M	5.84M
Errors	12.6%	12.6%	15.9%	13.5%	16.6%	17.1%	21.6%

Table 3.6: Performance and result quality across *qmin* modes and resolvers. Results are mean (μ) across all runs, with all standard deviations $\sigma < 2\%\mu$.

validation off[2]. We cycle through all configurable *qmin* behaviors for Unbound and BIND; Knot has *relaxed qmin* hardcoded. As target domains, we use the Cisco Umbrella Top 1M [149] list as a sample of popular domain names, and aggregate all domains names for a 2-week period to avoid daily fluctuations and weekly patterns [56], resulting in 1.56M domain names. To even out caching effects, we sort our target domain names in 4 different random orders. We conduct several iterations of these measurements from October 1 through October 15, 2018, starting each measurement with an empty cache. We report means from all measurement runs, and find little variation in all numbers, typically one standard deviation σ is smaller than 2% of the mean μ. Table 3.6 gives an overview of our results.

Performance: *qmin* shows a clear impact on the number of packets sent to resolve our 1.56M domains. For Unbound, the 5.7M packets without *qmin* require 6.82M (relaxed) and 6.71M (strict) packets with *qmin*, a 17–19% increase. For BIND, the increase is 15–26%. It is to be expected that the strict mode requires fewer packets, as it will give up on receiving an error, whereas relaxed modes continue through `SERVFAIL` or `NXDOMAIN` error codes. This increase in packet count is not offset by smaller packets, across resolvers we see average packet sizes only decrease by 5% or less with *qmin* enabled.

This confirms that *qmin* in its current form does come with a performance penalty of up to 26%. We argue that the full cache in a production resolver will soften that overhead, since many queries will be served from cache, and thus not initiate a *qmin* chain. Please note that a comparison of packet counts between different resolvers implicitly compares many other details such as caching strategies, which is why comparison between resolvers should be conducted very carefully.

Result quality: Another critical aspect of resolver performance is the result quality: Will a resolver be able to work through numerous edge cases and misconfigurations to deliver a response, or will it hang up on certain errors? To answer this question, we compare the amount of errors (NXDOMAIN or SERVFAIL) in our resolution results between different resolver and *qmin* approaches. Across resolvers, we see a significantly higher share of errors with

[2]We turn DNSSEC validation off to achieve comparable behavior (validating DNSSEC requires more queries to be sent); we also note that the combination of *qmin* and DNSSEC may induce further complexities beyond the scope of this work.

strict *qmin* enabled. For example, the 3.3% increase for Unbound translates to ≈50k domains, a significant share of these popular DNS domain names. The difference in resolvers corresponds to our observations on resolver behavior: As reported in Section 2.3.3, a portion of authoritative name servers fails to respond to NS queries. As Unbound uses type A queries to discover zone boundaries, and Knot and BIND use NS queries (as suggested by RFC 7816), higher error rates are expected for Knot and BIND. The surprisingly high baseline of non-resolving domains of 12-16% is a characteristic of the Umbrella Top 1M list recently discussed in [56].

These findings show that *qmin* comes with two drawbacks: Packets and bytes transferred increase, and, depending on the detailed algorithm, also a significant share of popular DNS names fails to resolve.

3.4.4 Discussion

This section covered *qmin* from various angles: we performed **1)** active and passive measurements in the Internet that confirm from both the client and authoritative server side that *qmin* adoption is rising, and **2)** controlled experiments that confirm that *qmin* can have negative performance and result quality implications We also explored the various problems and workarounds that have been deployed, and want to conclude and discuss further aspects:

***qmin* is complex:** Like many DNS mechanisms, *qmin* sounds simple, but broken deployments make it difficult to implement without collateral damage. Resolvers' iterations towards a relaxed *qmin* algorithm reflect this, and important take-aways are: *(i)* Using NS queries to detect zone cuts results in a considerable number of failures; using A queries instead seems reasonable. *(ii)* responding to SERVFAIL/NXDOMAIN by sending the full name (*i.e.,* disabling *qmin* for this query) is currently a necessity to avoid significant error rates.

***qmin* can be a security risk:** Having a resolver step through many iterations for a name with an excessive number of labels is a Denial-of-Service (DoS) attack vector. All implementations we encountered mitigate this. Unbound jumps over labels to decrease the number of queries to some maximum, considerably saving on query count. Knot's and BIND's approaches go further: Knot stops *qmin* if it encounters a label that has not been delegated (except for some exceptions, such as *.co.uk*). BIND has both a limit on the maximum number of labels (default 9), in addition to having a maximum number of undelegated labels (default 3). We consider these approaches good, as they mitigate security risks while still providing *qmin* privacy against the top levels in the DNS hierarchy.

***qmin* can impact resolver performance and result quality:** Currently, *qmin* comes with a 15%+ performance penalty, and unless implemented very carefully, will also impair result quality. Please note that, as *qmin* queries are sent sequentially, the measured increase in query volume will correlate to latency.

Recommendations: Based on the insights collected in this section, we offer the following recommendations: First, despite its performance and quality caveats, *qmin* improves privacy and should be universally deployed. Second, *qmin* deployment must be conducted carefully: We recommend an algorithm that combines Unbound's and BIND's algorithms, *i.e.,* conducts fallback upon error, replaces NS (and other) query types by A queries, and stops *qmin* after a configurable number of labels. Third, over time, heuristics may be added to alleviate certain cases where *qmin* will unlikely add privacy. For example, DANE-TLSA labels such as `_443._tcp` could be exempt from *qmin*.

The currently still rather low *qmin* adoption already causes a significant positive effect for query privacy at both Root and TLD authoritative name servers. While there are legitimate performance, result quality, and security concerns, we already see resolver implementers tackle these, and are confident that these negative implications will be further reduced, assisted by the quantitative evidence and tangible recommendations in this study. We fully expect more and more DNS operators to enable *qmin* to further improve privacy of end-users on the Internet.

3.5 Related Work

The idea for the ECS extension was first tabled in 2011, supported by a coalition of parties promoting a "Faster Internet". Partners in this project include both CDN operators and operators of large public DNS resolvers.

Otto et al. were the first to study ECS. In their paper [42], they study the impact of the use of public DNS resolvers on web CDN performance, and highlight the performance improvement ECS could offer in this context. Furthermore, they study the first preliminary uptake of ECS by CDN operators. In follow-up work [43], Sánchez et al. study the performance improvement of CDN web delivery if ECS is used via GPDNS. Streibelt et al. use ECS to study the infrastructure of CDNs that support ECS. Their paper [44] shows how ECS can be used to provide insight into CDN server deployments, and CDN server-to-client mappings. They highlight that they can perform such mappings from a single vantage point, by inserting arbitrary prefixes in the scope field of an ECS query, provided that CDN operators do not limit from which sources they are willing to respond to ECS-enabled DNS queries. Additionally, Streibelt et al. also study aggregation by ECS-enabled CDNs, showing different strategies where some CDNs return ECS responses with larger scopes (i.e. returning an ECS response for an IPv4 `/16` prefix when a smaller `/24` prefix is specified in the DNS query), whereas others respond with narrower scopes than asked for, going as low as a `/32`. The authors speculate that CDNs that follow the latter practice essentially want to force DNS resolvers to cache the result only for a single client.

Calder et al. use ECS for a longitudinal study of Google's service delivery CDN [45]. Their work shows that using ECS they can create a complete mapping of this CDN and can uncover dramatic growth of this CDN over a ten-month period. Fan et al. also use ECS to study Google's CDN, but rather than focusing on the CDN infrastructure itself, they study changes in client prefix to CDN front-end mappings over time [46].

Chen et al. study the impact of ECS deployment from inside the Akamai CDN [49]. The introduction of ECS at Akamai resulted in a 30% improvement in startup time for connections to the CDN, at the cost of an eight-fold increase in the number of DNS queries to their name servers. Chen et al. are the first to show the RTT performance penalty incurred by users due to their DNS requests getting routed to geographically remote public resolvers. In this work we significantly extend on this by using longitudinal data covering 2.5 years, showing, e.g., changes in out-of-country query handling over time.

A common denominator of the related work to date has been that it exclusively focused on using ECS to study service delivery by CDNs or to study how ECS can improve this service delivery. In contrast, in this work, we leverage ECS to study the behavior of and use of a large public DNS provider.

Kintis et al. discuss some of the privacy implications of ECS [57]. Their focus is the privacy risks imposed by on-path attackers between the public DNS resolver and authoritative name servers on the Internet. They observe how an on-path attacker can perform selective surveillance on clients of public DNS resolvers. In addition to this, they also show how an attacker can selectively poison a public DNS resolver's cache for specific clients using ECS. In this work, we extend this by showing new privacy risks where the use of public DNS resolvers by SMTP servers to perform DNS resolution leaks information about the IP addresses and domains of hosts sending e-mail to these servers.

In Section 3.3.3 we use a similar methodology as Li et al. [58], who studied the performance of IP anycast at Internet scale, to determine the deviation in distance between the optimal GPDNS PoP and the one that traffic is actually routed to.

Hardaker et al. [27] showed that root servers receive a considerable amount of privacy-sensitive query names, and propose using local instances of root servers to alleviate this issue. Imana et al. [59] study this aspect from a broader perspective, covering all name servers above the recursive resolver, and report similar privacy issues.

Schmitt et al. [60] propose Oblivious DNS, an obfuscation method introducing an additional intermediate resolver between recursive resolver and authoritative name servers. Oblivious DNS prevents the additional resolver from learning the user's IP address and the recursive resolver from learning the query name.

Recent work [61] has also shown that *qmin* increases the number of queries per lookup, increasing the load on authoritative name servers. They provide a technique called `NXDOMAIN` optimization that reduces the number of queries in case the resolver encounters an `NXDOMAIN`. We reinforce this work by providing longitudinal measurements, showing various implementations of *qmin* algorithms and by quantifying the increase in queries per resolver implementation.

3.6 Discussion

The ECS Request for Comments (RFC) is already very critical about the potential privacy implications of the use of ECS. In the privacy note included *before* the actual protocol description, the authors write:

> *"If we were just beginning to design this mechanism, and not documenting existing protocol, it is unlikely that we would have done things exactly this way."*

They go on further to suggest that ECS, if supported in DNS resolver software, should be turned off by default. Clearly, as the research in this paper illustrates, Google has not followed this suggestion. On the contrary, Google actively attempts to detect support for ECS on authoritative name servers, and will include the option in queries when an authoritative name server responds with an ECS option to their probing queries. This makes it trivial to solicit ECS information from Google.

This raises the question what other operators of large public resolvers do. Table 3.7 shows, as an example, five well-known and commonly used public resolver operators. As the table shows, there are wildly varying policies. Two operators do not support ECS at all, in both cases explicitly citing privacy as a reason not to support ECS. The three others all support ECS, but all have a different policy on when they send ECS to authoritative name servers. Of the three, the OpenDNS policy of active whitelisting probably makes the best tradeoff between protecting the privacy of users versus maintaining an efficient service for users towards CDN providers that support ECS and ask to be whitelisted. Arguably, in terms of privacy, VeriSign's policy is the least favourable of the three, as *any* authoritative name server operator is likely to receive ECS information (and thus able to record this), regardless of whether their deployed name server software actually supports ECS.

Table 3.7 also shows the scope prefix each operator includes in queries. All three operators that send ECS information include the smallest prefix recommended in the RFC for IPv4, but there are significant differences in the prefix size included for IPv6, from relatively large (OpenDNS), to a very narrow prefix that may identify individual end users (VeriSign).

The last column in Table 3.7 shows whether those operators that send ECS information allow clients control over whether or not ECS information should be sent to authoritative name servers for their queries. The RFC allows clients to exert control over the ECS behaviour of resolvers by including an ECS

		Scope		
Operator	**ECS policy**	**IPv4**	**IPv6**	**Respects Scope-Zero?**
Google	Auto-detect	/24	/56	Yes
OpenDNS	Whitelist	/24	/48	No
Quad9	No ECS	n/a	n/a	n/a
Cloudflare	No ECS	n/a	n/a	n/a
VeriSign	Always	/24	/64	Yes

Table 3.7: EDNS Client Subnet policies of public resolver operators

option in the query they send to a resolver. If a client includes an ECS option with the scope prefix length set to zero, then resolvers should interpret this as an indication that the client does not want information about their IP to be included in queries to authoritative name servers. As Table 3.7 shows, both Google and VeriSign respect such requests from clients, whereas OpenDNS does not. It is debatable whether this option really gives clients control, as the stub resolvers included in all main stream operating systems do not support this behaviour.

Finally, we reflect on what operators of DNS resolvers should do with respect to ECS. Many open source implementations of DNS resolver software now support ECS. Given the potential privacy impact demonstrated in this paper, and identified by others [57], we argue that the default should be not to enable ECS. Operators of large networks, or of public DNS resolvers, that do wish to enable ECS support to better serve their clients when querying CDNs should practice active whitelisting of authoritative name servers that will receive ECS information, and should also carefully consider what source prefix length to include in the ECS option. While a separate study would be required to recommend specific prefix lengths to support, e.g., correct identification of users at the country or regional level, we note that initial experiments with different prefix sizes for IPv4 against popular Geo IP databases shows that sending a /22 source prefix length allows accurate identification at the country level in up to 95% of cases.

3.7 Concluding Remarks

In this chapter, we presented the results of the first longitudinal study of a large public DNS resolver, GPDNS, using passive observations of DNS queries that contain the ECS extension. Additionally, we quantified the use of a privacy enhancing DNS technique called *qmin*.

In this chapter we show that ECS can be used to study the day-to-day operations of a public DNS resolver, in our case GPDNS. As a key result of this chapter, this allowed us to show that traffic to GPDNS is frequently routed to out-of-country PoPs for weeks at a time, even though an in-country PoP is available. This potentially exposes DNS traffic to state-level surveillance.

Additionally, we showed that it is not uncommon for a suboptimal PoP, in terms of geographical distance, to be selected, leading to additional round-trip times for DNS queries, and a possible decrease in DDoS resilience. Importantly, this leads us to conclude that even a large operator such as Google struggles to optimize their catchments, and that further research into anycast is important.

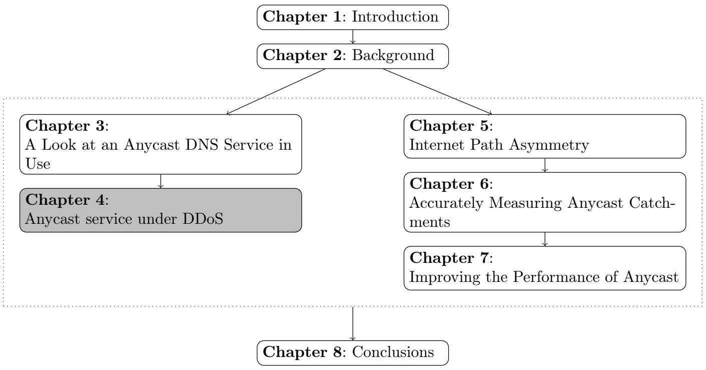
We also showed that certain events such as DDoS attacks on ISP DNS resolvers cause users to switch to GPDNS *en masse*, and, more importantly, once users have switched to Google they do not switch back.

There has been much debate about the privacy risks of using public DNS resolvers. The obvious argument is that the operator of such a resolver gets access to extremely privacy-sensitive information in the form of DNS queries. One aspect of privacy in the context of public DNS resolvers remains under-exposed. In order for CDNs to be able to make Geo IP-based decisions, many public resolvers use a DNS extension called ECS, which allows them to reveal part of a client's IP to the CDN. In essence, the need for ECS is an unintended side-effect of the use of public DNS resolvers. In this chapter we have also shown that the privacy enhancing technique *qmin* is slowly but steadily gaining traction in the Internet at large.

In the next chapter, we will continue looking at large scale anycast deployments. We will specifically focus on the performance of anycast while it is under stress from a DDoS attack.

CHAPTER 4

Anycast service under DDoS


Chapter 1: Introduction
Chapter 2: Background
Chapter 3: A Look at an Anycast DNS Service in Use
Chapter 4: Anycast service under DDoS
Chapter 5: Internet Path Asymmetry
Chapter 6: Accurately Measuring Anycast Catchments
Chapter 7: Improving the Performance of Anycast
Chapter 8: Conclusions

In the previous chapter we looked at a large anycast network in the course of its normal operation. However, a key benefit of using anycast is its potential for resilience against DDoS attacks. In this chapter we therefore focus on an anycast network under stress. Specifically, we focus on a DDoS event that took place in the months November and December of 2015, the target of which was the Root DNS (see Section 2.3). We analyze data gathered from various data sources, such as RIPE Atlas and RSSAC-002, to see what happens to this system while it is under attack. The Root is a unique system in that it is run by twelve different organizations, each operating their own independent instance of the Root, serving the same data, for resilience purposes. The different instances of the Root are labeled with letters (A-K). All but one of these use Anycast to provide additional resilience. In this chapter we show that different services (i.e. the letters representing different organizations), use different strategies when dealing with an attack. Some choose to absorb the traffic, while others withdraw routes in the hopes that another instance will handle the load. We also show that in some cases collateral damage occurs when a service is hosted alongside another, for example in the same data center or in the same network. The key takeaway from this chapter is that operating an Anycast network facing a DDoS attack requires careful consideration in order to minimize service down time and collateral damage. The work in this chapter was published as a research paper [3].

4.1 Introduction

Although not new, Denial-of-Service (DoS) attacks are a continuing and growing challenge for Internet services [150], [151]. In most DoS attacks the attacker overwhelms a service with large amounts of either bogus traffic or seemingly legitimate requests. Actual legitimate requests can then be lost, for example because the service's network bandwidth is exhausted, or because of limited compute resources. Once a service is overwhelmed, the service might become susceptible to extortion [152]. In the case of persistent attacks clients may be driven to other, more reliable, services. There have been cases where attacks have lasted for weeks [153]. Refer to Section 2.4 for a more detailed background on DDoS attacks.

The DNS is a common service, and the root servers (see Section 2.3.4) are a fundamental, high-profile, and publicly visible service that have been subject to DoS attacks in the past. As a public service, they are monitored [154] and strive to self-report their performance. Perhaps unique among many large services, the Root DNS service is operated by 13 different organizations, with different implementations and infrastructure. Although the internals of each implementation are not public, some details (such as the number of anycast sites) are.

To evaluate the effects of DoS attacks on real-world infrastructure, we analyze two specific events: the Root DNS events of Nov. and Dec. 2015 (see Section 4.2 for discussion and references). We investigate how the DDoS attack affected reachability and performance of the anycast deployments. In this chapter we present the first study that explores the response of real infrastructure across several levels, from specific anycast services to physical sites of those services, down to the level of individual servers. We also found that an important consequence of high load on sites are routing changes, as users "flip" from one site to another after a site becomes overloaded.

Although we consider only two specific events, we explore their effects on 13 different DNS deployments of varying size and capacity. From the considerable variation in response across these deployments we identify a set of potential responses, we have explored this in theory (Section 2.4.1) and now show this in practice (Section 4.4). Exploration of additional attacks, and of the interaction of IP anycast and site selection on other layers (for example, in Bing [62]) is future work.

The main contribution of this chapter is that it presents the *first evaluation of several IP anycast services under stress with public data.* Anycast is in wide use and while commercial operators have been subject to repeated attacks, some of which have been reported [127], [134], [152], [153], [155], [63], [156], the details of those attacks are often withheld as proprietary. We demonstrate that in large anycast instances, *site failures* can occur even if the service as a whole continues to operate. Anycast can both *absorb* attack traffic inside sites, and also *withdraw* routes to shift both good and bad traffic to other sites. We explore these policy choices in the context of a real-world attack, and show that *site flips do not necessarily help* when the new site is also overloaded, or when the shift of traffic overloads it.

In comparison to the previous chapter (Chapter 3), we focused on the effects of a large DDoS attack, rather than on the anycast deployment by itself. The challenges we have shown to exist in the real world suggest future research into anycast deployments to be important.

4.2 The Events of Nov. 30 and Dec. 1

On November 30, 2015 from 06:50 to 09:30 (UTC), then again on December 1, 2015 from 05:10 to 06:10, many of the Root DNS Letters experienced an unusual high rate of requests [127]. Traffic rates peaked at approximately 5M queries/s, which, for A-Root [63], is more than 100× the normal load. We show the query rates per day for all letters that report according to RSSAC-002 (see Section 4.3.2) in Figure 4.1.

While we characterize these events as an “attack” here, since sustained traffic of this volume is unlikely to be accidental, the actual intent of these events is unclear.

An early report by the Root Operators stated that several letters received high rates of queries for 160 minutes on Nov. 30 and 60 minutes on Dec. 1 [127]. Queries used fixed names, but source addresses were randomized. Some letters saw up to 5 million DNS queries per second, and some sites at some letters were overwhelmed by this traffic, although several letters remained reachable throughout the attack (either because they had sufficient capacity or because they were not attacked). There were no known reports of end-user visible errors, likely because top level names are extensively cached, and the DNS system is designed to retry and operate in the face of partial failure.

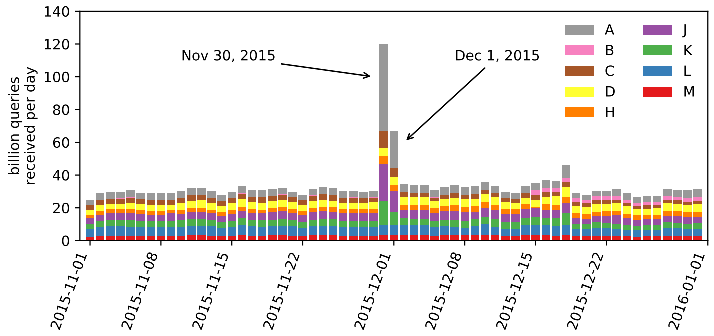


Figure 4.1: Received queries per day (in billions) for November and December 2015, highlighting the attack, based on RSSAC-002 data.

A subsequent report by Verisign, operator of A- and J-Root, provides additional details [63]. They stated that the attack was limited to IPv4 and UDP packets, and that D-, L-, and M-root were not targeted. They confirm that the queries that were part of the attack used fixed names, namely `www.336901.com` on Nov. 30 and `www.916yy.com` on Dec. 1. They reported that A and J together saw 895M different IP addresses, strongly suggesting source address spoofing, although the top 200 source addresses accounted for 68% of the queries. They reported that both A- and J-Root were attacked, with A continuing to serve all regular queries throughout, and J suffering a small amount of packet loss. They reported that Response Rate Limiting was effective [63], identifying duplicated queries to drop 60% of the responses, and filtering on the fixed names was also able to reduce outgoing traffic. They suggested the traffic was caused by a botnet.

Motivation: We do not have firm conclusions about the motivation for these events. As Wessels first observed [157], the intent is unclear. The events do not appear to be DNS amplification to affect others since the spoofed sources spread reply traffic widely. They might be a DDoS targeted at services at the fixed names listed above, but an attack on the fixed names would be much more effective if the root lookup was cached and not repeated. Possibly it was an attack on those targets that went awry due to bugs in the attack code. It may be a direct attack on the Root DNS, or even a diversion from other activity. Regardless of the intent of the event we can analyze it to further our understanding of anycast systems under stress.

4.3 Datasets

For this study we used publicly available datasets provided by RIPE, several of the Root operators, and the BGPmon project. We thank these organizations for making this data available to us and other researchers. We next describe these data sources and how we analyze it. We make the resulting datasets publicly available [158].

4.3.1 RIPE Atlas Datasets

RIPE Atlas is a measurement platform with more than 10k global devices (Atlas Probes) that provide *vantage points* (*VPs*) that conduct network measurements [64], [65]. All Atlas VPs regularly probe all Root DNS Letters. A subset of this data appears in RIPE's DNSMON dashboard evaluating the Root DNS [154]. RIPE identifies data from all VPs that probe each root letter with a distinct measurement ID [159]. Our study considers all available Atlas data (more than DNSMON reports), with new processing as described below.

RIPE's baseline measurements send a DNS CHAOS query to each Root Letter every 4 minutes. At the time of the event, A-Root was an exception and was probed only every 30 minutes, too infrequent for our analysis (Section 4.4.2) (it is now probed as frequently as the other letters). Responses to CHAOS queries are specific to root letters (after cleaning, described below) but each letter follows a pattern that can be parsed to determine the site and server that VP sees. For this report we normalize identification of roots in the format *X-APT*, where *X* is the Root Letter (A to M) and *APT* is a three-letter airport code near the site.

We focus predominantly on E- and K-Root, since they provide anycast deployments with dozens of sites. These examples illustrate the operational choices (Section 2.4.1) all anycast deployments face.

Data cleaning: We take several steps to clean RIPE data for using it in our analysis. Cleaning preserves nearly all VPs (more than 9000 of the 9363 that were active at the time), but discards data that appears incorrect or provides outliers. We discard data from VPs with Atlas firmware before version 4570. Atlas firmware is regularly updated [160], and version 4570 was released in early 2013. Out of caution, we discard measurements from earlier firmware on non-updating VPs to provide consistent (current) methods of measurement. Moreover, we also discard measurements of a few VPs where traffic to a root appears to be served by third parties. We identify hijacking in 74 VPs (less than 1%) by the combination of a CHAOS reply that does not match that letter's known patterns and unusually short RTTs (less than 7 ms), following prior work [66].

After cleaning we map all observations into a time series with ten-minute bins. In each time bin we identify, for each Root Letter, the response: either a site the VP sees, a response error code [17], or an absence of a reply after 5 seconds (the Atlas timeout). Each time bin represents 2.5 RIPE probing intervals, allowing us to synchronize RIPE measurements that otherwise occur at arbitrary phases. (When we have differing replies in one bin, we prefer sites over errors, and errors over missing replies.)

Limitations of RIPE Atlas: RIPE Atlas has known limitations: although VPs are global, their locations are heavily biased towards Europe. This bias means Europe is strongly over-represented in per-letter reachability (Section 4.4.2), but it does not influence our analysis of specific user behavior (Section 4.4.4). The largest risk uneven distribution of VPs poses is that some anycast sites may have too few VPs to provide reliable reporting. While we report on all anycast sites we observe, we only consider sites whose catchments contain a median of at least 20 VPs during the two days.

In addition, RIPE VPs query specific Root letters, so they do not represent "user" queries. (Regular user queries employ a recursive resolver that selects one or more letters to query.) We take advantage of this approach to study specific letters and sites, but it prevents us from studying Root DNS reachability as a whole (Section 4.4.2).

Finally, VPs fail independently. We focus our attention on sites typically seen by 20 or more VPs to avoid bias from individual VP failure over the two days.

4.3.2 RSSAC-002

RSSAC-002 is a specification for operationally-relevant data about the Root DNS [161]. It provides daily, per-letter query rates and distributions of query sizes.

All Root Letters have committed to provide RSSAC-002 data by 2017. At the time of the events, only five services (A, H, J, K, and L) were providing this data [133]. In addition, RSSAC-002 monitoring is a "best effort" activity that is not considered as essential as operational service, so reporting may be incomplete, particularly at times of stress.

4.3.3 BGPmon

We use BGP routing data from BGPmon [67]. BGPmon peers with dozens of routers providing full routing tables from different locations around the Internet. We use data from all available peers on both days on which the events occurred (152 peers) to evaluate route changes at anycast sites in Section 4.4.4.

4.4 Analysis of the Events

To evaluate the events we begin with overall estimates of their size, then drill down on how the events affected specific Root Letters, sites in some letters, and individual servers at those sites. We then reconsider the effects of the attack as a whole, both on Root DNS service and on other services.

4.4.1 How Big Were the Events?

We next estimate the size of the events. Understanding the size is important to gauge the level of resources available to the traffic originator. We begin with RSSAC-002 reports, but on Nov. 30, only a few letters provided this data, and as previously described (Section 4.3.2), best-effort RSSAC-002 data is incomplete. We therefore estimate an upper-bound on the event bandwidth based on inference from available data.

RSSAC-002 provides statistics over each day, so to estimate the event size we define a baseline as the mean of the seven days before the event. We then look at what changed on the two event days (A-Root had an independent attack on 2015-11-28, so we drop this data point and scale proportionally). Query sizes are reported in bins of 16 bytes. Verisign stated that the attacks were of specific query names (see Section 4.2), and RSSAC-002 reports query sizes in bins of 16 bytes, allowing us to identify attacks by unusually popular bins. Most queries fell in the the 32-to-47 B bin on Nov. 30 and the 16-to-32 B bin on Dec. 1, while response sizes are most frequently between 480 and 495 bytes in size for both events, see Figure 4.2. These sizes are for DNS payload only. We confirm total traffic size (with headers) in two ways, both by adding 40

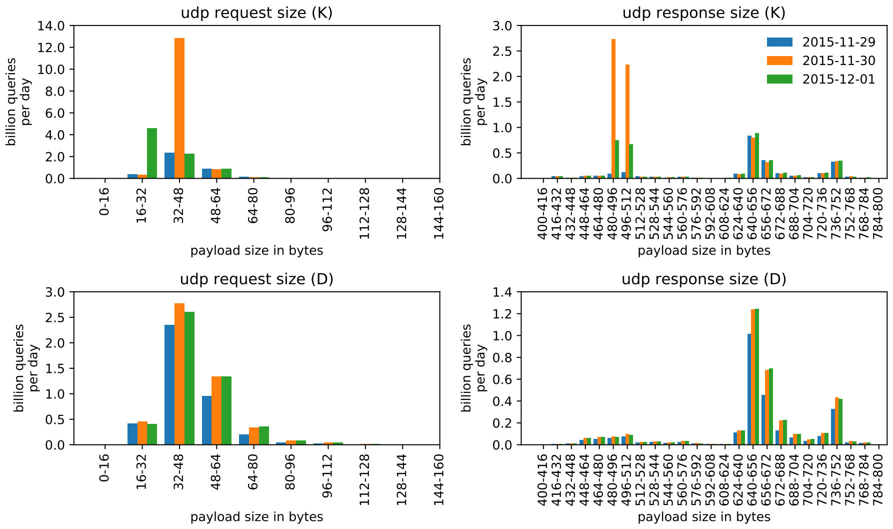


Figure 4.2: UDP Request and Response sizes for three days, for K and D root, based on RSSAC-002 data.

bytes to account for IP, UDP, and DNS headers, and by generating queries with the given attack names. We confirm full packets (payload and header) of 84 and 85 bytes for queries and 493 or 494 bytes for responses, consistent with RSSAC-002 reports.

We use these sizes to estimate incoming bitrates.

Table 4.1 gives our estimates on event traffic from the five letters reporting RSSAC-002 statistics. The baseline (right column) is only 1–10% of attack traffic (mean: 3%); we subtract the baseline from queries and responses therefore our estimations show only the *extra* (Δ) traffic caused by the events. These reported values differ greatly across letters and between queries and responses. We believe differences across letters represent measurement errors, with most letters under-measuring traffic when under attack (under-reporting is consistent with large amounts of lost queries described in Section 4.4.2). We see fewer responses than requests, likely because of Response Rate Limiting [135] which suppresses duplicate queries from the same source address [157]. We provide both a lower-bound on attacks that considers only known event traffic, and a scaled value that accounts for the six sites known to have been attacked that did not provide RSSAC-002 data at event time. This lower bound has a large underestimate because 3 of the 4 reports were known to drop event traffic, and there is an approximate 3% overestimate by including baseline queries.

We propose an upper-bound for the event size by correcting for both of these types of under-reporting. To correct, we accept that A-Root's RSSAC-002 data measured the entire event. Verisign reported [157] A-Root graphs of input

	2015-11-30 (160 min.)				
RSSAC	**Δ queries**			**Δ responses**	
reports	Mq/s	Gb/s	M IPs (ratio)	Mq/s	Gb/s
A	5.12	3.44	1,813 (340×)	3.84	15.13
H	0.23	0.15	36.14 (13.3×)	—	—
J	1.90	1.28	765.24 (280×)	1.10	4.32
K	1.07	0.72	39.23 (14.4×)	0.48	0.32
L	0.05*	0.04*	36.15 (13.3×)*	0.05*	0.19*
bounds (lower and **upper**):					
lower	8.32	5.59	–	5.42	19.77
(scaled)	(20.8)	(14.0)	–	(13.5)	(49.4)
upper	**51.22**	**34.42**	–	**38.37**	**151.31**

	2015-12-01 (60 min.)					*Baseline*	
RSSAC	**Δ queries**			**Δ responses**		**queries**	
reports	Mq/s	Gb/s	M IPs (ratio)	Mq/s	Gb/s	Mq/s	M IPs
A	5.21	3.54	1,345 (253×)	3.93	15.53	*0.04*	*5.35*
H	0.32	0.22	16.22 (6.5×)	—	—	*0.03*	*2.94*
J	2.29	1.56	355.68 (129×)	1.43	5.66	*0.05*	*2.78*
K	1.12	0.76	40.88 (15.0×)	0.28	1.09	*0.04*	*2.92*
L	0.10*	0.07*	16.22 (6.5×)*	0.09*	0.37*	*0.06*	*2.94*
bounds (lower and **upper**):							
lower	8.94	6.08	–	5.64	22.28	0.22	–
(scaled)	(22.4)	(15.2)	–	(14.1)	(55.2)	–	–
upper	**52.09**	**35.42**	–	**39.31**	**155.35**		

Table 4.1: RSSAC-002 reports for daily IPv4/UDP traffic for the two days of events, subtracted from the a 7-day mean baseline, and lower- and upper-bounds on event sizes. *L-Root was not attacked and therefore excluded from lower and upper bounds.

traffic showing about 5Mq/s at both A- and J-Root (although J's RSSAC-002 reports are much lower). They also report that 10 of 13 letters were attacked (D, L, and M were not attacked). We add the assumption that all attacked letters received equal traffic. We confirm this assumption in two ways. First, B-Root can confirm [162] that it saw offered load around 5 Mq/s, consistent with A-Root's statement. Second, we can infer event sizes by comparing accepted traffic loads in Table 4.1 with observed loss rates from Figure 4.3. That suggests H-Root should have received 1.6 Mq/s, J-Root about 2.44 Mq/s, and K-Root about 1.6 Mq/s.

Our estimates are somewhat rough, but it provides strong evidence that this was not a small attack. While 6× more than the lower-bound of directly observed traffic, this estimate reflects significant query loss that occurs during the event and in measurement systems tuned for regular operation.

If our upper-bound estimate is correct, the aggregate size of this attack across all letters is about 35–40 Gb/s. Although attacks exceeding 100 Gb/s have been demonstrated since 2012 [134], [150], such large attacks are usually performed using amplification [31] (for example, as the reply traffic of 151 Gb/s on Table 4.1). Directly sourced traffic of 35 Gb/s on the roots therefore represents a large attack.

We can also see for all letters a large increase (by a factor 6.5× to 340×) in the number of unique IPv4 addresses observed by each letter during the attacks. While this could be explained solely by the use of a botnet, initial reports have confirmed that this is caused by IP spoofing [157].

4.4.2 How Were Individual Letters Affected?

We next consider how each letter reacted to the event and measure overall Root DNS performance. Letter-specific queries from RIPE Atlas show that individual root letters suffered minimal to severe loss rates. We caution that these loss rates do not directly translate to end-user delays, since recursive resolvers cache and retry against different letters, and since users interact with specific anycast sites (Section 4.4.3).

Reachability of Specific Letters: Figure 4.3 shows the reachability for each Root Letter from RIPE Atlas. We plot D-, L-, and M-Root together because they see no visible change, consistent with reports that they were not attacked [63]. (In Section 4.4.6 we later show that a few D-root sites appear slightly affected by the event.) On these dates, Atlas probed A-Root less frequently than other letters (Section 4.3.1), so in this graph we scale A's observations to account for this difference. Because infrequent probing of A-Root makes the event dynamics impossible to discern, we omit A-Root from analysis in the rest of this chapter.

All the other letters experience different degrees of reachability problems during the reported attack intervals (Section 4.2). There is a strong correlation ($R^2 = 0.87$) between how many sites a letter has (Table 2.3) and how they were affected in terms of responsiveness, measured by the smallest number of Atlas VPs that successfully receive responses during the events (more sites$\rightarrow$more VPs receive responses). B-Root, which did not use anycast at the time of attack, suffered the most, followed by H, with two sites and primary-secondary routing. With many sites, J-Root sees some VPs lose service, but only a few. We evaluate the *causes* for service loss in Section 4.4.3, but this correlation reflects some combination of more sites providing greater aggregate capacity and isolating some users from some event traffic.

We can also evaluate overall performance for each letter by the RTT of successful queries, as shown in Figure 4.4. Note that each letter has a different baseline RTT, corresponding with the median distance from Atlas VPs to

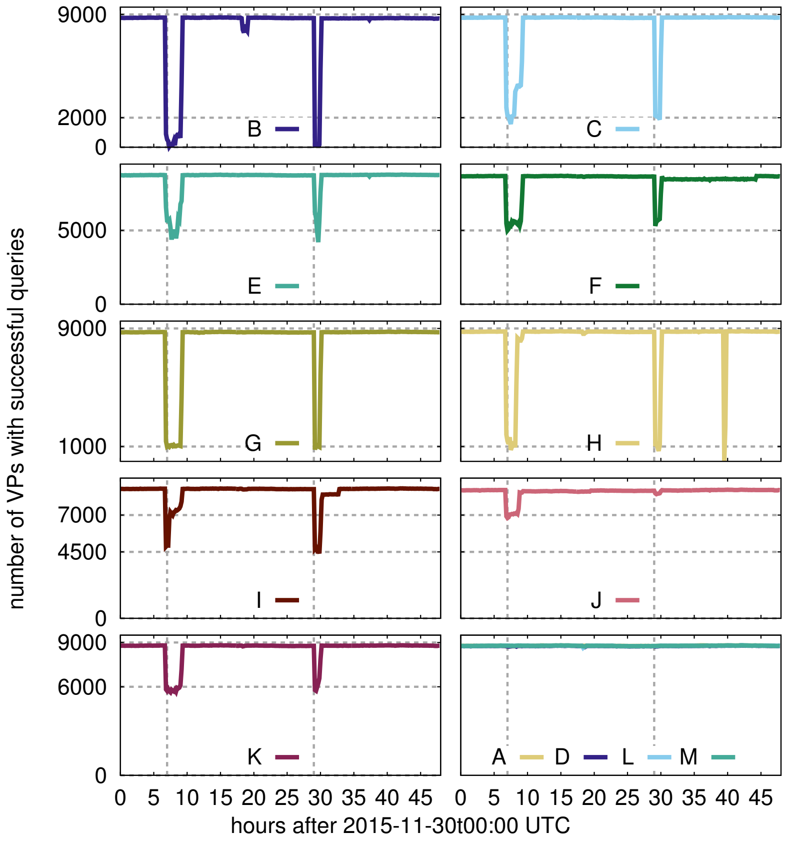


Figure 4.3: Number of VPs with successful queries (in 10-minute bins). (All plots are scaled consistently, with nearly 9000 VPs across 48 hours of observation. In all graphs, dotted lines highlight approximate event start times. Here they also show the lowest values for the dips.)

anycast sites for that letter. Although B-Root suffered the most in terms of reachability (Figure 4.3), it experienced little change in RTT when queries *were* successful. G- and H-Root, in turn, see large changes in latency. In the next section we show that anycast sites can *fail*, causing routing to shift their traffic to other locations. Thus we believe these shifts in RTT indicate route changes that shift VP traffic to more distant sites. For example, H-Root has sites on the U.S. East and West coasts (north of Baltimore, Maryland, and in San Diego, California). Most Atlas VPs are in Europe, so we infer that the primary site for H is the U.S. East coast, but when that route is withdrawn (during both events) traffic shifts to the west coast. This assumption is confirmed by H's median RTT at that time matching B-Root's RTT, since B-Root is also on

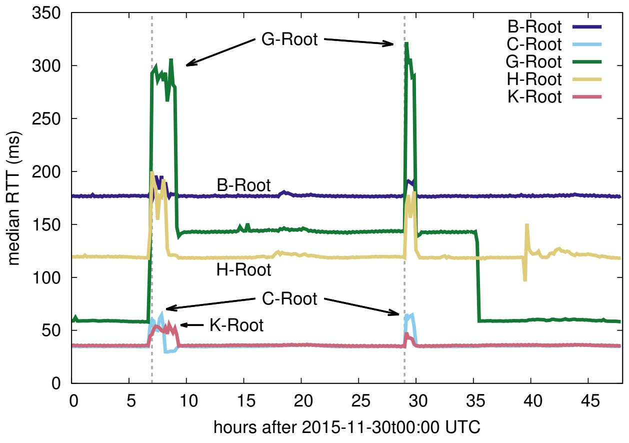
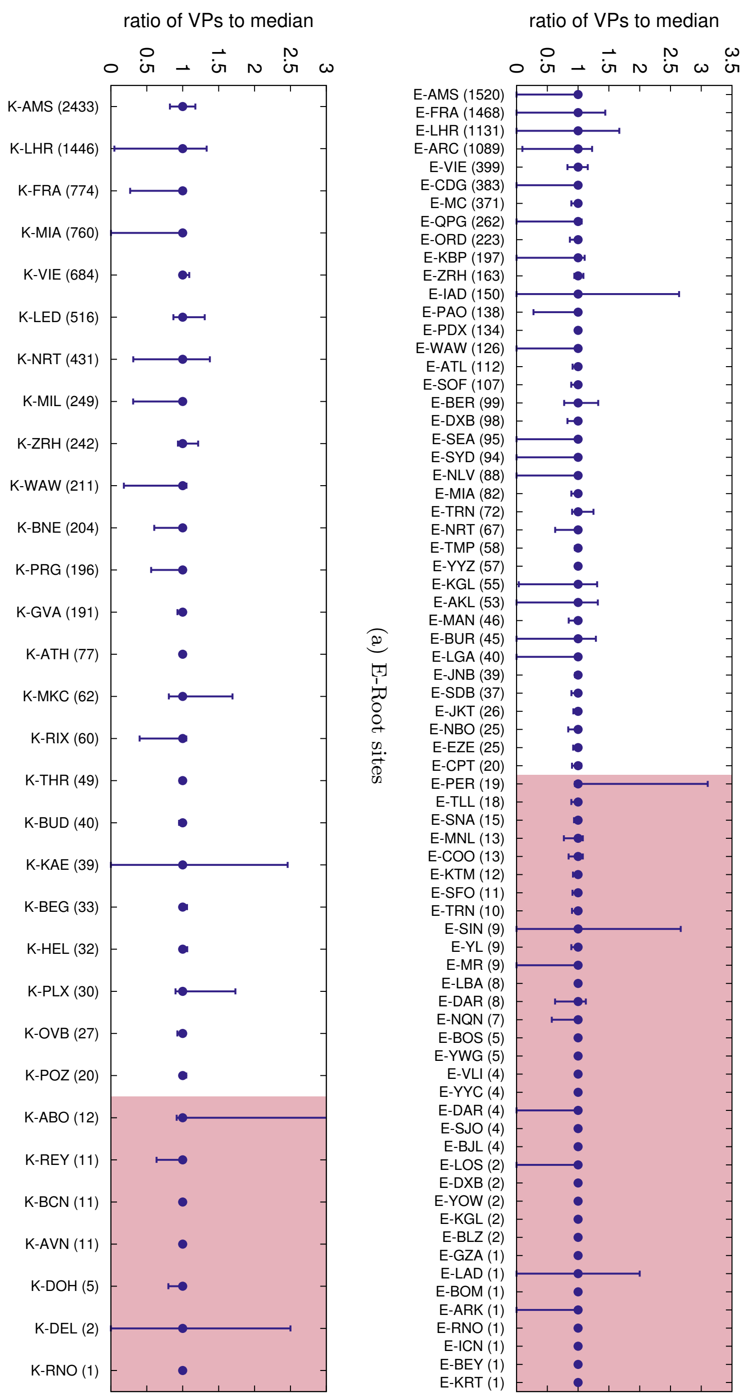


Figure 4.4: Median RTT for some letters during the attacks. Letters with no significant change (A, D, E, F, I, J, L, and M) are omitted.

the U.S. West coast. We examine site route withdrawals in more detail in Section 4.4.3.

Reachability of the Root DNS as a Whole: While we see that individual letters show degraded responsiveness under stress, the DNS protocol has several levels of redundancy, and a non-response from one letter should be met by a retry at another letter. We do not evaluate overall responsiveness of the Root DNS in this chapter, but our per-letter analysis shows some evidence of this redundancy.

L-Root was not subject to these attacks [157], yet Table 4.1 shows that L-Root experienced a significant increase in query rate during the second event, with a 1.66× increase in queries-per-second. More impressively, it sees a 6- or 13-fold increase in number of unique IPs on both event dates. We later describe “site flips”, where VPs change anycast sites (Section 4.4.4); this coarse data suggests *letter flips* also occur, as recursive resolvers switch from one letter to another, perhaps to prefer a shorter RTT [47], [68]. While not the focus of this chapter, these letter flips show the multiple levels of resilience in the Root DNS system (see also Section 2.3.4)



(a) E-Root sites

(b) K-Root sites

Figure 4.5: Minimum and maximum number of VPs, normalized to median (shown in parenthesis per individual site), for sites from E- and K-Root. Observations are grouped into 10-minute bins over two days. Sites are ordered by median number of VPs, and the red, shaded area highlights sites with fewer than 20 VPs (our threshold for stability, Section 4.3.1).

4.4.3 How Were Anycast Sites Affected?

Overall loss rates for each letter (Section 4.4.2) may suggest that query loss is uniform for all who use that letter. We next show that these loss rates are *not* uniformly seen by all users. Anycast services are composed of multiple sites (Table 2.3), and anycast operators and their hosting ISPs can design sites to withdraw routes or continue as degraded absorbers (i.e., a site that keeps announcing a BGP route, despite being overloaded) when under stress (Section 2.4.1). We next look at behavior across all sites of a given letter to identify evidence of these policies in action. **Site Reachability:** We first consider site reachability: how many Atlas VPs reach a letter's sites over the two days of observations, measured in each ten-minute bin. The median number of VPs over the observation provides a baseline of "regular" behavior, calibrating how RIPE Atlas maps to a given service. Atlas coverage is incomplete; some sites have zero or a few VPs, while others have thousands in their regular catchment. Our use of median normalizes coverage to identify trends, such as if the site gains or loses VPs. Addition of VPs to one site indicates withdrawal of routes to another site, possibly in reaction to stress. Reduction in VPs indicates that either that site withdrew some or all routes, or that it was overloaded and simply lost queries—reduction can therefore be caused by both withdrawal and absorption.

Figure 4.5 shows all sites for two letters (E- and K-Root, selected as representatives with many sites). Numbers in parenthesis show the median number of VPs at each site, while the lines show how much that site shrank or grew, in terms of VPs, over the two days, normalized to the median.

We see that sites show two responses indicating reduced capacity. Some (such as E-AMS) become completely unavailable, as shown by the minimum dropping to zero; some become nearly unavailable, such as K-LHR; K-Root confirmed unavailability of some sites [163]. Others (E-NRT, K-WAW) become partially available.

In addition, several sites show an increase above median over the period (the maximum blue value is greater than 1). Several of the well-observed K-Root sites show some increase (K-AMS, K-LHR, K-LED, K-NRT), as do many of the well-observed E-Root sites (E-FRA, E-LHR, E-ARC, E-VIE, E-IAD).

We confirm that these swings in catchments are directly correlated with the events and are not typical behavior. We repeated the analysis of Figure 4.5 over two days during the week following the events (2016-12-05 and 2016-12-06). On these "normal" days, considering sites with reasonable visibility (20 or more VPs, so medians are stable), we see *no* variation in VPs per site for K-Root, and only minor variation (mostly within 8%) for 13 sites of E-Root.

Second, Figure 4.6 shows the size of each site's catchment during the events, for E- and K-Root. Each mini-plot represents one site, with the line showing how many VPs are mapped to it relative to the site's median. The central line in each plot is the median, with the lower line 0 and the upper line 5× and 3× the median for E- and K-Root respectively. Red lines below the median indicate potential *critical* moments in which reachability dropped below the median number of VPs that can normally reach the site. Sites are sorted by median, in parentheses. From this figure we see that sites from these two letters behaved completely differently. While most sites of E-Root either see an increase or a decrease on their reachability, most sites of K-Root seem to overlook the attack. (Note that large increases observed for few sites, such as E-DXB and K-DEL, are caused by a very low median (two VPs)—any additional VP hitting these sites during the attack can cause a peak on reachability, we therefore exclude these from the figure.)

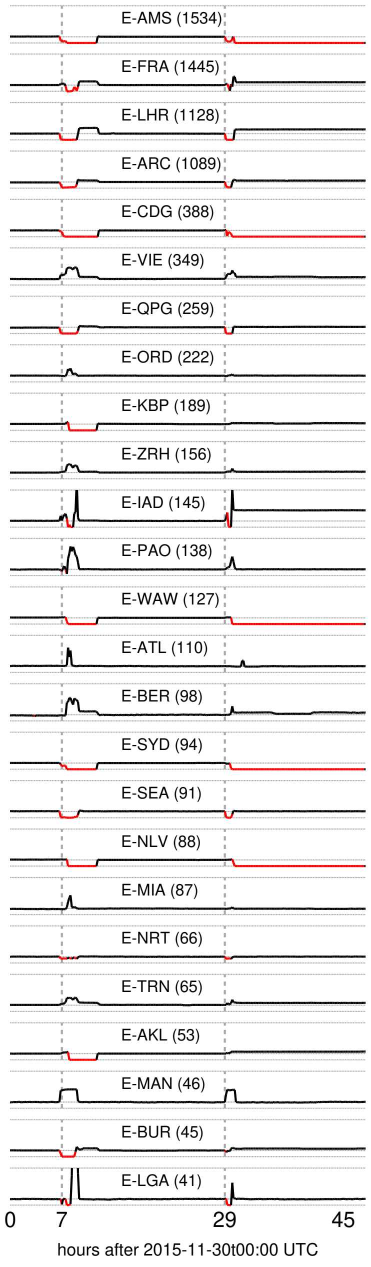
Figure 4.6 shows that five sites from E Root (E-AMS, E-CDG, E-WAW, E-SYD and E-NLV) seem to "shut down" after the attack of Dec. 1 (hour 29). These sites also had reachability strongly compromised during the first event on Nov. 30 (hour 7).

What is interesting to see for the sites of both letters in Figure 4.6 is that sites with large numbers of median VPs in their catchments showed reachability problems. An exception is K-AMS, with a large number of VPs in its catchment, which took on more traffic than usual during the whole period.

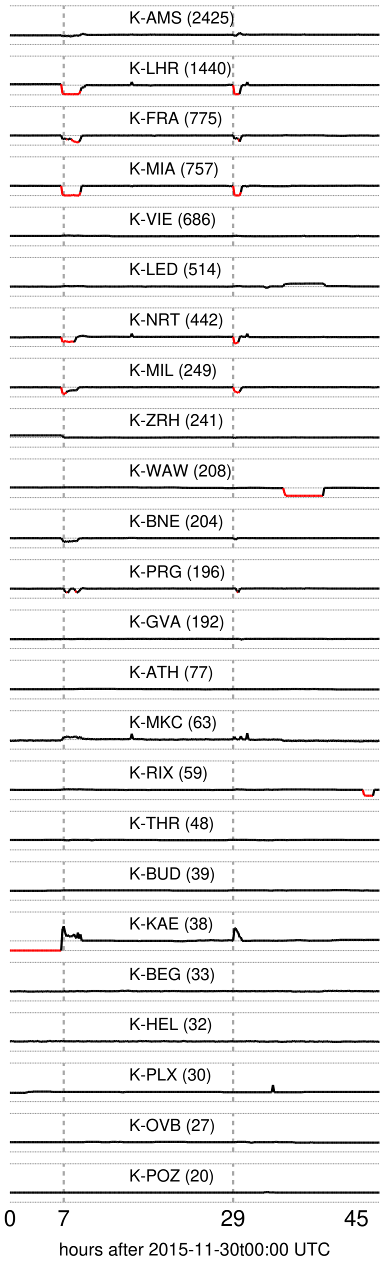
For E-Root, sites that show an increase over median suggest that some other sites are withdrawing some routes at other sites. However, that does not explain why letters show reduced overall reachability (Figure 4.3): if overloaded sites fail and traffic shifts, all queries should be answered. We next look for evidence of degraded absorption.

Site RTT Performance: To assess if sites that remain accessible are overloaded (implying they operate as degraded absorbers, i.e. sites that continue to operate despite failing to successfully serve all traffic), we next examine RTT of successful queries.

Figure 4.7 shows the median RTT for some K-Root sites that show stress during the events. Although the K-AMS site remained up and showed minimal loss, its median RTT showed a huge increase: from roughly 30 ms to 1 s on Nov. 30, and to almost 2 s on Dec. 1, strongly suggesting the site was overloaded. K-NRT shows similar behavior, with its median RTT rising from 80 ms to 1 s and 1.7 s in the two events. Overload does not always result in large latencies. B-Root (a single site) showed only modest RTT increases (Figure 4.4), since only few probes could reach it during the attack (Figure 4.3). We hypothesize that large RTT increases in site performance are the result of an overloaded link combined with large buffering at routers (industrial-scale bufferbloat [69]).



(a) E-Root sites



(b) K-Root sites

Figure 4.6: Reachability seen by VPs that received positive responses (RCODE 0) for sites of E- and K-Root. Sites with fewer than 20 VPs are excluded.

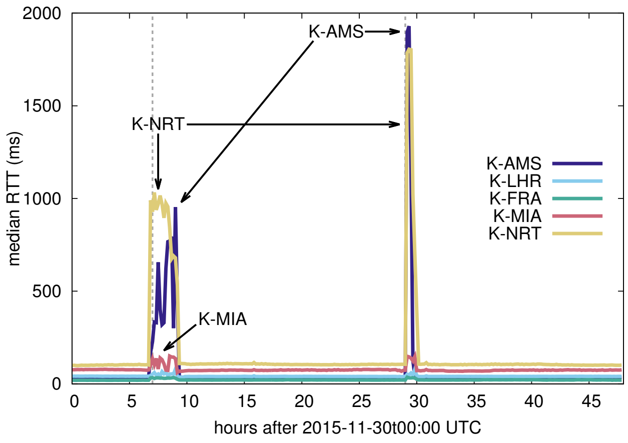


Figure 4.7: Performance for selected K-Root sites.

4.4.4 How Can Services Partially Fail?

We have shown that letters report different amounts of service degradation (Figure 4.3), and that their sites seem to follow two policies under stress (Section 4.4.3). We next look at service reachability from a *client* perspective to understand how services can partially fail, and how some clients see persistent failures.

Site Flips: Evidence of Stress:

A design goal of DNS and IP anycast is that service is provided by multiple IP addresses (DNS) and sites (anycast). Through their recursive resolvers, clients can turn to a service on another IP address (other Root Letters), and through route changes at upstream ISPs, to other anycast sites. A recursive DNS resolver will automatically retry with another name server if the first does not respond, which is intentional redundancy in the protocol and an operational best practice [47], [70]. Redundancy *inside* most letters depends on IP anycast, and the routing policies DNS service operators establish at each anycast site (withdraw or absorbing, as in Section 2.4.1).

To study a client's view of IP anycast redundancy, we look for changes in site catchments. We measure these as *site flips*: when a VP changes from its current anycast site to another. We expect each VP to have a preferred site (hopefully with a low RTT), and site flips to be rare, due to routing changes or site maintenance.

Figure 4.8 shows site flips measured in RIPE Atlas VPs, with bursts of site flips during the event periods for letters that saw event traffic. All letters see thousands of site flips during the event (note the scale of the y-axis), with E, H and K seeing many flips while C, I and J see fewer.

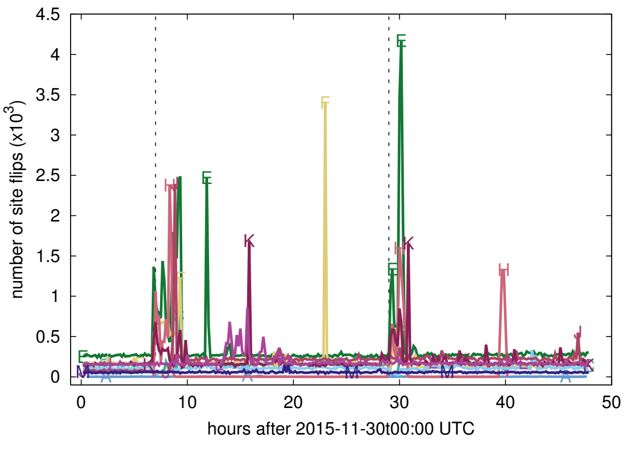


Figure 4.8: Number of site flips per Root letter.

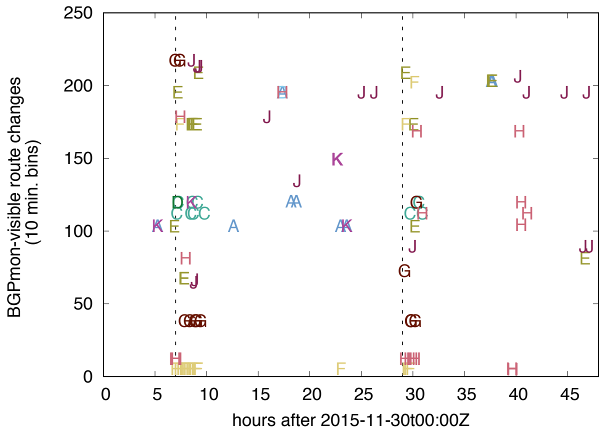


Figure 4.9: Route changes for each Root Letter (10 minute bins, seen from BGPmon route collectors).

To evaluate if these site flips are actually due to route withdrawals, we use route data from BGPmon (Section 4.3.3). These BGPmon VPs are in different locations from our RIPE Atlas VPs, so we do not expect them to see exactly the same results, but, if there were route withdrawals, we expect to see more routing activity during the events.

Figure 4.9 shows the route changes we observe across all Root Letters. With BGPmon VPs and Root anycast sites around the world, we see occasional route changes over the whole time period. With 152 VPs, a routing change near one site can often be seen at 100 or more VPs. But the *very frequent* sets of changes shown by *many* letters in the two event periods (4 to 6 hours and around 29 hours) suggests event-driven route changes for many letters (C, E, F, G, H, J, K). Route changes for K-Root do not appear at our BGP observers for the second event, and K's BGP changes are lower than we expect based on site flips. We suspect that is because our BGP vantage points are mostly U.S.-based, while we show site flips using VPs that are much more numerous in Europe.

Case Study: K-Root: We next consider K-Root as a case study to show what site flips mean in practice. K-Root's sites provide good examples of different policies under stress. We next consider VPs that start at K-LHR and K-FRA (London and Frankfurt) to see what happened to these clients during the event. We select these sites to illustrate possible design choices (Section 2.4.1) and because they lost nearly all or about half of the VPs during the event; they were more strongly affected then most K-Root sites. From Figure 4.3 we know that some clients were unsuccessful, while the maximums in Figure 4.5b show that some sites gained clients.

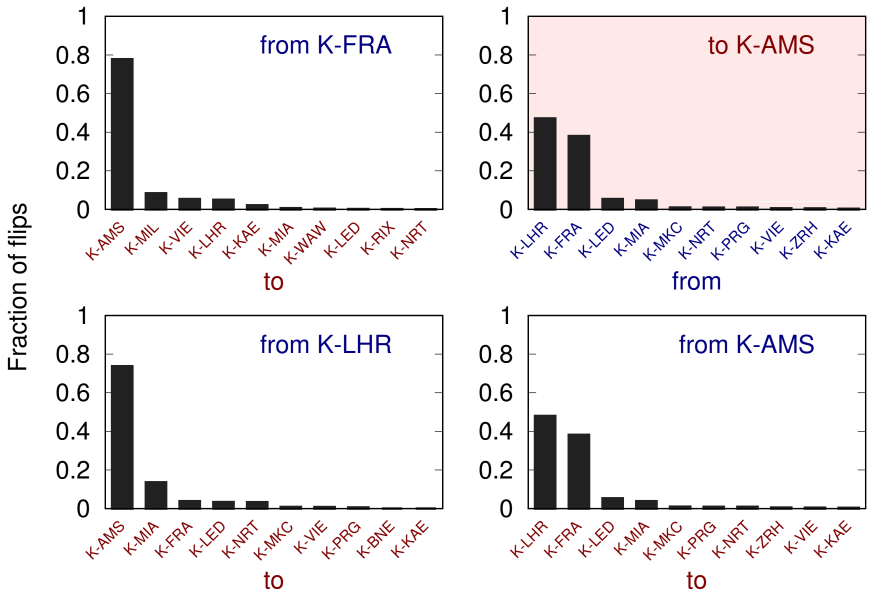
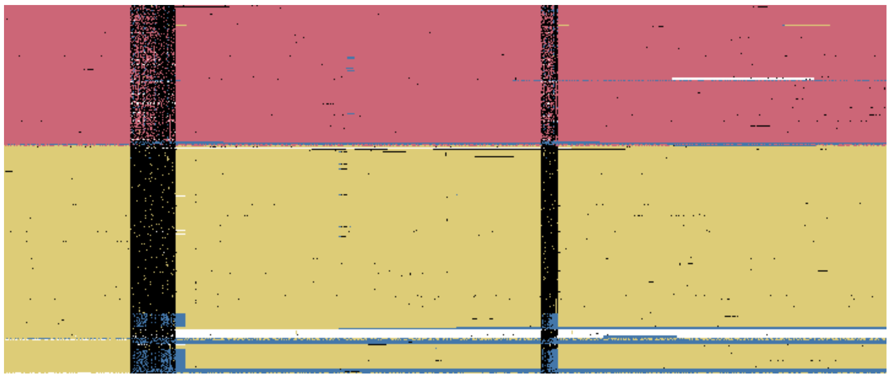


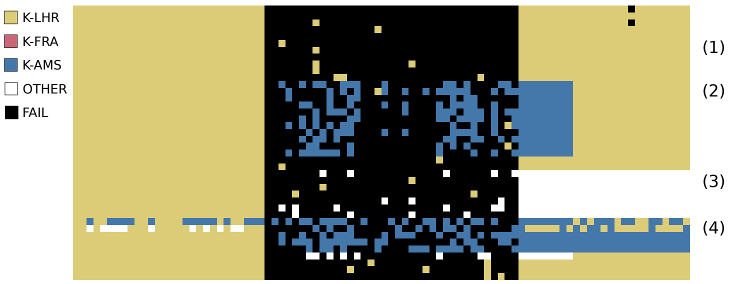
Figure 4.10: Site flips for selected K-Root instances over the two days. Two graphs on the left show where VPs went from K-FRA and K-LHR. Top right graph shows where VPs came from when switching to K-AMS. Lower right graph shows where VPs went after the event ended, from K-AMS.

Figure 4.10 shows where sites from K-LHR and K-FRA went over the measurement period—the left two graphs show that about 70-80% of all VPs that shifted traffic during the events shift to K-AMS (Amsterdam). The top right (red background) graph shows where new VPs that see K-AMS just were, confirming they mostly arrive from K-LHR and K-FRA. The bottom right graph shows that K-AMS sites also shift back to K-LHR and K-FRA as their preferred catchments after the events.

However, we still ask: *if traffic shifts to other sites and K has excess capacity, why do some VPs fail to reach K during the attack?* VP query failure must result from routing policies and implementation details (Section 2.4.1) at each site and its hosts: those policies and details can result in a site that will continue to receive traffic from its peer and operate as a degraded absorber, or that will withdraw its route and reallocate its catchment. We see evidence of both outcomes.

To demonstrate these policies at work, we must look at the actions of individual VPs. Figure 4.11 shows 300 randomly selected VPs that start at K-LHR (yellow or light gray) and K-FRA (salmon or medium gray) for 36 hours. Each pixel represents the site choice of that VP in 4-minute bins. Black


(a) A sample of 300 VPs; start 2015-11-30t00:00Z for 36 hours.



(b) A smaller sample: 40 K-LHR-preferring VPs around the first event.

Figure 4.11: A sample of 300 VPs for K-Root that start at K-LHR and K-FRA, with locations before, during, and after attacks. Time is shown horizontally, each column representing a 4 minute time period (the measurement interval), while each row represents a single probe. Dataset: RIPE Atlas.

indicates the VP got no reply, while blue (or dark gray) and white indicates selection of K-AMS or some other K-Root site.

We focus on the 40 VPs shown in Figure 4.11b and see two behaviors during the event and three after. During the event, the top 10 VPs (labeled (1)) stick to K-LHR, but only get occasional replies. They represent a degraded absorbing peering relationship; these clients seem "stuck" to the K-LHR site. The next group labeled (2) shift to K-AMS during the event and for a short period after, then return to K-LHR. However, during their visit to K-AMS only about a third of their queries are successful. This group shows that K-AMS is overloaded but up, and that these VPs are in ASes that are not bound to K-LHR. For the third group, marked (3), some stay at K-LHR during the event, while others shift to other sites, but all find other sites after the event. Finally, the group (4) shifts to K-AMS during the event and remains there afterward.

We see similar groups for the K-AMS sites in the first event and for both sites in the second event.

We believe this kind of *partial failure* represents a *success* of anycast in isolating some traffic to keep other sites functional, but this degraded absorbing policy results in some users suffering during the event due to the overload at K-LHR. While this policy successfully protects most K-Root sites during the event, it also suggests opportunities for alternate policies during attack. Rather than let sites fail or succeed, services may choose to control routing to engineer traffic to provide good service to more users. Alternatively, if attack traffic is localized, operators may choose to target routing so that only one catchment is affected—a policy particularly appropriate for attacks where all traffic originates from a single location, even if it spoofs source addresses.

4.4.5 How Were Individual Servers Affected?

Large anycast sites may operate multiple servers behind a load balancer (Figure 2.12). We now examine how the events affected individual servers within specific anycast sites. We look at two sites of K-Root, K-FRA and K-NRT as examples, selected because they show different responses to stress. These behaviors are also seen at other sites, but we do not identify or count behaviors across all sites. These examples show it is important to use measurement strategies that consider all servers at a given site.

Figure 4.12 shows a time series of servers that respond at K-FRA (top) and K-NRT (bottom) during the events. At K-FRA, we typically saw replies from each of the three servers. As the load of each event rose, replies shifted to come from only one server, with none from the other two we previously

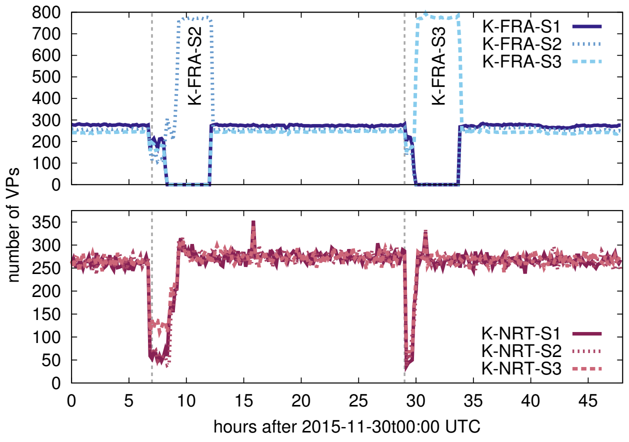


Figure 4.12: Reachability for individual servers from K-FRA (top) and K-NRT (bottom).

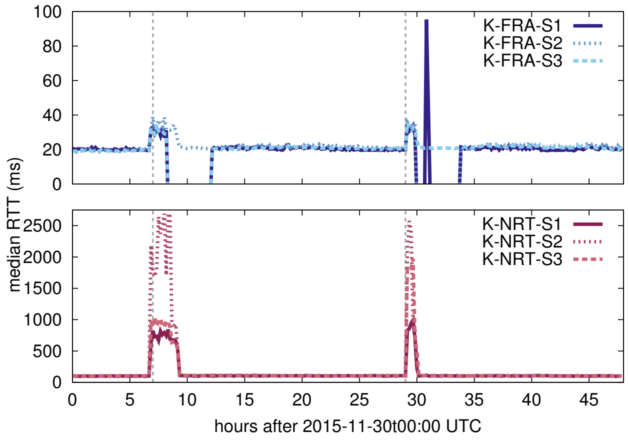


Figure 4.13: Performance for individual servers from K-FRA (top) and K-NRT (bottom).

saw replying. Which server responded was different in the two events, with K-FRA-S2 replying in the first event and -S3 in the second. We do not know if the other two servers failed, or if they were only serving attack traffic, or if traffic from these VPs was somehow isolated from attack traffic. Either way, this strategy seems to work reasonably well since Figure 4.13 shows that, after a short increase in RTT at the beginning of the attack, the median RTT for K-FRA remains stable for successful replies throughout the attack. However, K-FRA seems to be overloaded and dropping queries, as shown in Figure 4.6b and Figure 4.11b.

K-Root's Tokyo site (K-NRT) shows a different result. Figure 4.12 (bottom) shows that VPs had difficulty reaching all three servers from K-NRT during the events. This difficulty suggests that the events affected all K-NRT servers, either because load balancing was mixing our observations with attack traffic, or because attack traffic was congesting a shared link. Figure 4.13 (bottom) shows larger latencies for successful queries at K-NRT, perhaps suggesting queuing at the router. We also observe that K-NRT-S2 seems more heavily loaded than the other two servers at K-NRT.

These examples show that individual *server* performance and reachability may not reflect overall *site-wide* performance and reachability. Measurement studies of anycast services should therefore ensure they study all servers at a site (not just specific servers) to get a complete picture of site and end-user-perceived performance.

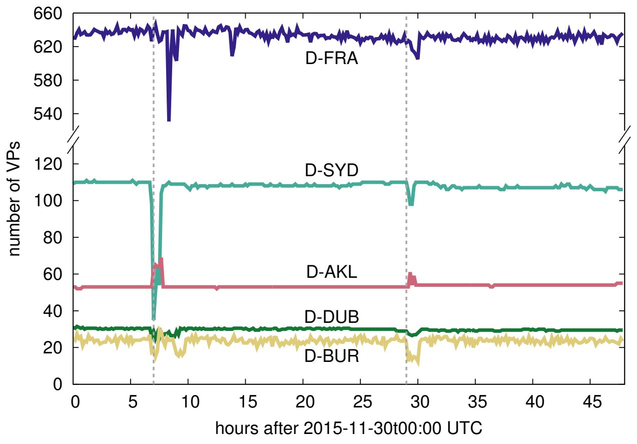


Figure 4.14: Reachability of those D-Root sites that were affected by the DDoS.

4.4.6 Are There Signs of Collateral Damage?

Servers today, such as the Root DNS servers we study, sometimes are located in data centers that are shared with other services. These services may be unrelated, other infrastructure (such as other top-level domains, TLDs), or even other Root DNS sites. Co-locating services creates some degree of shared risk, in that stress on one service may spill over into another causing *collateral damage*. Collateral damage is a common side-effect of DDoS, and data centers and operators strive to minimize collateral damage through redundancy, over-capacity, and isolation. Prior reports describe it as a problem but provide few details [155]. Recent work by Fontugne et al [71] takes a more in depth look at increased link latency caused by DDoS attacks.

Hosting details are usually considered proprietary, and commonality can exist at many layers, from the physical facility to peering to upstream providers, making it difficult to assess shared risk. From public data we therefore cannot establish direct causation in a specific common point. Instead, we assess shared risk by *end-to-end evaluation*: we look for service problems in other services that are not directly the target of event traffic. We study two services: D-Root, a letter that was not directly attacked [63], and the `.nl` TLD. They are chosen because they both show reduced end-to-end performance with timing consistent with the events, strongly suggesting a shared resource with event targets.

D-Root: Figure 4.14 shows the absolute counts of the number of RIPE Atlas VPs that reach several D-Root sites. D-Root has many sites; we report only subsets that had at least a 10% decrease in reachability during the time of the attacks and were reached by at least 20 RIPE Atlas probes.

These figures show that D-FRA and D-SYD sites both lost VPs during the event. Which data centers host these sites is not public, but correlation

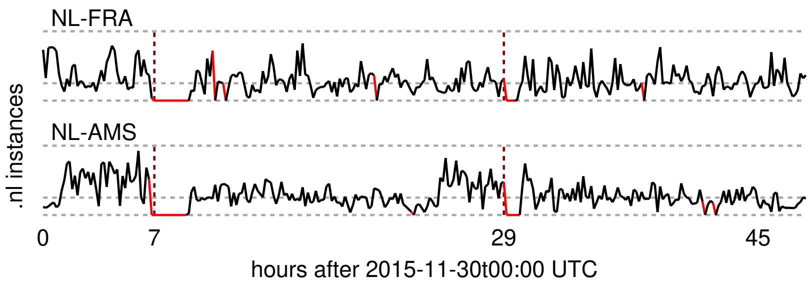


Figure 4.15: Normalized number of queries for `.nl`, measured at the servers in 10 minutes bins.

of these changes with the events suggests potential collateral damage. (Recall that RIPE VPs probe only one letter, so a reduction in VPs to one site implies either query loss or re-routing, not switching to another letter.)

Frankfurt: There are seven Root Letters hosted in Frankfurt (A, C, D, E, F, I, and K), and we previously observed that traffic shifted to K-FRA and yet that site suffered loss (Figure 4.4.4).

D-FRA sees only small decreases in traffic, suggesting it was only slightly affected by the events to sites for other letter in the same city. However, this change indicates some collateral damage for D-FRA.

The .nl Top Level Domain: Finally, we have also observed collateral damage at servers that are not part of the Root DNS. We see evidence for collateral damage occurring to the `.nl` top level domain. In addition to four unicast deployments, SIDN operates `.nl` on multiple anycast services. Figure 4.15 shows query rates for two anycast deployments located near Root DNS servers (exact rates and locations are anonymized). We see both sites show nearly no queries during both events. As a result of this collateral damage, during this period, `.nl` service was carried by other `.nl` servers.

4.5 Related Work

Distributed Denial-of-Service attacks are a broad area of study that has been addressed from many different angles in the past years. Studies have shown that DDoS attacks are effective [72].

DDoS attacks are common and growing: Arbor has documented their increasing use and growth in size [150], [151], and there have been DDoS attacks reaching 1.3 Tb/s [164]. Very large attacks often use different protocols to amplify basic attack traffic [31], [73], [74]. Yet DDoS-for-hire ("Booter" services) are easily available for purchase on the gray market—for only a few U.S. dollars, Gb/s attacks can be ordered on demand [75], [165].

Some approaches have been proposed to mitigate amplification [135], [76], spoofing [33], or collateral damage [77]. The continued, growing attacks show that mitigation has been incomplete and that spoofing remains widespread [36].

Many studies have looked at the Root DNS server system, considering performance [55], [66], [78]–[81], [166], [82]–[86], [167], client-server affinity [83], [168], and effects of routing on anycast [87], [169], as well a proposal to improve anycast performance in CDNs [88]. We draw on prior measurement approaches, particularly the use of CHAOS queries to identify anycast catchments [66].

Closest to the work in this chapter are prior analyses of the Nov. 30 events [127], [63], [157], [163]. These reports lend insight into the events, but were high level [127], [163] or reported only on specific letters [63], [157], [163].

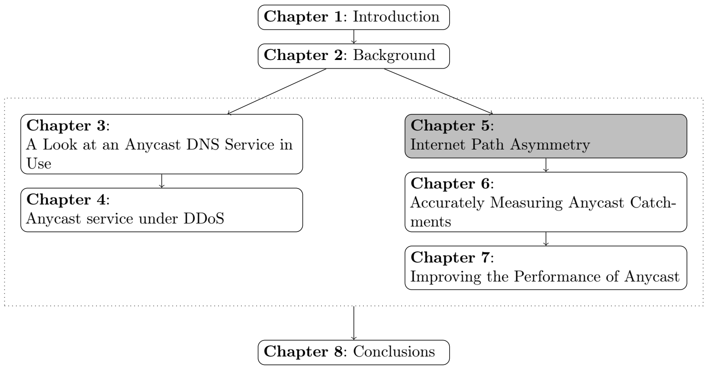
To the best of our knowledge, we are the first to combine multiple sources of measurement data to assess how a DDoS attack affects the several layers of the anycast deployment of Root DNS service. In addition, we are aware of no prior public studies on diverse anycast infrastructure operating under stress, including at the site and server level and its consequences on other services (collateral damage).

4.6 Concluding Remarks

This chapter provides the first evaluation of how anycast services behave under a DDoS attack. Our work evaluates the Nov. 30 and Dec. 1, 2015 events on the Root DNS, showing the effects of those events on a service that is run by multiple operators, with most analysis based on publicly available data. Our analysis shows different behaviors across different letters (each a separate anycast services), at different sites of each letter, and at servers inside some sites. We identify the role of different policies at overloaded anycast sites: the choice to absorb attack traffic to protect other sites, or to withdraw the service in hope that other sites can take over. We believe overall DNS service was robust to this attack, due to caching and the availability of multiple letters for service. However, we show that large attacks can overwhelm some sites of some letters. In addition, we show evidence that high traffic on one service can result in collateral damage to other services, possibly in the same data center. Our study shows the need to understand anycast design for critical infrastructure, paving the way for future study on anycast networks that may improve resilience.

CHAPTER 5

Internet Path Asymmetry


Chapter 1: Introduction
Chapter 2: Background
Chapter 3: A Look at an Anycast DNS Service in Use
Chapter 4: Anycast service under DDoS
Chapter 5: Internet Path Asymmetry
Chapter 6: Accurately Measuring Anycast Catchments
Chapter 7: Improving the Performance of Anycast
Chapter 8: Conclusions

In the previous chapters we looked at a number of anycast services, both during normal operation as well as under stress from a DDoS attack. As our goal is to improve anycast networks using a measurement-based approach, in this chapter we investigate Internet routing in order to gain a better understanding of what difficulties can occur when performing measurements. Specifically, we focus on investigating the phenomenon which is known as routing asymmetry, i.e. the forward and reverse route between two hosts on the Internet being different. We use the RIPE Atlas measurement framework to perform large-scale measurements to assess routing asymmetry as it occurs on the Internet. We find that many routes, approximately 87%, are asymmetric, and that their length is approximately 5 hops. Additionally, we show that Internet routes in general are stable over our measurement period. The key takeaway from this chapter is that asymmetry is widespread. This means that using standard tools such as traceroute, from the point of view of an anycast operator, has limited use since it does not necessarily discover the actual routes that clients use to reach its service. The work in this chapter was published as a research paper [89].

5.1 Introduction

The fact that Internet routing shows some degree of asymmetry is a well established fact [90]–[93]. Routing asymmetry means that, given two hosts A and B, the path from A to B (the forward path) is different from the path from B to A (the reverse path). Asymmetry can be problematic when trying to troubleshoot problems at host A that occur on the reverse path. The reason for this is that standard tools, such as `Traceroute`, are only able to determine the forward path from the viewpoint of host A.

There have been various studies that quantify Internet routing asymmetry. In this chapter we aim to reinforce those studies and provide a more in-depth analysis, by performing large scale measurements, to quantify asymmetry and determine where exactly it occurs. A better understanding of the characteristics of Internet asymmetry can help when attempting to troubleshoot problems that occur on the reverse path when only the forward path is known.

In this chapter we look into the asymmetry of network paths. We investigate to what extent the reverse path can still be determined using the forward path if the characteristics of Internet asymmetry are known. The goal of this study is to provide an in-depth analysis of Internet routing asymmetry. To perform this analysis we measure network paths between 4,000 probes across the world. We analyze the resulting data for network path asymmetry on the Autonomous System (AS) level, as opposed to individual router IPs. We show that most routes are not completely symmetrical, although the routes do have properties that still make them useful for specific applications, such as troubleshooting and collaboration with upstream providers. The contribution of this chapter is providing a quantification of the degree of asymmetry that occurs on the Internet, to improve further studies of the Internet, as well as assist operators particularly with regards to troubleshooting.

5.2 Related Work

Internet routing asymmetry has been the subject of a number of past studies [90], [91], [93], [94]. In this section we discuss studies that have investigated the level of routing asymmetry on the Internet and indicate what shortcomings they have that we aimed to solve.

First, the research in [91] on route asymmetry covers the AS level. They conclude that route asymmetry, at the AS level, is only present in approximately 14% of the routes. However, this research is based on results gathered using the Active Measurement Project (AMP) which runs mainly on academic networks and uses only 135 probes. In their follow up study [90], they use 350 probes selected from 1,200 public traceroute servers. They note that the routing asymmetry percentage is much higher on commercial networks, namely 65%, which negatively impacts the usability of Traceroute to measure reverse network paths.

In addition, while they have conducted extensive research on route asymmetry on the AS level they have not looked at the relative position of asymmetry (e.g. close to the target of the traceroute, in the middle or close to the source of the traceroute). If we are interested in the remaining usability of reverse paths this is a relevant metric, for example for applications that do not require the entire path to be symmetric, e.g. during troubleshooting. They proposed an interesting framework for quantifying the change in paths in which they use the Levenshtein Edit Distance (ED) algorithm as a way to determine the distance between two paths.

Secondly, research in [93] concluded that the asymmetry on the AS level is substantially higher than in [90], [91]. According to them, asymmetry on the AS level is as high as 90%. The cause of this difference could, for example, be that this study was conducted 5 years later or that their dataset is obtained using only a total of 220 probes with a biased distribution.

Finally, the research in [94] proposes a way of determining the actual path that a packet has taken to reach a point in a network with routing asymmetry in mind, from the viewpoint of the receiver. That research focused mainly on determining reverse routes for troubleshooting purposes (e.g. which network is dropping packets). Their method involves a system of widely deployed probes, IP spoofing and the use of an option in the IP header that is often not implemented. While the theory behind this method is sound, it can be difficult to deploy in practice for a few reasons. First, potential users need to have widely deployed probes in place. Secondly, their method uses the Record Route option in the IP header. However, this option is often ignored[170] and packets that use this option are usually dropped. Finally, the use of IP spoofing, the act of forging the source address, can be problematic due to issues with company policies, ethics and the fact that there are techniques to block IP spoofing such as proposed in Request for Comments (RFC) 2827 [95], which is currently known as Best Current Practice (BCP) 38.

5.3 Methodology

In this chapter we consider a *network path* an ordered list of networks that connect two end-systems on the Internet. Although there are studies that differentiate networks by IP address or even as IP address range[90], we chose to represent networks as AS. By using ASes it is trivial to cluster IP addresses that belong to the same administrative network.

As shown in Figure 5.1 there are two distinct paths between a pair of end-systems A and B: The forward path and the reverse path. When both paths are completely equal then the path is symmetric, otherwise it is asymmetric. To reliably determine a complete network path (both the forward and reverse parts) from the viewpoint of the receiver, the Internet would have to be completely symmetric.

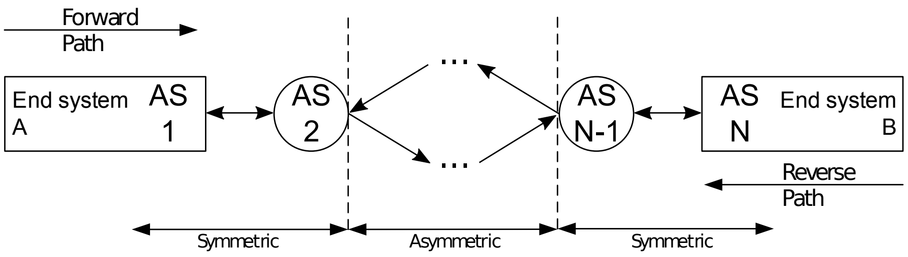


Figure 5.1: Network path

Our goal is to quantify the occurrence of path asymmetry on the Internet, and to quantify where this asymmetry is most likely to occur. Intuitively, diverging forward and reverse paths are less likely to occur near the end systems.

The main requirement for investigating our hypothesis was having a large amount of Internet connected computer systems which we could control. In order to meet this requirement we use RIPE Atlas. This project manages probes around the world for the specific purpose of network measurements. A probe is a dedicated network measurement device that can be placed in a network to allow measurements to be performed remotely. The Atlas project consists of approximately 10,000 distributed probes [96] worldwide. Although we are aware of several other measurement infrastructures, such as PlanetLAB [171], EmanicsLAB [172] and the NLNOG Ring [173], these do not provide the scale and distribution that was required for measurements that are representative of the Internet.

RIPE Atlas has imposed a credit system that limits measurements in three ways. The credits that are consumed per day, the number of measurements that can be run concurrently and the total number of credits that can be consumed. These limits have a consequence on the number of probes that can be used and in which combination. Credits can, for example, be earned by hosting a RIPE Atlas probe. It is due to this credit limit that not all probes that are available can be used. This further depends on the measurement layout, which probe measures what and to what other probe.

5.3.1 Measurement configuration

We considered three layouts in which the probes can conduct the measurements. Note that to be able to determine route asymmetry between two probes, each probe has to perform a traceroute to the other. In the considered layouts each probe performs traceroutes to the probes to which it is connected.

Fully connected layout (Figure 5.2a) - This layout has the advantage of utilizing the complete potential of the involved probes, every probe measures the path to every other probe. The disadvantage is that due to the credit limit only a very limited number of probes from the total can be used. For example:

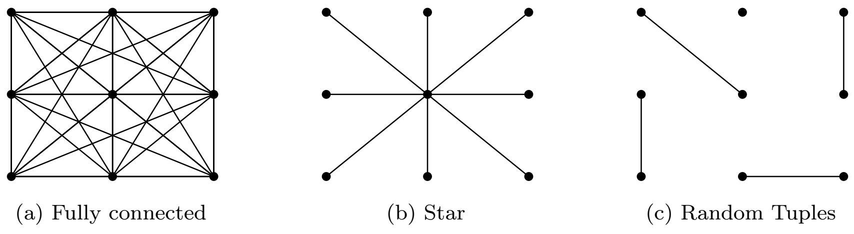


Figure 5.2: Probe layout

considering the 1 million credit limit only 112 probes can be used due to the high number of paths in this type of layout. A small number of probes means that specific network issues that occur at individual probes have a large impact.

Star layout (Figure 5.2b) - In comparison to the fully connected topology this has the advantage of allowing many more probes to be used. However, in this case the center probe will have a large impact on each measurement. Network issues at the center probe can cause the entire experiment to fail.

Random tuple layout (Figure 5.2c) - In this layout random tuples of probes are selected. This has the advantage of minimizing the impact of a single misbehaving probe. Furthermore, it allows for a much larger selection of probes, considering the Atlas limits. Because of these advantages this is the layout that we used.

Using the random tuple layout we selected 4,000 probes, meaning 2,000 tuples, in a way that favoured longer geographic distances. The attempt to have longer geographic distances is to prevent a large concentration of probes in Europe, as most probes are located there. The algorithm used to select the probes works by randomly picking probes and comparing the distance between them to some threshold (in our case: 10,000 km), if the threshold is exceeded then the probe tuple is added to the final result set. If, after a number of attempts (in our case: 2,000), no probe tuples can be found that exceed the threshold then the threshold will be lowered.

The distribution over continents in terms of numbers is shown in Table 5.1. There is a large skew towards Europe which is caused by the relatively large number of probes located there. The average distance between two probes in a tuple is 6,945 kilometers (as the crow flies).

For every selected pair consisting of probe A and probe B two measurements were scheduled. One measurement, consisting of a traceroute, was configured from probe A to probe B (the forward path) and another from probe B to probe A (the reverse path).

Network variances over time were smoothed out by scheduling the measurements to run every three hours, for ten days. This was limited by the total amount of credits we were allowed to consume. The measurements were

Continent	Selected	Available	Fraction	Fraction of selected
Europe	2,681	5,200	51.56%	67.03%
North America	724	1,003	72.18%	18.10%
Asia	267	420	63.57%	6.68%
Africa	157	223	70.40%	3.93%
Oceania	109	145	75.17%	2.73%
South America	59	87	67.82%	1.48%
Antarctica	1	1	100.00%	0.03%
Unknown	*2*	*4*	*50.00%*	*0.05%*
Total	***4,000***	***7,083***		***100%***

Table 5.1: Distribution over continents

performed from 14:00 on the 28th of July 2014 to 14:00 on the 7th of August 2014, Coordinated Universal Time (UTC).

5.3.2 Preliminary considerations

RIPE Atlas probes conduct their traces on the IP level where each hop consists of a single IP address. Because we want to look at the network paths from the AS level it was necessary to convert the measured paths. In order to convert IP addresses to their corresponding AS numbers we used the Border Gateway Protocol (BGP) routing table dumps obtained from the Remote Route Collectors (RRCs) managed by the Routing Information Service (RIS), which in turn is operated by RIPE. These routing tables contain a large amount of routes that are announced on the Internet by different ASes. Using these routes we are able to determine the AS number for a given IP range. The tool we used for this and its source is available online [174]. Alternatives to this method are provided by CAIDA [175] or MaxMind [176].

Each Internet Protocol (IP) address in the paths at the router level was converted to their corresponding AS number. We observe that it is common for multiple hops to occur within the same network. This is shown by the reduction in the number of hops in network paths at the router level in comparison to network paths at the AS level, which is, on average, 64.46%.

Our choice of probes was optimized to prevent a large cluster of probes in Europe by increasing the geographic distance between pairs, this may have introduced a bias in network path length. In order to show that this is not the case we plot the geographic distance, which is shortest distance between two points on a sphere (great circle distance), against the number of hops in the forward network path on the AS level. The result of this is shown in Fig. 5.3. As we expected there appears to be no clear correlation between the geographic distance and the number of hops.

In Figure 5.4 the distribution of the length of the measured paths is shown. Most paths contain five different AS-numbers. This means that in those cases three autonomous systems aside from the one the receiver and the sender are

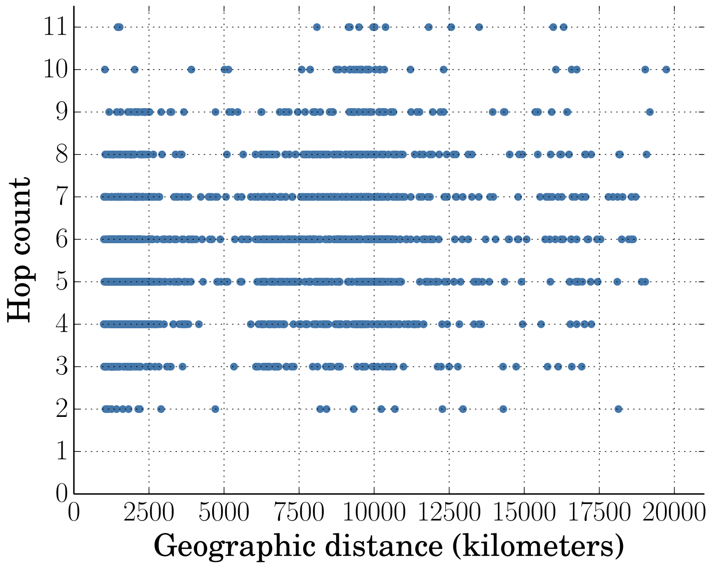


Figure 5.3: Geographic distance vs path length

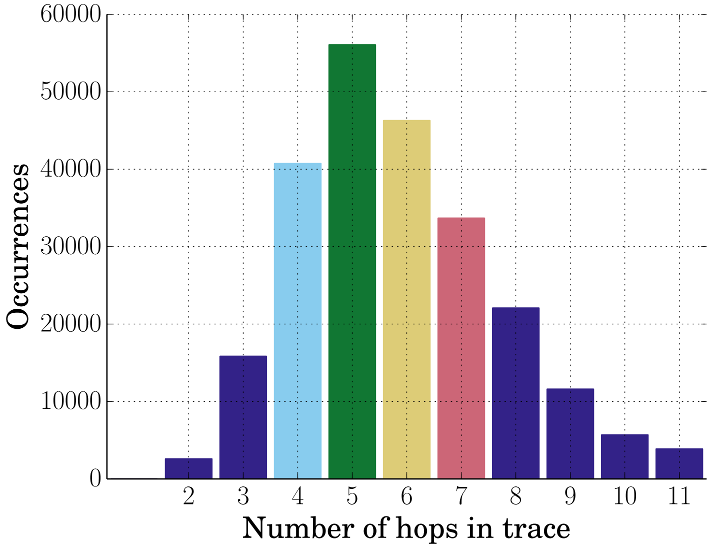


Figure 5.4: Distribution of path length

in (e.g. their Internet Service Providers (ISPs)) are involved in routing the packets.

The measurements that were performed by the probes were not completely perfect or complete. For example, some probes that delayed their measurements for too long or did not perform them at all. To filter out such measurements we have applied some filters to the dataset. Prior to the filtering we had 153,638 potential forward/reverse path pairs, of a theoretical 160,000. The absence of some paths can be attributed to some probes that did not respond or did not complete any measurements.

Since forward and reverse path measurements are initiated from two different probes there can be a delay between the two measurements being executed. To prevent the time difference from influencing the results we only match paths if they were measured reasonably close to each other in time, we arbitrarily set this limit at 600 seconds. If they are outside that time limit the forward/reverse pair is discarded. This prevents path instability from being interpreted as path asymmetry. This filter reduced our potential forward/reverse path pairs by 16,103 ($\approx$ 10.5%).

The second filter we implemented is based on the principle that the first hop of the forward path should be the last hop of the reverse path, because these are the origin and destination networks. The same principle applies in the opposite direction. Measurements where this is not the case can be caused by incomplete traces. We filtered all forward/reverse path pairs where this was the case. This removed 14,620 results from the set ($\approx$ 9.5%).

To prevent probes that measured completely empty paths from influencing the results we filtered all pairs that contained a completely empty path. Completely empty paths do not exist in actual networks, as a network path always contains at least a single hop, even if the source and target IP addresses are in the same network. Empty paths can be caused by incomplete traces or probes that are not executing their measurements. This filter reduced our result set by 3,365 ($\approx$ 2.2%).

The three filters that we implemented left a total of 119,550 or 74.72% of the theoretical 160,000 pairs.

For paths that contained unresolvable hops we considered a few options (i.e. hops for which no response is obtained, typically these are marked with the star symbol when performing a traceroute). The first option is to discard all path pairs that contained such a hop. However, this would impact a significant part of the result set as unresolvable hops are common. Instead, based on what was done in He et al. [97], we consider an unresolvable hop as a wild card, meaning that it will match any hop in the opposite path that is in the same position.

5.4 Analysis

In this section we analyze the dataset that was obtained using the methodology described in the previous section. Our dataset contains a total of 2,275 unique AS numbers, of which 1,717 contain one or more probes. Of all results in our dataset, 15,053 (12.6%) forward/reverse path tuples are completely symmetric and 104,497 (87.4%) show asymmetry. This is in line with the results found in [93], however, we use far more probes. The large percentage of asymmetric paths further justifies studying the characteristics of Internet asymmetry.

Before we start the analysis we introduce two variants for calculating the ED between two paths. One is the Levenshtein algorithm [98] which was first used for this purpose in [90], [91]. The Levenshtein algorithm counts the number of required insert, delete or change operations to make two paths equal to each other. The Levenshtein algorithm was originally intended to be used to measure the differences between strings, however, it can be used without modification for measuring the change in network paths. In addition to the Levenshtein algorithm we also use a variation called Damerau-Levenshtein [99]. Damerau-Levenshtein extends the original algorithm by also counting transpose operations as a single change. It is much less sensitive to swapped hops. The extended algorithm is interesting in contexts where the presence of ASes on a path are of more importance than their specific location.

5.4.1 Stability over time

We begin our investigation by determining the change of paths over time. This is of interest because it is not always possible to measure the reverse path at the exact time that the forward path was established. We calculate the average ED over all paths over time. The ED is determined as follows: The first path to a destination is taken as a ground truth to which each consecutive path is compared. We then calculate the ED based on the Levenshtein algorithm. We had to modify the algorithm slightly because not all paths are of the same length, which would cause longer paths to have a much higher impact on the results than shorter paths. Therefore, we normalize the ED by dividing it by the path length as shown in Equation 5.1.

$$\frac{ED(path_1, path_2)}{MAX(len(path_1), len(path_2))} \tag{5.1}$$

The normalized ED is then between 0.0 (i.e. completely the same) and 1.0 (i.e. completely different). Figure 5.5 shows the results of this analysis. Note that the graphs indicate that network paths are not subject to great change over time. The instability appears to stop increasing after 8 days, therefore measurements should be done over a longer period of time to show if this behavior persists. Furthermore, we compared the results using the Levenshtein algorithm to the Damerau-Levenshtein algorithm and this showed results which are almost completely identical. This indicates that the relative position of a network in a path is stable.

5.4.2 Absolute difference

We look at the absolute difference between the forward and reverse path pairs to get an understanding of how big the impact of routing asymmetry is. We define

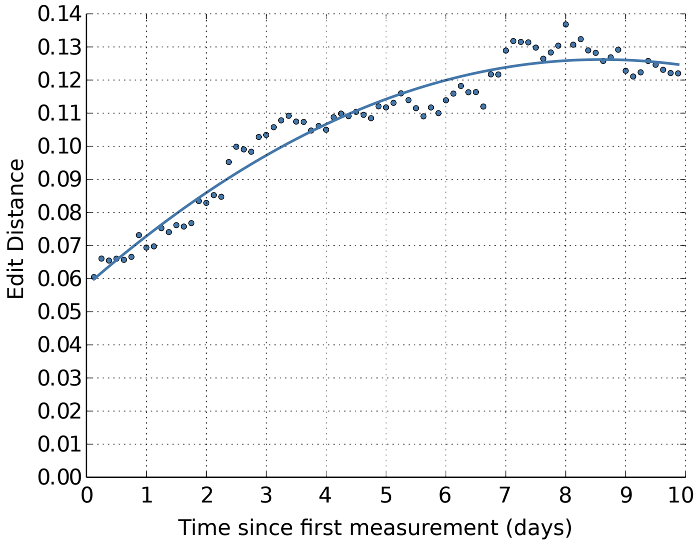


Figure 5.5: ED over time using Levenshtein algorithm

the absolute difference as the ED between the forward and reverse path. The ED between all path pairs is shown in Figure 5.6. Note that the difference between the results of the two algorithms indicates that it is a common occurrence for two hops to be swapped in either the forward or reverse path. Furthermore, most forward/reverse path pairs show a distance of either 1 or 2 from their counterpart.

5.4.3 Relative difference by position

In this section we show the similarity of hops based on their relative position in the path. This shows if a certain hop is usable for mitigation. If a forward and reverse trace have different lengths then they are not included in this figure, which results in 28,139 result pairs being used in Figure 5.7. This shows how the symmetry decreases as we move closer to the middle of the path, as expected. It also shows that for the longest path (7 hops) the middle hop is equal in both the forward and reverse path in approximately 60% of the cases. The asymmetry in the figure itself indicates that the accuracy can be further increased by performing more measurements, which we leave as future work.

Given this measure of asymmetry we try to find out if the majority of asymmetry is caused by a small number of networks (i.e. ASes). We look at which ASes are involved when asymmetry occurs. From the approximately 500 ASes that are involved we see that the top 10 is responsible for 48% of the total asymmetry. We manually categorized these ten ASes in three types: T1

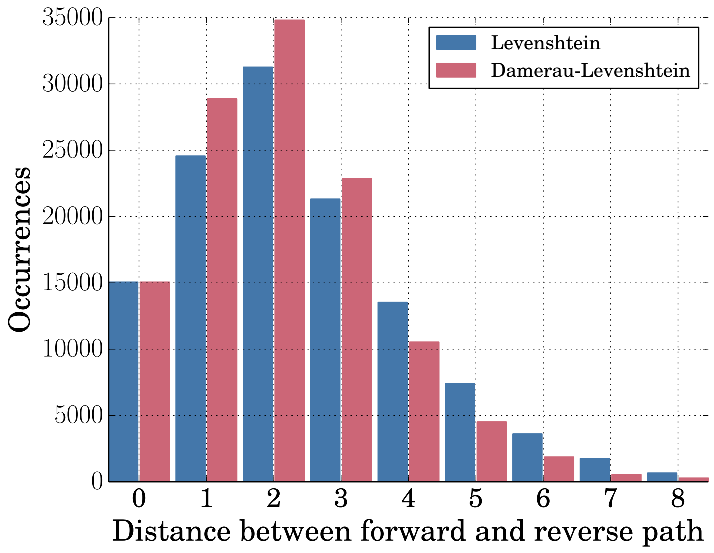


Figure 5.6: Distance between forward and reverse path

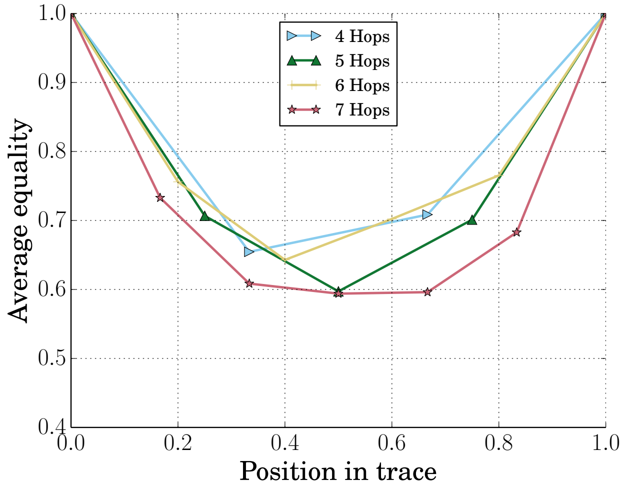
Position	ASN	Name	Type
1	3356	Level 3 Communications, Inc.	T1
2	174	Cogent Communications	T1
3	1299	TeliaSonera International Carrier	T1
4	3257	Tinet SpA	T1
5	3216	OJSC Vimpelcom	T2
6	34984	TELLCOM ILETISIM HIZMETLERI A.S.	T2
7	1200	Amsterdam Internet Exchange B.V.	IXP
8	2914	NTT America, Inc.	T1
9	6453	TATA Communications, Inc.	T1
10	6695	DE-CIX Management GmbH	IXP

Table 5.2: Top 10 ASes involved in asymmetry

for Tier 1 providers, T2 for Large ISPs and IXP for Internet Exchange Points. The results are shown in Table 5.2. It is clear that the largest Internet Service Providers (i.e. Tier 1 providers), cause the largest part of the asymmetry. It is likely that this is because those providers are also the ones which have the highest number of peering connections.

5.4.4 Consecutive equal hops

We count the number Consecutive Equal Hops (CEH) from each side of the forward/reverse path that are equal, not counting the source and target networks. This approach can be used even if the lengths of the forward/reverse path are



#	Average equality						
1	1.00						
2	1.00	1.00					
3	1.00	0.81	1.00				
4	1.00	0.65	0.71	1.00			
5	1.00	0.71	0.60	0.70	1.00		
6	1.00	0.76	0.64	0.70	0.77	1.00	
7	1.00	0.73	0.61	0.59	0.60	0.78	1.00

Figure 5.7: Average equality by position in trace

unequal. The average number of CEH, divided by two to get an average for each side, is plotted against the total number of hops in the forward path in Figure 5.8.

Included in Figure 5.8 is the 95% confidence interval. This figure shows that for path lengths 6 and 7 there is on average at least one additional equal network aside from the source and target networks. For the most common path length, five, there is one network that is the same in both the forward and reverse path in approximately 75% of the cases.

In Figure 5.9 only the first complete result for each pair is considered. These graphs show that it is not necessary to do repeated measurements over a longer period of time to determine route asymmetry. Note that this suggests that route asymmetry does not vary significantly over time.

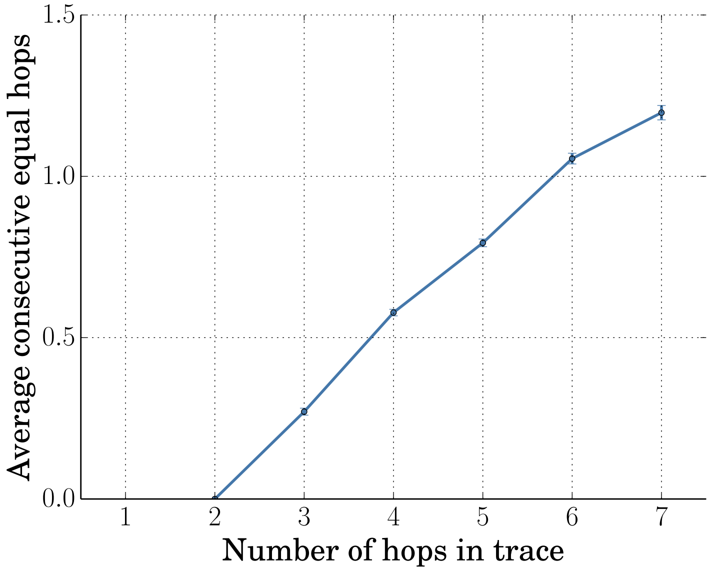


Figure 5.8: Average number of CEH between Forward and Reverse paths
All results

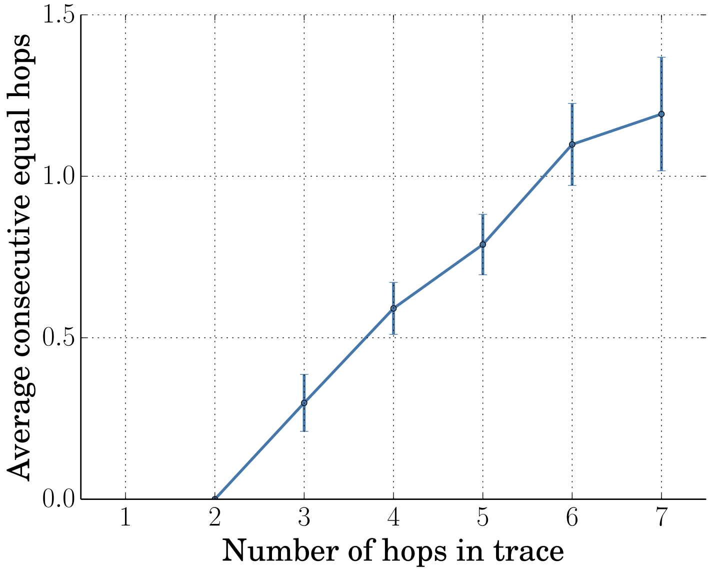


Figure 5.9: Average number of CEH between Forward and Reverse paths
First complete trace only

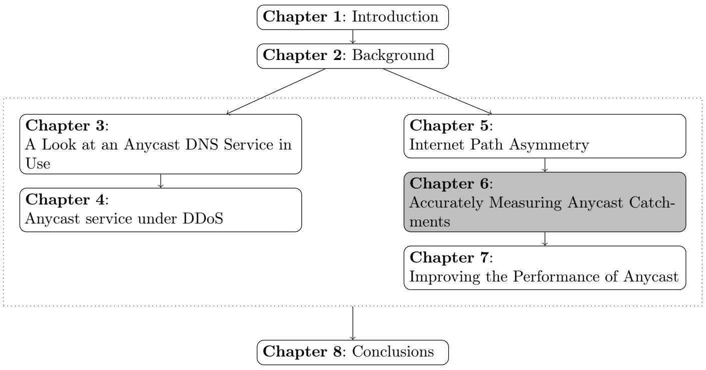
5.5 Concluding Remarks

In this chapter we have analyzed and characterized several aspects of Internet routing asymmetry. Our analysis has been conducted on a large scale using RIPE Atlas. The results from our study contribute to assist researchers and engineers in making valid assumptions while using forward/reverse paths data. In addition, we contribute to give a conclusive overview on the partial asymmetry of Internet routing.

The usability of Traceroute for measuring reverse paths is, depending on the application, questionable. We have confirmed the presence of asymmetry in the majority of Internet routes, and determined where this asymmetry occurs. Our hypothesis, that reverse network paths can be reliably discovered via standard tools near the end-systems has been confirmed. We have found, in the worst case, a hop, representing an AS, is the same in the forward and the reverse path in 59% of the cases, but often more.

CHAPTER 6

Accurately Measuring Anycast Catchments



Chapter 1: Introduction
Chapter 2: Background
Chapter 3: A Look at an Anycast DNS Service in Use
Chapter 4: Anycast service under DDoS
Chapter 5: Internet Path Asymmetry
Chapter 6: Accurately Measuring Anycast Catchments
Chapter 7: Improving the Performance of Anycast
Chapter 8: Conclusions

In the previous chapter we assessed why traceroute misses certain aspects of Internet routes because of path asymmetry. In an anycast scenario, path asymmetry in the case of communications initiated from the side of the service is very likely, and the reverse (or return) path often leads to a different anycast site. Before that, we showed that although many deployments of anycast exist, due to the size and opaqueness of the Internet it can be hard to achieve the ideal catchment for a certain goal, e.g. robustness against Distributed Denial-of-Service (DDoS) attacks. In this chapter we introduce a novel methodology that allows operators to determine the catchment of their anycast network globally, with a high resolution in terms of IPv4-space that is covered. In comparison to traditional measurement systems, such as RIPE Atlas, this method offers 430× the number of vantage points. We show that, by performing a case study on B root, this methodology can be used to assess the load distribution on a production system ahead of time by deploying a test prefix. Additionally, by mapping load by time slots, the distribution over time across different anycast sites can also be determined. The work in this chapter was published as a research paper [48].

6.1 Introduction

One of the main challenges in managing anycast is that Border Gateway Protocol (BGP) routing is not always what one would expect. Absent other policies, BGP defines nearness in terms of Autonomous System (AS)-hops, but one AS hop across an organization with a global network (such as AT&T, Tata, and NTT) can have very different performance characteristics than one AS hop across a small Internet Service Provider (ISP). In addition, the trend of a flatter Internet [100] means that AS hops provide coarser control than it did in the past. More importantly, BGP policy controls allow ISPs to manipulate routing for business reasons; policy controls are widely used to do traffic management.

Current approaches to manage anycast catchments use one-off active measurements [62], platforms for active measurement such as RIPE Atlas [65], [101], [102], commercial services (for example, [177]), and analysis of anycast service logs [4]. While these approaches have provided insight, and RIPE Atlas and commercial services are in wide use, even the largest services have relatively small numbers of vantage points (from hundreds to 10.000 or so), and it is unclear how these measurement systems relate to actual operational traffic. Analysis of anycast service logs offer an accurate representation of actual load, but require the anycast service to be in operation and active use.

The overall contribution of this chapter is to provide a new approach to mapping anycast catchments (Section 6.3), named Verfploeter, that has been validated through real world ground truth. This approach provides *broad coverage* and can be combined with traffic history to provide *estimated load*, providing *operational value* for an anycast service. The insight in our new measurement approach is to use *active probing using the anycast service itself* and we can use *historical traffic to predict future load.* In Section 6.5 we show that active probing allows coverage of the ping-responsive Internet, currently about 4M /24 networks, providing 430× more information than current public measurement platforms. By contrast, coverage from existing platforms scales with the ability to deploy physical devices or virtual machines, both of which are limited.

The second contribution of this work is to use Verfploeter to examine the operational catchment for B-Root and to study anycast in Tangled, a nine-site anycast testbed (Section 6.6). B-Root deployed anycast in May 2017, and *our approach contributed to the success of this planning and deployment.* Analyzing this active network deployment allows us to compare the predictive capability of our approach to prior approaches such as RIPE Atlas. Evaluation of our Tangled testbed lets us test a larger anycast deployment (nine sites compared to B-Root's two sites). Our approach provides a new way to evaluate anycast stability with much broader coverage than recent studies [103].

Although our case study with B-Root and Tangled focuses on the Domain Name System (DNS), Verfploeter can examine any anycast service, although load prediction requires a system that can estimate historical traffic load.

A complete version of Verfploeter is available as open source at `https://github.com/woutifier` and `https://ant.isi.edu/software/verfploeter/`. We have released all the data used in this chapter (except *LN-4-12*, which is not ours); see citations in Table 6.1 and Table 6.2.

6.2 Related Work

There have been several prior approaches to measure anycast catchment using a variety of techniques.

Use of Open Resolvers: Early work used Open DNS Resolvers in combination with PlanetLab and Netalyzr to map catchments of anycast services [66]. While Open Resolvers provided a broad view at the time of their study (300k Vantage Points (VPs)), they are being steadily shut down out of concerns about their use in DNS amplification attacks [178]. While open resolvers offered a very large set of vantage points, they are fewer than the method we propose that uses ping-responsive networks. (A direct comparison is potential future work.)

Measurement Platforms: The most common method of assessing anycast is to use public or private measurement platforms that offer physical or passive VPs around the Internet. RIPE Atlas [65] and PlanetLab [171] are both openly available and widely distributed, and a number of commercial platforms are also available. Systems we are aware of range from hundreds to around 10k VPs.

Several studies, both by others and us, have used measurement platforms to study anycast [3], [62], [66], [167], [101], [102], [179], [180]. As pre-deployed measurement platforms these systems are available and can measure anycast services externally (without requiring support from the service operator). The main weaknesses of these systems are that they are slow and expensive to grow, and deployment is often skewed relative to the population of Internet users. This skew has been noted in many prior studies and was studied also studied explicitly. [104].

Client-side measurements: Recent work examined the Microsoft Bing Content Delivery Network (CDN) [62], using both log analysis (see below) and active client-side measurements. Their client-side analysis measures performance using JavaScript injected into search results of a small fraction of Bing users. Client-side measurements can get very broad results (like Verfploeter), but are not possible for all services. DNS and other non-web services do not support client-side modifications, and they may also be difficult for websites hosted by multiple parties.

Traffic and Log Analysis: Anycast operators have always been able to assess current anycast performance by analyzing their own traffic and server logs. Recent work examined a variety of CDNs [4], [62]. As the service operator, log analysis requires no external measurements and can cover the entire service. While important, analysis of existing services can only study the *cur-*

rent deployment—it requires active use by a large number of users and cannot directly support pre-deployment planning. Second, log files may be unavailable due to privacy concerns, cost of storage or retrieval, or concerns about performance impact on operational services. We use logs when available, but do not require them.

Performance Analysis of DNS Services: There have been a number of analyses of root DNS service, both pre-anycast [79] and with anycast for latency [66], [86], [167], [102], [105] and DDoS [3].

To the best of our knowledge, our work is the first to present this Internet Control Message Protocol (ICMP)-based anycast catchment determination approach. Further, we do not know of any larger scale catchment measurement with open datasets against a real-world anycast deployment.

6.3 Verfploeter: Global Probing

Our approach has components to map anycast catchments for a large fraction of prefixes in the Internet, and to estimate load from each of these prefixes.

6.3.1 Mapping Anycast Catchments

Traditional approaches to measuring anycast catchments use many VPs around the Internet; each VP queries the anycast service to determine its catchment. In prior work for DNS, the VPs are typically RIPE Atlas probes [154], [102], and the queries use DNS TXT records, with the special CHAOS network type, and the name "hostname.bind" [106], or the newer NSID option in DNS [107]. One can augment these methods with traceroutes to detect possibly spoofed replies [66]. All of these approaches require deployment of active probes around the Internet. The largest studies we know of use between 9,000 and 10,000 VPs, all the active VPs in RIPE Atlas.

Our insight is that we do not need control over active vantage points if we can *solicit* messages from around the Internet that will identify their catchment. Rather than handle both queries and responses from the VPs, we instead generate queries that cause *VPs to respond and reply to the anycast system*; we define these as **passive VPs**. If we can capture traffic at all anycast sites, then for each VP that responds, we can determine to which anycast site it is associated, and thus in which catchment it belongs. In effect, we shift the active side that generates and receives queries from the VP to the anycast network itself, yet capture observations from millions of passive VPs. (Although the anycast sites capture the data, the ping targets are the vantage points because they each generate a catchment report.)

Figure 6.1 compares these methods. On the left, traditional mapping sends queries (black arrows) from VPs into the anycast system. On the right, we send queries *from* the anycast network block (defined by the source address), *to passive VPs* in most /24 IPv4 networks. Their replies return to the site for

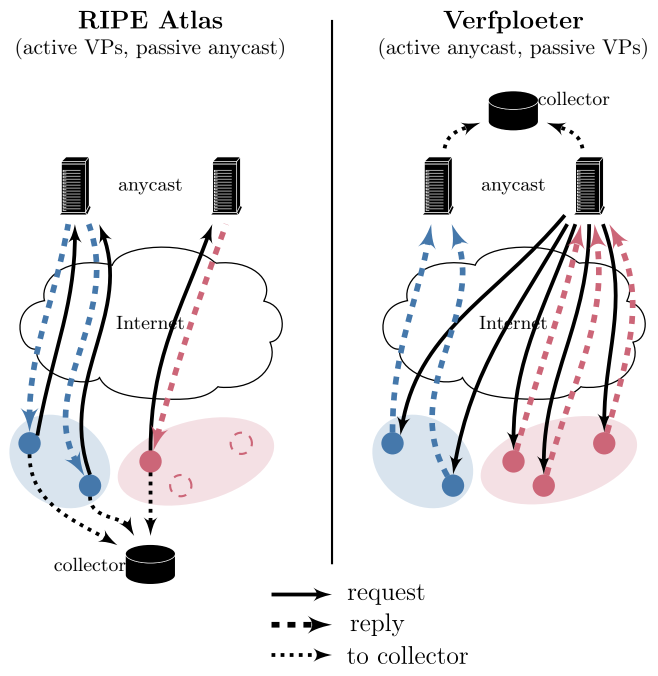


Figure 6.1: Traditional catchment mapping from active VPs using a testbed like RIPE Atlas (left); and using Verfploeter with queries originating in the anycast system (right).

their catchment, even if it is not the site that originated the query. We can think of paths in the scenario on the right as being highly asymmetric, however, in the case of anycast asymmetry is much more likely to occur than in a unicast scenario as studied in the previous chapter.

In Verfploeter, our *queries* are ICMP Echo Requests (pings), sent using a custom program, soliciting ICMP Echo Replies. Queries are sent from a designated measurement address that *must be in the anycast service IP prefix.* Unlike traditional catchment mapping, it is not the reply payload that indicates catchment, but instead *the catchment is identified by the anycast site that receives the reply.*

Our *passive VPs* are any computers in the Internet that reply to pings. We use a recent ISI IPv4 hitlist [108]. In principle, we could ping every IPv4 address to get complete coverage from all addresses that reply. We use hitlists instead because they provide representative addresses for each /24 block. They are most likely to reply to pings, and with one address per /24 block, we can reduce measurement traffic to 0.4% of a complete IPv4 scan. (We select /24s as the smallest routable prefix in BGP today, since anycast depends on BGP and assume that there is no traffic steering within such prefixes.)

We send requests in a pseudorandom order (following [109]), and relatively slowly (about 10,000 queries per second), to spread traffic, limiting traffic to any given network to avoid rate limits and abuse complaints. Although it is technically relatively trivial to perform the measurement much faster, there is little penalty for probing over 10 or 20 minutes.

We must capture traffic for the measurement address with our response collection system. We can capture traffic at the routers (without having a computer at the address), or by running computers that capture traffic on the address itself. These captures must happen concurrently at *all anycast sites*. We have three different *response collection systems*: first is a custom program that does packet capture and forwards responses to a central site in near-real-time. Second, we collect replies with LANDER [181], an existing packet capture system that collects data continuously. Third, we have also used tcpdump directly to capture traffic specifically for the measurement address. We use the first method for Tangled and the latter two methods at B-Root.

While the measurement address is in the service's /24 prefix, it can be a different address and does not need to see non-measurement traffic. Operators already operate services on these networks, and measurement can be done either on a virtual IP address associated with the computer providing service, on dedicated measurement hardware, or by using virtual machines on the same network. The systems participating in the measurement should be time synchronized so that collected data can be easily combined, but standard techniques like Network Time Protocol (NTP) are sufficient.

We send only a single request per destination IP address, with no immediate retransmissions. We see replies from about 55% of blocks (Table 6.4), consistent with the 56% and 59% seen in previous studies [108]. While incomplete, we get responses for millions of blocks. We could improve the response rate by probing multiple targets in each block (as Trinocular does [110]), or retrying immediately. Exploration of these options is future work. Finally, we copy all responses to a central site for analysis. Total traffic across the service is a modest 128 MB per measurement. We currently copy data manually, or use a custom program that forwards traffic after tagging it with its site.

Our approach to catchment mapping requires active participation at all anycast sites—it requires cooperation of the anycast operator, but it does not require additional Internet-wide infrastructure (such as distributed VPs). Fortunately, anycast operators are strongly motivated to understand their systems. These trade-offs are the opposite of traditional anycast mapping, which requires active VPs but not support of the target anycast system.

We do not model BGP routing to predict future catchments, *we measure actual deployment*. To predict possible future catchments from different policies, one must deploy and announce a test prefix that parallels the anycast service, then measure its routes and catchments. (We assume the test prefix will encounter the same policies as the production prefix.) Fortunately, anycast providers often announce anycast on a /24 prefix, and a larger, covering,

/23 prefix with unicast (this approach protects against corner cases with some routing policies [182]). The non-operational portion of the /23 could serve as the test prefix.

6.3.2 Load Estimation

Planning anycast deployment is more than just mapping catchments—different services can experience very different loads, depending on the distribution and usage patterns of its client base. We therefore build load estimates for each network block (/24 prefix) that accesses a service, so we can calibrate the loads that will be generated by a given catchment.

We assume operators track traffic volumes for their systems and can use the recorded historical data to estimate future loads. (For example, all DNS root operators collect this information as part of standard RSSAC-002 performance reporting [183].) For our study of B-Root we use historical data from its unicast deployment. When no operational load data is available, as in Tangled, one must estimate load using data from a similar service, or assume uniform load if no better estimates are available.

We consider three types of load: queries, good replies, and all replies. Queries represent incoming load on the servers, while replies are the results. Query packet load counts may differ from replies if response rate limiting is used to blunt DNS amplification attacks [184]. We separate out good replies from all replies because of the large fraction of queries to non-present domains in root-server traffic (first observed in 1992 [111] and still true today); operators may wish to optimize for volume or for good replies.

In principle, we can estimate load over any time period. Practically, we compute it over one day, and look at overall traffic using hourly bins.

6.4 Measurement Setup and Dataset

Using the proposed ICMP-based method, Verfploeter, we measure the catchment of two anycast services, B-root and an anycast testbed (Tangled), from more than 6.4M VPs (IP addresses). (Table 6.1 lists all datasets we use, and each figure or table reports the dataset it uses in the caption.) We add geolocation information for these blocks using MaxMind [176]. Accuracy of this geolocation is considered reasonable at the country level [112]. We also use Route Views and RIPE Routing Information Service (RIS) data to determine the AS number for each scanned IP address and the prefixes that are announced by each AS.

Data cleaning: We remove from our dataset the duplicate results, replies from IP-addresses that we did not send a request to, and late replies (15 minutes after the start of the measurement). Duplicates are caused by systems replying multiple times to a single echo request, in some cases up to thousands of times, accounting for approximately 2% of all replies. Other systems, when pinged,

Id	Service	Method	Start	Duration
SBA-4-20	B-Root	Atlas	2017-04-20	8 m
SBA-4-21	[187]		2017-04-21	8 m
SBA-5-15			2017-05-15	10 m
SBV-4-21	B-Root	Verfploeter	2017-04-21	20 m
SBV-5-15	[188]		2017-05-15	20 m
STA-2-01	Tangled [189]	Atlas	2017-02-01	10 m
STV-2-01	Tangled	Verfploeter	2017-02-01	10 m
STV-3-23	[190]		2017-03-23	24 h

Table 6.1: Scans of anycast catchments for B-Root and our testbed (Tangled). Scans were done on various days for comparison. Dataset *STV-3-23* contains 96 measurements over 24 hours, each 10 minutes long.

reply from a different IP-address than the original target destination. Methods such as alias resolution might clarify this, however, further investigation is out of the scope of this thesis.

6.4.1 B-Root

We validate the proposed methodology by providing a detailed view of the catchment of one of the DNS root-servers. B-root is the most recent root letter to make the change from unicast to anycast. B-Root deployed anycast at the beginning of May, 2017 [185], adding a site in Miami to its original site in Los Angeles (Table 6.3).

B's new deployment of anycast makes it an interesting analysis target. Unlike the other DNS Roots, B does not have a history of anycast deployment to guide its choices (although of course it draws on experience of other anycast deployments).

Dataset: We study B-Root catchments using several scans using both RIPE Atlas and Verfploeter, as shown in Table 6.1. We estimate B-Root load using two day-long datasets listed in Table 6.2. As a baseline we use data from Day in the Life of the Internet (DITL) 2017 (A Day in the Life of the Internet [186]), taken Wednesday, 2017-04-12 Coordinated Universal Time (UTC), before B-Root was using anycast. We then test against Thursday, 2017-05-15 UTC, after B-Root anycast was well established.

6.4.2 Anycast Testbed

We augment our measurements of B-Root with measurements of our anycast testbed, *Tangled.* This testbed has 9 sites around the world: 5 sites in Europe, 2 in the USA, and 3 other sites spread across Asia, Oceania and South America

				Queries	
Id	**Service**	**Date**	**Site**	q/day	q/s
LB-4-12	B-Root [191]	2017-04-12	LAX	2.34G	27.1k
LB-5-15	B-Root [192]	2017-05-15	both	2.20G	25.4k
			LAX	1.78G	20.6k
			MIA	0.407G	4.71k
LN-4-12	NL ccTLD	2017-04-12		redacted	

Table 6.2: Datasets used to study load (IPv4 UDP queries only).

Service	**Airport code**	**Location**	**Host**	**Upstream**
B-Root	LAX	US, Los Angeles	USC/ISI	AS226
	MIA	US, Miami	FIU/AMPATH	AS20080
Tangled	SYD	AU, Sydney	Vultr	AS20473
	CDG	FR, Paris	Vultr	AS20473
	HND	JP, Tokyo	WIDE	AS2500
	ENS	NL, Enschede	Univ. of Twente	AS1103
	LHR	UK, London	Vultr	AS20473
	MIA	US, Miami	Florida Int. Univ.	AS20080
	IAD	US, Washington	USC/ISI	AS1972
	GRU	BR, Sao Paulo	Florida Int. Univ.	AS1251
	CPH	DK, Copenhagen	DK Hostmaster	AS39839

Table 6.3: List of anycast sites used in our measurements.

(Table 6.3). Tangled allows us to study how a larger amount of sites interact, and to perform experiments which we cannot do in an operational anycast service. We use it to understand anycast instability and ASes that appear in multiple catchments (Section 6.6).

Limitations: Three of the testbed sites share a common ISP, which might impact the overall catchment. The anycast site in São Paulo has all its traffic routed via the same link as the site in Miami, which might cause announcements from São Paulo to be hidden. Finally, the connectivity at the site in Japan is such that it does not attract much traffic since announcements from other sites are almost always preferred over it. Prior to the measurement, the connectivity of each site was validated individually by announcing our prefix from that location only. Such limitations are not particular to our testbed as similar features can also be observed in public anycast services [102].

Dataset: As shown in Table 6.1, we measured the catchment using both Verfploeter and Atlas on Wednesday, 2017-02-01 UTC. We also determined the catchment of Tangled, using only Verfploeter, every 15 minutes during a 24 hour period starting 2017-03-23 10:57 UTC, for a total of 96 measurements.

In total we collected 342,604,759 ICMP replies, of which 324,675,876 ($\approx 95\%$) remained after cleaning.

For each measurement we transmitted one ICMP packet to each of the 6.4M IPs from the hitlist, at a rate of 10k/second to prevent overloading networks or network equipment. Each measurement round took 10.5 minutes to complete. A unique identifier in the ICMP header was used in every measurement round to ensure dataset separation.

6.5 Analysis of the Verfploeter Mechanism

In this section we examine the Verfploeter measurement method. We show the broader coverage of Verfploeter compared to RIPE Atlas, and how catchment mapping from Verfploeter can be combined to historic traffic load to accurately predict load at individual anycast sites.

6.5.1 Utility: Operational Evaluation of Anycast Catchments

A long-standing goal of anycast mapping is to assess load balancing and routing problems [167], [102]. We next look at B-Root's anycast distribution. Deployed recently in May 2017, it has only two sites, but we are able to deploy Verfploeter on it.

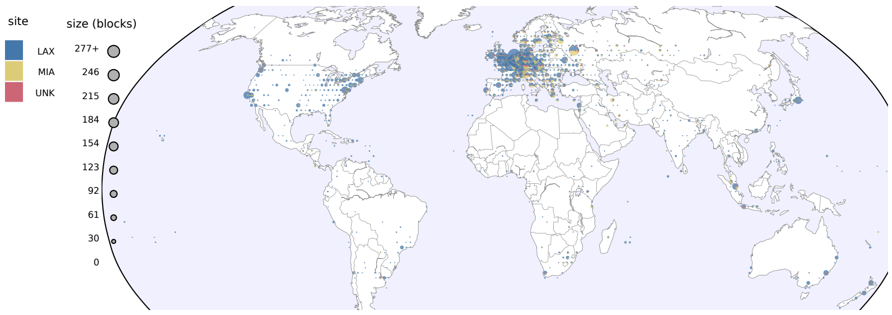
We have measured the geographic footprint of B-Root with RIPE Atlas (Figure 6.2a) and Verfploeter (Figure 6.2b). These maps highlight a couple of important differences between these measurement methods.

First, Verfploeter *has much broader coverage*: Atlas coverage is good in Europe and reasonable in North America, but sparse elsewhere and almost absent in China. Verfploeter provides good coverage for most of the populated globe. Second, even where coverage is good, Verfploeter *provides far more numerous observations*—the scale of Figure 6.2b is three orders of magnitude greater than Figure 6.2a.

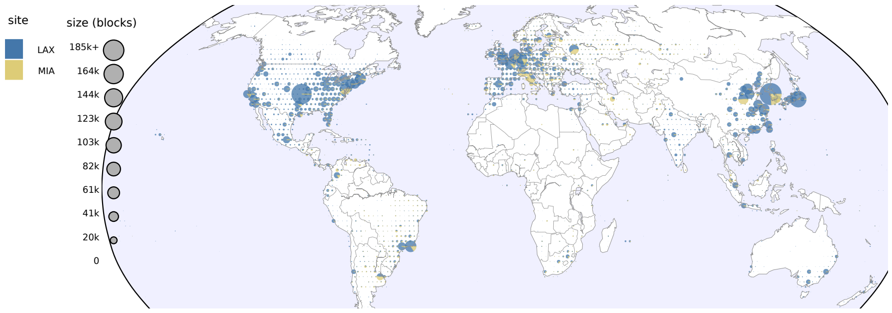
These differences are particularly important for examination of B-Root catchments in South America and China. The broader coverage is important to understand, for example, how a host in China might select a B-Root site: Atlas cannot comment, but Verfploeter shows most of China selects the LAX site.

The denser coverage in South America also helps highlighting the impact of B-Root's hosting ISPs. B-Root's ISP in MIA (AMPATH) is very well connected in Brazil and Argentina, but does not have direct ties to the west coast of South America. This difference shows in the wider use of the MIA site in Brazil, and less use of it in Peru and Chile.

Better coverage in locations like these that currently have poorer coverage by RIPE Atlas are important, particularly since East and South Asia are home to many Internet users but few Atlas VPs.



(a) RIPE Atlas coverage of B-Root (Dataset: *SBA-5-15*)



(b) Verfploeter coverage of B-Root (Dataset: *SBV-5-15*).

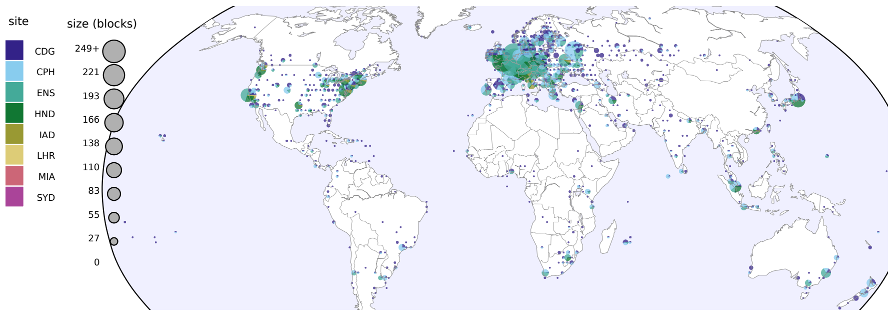
Figure 6.2: Geographic coverage of vantage points for RIPE Atlas and Verfploeter for B-Root, in two-degree geographic bins. The pie in each bin is colored by site (blue: LAX; yellow: MIA; red: other). Circle areas show number of address blocks (Verfploeter) or VPs (Atlas) at different scales.

B-Root's goal in measuring anycast is to understand routing choices; we return to this question in Section 6.6.1.

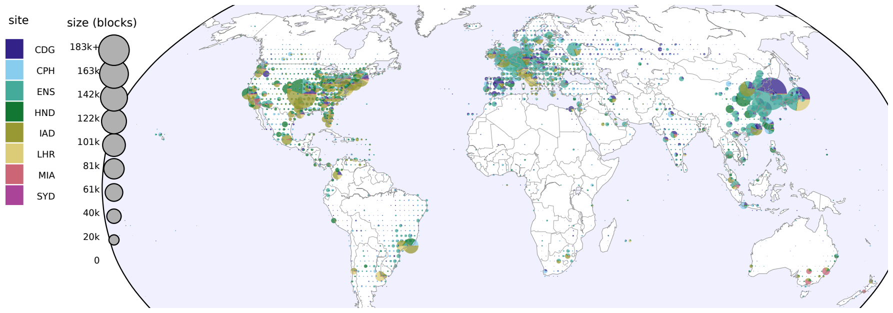
6.5.2 Utility in Mapping Multi-Site Anycast

B-Root shows the benefits of increased number of VPs with Verfploeter, but we would like to understand how the different approaches work on anycast deployments with more sites. We therefore turn to Tangled: an anycast testbed designed and deployed by us (Section 6.4.2).

Figure 6.3 maps the catchments of Tangled with Atlas and Verfploeter. Again, outside of Europe, the greater density of coverage of Verfploeter provides clear qualitative differences between the two maps. For example, the IAD site (dark yellow) shows up prominently across North America with Verfploeter, but with Atlas, CDG and ENS seem to serve that region. We also see very



(a) RIPE Atlas coverage of Tangled (Dataset: *STA-2-01*).



(b) Verfploeter coverage of our nine-site testbed (Dataset: *STV-2-01*).

Figure 6.3: Catchments for Tangled from RIPE Atlas and Verfploeter. Circle areas show number of blocks (Verfploeter) or VPs (Atlas) at different scales; each is a pie chart with colors showing each site.

different mixes of sites in Australia. And only Verfploeter provides coverage of China.

The key result from these graphs is that *Verfploeter coverage tracks the Internet as a whole*, not just where physical VPs can be placed. We quantify this difference in the next section.

6.5.3 Greater Coverage in Verfploeter

In Section 6.5.1 and Section 6.5.2 we showed how the greater coverage in Verfploeter reveals aspects of B-Root and our testbed Tangled that would otherwise be missed. This coverage is possible because Verfploeter's passive VPs only require a computer that responds to ICMP, instead of physically deployed devices (Figure 6.1); this way we can cover millions of /24s.

To quantify the difference in coverage that is visible in Figure 6.2, Table 6.4 compares how many blocks the two measurement approaches see. For both systems we try to use all available VPs, but some VPs are unavailable: for

	RIPE Atlas		Verfploeter
	(VPs)	(/24s)	(/24s)
considered	9,807	9,083	6,877,175
non-responding	455	406	3,090,268
responding	9,352	8,677	3,786,907
no location	0	0	678
geolocatable	9,352	8,677	3,786,229
unique		2,079	3,606,300

Table 6.4: Coverage of B-Root from the perspective of the RIPE Atlas and Verfploeter measurement systems, measured in VPs (Atlas) or /24 blocks (both). (Datasets: *SBA-5-15*, *SBV-5-15*)

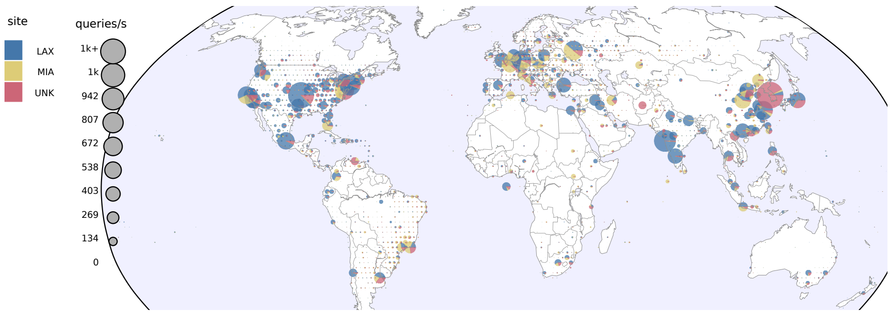
Atlas, 455 VPs do not respond (within 406 blocks), presumably because they are temporarily down. For Verfploeter, about 3M ping targets do not reply, presumably because the target was temporarily down, or it was in a block of dynamic addresses and temporarily unused. If desired, both of these non-response rates could be reduced by retrying later, or with additional addresses for Verfploeter. All Atlas VPs have geolocation (set when the VP is registered), but we discard a few Verfploeter blocks (678) that we cannot geolocate.

The key result about coverage is that Verfploeter *sees around* 430× *more blocks than Atlas.* Although Atlas finds a number of unique blocks (presumably blocks that discard all pings), about 77% of Atlas blocks are also seen by Verfploeter, and Verfploeter sees around 3.61M additional blocks.

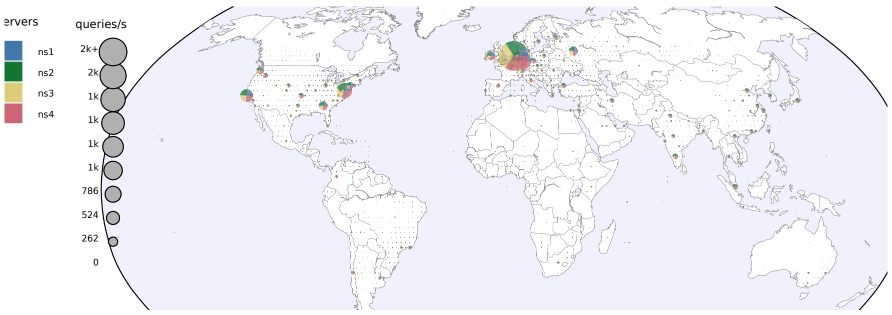
6.5.4 From Observations to Load

We next look at how well different measurement systems relate to actual load on an anycast service. It is well known that the distribution of RIPE Atlas reflects more about who RIPE interacts with than global Internet traffic—as an European project, and Europe being the main region of RIPE NCC operation, Atlas' deployment is far greater in Europe than in other parts of the globe (this is a well known shortcoming [104]). Our goal here is to *calibrate* different measurement systems to best match actual traffic. We show that, once calibrated, we can get very accurate predictions about expected service load, but the calibration is necessary to account for variation in load per block. Calibrated predictions are important if Verfploeter is to be used for capacity planning.

Estimating Load: To estimate load on B-Root, we begin with our prediction about anycast catchments from Verfploeter, then we weight each /24 block by our measurements of its known traffic load (Section 6.3.2). There are blocks for which we do not have anycast mapping, either because they do not reply to our probes, or because the specific address we chose to contact did not reply; these blocks are mapped to "unknown", indicating we cannot determine



(a) Geographic distribution of load by site for B-Root, as inferred from Verfploeter (Datasets: *LB-4-12*).



(b) Geographic distribution of load for `.nl`, as determined by traffic logs (Dataset: *LN-4-12*).

Figure 6.4: Measured DNS traffic over geography for B-Root and `.nl`.

the anycast mapping. (Although we assume their traffic will go to our sites in similar proportion to blocks in known catchments.)

Figure 6.4a shows the result of this load prediction. It is useful to compare this estimate to Figure 6.2b, which counts /24 blocks that source traffic, and Figure 6.2a, which counts Atlas VPs.

The most striking operational difference between measurements of blocks and actual load estimates is that load seems to *concentrate* traffic in fewer hotspots. This outcome should not be surprising: DNS is a common service operated by most ISPs with a local recursive resolver. Thus an ISP with users spread over a large region may still send all DNS traffic through recursive resolvers housed at a few data centers. Weighting coverage by load corrects for these protocol-specific effects that are not seen directly in our ICMP-based measurements.

Second, Verfploeter can only map blocks that respond to our probes. Table 6.5 shows coverage as seen from B-Root's traffic logs, showing that there are a large number of blocks (about 12.9%) that are not mapped. Figure 6.4a plots the load from these blocks in red, showing that most are in Korea, with

	Blocks		Queries	
	/24s	%	q/day	%
seen at B-Root	1,388,338	100%	2.19G	100%
mapped by Verfploeter	986,605	87.1%	1.80G	82.4%
not mappable	401,733	12.9%	384M	17.6%

Table 6.5: Coverage of Verfploeter from B-Root. (Dataset: *SBV-5-15*, *LB-5-15*.)

Date	**Method**	**Measurement**		**% LAX**
2017-04-21	Atlas	967	VPs	68.8%
2017-05-15		9,682		82.4%
2017-04-21	Verf-	4.069M	/24s	82.4%
2017-05-15	ploeter	3.923M		87.8%
2017-05-15	+ *load*	n/a	q/day	*81.6%*
2017-05-15	*Act. Load*	*2.188G*	q/day	*81.4%*

Table 6.6: Quantifying differences B-Root anycast with different measurement methods and times.

some in Japan and central and southeast Asia. In Section 6.5.5 we show that these missing blocks do not alter our predictions.

Finally, we see that load is higher in some regions than the number of blocks would suggest, particularly in India. This difference may be explained by many users using relatively few IP blocks in these areas, with a great deal of deployed network address translation behind those blocks.

Quantifying Differences from VPs to Blocks to Load: While Figure 6.2a and Figure 6.2b show visual differences, we turn to Table 6.6 to quantify those differences and their impact on assessment of catchment sizes in B-Root. When we compare Atlas, Verfploeter, and Verfploeter with load, we see very different measurements (thousands of VPs, millions of blocks, or billions of queries per day). Load estimates (Section 6.3.2) determine different weighting factors and result in different fractions of traffic between the LAX and MIA sites, as shown in the "% LAX" column. In Section 6.5.5 we will compare these values to measured load to see which is most accurate, but next we see how these changes will be even larger for DNS services with less even global load.

Uneven Load: Load for B-Root is global and largely follows the distribution of Internet users, so Figure 6.4a has only moderate differences from Figure 6.2b.

Other DNS systems are more regional. Figure 6.4b shows load for four of the `.nl` nameservers, the country domain for the Netherlands. They (SIDN, the operator of the .nl Top-Level Domain (TLD)) cannot easily collect data from their two nameservers that use anycast, so data from those nameservers are omitted from this plot and it may under-represent global traffic, however, it captures at least half of all global traffic to this domain.

Unlike B-Root, we see that the majority of traffic to `.nl` is from Europe and the Netherlands. There is also significant traffic from the U.S. and some global traffic. With this type of client distribution, calibrating the measured catchment using load information is critical.

6.5.5 Using Verfploeter to Predict Load

We next examine how accurate Verfploeter's load modeling can *predict future load*. Our goal is to determine to what extent unmappable blocks (Section 6.5.4) affect accuracy, and how much routing and load shifts over time. In both cases we observe partial information and predict load for the unobserved remainder (observing responses per blocks and predicting load, or observing load now and predicting future load), then compare that against complete information. A study of long-term predictions will require more experience with Verfploeter, we focus on the short-term accuracy in this section.

We study the accuracy of load predictions with Verfploeter by analyzing what network blocks B-Root sees traffic from that Verfploeter has found to be unmappable by examining the DNS network load at B-Root on 2017-05-15 (Dataset: *LB-5-15*) and the Verfploeter analysis performed on the same day (Dataset: *SBV-5-15*). (Since Tangled is not a production service, we cannot study its operational load.) Recall from Table 6.6 that although Verfploeter finds 87.8% of network blocks reach LAX, the load prediction is that 81.6% of traffic should go to LAX. That prediction does not consider blocks that send traffic to B-Root but do not respond to Verfploeter (12.9% from Table 6.5).

Predicted vs. Measured Load: The last line of Table 6.6 shows the *actual* load of 81.4%, as measured at all B-Root sites on 2017-05-15. We see our 81.6% prediction using same-day Verfploeter and load is quite close to the measured result. Our first observation is that this result suggests *Verfploeter-unobservable blocks do* not *have significant effects* on our overall load estimate. (Future work could strengthen this claim by demonstrating it for services other than B-Root.) Although they account for 17.6% of queries (Table 6.5, and the red slices in Figure 6.4a), the fraction of traffic that goes to each B-Root site appears to follow the ratio seen in measured blocks.

Our second observation is that *our load-weighted predictions are very close to observed load.* Verfploeter without load adjustment is further off, with 87.8% of blocks going to LAX. We conclude that weighting by load is important. Surprisingly, Atlas estimates, at 82.4%, are actually closer than Verfploeter if Verfploeter is not load-weighted. It is likely that the reason for this is that the RIPE Atlas probe distribution more closely resembles a realistic client base for B-root.

The key take-away of this result is that with load-weighted Verfploeter *preliminary results suggest it is possible to make reasonable predictions about* *future anycast deployments* by measuring the deployment on a test network and predicting future traffic levels using recent load data. We hope to expand these results beyond B-Root as ongoing work.

Long-duration predictions: Finally, we can also look at long-duration prediction. We performed a similar prediction analysis in advance of the B-Root deployment using the Verfploeter data gathered on 2017-04-21 and network traffic from 2017-04-12. We see a fairly large shift in blocks between these dates, with Verfploeter shifting from 82.4% to LAX in April to 87.8% in May. By weighting the *SBV-4-21* Verfploeter dataset from the B-Root test prefix with the *LB-4-12* measured load, we find that the predicted DNS request load arriving at LAX is 76.2%. This is significantly less than the 81.6% measured load in *LB-5-15*, which highlights the discrepancy between shifts in routing over one month between the *SBV-4-21* and *SBV-5-15* dataset collection periods.

This shift suggests that the accuracy of load estimates depends on how old the data is. We know that routing changes in the Internet over time [167]; this early result suggests some care must be taking with long-duration predictions. We expect that predictions further into the future will be less accurate than short-term predictions. While we are collecting data to answer this question, such a study is future work.

6.6 Understanding Anycast with Verfploeter

We next use Verfploeter to explore three questions about anycast. These questions have each been raised in prior work; here we use Verfploeter to revisit them (and compare to them, in Section 6.6.1, Section 6.6.2 and Section 6.6.3), both to show its utility and to refine these prior results.

6.6.1 Use of AS Prepending in B-Root

An important operational question for B-Root is understanding how to balance load between sites. Although both sites are able to handle normal traffic, DNS operators need to shift load during emergencies, e.g. for DDoS attacks that can be absorbed using multiple sites [3]. Operators may also want to control load during regular operation, perhaps because different sites have cost structures that are traffic-sensitive.

We used RIPE Atlas and Verfploeter to investigate the use of AS Prepending to adjust the catchment of a test prefix on B's sites. AS Prepending is a traffic engineering approach where an operator increases the BGP path length at one site to make that route less desirable than other routes with shorter AS paths [113]. Figure 6.5 shows how the distribution changes as AS prepending is applied between the two sites, as measured with both methods. (Note that the units for each measurement are different: RIPE Atlas is measured in VPs, and Verfploeter is measured in /24 blocks.) By default, with no prepending, 74%

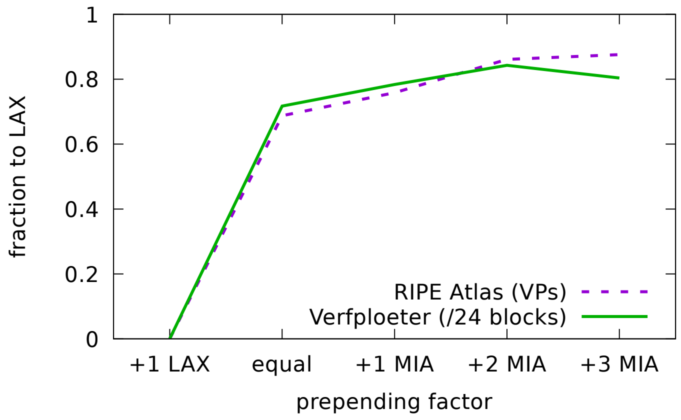


Figure 6.5: Split between MIA and LAX in VPs for Atlas and /24s for Verfploeter. (Dataset: *SBA-4-20*, *SBA-4-21*, *SBV-4-21*.)

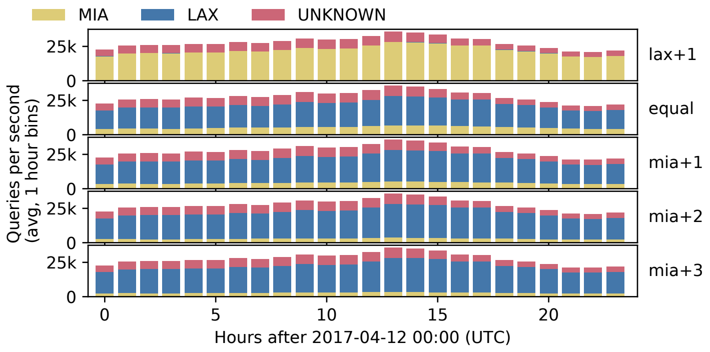


Figure 6.6: Predicted load for B-Root with multiple AS prepending combinations; catchment data from Verfploeter with load (Datasets: *SBV-4-21*, *LB-4-12*).

of Atlas VPs arrive at LAX, while Verfploeter shows that 78% of responsive /24 prefixes will arrive at LAX.

These results show that both measurement systems are useful to evaluate routing options. With only two sites, either measurement method seems sufficient for rough analysis. We expect the greater precision of Verfploeter will be important with more sites, and to assist with the trial-and-error process required when deploying more subtle methods of route control (for example, use of BGP communities traffic [113]).

We next study how load shifts at different prepending values over the course of a day. For this study we measure load over 24 hours, summarizing it per hour, then combine that with measured values from five different prepending configurations (each taken once on a different day). Figure 6.6 shows this combination using catchment data from Verfploeter combined with DITL data of B-Root (2017-04-12). In the top graph, nearly all traffic goes to the MIA site, since LAX's BGP announcement includes an "AS prepending" of one (and the small share of load, "UNKNOWN", that is not mappable by Verfploeter). When LAX and MIA announce routes without prepending, most of the traffic load shifts to LAX (second graph from top-down). The last three graphs show the results of prepending MIA's BGP announcement by up to 3 times, resulting in an increasing traffic share shifting to LAX. However, even by announcing our prefix with 3 times our AS at MIA (MIA+3), we still see a small fraction of traffic being mapped to MIA. These few networks are likely either customers of MIA's ISP, or perhaps ASes that choose to ignore prepending.

6.6.2 Discovering Divisions Within ASes

Prior work (particularly anycast studies using RIPE Atlas) often assumed that anycast catchments align with ASes, thus one VP can represent where the load of the entire AS goes. While generally true for smaller ASes, this assumption is less likely to hold for large, multi-national ASes where different parts of the AS may be served by different anycast sites. Such large ASes are likely to have geographically distributed peering locations and so may prefer to direct some of their users to different anycast sites to reduce service latency.

The high density of VPs in Verfploeter allows us to test this assumption by looking for differences in anycast catchments that occur *within* individual ASes. For this measurement, we use the Tangled testbed. We first remove those VPs from the dataset that show instability (see Section 6.6.3), to prevent unstable routing from being classified as a division within the AS. Without removing these VPs we observe approximately 2% more divisions (*e.g.,* ASes which are served by more than one site). We count the number of sites that are seen (from different VPs) within a single AS, in a single measurement round.

In total, we see multiple sites from 7,188 ASes, or approximately 12.7% of all ASes that were announcing at least a single prefix at the time of the measurement. Note that this is a lower-bound, using a larger and/or more diverse anycast service we might be able to determine a higher, and more accurate, percentage of ASes that are split into multiple individually routed parts.

Routing policies (such as hot-potato routing [16]) and physically distributed ASes are a likely cause for these divisions. And, as routing on the Internet is largely determined by BGP, we show the number of prefixes that are announced via BGP by an AS versus the number of sites that it sees in Figure 6.7. Indeed,

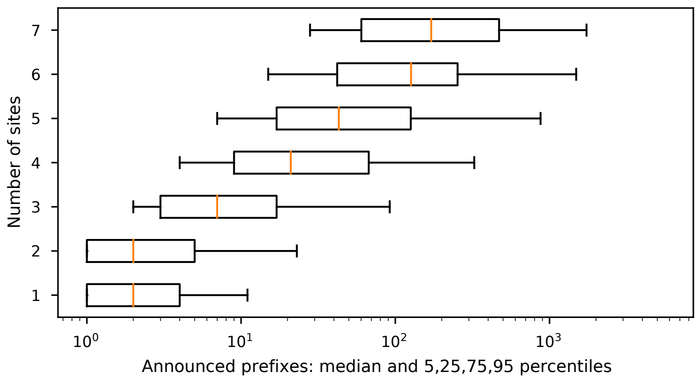


Figure 6.7: The number of sites that are seen from an AS versus the median amount of prefixes that are announced by those ASes. (Dataset: *STV-3-23*.)

those ASes that announce more prefixes tend to see a higher amount of sites from their network.

In Figure 6.8 we show the number of sites that are seen from announced prefixes, grouped by prefix length. We can see that VPs in prefixes shorter than a /11 are mapped to more than a single site in the majority of cases. About 20% of the routed prefixes have VPs that are routed to more than one site and thus require multiple VPs to accurately map. Larger prefixes are often divided even further—75% of prefixes larger than /10s see more than two sites and require multiple VPs. Although 20% might seem like a relatively small percentage, given the size of these prefixes this means that multiple VPs are required in prefixes that account for approximately 38% of the total measured address space.

These results show that, in order to get a complete view of the catchment, in many cases you need *more* than a single VP per AS. While the quantitative results in this chapter are specific to B-Root and Tangled, this qualitative result (ASes can be subdivided) applies more generally. Measurements from platforms with fewer VPs often assume that each VP can represent its AS, but likely lose precision in large ASes.

6.6.3 Stability of Anycast for Clients

A long-term concern with anycast is how *stable* the association of an anycast client is with its site [103]. Since Transmission Control Protocol (TCP) connections require shared state at both ends, if users switch anycast sites within the lifetime of a TCP connection, that connection will break and need to be restarted. The existence of multiple successful CDNs that use IP anycast (including Bing, Edgecast, and Cloudflare) suggest that anycast is almost always

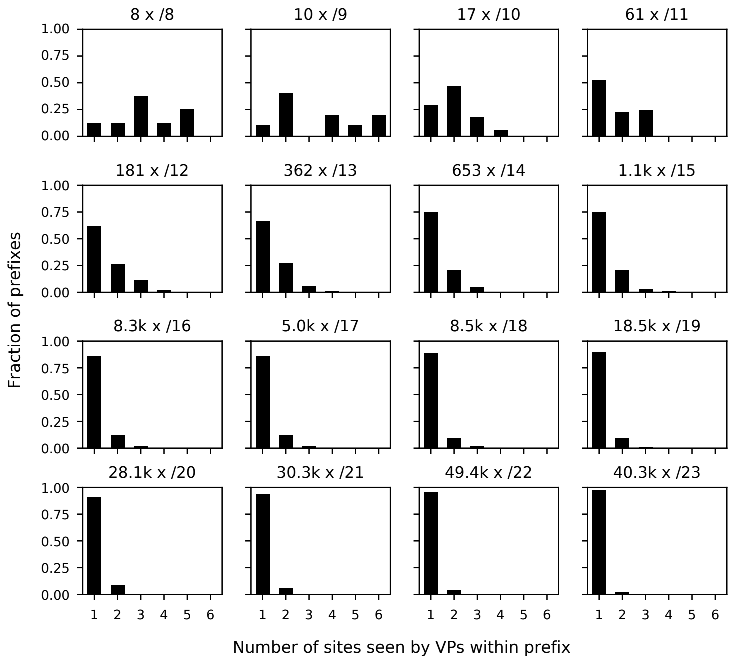


Figure 6.8: The number of sites that are seen for each prefix as announced in BGP. (Dataset: *STV-3-23*.)

stable, but recent work has suggested that anycast may be persistently unstable for a tiny fraction of (user, service) combinations (less than 1%) [103]. From the viewpoint of a service operator, it is interesting to know if a single measurement can be representative for a longer time, or if the catchment is continuously in flux.

Verfploeter allows us to revisit this question from Tangled to many VPs. We measured the global catchment of our testbed every 15 minutes for a day (96 observations). Considering the short-lived nature of many TCP connections this interval might be too long to detect rapid fluctuations, however, it is enough to give an impression of the overall stability of catchments. We categorize the responses (or non-responses) into 4 groups: *stable*, VPs that maintain the same catchment across measurements; *flipped*, VPs that change catchment, with responses sent to a different anycast site than the prior measurement; *to-NR*, VPs that switched to "not responding" in the current measurement; and *from-NR*, VPs that started responding in the current measurement. We do not count VPs that remain non-responsive after being counted as *to-NR*.

Figure 6.9 shows the results of one day of these measurement. Because the fractions of stable and flipping are so different, we break the graph into three sections. We see that the *catchment is very stable* across the measurement

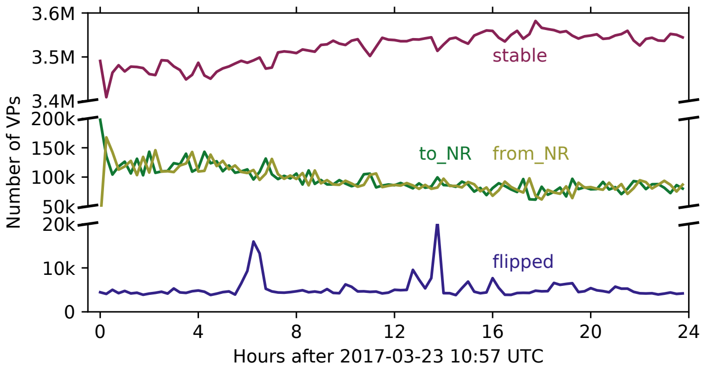


Figure 6.9: Stability over 24 hours. One data point per 15 minutes. If a VP stopped responding then it counts as to_NR, if it started responding it counts as from_NR. (Dataset: *STV-3-23*.)

	AS		IPs (/24s)	Flips	Frac.
1	4134	CHINANET	47,963	257,915	0.51
2	7922	COMCAST	3,933	19,133	0.04
3	6983	ITCDELTA	1,372	15,403	0.03
4	6739	ONO-AS	849	13,347	0.03
5	37963	ALIBABA	2,493	10,988	0.02
		Other	43,388	188,630	0.37
		Total	108,493	505,416	1.00

Table 6.7: Top ASes involved in site flips. (Dataset: *STV-3-23*.)

rounds, with a median of 3.54M (about 95% of the 3.71M that respond) VPs always replying and maintaining their prior catchment. The fraction of VPs that fluctuate between responsive and non-responsive states is small across all 96 measurements. A median of 89k (about 2.4%) VPs changed from responsive to non-responsive between measurements, and about the same number flipping back. Note that fluctuating and flipping VPs are not necessarily always the same ones, however, the fact that the from_NR line mostly follows the to_NR line indicates that most are.

Across the measurement period, we also see a median of 4.6k (about 0.1%) VPs change catchment (the blue line in Figure 6.9). All these VPs are located within 2809 ASes. Table 6.7 shows that 63% of the flipping VPs are part of only 5 ASes; and 51% are within AS 4134 (Chinanet). Catchment flips can be caused by changes in routing policies or link state, and frequent flipping can be caused by load balanced links. With flipping prominent in only a few ASes, these observations confirm prior observations taken with RIPE Atlas in Chapter 5 as well as Wei et al. [103], but from a larger set of vantage points:

that anycast instability is very rare, but as a property of certain ASes, it will be persistent for users of those ASes. An additional application of Verfploeter may be identification and resolution of such instability.

6.7 Concluding Remarks

The key takeaway of this chapter is to show that Verfploeter allows measurements of anycast catchments across millions of networks in the Internet. Verfploeter allows us to see 430× more network blocks than RIPE Atlas, a widely used, large-scale platform for active measurements.

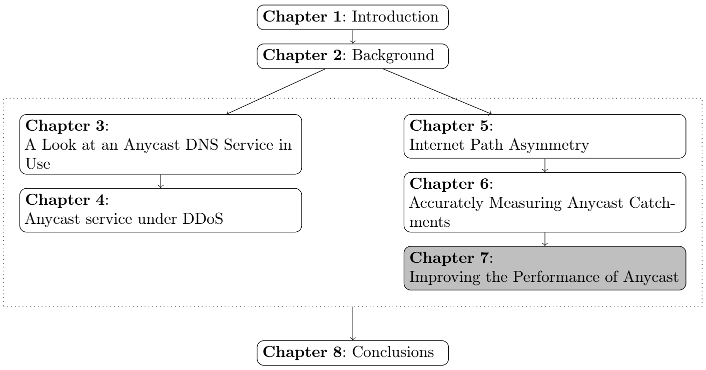
Such measurements are important for operating anycast services (see Section 6.5.1), and become more important as anycast services grow in number of sites (see Section 6.5.2). With large DNS and CDN anycast networks using hundreds or thousands of sites, catchment mapping with broad coverage (Section 6.5.3) is increasingly important, particularly since regular catchment evaluation is necessary to avoid performance errors [167], [102].

Furthermore, the combination of historic traffic load and catchment mapping (Section 6.5.4) can provide a predictive tool for anycast operation (Section 6.5.5). The broad coverage of Verfploeter allows us to identify individual networks that are very likely to be the source of larger amounts of traffic.

We have used Verfploeter to understand the new B-Root anycast system (Section 6.6.1), evaluate split catchments in large ASes (Section 6.6.2), and confirm prior results in anycast stability with a larger dataset (Section 6.6.3).

CHAPTER 7

Improving the Performance of Anycast


Chapter 1: Introduction
Chapter 2: Background
Chapter 3: A Look at an Anycast DNS Service in Use
Chapter 4: Anycast service under DDoS
Chapter 5: Internet Path Asymmetry
Chapter 6: Accurately Measuring Anycast Catchments
Chapter 7: Improving the Performance of Anycast
Chapter 8: Conclusions

In the previous chapter we have shown that the Verfploeter methodology can be used to create an accurate map of anycast catchments. We validated the methodology by deploying it on a small testbed and the B root Domain Name System (DNS). In this chapter, we take the next step and deploy Verfploeter on one of the world's largest anycast networks, the Cloudflare Content Delivery Network (CDN) with 192 Points-of-Presence (PoPs) worldwide. We perform three real-world case studies on network planning (what happens when PoPs are switched on or off), troubleshooting (reachability issues of an anycasted prefix) and security (detecting spoofed attack traffic). Using these three case studies, we show that Verfploeter is highly suitable for such a large-scale operation and gives operators vital insights that allow them to improve network management practices of their anycast service. The work in this chapter will be published as a research paper [114].

7.1 Introduction

In Chapter 6 we introduced a novel methodology to measure the catchments (i.e. which client will be served by which site) of anycast services, called “Verfploeter”. Key advantage of this methodology is that it does not require any external Vantage Points (VPs) such as RIPE Atlas probes, but instead relies on Internet Control Message Protocol (ICMP)-responsive Internet hosts. By sending ICMP Echo Requests to many hosts on the Internet, and collecting the responses, we can accurately establish the catchment of a service for the full IPv4 Internet, or a part thereof. Unlike an approach based on external vantage points, Verfploeter does not suffer from bias due to the distribution of these points.

We have demonstrated how Verfploeter performs from a deployment on a testbed, and, on a limited scale, the B root DNS server (which has just three anycast sites, as of March 2019). In contrast, in this chapter we describe a global-scale deployment in one of the worlds largest anycast CDNs. We discuss the challenges of deploying Verfploeter in an anycast network of this scale (with 192 global points-of-presence). Then, we show how Verfploeter can help large-scale anycast operators manage their network through three use cases:

Firstly, we show how Verfploeter’s detailed catchment information helps manage changes in the configuration of the active sites of an anycast service. For example, what would happen if large site A is taken down, in terms of the shift in clients to other sites. We argue that this is important since depending on the shift of traffic, one or more of the other sites might attract traffic exceeding its maximum capacity. This type of analysis is also particularly useful for planned maintenance.

Secondly, we show how Verfploeter can be used to regain traditional ICMP-based troubleshooting capabilities. For example, traditionally connectivity issues are confirmed using `ping`, i.e. by sending an ICMP Echo Request packet. However, in the case of anycast, the response to this packet will likely end up in a different location. From the viewpoint of the sender of the request packet this would appear as a timeout. Using Verfploeter these packets are matched regardless of the location where it is received, in essence allowing an asymmetric ping.

Lastly, we show how Verfploeter can be used to detect spoofed traffic, by matching the known ingress location of traffic from a specific client using high resolution catchment data. Essentially we use Verfploeter to establish ground truth on client-to-anycast mappings, and mark traffic (supposedly) from that client that ends up in a different anycast site as likely to be spoofed. This mark then allows operators to filter or rate-limited traffic to mitigate the effects of spoofed Distributed Denial-of-Service (DDoS) attacks.

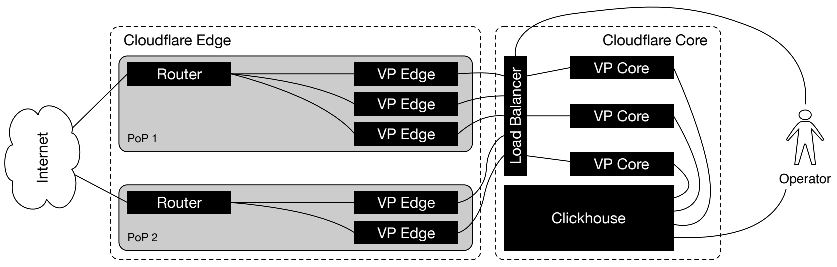


Figure 7.1: Setup for Verfploeter at Cloudflare

7.2 Operational implementation

In the previous chapter in which we introduced Verfploeter (Chapter 6), we discussed how to deploy Verfploeter on a limited scale (an anycast testbed and DNS B Root, which has three locations as of March 2019). In this chapter our focus shifts to how we can deploy Verfploeter on a massive scale, on Cloudflare's anycast network. As of August 2019, Cloudflare has 192 Points-of-Presence worldwide, with over 30 Tbps of aggregate capacity [136]. It is used to deliver many services which includes Public DNS and Authoritative DNS, as well as a CDN and to provide DDoS mitigation. They announce more than 700 prefixes, covering roughly 1.5M IPv4 addresses [193].

In this section we discuss the challenges this creates and how we tackle them. In particular we discuss **(a)** the goal and requirements of the system, **(b)** the design choices we made when implementing, with regard to the requirements and goal and **(c)** the final architecture as it is currently in use.

Goal and requirements: Our goal for this implementation is to set up a measurement infrastructure that can take *snapshots* of the anycast catchment of a given prefix, on-demand as well as scheduled, for the entirety of the global deployment. The system should deliver the results in an accessible manner such that the results can be manually or automatically analyzed. To reach this goal, we defined the following requirements:

R1 **Service wide measurements should be trivial to start**: to maximize the utility of this system, it should be trivial for operators to start a measurement, without having to consider the large scale and distributed nature of the underlying service.

R2 **Incoming packets that are part of the measurement should be authenticated**: to prevent third parties from injecting results into the system without authorization, incoming packets should be authenticated.

R3 **Authenticated results should be inserted into a data store for analysis**: to allow both manual as well as automated analysis, data should be stored in a suitable data store.

R4 **Communication between system components should be reliable and secure**: given that the system will be running spread over many PoPs across the Internet, communication should be secure, and as reliable as possible to increase the accuracy of the results.

Design choices: Considering the goal and requirements we now discuss major design choices we made while developing the system below:

Centralized control: in order to meet **R1**, it is impractical to have to contact each of the over 190 sites to start a measurement. We therefore chose a design where there is a central component (called *VP-core*) that manages the measurements, which the systems in the anycast sites connect to, to poll for jobs (this component is called *VP-edge*). Alternatively, the connection could be setup the other way around, but that would require the centralized component to have full knowledge over what systems are active on the edge sites, while that knowledge exists, it increases the complexity considerably.

Authentication: to prevent spoofed packets, and meet **R2**, we add a Hash-based Message Authentication Code (HMAC) in the ping payload. This ensures that received packets are in response to packets that were sent by the system, and not randomly generated by a third-party. Additionally, the payload contains the original target IP address, as well as the transmission timestamp. This information can then be used to filter out invalid responses, e.g. those that are late, duplicated or otherwise not authentic.

Robustness: To meet **R3**, each component buffers data such that connection or transmission failures do not result in loss of data. The core component is suitable to run in Kubernetes (K8s) [194], which automatically restarts processes upon failure, possibly on a different node of the K8s cluster, as well as facilitating horizontal scaling. In addition, data is buffered between the database and the core component by using a robust message bus, Apache Kafka [195].

Secure communication: Transport Layer Security (TLS) is used throughout the system for communication, to meet **R4**, in combination with gRPC [196]. Edge components are persistently connected to the core components.

Architecture: Verfploeter is a distributed system, written in Go, which is schematically depicted in Figure 7.1. It consists of several components, the most important two of which are shown. The system must have, due to the nature of anycast, a component that runs on all nodes within the anycast service that potentially receive traffic on an anycast prefix. This component is referred to as *VP-edge*. The *VP-core* component on the other hand runs centralized, but is replicated horizontally for performance reasons.

The VP-edge component is responsible for sending and/or receiving the ICMP Echo Requests and Replies. In any given PoP there can be thousands of instances of this component. Control of it is exercised via the central VP-cores. Received responses are collected and transmitted, batch-wise, to the VP-cores. Authentication and validation of the incoming data occurs at the edge, such that only valid data is collected at the VP-core.

The VP-core component has four functions:

- Controlling the VP-edge outbound functionality.
- Collecting the results from the VP-edges and inserting them into an Apache Kafka messagebus, for later insertion into the central data warehouse application (Clickhouse).
- Providing an Application Programming Interface (API) for external users.

The VP-cli component (not shown) is used to initiate a measurement. It allows various command line options, e.g. to indicate the source address to use (e.g. the anycast prefix), which target addresses to use and which systems to initiate the measurement from. Results become available in the central data warehouse (based on Clickhouse[1]) when the measurement completes, or are forwarded to the command line client directly.

7.3 Use cases

The system we discussed in Section 7.2 was implemented as a production service in Cloudflare's anycast network. To show how Verfploeter supports better management of anycast services, we now present three real-world use cases of how Verfploeter can be used, and demonstrate them on Cloudflare's network. These range from planning, to troubleshooting, as well as securing networks.

7.3.1 Planning

The primary reason to implement and use Verfploeter is for planning purposes. What consequences, in terms of anycast catchments, does taking a particular PoP offline have. There are several reasons why a PoP might be taken offline, for example: for scheduled or unscheduled maintenance, or the PoP is overloaded and some traffic needs to be shifted to different PoPs.

Due to the nature of anycast and the opaqueness of the Internet it is hard to predict where traffic will be rerouted in case a prefix is withdrawn. This might cause difficulties, for example in case the traffic will be rerouted to a different PoP that might not be able to handle the added load.

For this section we have performed 165 measurements (for operational reasons not all PoPs were included), which we show in Table 7.1. In each of these measurements a different PoP (and in two cases two PoPs) is taken offline, and we send ICMP Echo Requests towards approximately 6 million targets (see Section 6.3.1). The number of results varies between measurements, but typically lies around 3.5M, meaning approximately 55% of the requests resulted

[1]A column-oriented data warehouse system developed by Yandex, `https://clickhouse.yandex`

#	PoP(s) offline	Count	Response fraction
P0	*None*	3.49M	0.57
P1	`AMS`	3.44M	0.56
P2	`LHR`	3.30M	0.54
P3	`CDG`	3.42M	0.56
P4	`AMS`, `LHR`	3.45M	0.56
P5	`AMS`, `CDG`	3.50M	0.57
P6	one per measurement 159 measurements different PoP each measurement	≈3.5M	≈0.55

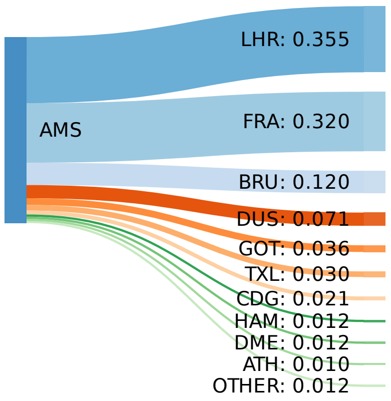
Table 7.1: Measurements with zero or more PoPs taken offline. Count is the number of unique responses we see. Response fraction is the fraction of requests for which a response is received.

in a response. We use P0 as a baseline measurement to determine which IPs are associated with each of the PoPs, and use that to track where these IPs move to in the subsequent measurements. Note that by *taken offline* we refer to withdrawing the announcement of a test prefix, production traffic to the anycast network is unaffected by these measurements.

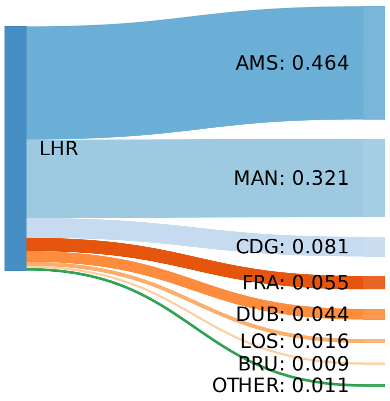
Consider the case where a single PoP is taken offline. By using two measurements, one in the *normal* state, and one in a state where that PoP is down, using a test prefix, we can show how prefixes will be rerouted. In Figure 7.2a we show what happens if the PoP in Amsterdam (AMS) is taken down, note that the different colors have no purpose or meaning other than to make the lines easier to distinguish. Interestingly, the `OTHER` category includes 20 additional PoPs, each receiving a small number of rerouted prefixes. In the case of `AMS` the majority of traffic is redirected to `LHR` and `FRA`, which is expected. However, we would also expect `CDG`, another large PoP relatively close to `AMS` and `LHR`, to rank high, but the measurements show that this is not the case.

Complexity increases when taking multiple PoPs offline. For example, consider Figure 7.2b, here, instead of `AMS`, London (`LHR`) is taken offline. We can see that in that case the majority of prefixes would reroute to `AMS`. However, what happens if `AMS` *and* `LHR` are taken offline. We show a measurement of this scenario in Figure 7.3b. We point out that there appears to be a strong topological *dependency* between LHR and AMS, meaning that when LHR goes down a large fraction of traffic goes to AMS and vice versa. The consequence of that is that calculating what happens when both are taken offline based on the individual measurements we performed in Figure 7.2a and Figure 7.2b leads to a high degree of uncertainty of 42%, see X in Figure 7.3a.

In contrast, `AMS` and `CDG` (Paris) have a much lower *dependency* on each other. In Figure 7.4 we show that the calculated (Figure 7.4a) and the measured (Figure 7.4b) route changes are very similar. In practice, this means that PoPs which have a low *inter-dependency* do not require additional measurements

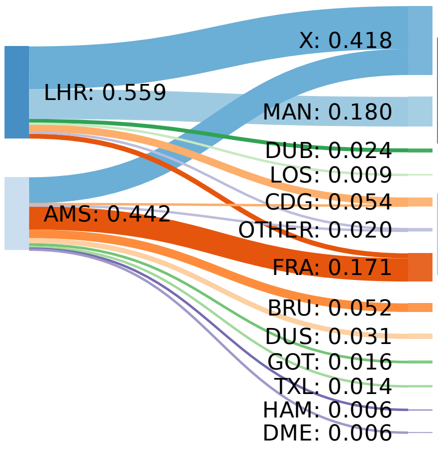


(a) AMS. Dataset: P0, P1.

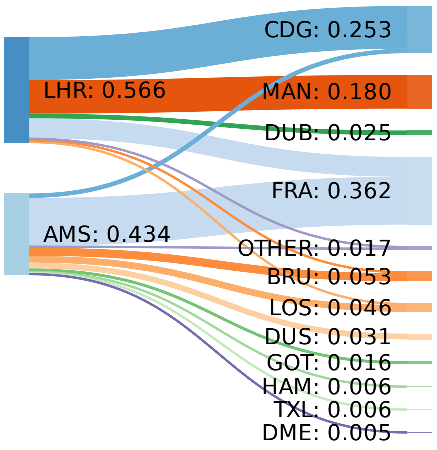


(b) LHR. Dataset: P0, P2.

Figure 7.2: Rerouting of prefixes that occurs when a single PoP goes down.



(a) Calculated. Datasets: P0, P1, P2.

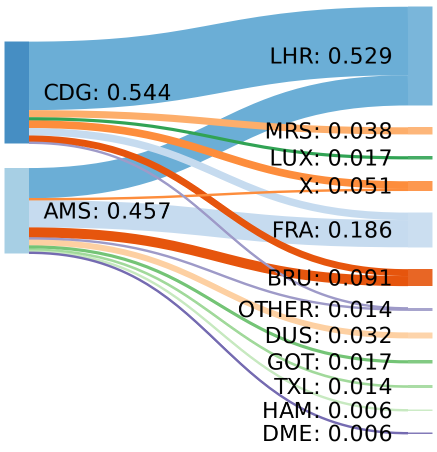


(b) Measured. Dataset: P0, P4.

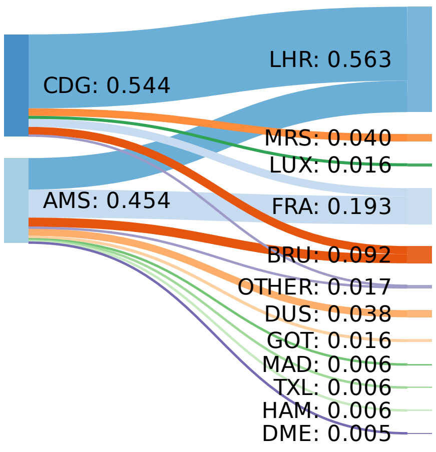
Figure 7.3: Rerouting of prefixes with two PoPs down: AMS & LHR.

when taken offline together, as this can be calculated from the individual measurements.

In Figure 7.5 we show to how many other PoPs prefixes get redistributed in the case the PoP that they were associated to gets taken offline. Overall, most prefixes (>95%) that are associated with one PoP get redistributed to a relatively small number, 3 or fewer, other PoPs. This means that most combinations of PoPs that are taken offline can be calculated from *Single PoP*



(a) Calculated. Datasets: P0, P1, P3.



(b) Measured. Dataset: P0, P5.

Figure 7.4: Rerouting of prefixes with two PoPs down: AMS & CDG.

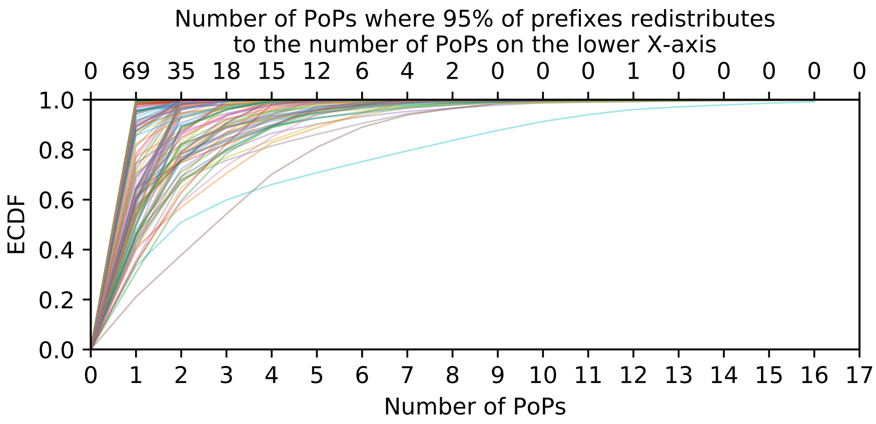


Figure 7.5: Cumulative Distribution Function (CDF) for each measurement, showing to how many PoPs the measured prefixes are redistributed. Datasets: P0, P1, P2, P3, P6.

offline measurements. We do point out that there is a long tail, where for some PoPs the last 5% of prefixes get redistributed to 17 PoPs or more.

We emphasize two key takeaways from this: **a)** taking a PoP down causes prefixes to be redistributed across a limited number of PoPs, albeit with a long tail of small PoPs that take a small number of prefixes. **b)** depending on the collection of PoPs, the redistribution can be calculated based on measurements where only a single PoP is taken down, with a known degree of certainty.

7.3.2 Identifying connectivity issues

One of the problems that arise when deploying an anycast service is that it is typically not possible to use industry standard tools such as `ping` and `traceroute`. The reason for this is that the response will often, depending on the size of the anycast network, be routed to a different system in a different physical location. This makes it hard to troubleshoot connectivity issues. In this section, we show how Verfploeter can be used to troubleshoot networks similarly to how a `ping` is normally used.

On the 1st of April 2018 Cloudflare launched its public DNS service. An interesting aspect of this launch was that it was launched on the *novelty* IP-ranges `1.1.1.0/24` and `1.0.0.0/24`. Obviously, this range has an interesting property in that it is particularly trivial to remember. However, this address range had not before been used for any production service, in fact, several manufacturers and Internet Service Providers (ISPs) appear to have assumed that the range is suitable for internal use. Contexts in which the range was used include captive portals and routers. It is important to note that Internet Assigned Numbers Authority (IANA) has not designated the range `1.0.0.0/8` or any part thereof as reserved [2]. The range is therefore, according to the relevant authorities, suitable for public use. The range was obtained via a research agreement with APNIC[3].

The effect of the use of addresses in `1.0.0.0/8` in examples, captive portals, etc., is that especially the `1.1.1.1` address is not always reachable from edge networks. Typically, a network operator would use pings to test connectivity, which, as mentioned, will not trivially work on an anycast network. However, with Verfploeter we are again able to use ICMP to find connectivity problems. Specifically we used it to find issues with connectivity to `1.1.1.1`, compared with `1.0.0.1` and a third, unrelated IP address, `104.23.98.190` which has no known issues.

We perform measurements from each of the three source addresses, towards approximately 6 million target addresses. To increase the number of results we repeat the measurement once, and combine the results. In Table 7.2 we show the number of responses, as well as the response fraction, for each of the measurements.

In Table 7.3 we show the different combinations of reachability that we encounter. By far the largest group of responders (almost 92%) do so to each of the three source IP addresses. These responders have no issues reaching `1.1.1.1`. The second category consists of responders that only respond for two

[2] `https://www.iana.org/assignments/iana-ipv4-special-registry/iana-ipv4-special-registry.xhtml`

[3] `https://labs.apnic.net/?p=1127`

Source	Count	Response fraction
1.0.0.1	3.47M	0.56
1.0.0.1	3.49M	0.57
1.1.1.1	3.28M	0.53
1.1.1.1	3.28M	0.53
104.23.98.190	3.48M	0.57
104.23.98.190	3.5M	0.57
Combined		
1.0.0.1	3.58M	0.58
1.1.1.1	3.36M	0.55
104.23.98.190	3.59M	0.58

Table 7.2: Measurements from three different source addresses, and combined totals.

IPs	Count	Fraction
1.0.0.1, 1.1.1.1, 104.23.98.190	3,324,062	0.917
1.0.0.1, 104.23.98.190	232,160	0.064
104.23.98.190	18,526	0.005
1.1.1.1, 104.23.98.190	17,508	0.005
1.0.0.1, 1.1.1.1	16,473	0.005
1.0.0.1	8,125	0.002
1.1.1.1	6,707	0.002

Table 7.3: The different combinations of reachability and counts for each

out of three addresses, where the missing address is `1.1.1.1`. This category, while smaller than the first category is still quite substantial with approximately 6% of the addresses falling in this category. The remaining categories are much smaller and while we do not investigate these in depth we mostly attribute them to random noise, e.g. hosts intermittently not responding or packet loss.

Zooming in on the second category, we investigate which Autonomous Systems (ASes) are involved in the unreachability of `1.1.1.1`. Table 7.4 shows

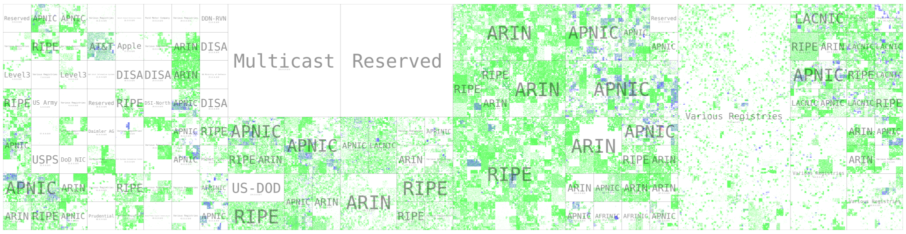


Figure 7.6: Hilbert curve presenting connectivity issues, highlighting places where 1.0.0.1 has no issues, but 1.1.1.1 does in blue, vice versa in red, and where both have no issues in green.)

ASN	AS Name	Count
4837	China Unicom Backbone	70,649
4134	China Telecom Backbone	25,447
3352	Telefónica de España	11,049
7018	AT&T Services, Inc.	9,318
26615	TIM Celular S.A.	8,394

Table 7.4: Top 5 Autonomous System Numbers (ASNs) showing prefixes unable to reach 1.1.1.1, but able to reach 1.0.0.1 and 104.23.98.190

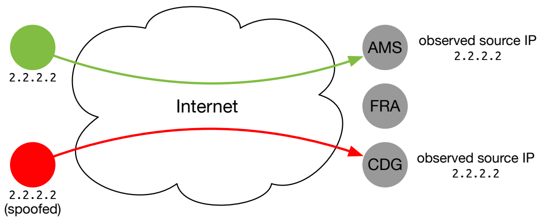


Figure 7.7: Two traffic sources, using the same source IP, one spoofed and one legitimate. The observed IP at the server side is the same, the only difference is the location.

the top 5 ASes involved in `ping` failures. Interestingly China stands out with originating a large part of the failures.

In Figure 7.6 we show a Hilbert curve [115], a way of representing 1-dimensional information in a 2-dimensional image, while maintaining proximity. This curve shows where on the Internet, in terms of IP-space, connectivity issues to `1.1.1.1` and `1.0.0.1` occur. It appears that IP-space managed by APNIC, the Asia-Pacific Regional Internet Registry (RIR), has the highest number of issues, while space managed by RIPE and ARIN is far less problematic.

These results show, how, when operating an anycast network, our methodology can be applied to regain some of the classic methods to troubleshoot a network. As a by-effect, we also show where on the Internet a major public DNS resolver, `1.1.1.1`, might still be inaccessible due to companies or organizations using ranges for purposes outside of what they were assigned for by the relevant authorities.

7.3.3 Securing against spoofed traffic

Spoofed traffic, in which malicious actors falsify source IP addresses in packets, is an ongoing issue for operators. There are initiatives to counter IP spoofing, such as Best Current Practice (BCP) 38, which specifies that network operators should perform ingress filtering to block spoofed traffic. While anti-spoofing techniques are becoming more widespread, there is still a significant amount of spoofed traffic on the Internet [116], [117].

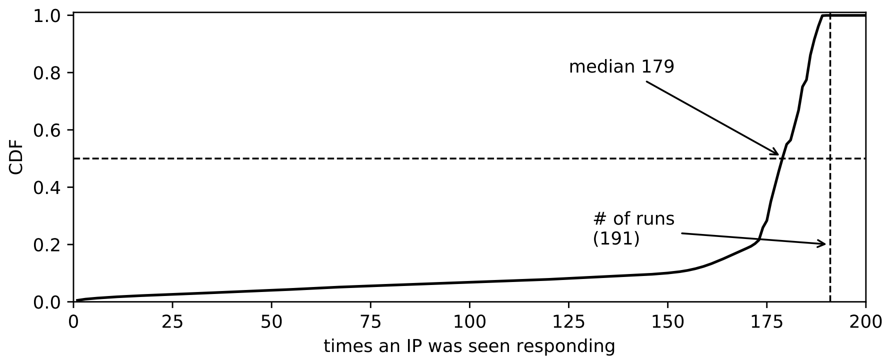


Figure 7.8: Responses per IP

Three important reasons for bad actors to perform IP spoofing when attacking are: **a)** it is inherent to DDoS amplification attacks, where the IP is spoofed to match that of the target of the attack. **b)** to prevent identification of sources of traffic. **c)** to make it harder to filter out traffic.

In this section, we investigate the possibility to detect spoofed traffic by using comprehensive anycast mappings. The principle behind it is that because Internet routes are mostly stable, and traffic from most IP-ranges is routed to the same PoP every time, any traffic from those IP-ranges reaching a different PoP should be considered as *likely spoofed*, as depicted in Figure 7.7.

We investigate two assumptions, namely that specific /24 IP-ranges are consistently routed to the same PoP, as well as that most /24 prefixes will route to the same PoP, regardless of the origin of the Verfploeter measurement. We suspect that the origin of the measurement can have an impact in the case that the target of the ICMP Echo Request, as part of the measurement, is itself an anycast service. We also speculate that there are other reasons for this, such as the ingress point of traffic having some effect on the egress point for some particular networks.

We perform 191 Verfploeter measurements (not all PoPs are available, due to operational reasons), towards the IPv4 hitlist as described in Section 6.3.1, one for each active PoP in the Cloudflare CDN. Due to the unreliable nature of ICMP, combined with the fact that the hosts in the IPv4 hitlist may or may not respond, we expect that we will not gather 191 responses for each IP in the hitlist. We show how many responses we gathered, per Internet Protocol (IP), in Figure 7.8. We observe a median for *the number of times an IP was seen* of 179.

One of our primary interests is whether the IPs that we measure are associated with just a single PoP. Our assumption is that they are, and in Figure 7.9 we can see that 95.3% of the IPs are in fact seen at only a single PoP across all measurements. Interestingly this figure shows that there is a significant tail

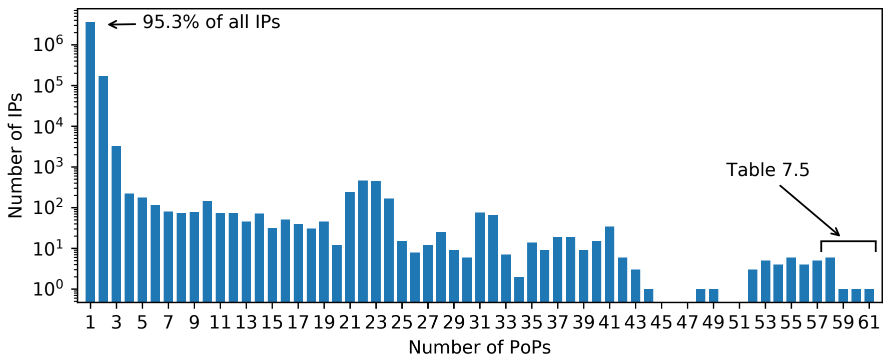


Figure 7.9: Number of Points-of-Presence seen for a single IP-prefix.

Prefix	PoPs	ASN	Rev. Hostname
192.58.128.0/24	61	26415	j.root-servers.net
204.61.216.0/23	60	42	ns.anycast.woodynet.net
192.33.14.0/24	59	26415	b.gtld-servers.net
189.201.244.0/23	58	42	e.mx-ns.mx
204.19.119.0/24	58	42	c.ns.apple.com
200.108.148.0/24	58	42	c.dns.ar
206.51.254.0/24	58	42	lns61.nic.tr
13.107.4.0/24	58	8068	ns1.c-msedge.net
194.0.17.0/24	58	42	e.nic.ch

Table 7.5: IP-prefixes hitting the highest number of PoPs. AS8068 = Microsoft, AS26415 = ICANN, AS42 = PCH

with some IPs appearing at as many as 61 PoPs. As we have said before, we assume that a likely reason for this is that those IPs are actually in an anycasted prefix themselves. To confirm that those IPs are anycasted, we take a look at those IPs that show up in the most PoPs and show their corresponding AS, and their reverse hostname in Table 7.5.

Interestingly, the top 9 IPs that hit the highest number of PoPs all belong to one of three organizations, all of which manage large anycast networks, namely Microsoft, Verisign and Packet Clearing House (PCH). Two interesting examples are the J DNS root server (reverse hostname *j.root-servers.net*), which is hosted by Verisign, which is seen at 61 different PoPs, as well as one of the Generic Top-Level Domain (gTLD) authoritative nameservers (reverse hostname *b.gtld-servers.net*), which is also hosted by Verisign. We interpret this as confirmation that anycasted services can indeed be revealed using this methodology.

Given that 95% of the IP addresses are only seen at a single PoP, it seems likely that also many Autonomous Systems, and all their addresses, are only

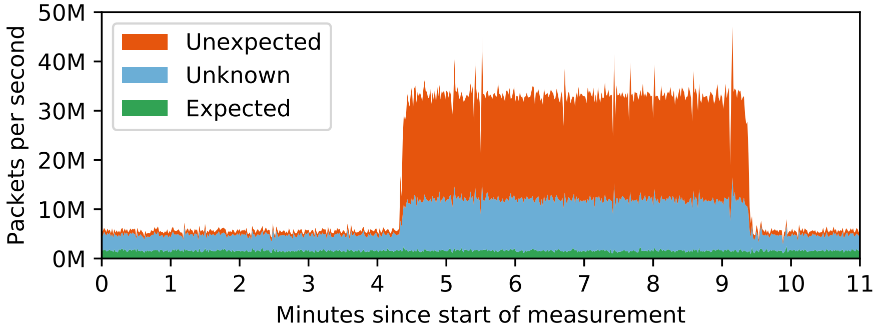


Figure 7.10: Detecting spoofed traffic, according to the previously established client to PoP mapping, during a known SYN-flood attack

seen at a single PoP. We investigate this by looking up the ASNs of each of the addresses, and confirm that out of a total of 60,295 ASNs that we see, from 52,533 ASNs all IPs are seen at a single PoP, but not necessarily the same one for each IP. 52,533 ASNs have IPs that are also only seen at a single PoP, and that PoP is also the same for all of the addresses. 2,541 only have addresses that are seen at multiple PoPs, while 3,069 contain addresses that are seen at both single as well as multiple PoPs.

These results show that most IP addresses that are seen on the Internet, and in fact most of the ASes, are, in a stable anycast network, only expected to show up at a single Point-of-Presence. Theoretically this should mean that any traffic that arrives at a PoP that it is is not supposed to, according to this mapping, can be considered as *likely to be spoofed.*

To demonstrate this in practice we recorded flow measurements during a known SYN-flood attack, on the same day the measurements for the client to PoP mapping were run. For each flow we checked if the source IP was supposed to arrive at the PoP it did, according to the mapping. The results of this are shown in Figure 7.10. We can see that the *expected* traffic is constant over the duration (11 minutes). We see a steep increase in *unexpected* source addresses during the attack itself, indicating that the mapping provides a strong signal for detecting spoofed traffic. The *unknown* category indicates that even though our measurements cover many prefixes, there are still those for which we have no mapping. We intend to investigate the distribution of IPs inside the *Unknown* category in future work, a possible explanation could be a high baseline of continuous spoofed traffic. Compared to the traditional method of looking at Time to Lives (TTLs) to identify spoofed traffic [118], our method promises to provide a stronger signal, which we also intend to further investigate in the future.

The key takeaway from this section is that we have shown that Verfploeter can be applied to help operators defend against DDoS attacks that consist of spoofed traffic. Once identified this traffic can then be filtered or rate-limited.

7.4 Related Work

There is a large body of past work that used public measurement platforms, such as RIPE Atlas [65], PlanetLab [171], and the NLNOG RING [173], to study the operation of anycast services [3], [62], [66], [167], [101], [102], [179], [180]. Using public measurement platforms is a flexible way to study anycast services. A key characteristic of these platforms is that they provide some form of access, e.g. via Secure Shell (SSH), an API, or a Web UI, to physical or virtual *probes*. Typically they allow a wide array of measurements to be performed. Compared to the Verfploeter method used in this chapter, however, the use of public measurement platforms has two limitations. First, the number of vantage points from which measurements can be conducted is limited, and often many orders of magnitude less than what Verfploeter can achieve. Second, these measurements suffer from an inherent bias because of the placement of vantage points. This means they will always provide an incomplete view of actual anycast catchments.

Li et al [58] looked at one of the root DNS servers for over a year, and find that site loads are often unbalanced, and that many clients are routed to a PoP that is suboptimal in terms of great-circle distance by several thousands of kilometers. They attribute this problem to inter-domain routing topology and policies, as well as poor route selection in the case that routes to multiple sites are available. Earlier work by Schmidt et al [102] finds similar results. These finds strengthen the case for a system that is able to perform fast and precise measurements of anycast catchments.

Past work has also focused specifically on anycast performance in CDNs. Calder et al. study the anycast network for Microsoft's Bing search engine and show that while anycast generally performs well, up to 20% of clients get redirected to sub-optimal PoPs. They introduce a simple scheme to improve PoP assignment using other means than anycast (specifically DNS-based methods). Flavel et al. introduce FastRoute, a hybrid approach that combines anycast and DNS-based load balancing to route clients to optimal PoPs. Where these past studies focused on anycast as just one means of distributing traffic, and augment this with other load balancing approaches, in this chapter, we focus on a network that is managed purely as an anycast service, and show how Verfploeter can play a vital role in understanding and managing this network.

Finally, very recent work by McQuistin et al. shows that anycast services that make use of multiple network operators can benefit from tailoring their Border Gateway Protocol (BGP) announcements to optimise end-user performance [119].

7.5 Concluding Remarks

In this chapter we show how a global scale anycast network can be managed, using the Verfploeter methodology that we introduced in this Thesis. Compared to the previous chapter, we have validated it on a much larger scale, on one of the world's largest anycast CDNs. We have also shown three use cases, that exemplify how this methodology can be applied.

We have described what the goals were for the implementation, how we chose to meet those goals and what the final operational implementation looks like. This implementation validates that Verfploeter can be deployed on any scale, from small dual node anycast services, such as shown in our initial work, up to the largest scale networks on the Internet.

Our three use cases show: **Firstly**, how this methodology can be applied by operators for planning purposes, and have shown that, depending on the case, the outcome of taking offline different combinations of anycast PoPs can be determined based on measurements in which only a single PoP was taken offline. **Secondly**, how it brings a traditional troubleshooting tool `ping`, to the realm of anycast, and demonstrate how it was used to assess the reachability of a particularly interesting IP address. **Lastly**, we show how the methodology can be used to assist in the mitigation of DDoS attacks by providing a way to identify spoofed traffic based on known client to IP mappings.

CHAPTER 8

Conclusions

8.1 Goal and Research Questions

In Chapter 1, we identified five research questions to guide the research presented in this work. In this section, we return to these research questions and show how they were addressed throughout this thesis.

Recall that we identified that there was a limited body of research in the area of anycast. Given the large role it plays on the Internet nowadays, we set out to investigate anycast and improve it, particularly because of its potency in DDoS mitigation. We summarized this with the following goal:

> *To measure anycast deployments and develop methods to optimize anycast deployments in order to improve service resilience against DDoS attacks*

We then derived five research questions to help reach this goal, the first of which was:

> **RQ1** – *What can we learn from analyzing the behaviour of a large-scale anycast service?*

In Chapter 3, we took an in-depth look at the Google Public DNS (GPDNS), a service hosted on a large anycasted network. A key find towards RQ1 is that routing in anycast is very difficult to get right. This is demonstrated by the fact that users are frequently routed out of country, when an in country Point-of-Presence (PoP) is available. The difficulty is further illustrated by the choice of sub-optimal Points-of-Presence when considering the great-circle distance between the possible PoPs and the PoP that clients are routed to.

It is important to note that traffic being routed outside of the country of origin, while an in country PoP is available, can be detrimental to user privacy. For example, we saw large numbers of queries, originating in the European Union, being served from the United States, while there were several PoPs available inside the EU.

Another aspect to highlight is the stability of the network over time, which was made possible because our analysis spanned more than 2.5 years. We observed that large deviations occurred several times. The exact cause of those changes is unknown, the fact that there *are* sudden changes that influence a large part of the client-base indicates that network engineering, in the context of anycast, is not a fine-grained art.

An unexpected find, unrelated to anycast but important from a privacy standpoint is that many SMTP servers appear to be using GPDNS as a resolver. The fact that such servers frequently perform reverse DNS lookups on IP addresses that they interact with, e.g. during an SMTP-session for sending and/or receiving e-mail, means that users who interact with a service unrelated to Google, might unknowingly end up in their logs, as well as in the logs of any other name server that is involved in the lookup process.

We observed a large anycast network under normal conditions, and conclude that operating such a network is not trivial. We now turn to an anycast network while it is suffering from a DDoS attack:

RQ2 – *How does anycast perform in the face of a DDoS attack?*

In Chapter 4, we investigated the effect that a large DDoS attack had on a well-known anycasted service: the root DNS. This service is particularly interesting for three reasons: **a)** it is critical for the Internet to properly function, **b)** it is composed of individual replicas which are hosted by several individual organizations and **c)** each of these vary widely in size. We argue that these aspects make the conclusions we draw from this analysis important from a societal perspective considering the stability of the Internet. They also make the conclusions applicable to a wider audience considering the wide variety of deployments that the root DNS consists of.

In our analysis we looked at how operators respond when they are facing a DDoS attack, in terms of networking engineering. We identified two main strategies:

- **Withdraw:** operators withdraw the anycast route announcement from a site that is (close to being) overloaded. This leads to traffic shifting to different sites, that in the positive case are capable of sustaining the added load. In the negative case those other sites can also become overloaded. It is not always clear if a route was intentionally withdrawn or not, BGP sessions may fail when the network is overloaded, automatically causing a withdraw.

- **Absorb:** in the case of an overloaded site, the operator decides to keep the route announcement, causing service degradation within the catchment of that site. This strategy is especially useful if withdrawing the route would cause the traffic to reroute to other sites that are known to be close to being overloaded.

A side observation is that collateral damage occurs during large attacks. This means that other servers that are topologically close, and practically that often means also physically close, can be affected by attacks on nearby servers.

In answer to **RQ2**, we observe that even though the capacity of the root DNS in aggregate is very high, and the attack traffic did not exceed that maximum combined capacity, we still show significant impact on reachability of individual root letters. In the DNS this is problem is mitigated by having many different service operators, and resolvers will automatically retry in case of failures. However, other services, such as CDNs, typically do not have this built-in redundancy, where a client will attempt a different server, with a different address, automatically.

RQ3 – *What challenges are there in measuring anycast networks?*

Many challenges exist when performing measurements on the Internet, and it is infeasible to cover all of them in this thesis. However, we have focused on a challenge that is of particular interest in the case of anycast, namely routing asymmetry. In Chapter 5 we investigated the principle of asymmetric Internet routing. Asymmetry means that the path that is taken when a packet travels from point A to point B on the Internet is not the same as it would take in the reverse direction. This phenomenon directly impacts our ability to measure routes on the Internet, and therefore limits our ability to use such route information to reason about anycast networks.

Path asymmetry has a significant impact on what can be measured about Internet routes. Traditional path measurement tools such as `traceroute` work by sending packets with an increasing TTL, iteratively discovering the path that the packets travel along. However, the assumption that the path that is measured from side A, is the same as that measured from side B, does not hold. As a key find we have shown that asymmetry occurs in considerable fraction of the Internet, namely on 85% of the measured paths.

When developing a new methodology to measure anycast networks it is important to take into account that the paths which determine anycast catchments cannot be reliably measured from the server side using a `traceroute`. On the other hand, a methodology which relies on measurements from inside the anycast network, as opposed to measurements from the client side *into* the anycast network, has the potential to have far more coverage as it does not rely on software or hardware being distributed throughout the world.

The fact that asymmetry is so widespread makes it clear that relying on the discovery of paths, from the server side, as input to an algorithm that is able to determine anycast catchments is infeasible.

RQ4 – *How can we accurately measure anycast performance?*

We concluded that the measurement of paths in the Internet is unlikely to lead to an accurate method to assess the performance of an anycast network, in terms of client to PoP mapping. To answer **RQ4** we therefore developed a measurement methodology which is not dependent on knowledge of the actual routes.

The measurement methodology, which we named Verfploeter (see Chapter 6), works by using ICMP Echo Requests (i.e. *pings*), and relies on the fact that the responses to these requests will be routed independently. When an anycast site receives a response to such a ping, it can be assumed that the sender of that response (i.e. *the client*) is in the catchment of that site. Using this method we have shown that the associated anycast site can be determined for any ICMP-responsive host on the Internet.

An significant benefit of this method is that it does not rely on probes that need to be physically distributed throughout the world. Systems that do depend on such probes typically have a bias in terms of distribution. For example, RIPE Atlas is biased towards Western Europe and North America.

Previous work [108] has shown that for many /24s on the Internet a representative address can be selected that is deemed most likely to respond. The authors of that work regularly distribute a hitlist of these addresses. Leveraging that list, we can determine the catchment of a large portion of the Internet in a relatively short time span.

We have shown that this method is both feasible as well as beneficial for operators in three ways: **a)** we set up an anycast testbed, on which we implemented a system that uses the Verfploeter methodology to measure catchments. Using this testbed we performed measurements showing, for example, that all sub-prefixes of an announced prefix do not always behave in the same way in terms of routing. We also showed that anycast catchments are relatively stable over the course of at least 24 hours. **b)** we deployed the same system on the B root DNS server, hosted by USC/ISI, at that time the B root was the only remaining unicast root server and Verfploeter was used to assess the anycast catchment prior to moving to a two site anycast setup. Finally, **c)** as shown in Chapter 7, we deployed a measurement system that implements Verfploeter at Cloudflare, which operates one of the largest anycast networks in the world.

RQ5 – *How can we improve anycast performance and operations?*

With our final research question, we look towards how we can improve anycast networks and assist operators in managing them. With our measurement methodology in hand, we demonstrated how it can be used, both on the B root DNS as well as on Cloudflare's anycast network.

In Chapter 6 we have shown how Verfploeter can be used in the deployment of a new anycast site. As well as how to correctly weigh the results by mapping load information onto the measurement data. In that particular example, the measurement with Verfploeter matched the actual load after deployment with an accuracy of 99.8%. We also showed that this method allows for more accurate results than what can be obtained using, for example, RIPE Atlas.

In Chapter 7, we introduce three use cases, which we have also validated on a large anycast network:

- **Planning:** we show that by performing measurements with specific anycast sites disabled, an overview can be obtained of how the anycast catchments will respond in those cases. Additionally, we show that predictions can be made for when multiple anycast sites are disabled even based on measurements where only a single anycast site is disabled.
- **Securing:** by initiating measurements from a large number of different anycast sites, each time towards the same targets, we have established that a large percentage (95%) of the measured IP-ranges is associated with just a single anycast site. We argue that a regularly updated catchment map can then be used to detect incoming spoofed traffic. IP spoofing is a significant contributor to DDoS attacks, and as such we argue that this is an important contribution.
- **Troubleshooting:** due to the nature of anycast, performing *pings* has in many cases become impossible. Since the response of the ping will often end up in a different anycast site from the one originating the request packet. However, we show that Verfploeter offers similar capabilities, while also working on an anycasted network, restoring a classic troubleshooting technique to the operators.

To conclude, in this thesis we have first investigated two global anycast services, the root DNS and Google Public DNS. We show that their deployments are not optimal, for example in terms of physical distance between the client and the anycast site. We show that DDoS attacks still manage to do damage, despite the fact that bandwidth is, in aggregate, abundantly available. We identified that tools to measure anycast networks are lacking, and developed a new methodology that allows operators to assess their anycast catchments with a high resolution, covering tens of thousands of ASes. In addition, our methodology is free of the bias that is introduced by the deployment of physical probes, such as those used by RIPE Atlas. We have validated the methodology and tools by deploying it on a testbed, the B root DNS and one of the largest anycast networks in the world, Cloudflare's CDN.

8.2 Future Research Directions

We have identified a number of directions in which the work in this thesis can be expanded as future work. These directions fall into four categories: validation in the context of DDoS, expansion of the used hitlist, assessing anycast catchments based on control plane information and enumerating anycast networks on the Internet.

Firstly, we have validated the approach by demonstrating its use in the real-world, in several scenarios. However, we have not yet demonstrated use cases in the context of DDoS. We believe that our methodology can be used to make better network engineering decisions *during* attacks.

Secondly, the research in this thesis focused mainly on anycast on IPv4. While the principles of routing are unchanged for IPv6, and as such the methodology developed in this thesis can be applied unchanged, work on hitlists is an ongoing field of research and is yet to match the breadth of IPv4 hitlists. Finding other sources of targets for our methodology, and validating our approach for IPv6 is important future work, and we have already started work on adding IPv6 support in our implementation. Additionally, during our work we performed many measurements towards millions of targets obtained from IPv4 hitlists, and the response rate varies between 50% and 60%. Future research might improve this response rate, and consequently cover a larger share of the Internet. This can be achieved by repeating measurements more often and/or by selecting multiple representative IP-addresses per /24 prefix, as opposed to the current single address.

Thirdly, in terms of BGP, there is limited insight into the routers that make up the Internet. While projects such as RouteViews and RIPE RIS attempt to provide routing data, the number of routers that are connected to such so-called *route collectors* is limited. In addition, only active routes (in terms of the BGP route selection process) are visible through these collectors. BMP [120] was proposed to gain more visibility into Internet routes, but its adoption is slow. This makes it hard to measure anycast catchments from a control plane perspective. When more data becomes available in due time, this option should be revisited.

Finally, in Chapter 7, we argued that it is possible to detect anycast networks by measuring from inside another anycast network. We showed that this is indeed the case. More in-depth investigation of this that could potentially lead to revealing a large part of the anycast networks on the Internet, as well as validation against existing enumeration methods, is future work.

Open Data Management

During the studies that resulted in this thesis various data sets were created. In this appendix we list those that are public, where they can be found and to which chapter they relate.

Chapter	Description	Location
Chapter 3	DNS queries from Google	`https://doi.org/10.4121/uuid:1ef815ea-cb39-4b41-8db6-c1008af6d5aa`
Chapter 3	Measurement data for QName Minimization	`https://www.simpleweb.org/wiki/index.php/Traces#A_First_Look_at_QNAME_Minimization_in_the_Domain_Name_System`
Chapter 4	RIPE Atlas measurements	`https://www.simpleweb.org/wiki/index.php/Traces#Anycast_vs._DDoS:_The_Nov_2015_Root_DNS_Event`
Chapter 6	RIPE Atlas and Verfploeter measurements	`https://www.simpleweb.org/wiki/index.php/Traces#Broad_and_Load-Aware_Anycast_Mapping_with_Verfploeter`

Table A.1: Open data sets

Bibliography (refereed)

[1] J. J. C. de Santanna, “DDoS-as-a-Service: investigating booter websites”, PhD thesis, University of Twente, 2017.

[2] J. Mirkovic and P. Reiher, “A taxonomy of DDoS attack and DDoS defense mechanisms”, *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004.

[3] 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, pp. 255–270. [Online]. Available: `http://www.isi.edu/%7ejohnh/PAPERS/Moura16b.html`.

[4] D. Giordano, D. Cicalese, A. Finamore, M. Mellia, M. Munafo, D. Joumblatt, and D. Rossi, “A First Characterization of Anycast Traffic from Passive Traces”, in *IFIP workshop on Traffic Monitoring and Analysis (TMA)*, Apr. 2016, pp. 30–38.

[5] K. Lougheed and J. Rekhter, “Border Gateway Protocol (BGP)”, RFC Editor, RFC 1105, Jun. 1989. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc1105.txt`.

[6] D. Mills, “Exterior Gateway Protocol formal specification”, RFC Editor, RFC 904, Apr. 1984. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc904.txt`.

[7] K. Lougheed and Y. Rekhter, “Border Gateway Protocol (BGP)”, RFC Editor, RFC 1163, Jun. 1990. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc1163.txt`.

[8] ——, “Border Gateway Protocol 3 (BGP-3)”, RFC Editor, RFC 1267, Oct. 1991. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc1267.txt`.

[9] Y. Rekhter and T. Li, “A Border Gateway Protocol 4 (BGP-4)”, RFC Editor, RFC 1654, Jul. 1994. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc1654.txt`.

[10] ——, “A Border Gateway Protocol 4 (BGP-4)”, RFC Editor, RFC 1771, Mar. 1995. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc1771.txt`.

[11] Y. Rekhter, T. Li, and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC Editor, RFC 4271, Jan. 2006. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4271.txt.

[12] V. Fuller, T. Li, J. (Y. Yu, and K. Varadhan, "Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy", RFC Editor, RFC 1519, Sep. 1993. [Online]. Available: http://www.rfc-editor.org/rfc/rfc1519.txt.

[13] Q. Vohra and E. Chen, "BGP Support for Four-octet AS Number Space", RFC Editor, RFC 4893, May 2007. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4893.txt.

[14] V. Giotsas, A. Dhamdhere, and k. claffy k., "Periscope: Unifying Looking Glass Querying", in *Passive and Active Network Measurement Workshop (PAM)*, Mar. 2016.

[15] R. Anwar, H. Niaz, D. Choffnes, Í. Cunha, P. Gill, and E. Katz-Bassett, "Investigating Interdomain Routing Policies in the Wild", in *Proceedings of the 2015 ACM Conference on Internet Measurement Conference*, ser. IMC '15, New York, NY, USA: ACM, 2015, pp. 71–77. [Online]. Available: http://doi.acm.org/10.1145/2815675.2815712.

[16] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford, "Dynamics of Hot-potato Routing in IP Networks", *SIGMETRICS Perform. Eval. Rev.*, vol. 32, no. 1, pp. 307–319, Jun. 2004. [Online]. Available: http://doi.acm.org/10.1145/1012888.1005723.

[17] P. Mockapetris, "Domain names - implementation and specification", RFC Editor, STD 13, Nov. 1987. [Online]. Available: http://www.rfc-editor.org/rfc/rfc1035.txt.

[18] C. Partridge, "Mail routing and the domain system", RFC Editor, STD 10, Jan. 1986. [Online]. Available: https://www.rfc-editor.org/rfc/rfc974.txt.

[19] J. Damas, M. Graff, and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", RFC Editor, STD 75, Apr. 2013. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6891.txt.

[20] C. Contavalli, W. van der Gaast, D. Lawrence, and W. Kumari, "Client Subnet in DNS Queries", RFC Editor, RFC 7871, May 2016. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7871.txt.

[21] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "DNS Security Introduction and Requirements", RFC Editor, RFC 4033, Mar. 2005. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4033.txt.

[22] ——, "Resource Records for the DNS Security Extensions", RFC Editor, RFC 4034, Mar. 2005. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4034.txt.

[23] ——, “Protocol Modifications for the DNS Security Extensions”, RFC Editor, RFC 4035, Mar. 2005. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4035.txt.

[24] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman, “Specification for DNS over Transport Layer Security (TLS)”, RFC Editor, RFC 7858, May 2016. [Online]. Available: https://rfc-editor.org/rfc/rfc7858.txt.

[25] P. Hoffman and P. McManus, “DNS Queries over HTTPS (DoH)”, RFC Editor, RFC 8484, Oct. 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc8484.txt.

[26] K. Fujiwara, A. Kato, and W. Kumari, “Aggressive Use of DNSSEC-Validated Cache”, RFC Editor, RFC 8198, Jul. 2017. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8198.txt.

[27] W. Hardaker, “Analyzing and Mitigating Privacy with the DNS Root Service”, in *NDSS: DNS Privacy Workshop, 2018*, 2018.

[28] A. Cooper, H. Tschofenig, B. Aboba, J. Peterson, J. Morris, M. Hansen, and R. Smith, “Privacy considerations for internet protocols”, RFC Editor, RFC 6973, Jul. 2013. [Online]. Available: https://rfc-editor.org/rfc/rfc6973.txt.

[29] V. Pappas, D. Wessels, D. Massey, S. Lu, A. Terzis, and L. Zhang, “Impact of Configuration Errors on DNS Robustness”, *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 3, pp. 275–290, 2009.

[30] S. Bortzmeyer and S. Huque, “NXDOMAIN: There Really Is Nothing Underneath”, RFC Editor, RFC 8020, Nov. 2016. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8020.txt.

[31] C. Rossow, “Amplification Hell: Revisiting Network Protocols for DDoS Abuse”, in *Proceedings of the 2014 Network and Distributed System Security (NDSS) Symposium*, 2014, pp. 23–26.

[32] J. P. John, A. Moshchuk, S. D. Gribble, and A. Krishnamurthy, “Studying Spamming Botnets Using Botlab”, in *Proceedings of the 6th USENIX Symposium on Network Systems Design and Implementation*, Boston, Massachusetts, USA: USENIX, Apr. 2009. [Online]. Available: http://www.cs.washington.edu/homes/jjohn/papers/botlab-%20nsdi09.pdf.

[33] P. Ferguson and D. Senie, “Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing”, RFC Editor, RFC 2267, Jan. 1998. [Online]. Available: https://www.rfc-editor.org/rfc/rfc2267.txt.

[34] D. E. 3rd and M. Andrews, “Domain Name System (DNS) Cookies”, RFC Editor, RFC 7873, May 2016. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7873.txt.

[35] L. Zhu, Z. Hu, J. Heidemann, D. Wessels, A. Mankin, and N. Somaiya, “Connection-Oriented DNS to Improve Privacy and Security”, in *Proceedings of the 36th IEEE Symposium on Security and Privacy*, IEEE, May 2015, pp. 171–186. [Online]. Available: `http://www.isi.edu/%7ejohnh/PAPERS/Zhu15b.html`.

[36] R. Beverly, A. Berger, Y. Hyun, and K. Claffy, “Understanding the Efficacy of Deployed Internet Source Address Validation Filtering”, in *Proceedings of the ACM Internet Measurement Conference*, ser. IMC, ACM, Nov. 2009, pp. 356–369. [Online]. Available: `http://rbeverly.net/research/papers/spoofer-imc09.pdf`.

[37] D. Gillman, Y. Lin, B. Maggs, and R. K. Sitaraman, “Protecting Websites from Attack with Secure Delivery Networks”, *IEEE Computer*, vol. 48, no. 4, pp. 26–34, Apr. 2015. [Online]. Available: `https://people.cs.umass.edu/~ramesh/Site/HOME_files/r4mag-%20v4.pdf`.

[38] A. Shaikh, L. Kalampoukas, R. Dube, and A. Varma, “Routing Stability in Congested Networks: Experimentation and Analysis”, in *Proceedings of the ACM SIGCOMM Conference*, ACM, Aug. 2000, pp. 163–174. [Online]. Available: `http://www.acm.org/sigcomm/sigcomm2000/conf/paper/%20sigcomm2000-5-1.ps.gz`.

[39] R. Bush, D. Karrenberg, M. Kosters, and R. Plzak, “Root Name Server Operational Requirements”, RFC Editor, RFC 2870, Jun. 2000. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc2870.txt`.

[40] 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”, in *2018 Network Traffic Measurement and Analysis Conference (TMA)*, IEEE, 2018, pp. 1–8.

[41] W. B. de Vries, Q. Scheitle, M. Müller, W. Toorop, R. Dolmans, and R. van Rijswijk-Deij, “A First Look at QNAME Minimization in the Domain Name System”, in *International Conference on Passive and Active Network Measurement*, Springer, 2019, pp. 147–160.

[42] J. S. Otto, M. A. Sánchez, J. P. Rula, and F. E. Bustamante, “Content Delivery and the Natural Evolution of DNS: Remote DNS Trends, Performance Issues and Alternative Solutions”, in *Proceedings of ACM IMC 2012*, Boston, MA, USA: ACM Press, 2012, pp. 523–536.

[43] M. Sánchez, J. S. Otto, Z. S. Bischof, D. Choffnes, F. Bustamante, B. Krishnamurthy, and W. Willinger, “Dasu: Pushing Experiments to the Internet’s Edge”, in *Proceedings of USENIX NSDI 2013*, Lombard, IL, USA: USENIX Association, 2013, pp. 487–499.

[44] F. Streibelt, J. Böttger, N. Chatzis, G. Smaragdakis, and A. Feldmann, “Exploring EDNS-Client-Subnet Adopters in Your Free Time”, in *Proceedings of ACM IMC 2013*, Barcelona, Spain: ACM Press, 2013, pp. 305–312.

[45] M. Calder, X. Fan, Z. Hu, E. Katz-Bassett, J. Heidemann, and R. Govindan, “Mapping the Expansion of Google’s Serving Infrastructure”, in *Proceedings of ACM IMC 2013*, Barcelona, Spain: ACM Press, 2013, pp. 313–326.

[46] X. Fan, E. Katz-Bassett, and J. Heidemann, “Assessing Affinity Between Users and CDN Sites”, in *Traffic Measurements and Analysis*, ser. LNCS, M. Steiner, P. Barlet-Ros, and O. Bonaventure, Eds., vol. 9053, Barcelona, Spain: Springer Berlin Heidelberg, 2015, pp. 95–110.

[47] Y. Yu, D. Wessels, M. Larson, and L. Zhang, “Authority Server Selection in DNS Caching Resolvers”, *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 2, p. 80, 2012.

[48] 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 ACM IMC 2017*, ACM, 2017, pp. 477–488.

[49] F. Chen, R. K. Sitaraman, and M. Torres, “End-User Mapping: Next Generation Request Routing for Content Delivery”, in *Proceedings of ACM SIGCOMM 2015*, London, UK: ACM Press, 2015, pp. 167–181.

[50] M. Kucherawy, “Message Header Field for Indicating Message Authentication Status”, RFC Editor, RFC 7601, Aug. 2015. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc7601.txt`.

[51] J. Klensin, “Simple Mail Transfer Protocol”, RFC Editor, RFC 5321, Oct. 2008. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc5321.txt`.

[52] S. Bortzmeyer, “DNS Query Name Minimisation to Improve Privacy”, RFC Editor, RFC 7816, Mar. 2016. [Online]. Available: `https://www.rfc-editor.org/rfc/rfc7816.txt`.

[53] ——, “DNS Privacy Considerations”, RFC Editor, RFC 7626, Aug. 2015. [Online]. Available: `https://www.rfc-editor.org/rfc/rfc7626.txt`.

[54] M. Wullink, G. C. Moura, M. Müller, and C. Hesselman, “ENTRADA: A High-Performance Network Traffic Data Streaming Warehouse”, in *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*, IEEE, 2016, pp. 913–918.

[55] S. Castro, D. Wessels, M. Fomenkov, and K. Claffy, “A Day at the Root of the Internet”, *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 5, pp. 41–46, 2008.

[56] Q. Scheitle, O. Hohlfeld, J. Gamba, J. Jelten, T. Zimmermann, S. D. Strowes, and N. Vallina-Rodriguez, “A Long Way to the Top: Significance, Structure, and Stability of Internet Top Lists”, in *IMC’18*, Boston, USA, Nov. 2018.

[57] P. Kintis, Y. Nadji, D. Dagon, M. Farrell, and M. Antonakakis, “Understanding the Privacy Implications of ECS”, in *Proceedings of DIMVA 2016*, ser. LNCS, J. Caballero, U. Zurutuza, and R. Rodríguez, Eds., vol. 9721, Springer Berlin Heidelberg, 2016, pp. 343–353.

[58] Z. Li, D. Levin, N. Spring, and B. Bhattacharjee, “Internet Anycast: Performance, Problems and Potential”, in *Proceedings of ACM SIGCOMM 2018*, Budapest, Hungary: ACM Press, 2018, pp. 59–73.

[59] B. Imana, A. Korolova, and J. Heidemann, “Enumerating Privacy Leaks in DNS Data Collected Above the Recursive”, in *NDSS: DNS Privacy Workshop, 2018*, San Diego, California, USA, Feb. 2018. [Online]. Available: `https://www.isi.edu/%5C%7ejohnh/PAPERS/Imana18a.html`.

[60] P. Schmitt, A. Edmundson, and N. Feamster, “Oblivious DNS: Practical Privacy for DNS Queries”, *arXiv:1806.00276*, 2018.

[61] Z. Wang, “Understanding the Performance and Challenges of DNS Query Name Minimization”, in *17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, IEEE, 2018, pp. 1115–1120.

[62] M. Calder, A. Flavel, E. Katz-Bassett, R. Mahajan, and J. Padhye, “Analyzing the Performance of an Anycast CDN”, in *Proceedings of the ACM Internet Measurement Conference*, Tokyo, Japan: ACM, Oct. 2015, pp. 531–537. [Online]. Available: `http://conferences2.sigcomm.org/imc/2015/papers/p531.pdf`.

[63] M. Weinberg and D. Wessels, “Review and Analysis of Anomalous Traffic to A-Root and J-Root (Nov/Dec 2015)”, in *24th DNS-OARC Workshop*, (presentation), Apr. 2016. [Online]. Available: `https://indico.dns-oarc.net/event/22/session/4/contribution/%207/material/slides/0.pptx`.

[64] T. Holterbach, C. Pelsser, R. Bush, and L. Vanbever, “Quantifying interference between measurements on the RIPE Atlas platform”, in *Proceedings of the ACM Internet Measurement Conference*, ser. IMC, ACM, Oct. 2015, pp. 437–443.

[65] RIPE NCC Staff, “RIPE Atlas: A Global Internet Measurement Network”, *The Internet Protocol Journal*, vol. 18, no. 3, pp. 2–26, Sep. 2015.

[66] X. Fan, J. Heidemann, and R. Govindan, “Evaluating Anycast in the Domain Name System”, in *Proceedings of the IEEE Infocom*, IEEE, Apr. 2013, pp. 1681–1689. [Online]. Available: `http://www.isi.edu/~johnh/PAPERS/Fan13a.html`.

[67] H. Yan, R. Oliveira, K. Burnett, D. Matthews, L. Zhang, and D. Massey, "BGPmon: A real-time, scalable, extensible monitoring system", in *Proceedings of the IEEE Cybersecurity Applications and Technologies Conference for Homeland Security (CATCH)*, IEEE, Mar. 2009, pp. 212–223.

[68] M. Lentz, D. Levin, J. Castonguay, N. Spring, and B. Bhattacharjee, "D-mystifying the D-root Address Change", in *Proceedings of the ACM Internet Measurement Conference*, ser. IMC, ACM, 2013, pp. 57–62.

[69] J. Gettys and K. Nichols, "Bufferbloat: dark buffers in the Internet", *Communications of the ACM*, vol. 55, no. 1, pp. 57–65, Jan. 2012.

[70] R. Elz, R. Bush, S. Bradner, and M. Patton, "Selection and Operation of Secondary DNS Servers", RFC Editor, BCP 16, Jul. 1997. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc2182.txt`.

[71] R. Fontugne, C. Pelsser, E. Aben, and R. Bush, "Pinpointing delay and forwarding anomalies using large-scale traceroute measurements", in *Proceedings of the 2017 Internet Measurement Conference*, ACM, 2017, pp. 15–28.

[72] A. Welzel, C. Rossow, and H. Bos, "On Measuring the Impact of DDoS Botnets", in *7th European Workshop on System Security*, Apr. 2014.

[73] R. van Rijswijk-Deij, A. Sperotto, and A. Pras, "DNSSEC and Its Potential for DDoS Attacks: a comprehensive measurement study", in *Proceedings of the ACM Internet Measurement Conference*, ser. IMC, ACM, Nov. 2014, pp. 449–460.

[74] J. Czyz, M. Kallitsis, M. Gharaibeh, C. Papadopoulos, M. Bailey, and M. Karir, "Taming the 800 Pound Gorilla: The Rise and Decline of NTP DDoS Attacks", in *Proceedings of the ACM Internet Measurement Conference*, ser. IMC, ACM, Nov. 2014, pp. 435–448. [Online]. Available: `http://doi.acm.org/10.1145/2663716.2663717`.

[75] J. J. Santanna, R. van Rijswijk-Deij, R. Hofstede, A. Sperotto, M. Wierbosch, L. Zambenedetti Granville, and A. Pras, "Booters-An analysis of DDoS-as-a-service attacks", in *IFIP/IEEE Intl. Symposium on Integrated Network Management (IM)*, IEEE, May 2015, pp. 243–251.

[76] M. Kührer, T. Hupperich, C. Rossow, and T. Holz, "Exit from Hell? Reducing the Impact of Amplification DDoS Attacks", in *23rd USENIX Security Symposium*, Aug. 2014, pp. 111–125.

[77] J. C.-Y. Chou, B. Lin, S. Sen, and O. Spatscheck, "Proactive Surge Protection: A Defense Mechanism for Bandwidth-Based Attacks", *IEEE/ACM Transactions on Networking (TON)*, vol. 17, no. 6, pp. 1711–1723, Dec. 2009.

[78] N. Brownlee, K. Claffy, and E. Nemeth, “DNS Root/gTLD Performance Measurement”, in *Proceedings of the USENIX Large Installation System Administration conference*, Dec. 2001, pp. 241–255.

[79] M. Fomenkov, k. c. claffy, B. Huffaker, and D. Moore, “Macroscopic Internet Topology and Performance Measurements From the DNS Root Name Servers”, in *Proceedings of the USENIX Large Installation Systems Administration Conference*, San Diego, CA, USA: USENIX, Dec. 2001, pp. 221–230. [Online]. Available: `http://www.caida.org/publications/papers/2001/Rssac2001a/%20rssac_lisa.pdf`.

[80] N. Brownlee and I. Ziedins, “Response Time Distributions for Global Name Servers”, in *Proceedings of the International conference on Passive and Active Measurements*, ser. PAM, Mar. 2002.

[81] T. Lee, B. Huffaker, M. Fomenkov, and K. Claffy, “On the problem of optimization of DNS root servers’ placement”, in *Proceedings of the International conference on Passive and Active Measurements*, ser. PAM, Mar. 2003.

[82] H. Ballani, P. Francis, and S. Ratnasamy, “A Measurement-based Deployment Proposal for IP Anycast”, in *Proceedings of the ACM Internet Measurement Conference*, ser. IMC, ACM, Oct. 2006, pp. 231–244.

[83] S. Sarat, V. Pappas, and A. Terzis, “On the use of Anycast in DNS”, in *Proceedings of the 15th International Conference on Computer Communications and Networks*, Oct. 2006, pp. 71–78.

[84] Z. Liu, B. Huffaker, M. Fomenkov, N. Brownlee, and K. Claffy, “Two Days in the Life of the DNS Anycast Root Servers”, in *Proceedings of the International conference on Passive and Active Measurements*, ser. PAM, Apr. 2007, pp. 125–134.

[85] B.-S. Lee, Y. S. Tan, Y. Sekiya, A. Narishige, and S. Date, “Availability and Effectiveness of Root DNS servers: A long term study”, in *Proceedings of the IEEE Network Operations and Management Symposium*, ser. NOMS, Apr. 2010, pp. 862–865.

[86] J. Liang, J. J. H. D. K. Li, and J. Wu, “Measuring Query Latency of Top Level DNS Servers”, in *Proceedings of the Passive and Active Measurement Workshop*, Hong Kong, China: Springer, Mar. 2013, pp. 145–154. [Online]. Available: `http://link.springer.com/chapter/10.1007/978-3-642-36516-%204_15`.

[87] H. Ballani and P. Francis, “Towards a Global IP Anycast Service”, in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, Aug. 2007, pp. 301–312.

[88] A. Flavel, P. Mani, D. Maltz, N. Holt, J. Liu, Y. Chen, and O. Surmachev, “Fastroute: A scalable load-aware anycast routing architecture for modern CDNs”, in *12th USENIX Symposium on Networked Systems Design and Implementation*, May 2015, pp. 381–394.

[89] W. de Vries, J. J. Santanna, A. Sperotto, and A. Pras, "How asymmetric is the Internet?", in *IFIP International Conference on Autonomous Infrastructure, Management and Security*, Springer, 2015, pp. 113–125.

[90] Y. He, M. Faloutsos, S. Krishnamurthy, and B. Huffaker, "On routing asymmetry in the Internet", in *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, vol. 2, Nov. 2005, 6 pp.–.

[91] Y. He, M. Faloutsos, and S. V. Krishnamurthy, "Quantifying routing asymmetry in the Internet at the AS level.", in *GLOBECOM*, IEEE, 2004, pp. 1474–1479. [Online]. Available: `http://dblp.uni-trier.de/db/conf/globecom/%20globecom2004.html%5C#HeFK04`.

[92] V. Paxson, "End-to-end Routing Behavior in the Internet", in *Conference Proceedings on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '96, New York, NY, USA: ACM, 1996, pp. 25–38. [Online]. Available: `http://doi.acm.org/10.1145/248156.248160`.

[93] Y. Schwartz, Y. Shavitt, and U. Weinsberg, "On the Diversity, Stability and Symmetry of End-to-End Internet Routes", in *INFOCOM IEEE Conference on Computer Communications Workshops , 2010*, 2010, pp. 1–6.

[94] E. Katz-Bassett, H. Madhyastha, and V. Adhikari, "Reverse traceroute", in *Network Systems Design and Implementation*, 2010. [Online]. Available: `https://www.usenix.org/legacy/event/nsdi10/tech/%20full%5C_papers/katz-bassett.pdf`.

[95] P. Ferguson and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", RFC Editor, BCP 38, May 2000. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc2827.txt`.

[96] RIPE NCC Staff, "RIPE Atlas: A Global Internet Measurement Network", *The Internet Protocol Journal*, vol. 18, no. 3, pp. 2–26, Sep. 2015.

[97] Y. H. Y. He, W. C. W. Chen, B. X. B. Xiao, and W. P. W. Peng, "An Efficient and Practical Defense Method Against DDoS Attack at the Source-End", *11th International Conference on Parallel and Distributed Systems (ICPADS'05)*, vol. 2, 2005.

[98] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals", in *Soviet physics doklady*, vol. 10, 1966, pp. 707–710.

[99] F. J. Damerau, "A technique for computer detection and correction of spelling errors", *Communications of the ACM*, vol. 7, no. 3, pp. 171–176, 1964.

[100] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian, "Internet Inter-Domain Traffic", in *Proceedings of the ACM SIGCOMM Conference*, New Delhi, India: ACM, Aug. 2010, pp. 75–86.

[101] D. Cicalese, J. Augé, D. Joumblatt, T. Friedman, and D. Rossi, "Characterizing IPv4 Anycast Adoption and Deployment", in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, 2015, 16:1–16:13.

[102] R. d. O. Schmidt, J. Heidemann, and J. H. Kuipers, "Anycast Latency: How Many Sites Are Enough?", in *Proceedings of the Passive and Active Measurement Workshop*, Sydney, Australia: Springer, Mar. 2017, pp. 188–200. [Online]. Available: `http://www.isi.edu/%7ejohnh/PAPERS/Schmidt17a.html`.

[103] L. Wei and J. Heidemann, "Does Anycast Hang up on You?", in *IEEE International Workshop on Traffic Monitoring and Analysis*, Dublin, Ireland: IEEE, Jul. 2017, to appear. [Online]. Available: `http://www.isi.edu/%7ejohnh/PAPERS/Wei17b.html`.

[104] V. Bajpai, S. J. Eravuchira, and J. Schönwälder, "Lessons Learned from using the RIPE Atlas Platform for Measurement Research", *ACM Computer Communication Review (CCR)*, vol. 45, no. 3, pp. 35–42, Jul. 2015.

[105] D. Cicalese, D. Joumblatt, D. Rossi, M.-O. Buob, J. Auge, and T. Friedman, "A Fistful of Pings: Accurate and Lightweight Anycast Enumeration and Geolocation", in *IEEE Conference on Computer Communications (INFOCOM)*, Apr. 2015, pp. 2776–2784.

[106] S. Woolf and D. Conrad, "Requirements for a Mechanism Identifying a Name Server Instance", RFC Editor, RFC 4892, Jun. 2007. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc4892.txt`.

[107] R. Austein, "DNS Name Server Identifier (NSID) Option", RFC Editor, RFC 5001, Aug. 2007. [Online]. Available: `http://www.rfc-editor.org/rfc/rfc5001.txt`.

[108] X. Fan and J. Heidemann, "Selecting Representative IP Addresses for Internet Topology Studies", in *Proceedings of the ACM Internet Measurement Conference*, Melbourne, Australia: ACM, Nov. 2010, pp. 411–423. [Online]. Available: `http://www.isi.edu/~johnh/PAPERS/Fan10a.html`.

[109] J. Heidemann, Y. Pradkin, R. Govindan, C. Papadopoulos, G. Bartlett, and J. Bannister, "Census and Survey of the Visible Internet", in *Proceedings of the ACM Internet Measurement Conference*, Vouliagmeni, Greece: ACM, Oct. 2008, pp. 169–182. [Online]. Available: `http://www.isi.edu/%7ejohnh/PAPERS/Heidemann08c.html`.

[110] L. Quan, J. Heidemann, and Y. Pradkin, "Trinocular: Understanding Internet Reliability Through Adaptive Probing", in *Proceedings of the ACM SIGCOMM Conference*, Hong Kong, China: ACM, Aug. 2013, pp. 255–266. [Online]. Available: `http://www.isi.edu/%7ejohnh/PAPERS/Quan13c.html`.

[111] P. B. Danzig, K. Obraczka, and A. Kumar, "An Analysis of Wide-Area Name Server Traffic: A study of the Domain Name System", in *Proceedings of the ACM SIGCOMM Conference*, Jan. 1992, pp. 281–292.

[112] I. Poese, S. Uhlig, M. A. Kaafar, B. Donnet, and B. Gueye, "IP Geolocation Databases: Unreliable?", *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, pp. 53–56, Apr. 2011. [Online]. Available: `http://doi.acm.org/10.1145/1971162.1971171`.

[113] B. Quoitin, C. Pelsser, O. Bonaventure, and S. Uhlig, "A performance evaluation of BGP-based traffic engineering", *International Journal of Nework Management*, vol. 15, no. 3, pp. 177–191, May 2005. [Online]. Available: `https://inl.info.ucl.ac.be/system/files/ijnm-evaluation-BGP-%20TE.pdf`.

[114] W. B. de Vries, S. Aljammaz, and R. van Rijswijk-Deij, "Global-Scale Anycast Network Management with Verfploeter", in *NOMS 2020 IEEE/IFIP Network Operations and Management Symposium*, IEEE, 2020, pp. 1–9.

[115] H. V. Jagadish, "Analysis of the Hilbert curve for representing two-dimensional space", *Information Processing Letters*, vol. 62, no. 1, pp. 17–22, 1997.

[116] R. Beverly and S. Bauer, "The Spoofer project: Inferring the extent of source address filtering on the Internet", in *Usenix Sruti*, vol. 5, 2005, pp. 53–59.

[117] R. Beverly, R. Koga, and K. Claffy, "Initial longitudinal analysis of IP source spoofing capability on the Internet", *Internet Society*, p. 313, 2013.

[118] Q. Scheitle, O. Gasser, P. Emmerich, and G. Carle, "Carrier-Grade Anomaly Detection Using Time-to-Live Header Information", *CoRR*, vol. abs/1606.07613, 2016. [Online]. Available: `http://arxiv.org/abs/1606.07613`.

[119] S. McQuistin, S. Priyanka, and M. Flores, "Taming Anycast in a Wild Internet", in *Proceedings of the 19th ACM SIGCOMM Internet Measurement Conference (IMC 2019)*, ACM, 2019.

[120] J. Scudder, R. Fernando, and S. Stuart, "BGP Monitoring Protocol (BMP)", RFC Editor, RFC 7854, Jun. 2016. [Online]. Available: `https://www.rfc-editor.org/rfc/rfc7854.txt`.

Bibliography

[121] Internet Hall of Fame, *Donald Davies*, https://www.internethalloffame.org/inductees/donald-davies, Accessed: 2019-11-08.

[122] *Perhaps the First Denial-of-Service Attack?*, http://www.platohistory.org/blog/2010/02/perhaps-the-first-denial-of-service-attack.html, Accessed: 2019-05-20.

[123] *Distributed Denial of Service Tools*, https://web.archive.org/web/20081113105659/http://www.cert.org/incident_notes/IN-99-07.html, Accessed: 2019-05-20.

[124] Arbor NetScout, https://www.netscout.com/blog/security-17tbps-ddos-attack-makes-history, Accessed: 2019-11-08.

[125] Akamai, *Prolexic Routed*, https://www.akamai.com/uk/en/products/security/prolexic-solutions.jsp, Accessed: 2019-11-08.

[126] NBIP, *NaWas*, https://www.nbip.nl/nawas/, Accessed: 2019-11-08.

[127] Root Server Operators, *Events of 2015-11-30*, Nov. 2015. [Online]. Available: http://root-servers.org/news/events-of-20151130.txt.

[128] Geoff Houston, *BGP Reports*, https://bgp.potaroo.net/, Accessed: 2019-11-22.

[129] NLnet Labs, *NLNet Labs: Unbound Changelog*, 2018. [Online]. Available: https://nlnetlabs.nl/svn/unbound/tags/release-1.8.0/doc/%20Changelog.

[130] CZ.NIC, *Knot Resolver 1.0.0 released*, 2016. [Online]. Available: https://www.knot-resolver.cz/2016-05-30-knot-resolver-%201.0.0.html.

[131] ISC, *Release Notes for BIND Version 9.13.2*, 2018. [Online]. Available: https://ftp.isc.org/isc/bind9/9.13.2/RELEASE-NOTES-bind-%209.13.2.txt.

[132] S. Bortzmeyer, *PowerDNS - Add Qname minimisation*, 2015. [Online]. Available: https://github.com/PowerDNS/pdns/issues/2311.

[133] Root Operators, *Root Server Technical Operations*, Apr. 2016. [Online]. Available: http://www.root-servers.org.

[134] Arbor Networks, *Rio Olympics Take the Gold for 540gb/sec Sustained DDoS Attacks!*, Aug. 2016. [Online]. Available: https://www.arbornetworks.com/blog/asert/rio-olympics-take-gold-540gbsec-sustained-ddos-attacks/.

[135] P. Vixie, *Response Rate Limiting in the Domain Name System (DNS RRL)*, blog post http://www.redbarn.org/dns/ratelimits, Jun. 2012. [Online]. Available: http://www.redbarn.org/dns/ratelimits.

[136] Cloudflare, https://www.cloudflare.com, Accessed: 2019-11-22.

[137] J. Abley and K. Lindqvist, "Operation of Anycast Services", Internet Request For Comments, RFC 4786, Dec. 2006, (also Internet BCP-126). [Online]. Available: http://www.rfc-editor.org/rfc/rfc4786.txt.

[138] Google, *Google Public DNS and Location-Sensitive DNS Responses*, 2014. [Online]. Available: https://webmasters.googleblog.com/2014/12/google-public-dns-%20and-location.html.

[139] CAIDA, *The CAIDA UCSD Routeviews Prefix to AS mappings Dataset (pfx2as) for IPv4 and IPv6*, 2018. [Online]. Available: http://www.caida.org/data/routing/routeviews-prefix2as.xml.

[140] IP2Location, *IP2Location LITE IP-COUNTRY Database.* [Online]. Available: https://lite.ip2location.com/database/ip-country.

[141] Google, *Google Public DNS FAQ*, 2018. [Online]. Available: https://developers.google.com/speed/public-dns/%20faq#locations.

[142] Internet Archive, *The Wayback Machine*, https://www.archive.org, Accessed: 2019-11-22.

[143] NOS, *Is your provider under attack? This is how you restore service. [in Dutch]*, Aug. 2015. [Online]. Available: https://nos.nl/op3/artikel/2052944-aanval-op-je-provider-zo-%20krijg-je-weer-verbinding.html.

[144] *RIPE Atlas measurement for a.b.qnamemin-test.internet.nl TXT*, https://atlas.ripe.net/measurements/8310250/, 2017.

[145] *RIPE Atlas measurement for o-o.myaddr.l.google.com TXT*, https://atlas.ripe.net/measurements/8310237/, 2017.

[146] *RIPE Atlas measurement for ripe-hackathon6.nlnetlabs.nl AAAA*, https://atlas.ripe.net/measurements/8310366/, 2017.

[147] *RIPE Atlas measurement for whoami.akamai.net A*, https://atlas.ripe.net/measurements/8310245/, 2017.

[148] DNS OARC, *Day In The Life of the Internet*, 2017 and 2018. [Online]. Available: https://www.dns-oarc.net/oarc/data/ditl.

[149] Cisco, *Cisco Umbrella Top 1M List*, https://s3-us-west-1.amazonaws.com/umbrella-static/index.html, Sep 14 – Sep 30, 2018.

[150] Arbor Networks, *Worldwide Infrastructure Security Report*, Volume VIII, Sep. 2012. [Online]. Available: `http://www.arbornetworks.com/resources/infrastructure-%20security-report`.

[151] ——, *Worldwide Infrastructure Security Report*, Volume IX, Jan. 2014. [Online]. Available: `http://pages.arbornetworks.com/rs/arbor/images/%20WISR2014.pdf`.

[152] N. Perlroth, *Tally of Cyber Extortion Attacks on Tech Companies Grows*, Jun. 2016. [Online]. Available: `http://bits.blogs.nytimes.com/2014/06/19/tally-of-cyber-extortion-attacks-on-tech-companies-grows/?_r=0`.

[153] R. Chirgwin, *Linode: Back at last after ten days of hell*, Jan. 2016. [Online]. Available: `http://www.theregister.co.uk/2016/01/04/linode_back_at_last_after_ten_days_of_hell/`.

[154] RIPE NCC, *DNSMON*, `https://atlas.ripe.net/dnsmon/`, 2015. [Online]. Available: `https://atlas.ripe.net/dnsmon/`.

[155] ProtonMail, *Guide to DDoS protection*, `https://protonmail.com/blog/ddos-protection-guide/`, Dec. 2015.

[156] Root Server Operators, *Events of 2016-06-25*, Jun. 2016. [Online]. Available: `http://www.root-servers.org/news/events-of-20160625.txt`.

[157] D. Wessels, *Verisign's Perspective on Recent Root Server Attacks*, Dec. 2015. [Online]. Available: `http://www.circleid.com/posts/20151215_verisign_perspective_on_recent_root_server_attacks/`.

[158] G. C. M. Moura, R. de O. Schmidt, J. Heidemann, W. B. de Vries, M. Müller, L. Wei, and C. Hesselman, *Nov. 30 Datasets*, `http://traces.simpleweb.org/` and `https://ant.isi.edu/datasets/anycast/`, 2016.

[159] RIPE NCC, *RIPE Atlas Root Server Data*, `https://atlas.ripe.net/measurements/ID`, ID is the per-root-letter experiment ID: A: 10309, B: 10310, C: 10311, D: 10312, E: 10313, F:10304, G: 10314, H: 10315, I: 10305, J: 10316, K: 10301, L: 10308, M: 10306, 2015.

[160] Ripe Atlas, *Graphs: Probe firmware versions*, Sep. 2016. [Online]. Available: `https://atlas.ripe.net/results/graphs/`.

[161] RSSAC, *Advisory on Measurements of the Root Server System*, Nov. 2014. [Online]. Available: `https://www.icann.org/en/system/files/files/rssac-002-%20measurements-root-20nov14-en.pdf`.

[162] B-Root Operators, *Personal communication*, Dec. 2015.

[163] R. Zwart and A. Buddhdev, *Report: K-root on 30 November and 1 December 2015*, Feb. 2015. [Online]. Available: `https://labs.ripe.net/Members/romeo_zwart/report-on-the-%20traffic-load-event-at-k-root-on-2015-11-30`.

[164] L. H. Newman, *GitHub Survived the Biggest DDoS Attack Ever Recorded*, Wired.com, Ed., Online; accessed 7-July-2019, Mar. 2018. [Online]. Available: `https://www.wired.com/story/github-ddos-memcached/`.

[165] B. Krebs, *Israeli Online Attack Service 'vDOS' Earned $ 600,000 in Two Years*, Sep. 2016. [Online]. Available: `http://krebsonsecurity.com/2016/09/israeli-online-attack-service-vdos-earned-600000-in-two-years/`.

[166] L. Colitti, *Effect of anycast on K-root*, 1st DNS-OARC Workshop, Jul. 2005.

[167] R. Bellis, *Researching F-root Anycast Placement Using RIPE Atlas*, `https://labs.ripe.net/Members/ray_bellis/researching-f-root-anycast-placement-using-ripe-atlas`, Oct. 2015.

[168] P. Boothe and R. Bush, *Anycast Measurements Used to Highlight Routing Instabilities*, NANOG 34, May 2005.

[169] R. Bush, *DNS Anycast Stability: Some Initial Results*, CAIDA/WIDE Workshop, Mar. 2005.

[170] Iputils, *ping(8)*, 2014. [Online]. Available: `http://man7.org/linux/man-pages/man8/ping.8.html`.

[171] PlanetLab, `https://www.planet-lab.org/`.

[172] EmanicsLab, `http://www.emanicslab.org/`, Accessed: 2019-11-22.

[173] NLNOG Ring, `http://ring.nlnog.net/`.

[174] W.B. de Vries, *IPASNExporter*, `https://bitbucket.org/woutifier/ipasnexporter`, Accessed: 2019-11-22.

[175] CAIDA, *Center for Applied Internet Data Analysis*, `https://www.caida.org/home/`, Accessed: 2019-11-22.

[176] MaxMind, *MaxMind GeoLite2*. [Online]. Available: `http://dev.maxmind.com/geoip/geoip2/geolite2/`.

[177] Thousand Eyes, *DNS Monitoring Service*, 2017. [Online]. Available: `https://www.thousandeyes.com/solutions/dns`.

[178] J. Mauch, *Open Resolver Project*, Talk at DNS-OARC, May 2013. [Online]. Available: `https://indico.dns-oarc.net/indico/getFile.py/access?contribId=24&sessionId=0&resId=1&materialId=slides&confId=0`.

[179] D. Madory, A. Popescu, and E. Zmijewski, *Accidentally Importing Censorship - The I-root instance in China*, Presentation, Renesys Corporation, Jun. 2010. [Online]. Available: `https://www.nanog.org/meetings/nanog49/presentations/%20Tuesday/Madory-I-root-lightning-talk.pdf`.

[180] E. Aben, *DNS Root Server Transparency: K-Root, Anycast and More*, Web, RIPE NCC, Jan. 2017. [Online]. Available: `https://labs.ripe.net/Members/emileaben/dns-root-server-%20transparency`.

[181] A. Hussain, G. Bartlett, Y. Pryadkin, J. Heidemann, C. Papadopoulos, and J. Bannister, "Experiences with a Continuous Network Tracing Infrastructure", USC/Information Sciences Institute, Tech. Rep. ISI-TR-2005-601, Apr. 2005. [Online]. Available: `http://www.isi.edu/~johnh/PAPERS/Hussain05a.html`.

[182] L. Colitti, *K-root Anycast Deployment*, Presentation at RIPE-52, Apr. 2006. [Online]. Available: `https://meetings.ripe.net/ripe-52/presentations/ripe52-plenary-kroot-anycast.pdf`.

[183] RSSAC, "RSSAC Advisory on Measurements of the Root Server System", ICANN, Tech. Rep. RSSAC002v3, Jun. 2016. [Online]. Available: `https://www.icann.org/en/system/files/files/rssac-002-%20measurements-root-06jun16-en.pdf`.

[184] P. Vixie and A. Kato, "DNS Referral Response Size Issues", IETF Secretariat, Internet-Draft draft-ietf-dnsop-respsize-14, May 2012. [Online]. Available: `http://www.ietf.org/internet-drafts/draft-ietf-dnsop-respsize-14.txt`.

[185] B-Root, *B-Root Begins Anycast in May*, `http://root-servers.org/news/b-root-begins-anycast-in-may.txt`, Apr. 2017.

[186] DNS-OARC, *Day In The Life of the Internet (DITL) 2017*, `https://www.dns-oarc.net/oarc/data/ditl/2017`, Apr. 2017. [Online]. Available: `https://www.dns-oarc.net/oarc/data/ditl/2017`.

[187] RIPE, *RIPE Atlas measurements of B-Root*, `https://atlas.ripe.net/measurements/`, measurement IDs 10310, 8310594, 8312460, 8312974, 8313009, 8313262, 2017.

[188] W. Hardaker, *Verfploeter Measurements of B-Root*, USC-LANDER dataset B-Root_Verfploeter-20170421 and B-Root_Verfploeter-20170515, `https://ant.isi.edu/datasets/`, 2017.

[189] W. de Vries, *RIPE Atlas Measurements of Tangled*, Tangled_Atlas-20170201 `https://traces.simpleweb.org/` and RIPE Atlas measurement IDs 7794356-7794364, 2017.

[190] ——, *Verfploeter Measurements of Tangled*, dataset Tangled_Verfploeter-20170201 and Tangled_Verfploeter-20170323, `https://traces.simpleweb.org/`, 2017.

[191] B-Root, *DITL 2017 for B-Root*, dataset DITL_B_Root-20170407 at `https://ant.isi.edu/traces/`, also available through `https://www.dns-oarc.org/`, 2017.

[192] ——, *Load for 2017-05-15 for B-Root*, dataset Load_B_Root-20170515 at `https://ant.isi.edu/traces/`, 2017.

[193] Hurricane Electric, *AS13335*, `https://bgp.he.net/AS13335`, Accessed: 2019-11-22.

[194] K. Hightower, B. Burns, and J. Beda, *Kubernetes: Up and Running Dive into the Future of Infrastructure*, 1st. O'Reilly Media, Inc., 2017.

[195] Apache, *Kafka*, `https://kafka.apache.org/`, Accessed: 2019-11-22.

[196] gRPC, `https://grpc.io`, Accessed: 2019-11-22.

Acronyms

qmin QNAME Minimization

ACL Access Control List
API Application Programming Interface
AS Autonomous System
ASN Autonomous System Number

BAF Bandwidth Amplification Factor
BCP Best Current Practice
BGP Border Gateway Protocol

ccTLD Country Code Top-Level Domain
CDF Cumulative Distribution Function
CDN Content Delivery Network
CEH Consecutive Equal Hops
CIDR Classless Inter-Domain Routing

DDoS Distributed Denial-of-Service
DHCP Dynamic Host Configuration Protocol
DITL Day in the Life of the Internet
DNS Domain Name System
DoS Denial-of-Service
DRDoS Distributed Reflection Denial-of-Service

ECS EDNS Client Subnet
ED Edit Distance
EDNS Extension mechanisms for DNS

GPDNS Google Public DNS
gTLD Generic Top-Level Domain

HMAC Hash-based Message Authentication Code

IANA Internet Assigned Numbers Authority
IATA International Air Transport Association

ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISP	Internet Service Provider
IXP	Internet Exchange Point
LOIC	Low Orbit Ion Cannon
NREN	National Research and Education Network
NTP	Network Time Protocol
PoP	Point-of-Presence
rDNS	Reverse DNS
RDoS	Reflection Denial-of-Service
RFC	Request for Comments
RIR	Regional Internet Registry
RIS	Routing Information Service
RRC	Remote Route Collector
RTT	Round Trip Time
SaaS	Software-as-a-Service
SMTP	Simple Mail Transfer Protocol
SSH	Secure Shell
TCP	Transmission Control Protocol
TLD	Top-Level Domain
TLS	Transport Layer Security
TTL	Time to Live
TWM	The Wayback Machine
UDP	User Datagram Protocol
UTC	Coordinated Universal Time
VP	Vantage Point

About the Author

Wouter de Vries was born in Hengelo, the Netherlands on the 3rd of November 1990. After finishing his secondary education in his birth town in 2008 he moved into the world of computers. In 2012 he obtained his Bachelor's degree at Saxion University of Applied Science in Enschede, and in that same town he earned his Master's degree from the University of Twente in 2014. He left the university to work in industry, but returned after half a year to pursue his PhD at the *Design and Analysis of Communication Systems (DACS)* group at the University of Twente.

Between July 2015 and December 2019 he worked on his research in the field of Anycast and the Domain Name System (DNS), mostly within SURFnet's Research on Networks (RoN) funding program. He was also partially funded by the EU project CONCORDIA (H2020 - #830927). During his PhD Wouter also spent 6 months at Cloudflare, to implement his ideas on anycast measurements on a large scale, which led to chapter 7 of this thesis. He was supervised by Prof. Aiko Pras, and co-supervised by Pieter-Tjerk de Boer and Roland van Rijswijk-Deij.

Publications by the Author

This is a comprehensive list of publications by the author, sorted in reverse chronological order.

- W.B. de Vries, S. Aljammaz, R. van Rijswijk-Deij. *Global-scale Anycast Network Management with Verfploeter.* In: IEEE/IFIP Network Operations and Management Symposium (NOMS), 20-24 April 2020, Budapest, Hungary.
- W.B. de Vries, R. van Rijswijk-Deij, P.T. de Boer, A. Pras. *Passive observations of a large DNS service: 2.5 years in the life of Google.* In: IEEE Transactions on Network and Service Management (TNSM), 2019. Based on conference paper at TMA 2018 (see below).
- W.B. de Vries, Q. Scheitle, M. Muller, W. Toorop, R. Dolmans, R. van Rijswijk-Deij. *A First Look at QNAME Minimization in the Domain Name System.* In: Passive and Active Network Measurement Conference, PAM 2019, 27-29 March 2019, Puerto Varas, Chile – **Best Dataset Award**.
- W.B. de Vries, R. van Rijswijk-Deij, P.T. de Boer, A. Pras. *Passive observations of a large DNS service: 2.5 years in the life of Google.* In: Network Traffic Measurement and Analysis Conference, TMA 2018, 26-29 June 2018, Vienna, Austra – **Best Open Dataset Award**.
- W.B. de Vries, R. de O. Schmidt, W. Hardaker, J. Heidemann, P.T. de Boer, A. Pras. *Broad and load-aware anycast mapping with verfploeter.* In: Internet Measurement Conference, IMC 2017, 1-3 November 2017, London, United Kingdom.
- G. Moura, R. de O Schmidt, J. Heidemann, W.B. de Vries, M. Muller, L. Wei, C. Hesselman. *Anycast vs. DDoS: Evaluating the November 2015 root DNS event.* In: Internet Measurement Conference, IMC 2016, 14-16 November 2016, Santa Monica, USA.
- W.B. de Vries, R. de O. Schmidt, A. Pras. *Anycast and its Potential for DDoS Mitigation.* In: Management and Security in the Age of Hyperconnectivity, AIMS 2016, 20-23 June 2016, Munich, Germany.
- W. de Vries, J.J. Santanna, A. Sperotto, A. Pras. *How Asymmetric Is the Internet?* In: Intelligent Mechanisms for Network Configuration and Security, AIMS 2015, 22-25 June 2015, Ghent, Belgium – **Best Paper Award**.