Rafael Luís Caldas Almeida

Classification of Load Balancing in the Internet

Advisor:

Ítalo Fernando Scotá Cunha

Belo Horizonte

June 2019

Ficha catalográfica elaborada pela Biblioteca do ICEx - UFMG

Almeida, Rafael Luís Caldas

A 447c Classification of load balancing in the internet / Rafael Luís Caldas Almeida. — Belo Horizonte, 2019.
xxii, 64 p.:il.; 29 cm.

Dissertação (mestrado) - Universidade Federal de Minas Gerais – Departamento de Ciência da Computação.

Orientador: Ítalo Fernando Scotá Cunha

1. Computação – Teses. 2. Redes de computadores - Teses. 3. Internet – Protocolos – Teses. 4. IPv6 – Teses. I. Orientador. II. Título.

CDU 519.6*22(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Classification of Load Balancing in the Internet

RAFAEL LUIS CALDAS ALMEIDA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

Ítalo Fernando Scotá Cunha
Prof. Ítalo Fernando Scotá Cunha - Orientador
Departamento de Ciência da Computação - UFMG

Renata Teixeira.
Profa. Renata Cruz Teixeira
Laboratório de Ciências de Informática, Redes e Comunicações - Inria

Luiz Filipe Menezes Vieira
Prof. Luiz Filipe Menezes Vieira
Departamento de Ciência da Computação - UFMG

Profa. Ana Paula Couto da Silva
Departamento de Ciência da Computação - UFMG

Prof. Antonio Alfredo Ferreira Loureiro
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 1 de Julho de 2019.

Resumo

Um roteador pode realizar balanceamento de carga e distribuir tráfego entre múltiplas rotas que têm o mesmo custo. Balanceamento de carga melhora a banda disponível, robustez a falhas e desempenho. Roteadores que fazem balanceamento de carga (chamados de *balanceadores de carga*) calculam qual enlace cada pacote deve ser encaminhado em função do *identificador de fluxo*, um subconjunto de campos nos cabeçalhos do pacote (e.g., endereços IP e números de porto).

Operadores de rede e pesquisadores dependem de ferramentas de medição que identifiquem balanceamento de carga e caracterizem seu comportamento. No entanto, avanços recentes em planos de dados programáveis, redes definidas por *software* e até mesmo a adoção de IPv6 suportam novas e mais complexas estratégias de balanceamento de carga, permitindo a definição de identificadores de fluxo incompatíveis com ferramentas existentes.

Neste trabalho, introduzimos o *Multipath Classification Algorithm* (MCA). Generalizamos o formalismo de rede utilizado para descrever balanceamento de carga e estendemos técnicas existentes para o cenário onde balanceadores de carga podem usar identificadores de fluxo compostos por combinações arbitrárias de bits nos cabeçalhos dos pacotes. O MCA detecta balanceadores de carga que técnicas existentes são incapazes de detectar, independente de quais bits compõem os identificadores de fluxo. Além disso, o MCA permite classificar o comportamento de cada balanceador de carga e seu impacto sobre o tráfego de aplicações. Para limitar o custo de medições usando MCA, propomos otimizações que reduzem o custo da classificação em 11% e o custo global em 6%, sem perda de acurácia. Nossa avaliação mostra que o processo de classificação acarreta um custo semelhante ao custo do processo de detecção, demonstrando a utilidade prática do MCA.

Por fim, utilizamos o MCA para coletar um conjunto de dados representativo de rotas na Internet para caracterizar o balanceamento de carga na Internet. Nossos resultados mostram que o balanceamento de carga na Internet hoje é mais prevalente e mais moderno em relação a caracterizações anteriores.

Abstract

A router may perform load balancing and distribute traffic across multiple routes that have the same cost. Load balancing improves available bandwidth, robustness to failures, and performance. Routers that perform load balancing (referred to as *load balancers*) compute the link a packet should be forwarded to as a function of the packet's *flow identifier*, a subset of fields in the packet's headers (e.g., IP addresses and port numbers).

Network operators and researchers rely on measurement tools to identify and characterize load balancing. However, recent advances in programmable data planes, software defined networks, and even the adoption of IPv6, support novel, more complex load balancing strategies. These strategies allow the definition of flow identifiers that existing measurement tools are incompatible with.

In this work, we introduce the Multipath Classification Algorithm (MCA). We generalize the network formalism used to describe load balancing and extend existing techniques to consider that load balancers may use arbitrary combinations of packet header fields for load balancing. MCA detects load balancers that existing tools cannot, regardless of the bits load balancers consider in flow identifiers. Furthermore, MCA classifies the behavior of load balancers and their impact on application traffic. We propose optimizations that reduce the classification cost by 11% and the overall cost by 6%, without loss of accuracy. Our evaluation shows that the process of classifying load balancers entails a cost similar to the cost of the detection process, demonstrating MCA is a practical tool.

Finally, we use MCA to collect a representative dataset of route measurements to characterize load balancing in the Internet. Our results show that load balancing is more prevalent and load balancing strategies are more mature than previous characterizations have found.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Understanding the structure of the Internet and the properties of Internet routes serves multiple goals. Network operators use Internet route measurements to troubleshoot failures [1], routing anomalies [2–4], bad performance [5], and misconfigurations [6]. Researchers study properties of Internet routes to develop new models of its topology and to design solutions to address limitations [7–11].

The Internet is a decentralized network composed of thousands of autonomous systems (ASes) that establish peering agreements to exchange routes and traffic with each other. Routers execute routing protocols that disseminate available routes and populate a routing table whose entries specify the next router (referred to as *next hop*) where traffic towards a destination IP prefix should be forwarded. Routing decisions are local, i.e., each router chooses its preferred route towards each IP prefix among the available routes and maintains its own routing table. For each packet, a router searches the longest prefix matching the destination address in its routing table, and forwards the packet to the corresponding next hop.

Between ASes, routing information is distributed by the Border Gateway Protocol (BGP). Within an AS, network operators employ an Interior Gateway Protocol (IGP) to distribute routes, e.g., Open Shortest Path First (OSPF) or Intermediate System to Intermediate System (IS-IS). Routing protocols select a preferred (e.g., cheapest or shortest) route to each destination prefix. If multiple routes have identical metrics (e.g., same cost or length), a routing protocol can install multiple routes towards a destination and perform load balancing. For example, load balancing can be configured automatically by mechanisms such as Equal-Cost Multi-Path (ECMP).

Internet traffic load balancing helps increase bandwidth, reliability, and reduce maximum link utilization. Like forwarding, load balancing decisions are also local. Routers that perform load balancing (which we call *load balancers*) compute which

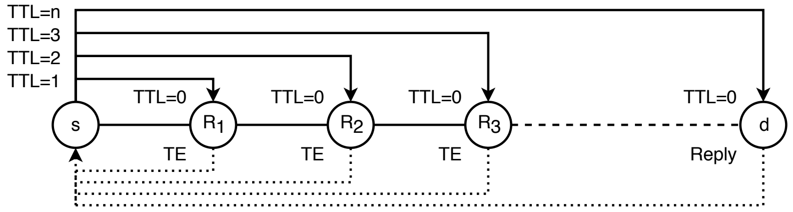


Figure 1.1: Traceroute sends TTL-limited probes towards a destination to identify interfaces between a source and a destination.

network link a packet should be forwarded to as a function of the packet's *flow identifier*, a subset of fields in the packet's headers (e.g., IP addresses and transport port numbers) [12].

Operators widely deploy load balancing, impacting most Internet routes [13,14]. Given the applicability and importance of route measurements, researchers and network operators need route measurement tools to characterize load balancing in the Internet.

In this thesis we develop and evaluate new methods to measure load balanced routes and perform a large-scale characterization study to uncover current load balancing practices in the Internet.

1.1 Measurement of Internet Routes

The standard tool for measuring Internet routes is traceroute [15]. Traceroute reveals interfaces in a route from a source to a destination in the Internet by sending specially-crafted packets (*probes*) towards the destination. Traceroute sends probes with limited values in the *time-to-live* field (TTL) in the packet header. Routers along the path decrement the value of the TTL field before forwarding a packet. When the TTL of a probe expires (i.e., gets decremented to zero), the router discards the probe and sends an ICMP Time Exceeded message to the source (the device executing traceroute). Traceroute identifies an interface by inspecting the source IP address of the ICMP Time Exceeded message.

Figure 1.1 shows an overview of traceroute. Traceroute sends probes with increasing TTL values until it receives a response from the destination. For ICMP Echo Request (*ping*) probes, traceroute identifies the destination when it receives an ICMP Echo Reply (*pong*) response; for UDP probes, traceroute identifies the destination when it receives an ICMP Port Unreachable response. Traceroute implementations also set a maximum TTL parameter, and stop the measurement if the destination is not reached by the maximum TTL; this stop condition is useful, e.g., in case of routing loops. Traceroute also stops probing if a number of consecutive routers (e.g., five routers) do not reply to probes; this stopping condition is useful if there is no route to the destination or if a firewall is blocking the measurement.

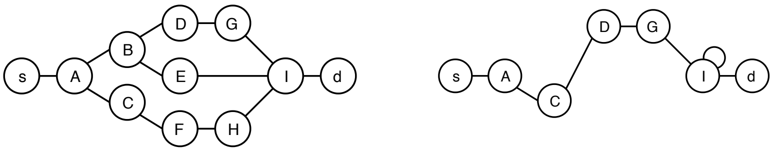
A key step in traceroute is matching ICMP Time Exceeded responses from routers to the original probes. This step is critical to identify the hop distance (the TTL value) at which each interface on the path is reached. This matching is possible because an ICMP Time Exceeded message encapsultes the header of the (expired) probe that triggered the message. Traceroute inserts a *matching key* in each probe. When an ICMP Time Exceeded message is received, traceroute inspects the encapsulated header and uses the matching key to identify the probe that triggered the message (and the hop distance of the interface that sent it).

1.2 Traceroute and Load Balancing

Load balancing in Internet routes can lead to anomalies in traceroute measurements. Classic traceroute implementations use the destination port of the UDP protocol to store a probe's matching key. Unfortunately, the destination port is often considered for load balancing in the Internet; in other words, the destination port is often part of a packet's flow identifier. When probes sent by classic traceroute, each with a different matching key in the UDP destination port, trigger load balancing, they may be forwarded to different next hops through different links. This causes measurement artifacts that lead to the inference of inexistent links, loops, and cycles [13, 16, 17]. Traceroute is also incapable of discovering the complete set of interfaces and links in the route towards a destination. Measurement artifacts and incompleteness negatively impact tasks such as network management and troubleshooting [13, 16–19]. Next, we discuss these limitations and existing techniques designed to address them.

Inference of False Links

Because traceroute measures a route iteratively and since classic implementations do not control probe flow identifiers, load balancers along the route may forward probes to different next hops, leading to the inference of false links. For example, Figure 1.2 shows an example route with load balancing (a), where A and B are load balancers, and a possible incomplete traceroute output (b). In the example, a probe with TTL = 2 may expire at C and a probe with TTL = 3 may expire at D, leading to



(a) Route with load balancing (b) Traceroute output

Figure 1.2: Possible traceroute output in the presence of load balancing

the inference of a false link between *C* and *D*. Similarly, when routes have branches with different lengths, traceroute may infer loops when TTL-limited probes cross different branches over multiple iterations. In the example, a probe with TTL $= 4$ may traverse *A*, *B*, *E*, and expire at *I*; another probe with TTL $= 5$ may traverse *A*, *C*, *F*, *G*, and also expire at *I*. This would lead to the inference of a self-loop at interface *I*. Finally, the traceroute output is not representative of the actual route.

Paris traceroute is an extension of traceroute that stores a probe's matching key in header fields not normally considered for load balancing, i.e., not included in flow identifiers, like the IP identifier field [12]. This allows Paris traceroute to carefully control values in packet header fields to keep probe flow identifiers constant (§2.2), avoiding triggering load balancing and inferring of false links.

Missing Nodes and Links

Classic traceroute and Paris traceroute do not identify all links and interfaces in a route. In Figure 1.2(b), traceroute misses nodes *B*, *E*, *F*, *H* as well as their links. Increasing the number of probes that classic traceroute sends to each TTL can lead to the discovery of additional interfaces, but the lack of control over flow identifiers prevents inference of links between interfaces. Paris traceroute avoids load balancing and the inference of false links, but identifies a single branch between the source and the destination.

The Multipath Detection Algorithm (MDA) [20], is a probabilistic algorithm that identifies load balancers in Internet routes. MDA systematically varies flow identifiers to trigger load balancing and identify all interfaces, as well as links between interfaces, in branched routes [20]. By controlling the number of probes sent to each TTL, MDA can calculate the probability of missing interfaces or links to within a configurable confidence level.

1.3 Limitations of Current Techniques

Recent advances in programmable data planes, software-defined networking (SDN), and even the adoption of IPv6 lead to increasingly complex load balancing deployments [21–31]. Moreover, new router implementations support load balancing using flow identifiers composed of arbitrary set of packet header fields [32, 33]. Existing MDA implementations, however, only vary transport port numbers and the ICMP checksum [20, 34]. This prevents MDA from identifying load balancers when they do not consider port numbers for load balancing. Load balancing misidentification may be aggravated over time as router implementations evolve and the adoption of programmable data planes, SDN, and IPv6 increases.

MDA detects load balancers in Internet routes but does not classify their behavior, i.e., it does not identify which bits or header fields in the packet headers are used for load balancing. This information is relevant as different load balancing behavior can have different impact on applications and traffic. Although previous work have applied ad-hoc extensions to MDA to classify load balancers as per-packet, per-flow, or per-destination [13, 14, 35], these three predefined classes are not comprehensive and the classification method does not generalize to other load balancer classes. For example, IPv6 load balancers may rely on the Flow Label header field for load balancing [36]. Moreover, these ad-hoc extensions do not provide bounds on the accuracy of their classifications.

1.4 Contributions

In this thesis, we extend the existing formalism [20] to consider that load balancers may use arbitrary combinations of bits in the packet header for load balancing. We introduce the Multipath Classification Algorithm (MCA), a probing algorithm that identifies the set of bits in the packet header used by load balancers. We release an open-source implementation of MCA, and Route Explorer, a web-based visualization and route analysis tool.

MCA and MDA significantly increase the probing cost of route measurements [20, 37]. We present optimizations applicable to both MCA's and MDA's probing algorithms that reduce the average number of probes sent by 6% for IPv4 and 8% for IPv6, without any loss of accuracy. This reduction may help adoption of MCA and MDA measurements (e.g., by Ark or RIPE Atlas [38, 39]).

We conduct large-scale campaigns of MCA measurements of IPv4 and IPv6 routes in the Internet using a diverse set of vantage points. We characterize the prevalence and deployment of load balancing in the Internet today, and reappraise previous characterizations of load balancing [13, 14].

Our results indicate that load balancing is more prevalent in Internet routes and their configurations are more complex than previously reported. Our reappraisal of previous results also reveals that IPv4 load balancing is still more prevalent than IPv6 load balancing. We observe, for the first time, load balancers considering ToS/traffic class and flow label fields for load balancing. Finally, we detect and report on networks employing inter-domain load balancing.

This work improves the foundations upon which we build tools to identify load balancing in the Internet. Our implementation of MCA provides more accurate information than existing tools, and can be useful in network characterization studies, particularly those concerned with traffic engineering and reliability to failures.

Chapter 2

Definitions and Background

In this chapter we lay the foundations for the rest of this thesis. We first define terminology (Section 2.1). We then discuss common load balancing practices and the possibility of using arbitrary set of packet header fields for load balancing (Section 2.2). We describe the Multipath Detection Algorithm (MDA) in detail (Section 2.3) and discuss its limitations (Section 2.4).

2.1 Definitions

We follow the same notation as in previous work [40]. At any given time, the connectivity between a fixed source s and a destination d is realized by its *current route*. A *route* can be *simple*, consisting of a sequence of IP interfaces from s toward d, or *branched*, when one or more *load balancing* routers (LB) are present, giving rise to multiple overlapping sequences (called "multi-paths" in [20]). Thus a route is a directed graph with interface-labelled nodes. A route can be a sequence that terminates before reaching d, due to routing changes or the absence of a complete route to the destination.

We define a *diamond* as the set of all interfaces between an LB and its *join point*, the interface where all the sub-branches originating from the LB first rejoin. An *outer* diamond is one that contains other *nested* LBs, and therefore their own diamonds. We define an *outermost diamond* as one which is not contained in some other diamond. Figure 2.1 shows an example branched route where the LB A defines an outermost diamond with join point G, containing a diamond defined by nested LB B, also with join point G. A branched route can have multiple (nonoverlapping) outermost diamonds.

By *hop-set* or *hop* we define the set of interfaces found at some fixed distance,

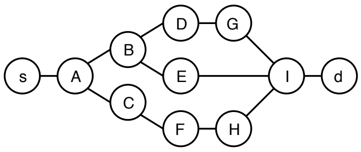


Figure 2.1: Example of a branched route.

or *radius*, from the source. We denote the hop-set at radius r in route p as $p[r]$. Conversely, for some valid hop-set h in route p, we let $p\langle h\rangle$ denote the radius of h. Thus $h = p[p\langle h\rangle]$. In Figure 2.1, $p = [s, A, \{B, C\}, \{D, E, F\}, \{G, H, I\}, \{I, d\}, d]$ is a route with three branches. The first hop is $p[1] = \{A\}$, the second hop is $p[2] = \{B, C\}$, and hop $\{G, H, I\}$ is found at radius $p\langle\{G, H, I\}\rangle = 4$. Assuming loop-free routes, hops are unique, although individual interfaces can be found in multiple hops when route branches have different lengths (e.g., I and d).

2.2 Load Balancing

Hashing different IP header fields allows control of a trade-off between the granularity of load balancing and impact on traffic. Common load balancing configurations include [41]:

- **Per-destination.** Hashes source and destination IP addresses to compute next-hops. All packets from a source to a destination will traverse the same branch.
- **Per-flow.** Hashes source and destination IP addresses, ports (or the ICMP checksum), and the protocol identifier. May consider the IPv6 flow label. Packets belonging to the same connection will take the same branch, but different connections between a source and a destination may take different branches.
- **Per-packet.** The next-hop is chosen at random or by a round-robin mechanism. Packets in the same connection may take different paths and be delivered out of order, degrading TCP performance.

Some router vendors and SDN-based load balancing may allow the hashing of arbitrary header fields, e.g., source and destination ports [32,33,42], creating different load balancing behavior classes.

2.3 Multipath Detection Algorithm

The Multipath Detection Algorithm (MDA) is a probabilistic algorithm to identify all routers and links in branched routes [20]. MDA proceeds hop-by-hop, and identifies whether each interface i in a hop h is a load balancer as follows.

Each interface i in hop h in a route (other than the destination) forwards traffic to at least $n = 1$ interface in the next hop, i.e., the hop at radius $p\langle h \rangle + 1$. MDA iteratively tests whether interface i forwards packets to $n + 1$ interfaces in the next hop. On each iteration, MDA sends multiple probes through i to i's next hop, each probe with a different port number to trigger load balancing (if any). The number of probes to send in an iteration is a function of the number n of interfaces identified in the next hop so far and on a configurable confidence level that controls the probability of missing a next hop. The value of n and the number of probes to send is updated according to probe replies at the end of each iteration. MDA stops the iteration and proceeds to the next interface or hop when no new interfaces are found in i's next hop.

To identify the branched route in Figure 2.1 with a confidence level $\alpha = 0.95$, MDA starts sending 9 probes with varying flow identifiers to A's next hop. If MDA receives responses from different interfaces,[1] MDA labels A as a load balancer, and performs an additional iteration to check for additional interfaces in A's next hop. To check if A's next hop has three interfaces with $\alpha = 0.95$, MDA sends 8 additional probes (for a total of 17 probes). As no new interfaces are found in A's next hop, MDA proceeds to the next hop. This process repeats until MDA reaches the destination.

2.4 Limitations of the Multipath Detection Algorithm

Existing MDA implementations vary a single field in packet headers, usually port numbers [34, 35]. This approach, however, cannot identify load balancers that do not consider ports.

Some implementations have included an ad-hoc extension to MDA to classify load balancers. For example, previous work have applied the following steps to classify load balancers [13, 35]. First, send multiple probes with fixed port numbers (identical flow identifiers) and classify any identified load balancers as per-packet. Second, run MDA varying port numbers and classify any additional load balancers

[1] Assuming uniform load balancing, the probability that all probes take the same branch and are replied by the same interface is $0.5^9 = 0.001953$.

as per-flow. Finally, run MDA varying the last bits of the destination IP address and classify any additional load balancers as per-destination.[2] This approach, however, relies on the property that per-destination, per-flow, and per-packet load balancers will forward packets with different port numbers on different branches. This approach does not generalize to other load balancer types and does not provide bounds on the accuracy of classifications.

These problems are aggravated by IPv6 load balancing, which may rely on the IPv6-specific flow label field introduced to facilitate load balancing [36], and SDN, which enables network administrators to configure load balancing over a large set of packet header fields [21]. More generally, there has been no systematic studies of which fields are used for load balancing and how they vary across routers.

[2] Most IPv4 and IPv6 prefixes routed in the Internet are less specific than /24 and /48 bits, respectively [43]. Varying the least significant bits does not affect the routing table entry used to forward packets.

Chapter 3

Related Work

In this chapter we present a review of work related to Internet route measurement. We discuss limitations of traceroute orthogonal to load balancing (Section 3.1) and alternative measurement techniques that extend or complement traceroute (Section 3.2). We then present measurement platforms and ongoing efforts to monitor the Internet's topology (Section 3.3), and close with a summary of previous Internet characterization studies (Section 3.4).

3.1 Limitations of Traceroute

Traceroute works by leveraging the ICMP Time Exceeded messages routers send in response to TTL-expired probes. As traceroute is not a standard and the ICMP standard leaves several aspects open to interpretation, vendors and network administrators can configure devices with different behaviors when dealing with TTL-expired packets. This imposes limitations on traceroute measurements.

In this thesis we focus on limitations of traceroute measurements in the presence of load balancing, discussed in Section 1.1. We next discuss work related to other limitations of traceroute when measuring Internet routes.

Firewalls and Anonymous Routers

Network operators can configure routers to not respond to traceroute probes (e.g., to avoid malicious attacks and keep their network topology hidden) and firewalls can block ICMP Time Exceeded messages, thus creating *anonymous routers*. Although anonymous routers do not respond to traceroute probes, they do decrement the TTL and forward probes. As a result, anonymous routers appear in traceroute measurements as a hop without any response; their interface's IP address are unknown.

Most traceroute implementations output an asterisk (*) as a placeholder for anonymous routers. The task of assigning IP addresses to anonymous routers in traceroute measurements is known as anonymous router resolution. Gunes et al. [44] propose a graph data mining approach that identifies subgraphs of common structures in a topology graph to resolve anonymous routers.

Firewalls can also block traceroute probes. This prevents traceroute probes from getting to the destination, making it more difficult to trace the complete path. Depending on the application running on end hosts, network administrators can configure firewalls to only allow traffic of certain protocols and ports, e.g., only allow TCP traffic on port 443 towards a Web server. Trying to increase the probability that probes reach the destination, some traceroute implementations (e.g., tcptraceroute [45]) allow probing using different protocols, including ICMP and TCP.

Researchers have also proposed techniques that include traceroute probes in legitimate Internet traffic [46] to avoid filtering at firewalls. For example, Morandi et al. [47] propose a technique that reveals the route that traffic from a real application (e.g., a Web browser or Netflix) traverses by injecting TTL-limited packets on the same connection as the legitimate traffic.

ICMP Response Rate Limits

ICMP error messages are generated by the router's general-purpose CPU, which has limited capacity and bandwidth (packets handled by the router's CPU are said to traverse the *slow path*). Network operators enable ICMP rate-limiting to prevent abuses, such as DoS attacks, of a router's CPU. Some vendors (e.g., MikroTik [48]) rate-limit ICMP responses by enforcing a minimum time interval between them. Packets that would trigger an ICMP response before the minimum time interval are simply discarded and do not elicit a response from the router. More flexible implementations (e.g., Cisco [49]) use a token bucket for generating ICMP responses. A token bucket uses a counter that is incremented at a specified rate and decreases whenever an ICMP response is generated; a token bucket's counter has a predefined maximum value that allows short bursts of ICMP responses.

Routers that rate-limit ICMP messages may fail to respond to traceroute probes, similarly to an anonymous router. Ravaioli et al. [50] studied ICMP rate limiting for ICMP Time Exceeded messages and identified that rate-limiting is often *on-off*: routers respond to all probes for a period of time and then remain silent for another period of time. IPv6 rate-limiting may be more restrictive than IPv4, since RFC4443 explicitly states that IPv6 routers must rate-limit ICMP responses [51,52].

Implementations of traceroute often allow users to increase the number of probes to be sent to each radius and the time interval between each probe. Measurement techniques that probe the same radius many times may trigger the rate-limiting; which is usually circumvented by waiting longer between probes.

Hidden Routers

Network operators can also configure routers to not decrement the TTL of forwarded packets, thus creating *hidden routers*. Hidden routers are invisible to traceroute measurements and cause the inference of false links between interfaces: two interfaces connected via an invisible router will appear directly connected in traceroute measurements. Marchetta et al. [18, 53] proposed a technique to detect hidden routers and estimated that at least 6% of IP paths traverse hidden routers.

3.2 Measurement Techniques

In this section we discuss alternative techniques for measuring Internet routes, most of them based on and improving aspects of traceroute. First, we talk about techniques for reducing probing cost. Next, we discuss differences in IPv4 and IPv6 route measurements. Finally, we discuss complementary data that are commonly employed in Internet topology studies.

3.2.1 Reducing Probing Cost

Operators and researchers deploy measurement tools in topologically-distributed vantage points aiming to maximize the coverage of Internet measurements. But even with distributed vantage points, the scale of the Internet imposes challenges to the mapping of its topology. Traceroute may send dozens of probes to measure a single Internet route and discovering the complete route (e.g., when running MDA to identify load balancers) comes with an elevated probing cost. For instance, MDA executions frequently send ten times more probes than Paris traceroute, and a few measurements send a hundred times more.

Efforts such as CAIDA's Ark [38] and M-Lab [54] maintain up-to-date maps of the Internet topology by periodically measuring routes towards a large set of destination addresses (e.g., one address at each /24 prefix). Each cycle of measurements usually takes days to conclude because of the high probing cost. These periodic measurements may waste probes on paths that have not changed.

To reduce probing cost when issuing measurements to keep an up-to-date map of the Internet, Cunha et al. [40] propose DTRACK, a system that predicts route stability and focuses the probing budget on paths that are more likely to have changed since the last measurement. Another challenge in topology mapping studies is choosing which destinations to probe. Beverly et al. [55] propose a technique that dynamically chooses destinations to maximize network discovery while keeping the probing cost (number of destinations) in check.

Mapping the topology of a target network benefits from having many measurements through the target network. However, blindly issuing measurements will result in high probing cost and redundant measurements. RocketFuel [56] combines techniques to reduce probing cost when mapping a target network. RocketFuel estimates on which routers traceroutes will ingress and egress the target network, and chooses traceroutes whose (ingress, egress) pair has not been measured yet. Donnet et al. propose Doubletree [57] to further reduce redundancy in Internet route measurements. Destination-based routing in the Internet means that routes toward one destination have a tree-like structure. Doubletree takes advantage of this tree-like structure to reduce redundant probes. In particular, measurements from one source to many destinations probe routers close to the source repeatedly (once towards each destination), and measurements from multiple sources to one destination probe routers close to the destination repeatedly (once from each source). Doubletree shares information between vantage points to avoid these redundant probes.

When identifying load balancers in Internet routes, MDA needs to search probes traversing a specific interface to identify its next hops. This process of searching probes accounts for a significant part of MDA's probing cost. MDA-Lite [37] makes assumptions about load balancing to significantly reduce the number of probes that MDA needs to search for. Although MDA-Lite can sometimes identify violations of its assumptions and revert to the original MDA, it incurs loss of accuracy when assumptions are violated and the violation is not detected. In this work, we identify and classify load balancers in Internet paths. Like MDA, the probing cost of MCA's classification is high. In this work we propose optimizations that allow us to save probes without any loss of accurary.

Finally, although probing the Internet topology takes significant probing, it may also take significant time. However, as routes can change over time, collecting a snapshot of the topology requires fast measurements. Inspired by ZMap [58], Beverly proposed Yarrp [59], a tool that combines two ideas to speed up traceroute measurements toward a large set of destinations. First, Yarrp stores all information necessary to process a response in the probe's header (which gets encapsulated in the response), avoiding the need to maintain local state. Second, Yarrp uses a block cypher to generate a pseudorandom probing sequence that deterministically covers IPv4 × TTL space while distributing probes across destination networks and TTLs, allowing very high probing rates.

3.2.2 Complementary Data

To better understand Internet routes, traceroute measurements are frequently combined with data from different sources. In this work we augment our measurements with reverse DNS (PTR records) for IP addresses, IP-to-ASN mapping provided by Team Cymru [60], network types extracted from PeeringDB [61] and AS relationships as inferred by CAIDA [62].

IP-to-ASN Mapping

Each autonomous system (AS) in the Internet receives an identification number (ASN) from a Regional Internet Registry (RIR) to be used in the exchange of routing information (e.g., through BGP). IP-to-ASN mapping is often used in topology studies to identify the ASes owning the detected IP interfaces. This allows, e.g., the use of traceroute to infer Internet routes at the AS level.

Routers share routing information through BGP messages, which consist of reachability information of a destination IP prefix through a certain router (next hop). Routers populate a BGP table and learn the AS-path (a sequence of ASNs) through which each destination prefix can be reached. The last ASN in an AS-path is (usually) the origin ASN for the advertised prefix, which allows mapping the prefix to its origin. Team Cymru's IP-to-ASN service [60], which we use to map IP addresess to ASN in this thesis, gathers data from both public [63, 64] and private BGP feeds.

Traceroute reveals Internet routes at the interface level but users are often interested in knowing which ASes routes traverse. Mapping IPs in traceroute measurements to ASNs is a common approach. Two challenges complicate this approach: anonymous routers and IP addresses without a mapping. Chen et al. [65] propose a multi-pass algorithm that combines BGP information and several heuristics and attempts to resolve the ASN of unresponsive routers and unmapped IPs.

AS Relationships and AS Business Types

Autonomous systems establish contractual relationships for exchanging routes and traffic with each other. Relationships are commonly classified into customer-to-provider (or provider-to-customer, *p2c/c2p*), when a customer network pays a provider network for traffic sent between them; peer-to-peer (*p2p*), when networks exchange traffic free of charges; and sibling-to-sibling (*s2s*) networks belonging to the same administrative domain exchange traffic free of charges and sometimes implement uncommon routing policies.

Knowing contractual relationships between ASes is useful to studying performance, robustness, and evolution of the Internet on technical or economical aspects. Dimitropoulos et al. [66] propose heuristics to infer AS relationships based on BGP paths and the valley-free model, which says that AS paths should not traverse a c2p or p2p link after traversing a p2c or p2p link [67]. Luckie et al. [8] also use BGP paths to infer AS relationships, but do not assume presence of valley-free paths. Giotsas et al. [9] use BGP paths, traceroute, and geolocation to infer hybrid relationships (when two ASes participate in different relationships depending on the interconnection point) and partial transit relationships (when an AS offers transit to another for a subset of destinations in the Internet).

ASes are usually classified in types depending on their business; common types include content provider networks (CDNs), Internet service providers (ISPs), Internet exchange points (IXPs), network/transit providers, and stub/enterprise networks. These types are commonly employed in studying the evolution of the Internet because different network types present different peering footprints and growth patterns. Furthermore, traffic properties vary significantly across network types, e.g., inbound/outbound traffic volumes, daily distribution, and contractual relationships with other ASes. Dimitropoulos et al. [68] and Dhamdhere et al. [69] use data from Internet Routing Registries (IRRs) and RouteViews [63] to identify differences across AS types and propose a classification algorithm. Today, some networks volunteer their (self-defined) business activity on PeeringDB [61]; Lodhi et al. [70] found that PeeringDB offers up-to-date information on AS business types and networks covered by PeeringDB are representative of Internet transit, content and access networks.

IP Alias Resolution

Traceroute measures routes at the granularity of IP interfaces. Sometimes, information about which interfaces belong to which routers is important, e.g., to build network topology maps [56]. Researchers have developed several techniques to perform IP alias resolution, the process of identifying if two given IP addresses belong to the same device [56, 71–75]. Techniques for IP aliasing usually exploit differences or particularities of router implementations, and are thus unreliable and complementary. Illustrative examples are Ally [56] and RadarGun [71], techniques for IPv4 alias resolution that work by observing the IP identifier field of responses coming from different IP interfaces: if a pair of responses from two IP interfaces have correlated, increasing values of IP identifiers, they are inferred to belong to a single device. MIDAR [72] combines alias resolution techniques based on the IP identifier with a scheduling algorithm to scale IP aliasing to Internet-wide topology measurements from the Ark platform [38].

Most IPv4 alias resolution techniques do not work on IPv6. For example, the IP identifier field is not present in the IPv6 header, making Ally, RadarGun, and MIDAR ineffective. Luckie et al. [76] propose Speedtrap, an IPv6 alias resolution technique that probes different IPv6 interfaces with ICMPv6 Packet Too Big messages, which cause IPv6 routers to send fragmented responses. IPv6 fragment extension headers include an identification field that is used to reassembly the original packet. Speedtrap then applies a technique similar to Ally's on IPv6 fragment extension headers to identify IP aliases.

Measurements of Reverse Routes

Traceroute is only capable of measuring the forward path that packets take from a source towards a destination. In some cases, however, network operators and researchers are interested in discovering the reverse path (from the destination to the source). Measuring the reverse path with traceroute requires control over the destination host to run traceroute towards the source host. Unfortunately, operators and researchers rarely have control of arbitrary destinations on the Internet. As an alternate approach, operators and researchers often resort to issuing a traceroute measurement towards the source from a vantage point topologically close to the destination in an attempt to discover the reverse path. Typical vantage points used include those of measurement platforms (§3.3) and looking glass servers [77]. Katz-Bassett et al. [78] designed Reverse Traceroute, a system that uses a set of vantage points probes combining techniques such as source spoofing and the use o IP options to infer the reverse path without control of the destination host. Unfortunately, Reverse Traceroute cannot measure all reverse paths in the Internet.

3.2.3 Measurements of the IPv4 and IPv6 Internet

Due to standardization and implementation differences between IPv4 and IPv6, measurement techniques created for IPv4 may not work on the IPv6 Internet. For example, some techniques for IP alias resolution depend on the identification field of the IPv4 header [56, 71, 72], not present in the IPv6 header. Researchers have proposed alias resolution techniques for IPv6, based on IPv6 extension headers [76] and ICMPv6 address unreachable messages [79]. Similarly, other techniques use IPv6-specific headers, (e.g., for border router detection [80] and router outage detection [81, 82]) and do not work on IPv4. Although identifying and classifying IPv4 and IPv6 load balancing requires specific probing, we provide a formalism to control probing that generalizes to both protocols.

3.3 Measurement Platforms

To make Internet measurements representative and scalable, researchers depend on a large and diverse set of vantage points [83]. Maintaining such infrastructure is a challenge. Fortunately, a number of efforts maintain Internet measurements platforms and release publicly-available datasets. While some platforms execute a predefined set of measurements, more open platforms allow researchers to deploy their own tools. Below we present a few Internet measurement platforms.

The Center for Applied Internet Data Analysis (CAIDA) [38], operates Archipelago (Ark), a measurement platform that hosts monitors distributed globally. CAIDA conducts traceroute measurements towards one IP destination in each /24 network and releases publicly-available datasets. These datasets have been used by researchers to study Internet infrastructure and its evolution. The Measurement Lab (M-Lab) [54] operates different measurements (including traceroute and Reverse Traceroute) to study Internet performance. M-Lab also offers tools to explore and analyze measurement data.

Some platforms offer access for users to execute a predefined set of measurement tools from globally distributed vantage points. RIPE NCC, for example, operates RIPE Atlas [39], a network of thousands of monitors distributed around the globe that allow users to run measurements (including ping and traceroute) towards user-defined destinations. As an incentive for global distribution and user support, anyone can host RIPE Atlas probes, and hosts earn credits that can be used to schedule their own measurements on the platform.

Some platforms provide additional flexibility, and let researchers deploy their own measurement tools and experiments. This is the case of PlanetLab [84], a network of hundreds of globally distributed nodes, usually hosted by academic institutions. Researchers submit experiment proposals and receive access to a set of virtual machines in a subset of nodes, where they can deploy custom measurement tools.

Complementary to these active measurement platforms, RouteViews [63] and RIPE RIS [64] operate a network of passive BGP route collectors. Route collectors are routers that establish BGP sessions with dozens of routers in other networks to collect BGP update messages. These platforms make these BGP update messages publicly-available, which have been used in several studies of the Internet (e.g., [69, 85]), and are the basis for IP-to-ASN mapping.

Seeking representativeness and scalability, we deploy our measurements in a diverse set of 31 vantage points, distributed across 16 countries in 5 continents. Our vantage points include 19 monitors hosted by cloud providers (Linode, Vultr and Digital Ocean), 11 monitors hosted by CAIDA's Ark, and one monitor hosted by our home institution (UFMG). We resort to this approach because PlanetLab does not support IPv6 and other measurement platforms do not support execution of custom tools such as MCA. We also make our dataset public-available.

3.4 Characterization of Internet routes

Paxson [86] was the first to use traceroute to study stability of Internet routes. By conducting a large number of traceroute measurements from different Internet hosts, he investigated the prevalence and persistence of routes. Paxson also studied anomalies such as loops, erroneous routing, and infrastrucure failures. Following Paxson's work, researchers have used traceroute to study properties of Internet routes, propose new models, and develop solutions to address limitations [7–11].

As discussed above (Section 3.3), Internet characterization studies need a diverse set of vantage points to achieve good coverage of the Internet's topology. The other side of this coin is choosing sets of destinations. Topology monitoring studies choose traceroute destinations to achieve goals such as maximizing coverage [1, 10, 11, 87, 88] or focusing on specific networks [40, 56, 88–90]. IPv4 hitlists can be generated, e.g., by scanning the entire address space. While the entire IPv4 address space can be scanned in under 45 minutes [51, 58], scanning does not scale to IPv6. Researchers have proposed techniques to identify active IPv6 addresses resolving DNS names [91–93], passively monitoring IPv6 traffic [91], identifying router addresses with traceroute [91], and generating candidate addresses based on previous known IPv6 hosts [94,95]. For our experiments, we generate IPv4 and IPv6 target lists by resolving domains on Internet 'toplists' and from publicly-available Internet hitlists.

The transport protocol used in traceroute probe packets impact interface discovery and the probability of sucessfully reaching the destination, due to protocol-specific handling in intermediate routers and forwarding policies [13, 96]. Our implementation provides flexibility by supporting TCP, UDP, and ICMP probing. We use this flexibility to revisit previous results showing that ICMP probing reaches destinations more often than UDP and TCP [96] (§6), and that some per-flow load balancers do not perform load balancing on traceroute ICMP probes when varying only the ICMP checksum [13] (§8.2).

Augustin et al. [13,35] characterized load balancing on the IPv4 Internet through MDA measurements. The authors were the first to quantify load balancing in the Internet, and found it to be prevalent. Augustin et al. also characterized several load balancer properties. We are the first to study load balancing in the IPv6 Internet [14]. In this work, we characterize load balancing in the IPv4 and IPv6 Internet and compare our observations with those of previous studies.

Chapter 4

Load Balancer Model

Load balancing at a given router has two main aspects: (i) *when* load balancing is performed, and (ii) the *type* of load balancing when it is performed.

Aspect (i) is a function of whether multiple routes (through different next hops) to a destination prefix are known, the incoming link at the router, and the policy in place. To capture these dependencies, we test for LB behavior at the granularity of $\langle$ingress link, IP interface, destination prefix$\rangle$ triples, where the interface and upstream IPs are those returned by traceroute, used here, as usual, as surrogates for the true, unknown interfaces.

Aspect (ii) is a function of the router's forwarding decision algorithm. We model this as an ideal hash function over a *hash domain* consisting of a subset, in general not contiguous, of packet header bits. Each possible bit-field value over the domain is mapped to one of the available next hops. We consider *hash domains* are LB-specific.

The goal of the above model is to capture very general LB behavior. For (i), it allows load balancing to be triggered on some paths but not others, and the load balancing type to be path dependent. For (ii), it includes the classic LB types mentioned earlier, but allows for much richer structure that router vendors or SDNs may be implementing today or in the future. Per-packet LBs are included as the special case of an empty hash domain.

The generality of arbitrary hash domains may seem intractable at a practical level. There are two important reasons why this is not the case which we exploit in the sections below:

1. *LB discovery:* By sending probes which vary all header bits (within some given target set), the presence of a LB can be detected *without needing to know* the exact hash domain. Moreover, if we assume the hash function maps uniformly

over the next hops, then the same stopping rules and statistical guarantees used for MDA [20] still apply *regardless* of the unknown hash domain.

2. *LB classification:* It is not necessary to fully determine the hash domain to gain useful results. For example, under the uniform hash assumption, if we vary a single field (e.g., origin port) uniformly, we will induce a uniform sampling of the next hops provided the hash domain and the field have a non-empty intersection. Thus, the involvement of fields in LB decisions can be determined without a full resolution at the bit level.

Hash functions are employed to approximate random selection of next hops, but are actually deterministic. This can result in undesirable *polarization* effects when nested LBs are of the same type. To describe this we define, for each interface i in hop $p[r]$ of route p, the set $\text{flows}(i, r)$ of *flow identifiers* (bit-field values in the packet header) with which $i \in p[r]$ can be reached. For example in Figure 2.1, if A and B were polarized, then all flows that A sends to B would be sent on to D, that is $\text{flows}(B, 2) = \text{flows}(D, 3)$. It would then be *impossible* to observe interface E, regardless of the probing strategy used. Polarization has been observed in real routers and flagged as undesirable, as it prevents effective use of all available routes, and modern implementations include mechanisms to avoid it. For example, Cisco and Juniper hash a router-specific identifier together with the packet's flow identifier, while Arista and Cumulus allow configuration of a seed for the hash function.

Another challenge to identifying load balancing is dealing with non-uniform hash functions. These are sometimes deployed to better align link capacity and traffic volumes [22, 97]. For example, in Figure 2.1, router A could send 2/3 of the hash domain towards B and the remainder towards C. Our techniques can identify and classify non-uniform load balancers, however our bounds on the probability of error assume uniformity.

Chapter 5

Multipath Classification Algorithm

In this chapter we present the components of our Multipath Classification Algorithm (MCA): an extension to MDA's probing algorithm for enumerating next hops considering that routers can use arbitrary hash domains (§5.1), our probing algorithm for classifying load balancers (§5.2), and optimizations to reduce the probing cost of both probing algorithms (§5.3).

MCA receives as input the destination IP address d to trace; a confidence level α_d to bound the probability of detection errors, a confidence level α_c to bound the probability of classification errors; and a set $\mathcal{B}$ of *target* bits in the packet header defining the scope of LB detection and classification. We call the set of header fields that intersect $\mathcal{B}$ the *target* header fields, denoted $\mathcal{F}$.

MCA measures a route p hop-by-hop in increasing radius order. At each radius r, MCA first executes a modified version of MDA to enumerate next hops of routers in $p[r]$ that perform load balancing involving bits in $\mathcal{B}$ (§5.1). MCA then sends additional probes to classify the type of any LBs found (§5.2). During both detection and classification phases, MCA employs optimizations to reduce the probing cost (§5.3). After reaching the destination or stopping conditions (§5.4), MCA outputs a directed graph representing the route discovered and the classification of each LB.

5.1 Load Balancer Discovery

MCA computes the *number* of probes to send through an interface to enumerate next hops following MDA (§2.3, [20]). The novelty of MCA is its probe generation approach designed to guarantee detection of load balancers with arbitrary hash domains. MCA generates probe *contents* such that the values of the multisets defined by any (not necessarily contiguous) combination of bits in $\mathcal{B}$ have high entropy.

More specifically, denote $\phi_1, \phi_2, \ldots, \phi_n$ the flow identifiers (i.e., the values for bits in $\mathcal{B}$) generated for the first n probes; we generate a new flow identifier ϕ_{n+1} such that

$$\max_{\phi_{n+1}} \sum_{m \in \mathcal{B}^{\star}} \boldsymbol{H}(\Phi_{m,n+1}), \tag{5.1}$$

where $\mathcal{B}^{\star}$ is the power set of $\mathcal{B}$ (m can be thought of as a bitmask); $\Phi_{m,n+1}$ is the probability mass function of the values of bits m in the $n+1$ flow identifiers, i.e., $\{\phi_1 \wedge m, \phi_2 \wedge m, \ldots, \phi_{n+1} \wedge m\}$; $\boldsymbol{H}$ is Shannon's entropy $-\sum_i p_i \log_2(p_i)$.

Given the exponential cost of solving Eq. (5.1), we apply a greedy heuristic where we iteratively choose the value of one bit $b \in \mathcal{B}$ chosen at random. For each bit, we set its value to 0 or 1 by maximizing $\boldsymbol{H}(\Phi_{m,n+1})$ where m is the set of bits greedily chosen at random so far. In case of a tie (i.e., the entropy is the same regardless of whether bit b is set to 0 or 1), we choose the value of bit b at random. Given probes have high entropy for arbitrary combinations of bits in $\mathcal{B}$, this approach ensures a number of different inputs to the hash function and reliably triggers load balancing for any hash domain that overlaps $\mathcal{B}$ (regardless of the overlap).

We note that although MCA (and measurement tools in general) cannot control all bits in the source and destination addresses while performing a measurement, MCA still supports testing whether they are included in hash domains by allowing $\mathcal{B}$ to include the least significant bits of source and destination addresses.[1]

5.2 Load Balancer Classification

For each LB identified in the previous step, our goal is to identify which header bits in $\mathcal{B}$ overlap its hash domain.

For each interface i identified as a LB, MCA chooses one flow identifier known from the detection phase to traverse i, and sends multiple probes with that *fixed* identifier to i's next hop. We compute the number of probes to send as in MDA [20]. MCA classifies i as a per-packet LB if responses arrive from multiple next hops.

If interface i is not classified as per-packet, MCA sends additional probes according to Algorithm 1 to infer which bits in $\mathcal{B}$ overlap with i's hash domain and classify the LB type (line 3). The number n of trials to perform for each bit is a function of a configurable confidence level α_c that determines the acceptable probability of classification error (line 2).

[1] Varying the source IP address requires the measurement device to control multiple IP addresses (e.g., when allocated a /64 IPv6 address).

Algorithm 1: Classify load balancer interface i at hop h

```
   Input  : Load balancer interface $i \in h$ with set of next hops $\mathcal{N}$, known not to
            be a per-packet load balancer; target bit set $\mathcal{B}$; confidence level $\alpha_c$
   Output: Interface $i$'s hash domain $\mathcal{H}_i \subseteq \mathcal{B}$
 1 $\mathcal{H}_i \leftarrow \emptyset$
 2 $n \leftarrow \textsc{NumTrials}(|\mathcal{B}|, |\mathcal{N}|, \alpha_c)$                Eq. (5.2)
 3 foreach bit $b \in \mathcal{B}$ do
 4     for $j \leftarrow 1$ to $n$ do
 5         $\phi, \phi' \leftarrow \textsc{SelectFlows}(i, h, b)$           §5.3
 6         $reply \leftarrow \textsc{ProbeReply}(\phi, p\langle h \rangle + 1)$
 7         $reply' \leftarrow \textsc{ProbeReply}(\phi', p\langle h \rangle + 1)$
 8         if $reply \neq reply'$ then
 9             $\mathcal{H}_i \leftarrow \mathcal{H}_i \cup b$
10             break
11 return $\mathcal{H}_i$
```

In each trial, MCA identifies a flow identifier ϕ that traverses i and a flow identifier ϕ' that also traverses i and is identical to ϕ except for the value of b (line 5, §5.3). MCA sends probes with ϕ and ϕ' to i's next hop (if not yet sent) and waits for the responses (lines 6–7). This process is repeated until MCA identifies flow identifiers ϕ and ϕ' that i forwards to different interfaces (lines 8–10), or until enough trials are performed to infer that b does not overlap with i's hash domain at the configured confidence level (i.e., when $j = n$ in line 4).

MCA computes the number of trials to send when classifying an LB as a function of the confidence level α_c. We define the classification of an LB interface i as correct when *all* target bits in $\mathcal{B}$ are correctly labelled as overlapping or not with i's hash domain $\mathcal{H}_i$. We assume LBs distribute flow identifiers uniformly over next hops. Violation of this assumption increases the misclassification probability.

We denote the probability that an interface i forwards traffic to its next hop j as $P_{\text{next}}(i, j)$. Using the uniformity assumption across the set of next hops $\mathcal{N}_i$, we have $P_{\text{next}}(i, j) = 1/|\mathcal{N}_i|$ for all $j \in \mathcal{N}_i$.

Mislabelling can only occur for a header bit $b \in \mathcal{B}$ that overlaps $\mathcal{H}_i$. We compute the number of trials considering the worst case scenario where $\mathcal{B} \cap \mathcal{H}_i = \mathcal{B}$; if $\mathcal{B}$ is a proper subset of $\mathcal{H}_i$, then we overestimate the number of trials and the probability of error will be smaller than α_c. Mislabelling b when performing n trials varying b happens when b is in i's hash domain, and probes on all n trials are forwarded to the same next hop, which happens with probability $(1/|\mathcal{N}_i|)^{n-1}$. The probability of misclassification is then given by the probability that we mislabel any bit in $\mathcal{B}$:

$$P_{\text{miss}}(i, n, \mathcal{B}) = 1 - \left[1 - \left(\frac{1}{|\mathcal{N}_i|}\right)^{n-1}\right]^{|\mathcal{B}|}.$$

To bound $P_{\text{miss}}(i, n, \mathcal{B})$ below $1 - \alpha_c$ we set

$$n = \left\lceil -\log_{|\mathcal{N}_i|}\left(1 - \alpha_c^{\frac{1}{|\mathcal{B}|}}\right)\right\rceil + 1. \tag{5.2}$$

To classify an interface i with confidence $\alpha_c = 0.95$ when $|\mathcal{B}| = 24$ and $|\mathcal{N}_i| = 2$, MCA first sends up to 9 probes to check if i is a per-packet load balancer and, if not, performs up to 9 trials (up to 18 probes) for each bit $b \in \mathcal{B}$.

5.3 Optimizations for Searching Flow Identifiers

In this section, we discuss the existing approach (baseline) for searching new flow identifiers [35] and optimizations to reduce probing cost.

5.3.1 Problem

MDA and MCA need to generate a number of probes with varying flow identifiers through each interface i in a route to identify i's next hops and, if load balancing is identified, i's hash domain. Consider the MDA probing process for the route in Figure 5.1. MDA will send 17 probes[2] varying flow identifiers through s to identify s's next hops a_1 and a_2. To send a probe through an interface i requires that MDA first check that the probe's flow identifier reaches i. To identify the next hops of a_1, MDA will also send 17 probes varying flow identifiers through a_1 to identify a_1's next hops b_1 and b_2. Any flow identifier MDA knows to reach a_1 from the 17 probes sent to enumerate s's next hops can be reused. However, MDA will need to send *search probes* to identify new flow identifiers that reach a_1 (as it needs 17 flow identifiers through a_1).

5.3.2 Randomized Search (Baseline)

New flow identifiers that reach a given interface are found by trial and error, and can amount to a significant fraction of the probing cost. The probability that a given

[2] We consider a confidence level $\alpha = 0.95$, which requires that MDA send 9, 17, or 24 probes through an interface that has one, two, or three next hops, respectively [20].

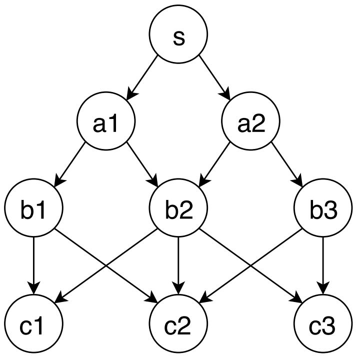


Figure 5.1: Example toplogy

flow identifier follows a branch segment $\beta = [i_1, i_2, \ldots, i_{|\beta|}]$, assuming each interface performs load balancing independently, is $P_{\text{branch}}(\beta) = \prod_{1 \leq k < |\beta|} P_{\text{next}}(i_k, i_{k+1})$. The probability that a flow identifier that goes through interface i at radius r_i also goes through interface j at radius $r_j > r_i$ is given by

$$P(i, j) = \sum_{\beta} P_{\text{branch}}(\beta), \text{ for all branches } \beta \text{ between } i \text{ and } j.$$

For example, in Figure 5.1, where we assume all interfaces distribute flows uniformly, $P(s, b_1) = \frac{1}{2}\frac{1}{2}$ and $P(s, b_2) = \frac{1}{2}\frac{1}{2} + \frac{1}{2}\frac{1}{2}$. The average number of trials to find a new identifier that reaches i from s is $1/P(s, i)$.

5.3.3 Reusing Flow Identifiers for Identification

When searching for new flows that reach an interface i in order to enumerate its next hops, classic MDA will try new flow identifiers from the source s. Alternatively, we propose the reuse of flow identifiers sent to previous hops. We define a flow identifier ϕ as *reusable* if it has been sent to some radius smaller than i's radius and has not yet been sent to i's radius. For each flow identifier ϕ, we define the interface with the highest radius ϕ is known to reach as $\text{tip}(\phi)$. For each reusable flow identifier ϕ, we estimate the probability that it will reach interface $i \in h$ as $P(\text{tip}(\phi), i)$.

We try reusable flows in order of decreasing probability to reach i. Let $\mathcal{A}_i$ denote the set of ancestors of interface i in i's outermost diamond. In Figure 5.1, $\mathcal{A}_{c_1} = \{s, a_1, a_2, b_1, b_2\}$. The optimum flow trial order for finding new flows through c_1 is given by $P(b_1, c_1) = \frac{1}{2} > P(a_1, c_1) = \frac{5}{12} > P(b_2, c_1) = \frac{1}{3} > P(s, c_1) = \frac{7}{24}$. When searching flows through c_1, MCA will *not* reuse any flow ϕ with $\text{tip}(\phi) = a_2$, as these flows have a lower chance, $\frac{1}{24}$, to reach c_1 than new random flow identifiers from the source (counted as a special case of reusable). Given that route measurement proceeds hop-by-hop and all radii up to i have been probed prior to identifying i's next hops, $\mathcal{A}_i$ and probabilities $P(a, i)$ of reaching i from all ancestors $a \in \mathcal{A}_i$ can be computed backwards from i in time $O(|\mathcal{A}_i|)$.

5.3.4 Searching Sequence for Identification

When identifying next hops of interfaces at a hop h, MDA may need to find a different number of new flows through each interface $i \in h$. The total number of trials required depends on the order in which we try flows. Let $M_h[i]$ denote the fraction of new flow identifiers missing for interface $i \in h$ relative to the total over h, and M_h the corresponding vector over all i. To minimize the total number of trials across h, we issue a probe with a flow identifier that best matches M_h. More precisely, for each trial, we select the identifier ϕ over all reusable flows that maximizes the utility

$$U(\phi) = \sum_{i \in h} M_h[i] P(\text{tip}(\phi), i). \tag{5.3}$$

We update M_h and recompute the optimal ϕ after each trial.

Example: Consider that MDA needs to find 1, 2, and 3 new flow identifiers through c_1, c_2, and c_3, respectively. Then $M_h[c_1] = \frac{1}{6}, M_h[c_2] = \frac{2}{6}$, and $M_h[c_3] = \frac{3}{6}$, and the flow reuse order is given by the utilities of available tips: denote Φ_x any flow ϕ such that $\text{tip}(\phi) = x$, then $U(\Phi_{a_1}) = 0.29 < U(\Phi_s) = U(\Phi_{b_2}) = 0.33 < U(\Phi_{a_2}) = 0.37 < U(\Phi_{b_3}) = 0.42$.

5.3.5 Varying a Bit Value for Classification

After identifying next hops, MCA needs to send probes with flow identifiers ϕ and ϕ' that differ for a single header bit through an interface i to classify its behavior. It is potentially onerous to find enough of such ϕ', however here the fact that LBs on the route can be of different types, usually seen as a classification burden, can be exploited to obtain a supply of ϕ' efficiently. This optimization provided the most significant gains in our experiments, and works as follows.

Any flow identifier ϕ' that overwrites a header bit b in ϕ is guaranteed to also traverse i if b does not overlap the hash domains of LBs in the branch taken by ϕ to reach i. For such bits, MCA can generate a ϕ' flow identifier through an interface i from any known flow identifier ϕ without trials. This implies that bits that are seldom used for load balancing incur a known, fixed probing overhead given by Eq. (5.2).

Algorithm 2: SelectFlows algorithm

```
   Input  : Load balancer interface $i \in h$, bit $b$
   Output: Flow identifiers $\phi$ and $\phi' = \phi \oplus b$ known to reach $i \in h$
1  foreach $\phi \in$ flows$(i, h)$ do
2      $\mathcal{A} \leftarrow$ Branch$(\phi, i)$
3      $\mathcal{P}_\phi \leftarrow \cup_{a \in \mathcal{A}} \mathcal{H}_a$
4      if $b \notin P_\phi$ then
5          return $(\phi, \phi \oplus b)$
6  foreach $\phi \in$ flows$(i, h)$ do
7      if CheckInterface$(\phi \oplus b, r, i)$ then
8          return $(\phi, \phi \oplus b)$
9  do
10     do
11         $\phi \leftarrow$ NewFlowIdentifier$()$
12     while not CheckInterface$(\phi, r, i)$
13 while not CheckInterface$(\phi \oplus b, r, i)$
14 return $(\phi, \phi \oplus b)$
15
16 function CheckInterface$(\phi, r, i)$
17     $R \leftarrow$ ProbeReply$(\phi, r)$
18     return $R.src = i$
```

The same property can be used when hash domains $\mathcal{H}$ of all LBs in a branch β are identical. In these cases, we can generate new flow identifiers by overwriting all bits in $\mathcal{B} \setminus \mathcal{H}$ in flows known to reach i through β, and the new flows are guaranteed to also reach i. This property allows us to repurpose flow identifiers generated during the next hop identification phase, each with a distinct value over $\mathcal{H}$, into probes useful for classifying i's hash domain.

Consider that s and a_1 in Figure 5.1 perform per-destination load balancing. Consider MCA knows flows $\phi_1, \phi_2, \ldots, \phi_k$ reach b_1 through s and a_1. We can generate flow identifiers that reach b_1 and vary a bit b not in the destination address by picking, e.g., ϕ_1, and overwriting b. We can also generate k flow identifiers that reach b_1 and vary only the destination address by overwriting all fields other than the destination address in $\phi_1, \phi_2, \ldots, \phi_k$, for example with a fixed value.

Algorithm 2 shows pseudocode for the search for flow identifiers for classifying whether bit b overlaps interface i's hash domain. MCA first searches for a flow identifier ϕ that reaches $i \in h$ for which b does not overlap with the hash domains of load balancers on the branch taken by ϕ (lines 1–4).[3] If such a ϕ exists, MCA creates a new flow identifier by swapping bit b in ϕ and returns the pair of flow identifiers (line 5). Next, MCA iterates over flows known to reach i (line 6) and sends probes to verify if $\phi' = \phi \oplus b$ also reaches i (line 7). If MCA finds a ϕ' that also reaches i, then it returns the pair of flow identifiers (line 8). Finally, MCA resorts to finding new flows ϕ that reach i (lines 10–12), then checking if $\phi' \oplus b$ also reaches i (line 13). This process is repeated until MCA finds a pair or flow identifiers ϕ and ϕ' that reaches i (line 14).

5.3.6 Varying a Field Value for Classification

We have yet to address the question of the choice of $\mathcal{B}$. It is potentially highly onerous to select a large set, and then to determine each hash domain to bit resolution as described above (consider Eq. (5.2)). However, it is possible to work at the level of header fields instead. LB discovery is unchanged: the hash domains are still general, we simply set $\mathcal{B}$ to the union of the header fields of interest. For LB classification, the same algorithms work unmodified by operating on $\mathcal{F}$ instead of $\mathcal{B}$ ($\mathcal{F}$ was defined in the first paragraph of §5 as the set of packet header fields that overlap $\mathcal{B}$). The end result is simply to report only at field granularity: i.e., the fields which have been found to intersect each hash domain.

5.4 Operational Considerations

MDA can identify and MCA can classify routers that load balance traffic over *visible* MPLS tunnels [98, 99], i.e., MPLS tunnels that do not reset the TTL field of probes and whose router's ICMP time exceeded responses include the MPLS label stack. MDA can identify load balancing (but not the intermediate hops) and MCA can (partially) classify load balancing over invisible MPLS tunnels when (i) probes arrive at different interfaces at the router where the invisible MPLS tunnels end and (ii) the router originates ICMP time exceeded replies using different IP addresses.

Some approaches to load balancing include the TTL in the hash domain [97]. These mechanisms make it impossible to control the branches a flow identifier will follow, as traceroute cannot function without varying the TTL of probes. For example, consider that A in Figure 1.2a includes the TTL in its hash domain, and that it forwards packets with flow identifier ϕ sent to radius 3 towards B. When identifying or classifying load balancing at B, MCA needs to send probes to B's next hops at

[3] If ϕ has not been sent to all radii up to $p\langle h \rangle$, then the branch it takes to reach i it not entirely known. In this case, we add all possible ancestors ϕ can traverse in the missing radii to $\mathcal{A}$. This is a conservative approach that does not introduce any error, but may reduce probe savings.

radius 4, but the different TTL may lead A to forward the probes to C, which would lead to the incorrect inference of a link between B and F.

When varying the last bits of the destination address inside the destination's AS, MCA probes may discover subnets other than the destination's [55]. To avoid reporting interfaces found towards destinations on other subnets, MCA only reports interfaces found inside the destination AS that are observed on probes targeting the original destination.

5.5 MCA Implementation

We implement MCA in a command-line tool to identify and classify load balancers. Our MCA implementation supports TCP, UDP, and ICMP measurements using both IPv4 and IPv6. It supports detection and classification covering bits in the DSCP, traffic class, flow label, destination address, source port, destination port, and ICMP checksum fields. Table 5.1 lists MCA's configuration parameters and their default values.

TTL-limited probes may remain unanswered due to anonymous and hidden routers, as well as ICMP rate limiting [18,53,100]. For both discovery and classification steps, our implementation retries probes that are not answered after a timeout. The number of retries, timeout, and probing rate are configurable. The measurement cost and probability of measurement errors can be controlled by setting the value of the α confidence threshold.

Our implementation stops probing under the following conditions: (i) after receiving any packet from the destination, (ii) after receiving an ICMP destination unreachable message from an intermediate router, (iii) after discovering more than a configurable number of next-hops of an interface (16 next hops by default), (iv) after identifying more than a configurable number of interfaces at a given hop (32 interfaces by default), (v) after a configurable number of consecutive unresponsive hops (3 hops by default).

We also implement Route Explorer, a front-end for MCA measurements, which allows inspection of probes and responses, and provides graphical visualizations integrating metadata such as IP-to-AS mapping and CAIDA's AS-rank [60,62].

[3] Our MCA implementation, dataset explorer, Route Explorer source code, and data are available at https://mca.speed.dcc.ufmg.br.

Table 5.1: MCA's parameters and default values

Parameter	Default	Meaning
max-ttl	32	Maximum number of hops to probe
alpha	0.95	Confidence level α
max-next-hops	16	Maximum number of next hops for a load balancer
max-width	32	Maximum number of interfaces in a hop
gap-limit	3	Maximum number of consecutive unresponsive hops
max-retries	3	Maximum number of retries for unanswered probes
probe-timeout	1.0	Time to wait for a probe answer (in seconds)
transport	UDP	Probe transport protocol
pps	50	Number of probes to send per second
fields	—	Set of target fields $\mathcal{F}$

Chapter 6

Dataset

In this chapter we describe the dataset we collected using MCA to evaluate our optimizations and characterize load balancing in the Internet. We collect a dataset running MCA from a diverse set of vantage points to a large set of destinations. To better understand load balancing practices and deployment across networks, we augmented our MCA measurements with IP-to-AS mapping information from Team Cymru's IP-to-AS service [60], reverse DNS entries (PTR records) for IP addresses in our measurements, and networks types from PeeringDB [61]. We make our dataset and ongoing measurements available through a dataset explorer at `https://mca.speed.dcc.ufmg.br/`.

6.1 Measurement Setup

We run six measurement campaigns to cover all combinations of IP protocol version (v4, v6) and transport (TCP, UDP, and ICMP). In each run, we configure MCA with $\mathcal{F}_{\text{IPv4}} = \{$destination IP, source port, destination port, DSCP$\}$ and $\mathcal{F}_{\text{IPv6}} = \{$destination IP, source port, destination port, flow label, traffic class$\}$ to cover currently understood standard load balancer types (§2.3) as well as more general but still reasonable hash domains for novelty. We use MCA's default configuration parameters (Table 5.1).

Figure 6.1 shows packet headers and fields we configure MCA to vary to identify IPv4 and IPv6 load balancing with different transport protocols. MCA stores a matching key of two bytes in the sequence number field for TCP probes, in the checksum field for UDP probes and in the identifier field for ICMP probes. To control the value of checksum fields (e.g., when storing the matching key in UDP's checksum field or varying the ICMP checksum), MCA adds a two-byte payload to

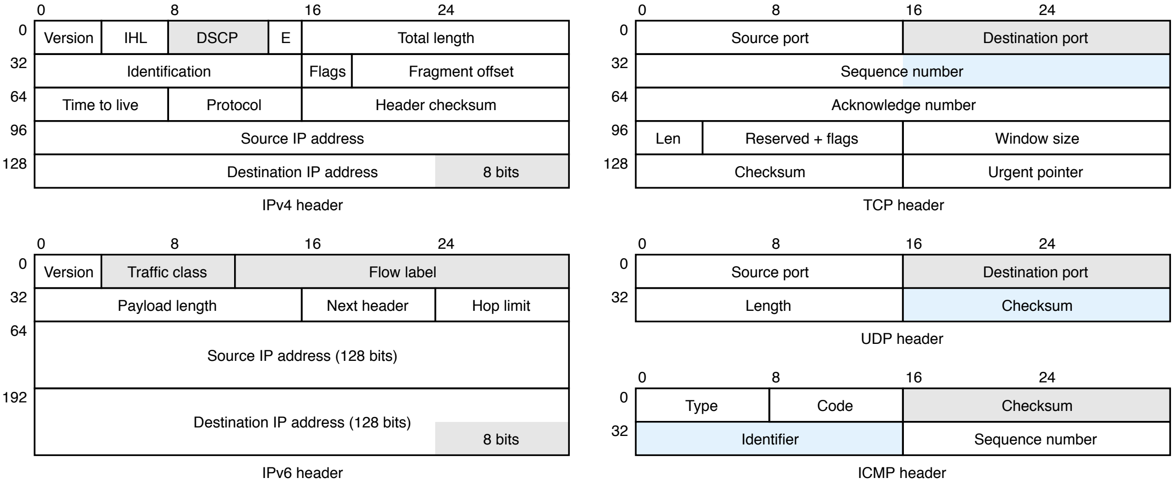


Figure 6.1: Packet headers, flow identifiers (in gray), and matching keys (in blue) we use to identify and classify load balancers.

Table 6.1: Dataset summary

				Number of traces		
Platform	VPs	Period		IPv4	IPv6	ASes
UFMG [103]	1	2018-08-21	2018-09-06	16,272	18,684	1,540
Linode [104]	6	2018-08-21	2019-03-01	262,752	242,088	6,787
Vultr [105]	6	2018-08-21	2019-03-01	305,628	263,136	7,586
DOcean [106]	7	2018-08-21	2019-03-01	356,808	321,180	7,587
Ark [38]	11	2018-08-21	2019-04-27	571,104	469,464	8,939
All	31	2018-08-21	2019-04-27	1,512,564	1,314,552	10,454

the probe and varies the value of the payload to control the checksum.

Choosing Vantage Points

We deployed MCA in 31 vantage points (VPs) in 6 platforms (cloud providers, monitoring testbeds, and one university) that provide IPv4 and IPv6 connectivity and that allow crafting and sniffing packets using Scapy [101]. The vantage points are spread across 16 countries in 5 continents. Previous work has used PlanetLab nodes as vantage points; we choose not to use PlanteLab as it has no support for IPv6. We believe the set of vantage points we use in this paper provides more diversity as most PlanetLab nodes are hosted in educational networks [102]. Table 6.1 summarizes our vantage points and dataset.

Choosing Destinations

We run MCA from each vantage point towards a list of 36,540 IP destinations built as follows. We resolved A and AAAA DNS records for domains on three Internet 'toplists': Alexa [92], Cisco Umbrella [93], and Majestic [107]. To increase topological diversity of probed destinations while restricting the number of destinations, we group IPv4 and IPv6 addresses in /16 and /32 prefixes, respectively, and choose one representative address per prefix (the one which maximizes the average prefix length shared with all other addresses in the prefix). This resulted in a set of 11,946 IPv4 and 8,497 IPv6 addresses.

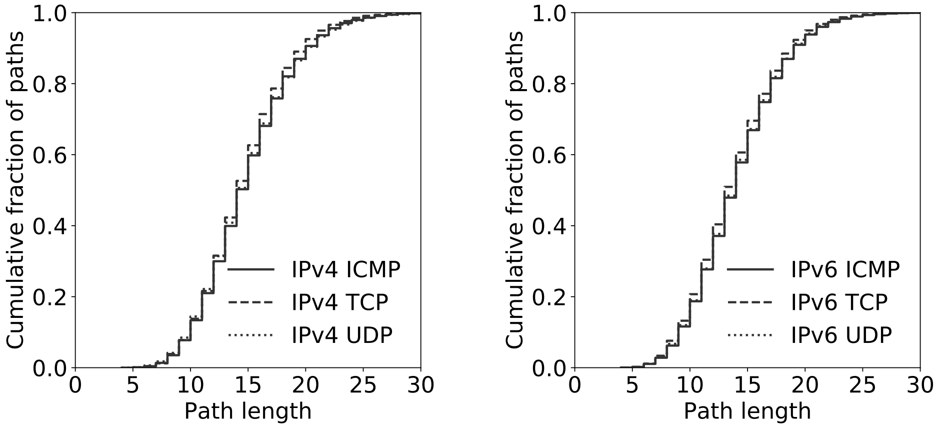
We complement the lists above with IP addresses selected from ISI's IPv4 Internet census data from Sep. 2018 [108] and Gasser et al.'s IPv6 hitlist from Jan. 2019 [109]. We pick the IPv4 address with the highest response rate (during ISI's census mesurements) in each /16 prefix not covered by toplists, but ignore /16 prefixes without any IP address with at least a 10% response rate. We group IPv6 addresses in /32 prefixes not covered by 'toplists' and picked the address which maximizes the average prefix length shared with all other addresses in the prefix. This resulted in a set of 9,893 IPv4 and 7,264 IPv6 addresses.

6.2 Dataset Properties

Our IPv4 destinations are distributed in 4,387 ASes (61% in stub ASes), and our IPv6 destinations are in 8,106 ASes (50% in stub ASes). Destination ASes include all Tier-1 ASes and 95% of ASes with more than 500 indirect customers (as inferred by CAIDA's AS relationship database [8]). We find that 80 and 974 ASes concentrate 50% of the IPv4 and IPv6 destinations, respectively, and that 3,134 and 6,385 ASes contain one IPv4 and IPv6 destination, respectively. Manual inspection of the top 50 ASes with the most addresses reveal major content and cloud infrastructure providers such as Google, Amazon, and Microsoft.

Figure 6.2 shows the distribution of the length of the longest branch in route measurements. We find negligible difference between network protocols, with IPv6 measurements being 0.9 hops shorter than IPv4 routes on average, and between transport protocols, with TCP measurements yielding slightly shorter routes than UDP and ICMP.

Table 6.2 shows the fraction of measurements that reach the destination and that reach any interface whose IP address maps to the same AS as the destination's. Similar to previous work [96], we find that ICMP measurements reach a higher fraction



(a) IPv4

(b) IPv6

Figure 6.2: Multi-route length (number of hops)

Table 6.2: Fraction of measurements that reach the destination IP interface and AS per protocol

	IPv4			IPv6		
	UDP	TCP	ICMP	UDP	TCP	ICMP
Reach destination	17.3%	16.9%	39.3%	39.4%	27.0%	67.0%
Reach dest. AS	73.6%	73.7%	79.6%	76.1%	71.8%	83.9%

of destinations and destination ASes compared to TCP and UDP for both IPv4 and IPv6. Of the measurements that stop early, 13% stop by exceeding 32 interfaces in a given hop and 87% stop by not receiving any reply for three consecutive hops. We truncate less than 1% of measurements at the first interface that implies a routing loop. We observe exhaustion of flow identifiers when varying the Traffic Class for 1.6% of IPv6 interfaces and exhaustion of flow identifiers when varying DSCP values for 2.8% of IPv4 interfaces.

For each sequence of unresponsive interfaces in a measurement, we check if the surrounding responsive interfaces have a single sequence of responsive interfaces between them in other measurements. If this is the case, we substitute the unresponsive interfaces with the responsive ones. We apply a similar strategy for interface IP addresses not mapped to AS numbers. For a sequence of unresponsive interfaces (and hops not mapped to ASNs) we check if the ASes of the surrounding mapped interfaces are interconnected by a single sequence of ASes in public BGP feeds. If this is the case, we substitute the missing ASes by the sequence of ASes observed in public BGP feeds.

Chapter 7

Evaluation

In this chapter we evaluate MCA's probing cost with and without our optimizations introduced in §5.3. Key to this analysis is that the MCA measurement cost for a route is variable: the number of probes sent depends on the random flow identifiers generated during the identification and classification phases. In particular, it depends on the number of probes sent when searching for flow identifiers through the identified interfaces.

As the number of probes is high, rate-limits impose a maximum probing rate, and paths may change over time, estimating the average probing cost with real measurements is not viable. Instead, we compute the average probing cost using trace-driven simulations. For each route measurement in our dataset, we simulate the unoptimized and optimized versions of MCA until the 95% confidence interval around the average probing cost of each outermost diamond is below ±1 probe. We hash flow identifiers using SHA-256.

Figure 7.1 shows the distribution of the average number of probes issued by MCA for all outermost diamonds in our dataset on our simulations. We split costs between the identification and classification phases when using and not using our optimizations. We note that MCA's classification incurs probing cost that is of the same order of magnitude than that of identification, but often significantly less. When using our optimizations, classification amounts to 34% of the total probing cost of IPv4 measurements and to 39% of IPv6 measurements, on average.

Our optimizations only apply to outermost diamonds with nested diamonds. This means that our optimizations only save probes on complex diamonds. To illustrate this, Figure 7.2 shows the distribution of the number of probes issued in our simulations for outermost diamonds that have nested diamonds. In Figure 7.2 we observe higher probing cost overall and more significant savings for classification.

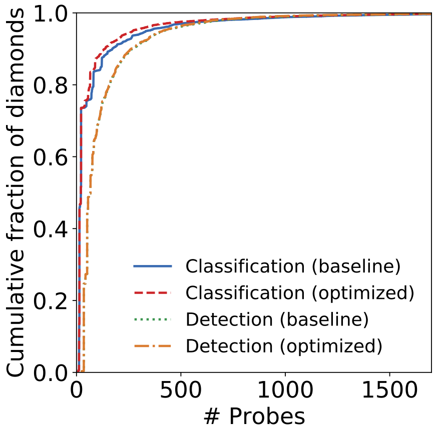


Figure 7.1: Probing cost for all outermost diamonds

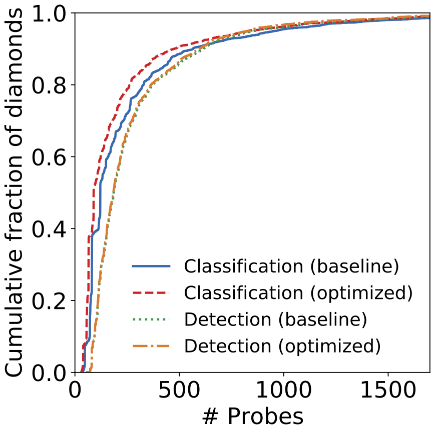


Figure 7.2: Probing cost for outermost diamonds with nested diamonds

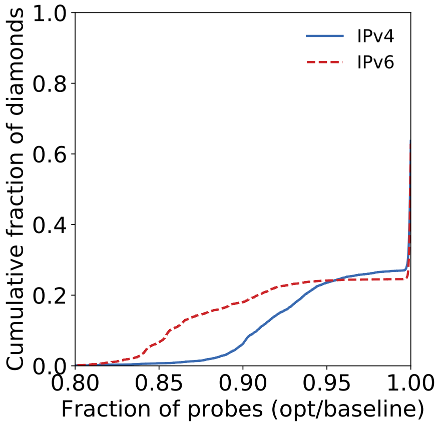


Figure 7.3: Normalized probing cost

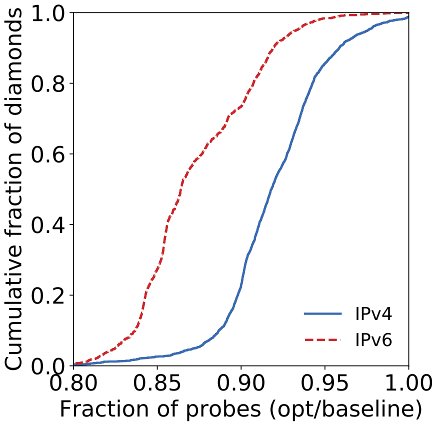


Figure 7.4: Normalized probing cost for outermost diamonds with depth ≥ 2

Figure 7.3 shows the number of probes sent during detection and classification by the optimized MCA normalized by the number of probes sent by the unoptimized MCA (baseline) for IPv4 and IPv6 outermost diamonds. Similarly, in Figure 7.4 we show savings for outermost IPv4 and IPv6 outermost diamonds with nested diamonds (i.e., with depth ≥ 2). Although our optimizations provide a small reduction in the number of probes, they have *no* impact on accuracy and apply to *all* load balancers; we argue implementations would benefit from integrating the optimizations.

Figure 7.5 shows the normalized probing cost for identifications as a function of an outermost diamond's *depth*, defined as the maximum number of load balancers

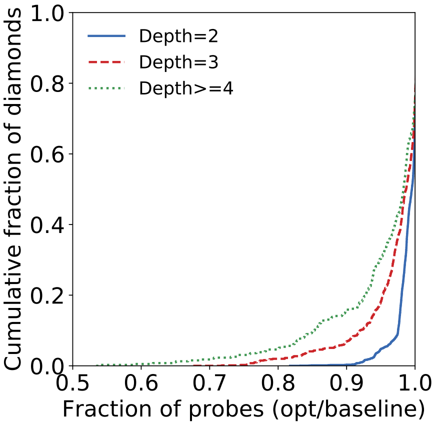


Figure 7.5: Normalized probing cost for detection

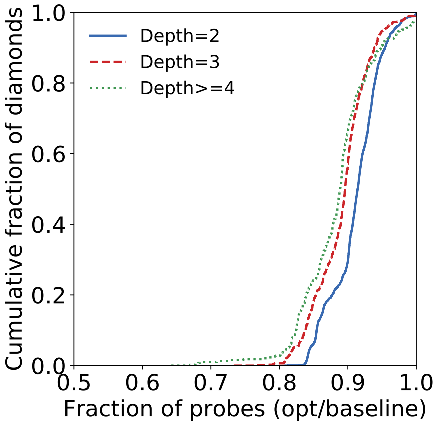


Figure 7.6: Normalized total (detection and classification) probing cost for outermost diamonds

in any of the outermost diamond's branches. Diamond depth is a measure of complexity, and we observe that our optimizations save more probes on more complex diamonds, which also have higher probing cost. In the same way, Figure 7.6 shows the normalized total probing cost as a function of an outermost diamond's depth.

Our optimizations provide only modest probe savings for detection with, on average, 2.8% less probes for IPv4 and 0.7% for IPv6. For classification, the savings are higher: reducing classification costs by 11% for IPv4 and 18% for IPv6 routes. Savings for classification for IPv6 are higher than for IPv4 because our MCA executions included four header fields for IPv6 and three for IPv4, and more fields means opportunities to apply optimization. In measurement campaigns covering more fields, savings would be higher than reported here. In total, we estimate by simulation that the optimizations reduce the total probing cost for routes in our datasets by 6% for IPv4 and 8% for IPv6 for the default MCA configuration. Although our optimizations provide only single-digit savings, implementations would benefit from integrating them as they have *no* impact on accuracy and apply to *all* measurements.

Chapter 8

Characterization of Load Balancing

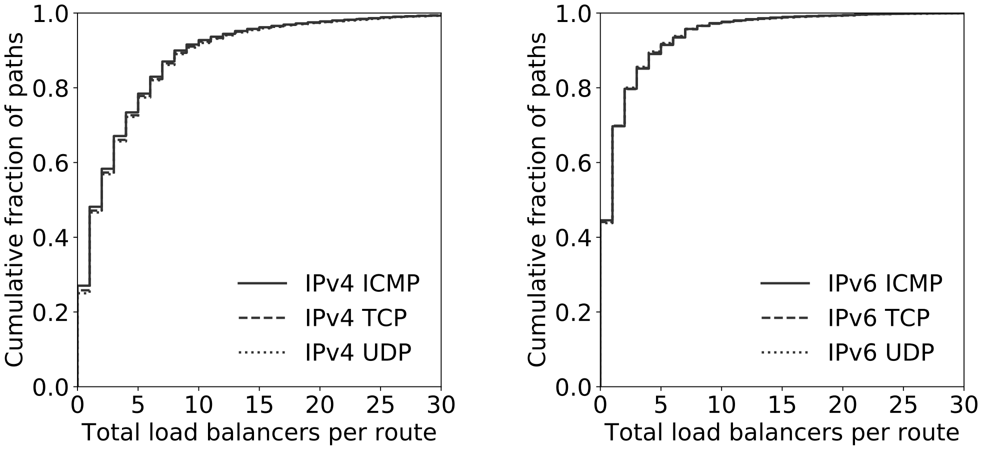
In this chapter, we characterize IPv4 and IPv6 load balancing on the Internet. We study the prevalence of load balancing on routes (§8.1) and identify common hash domains used for load balancing and classes of load balancer behavior (§8.2). We revisit previous work and discuss the evolution of load balancing and path diversity (§8.3). Finally, we discuss routers that override packet header fields and complicate load balancing detection as well as classification (§8.4).

8.1 Occurrence of Load Balancing

In this section we provide an overview of load balancing observed in our measurements. The majority of cloud providers we used, some networks hosting Ark nodes, and the university's provider (an educational network) employ load balancing in their networks. In these cases, most paths from vantage points traverse at least one load balancer. Unless otherwise noted, we report results on load balancers identified *outside* the vantage point's network to avoid biasing our results.

Load balancing is prevalent

Figure 8.1 shows the distribution of the number of load balancers in Internet routes, and Table 8.1 shows the prevalence of each of the most common types of load balancer (§2.2). We find that load balancing is applied similarly across all transport protocols. Even when ignoring load balancing in the vantage point's network, we find that 74% of IPv4 routes and 56% of IPv6 routes traverse at least one load balancer, and some traverse many more. When considering load balancers in the origin network, we find that 86% of IPv4 routes traverse at least one load balancer, a fraction similar to that observed in 2009 [13], and that 77% of IPv6 routes traverse at



(a) IPv4

(b) IPv6

Figure 8.1: Number of load balancers per route

least one load balancer, slightly larger than the 74% observed in 2016 [14]. This result indicates the extent of traffic engineering used in the Internet and the need for considering load balancers in Internet measurement efforts. In particular, the growth in the observed fraction of IPv6 routes with load balancing may reflect the growth of IPv6 deployment and the resulting need for traffic engineering [28, 110].

IPv4 load balancing is more widespread

We find 55% of ASes traversed in our IPv4 measurements employ IPv4 load balancing; this fraction is 41% for load balancing in IPv6 measurements. Moreover, although 58% of ASes that appear in both IPv4 and IPv6 measurements employ both IPv4 and IPv6 load balancing, ASes more often employ IPv4 load balancing exclusively. We find that 82% of the ASes that employ IPv6 load balancing and are traversed in IPv4 measurements employ IPv4 load balancing, but that only 66% of the ASes that employ IPv4 load balancers and are traversed in IPv6 measurements employ IPv6 load balancing. These results indicate richer traffic engineering in IPv4 than in IPv6, possibly due to IPv4 still carrying most traffic or simply due to being around (and engineered) for longer.

8.2 Classes of Load Balancing

As a measure of the accuracy of our classifications, we check whether multiple measurements of the same load balancer identify identical hash domains. When using TCP probes, we find that 9.7% of IPv4 and 8.8% of IPv6 load balancers are measured

Table 8.1: Breakdown of load balancer prevalence by protocol

	IPv4			IPv6		
	UDP	TCP	ICMP	UDP	TCP	ICMP
Per-flow	70.8%	70.1%	1.2%	51.4%	51.3%	0.7%
Per-dest	18.4%	17.5%	72.6%	8.4%	8.4%	53.3%
Per-packet	0.1%	0.1%	0.1%	0.0%	0.0%	0.1%
Per-flow + FL	0.0%	0.0%	0.0%	0.5%	0.6%	0.0%
Per-dest + FL	0.0%	0.0%	0.0%	0.0%	0.1%	0.5%
Any	74.9%	74.2%	72.9%	56.1%	55.8%	55.4%

Table 8.2: Breakdown of load balancer classifications by protocol

	IPv4			IPv6		
	UDP	TCP	ICMP	UDP	TCP	ICMP
Per-flow	69.6%	69.8%	1.5%	77.6%	78.5%	0.3%
Per-dest	24.4%	24.1%	94.2%	13.3%	13.5%	90.1%
Per-app	1.8%	2.1%	0.0%	0.9%	0.8%	0.0%
Per-packet	0.1%	0.1%	0.1%	0.1%	0.1%	0.3%
Per-flow + FL	—	—	—	2.9%	2.4%	0.0%
Per-dest + FL	—	—	—	0.2%	0.3%	3.2%
Other	2.3%	2.6%	2.7%	3.2%	2.8%	3.9%
Not classified	1.8%	1.4%	1.5%	1.7%	1.6%	2.2%
Total	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%

multiple times (this ratio is similar across TCP, UDP, and ICMP measurements), and that 98.1% these load balancers have a single hash domain, which is within our 95% confidence threshold.

Table 8.2 shows the percentage of load balancers of each class per protocol in Internet routes, ignoring load balancers in the vantage point's network. As explained before, load balancing is more prevalent on IPv4 routes and there is no significant difference in prevalence between transport protocols. We observe that per-flow load balancers are the most common, followed by per-destination.

We note a few load balancers considering only transport port numbers in their hash domain (we found this can be configured in RouterOS as "per-application load balancing" [33]) and some IPv6 load balancers employing the flow label field (the default in JunOS [32]).

Load balancing is improving

Per-packet load balancing is at an all-time low relative to previous characterizations. As per-packet load balancers induce packet reordering and TCP performance degradation, this is a positive trend.

We find few load balancers whose hash domains include the ICMP checksum field (shown as per-flow in Table 8.2). Augustin et al. [13] reported a considerable decrease in the number of routers performing per-flow load balancing for ICMP between 2006 and 2009; we find this trend has continued, with an all-time low of IPv4 load balancers considering the ICMP checksum field (3.2% of load balancers, down from 20% in 2009). This behavior may follow from more mature implementations defaulting to per-destination load balancing for ICMP packets.

We find that an average of 2.8% of IPv6 load balancers consider the flow label field in their hash domain, when measuring with TCP. We also observe the use of flow label in conjunction with per-flow or per-destination load balancing classes: we classify 2.7% of IPv6 load balancers as having this behavior. A significant fraction (85%) of load balancers including the flow label in their hash domains are in content and infrastructure providers' networks (e.g. Facebook's AS32934 and Google's AS15169), which adopt IPv6 and rely on advanced traffic engineering [29,30].

Traffic engineering triggers load balancing

IPv6's traffic class and IPv4's DSCP fields serve a similar purpose of classifying packets. Although their use is not widespread, these fields are included in the default hash domain of some versions of Juniper's JunOS [32]. We identified a non-negligible number of load balancers with the IPv6 traffic class (2.7%) and IPv4 DSCP fields (2.6%) in their hash domains when measuring with TCP.

Load balancers are widespread

Not only do many routes traverse load balancers, but load balancers are (close to) uniformly distributed along all hops of branched routes (not shown).

Classification errors

When measuring with TCP, MCA failed to classify 1.4% IPv4 and 1.6% IPv6 load balancers; i.e., MCA detected load balancing, but did not observe load balancing during the classification phase. Possible reasons for this include measurement errors (possibly aggravated by non-uniform load balancing) or hash domains that only trigger load balancing when multiple fields are varied simultaneously.

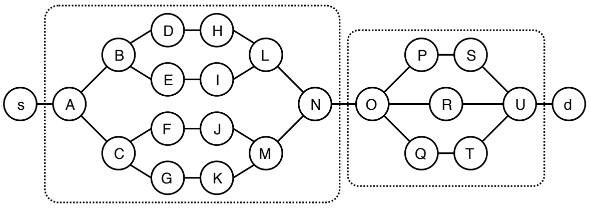


Figure 8.2: Example multi-route with two diamonds

8.3 Diamonds and Branched Route Properties

In this section we characterize load balancer outermost diamonds (referred to as simply *diamonds* in this section) in Internet routes in an attempt to better understand how load balancing is used. Our dataset contains measurements of 188,448 outermost diamonds, 9% of which have unanswered probes. We find that our results are quantitatively similar whether we consider or ignore the 9% of diamonds with unanswered probes; we present results including all diamonds. Among the load balancers measured by multiple MCA executions, we find 97% have a single unique diamond (identical across all measurements of the load balancer), indicating that diamonds are stable. Overall, we find that IPv4 diamonds are more complex, traversing a higher number of load balancers and resulting in greater path diversity.

We revisit Augustin et al.'s original diamond metrics and consider new ones [13]. Figure 8.2 shows a route with two diamonds. Augustin et al. [13] defined the *length* of a diamond as the number of edges in the longest sequence of interfaces across the diamond; the *asymmetry* as the maximum length difference between any sequence of interfaces across the diamond (a diamond with asymmetry zero is said to be symmetric); the *min-width* of a diamond as the number of edge-disjoint sequences of interfaces across the diamond; and the *max-width* as the maximum number of reachable interfaces at any given hop. The min- and max-width give lower and upper bounds on route diversity. We defined the *depth* of a diamond in Section 7 as the maximum number of load balancers traversed by any of its branches. In Figure 8.2, diamond *B–O* is symmetric, has length 5, min-width 2, max-width 4, and depth 2; diamond *P–V* has asymmetry 1, length 3, min-width 3, max-width 3, and depth 1. We have compiled a list of illustrative branched route measurements, which can be interactively inspected in Route Explorer, at `https://mca.speed.dcc.ufmg.br`.

Diamonds are similar across transport protocols

We do not observe significant differences between diamonds measured with TCP, UDP, or ICMP (not shown). In this section we report on diamonds measured using

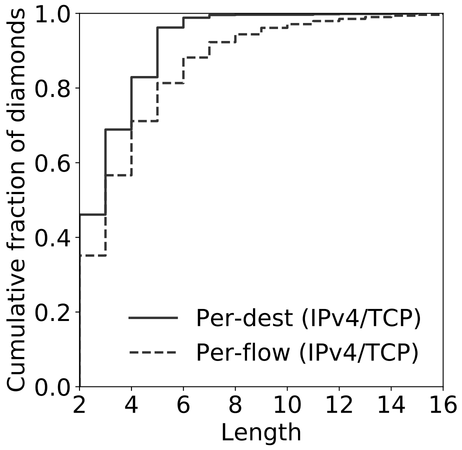


Figure 8.3: Diamond length

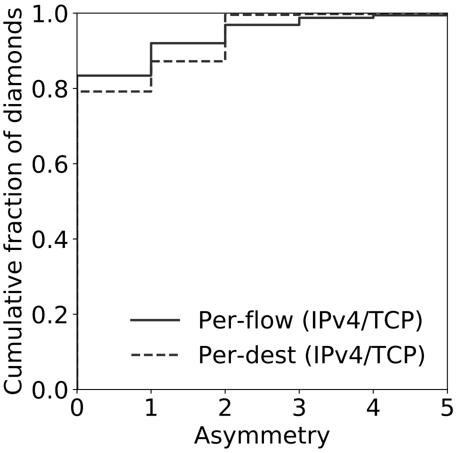


Figure 8.4: Diamond asymmetry

TCP probes, but other transport protocols yield quantitatively similar results. For most metrics, we observe no significant differences between diamonds measured using IPv4 or IPv6, or between diamonds from load balancers with different hash domains; in the following paragraphs we point out the significant differences we identified.

Diamond Length and Asymmetry

Figure 8.3 shows the distribution of diamond lengths for per-flow and per-destination load balancers. We find per-destination diamonds to be shorter than per-flow diamonds, as previously observed by Augustin et al. [13]. We also find that diamonds are usually short (80% of diamonds span at most 5 hops) and that IPv6 diamonds are shorter than IPv4 diamonds (not shown).

We observe asymmetry to be rare and small for both IPv4 and IPv6 (83% of the diamonds are symmetric), following previous results on diamond asymmetry [13]. Figure 8.4 shows the distribution of diamond asymmetry. We find that per-destination load balancers have slightly higher asymmetry than per-flow load balancers.

Diamond Width and Route Diversity

Figure 8.5 shows the distributions of min-width and max-width for diamonds in our dataset. We observe the min-width to be small, with 86% of diamonds having only two edge-disjoint branches. As previously observed by Augustin et al. [13], we find that diamonds are narrow, i.e., their max-width is small, and that IPv4 diamonds are slightly wider than IPv6 diamonds. The difference between min- and max-width is

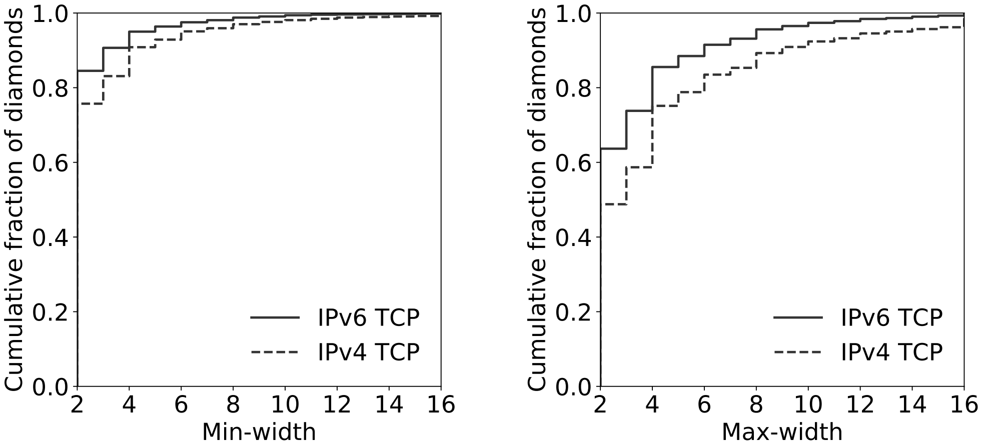


Figure 8.5: Diamond min- and max-width.

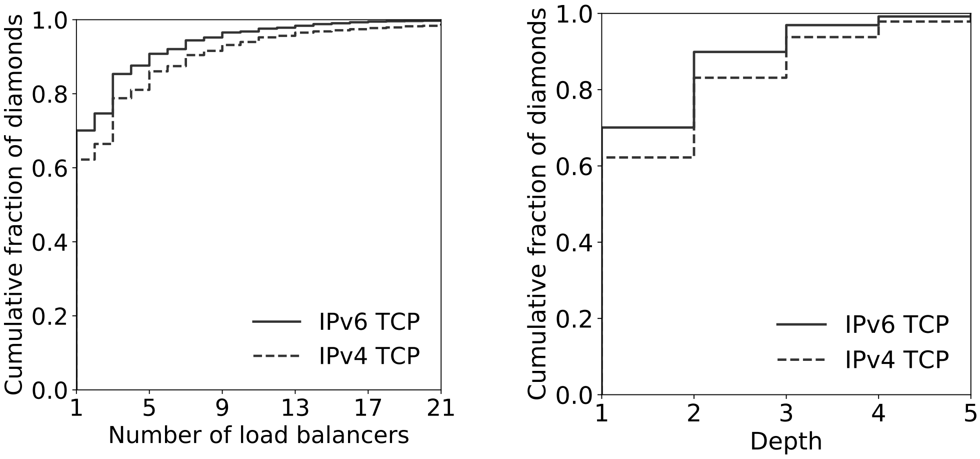


Figure 8.6: Diamonds load balancers and depth

correlated with diamond depth: We find that 86% of the diamonds with a max-width of 8 or larger have a depth of 2 or more.

Augustin et al. [13] previously observed that wide and long IPv4 diamonds are rare. Our measurements corroborate this finding. We also observe that wide and long IPv6 diamonds are considerable more rare than in IPv4, indicating that traffic engineering practices are still more present in IPv4 routes.

Number of Load Balancers and Diamond Depth

Figure 8.6 shows the distribution of the number of load balancers and depth for diamonds in our dataset. We find that diamonds usually have few load balancers, but a few diamonds have more than 20. Diamonds often have between 1 and 3 load balancers, which often appear in a symmetric fan-out pattern as that of routers B, C, and *D* in Figure 8.2. IPv4 diamonds have slightly more load balancers and higher depth than IPv6 diamonds.

The majority of load balancers (80% for IPv4 and 82% for IPv6) have two next hops (not shown). We find less than 1% of load balancers have more than 16 next hops. This may be a limitation of router implementations; e.g., Juniper's JunOS limits routers performing ECMP to 16 next hops [32].

Interdomain Load Balancing

Intradomain routing protocols, such as IS-IS and OSPF, allow the installation of multiple routes through ECMP [111]. This creates instances of intradomain load balancing (intradomain diamonds). BGP, the interdomain routing protocol, selects a single best path towards a destination using a tie-breaking mechanism. Some vendors, however, extend BGP with ECMP support by allowing the BGP decision process to be interrupted before all tie-breakers are considered, leading to selection and load balancing among multiple equally-preferred interdomain routes [112].

Due to challenges in identifying the border between two neighboring networks [113], we use a conservative approach to identify interdomain load balancing and require that diamonds span at least two hops in each of at least two different ASes.

We identify that 4.7% of the IPv4 and 4.2% of the IPv6 diamonds are instances of interdomain load balancing. We validated some instances of interdomain load balancing in out dataset contacting network operators directly. We find NTT America (AS2914) participating in the highest number of interdomain load balancing instances, with a total of 62 other ASes.

Multi-Homing Load Balancing in Origin Networks

Infrastructure and content providers explore rich connectivity at their edges, balancing load across their transit providers or peering networks [29–31]. We observe load balancing across multiple transit providers on measurements from our vantage points in Linode and DigitalOcean. This type of load balancing (on the vantage point's network, not included in the previous discussions) induces high length (up to 18 hops) and asymmetry (up to 8 hops), as branched routes reach the destination by unrelated routes through different providers. All of these load balancers in our dataset are per-flow load balancers, which may have practical implications as applications can use multiple connections to exploit route diversity.

8.4 Overriding of Packet Header Fields

One challenge when trying to identify and classify load balancers whose hash domains include the DSCP, traffic class, and flow label fields is that these fields may be overwritten by intermediate routers and devices on destination networks (e.g. for traffic marking). Although RFCs state routers should not overwrite the IPv6 flow label field, this has been observed on the Internet [114]. Such overwriting will interfere with identification and classification of load balancers by preventing control of the field's value.

ICMP time-exceeded messages encapsulate the header of the expired TTL-limited probe. This allows us to recover the values of the DSCP, traffic class, and flow label fields received by each interface in a multi-route. When we find an interface in a route that receives a DSCP value, traffic class, or flow label field with a value different from the *expected* value, we infer that the preceding interface overwrites that header field. For any overwritten field, we identify whether it is overwritten with a fixed or variable value by looking at multiple encapsulated TTL-limited probes. Note that the *expected* values for DSCP, traffic class, and flow label fields we use for detecting overwriting change after an interface that overwrites them with a fixed value (but become undetermined after an interface that overwrites with a variable value). We identify interface behavior proceeding hop-by-hop starting from the vantage point.

We find that 4.1% of IPv4 interfaces overwrite the DSCP field and that 3.6% of IPv6 interfaces overwrite the traffic class field. Among these, 0.7% overwrite the DSCP and traffic class fields with a variable value. MCA tracks these interfaces to avoid incorrectly inferring per-packet load balancing if subsequent load balancers include DSCP and traffic class in their hash domains. The 3.4% and 3.0% of interfaces that overwrite the DSCP and traffic class fields with a fixed value prevent identification and classification of subsequent load balancing including these fields in their hash domains. This leads to an underestimation of load balancers using the DSCP and traffic class fields reported in §8.2. We did not find interfaces overriding the IPv6's Flow Label field.

Chapter 9

Conclusion and Future Work

In this thesis we presented a more general model for load balancing that considers that load balancers can use arbitrary bits in packet header fields for load balancing. We designed and implemented MCA, a probing algorithm that identifies and classifies load balancing, alongside optimizations to reduce its overall probing cost. We also presented optimizations that reduce MDA's probing cost. Overall, we found that MCA has probing cost comparable to that of previous tools, but provides more complete information about routers, making MCA a practical and useful addition to researchers' and operators' toolsets.

We collected a large dataset to characterize load balancing practices for all combinations of IP protocol version (v4 and v6) and transport protocol (UDP, TCP, and ICMP). Our results show that load balancing is more prevalent and load balancing strategies more mature than previously reported. We identified that existing measurement tools cannot identify 4.7% of load balancers, and will misclassify an additional 2.3%. Given the rise of IPv6 traffic [28,110], programmable data planes [23,24], and software-defined networking [21, 26], these percentages may increase and previous tools may become increasingly inadequate over time.

We also implemented Route Explorer, a visualization tool for Internet route measurements. Route Explorer integrates complementary data such as reverse DNS, IP-to-ASN mapping, AS relationships, and AS business types to show rich and interactive visualization of multi-routes in the Internet.

Compared to previous tools, MCA offers greater visibility of load balancing on Internet routes, but increases the probing cost. We believe that MCA probes can be integrated into techniques that reduce the cost of probing for large-scale measurements such as DTRACK [40]. We also believe MCA can benefit from techniques such as ZMap [58] and Yarrp [51, 59].

Bibliography

[1] E. Katz-Bassett, C. Scott, D. Choffnes, I. Cunha, V. Valancius, N. Feamster, H. Madhyastha, T. Anderson, and A. Krishnamurthy, "LIFEGUARD: Practical Repair of Persistent Route Failures," in *Proc. ACM SIGCOMM*, 2012.

[2] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing Network-wide Traffic Anomalies," in *Proc. ACM SIGCOMM*, 2004.

[3] K. Naidu, D. Panigrahi, and R. Rastogi, "Detecting Anomalies Using End-to-End Path Measurements," in *Proc. IEEE INFOCOM*, 2008.

[4] F. Silveira, C. Diot, N. Taft, and R. Govindan, "ASTUTE: Detecting a Different Class of Traffic Anomalies," in *Proc. ACM SIGCOMM*, 2010.

[5] P. Barford, N. Duffield, A. Ron, and J. Sommers, "Network Performance Anomaly Detection and Localization," in *Proc. IEEE INFOCOM*, 2009.

[6] F. Le, S. Lee, T. Wong, H. Kim, and D. Newcomb, "Minerals: Using Data Mining to Detect Router Misconfigurations," in *Proc. of MineNet*, 2006.

[7] V. Giotsas, M. Luckie, B. Huffaker, and K. Claffy, "IPv6 AS relationships, cliques, and congruence," in *Proc. PAM*, 2015.

[8] M. Luckie, B. Huffaker, K. Claffy, A. Dhamdhere, and V. Giotsas, "AS Relationships, Customer Cones, and Validation," in *Proc. ACM IMC*, 2013.

[9] V. Giotsas, M. Luckie, B. Huffaker, and K. Claffy, "Inferring Complex AS Relationships," in *Proc. ACM IMC*, 2014.

[10] Í. Cunha, P. Marchetta, M. Calder, Y. Chiu, B. Schlinker, B. Machado, A. Pescape, V. Giotsas, H. Madhyastha, and E. Katz-Bassett, "Sibyl: A Practical Internet Route Oracle," in *Proc. USENIX NSDI*, 2016.

[11] H. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "iPlane: an Information Plane for Distributed Services," in *Proc. USENIX OSDI*, 2006.

[12] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira, "Avoiding Traceroute Anomalies with Paris Traceroute," in *Proc. ACM IMC*, 2006.

[13] B. Augustin, T. Friedman, and R. Teixeira, "Measuring Multipath Routing in the Internet," *IEEE/ACM Trans. Netw.*, vol. 19, no. 3, pp. 830–840, 2011.

[14] R. Almeida, O. Fonseca, E. Fazzion, D. Guedes, W. Meira, and Í. Cunha, "A Characterization of Load Balancing on the IPv6 Internet," in *Proc. PAM*, 2017.

[15] V. Jacobson, "traceroute," 1989.

[16] C. Pelsser, L. Cittadini, S. Vissicchio, and R. Bush, "From Paris to Tokyo: On the Suitability of Ping to Measure Latency," in *Proc. ACM IMC*, 2013.

[17] F. Viger, B. Augustin, X. Cuvellier, C. Magnien, M. Latapy, T. Friedman, and R. Teixeira, "Detection, Understanding, and Prevention of Traceroute Measurement Artifacts," *Comput. Netw.*, vol. 52, no. 5, pp. 998–1018, 2008.

[18] P. Marchetta, A. Montieri, V. Persico, A. Pescapé, Í. Cunha, and E. Katz-Bassett, "How and How Much Traceroute Confuses Our Understanding of Network Paths," in *Proc. LANMAN*, 2016.

[19] Í. Cunha, R. Teixeira, and C. Diot, "Measuring and Characterizing End-to-End Route Dynamics in the Presence of Load Balancing," in *Proc. PAM*, 2011.

[20] D. Veitch, B. Augustin, T. Friedman, and R. Teixeira, "Failure Control in Multipath Route Tracing," in *Proc. IEEE INFOCOM*, 2009.

[21] A. Gupta, R. MacDavid, R. Birkner, M. Canini, N. Feamster, J. Rexford, and L. Vanbever, "An Industrial-scale Software Defined Internet Exchange Point," in *Proc. USENIX NSDI*, 2016.

[22] N. Kang, M. Ghobadi, J. Reumann, A. Shraer, and J. Rexford, "Efficient Traffic Splitting on Commodity Switches," in *Proc. ACM CoNEXT*, 2015.

[23] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4:

Programming Protocol-independent Packet Processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, 2014.

[24] A. Sivaraman, A. Cheung, M. Budiu, C. Kim, M. Alizadeh, H. Balakrishnan, G. Varghese, N. McKeown, and S. Licking, "Packet Transactions: High-Level Programming for Line-Rate Switches," in *Proc. ACM SIGCOMM*, 2016.

[25] A. Kumar, S. Jain, U. Naik, A. Raghuraman, N. Kasinadhuni, E. Zermeno, C. Gunn, J. Ai, B. Carlin, M. Amarandei-Stavila, M. Robin, A. Siganporia, S. Stuart, and A. Vahdat, "BwE: Flexible, Hierarchical Bandwidth Allocation for WAN Distributed Computing," in *Proc. ACM SIGCOMM*, 2015.

[26] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: Experience with a Globally-deployed Software Defined Wan," in *Proc. ACM SIGCOMM*, 2013.

[27] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving High Utilization with Software-driven WAN," in *Proc. ACM SIGCOMM*, 2013.

[28] Google, "IPv6 Adoption Statistics by Google," 2019. [Online]. Available: https://www.google.com/intl/en/ipv6/statistics.html

[29] B. Schlinker, H. Kim, T. Cui, E. Katz-Bassett, H. V. Madhyastha, I. Cunha, J. Quinn, S. Hasan, P. Lapukhov, and H. Zeng, "Engineering Egress with Edge Fabric: Steering Oceans of Content to the World," in *Proc. ACM SIGCOMM*, 2017.

[30] K.-K. Yap, M. Motiwala, J. Rahe, S. Padgett, M. Holliman, G. Baldus, M. Hines, T. Kim, A. Narayanan, A. Jain *et al.*, "Taking the Edge off with Espresso: Scale, Reliability and Programmability for Global Internet Peering," in *Proc. ACM SIGCOMM*, 2017.

[31] Z. Zhang, M. Zhang, A. G. Greenberg, Y. C. Hu, R. Mahajan, and B. Christian, "Optimizing Cost and Performance in Online Service Provider Networks," in *Proc. USENIX NSDI*, 2010.

[32] Juniper, "Configuring Per-Packet Load Balancing," 2019. [Online]. Available: https://www.juniper.net/documentation/en_US/junos/topics/usage-guidelines/policy-configuring-per-packet-load-balancing.html

[33] MikroTik, “ECMP Load Balancing With Masquerade,” 2019. [Online]. Available: https://wiki.mikrotik.com/wiki/ECMP_load_balancing_with_masquerade

[34] M. Luckie, “Scamper: a Scalable and Extensible Packet Prober for Active Measurement of the Internet,” in *Proc. ACM IMC*, 2010.

[35] B. Augustin, T. Friedman, and R. Teixeira, “Measuring load-balanced paths in the internet,” in *Proc. ACM IMC*, 2007.

[36] S. Amante, B. Carpenter, S. Jiang, and J. Rajahalme, “RFC 6437 - IPv6 Flow Label Specification,” 2011.

[37] K. Vermeulen, S. D. Strowes, O. Fourmaux, and T. Friedman, “Multilevel MDA-Lite Paris Traceroute,” in *Proc. ACM IMC*, 2018.

[38] CAIDA, “Archipelago (Ark) Measurement Infrastructure,” 2019. [Online]. Available: https://www.caida.org/projects/ark/

[39] RIPE, “RIPE Atlas,” 2013. [Online]. Available: https://atlas.ripe.net

[40] I. Cunha, R. Teixeira, D. Veitch, and C. Diot, “DTRACK: A System to Predict and Track Internet Path Changes,” *IEEE/ACM Trans. Netw.*, vol. 22, no. 4, pp. 1025–1038, 2014.

[41] Cisco, “Configuring a Load-Balancing Scheme,” 2018. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipswitch_cef/configuration/15-mt/isw-cef-15-mt-book/isw-cef-load-balancing.html

[42] MikroTik, “Per-traffic Type Load Balancing,” 2019. [Online]. Available: https://wiki.mikrotik.com/wiki/Per-Traffic_Load_Balancing

[43] D. Bayer, “Visibility of Prefix Lengths in IPv4 and IPv6,” 2010. [Online]. Available: https://labs.ripe.net/Members/dbayer/visibility-of-prefix-lengths

[44] M. Gunes and K. Sarac, “Resolving Anonymous Routers in Internet Topology Measurement Studies,” in *Proc. IEEE INFOCOM*, 2008.

[45] M. Toren, “tcptraceroute: An Implementation of Traceroute Using TCP SYN Packets,” 2001.

[46] D. Kaminsky, “Paratrace,” 2002.

[47] I. Morandi, F. Bronzino, R. Teixeira, and S. Sundaresan, "Service Traceroute: Tracing Paths of Application Flows," in *Proc. PAM*, 2019.

[48] MikroTik, "ICMP Rate Limit," 2019. [Online]. Available: https://wiki.mikrotik.com/wiki/Manual:IP/Settings

[49] Cisco, "IPv6 ICMP Rate Limit," 2019. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipv6/configuration/15-2mt/ip6-15-2mt-book/ip6-icmp-rate-lmt.html

[50] R. Ravaioli, G. Urvoy-Keller, and C. Barakat, "Characterizing ICMP Rate Limitation on Routers," in *Proc. Intl. Conference on Communications*, 2015.

[51] E. Gaston, "High-Frequency Mapping of the IPv6 Internet Using Yarrp," Ph.D. dissertation, Monterey, California: Naval Postgraduate School, 2017.

[52] A. Conta, S. Deering, and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification," 2006.

[53] P. Marchetta and A. Pescapé, "DRAGO: Detecting, Quantifying and Locating Hidden Routers in Traceroute IP Paths," in *Proc. IEEE INFOCOM*, 2013.

[54] M-Lab, "Measurement Lab," 2019. [Online]. Available: https://www.measurementlab.net/

[55] R. Beverly, A. Berger, and G. Xie, "Primitives for Active Internet Topology Mapping: Toward High-Frequency Characterization," in *Proc. ACM SIGCOMM*, 2010.

[56] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP Topologies with Rocketfuel," *IEEE/ACM Trans. Netw.*, vol. 12, no. 1, pp. 2–16, 2004.

[57] B. Donnet, P. Raoult, T. Friedman, and M. Crovella, "Efficient Algorithms for Large-Scale Topology Discovery," in *Proc. ACM SIGMETRICS*, 2005.

[58] Z. Durumeric, E. Wustrow, and J. Halderman, "ZMap: Fast Internet-Wide Scanning and Its Security Applications," in *USENIX Security Symposium*, 2013.

[59] R. Beverly, "Yarrp'ing the Internet: Randomized High-Speed Active Topology Discovery," in *Proc. ACM IMC*, 2016.

[60] T. Cymru, "IP to ASN Mapping," 2019. [Online]. Available: http://www.team-cymru.com/IP-ASN-mapping.html

[61] PeeringDB, "PeeringDB," 2019. [Online]. Available: https://www.peeringdb.com

[62] CAIDA, "AS Rank," 2019. [Online]. Available: http://as-rank.caida.org/

[63] R. Views, "University of Oregon Route Views Project," 2000.

[64] R. NCC, "Routing Information Service (RIS)," 2019. [Online]. Available: https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris

[65] K. Chen, D. Choffnes, R. Potharaju, Y. Chen, F. Bustamante, D. Pei, and Y. Zhao, "Where the Sidewalk Ends: Extending theInternet AS Graph Using Traceroutesfrom P2P Users," *IEEE/ACM Trans. Netw.*, vol. 63, no. 4, pp. 1021–1036, 2013.

[66] X. Dimitropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, K. Claffy, and G. Riley, "AS Relationships: Inference and validation," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 1, pp. 29–40, 2007.

[67] L. Gao, "On inferring autonomous system relationships in the internet," *IEEE/ACM Trans. Netw.*, vol. 9, no. 6, pp. 733–745, 2001.

[68] X. Dimitropoulos, D. Krioukov, and G. R. K. Claffy, "Revealing the Autonomous System Taxonomy: The Machine Learning Approach," in *Proc. PAM*, 2006.

[69] A. Dhamdhere and C. Dovrolis, "Twelve years in the evolution of the internet ecosystem," *IEEE/ACM Trans. Netw.*, vol. 19, no. 5, pp. 1420–1433, 2011.

[70] A. Lodhi, N. Larson, A. Dhamdhere, C. Dovrolis, and K. Claffy, "Using PeeringDB to Understand the Peering Ecosystem," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 2, pp. 20–27, 2014.

[71] A. Bender, R. Sherwood, and N. Spring, "Fixing Ally's Growing Pains with Velocity Modeling," in *Proc. ACM IMC*, 2008.

[72] K. Keys, Y. Hyun, M. Luckie, and K. Claffy, "Internet-scale IPv4 Alias Resolution with MIDAR," *IEEE/ACM Trans. Netw.*, vol. 21, no. 2, pp. 383–399, 2013.

[73] K. Claffy, Y. Hyun, K. Keys, M. Fomenkov, and D. Krioukov, "Internet Mapping: from Art to Science," in *Proc. CATCH*, 2009.

[74] J. Sherry, E. Katz-Bassett, M. Pimenova, T. A. H. Madhyastha, and A. Krishnamurthy, "Resolving IP Aliases with Prespecified Timestamps," in *Proc. ACM IMC*, 2010.

[75] R. Govindan and H. Tangmunarunkit, "Heuristics for Internet Map Discovery," in *Proc. IEEE INFOCOM*, 2000.

[76] M. Luckie, R. Beverly, W. Brinkmeyer, and K. Claffy, "Speedtrap: Internet-Scale IPv6 Alias Resolution," in *Proc. ACM IMC*, 2013.

[77] V. Giotsas and A. Dhamdhere and K. Claffy, "Periscope: Unifying Looking Glass Querying," in *Proc. PAM*, 2016.

[78] E. Katz-Bassett, H. Madhyastha, V. Adhikari, C. Scott, J. Sherry, P. van Wesep, A. Krishnamurthy, and T. Anderson, "Reverse Traceroute," in *Proc. USENIX NSDI*, 2010.

[79] R. Padmanabhan, Z. Li, D. Levin, and N. Spring, "UAv6: Alias Resolution in IPv6 Using Unused Addresses," in *Proc. PAM*, 2015.

[80] M. Luckie, A. Dhamdhere, B. Huffaker, D. Clark, and K. Claffy, "Bdrmap: Inference of Borders Between IP Networks," in *Proc. ACM IMC*, 2016.

[81] R. Beverly, M. Luckie, L. Mosley, and K. Claffy, "Measuring and Characterizing IPv6 Router Availability," in *Proc. PAM*, 2015.

[82] M. Luckie and R. Beverly, "The Impact of Router Outages on the AS-level Internet," in *Proc. ACM SIGCOMM*, 2017.

[83] M. Bagnulo, P. Eardley, T. Burbridge, B. Trammell, and R. Winter, "Standardizing Large-Scale Measurement Platforms," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 2, pp. 58–63, 2013.

[84] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "PlanetLab: An Overlay Testbed for Broad-Coverage Services," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 3, pp. 3–12, 2003.

[85] R. Oliveira, D. Pei, W. Willinger, B. Zhang, and L. Zhang, "The (In)Completeness of the Observed Internet AS-level Structure," *IEEE/ACM Trans. Netw.*, vol. 18, no. 1, pp. 109–122, 2010.

[86] V. Paxson, "End-to-End Routing Behavior in the Internet," *IEEE/ACM Trans. Netw.*, vol. 5, no. 5, pp. 601–615, 1997.

[87] H. Madhyastha, E. Katz-Bassett, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "iPlane Nano: Path Prediction for Peer-to-peer Applications," in *Proc. USENIX NSDI*, 2009.

[88] Z. Zhang, Y. Zhang, Y. C. Hu, Z. M. Mao, and R. Bush, "iSPY: Detecting IP Prefix Hijacking On My Own," in *Proc. ACM SIGCOMM*, 2008.

[89] G. Gürsun, N. Ruchansky, E. Terzi, and M. Crovella, "Routing State Distance: A Path-based Metric for Network Analysis," in *Proc. ACM IMC*, 2012.

[90] E. Katz-Bassett, H. Madhyastha, J. P. John, A. Krishnamurthy, D. Wetherall, and T. Anderson, "Studying Black Holes in the Internet with Hubble," in *Proc. USENIX NSDI*, 2008.

[91] O. Gasser, Q. Scheitle, S. Gebhard, and G. Carle, "Scanning the IPv6 Internet: Towards a Comprehensive Hitlist," in *Proc. TMA*, 2016.

[92] Alexa, "Top Global Sites," 2019. [Online]. Available: http://www.alexa.com/topsites

[93] Cisco, "Umbrella Popularity List," 2019. [Online]. Available: http://s3-us-west-1.amazonaws.com/umbrella-static/index.html

[94] P. Foremski, D. Plonka, and A. Berger, "Entropy/IP: Uncovering Structure in IPv6 Addresses," in *Proc. ACM IMC*, 2016.

[95] A. Murdock, F. Li, P. Bramsen, Z. Durumeric, and V. Paxson, "Target Generation for Internet-Wide IPv6 Scanning," in *Proc. ACM IMC*, 2017.

[96] M. Luckie, Y. Hyun, and B. Huffaker, "Traceroute Probe Method and Forward IP Path Inference," in *Proc. ACM IMC*, 2008.

[97] J. Zhou, M. Tewari, M. Zhu, A. Kabbani, L. Poutievski, A. Singh, and A. Vahdat, "WCMP: Weighted Cost Multipathing for Improved Fairness in Data Centers," in *Proc. ACM EuroSys*, 2014.

[98] B. Donnet, M. Luckie, P. Mérindol, and J. Pansiot, "Revealing MPLS Tunnels Obscured from Traceroute," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 2, pp. 87–93, 2012.

[99] Y. Vanaubel, P. Mérindol, J. Pansiot, and B. Donnet, "A Brief History of MPLS Usage in IPv6," in *Proc. PAM*, 2016.

[100] B. Yao, R. Viswanathan, F. Chang, and D. Waddington, "Topology Inference in the Presence of Anonymous Routers," in *Proc. IEEE INFOCOM*, 2003.

[101] P. Biondi, "Scapy's Documentation." [Online]. Available: https://scapy.net

[102] N. Spring, L. Peterson, A. Bavier, and V. Pai, "Using PlanetLab for Network Research: Myths, Realities, and Best Practices," *SIGOPS Oper. Syst. Rev.*, vol. 40, no. 1, pp. 17–24, 2006.

[103] DCC, "Department of Computer Science, Universidade Federal de Minas Gerais," 2019. [Online]. Available: https://www.dcc.ufmg.br/

[104] Linode, "Linode," 2019. [Online]. Available: https://www.linode.com/

[105] Vultr, "Vultr," 2019. [Online]. Available: https://www.vultr.com/

[106] DigitalOcean, "DigitalOcean," 2019. [Online]. Available: https://www.digitalocean.com/

[107] Majestic, "The Majestic Million," 2019. [Online]. Available: https://majestic.com/reports/majestic-million

[108] X. Fan and J. Heidemann, "Selecting Representative IP Addresses for Internet Topology Studies," in *Proc. ACM SIGCOMM*, 2010.

[109] O. Gasser, Q. Scheitle, P. Foremski, Q. Lone, M. Korczynski, S. D. Strowes, L. Hendriks, and G. Carle, "Clusters in the Expanse: Understanding and Unbiasing IPv6 Hitlists," in *Proc. ACM IMC*, 2018.

[110] Facebook, "IPv6 Traffic Statistics by Facebook," 2019. [Online]. Available: https://www.facebook.com/ipv6/?tab=ipv6

[111] D. Thaler and C. Hopps, "Multipath Issues in Unicast and Multicast Next-Hop Selection," 2000.

[112] P. Lapukhov, "Equal-Cost Multipath Considerations for BGP," 2017. [Online]. Available: https://tools.ietf.org/id/draft-lapukhov-bgp-ecmp-considerations-01.html

[113] A. Marder, M. Luckie, A. Dhamdhere, B. Huffaker, J. M. Smith *et al.*, "Pushing the Boundaries with bdrmapIT: Mapping Router Ownership at Internet Scale," in *Proc. ACM IMC*, 2018.

[114] J. Jaeggli, "IPv6 Flow Label: Misuse in Hashing." [Online]. Available: https://labs.ripe.net/Members/joel_jaeggli/ipv6-flow-label-misuse-in-hashing