A Characterization of Load Balancing on the IPv6 Internet

Rafael Almeida[1], Osvaldo Morais[1], Elverton Fazzion[12]
Dorgival Guedes[1], Wagner Meira Jr.[1], and Ítalo Cunha[1]

[1] Department of Computer Science
Universidade Federal de Minas Gerais
[2] Department of Computer Science
Universidade Federal de São João del-Rei
{rlca, osvaldo.morais, elverton, dorgival, meira, cunha}@dcc.ufmg.br

Abstract As IPv6 deployment grows, it is important to develop new measurement techniques that allow us to study the IPv6 Internet. We implement an IPv6 version of the Multipath Detection Algorithm and use it from 12 geographically-distributed vantage points on two different platforms to characterize IPv6 routers that perform load balancing. Overall, we find that 74% of IPv6 routes traverse at least one router that performs load balancing. Similar to previous reports for IPv4, we find per-destination is the most prevalent type of load balancing; surprisingly, we find a significantly higher prevalence of per-packet load balancing for IPv6 traffic than previously reported for IPv4. We investigate which header fields are used for load balancing, and find that 4% of IPv6 routers that perform load balancing consider IPv6's Traffic Class or Flow Label fields. Finally, we quantify how often routers modify the Traffic Class and Flow Label IPv6 header fields and their impact on load balancing.

Keywords: IPv6, traceroute, measurement, load balancing, topology.

1 Introduction

The growing deployment of IPv6 [7] increases its relevance for application performance and reliability. As a result, the networking community has developed new (and adapted existing IPv4) measurement tools to collect datasets and study the IPv6 Internet (e.g., [4, 12]).

Topology measurements collected with traceroute serve a number of purposes in Internet studies [17]. The introduction of Paris traceroute in 2006 [1] showed that load balancing is widely used in the Internet and causes several measurement artifacts in traceroute measurements. Since then, most traceroute implementations—including those used in Ark, iPlane, and RIPE Atlas—were updated to keep probe flow identifiers fixed to prevent load balancing and avoid measurement artifacts. This approach is adequate for ongoing measurement campaigns, as it prevents artifacts without increasing measurement cost; unfortunately, it does not identify if routers perform load balancing.

Studying load balancing properties, like the number of simultaneous routes between two networks, helps us understand the impact of load balancing on performance (e.g., due to out-of-order packet delivery) and robustness (e.g., against failures and congestion). Studying load balancing also provides insight into traffic engineering practices.

In this paper we implemented an IPv6 version of the Multipath Detection Algorithm (MDA) [16].[3] Our implementation identifies routers that perform load balancing and classifies load balancing behavior by systematically varying four different fields in the IPv6 and TCP headers.

We analyze IPv6 route measurements from 12 vantage points distributed across 7 countries in 3 continents. We characterize the prevalence of load balancing in the IPv6 Internet, different load balancing behaviors, and load balancing properties such as asymmetry. We also study whether routers overwrite the IPv6 traffic class and flow label fields, which might impact load balancing. Whenever possible, we compare our results against previous observations for IPv4 load balancing by Augustin et al. [1]. Our main findings are:

- IPv6 load balancing is widespread, although less so than previously observed for IPv4. We find 74% of IPv6 routes traverse at least one load balancer.
- Similar to IPv4, IPv6 per-destination is the most common class of load balancing. However, we find that IPv6 per-packet load balancing is significantly more common than previously reported for IPv4.
- A non-negligible fraction (4%) of IPv6 routers performing load balancing consider the traffic class and flow label header fields.

Our results further our understanding of the IPv6 Internet; as far as we are aware, this is the first study of IPv6 load balancing. Although IPv6 and IPv4 load balancing have many similarities, we identify differences. In particular, the higher prevalence of per-packet load balancing for IPv6 might negatively impact TCP performance as a result of higher risk of packet reordering.

2 Load balancing

Load balancing is traffic engineering and can be configured manually or automatically by mechanisms such as ECMP and EIGRP. Motivations for load balancing include increasing bandwidth and reducing maximum link utilization. Figure 1 shows a route traversing four routers that perform load balancing (*load balancers*) measured from a vantage point in the Linode cloud hosting service.

Load balancers choose the next hop of a packet based on a *flow identifier* computed from the packet's headers. Augustin et al. [1] defined three classes of load balancers depending on what header fields are used as flow identifiers. In decreasing order of load balancing granularity, the three classes are:

[3] Code available at `https://www.github.com/TopologyMapping/mda6`.

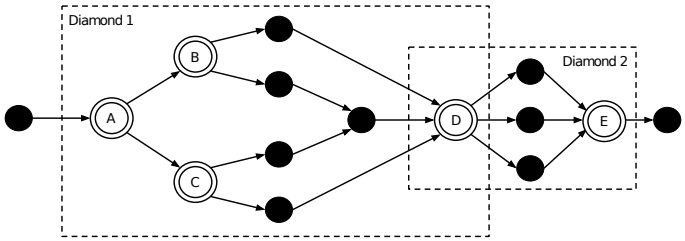

Figure 1: Real route measurement with four load balancers (A, B, C, and D) and two diamonds (A-D and D-E).

- *Per-destination* load balancers use a packet's source and destination IP addresses as the packet's flow identifier. This behavior ensures all packets exchanged between a source and a destination traverse the same sequence of interfaces and experience similar performance.
- *Per-flow* load balancers use a 5-tuple—source and destination addresses, source and destination ports, and protocol number—as the flow identifier. This guarantees that all packets belonging to the same connection will follow the same sequence of interfaces and experience similar performance. Different connections between the same source and destination pair might traverse different sequences of interfaces and experience different performance.
- *Per-packet* load balancers send packets to a random next hop regardless of header field values. Per-packet load balancing may result in packets from the same connection traversing different sequences of interfaces and experiencing different performance. This incurs higher risk of packet reordering, which might negatively impact traffic, e.g., decreasing TCP performance [5].

To detect load balancing, the Multipath Detection Algorithm (MDA) [16] systematically varies the flow identifier in traceroute probes to detect different next hops after a load balancer. MDA proceeds hop-by-hop. MDA assumes each load balancer b in hop h has $N_b + 1$ next hops, where N_b is the number of next hops of b detected so far. MDA then computes the number of probes necessary to identify $N_b + 1$ next hops with a given confidence α, usually set to 0.95.[4] This computation assumes load balancer b distributes flow identifiers uniformly among its next hops. If the number of computed probes is larger than the number of probes already sent to b's next hops, MDA sends additional probes to cover the difference. If the additional probes detect no new next hop, then MDA proceeds to the next load balancer or hop. If the additional probes detect new next hops, then MDA updates N_b and repeats the process.

Augustin et al. [1] characterize load balancer *diamonds*, defined as a subgraph containing all hops between a divergence hop (a load balancer) and a convergence hop, with the condition that all flow identifiers traverse both divergence and

[4] More precisely, MDA computes the number of probes required to bound the probability of not detecting a next hop, across all load balancers, to $1 - \alpha$.

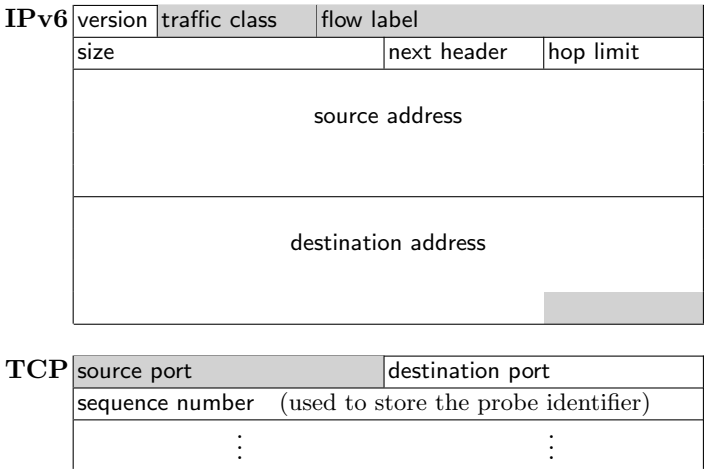

Figure 2: IPv6 header and first 8 bytes of the TCP header. We systematically vary gray bits to identify IPv6 load balancing.

convergence hops. Figure 1 shows two diamonds. Augustin et al. [1] defined the *length* of a diamond as the number of edges in the longest sequence of interfaces across the diamond; the *min-width* of a diamond as the number of edge-disjoint sequences of interfaces across the diamond; the *max-width* as the maximum number of reachable interfaces at any given hop; and *asymmetry* as the maximum length difference between any sequence of interfaces across the diamond (a diamond with asymmetry zero is said to be symmetric). In Figure 1, diamond 1 has asymmetry 1, length 4, min-width 2, and max-width 4; while diamond 2 is symmetric, has length 2, min-width 3, and max-width 3.

3 IPv6 load balancing and measurement methodology

The IPv6 header, shown in Figure 2, is simpler than the IPv4 header. To identify load balancing, we vary the IPv6-specific traffic class and flow label fields. The traffic class field serves a purpose similar to the TOS field in IPv4. RFC2460 says that routers may modify the traffic class field. Other routers or the destination should not expect the traffic class field to have the same value of when the packet was first created. The flow label field allows IPv6 routers to efficiently identify flows.[5] RFC6437 recommends that source hosts set one flow label value for all IPv6 packets belonging to the same connection or application. RFC6437 also specifies that a router may initialize the flow label when it is zero, but should not modify a nonzero flow label.

[5] The usual 5-tuple flow definition used in IPv4 is unsuitable in IPv6 as routers need to follow the variable-length chain of IPv6 extension headers (starting at the next header field) until the end to find the TCP header.

We also vary the last 8 bits of the destination address to identify per-destination load balancers. Current IPv6 prefixes routed in the Internet are less specific than /48s [2]; packets with differences in the last 8 bits of the destination address will take the same route up to the destination's network. Finally, we also vary the TCP source port to check whether IPv6 routers also consider port numbers for per-flow load balancing, as in IPv4. We choose source port numbers starting from 33435, as typically done in traceroute implementations. We use TCP packets to destination port 80 to improve reachability [13] and avoid complaints. We store probe identifiers in the TCP sequence number field.

We execute MDA between each source and destination pair. Each execution varies the four gray fields in Figure 2. This lets us identify load balancers and which header fields they use when computing a probe's flow identifier. At each hop, we probe with up to 256 different flow identifiers, a limitation imposed by the 8 bits available in the traffic class field and in the destination address. Having 256 different flow identifiers lets MDA identify up to 39 distinct next hops at the chosen $\alpha = 0.95$ confidence level; 256 probes were enough for 99.99% of the hops measured.

4 Dataset

We collect IPv6 route measurements[6] from 7 vantage points on CAIDA's Ark platform and from 5 vantage points on the Linode cloud hosting service, as shown on Table 1. The vantage points are spread across 7 countries in 3 continents. Each vantage point measures routes to a list of 51927 destinations built by sampling two addresses from each /48 prefix in a hitlist of 700 thousand IPv6 addresses by Gasser et al. [9]. The dataset was collected on Ark from August 29th to October 3rd, 2016; and on Linode from September 12th to October 3rd, 2016. We chose these platforms because, at the time of writing, PlanetLab does not support IPv6 and RIPE Atlas does not support MDA.

We discard MDA measurements that have loops at the interface level or that do not observe any router (less than 1% of measurements). We *do* consider MDA measurements that do not reach the destination up to the furthest hop common to all four MDA runs toward that destination. We look at IP interfaces and do not perform IP-to-router aliasing; as a result, one (physical) router might be counted multiple times (once for each interface we measure).

For IPv6 to AS mapping we use the AS mapping database provided by Team Cymru.[7] To better understand load balancing behavior, we also queried reverse DNS entries (PTR records) for IPv6 addresses in our measurements.

5 Results

In this section we characterize the prevalence of IPv6 load balancing (§5.1), the behavior of IPv6 routers performing load balancing (§5.2), and diamond prop-

[6] Dataset available at `http://www.dcc.ufmg.br/~cunha/datasets`.
[7] Available at `http://www.team-cymru.org/IP-ASN-mapping.html`.

Table 1: Vantage point locations and prevalence of load balancing

Platform	Location	Routes with load balancing (§5) Overall	Filtered	
Ark	Ballerup, DK (AS59469)	58%	31%	⋆
	Berkeley, CA, US (AS25)	100%	22%	⋆
	Quezon City, PH (AS6360)	16%	16%	
	Los Angeles, US/CA (AS2152)	25%	25%	
	San Diego, US/CA (AS1909)	23%	23%	
	Singapore, SG (AS37989)	99%	27%	⋆
	Barrie, CA (AS19764)	84%	38%	⋆
Linode	Fremont, US/CA	100%	28%	⋆
	London, UK	99%	37%	⋆
	Frankfurt, DE	100%	31%	⋆
	Newark, US/NJ	97%	35%	⋆
	Singapore, SG	98%	38%	⋆

erties (§5.3). Finally, we discuss some IPv6-specific confounding factors (§5.4). Our results mostly match previous reports on the IPv4 Internet, but we discuss a few punctual differences.

5.1 Load balancing prevalence

Table 1 shows the fraction of routes from each vantage point that traverse a load balancer ('Overall' column). We find load balancing is prevalent in IPv6 routes. The heterogeneity among vantage points can be explained by load balancers one or two hops upstream of some of the vantage points (marked with a ⋆). In the case of Linode, these load balancers are inside Linode's own network (as identified by IP-to-AS mapping). These load balancers appear on most routes and significantly impact observations. To remove the impact of these load balancers, we also show the fraction of routes traversing a load balancer when we ignore load balancers two IP hops upstream of vantage points if they are on the same (origin) AS ('Filtered' column). After filtering we observe more homogeneous prevalence of load balancing across vantage points. The filtered results give a better picture of load balancing on IPv6 transit networks and might be representative of other vantage points. We find that of the 45% of routes that traverse a Tier-1 AS, as identified by CAIDA's AS-relationship inference algorithm [10], 29% traverse a load balancer inside the Tier-1.

Figure 3 shows the distribution of the number of load balancers over all routes in our dataset for each platform. We find routes traverse multiple load balancers. (Note that one hop can have multiple load balancers, e.g., hop 2 in Figure 1.) In particular, 76% of Linode routes traverse three or more load balancers. This is because routes often traverse three load balancers in Linode's network (see 'Diamond 1' in Figure 1). Figure 3 also shows the number of load balancers traversed when we ignore load balancers two IP hops upstream of

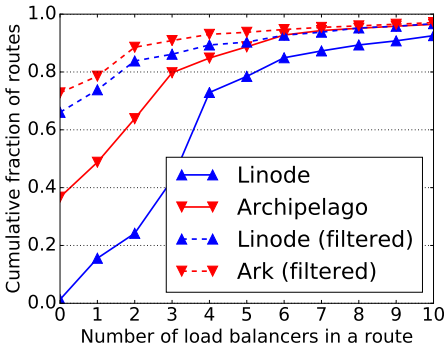

Figure 3: Load balancers

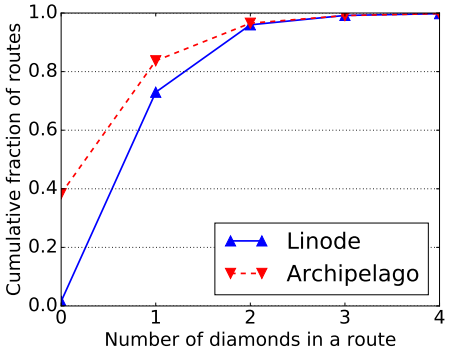

Figure 4: Number of diamonds

vantage points if they are on the same (origin) AS (dotted lines). After filtering, we observe similar load balancing from Ark and Linode vantage points.

Figure 4 shows the distribution of the number of diamonds over all routes in our dataset. As diamonds start and end on interfaces that all packets traverse, the number of diamonds on a route gives a lower bound of the number of load balancers that packets traverse to reach the destination. Although Figure 3 shows routes can traverse many load balancers, these are grouped into a small number of diamonds. As we will show later (§5.3), diamonds are complex and contain many load balancers. This result is similar to previous results for IPv4 load balancing [1].

5.2 Classes of load balancing behavior

We now investigate what IPv6 header fields load balancers use to compute flow identifiers to choose next hops. We identify load balancers by their IPv6 addresses. Table 2 shows the fraction of load balancers in each class and the percentage of routes that traverse at least one load balancer in each class. We also report results from Augustin et al. [1] for IPv4 load balancers. (Note that Augustin's results are from 2011, so the differences we discuss might also be due to network evolution and not only IP version).

We find per-destination, per-flow, and per-packet load balancers are not only the most common load balancer classes, but also the most prevalent across route measurements. This is expected, as these classes were used for IPv4 load balancing. Despite this similarity, we observe a significantly higher fraction of IPv6 routes traverse per-packet load balancers. We discuss this further in §5.4.

We also find other classes of load balancers. We find 3.2% of load balancers perform per-flow load balancing considering the traffic class field (in addition to the destination address and source port). This behavior is the default in at least JunOS 15.1. We could not find any reports on how many IPv4 load balancers consider the TOS field to compare. Interestingly, we find 6% of load balancers that use only the TCP ports for load balancing. We manually investigated these

Table 2: Classes of load balancing behavior

	Overall			Filtered	
	Fraction of Balancers	% Routes IPv6	% Routes IPv4 [1]	Fraction of Balancers	% Routes IPv6
Per-destination	29.3%	43.5%	78.0%	29.2%	11.1%
Per-flow	50.0%	30.0%	54.8%	50.1%	17.7%
Per-packet	10.7%	30.1%	1.0%	10.6%	7.7%
Per-flow with TC	3.2%	14.8%	—	3.2%	3.3%
Per-application	6.0%	5.1%	—	6.0%	3.3%
Others	0.8%	1.2%	—	0.9%	0.6%
Total	**100%**	**74%**	**92%**	**100%**	**29%**

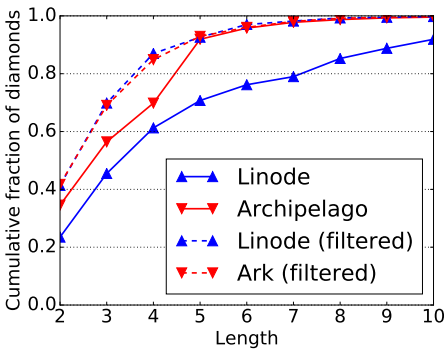

Figure 5: Diamond length

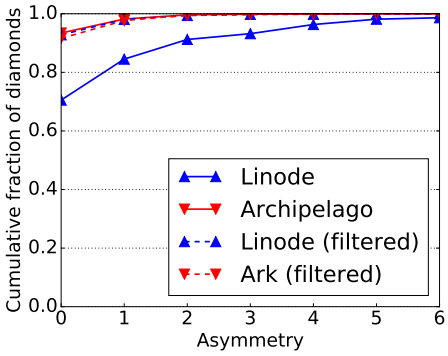

Figure 6: Diamond asymmetry

load balancers and found this behavior can be configured in RouterOS under the name of "per-application load balancing." Perhaps surprisingly, we find only 0.8% of load balancers that consider IPv6's flow label (with or without other fields). Overall, 4% of the load balancers consider either IPv6's traffic class or flow label fields.

5.3 Diamond characteristics

We now characterize diamonds on routes with load balancing using the same methodology and metrics as Augustin et al. [1] and compare the observations.

Diamond length. Figure 5 shows the distribution of diamond lengths. We find diamonds are usually short, and that load balancers one or two hops upstream of vantage points have longer diamonds than average. If we ignore these load balancers (dashed lines), then both datasets observe very similar distributions of diamond length, with 93% of diamonds of length 5 or less.

Diamond asymmetry. Figure 6 shows the distribution of diamond asymmetry in our dataset. Linode has asymmetric diamonds in its network that show up on many routes (solid blue line). If we ignore load balancers one or two hops

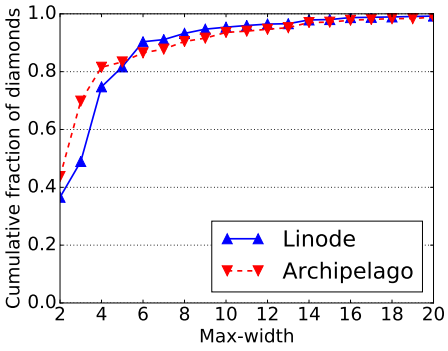

Figure 7: Max-width

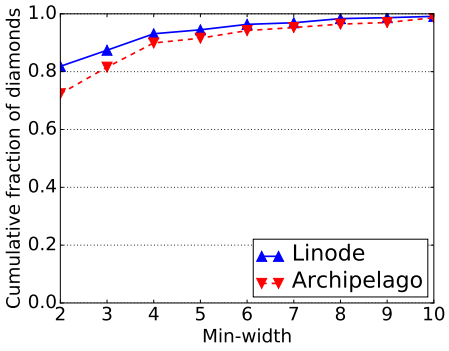

Figure 8: Min-width

upstream of the vantage point (dashed lines), these diamonds are not considered and we observe that 96% of diamonds are symmetric. The few asymmetric diamonds usually have asymmetry less than or equal to 2. We find that 71% of asymmetric diamonds are instances of inter-domain load balancing, i.e., when the diamond starts and ends in different ASes.[8] This illustrates that more complex inter-domain traffic engineering leads to more complex load balancing configurations. Conversely, only 26% of symmetric load balancers are instances of inter-domain load balancing. Different inter-domain routes might have different performance, fortunately, we find that 70% of asymmetric diamonds start at per-destination load balancers, which will send all packets from the same source and destination pair on the same inter-domain route.

Diamond max-width. Figure 7 shows the distribution of max-width for all diamonds in our dataset. By definition, the minimum max-width for a diamond is 2. We find most diamonds are narrow, in particular around 81% of diamonds across both platforms have max-width equal or less than 5. By varying the last 8 bits of the destination address, probe packets will follow the same route since IPv6 prefixes in global routing tables are usually shorter than /48 [2]. However, when reaching the destination network, packets may be directed to different hosts whose addresses share a /116 prefix with the destination. (We found this to be common in Microsoft's datacenters.) Figure 7 includes these measurements as diamonds with very large max-width, and shows that such errors are rare and do not impact the overall findings.

Diamond min-width. Figure 8 shows the distribution of min-width over all diamonds in our dataset. By definition, the minimum min-width is 2 and is bounded by the max-width. We find most diamonds have a min-width of 2.

[8] This can happen as a result of traffic engineering or, for example, when a BGP router with ECMP enabled receives and installs multiple routes to a prefix (e.g., at an IXP) or when multiple BGP routers redistribute different routes to the same prefix into an IGP (e.g., OSPF) with ECMP enabled.

Table 3: Fraction of routers that overwrite the traffic class and flow label fields.

Overwriting Behavior	Field: Traffic Class	Field: Flow Label
Variable value	0.7%	0.0%
Fixed value	4.7%	0.0%

Comparison to IPv4 diamonds. Our findings for diamond lengths, asymmetry, max-widths, and min-widths are similar but not quantitatively close to findings on IPv4 load balancers by Augustin et al. [1]. For example, they found that load balancers are often short and narrow, and reported that 55% of routes with load balancing traverse a diamond of length 2 and max-width less than or equal to 3; in our dataset, we find 24% diamonds of this kind. Augustin et al. also found that long and wide diamonds are rare; in our dataset, only 14% of diamonds have both length and max-width larger than 3. Similar to our results, Augustin et al. also found that most IPv4 load balancers are symmetric.

5.4 Confounding factors

Routers may override the traffic class or flow label fields for traffic engineering or other reasons. Such routers may interfere with our identification of load balancers by modifying a probe's traffic class or flow label fields with a variable value when we try to keep the values fixed, and by overwriting fields with a fixed value when we try to vary them.

ICMPv6 time-exceeded messages encapsulate the header of the expired TTL-limited probe. We use the encapsulated header to identify the values of the traffic class and flow label fields on all probes received by each router in a route. If we identify a router that received a probe with a traffic class or flow label field different from the expected value we infer that the previous router has overwritten it. If the field is overwritten, we identify whether it is overwritten with a fixed or variable value. Note that the 'expected' values for the traffic class and flow label fields change along the route as routers overwrite them. We identify router behavior proceeding hop-by-hop starting from the vantage point.

We find that a small (but not negligible) portion of the routers overwrite the traffic class field. Table 3 summarizes router behavior. The few routers (0.7%) that overwrite the traffic class field with a variable value might lead to the (incorrect) identification of per-packet load balancers. Routers that overwrite the traffic class field with a fixed value do not impact the identification of load balancing, but prevent us from identifying whether routers use the traffic class field for load balancing (leading to underestimation of 'per-flow with TC' in Table 2).

In general, traceroute measurements are challenged by factors such as tunneling and router behavior [14] as well as routers that do not respond to TTL-expired probes or firewalls that drop measurement probes [13]. As a result of these factors, we might underestimate the amount of load balancing.

6 Related work

Load balancing and its impact. The impact of load balancing on IPv4 traceroute measurements was first reported on by Augustin et al. in 2006 (see [1]). Since then, MDA has been proposed to bound load balancer identification errors [16] and an extensive characterization of IPv4 load balancing was published [1]. Paris traceroute was the first, but today most traceroute tools and measurement platforms keep flow identifiers fixed to avoid load balancing. Besides impacting traceroute measurements, load balancing has also been reported to impact latency measurements [15] and observed routing dynamics [6].

IPv6 measurement tools and characterization studies. As far as we are aware, Scamper [11] is the only other implementation of MDA that supports IPv6. Also, we are not aware of any other characterization of IPv6 load balancers. Other work have developed techniques to measure IPv6 routers, including IPv6 alias resolution [3, 12] and router availability [4]; while others have quantified IPv6 deployment and performance [7, 8].

7 Conclusions and future work

We implemented an IPv6 version of the MDA to identify routers that perform load balancing and classify their behavior. We collected measurements from 12 nodes in 7 countries to 51927 destinations. We find that IPv6 load balancing shares many similarities with IPv4 load balancing, with a few differences. First, although IPv6 load balancing is widespread, it is less so than IPv4 load balancing. Second, IPv6 routes have significantly higher probability of traversing per-packet load balancers than IPv4 routes, which may negatively impact TCP performance. Although we cannot explain the causes behind the higher prevalence of per-packet load balancers, this is partially explained by routers that overwrite the traffic class field in IPv6 headers with variable values. Other possible explanations include less mature IPv6 load balancing implementations or less established best practices when compared to IPv4.

The prevalence of per-packet load balancers we observe motivate investigation of the impact of IPv6 load balancing on IPv6 traffic. As future work, we plan to correlate performance metrics with load balancing behavior. We also plan to extend our MDA implementation to allow more fine-grained classification of load balancers. In particular, we plan to add support for IPv6 extension headers and to allow measurements varying a combination of fields in probe headers.

Acknowledgements

We thank Young Hyun for the support in setting up and running our measurements on the Ark platform. This work is supported by Comcast and Brazilian research funding agencies (CAPES, CNPq, and FAPEMIG).

References

1. Augustin, B., Friedman, T., Teixeira, R.: Measuring Multipath Routing in the Internet. IEEE/ACM Trans. Netw. 19(3), 830–840 (2011)
2. Bayer, D.: Visibility of Prefix Lengths in IPv4 and IPv6 (2010), `https://labs.ripe.net/Members/dbayer/visibility-of-prefix-lengths`
3. Beverly, R., Brinkmeyer, W., Luckie, M., Rohrer, J.P.: IPv6 Alias Resolution via Induced Fragmentation. In: International Conference on Passive and Active Network Measurement (PAM) (2013)
4. Beverly, R., Luckie, M., Mosley, L., Claffy, K.: Measuring and Characterizing IPv6 Router Availability. In: International Conference on Passive and Active Network Measurement (PAM) (2015)
5. Blanton, E., Allman, M.: On Making TCP More Robust to Packet Reordering. ACM SIGCOMM Computer Communication Review 32(1), 20–30 (2002)
6. Cunha, I., Teixeira, R., Diot, C.: Measuring and Characterizing End-to-End Route Dynamics in the Presence of Load Balancing. In: International Conference on Passive and Active Network Measurement (PAM) (2011)
7. Czyz, J., Allman, M., Zhang, J., Iekel-Johnson, S., Osterweil, E., Bailey, M.: Measuring IPv6 Adoption. Proc. SIGCOMM (2014)
8. Dhamdhere, A., Luckie, M., Huffaker, B., claffy, k., Elmokashfi, A., Aben, E.: Measuring the Deployment of IPv6: Topology, Routing and Performance. In: Proc. ACM Internet Measurement Conference (IMC) (2012)
9. Gasser, O., Scheitle, Q., Gebhard, S., Carle, G.: Scanning the IPv6 Internet: Towards a Comprehensive Hitlist. In: Proc. Traffic Monitoring and Analysis Workshop (TMA) (2016)
10. Luckie, M., Huffaker, B., kc claffy, Dhamdhere, A., Giotsas, V.: AS Relationships, Customer Cones, and Validation. In: Proc. ACM Internet Measurement Conference (IMC) (2013)
11. Luckie, M.: Scamper: a Scalable and Extensible Packet Prober for Active Measurement of the Internet. In: Proc. ACM Internet Measurement Conference (IMC) (2010)
12. Luckie, M., Beverly, R., Brinkmeyer, W., claffy, k.: Speedtrap: Internet-scale IPv6 Alias Resolution. In: Proc. ACM Internet Measurement Conference (IMC) (2013)
13. Luckie, M., Hyun, Y., Huffaker, B.: Traceroute Probe Method and Forward IP Path Inference. In: ACM Internet Measurement Conference (IMC) (2008)
14. Marchetta, P., Montieri, A., Persico, V., Pescape, A., Cunha, I., Katz-Bassett, E.: How and How Much Traceroute Confuses Our Understanding of Network Paths. In: Proc. International Symposium on Local and Metropolitan Area Networks (LANMAN) (2016)
15. Pelsser, C., Cittadini, L., Vissicchio, S., Bush, R.: From Paris to Tokyo: On the Suitability of Ping to Measure Latency. In: Proc. ACM Internet Measurement Conference (IMC) (2013)
16. Veitch, D., Augustin, B., Friedman, T., Teixeira, R.: Failure Control in Multipath Route Tracing. In: Proc. IEEE International Conference on Computer Communications (INFOCOM) (2009)
17. Willinger, W., Roughan, M.: Internet Topology Research Redux. In: Recent Advances in Networking, ACM SIGCOMM eBook, Volume 1 (2013)