The Razor's Edge: IPv6 Extension Headers Survivability

Justin Iurman[1][0000-0001-9561-1856], Benoit Donnet[1][0000-0002-0651-3398]

Université de Liège, Montefiore Institute, Belgium
{justin.iurman, benoit.donnet}@uliege.be

Abstract. While IPv6 was standardized in the 90's, only the last decade has seen a growth in its global adoption. In addition to dealing with IPv4 addresses exhaustion, IPv6 comes with a mechanism, called IPv6 Extension Header (IPv6 EH), allowing the protocol to be more flexible and extensible. In this paper, we investigate how IPv6 EHs are processed in the network. In particular, we focus on the survivability of IPv6 EHs, i.e., the fact that an IPv6 EH traverses the Internet and arrives unmodified at the destination. We first design experiments in a controlled environment, testing different IPv6 EHs and sizes on different routers from various vendors. Then, we confront our observations with several measurement campaigns between vantage points hosted by different Cloud Providers (CPs) around the world, and we compare them to the responses received from a survey of operators. Our results show that the survivability of IPv6 EHs is quite limited (around 50%) and is a consequence of operators' policies, with some Autonomous Systems being responsible for most of the IPv6 EHs drops. Measurement tool and data collected are provided to the research community.

1 Introduction

During the last decade, IPv6 has been more and more adopted [23]. If IPv6 allows for dealing with IPv4 address exhaustion [26], it also comes with a mechanism, called IPv6 *Extension Header* (IPv6 EH) [6, 13], that leads to more flexibility and innovation. Examples of such innovations based on IPv6 EHs are Segment Routing with IPv6 as forwarding plane [17, 18] and In-Situ Operations, Administration, and Maintenance (IOAM) [5] for in-band telemetry. The purpose of IPv6 EHs is to extend IPv6 without any modification to the core protocol. IPv6 EHs form a chain, using the IPv6 *Next Header* field, and are placed between the IPv6 header and the upper-layer protocol header. While new IPv6 EHs might be defined in the future, the current list mainly includes the `Hop-by-Hop Options` Header, the `Destination Options` Header, the `Routing` Header, the `Fragment` Header, the `Encapsulating Security Payload`, and the `Authentication` Header [6,13]. Up to now, few efforts have been made in assessing how operators process IPv6 EHs, e.g. [6, 14, 21, 24, 35, 47]], focusing mainly on a subset of IPv6 EHs or relying on limited measurements campaign.

In this paper, we provide a comprehensive view of how IPv6 EHs are processed in the network. In particular, we are interested in IPv6 EHs *survivability*,

i.e., the capacity of IPv6 EHs to traverse the Internet and arrive unmodified at the destination. This is important as we expect a complete survivability for some IPv6 EHs, such as the `Destination Options` Header or `Fragment` Header, but not necessarily for some others that are more designed for limited domain use cases (e.g., the `Hop-by-Hop Options` Header or `Routing` Header). Also, a too low level of survivability may damage the IPv6 extensibility and, consequently, innovation, leading to an ossification of IPv6. More precisely, this paper makes the following contributions:

- We develop an eBPF [45] program called eBP**F** **I**Pv6 Extens**i**on **H**eaders **Inject**ion (FishNet) to easily inject IPv6 EHs, whatever the type and size, in network traffic.
- We build a controlled environment to perform measurements with FishNet spanning all specified IPv6 EHs, with different parameters such as the IPv6 EH type and size, and so for routers from various vendors. We show that all IPv6 EHs have a perfect survivability rate with default configuration on routers, which tends to suggest that potential drops of packets with IPv6 EHs is mainly caused by operators' policies.
- Next, in order to determine whether our lab observations are applied in the real world, we deploy FishNet in different Cloud Providers (CPs) scattered around the world and perform measurements in full mesh. Our results show that, on the contrary to controlled environment experiments, IPv6 EHs survivability is quite limited (around 50% on average). We also show that IPv6 EHs drop is caused by some ASes, generally quite close to the packet source. We also compare those observations with the results from a survey of operators.
- Measurement software (i.e., FishNet) and collected data are provided to the research community.

The remainder of this paper is organized as follows: Sec. 2 provides the required background for this paper; Sec. 3 describes FishNet, the tool we implemented to inject IPv6 EHs in network traffic; Sec. 4 investigates IPv6 EHs survivability in a controlled environment; Sec. 5 introduces our Internet measurement methodology; Sec. 6 discusses our Internet measurement results; Sec. 7 positions this paper with respect to the state of the art; finally, Sec. 8 concludes this paper by summarizing its main achievements.

2 Background

The purpose of IPv6 EHs is to extend IPv6 without any modification to the core protocol. The IPv6 `Next Header` field specifies which upper-layer protocol comes after the IPv6 header. All IPv6 EHs share a common field in their respective headers, namely a `Next Header` field, whose name and purpose are identical to the one in the IPv6 header. This design allows for a chaining mechanism. Fig. 1 illustrates how it works with three examples: the first one represents a `TCP` segment, the second one represents a `Routing` Header followed by a `TCP` segment,

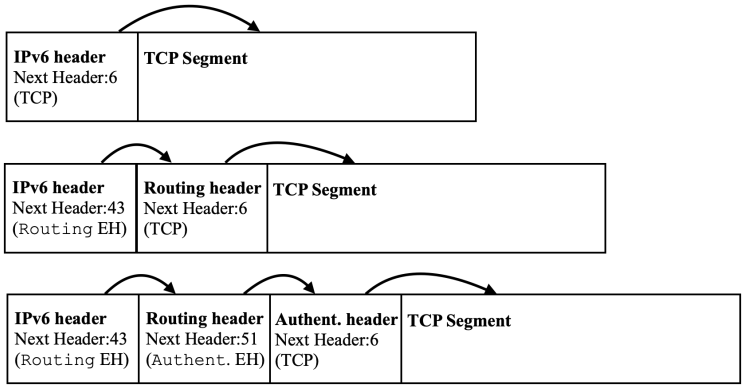


Fig. 1. Example of chain of pointers formed by the `Next Header` field in IPv6.

and the third one represents a `Routing` Header followed by an `Authentication` Header followed by a `TCP` segment.

The Internet Assigned Number Authority (IANA) currently defines the following IPv6 EHs [29]: the `Hop-by-Hop Options` Header, the `Destination Options` Header, the `Routing` Header, the `Fragment` Header, the `Encapsulating Security Payload`, the `Authentication` Header, the `Mobility` Header, the `Host Identity Protocol` Header, and the `Shim6 Protocol` Header. The `Hop-by-Hop Options` Header is used to carry optional information, also called *Options*, that may be examined and processed by every node along a packet's delivery path, while the `Destination Options` Header is used to carry optional information to be examined only by the packet's destination. An example of `Hop-by-Hop Options` Header or `Destination Options` Header usage is In-Situ Operations, Administration, and Maintenance (IOAM) [5]. With IOAM, telemetry data is carried within packets rather than being sent through packets specifically dedicated to that. The IOAM traffic is thus embedded in data traffic, but not part of the packet payload. The `Routing` Header is used by an IPv6 source to list one or more intermediate nodes to go through on the way to a packet's destination (i.e., to steer a packet), and has several types defined: Source route (type 0) and Nimrod (type 1) [7] which are both deprecated, Mobility support (type 2) [32], RPL (type 3) [25], and Segment Routing (type 4) [17]. The `Fragment` Header is used by an IPv6 source to send a packet larger than it would fit in the path MTU to its destination. It works like IPv4 fragmentation except that only the packet source can fragment the packet. The `Authentication` Header (sender authentication, data integrity) [33] and `Encapsulating Security Payload` (sender authentication, data integrity, confidentiality) [34] are both part of the IPsec protocol suite. The `Mobility` Header is used to allow devices to move from one network to another while maintaining a permanent IPv6 address. The `Host Identity Protocol` Header is used to separate the end-point identifier and locator roles of IPv6 addresses [39]. The `Shim6 Protocol` Header is used to determine valid locator pairs that could be used when an outage is detected [41].

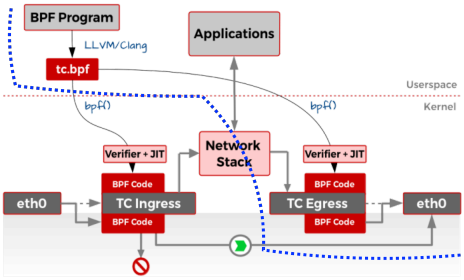


Fig. 2. Overview of how Tc works with eBPF [46].

3 FishNet

This section describes and evaluates eBP**F** **I**Pv6 Exten**s**ion **H**eaders I**nject**ion (FishNet), our tool for easily injecting IPv6 EHs in network traffic. The reason for using eBPF is twofold: (*i*) a fast implementation compared to the modification of existing probing solutions; and (*ii*) it injects IPv6 EHs in "real" traffic, not only the one from probing tools, which is really interesting as it allows for some corner cases to be tested, e.g., one may inject IPv6 EHs after the `TCP` 3-way handshake to check whether stateful filters influence the IPv6 EHs processing.

3.1 Overview

In order to inject one or multiple IPv6 EHs in outgoing traffic using eBPF, FishNet must be attached to an interface. More specifically, one needs to add a `clsact qdisc` [37] to an interface, which is like a scheduler holding only classifiers and that works both on ingress and egress. Then, FishNet must be attached to an egress filter on that interface, with a specific section to be run. Both commands use Tc [44], a traffic control tool as part of the `iproute2` [30] solution. Finally, the user program is used to configure the IPv6 EHs injection. Fig. 2 provides a high-level picture of how it works. What was previously described is represented on the right side of the dashed blue line.

Overall, FishNet is completely agnostic of whether one or more IPv6 EHs are injected, or their order. The only thing that it knows is that it has to inject a buffer of bytes. Therefore, the overhead only depends on the number of bytes to inject (see Sec. 3.2 for performance evaluation). Indeed, the buffer construction is delegated to the user program which is responsible for configuring what will be injected (one or more IPv6 EHs, their order, etc). Very briefly, IPv6 EHs can be injected with constraints on respective sizes, and in any order. If the chosen order does not respect RFC8200 [13], an error is returned as a security, although the user could force such a behavior with a special flag.

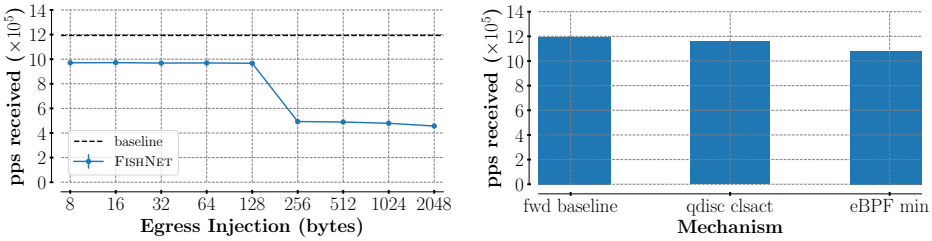
3.2 Evaluation

To evaluate FishNet, we rely on TRex [8], an open source, low cost, stateful and stateless traffic generator fueled by DPDK. It has multiple advantages, such as the ability to generate Layer3–7 traffic and multiple streams, as well as the ability to easily craft your own packets with the underlying Scapy [43] layer. TRex can scale up to 200Gbps with only one server.

The testbed is straightforward: one machine for TRex, and another one for the *Device Under Test* (`DUT`). Both are equipped with an *Intel XL710 2×40GB QSFP+* NIC, each connected port to port in order to close the loop (i.e., TRex client and server run on the same machine). This kind of topology provides an easy way to isolate a specific function on the `DUT` and evaluate it, i.e., the egress injection of IPv6 EHs with FishNet. The `DUT` has an *Intel Xeon cpu e5-2630 v3 at 2.40GHz*, with 8 Cores, 16 Threads, and has a 16GB RAM. It runs a kernel version $6.9.0 - rc6+$ (`net-next`) and FishNet was compiled with clang version 14.0.6. Equivalent `iproute2` version has been compiled with `libbpf` 1.4.0. During measurements, the `DUT` is configured to maximize its performance (e.g., CPU in performance mode, network settings). It is also configured to only use one queue for all traffic received, therefore only one core being responsible for that queue. Doing so allows us to see the impact on a single core, which is better to compare performance on a common basis. The `MTU` is set to $2,148$ so that the maximum size injected (i.e., $2,048$ bytes) would not make packet sizes to exceed it. Overall, each experiment (i.e., measurement) lasts 30 seconds and is run 20 times. We determine 95% confidence intervals for the mean based on the Student t distribution (they are too tight to be visible in the subsequent plots).

As explained in Sec. 3.1, i.e., FishNet is completely agnostic of the content of the buffer, only the number of bytes to be injected may have an impact on performance, whatever the combination of IPv6 EHs. Since most of these IPv6 EHs are limited to a maximum of $2,048$ bytes (except for the `Encapsulating Security Payload`), we evaluate the impact of an injection from 0 to $2,048$ bytes, although, less likely, a combination of IPv6 EHs could result in a much bigger buffer.

Fig. 3a shows the impact of different injection sizes on throughput. The forwarding baseline is, in our case, approximately $1,195,000$ packets per second (pps) on a single core (roughly 14.34Gbps with $1,500$-byte packets). One can directly observe a loss of 19% (i.e., approximately $225,000$ pps) when injecting the minimum size of 8 bytes, which then remains stable up to 128 bytes. The fact that such a loss occurs immediately will be discussed below based on Fig. 3b. When injecting 256 bytes or more, the loss rate bumps to 58%. This huge drop is due to a lack of space in the `sk_buff` headroom, where the headers of a packet are located, which involves implicit reallocation by the kernel to make the headroom larger. Depending on the architecture and the NIC driver, the headroom space may vary. In our case, i.e., $\times 86\_64$ architecture and $i40e$ driver, the headroom has an initial size of 256 bytes. If we remove 14 bytes for the Ethernet header (in our case), plus 2 bytes to align the IPv6 header, plus 40 bytes for the IPv6



(a) IPv6 EHs injection with FishNet and its impact on throughput.

(b) Impact on throughput between adding a `clsact qdisc` [37] and running a minimal version of a Tc/eBPF program on egress.

Fig. 3. FishNet performance evaluation.

header, we are left with 200 bytes available in the headroom. This means that as soon as we inject 201 bytes or more, the drop will happen, which is indeed between 128 and 256 in Fig. 3a.

Fig. 3b shows the impact on throughput when only a `clsact qdisc` is added and it is compared to when a minimal[1] eBPF kernel program is running. One can see that only adding the `qdisc` gives a 3% loss already, while running a minimal eBPF kernel program gives an additional 7% loss, which makes it a total of 10% loss. Despite being out of scope of this paper, it would be interesting to investigate if improvements can be made on that part. In fine, the real loss rate of FishNet is 9%, in addition to the initial and unavoidable 10% loss.

Another interesting observation is that some network drivers (tested with *e1000e* and *vmxnet3*) have issues with TX checksum offloading when there is an IPv6 EH or more in a packet, with or without FishNet. Indeed, the checksum calculated in Layer-4 is incorrect, even if adding one or more IPv6 EHs should have no impact since it does not modify the pseudo-header (except when there is a `Routing` Header, where the destination in the pseudo-header is the last segment). As a result, packets may disappear along the path, which could wrongly suggest a Layer-2 problem and is therefore hard to debug. Despite being out of scope of this paper, it should also be investigated to help NIC vendors address this issue. Note that we have also started fixing some bugs related to checksums with IPv6 EHs in the Linux kernel.

Finally, it is worth mentioning this section evaluates the worst case, i.e., line rate traffic on a single core. Overall, it is highly unlikely we would need FishNet to inject IPv6 EHs at line rate on a single core. For example, in Sec. 4 to Sec. 6, FishNet is used to carefully inject IPv6 EHs in `traceroute` traffic such that we avoid losing packets or hitting rate limits. However, independently of FishNet and its usage, people may want to inject IPv6 EHs in line-rate traffic. The initial cost shown in Fig. 3b could prevent them from following that path.

[1] Minimal means the section handler directly returns `TC_ACT_OK`.

Table 1. List of tested routers in our controlled environment.

	Vendor	Model	Version
R1	Cisco	ASR1001-X (ASIC based)	IOS XE 03.16.05.S
R2			IOS XE 17.06.04
R3		ASR9904 (NPU based)	IOS XR 7.9.21
R4	Huawei	AR617VW-LTE4EA	V300R019C10
R5	Juniper	vMX	Junos OS 20.2R1.10
R6	Linux	–	Kernel 6.11
R7	Nokia	7750 SR-7	20.10

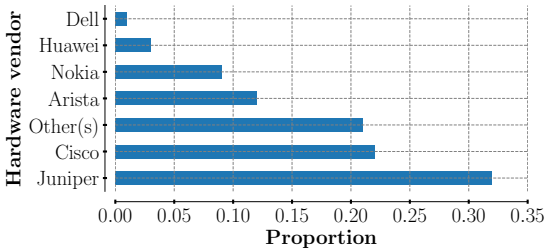


Fig. 4. Distribution of hardware vendors in our survey of operators.

4 IPv6 EHs Processing in a Controlled Environment

This section investigates the capacity of IPv6 EHs to traverse a single router and arrive unmodified at the destination, in a controlled environment. We first describe our infrastructure (Sec. 4.1) and, next, discuss our results (Sec. 4.2).

4.1 Infrastructure Setup

Our controlled environment is made of two devices, i.e., the sender and the receiver, and a physical router (i.e., the `DUT`, a real hardware) in between them. We evaluate seven different routers (`DUT`s) separately, each of them running a default configuration. Those routers are listed in Table 1 and were deliberately chosen based on the responses received from our survey of operators.[2] Fig. 4 shows the various hardware deployed by operators, according to our survey. Both Juniper and Cisco routers seem to share the biggest part of the market [1,2,38], followed by Linux or FreeBSD (with BIRD or FRRouting running on top – "Other(s)" category in Fig. 4), followed by Arista, Nokia, Huawei, and Dell. Note that Mikrotik and Ubiquity also appeared in the "Other(s)" category in Fig. 4. Unfortunately, we were unable to get routers from all vendors. But we

[2] The survey was sent on both RIPE and NANOG mailing-lists on September 4$^{\text{th}}$ 2024. At the time of writing this paper, we received 50 responses.

still managed to get routers from those that are highly represented. As for the Juniper router, its results may not be entirely accurate, since we had to test a virtual image of the MX series with Containerlab [10] instead of physical hardware. Overall, routers from different vendors are expected to share the same kind of behavior with a default configuration, which therefore gives a trend and a big picture of what should be observed in the wild. Therefore, we assume our infrastructure being as representative as possible.

The sender relies on FishNet (see Sec. 3) to inject IPv6 EHs in its egress traffic. The sender generates `TCP` and `UDP` traffic with `Netcat` [40], and `ICMPv6` traffic with `ping6` [36]. Traffic is collected on the sender and the receiver with `tcpdump`.

Table 2 shows all experiments (IPv6 EH types and sizes) performed on the routers. All existing IPv6 EHs are tested, except the `Routing` Header type 1 (Nimrod) as we were not able to find its `Routing` Header format in the RFCs, probably because it is too old and deprecated. `Routing` Header type 55 is included for testing a routing header with an undefined type and see how the routers behave. Each IPv6 EH with a specific size is tested three times: with `TCP`, `UDP`, and `ICMPv6`. The MTU is increased to $8,192$ on both links to support large IPv6 EH sizes (e.g., $2,048$ bytes or more).

4.2 Results

This section presents results based on experiments performed in our controlled environment. What is observed here might not necessarily reflect the reality in the Internet, although representative enough to provide a trend. That is why we will perform measurements in the wild in Sec. 5 and Sec. 6. Table 2 shows successful experiments (with a ✓) for each router. With a default configuration running on the routers, every single IPv6 EH is successfully forwarded, whatever its size or the Layer-4 in packets. Some combinations of two or three IPv6 EHs were also tested and were successful for all routers as well (e.g., a `Hop-by-Hop Options` Header followed by a `Destination Options` Header, up to $2 \times 2,048$ bytes in total), despite not being in Table 2 for readability reasons. Such results tend to suggest that there is no hardware limit by default on those routers. For example, Ouellette [42] reports a router running a default configuration with hardware limit (i.e., with a limited parsing buffer size for the headers), which seems to drop a packet as soon as the total size of IPv6 EHs reaches something between 160 and 192 bytes. This kind of limit exists in old routers, where the parsing buffer size is quite small (usually 256 bytes max, sometimes even smaller, e.g., 64 or 128 bytes, for older routers [9]). Also, an interesting observation that is worth mentioning is about a specific test where a `Hop-by-Hop Options` Header is not in first position (e.g., a `Destination Options` Header followed by a `Hop-by-Hop Options` Header). Some routers would drop the packet, while some would not. It is due to the fact that RFC8200 [13] (Sec. 4.1) does not use normative language to enforce those requirements. As a consequence, dropping the packet or not in such a situation are both valid. However, the best approach here is probably to not drop the packet, i.e., be liberal on what is

Table 2. IPv6 EHs survivability in the controlled lab, for each tested router. Each IPv6 EH is tested with different sizes when it makes sense. All tests are performed three times: one with `UDP`, one with `TCP`, and one with `ICMPv6`. Each ✓corresponds to a successful test, i.e., when it successfully goes through the router.

IPv6 EHs	**Routers**						
	R1	R2	R3	R4	R5	R6	R7
`Hop-by-Hop Options` Header (8, 16, 32, 64, 128, 256, 512, 1024, 2048)	✓	✓	✓	✓	✓	✓	✓
`Destination Options` Header (8, 16, 32, 64, 128, 256, 512, 1024, 2048)	✓	✓	✓	✓	✓	✓	✓
`Fragment` Header *atomic* (Fixed size: 8)	✓	✓	✓	✓	✓	✓	✓
`Fragment` Header *non-atomic* (Fixed size: 8)	✓	✓	✓	✓	✓	✓	✓
`Routing` Header *Type 0* (24, 72, 136, 264, 520, 1032, 2040)	✓	✓	✓	✓	✓	✓	✓
`Routing` Header *Type 2* (Fixed size: 24)	✓	✓	✓	✓	✓	✓	✓
`Routing` Header *Type 3* (24, 72, 136, 264, 520, 1032, 2040)	✓	✓	✓	✓	✓	✓	✓
`Routing` Header *Type 4* (24, 72, 136, 264, 520, 1032, 2040)	✓	✓	✓	✓	✓	✓	✓
`Routing` Header *Unknown Type 55* (24, 72, 136, 264, 520, 1032, 2040)	✓	✓	✓	✓	✓	✓	✓
`Authentication` Header (16, 32, 64, 128, 256, 512, 1024)	✓	✓	✓	✓	✓	✓	✓
`Encapsulating Security Payload` (16, 32, 64, 128, 256, 512, 1024, 2048)	✓	✓	✓	✓	✓	✓	✓
`Mobility` Header *Type 0, no option* (Fixed size: 8)	✓	✓	✓	✓	✓	✓	✓
`Host Identity Protocol` Header *Type 1* (Fixed size: 48)	✓	✓	✓	✓	✓	✓	✓
`Shim6 Protocol` Header (Fixed size: 8)	✓	✓	✓	✓	✓	✓	✓

received and conservative on what is sent out. This, for interoperability reasons and to avoid any protocol ossification. After all, operators tend to dislike when a router drops packets that do not break normative rules.

Routers usually need to parse past the IPv6 header because of lookups. Indeed, they also need Layer-4 for, e.g., ports. When there is one or more IPv6 EHs after the IPv6 header, the upper-layer protocol header is pushed further in the packet, and old routers may not have a parsing buffer large enough for the headers. As a result, such routers would drop the packet. As a comparison, we run the same experiments in Table 2 again but, this time, a simple filter on `TCP`

Table 3. IPv6 EH types and sizes tested during a measurement campaign. Each × corresponds to an experiment, for a total of 38 experiments.

IPv6 EH		IPv6 EH Size (Bytes)														
Name	Type	∅	8	16	24	32	40	48	56	64	128	256	512	680	1,024	1368
`Destination Options` Header			×	×	×	×	×	×	×	×	×	×	×			
`Hop-by-Hop Options` Header			×									×	×			
`Fragment` Header	atomic	×														
	non-atomic	×														
`Routing` Header	2	×														
	0				×									×		×
	3				×									×		×
	4				×									×		×
	55				×									×		×
`Authentication` Header				×									×		×	
`Encapsulating Security Payload`				×									×		×	
`Mobility` Header	0	×														
`Host Identity Protocol` Header	1	×														
`Shim6 Protocol` Header		×														

(destination port 22) is added to routers configuration. This is to make sure a lookup is performed by routers. The results are exactly the same as with default configuration on routers. This tends to suggest that routers with default configuration are not responsible for dropping a packet with IPv6 EHs, except for old routers that may have hardware limitation. Should we observe drops in the wild, the main reason would therefore likely be policies applied by operators. The TX checksum offloading issue described at the end of Sec. 3.2 may also affect IPv6 EHs survivability, since this feature is often enabled by default.

The question of whether routers apply different treatment to packets when there is an IPv6 EH must be answered. In our case, some routers punt a packet to the slow path every time a `Hop-by-Hop Options` Header is present. For other IPv6 EHs, a packet stays on the fast path (when it applies). This observation can be useful, especially for the `Hop-by-Hop Options` Header. Indeed, since such a packet goes through the slow path, a router under heavy load (quite frequent for some routers in the Internet) may drop it. In that case, IPv6 EHs are not really the direct cause but the consequence is the same, i.e., the packet is dropped.

5 Internet Measurement Methodology and Data Collection

In this paper, we want to provide a comprehensive view on how IPv6 EHs are processed in the network. To double-check observations made in a controlled environment (see Sec. 4), we conduct multiple *experiments*, each one being a five-step process run by a vantage point (VP) towards others VPs in our measurement infrastructure:

Step_1: 20 pings towards the destination VP;
Step_2: 1 vanilla Paris `traceroute` [4] towards the destination VP;
Step_3: 5 Paris `traceroutes` with a given IPv6 EH towards the destination VP. An IPv6 EH is injected in Paris `traceroute` using FishNet (see Sec. 3);

Table 4. Measurement infrastructure running our experiments. 23 Cloud Providers (CPs) are considered for full mesh experiments, leading to 506 pairs combinations, for a total of 403,788 traces per experiment (798 traces × 506 VPs pairs). The "Label" column is used to easily identify each VM in this paper.

Cloud Provider	ASN	VM Location	Label
Google Cloud	396982	Belgium	BEL
Huawei Cloud	136907	Ireland (Dublin)	DUB
AlphaVPS	203380	Bulgaria (Sofia)	SOF
Vultr	20473	South Korea (Seoul)	SEO
Linode	63949	USA (Dallas)	DAL
Alibaba Cloud	37963	China (Beijing)	BEI
MPVS.net	202448	Cyprus	CYP
Contabo	141995	Japan (Tokyo)	TOK
BlackHOST	174	Austria (Vienna)	VIE
Veesp	43317	Russia (Saint Petersburg)	RUS
Hostiko	203394	Ukraine (Kyiv)	UKR
HostZealot	57814	Georgia (Tbilisi)	TBI
DigitalOcean	14061	Australia (Sydney)	SYD
OVHCloud	16276	Singapore	SGP
Misaka Network	35487	Nigeria (Lagos)	NIG
Microsoft Azure	8075	India (Pune)	IND
EdgeUno	7195	Guatemala (Guatemala City)	GUA
Atlantic.NET	6364	USA (New York City)	NYC
ZappieHost	61138	Chile (Valdivia)	CHI
Heficed	61317	Brazil (Sao Paulo)	BRA
Amazon AWS	16509	South Africa (Cape Town)	AFR
Mythic Beast	44684	UK (Cambridge)	CAM
	60011	USA (Fremont, CA)	FMT

Step_4: 1 vanilla Paris `traceroute` towards the destination VP;

Step_5: 20 pings towards the destination VP.

Step_3 is the core of an experiment and aims at testing the survivability of a given IPv6 EH through five consecutive Paris `traceroutes` between two VPs. To limit the risk of a path from being changed between consecutive traces (e.g., load balancing), all Paris `traceroute` identifiers are kept identical between traces of all experiments. Pings and classic Paris `traceroutes` before and after Step_3 are there for reachability reasons, i.e., to check whether the experiment destination is reachable, and for comparison, i.e., to detect whether the path or the RTT changes compared to traffic with IPv6 EHs. Any experiment is thus made of seven Paris `traceroutes` and 40 pings. We run each experiment around a particular upper-layer protocol, i.e., `TCP`, `UDP`, and `ICMPv6` (leading thus to 21 Paris `traceroutes` per IPv6 EH). The objective here is to see whether the upper-layer protocol has an impact on IPv6 EHs processing. Finally, data is collected at each VP through `tcpdump`.

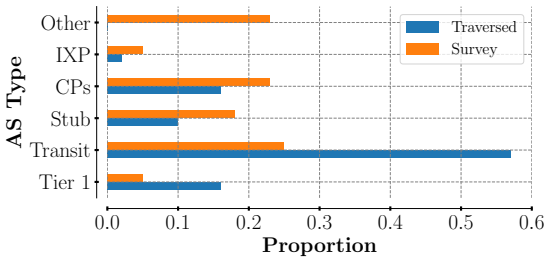


Fig. 5. Category of ASes traversed by our measurements (total is 64) and survey respondents. Important note: there is no intersection between the list of traversed ASes and the list of ASes that responded to the survey.

To obtain a comprehensive view of IPv6 EHs survivability, we conduct several *measurement campaigns*. We define a measurement campaign as a set of experiments, run in full mesh between VPs, considering all possible IPv6 EHs and possibly varying the IPv6 EH size and type when it makes sense. Table 3 lists all considered IPv6 EHs, with their varying parameters. Note that the size $\emptyset$ means that the corresponding IPv6 EH has a fixed and predefined size. Combining all parameters (i.e., those with $\times$) leads to 38 different experiments. This means that, for a pair of VPs in one direction, a measurement campaign corresponds to 798 Paris `traceroutes` (21 traces multiplied by 38 experiments). Table 3 also has all IPv6 EHs tested in Table 2, with some strategic choices on sizes. One reason is simply to reduce the measurements execution time (e.g., min, mid and max values for a `Routing` Header instead of all sizes, fewer sizes tested for a `Hop-by-Hop Options` Header since it has been reported a low survivability for small ones – see Sec. 7 –, etc), while another reason is to not exceed the MTU (e.g., sizes of $1,500$ bytes and more).

To run our measurement campaigns, we build a full mesh infrastructure around Cloud Providers (CPs). The reason for this approach is twofold: (*i*) we wanted full control on destinations in order to capture the received traffic and make sure they are configured to process all IPv6 EHs correctly, which is not possible with Internet data measurement systems such as RIPE Atlas; and (*ii*) we were interested in the core/edge point of view, without end-users. Table 4 gives the list of considered CPs and the exact location of the virtual machines (VMs) running our experiments. We made efforts in spreading our experiments over distinct CPs to avoid the particular case of inter data-center traffic, with VMs scattered around the world. The majority of the VMs are located in Europe (39.1%), followed by Asia (21.7%). 13% of VMs are located in North America, the same proportion in South/Latin America. Finally, we were able to deploy a few VMs in Africa (8.7%) and Oceania (4.3%).

We run five measurement campaigns (called *Runs* in the following) over two weeks. Each Run roughly lasts 40 hours. This allows us to finally collect $2,018,940$ traces ($5 \times 403,788$ traces per measurement campaign). Fig. 6 de-

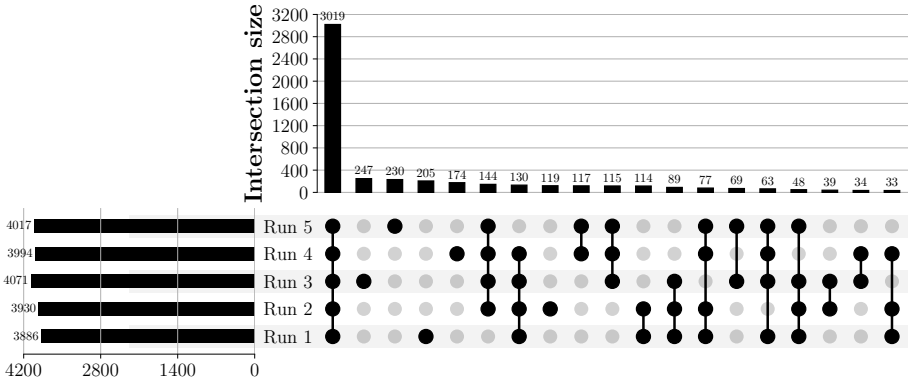


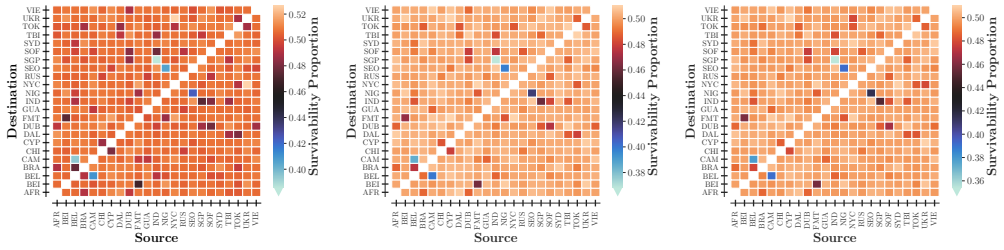
Fig. 6. UpSet plot – IPv6 addresses collected during Step_2, Step_3, and Step_4 of each Run, and the intersection between Runs.

picts an UpSet plot [11] illustrating the unique IPv6 addresses we collected during Step_2, Step_3, and Step_4 of each Run and how they intersect. It is another representation for a Venn diagram with a large number of sets (i.e., more than three sets). The figure is made up of three parts: the matrix (bottom right) shows the number of different IPv6 addresses collected in Runs. A dot in the matrix means that at least one IPv6 address has been collected during that Run (e.g., the dot, second column, third row, indicates that some IPv6 addresses were collected during Run 3). If there are multiple black dots on a column, it corresponds to IPv6 addresses collected during multiple Runs (e.g., the first column refers to IPv6 addresses seen in every Run). The histogram on the bottom left is the size of each matrix row, while the histogram on top right gives the number of IPv6 addresses in the corresponding column of the matrix (if we focus on the dot, second column, third row, 4,071 IPv6 addresses were collected during Run 3 – left histogram – with, in particular, 247 of them uniquely observed during Run 3 – top histogram). It is worth mentioning that, for readability reasons, intersection sizes lower than 30 are not shown on Fig. 6. Run 3 is the one in which we collected most IPv6 addresses (4,071) while the minimum was in Run 1 (3,886). Most of IPv6 addresses (3,019) were observed during every Run. Those addresses are mapped to 64 ASes (consistent over the five Runs – relying on ipwhois, for lookups against RIR's databases), most of them (56.25%) being Transit ASes, and 15.62% of them being Tier1 ASes (see Fig. 5).

6 Survivability in the Wild

This section describes our results for IPv6 EHs survivability in the wild, i.e., as we measure it in the Internet. As discussed in Sec. 5, survivability is assessed through Paris `traceroute` measurements (Step_3).

In Sec. 6.1, we provide a general overview of IPv6 EHs survivability, while Sec. 6.2 shows results on a per IPv6 EH basis. Then, Sec. 6.3 discusses where the drop of packets with IPv6 EHs occurs in the network. Finally, Sec. 6.4 and



(a) `ICMPv6` (b) `UDP` (c) `TCP`

Fig. 7. High level overview of IPv6 EHs survivability between CPs, according to a specific upper-layer protocol. Results have been merged between the five Runs.

Sec. 6.5 respectively discuss the impact of IPv6 EHs on path lengths (i.e., number of hops), and on the round-trip time.

6.1 General Overview

Fig. 7 provides a high level overview of IPv6 EHs survivability between CPs. In particular, it shows the proportion of survivability, whatever the IPv6 EH considered, between the full mesh of CPs. Tick labels used in Fig. 7 refer to labels in Table 4. IPv6 EHs survivability is summarized per upper-layer protocol, i.e., `ICMPv6` (Fig. 7a), `UDP` (Fig. 7b), and `TCP` (Fig. 7c). A value of 0 in the heatmap colorbar means that, somewhere on the path between two CPs, the packet with an IPv6 EH is dropped every time. Said otherwise, it never reaches the destination. On the contrary, a value of 1 means that the packet with an IPv6 EH is never dropped and always reaches the destination.

We see that IPv6 EHs with `ICMPv6` offers the best survivability (average of 50.4%), while `TCP` and `UDP` offer roughly the same survivability rate (average of 49.67% and 49.9% respectively). The lowest survivability, whatever the upper-layer protocol, is between SGP and IND (in that direction), i.e., 38.59% for `ICMPv6`, 36.84% for `UDP`, and 35.43% for `TCP`. On the contrary, the highest survivability depends on the upper-layer protocol: between UKR and NYC (in both directions – 52.63%) for `ICMPv6`, NYC to FMT (50.99%) for `UDP`, and UKR to CYP (51.01%) for `TCP`.

To summarize, half of the Paris `traceroutes` containing an IPv6 EH are dropped along the path. This result is inconsistent[3] with our results in the controlled environment (see Sec. 4), where we observed a perfect survivability rate with default configuration on routers. In the next section, we investigate

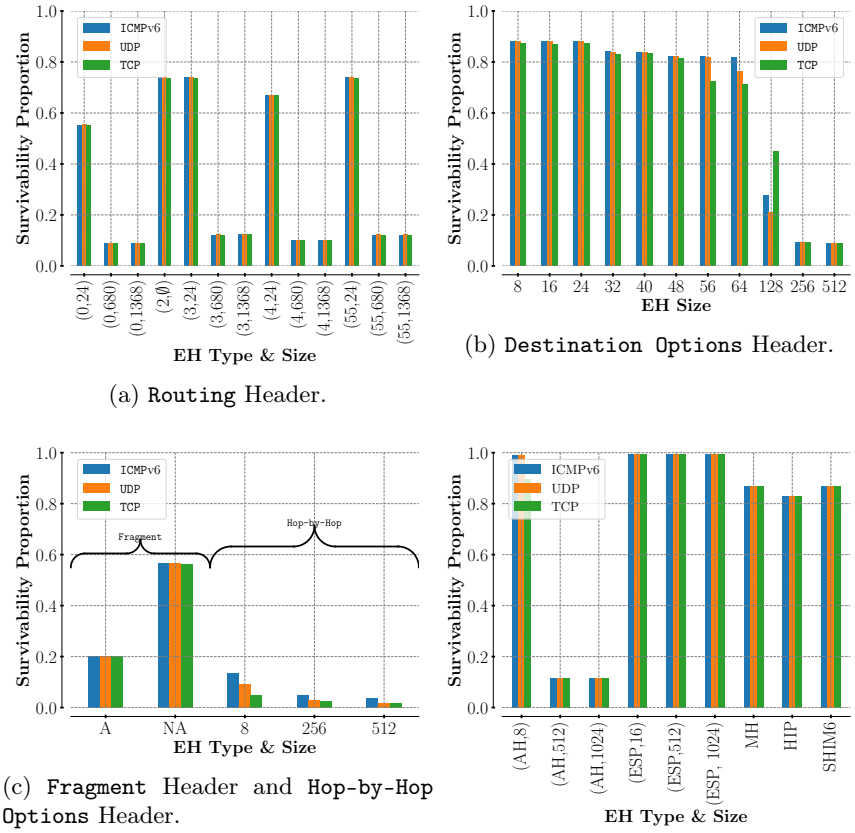
[3] An effort was made to identify [1] vendors for each encountered router. However, only 85 were identified, which only represents 2% of the total. The distribution is as follows: Juniper (57), Cisco (16), Huawei (12).

drops on a per IPv6 EH basis and we try to understand who is responsible for the drop of IPv6 EHs.

6.2 IPv6 EHs Survivability

Fig. 8 shows the survivability rate on a per IPv6 EH basis. For each plot, the results have been merged over the five Runs. It is worth mentioning that the observed survivability rates are consistent over all traces, i.e., if the drop of a packet with an IPv6 EH is observed in one trace during Step_3, then it is also observed in the four other traces. However, in a very limited number of cases (between 0.02% and 0.05%), we detected inconsistent experiments in which a packet with an IPv6 EH was dropped in a few traces, and reached the destination in others. Some incomplete vanilla traces (Step_2 and Step_4) were also detected: between 1.05% and 1.32% of two incomplete vanilla traces, and between 0.49% and 0.61% of only one (out of two) complete vanilla trace. More generally, we can say that the above does not affect our results and we believe such situations are most probably due to temporary failures.

Fig. 8a shows the survivability of the `Routing` Header according to its type (first value in the X-Axis couple) and its size (second value in the X-Axis couple – $\emptyset$ for type 2 as it has a fixed and predefined size). It is worth mentioning that a `Routing` Header mechanism is to be deployed only in limited domains, for security reasons, thus we might see a low survivability. In Fig. 8a, we see that the `Routing` Header survivability is dependent on its size, but not necessarily on its type. In particular, the best survivability (up to 0.7) is obtained with a 24-byte `Routing` Header (i.e., the smallest size), while considering larger ones leads to a low survivability (< 0.1). There is no real difference between upper-layer protocols, except for a small `Routing` Header where `UDP` provides a slightly better survivability. The fact that Type 0 suffers from more drops is not surprising since it is deprecated. Types 2 (Mobility support), 3 (RPL), 4 (Segment Routing), and 55 (undefined `Routing` Header type) receive the same treatment, except maybe for Segment Routing that suffers from a few more drops, which could be explained by the fact that it is more deployed and is treated more aggressively for security reasons. One can conclude that the `Routing` Header hits operators' policies, which is especially true for small sizes. Those policies are applied on Layer-3 and are therefore completely independent of the upper-layer protocol. As for a larger `Routing` Header, one cannot say for sure by looking at Fig. 8a alone. At first glance, it is most likely due to operators' policies as well, but there seems to be a pattern that may suggest hardware limits. Indeed, it is not impossible that some traversed ASes still have old routers deployed. However, large `Encapsulating Security Payload` sizes (see Fig. 8d) have a perfect survivability (compared to larger `Authentication` Header sizes). This hint is a clear indication that the main cause is therefore operators' policies. Overall, having a low `Routing` Header survivability in the wild is not a problem, since the `Routing` Header should be deployed in limited domains. The only condition for operators is to ensure that their hardware supports `Routing` Header deployment.



(a) `Routing` Header.

(b) `Destination Options` Header.

(c) `Fragment` Header and `Hop-by-Hop Options` Header.

(d) Remaining IPv6 EHs.

Fig. 8. Survivability on a per IPv6 EH basis. Results have been merged over the five Runs.

Fig. 8b focuses on the survivability of the `Destination Options` Header, according to its size. We expect packets with a `Destination Options` Header to have a high survivability, due to its nature (i.e., processed only by the destination). In Fig. 8b, we see that the `Destination Options` Header is slightly reliable until 64 bytes (above 0.7). On the contrary, for larger sizes, the survivability strongly drops below 0.2, except for `TCP` with a 128-byte size that offers a survivability of 0.45. Again, in the fashion of the `Routing` Header, there is no strong difference according to the upper-layer protocol for a small `Destination Options` Header, but `ICMPv6` performs slightly better than `UDP` and `TCP`. For sizes 56 and 64, `TCP` performs worse than `UDP`, but tends to decrease more slowly than `ICMPv6` and `UDP` until 128 bytes. Such a difference between upper-layer protocols rules out hardware limits as a cause. Instead, it is a clear indication

that operators' policies are the main reason, again. The difference with TCP was investigated since middleboxes were suspected to apply different treatment, but only a few MSS modifications were found and were not problematic. Overall, one may try to rely on the Destination Options Header over the global Internet, but one must pay attention to its size.

Fig. 8c shows the survivability of both a Fragment Header and a Hop-by-Hop Options Header. We focus first on the Fragment Header as an Atomic fragment ("A" on Fig. 8c – a packet that contains a Fragment Header without being actually fragmented into multiple pieces, i.e., the fragment offset is 0 and the M (more) bit is also 0) and as a Non-Atomic fragment ("NA" on Fig. 8c). We expect a low survivability for Atomic fragments [22], but a high survivability for Non-Atomic fragments. In fact, none of them are reliable, although Non-Atomic fragments survive more easily. Atomic fragments are more frequently dropped either for security reasons [19], or because stateful middleboxes drop them assuming they are unexpected whenever no previous related fragment has been seen. Considering the small size of a Fragment Header (i.e., 8 bytes), hardware limit is ruled out and the main cause is therefore operators' policies. Overall, one could hardly rely on the Fragment Header, as already observed by Jaeggli et al. [31]. Some actors seem to rely on a fixed and conservative MTU value rather than fragmentation. Note that the Fragment Header might be allowed for some specific services such as large DNS requests [27]. Due to its small size, we can say that a Fragment Header definitely hits policies on transit, which is bad for IPv6 fragmentation in general. Second, Fig. 8c shows the survivability of the Hop-by-Hop Options Header (processed by all devices along the path), according to its size. The observed Hop-by-Hop Options Header survivability is quite low (< 0.1), which makes it totally unreliable, even with the smallest 8-byte size. Therefore, the main reason is definitely operators' policies. Indeed, the Hop-by-Hop Options Header is heavily dropped by operators for security and performance reasons, whatever the size and due to the lack of use cases outside of limited domains. It is not necessarily due to the fact that it goes to the slow path under heavy load, considering the low survivability rate of even a small 8-byte Hop-by-Hop Options Header.

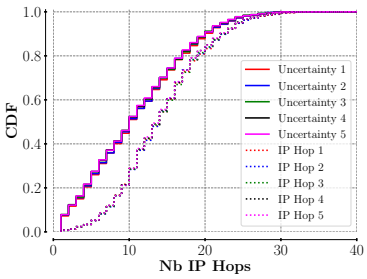
Fig. 8d shows the survivability of other IPv6 EHs. We first notice that there is no difference between upper-layer protocols. Also, both an 8-byte Authentication Header and Encapsulating Security Payload (whatever the size) offer nearly perfect survivability, which is good news for IPsec and proves that having a strong use case helps a lot. This is also another good example of how encryption can help bypass filters. However, unsurprisingly since Encapsulating Security Payload is generally preferred and more used for security reasons, larger Authentication Header sizes have very low survivability, which is a consequence of operators' policies. Finally, Mobility Header, Host Identity Protocol Header, and Shim6 Protocol Header offer quite good survivability, without any distinction between upper-layer protocols, even though Mobility Header and Host Identity Protocol Header are not standardized with an

upper-layer protocol yet. Considering their respective small sizes (i.e., 8, 48, and 8 bytes), this suggests that they hit operators' policies as well.

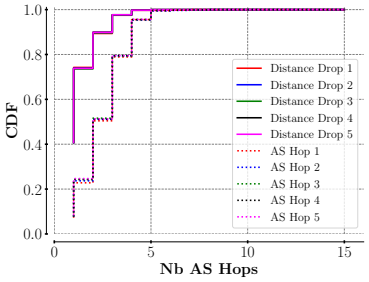
In summary, size matters when it comes to IPv6 EHs [20]. Note that IPv6 EHs modified on transit were also considered. However, 100% of the IPv6 EHs received were unmodified. Now, let us compare these observations with the responses received to our survey. First, 81.8% of operators declare they do not explicitly filter IPv6 EHs, while the other 18.2% declare they do. This is not representative of what we observed, probably because none of the respondent ASes were traversed during our measurements, and because it only represents a small set of operators. However, this could highlight some cases where old routers or bugs are the main cause of IPv6 EHs drops. A good illustration is the following comment from an operator: "*While we don't intentionally filter Extension Headers, we don't intentionally use them either and have never done any testing to confirm they are functional on our network*". Also, those who explicitly filter IPv6 EHs mainly do it on the type (100%), while filters on the size (12.5%) and data (12.5%) seem less frequent. It is not representative of what we observed either, as only the size seems to matter. Again, this can be explained by the fact that the respondent ASes only represent a small set of operators. But it could also highlight that hardware limits (i.e., old routers) may still be more present than one thinks, in addition to operators' policies. Finally, 72.7% of respondents declare they apply a filter on Layer-3 (e.g., next header, source or destination address), while 61.4% declare they apply a filter on Layer-4 (e.g., ports). While it was proved in Sec. 4 that it does not cause issues for IPv6 EHs, trying to filter on the IPv6 "Next Header" field versus the *protocol* (i.e., Layer-4 "proto", the upper-layer protocol after the IPv6 EHs chain) is not the same. Indeed, should an operator intentionally use (or not) the former and only accept, e.g., `TCP`, `UDP`, and `ICMPv6`, any IPv6 EH would therefore not be allowed. After all, it is up to operators to decide what they accept or not. IPv6 EHs are controversial and some operators would strictly forbid them, as illustrated by the following comments received to our survey: "death to extension headers", as well as "Extension Headers were a trash idea and need to be eradicated". Some others are more flexible and listen to IETF discussions, as illustrated by the following comment: "We are following the current tests and waiting for BCP in relation to EH processing". This one shows that good RFCs and BCPs can influence operators' decisions on IPv6 EHs. To conclude, some IPv6 EHs need incentives to be widely adopted, which is not the case right now.

6.3 IPv6 EHs Drop Attribution

This section tries to assign the IPv6 EHs drops to an AS. The problem is not trivial due to classic `traceroute` issues, e.g., incomplete traces. Indeed, some hops along the path may not respond to probes during a given `traceroute` but may respond to subsequent ones. This behavior is usually due to rate limiting [3] or `ICMPv6` "Time Exceeded" being lost in its way back to the source. Note that some hops never respond, probably because they are configured to not do so. We also have observed traces with load balancing effects or path modifications



(a) Uncertainty zone size distribution, with respect to `traceroute` length.



(b) Distance (in terms of AS hop) distribution of IPv6 EHs drop, with respect to `traceroute` AS path length.

Fig. 9. IPv6 EHs drop attribution.

(roughly, 8-9% of the cases, on average, for Step_3 for all experiments in a Run), despite our efforts to rely on Paris `traceroute` to maintain flows. This is discussed in Sec. 6.4 and Sec. 6.5.

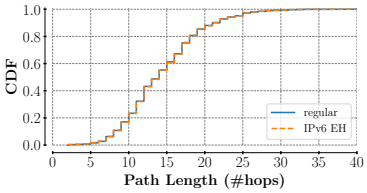
To deal with `traceroute` issues, we introduce the notion of *uncertainty zone*, i.e., a set of potentially guilty hops, starting from the IP hop after the last replying one, and ending at the penultimate IP hop (i.e., the one before the destination). From that, we derive the uncertainty zone size as the number of IP hops between those two points. Note that the destination is not part of the guilty possibilities because we have access to it and we made sure all IPv6 EHs were accepted. Obviously, the earlier a drop in a trace, the bigger the uncertainty zone, and the more difficult it will be to assign the drop responsibility to an AS. Said otherwise, the uncertainty zone size spans between 1 (the best case, as the uncertainty zone has the smallest size) and the entire path length (the worst case). Fig. 9a compares the uncertainty zone size distribution and the path length distribution (in terms of IP hops). In particular, curves labeled "IP Hop n" refer to the path length distribution of Run n, while "Uncertainty n" is the uncertainty zone size distribution for Run n. We first notice that most of the paths are between 5 and 25 hops long. It is not surprising to have longer paths (25+ hops) due to how CPs are spread all over the world. Some anomalies were also detected, i.e., traces with a loop, which explains some extreme hop counts (e.g., 35+). Over the five Runs, we have between 15% and 17% of *successful* vanilla traces (i.e., Step_2 and Step_4) with a loop, and between 8% and 9% of *successful* IPv6 EHs traces with a loop (i.e., Step_3). More importantly, we notice that most of the uncertainty zone sizes are between 2 and 20, which confirms that many traces have an uncertainty zone so large that it complicates the AS responsibility attribution.

In order to determine the AS responsible for a drop, we chose to follow this simple assumption: *an AS will generally apply ingress filtering on IPv6 EHs* [20]. This assumption and trend has been confirmed by contacting different operators. Of all the operators who responded to our survey, 62.5% declare they filter IPv6 EHs on the edges (on ingress), while 37.5% declare they filter IPv6 EHs everywhere. Note that some ASes may do it on egress as well, like some CPs that prohibit sending IPv6 EHs out. Based on the aforementioned rule and the fact that we reduce the probability of having errors by doing five IPv6 EH Paris `traceroutes` (Step_3) for a single experiment, we implemented the following algorithm: (*i*) we find the ASN for each hop in a trace and rebuild the AS path; (*ii*) we take the last responding AS in the AS path of each IPv6 EH trace; (*iii*) we keep the furthest AS based on the AS path from vanilla traces; (*iv*) we declare the next AS guilty, or the current one if it is the destination one. Fig. 9b compares the distance (in terms of AS hops) distribution of ASes responsible for dropping (curves labeled "Distance Drop n", where n refers to Run n) and the AS path length of the full trace (labeled "AS Hop n"). We first notice that most of the paths are between 1 and 5 ASes long. More importantly, we notice that, based on our algorithm, most of the drops seem to occur early in the path, i.e., within 1-3 ASes. It is worth mentioning that the IPv6 EH drop attribution algorithm was applied to 96.9% of all incomplete IPv6 EHs experiments (consistent over the five Runs). The other 3.1% represents incomplete experiments where the algorithm could not find a satisfactory solution. As a result, the drop percentage of IPv6 EHs per AS type is consistent over the five Runs as well and is as follows: Transit (46.95%), CP (36.03%), Tier1 (13.68%), Stub (3.31%), and IXP (0.03%).

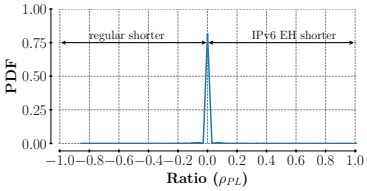
Overall, IPv6 EHs seem to have more difficulty passing through Transit and CP ASes. It is not surprising for CP ASes, i.e., an IPv6 EH is likely to hit strict policies, or a bug due to infrastructure complexity, both leading to packet drops. On the other hand, and according to their role, Transit ASes could do a better job and relax their policies regarding IPv6 EHs, or upgrade their hardware in case of limitation. Also, based on the algorithm naivety and the proximity between CP and Transit ASes in collected traces, it is possible that a part of guilty Transit ASes might be attributed to CP ASes instead. Others do a decent job, even though Tier1 ASes could do better. A good balance must be found between avoiding the ossification of the IPv6 protocol and security (i.e., filtering IPv6 EHs). It probably starts with educating people on this topic, so that operators can better evaluate the pros and cons of their policies. Depending on where someone operates, a small decision can have big consequences.

6.4 IPv6 EHs Path Length

As mentioned previously, we have observed traces with load balancing effects or path modifications (roughly 8–9% of the cases, on average), despite relying on Paris `traceroute` to maintain flows. Therefore, we want to understand if IPv6 EHs have an impact on the path length (i.e., number of hops). For each experiment, we compare the two vanilla traces with the five IPv6 EH traces. Only complete traces with no loop are kept, the others are ignored. Fig. 10a



(a) Cumulative distribution of path length (#hops) over the five Runs for both Paris `traceroute` and IPv6 EHs.



(b) Path length ratio (See Formula 1) between Paris `traceroute` and IPv6 EHs.

Fig. 10. Path length comparison between Paris `traceroute` and IPv6 EHs (data from Step_2, Step_3, and Step_4).

shows that most of complete traces are approximately between 10 and 20 hops long. There is no difference between the path length of a vanilla trace and the path length of an IPv6 EH trace, for a same experiment. It means that despite load balancers and path modifications, the number of hops remains consistent.

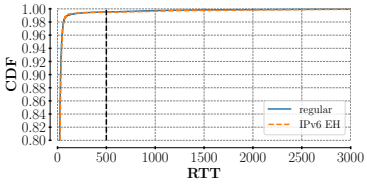
To better observe potential differences in path lengths between vanilla and IPv6 EH traces, we compute a path length ratio as follow:

$$\rho_{PL} = \frac{\#Hops_{IPv6EH} - \#Hops_{regular}}{\#Hops_{regular}}. \quad (1)$$

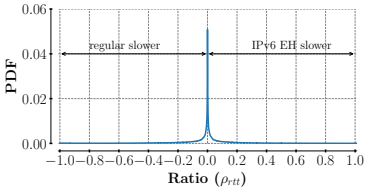
ρ_{PL} has values in [-1, 1], where negative values mean that vanilla traces are shorter than IPv6. Positive values, on the contrary, mean that IPv6 EH traces are shorter. Obviously, a null value means that both traces have the same number of hops. Fig. 10b plots ρ_{PL} for the five Runs merged. It shows that the ρ_{PL} distribution is quite symmetric and centered on 0, which confirms that both traces have generally the same length. Overall, IPv6 EHs do not have an impact on path length (i.e., IPv6 EHs do not arbitrarily expand or shorten paths).

6.5 IPv6 EHs Round-Trip Time

This section aims at understanding if IPv6 EHs have an impact on the round-trip time (RTT). For each experiment, we compare the two vanilla traces with the five IPv6 EH traces. Only complete vanilla traces with no loop are kept, as well as any IPv6 EH traces with no loop, while the others are ignored. Fig. 11a shows that most of the RTTs are approximately between 0 and 150 milliseconds. There is no difference between the RTTs of a vanilla trace and the RTTs of an IPv6 EH trace, for a same experiment. It means that routers do not spend additional time to process IPv6 EHs, even those in the slow path (e.g., the `Hop-by-Hop Options` Header), which would tend to suggest that most routers ignore (i.e., do not process) IPv6 EHs.



(a) Cumulative distribution of RTTs over the five Runs for both Paris `traceroute` and IPv6 EHs.



(b) RTT ratio (ρ_{RTT} – See Formula 2) between Paris `traceroute` and IPv6 EHs.

Fig. 11. Round-trip time comparison between Paris `traceroute` and IPv6 EHs (data from Step_1, Step_3, and Step_5).

In the fashion of path length, for better understanding the RTT potential differences between vanilla and IPv6 EH traces, we compute an RTT ratio as follow:

$$\rho_{RTT} = \frac{RTT_{IPv6EH} - RTT_{regular}}{RTT_{regular}}. \tag{2}$$

ρ_{RTT} has values in [-1, 1], where negative values mean that vanilla traces are slower than IPv6. Positive values, on the contrary, mean that IPv6 EH traces are slower. Obviously, a null value means that both traces follow roughly the same RTT. Fig. 11b plots ρ_{RTT} for the five Runs merge. In the fashion of ρ_{PL}, the distribution is symmetrical and centered on 0. Overall, for our datasets, IPv6 EHs do not have an impact on RTT.

7 Related Work

RFC7045 [6] provides guidelines on how IPv6 EHs should be transmitted, also with a focus on middleboxes influence on the traffic.

The seminal work by Gontt et al. [21] observes how an 8-byte `Hop-by-Hop Options` Header, an 8-byte `Destination Options` Header, and a `Fragment` Header survive over the Internet, which comes from the IETF and dates back in 2015. Gontt et al. perform `traceroute` measurements towards servers belonging to the Alexa top 1M domains and find that such IPv6 EHs are often dropped in transit networks. Since then, efforts have been made to measure the adoption of emerging standards around IPv6 EHs and the way they are processed within the network, as listed below.

Hendrikx et al. [24] state that dropping all traffic containing any IPv6 EH is the de facto rule applied by operators, for security reasons. To support their claim, they perform limited measurement campaign on a national research network (CSNET) and a campus network (UTNET). In the same spirit, Padurean et al. [47] run large-scale `traceroute` measurements to find the presence of Seg-

ment Routing [17,18] with IPv6 as forwarding plane. They reported no presence of such a deployment, probably due to IPv6 EHs filtering.

Elkins et al. [15] have proposed a methodology for isolating the reasons and network devices responsible for IPv6 EHs drops. In particular, they discuss a situation in which a tested server is behind a Content Delivery Network (CDN). However, they do not perform any measurements. Elkins et al. [14] also focus on the Performance and Diagnostic (PDM) `Destination Options` Header Option [16]. Such an IPv6 EH option provides sequence numbers and timing information as a basis for measurement. They do not report any drop when measurements are performed between hosting services while they observe some drops when measurements are sent towards Alexa top 1M domains.

Huston and Damas [28] report an improvement, over the years, in processing the IPv6 `Fragment` Header. They also notice that `Destination Options` Header and `Hop-by-Hop Options` Header IPv6 EHs are generally not supported on public Internet infrastructure.

Custura et al. [12] present an extensive measurement campaign with a focus on access and server edge networks, and provide results indicating the traversal across Internet paths of packets that include either a `Hop-by-Hop Options` Header or a `Destination Options` Header. Their results indicate that successful reception across an IPv6 path can currently depend on the type of included IPv6 EHs, its size, and on the transport protocol used.

JAMES [35, 49] tests a large set of IPv6 EHs in full mesh through multiple vantage points, with some of them located in distinct Autonomous Systems. By definition, both source and destination of measurements are controlled. Among others, JAMES reports that path traversal diminishes as the size of IPv6 EHs increases.

Finally, Ouelette [42] tested the forwarding of the `Hop-by-Hop Options` Header and `Destination Options` Header on six different routers. Ouelette noticed that one router failed in forwarding packets in certain scenarios (i.e., packets with IPv6 EHs of 256 and 512 bytes), which is likely due to hardware limitation.

This paper goes further by testing all IPv6 EHs currently defined, i.e., the `Hop-by-Hop Options` Header, the `Destination Options` Header, the `Fragment` Header, the `Routing` Header, the `Authentication` Header, the `Encapsulating Security Payload`, the `Mobility` Header, the `Host Identity Protocol` Header, and the `Shim6 Protocol` Header. Each of them is tested with different sizes when it applies. All vantage points are unique, i.e., hosted by different Cloud Providers and running in distinct locations, therefore evaluating the edge and core of the Internet. This paper also provides observations in a controlled environment, comparing the behavior of routers from different vendors when processing IPv6 EHs, and investigates additional features such as RTTs and path lengths.

8 Conclusion

This paper offers an extensive vision of IPv6 EHs survivability, i.e., the capacity of an IPv6 packet carrying an Extension Header to traverse the Internet

and arrive unmodified at its destination. To study this survivability, we first setup a controlled environment, tested multiple scenarios with default router configuration, and injected IPv6 EHs traffic with a specially crafted eBPF program, FishNet. We showed that, in such a context, the IPv6 EHs survivability is perfect. To confront those results with real world, we also performed measurements in the wild, relying on FishNet, in a full mesh of Cloud Providers virtual machines. On the contrary to controlled environment, our results show that IPv6 EHs survivability is quite limited (around 50% on average) in the wild. Depending on the type of IPv6 EH, survivability might be a good point (e.g., `Destination Options` Header), while it is not expected for others (e.g., `Routing` Header). We also showed that IPv6 EHs drops are caused by some ASes, generally quite close to the packet source, due to policies. In terms of path lengths and round-trip time, we did not observe any particular differences between regular IPv6 and IPv6 EHs traffic. Measurement software and collected data are provided to the research community.

Ethical Considerations

For our Internet measurement campaigns, we implemented RFC9511 [48] for attribution of Internet probes. In particular, we implemented both in-band and out-of-band recommendations, except for `TCP` probes where only out-of-band was used to avoid potential packet drops due to the presence of data with a `TCP SYN`. For in-band, an email address was added to the data payload. For out-of-band, a web server (both HTTP/HTTPS) was running, with the main page being an alias of `/.well-known/probing.txt`. That text file described the on-going measurement campaign. Until now, it is important to note that no one has contacted us.

Source Code

The source code of FishNet is available at `https://github.com/iurmanj/ebpf-ipv6-exthdr-injection`. The dataset with all our measurements is available at `https://shorturl.at/MHwKQ`.

Acknowledgments

This work has been supported by the CyberExcellence project, funded by the Walloon Region, under number 2110186.

References

1. Albakour, T., Gasser, O., Beverly, R., Smaragdakis, G.: Third time's not a charm: Exploiting SNMPv3 for router fingerprinting. In: Proc. ACM Internet Measurement Conference (IMC) (November 2021)

2. Albakour, T., Gasser, O., Beverly, R., Smaragdakis, G.: Illuminating router vendor diversity within providers and along network paths geolocation. In: Proc. ACM Internet Measurement Conference (IMC) (October 2023)
3. Alvarez, P., Oprea, F., Rula, J.: Rate-limiting of IPv6 traceroute is widespread: Measurements and mitigations (July 2017), https://shorturl.at/RyY4l, [Last Accessed: May 22nd, 2024]
4. Augustin, B., Cuvellier, X., Orgogozo, B., Viger, F., Friedman, T., Latapy, M., Magnien, C., Teixeira, R.: Avoiding traceroute anomalies with Paris traceroute. In: Proc. ACM Internet Measurement Conference (IMC) (October 2006)
5. Brockners, F., Bhandari, S., Mizrahi, T.: Data fileds for in-situ operations, administration, and maitenance(IOAM). RFC 9197, Internet Engineering Task Force (May 2022)
6. Carpenter, B., Jiang, S.: Transmission and processing of IPv6 extension headers. RFC 7045, Internet Engineering Task Force (December 2013)
7. Castineyra, I., Chiappa, N., Steenstrup, M.: The Nimrod routing architecture. RFC 1992, Internet Engineering Task Force (August 1996)
8. Cisco: TRex: Realistic traffic generator, https://trex-tgn.cisco.com, [Last Accessed: June 7th, 2024]
9. Cisco: IPv6 extension headers review and considerations (October 2006), https://www.cisco.com/en/US/technologies/tk648/tk872/technologies_white_paper0900aecd8054d37d.html, [Last Accessed: May 4th, 2024]
10. Containerlab: Containerlab, https://containerlab.dev/, [Last Accessed: October 7th, 2024]
11. Conway, J.R., Lex, A., Gehlenborg, N.: UpSetR: an R package for the visualization of intersecting sets and their properties. Bioinformatics **33**(18) (September 2017)
12. Custura, A., Secchi, R., Boswell, E., Fairhurst, G.: Is it possible to extend IPv6? Computer Communications **214**, 90–99 (January 2024)
13. Deering, S., Hinden, R.: Internet protocol, version 6 (ipv6) specification. RFC 8200, Internet Engineering Task Force (July 2017)
14. Elkins, N., Ackermann, M., Deshpande, A.: IPv6 extension headers (performance and diagnostic metics (PDM) destination option) testing across the Internet (July 2022), https://shorturl.at/GrC2a, [Last Accessed: April, 19th 2024]
15. Elkins, N., Ackermann, M., Dhody, D.: Deep dive into IPv6 extension header testing. Internet Draft (Work in Progress) draft-elkins-v6ops-eh-deepdive-fw-01, Internet Engineering Task Force (October 2022)
16. Elkins, N., Hamilton, R., Ackermann, M.: IPv6 performance and diagnostic metrics (PDM) destination option. RFC 8250, Internet Engineering Task Force (September 2017)
17. Filsfils, C., Dukes, D., Previdi, S., Leddy, J., Matsushima, S., Voyer, D.: Ipv6 segment routing header (srh). RFC 8754, Internet Engineering Task Force (March 2020)
18. Filsfils, C., Previdi, S., Grinsberg, L., Decraene, B., Likowski, S., Shakir, R.: Segment routing architecture. RFC 8402, Internet Engineering Task Force (July 2018)
19. Gont, F.: Processing of IPv6 atomic fragments. RFC 6946, Internet Engineering Task Force (May 2013)
20. Gont, F., Hilliard, N., Doering, G., Kumari, W., Huston, G., Liu, W.: Operational implications of IPv6 packets with extension headers. RFC 9098, Internet Engineering Task Force (September 2021)
21. Gont, F., Linkova, J., Chown, T., Liu, W.: Observations on the dropping of packets with ipv6 extension headers in the real world. RFC 7872, Internet Engineering Task Force (June 2016)

22. Gont, F., Liu, W., Anderson, T.: Generation of IPv6 Atomic Fragments Considered Harmful. RFC 8021, Internet Engineering Task Force (January 2017)
23. Google: IPv6 statistics (2008–2024), https://www.google.com/intl/en/ipv6/statistics.html, [Last Accessed: May, 21st 2024]
24. Hendrikx, L., Velan, P., Schmidts, R., De Boer, P.T., Pras, A.: Threats and surprises behind IPv6 extension headers. In: Proc. IFIP Network Traffic Measurement and Analysis (TMA) (June 2017)
25. Hui, J., Vasseur, J.P., Culler, D., Manral, V.: An IPv6 routing header for soruce routes with the routing protocol for low-power and lossy networks (RPL). RFC 6554, Internet Engineering Task Force (March 2012)
26. Huston, G.: IPv4 address report (2013–2024), https://ipv4.potaroo.net, [Last Accessed: May, 20th 2024]
27. Huston, G.: Dealing with IPv6 fragmentation in the DNS. https://blog.apnic.net/2017/08/22/dealing-ipv6-fragmentation-dns/ (August 2017), [Last Accessed: April 25th, 2024]
28. Huston, G., Damas, J.: IPv6 fragmentation and EH behaviours (March 2022), https://www.potaroo.net/presentations/2022-03-20-iepg-v6frag.pdf, [Last Accessed: April, 17th 2024]
29. IANA – Internet Assigned Numbers Authority: Internet protocol version 6 (IPv6) parameters - IPv6 extension header types. Tech. rep., Internet Assigned Numbers Authority (2024)
30. iproute2: Introduction to iproute2, see https://tldp.org/HOWTO/Adv-Routing-HOWTO/lartc.iproute2.html
31. Jaeggli, J., Colitti, L., Kumari, W., Vyncke, E., Kaeo, M., Taylor, T.: Why operators filter framgents and what it implies. Internet Draft (Work in Progress) draft-taylor-v6ops-fragdrop-02, Internet Engineering Task Force (December 2013)
32. Johnson, D., Perkins, C., Arkko, J.: Mobility support in IPv6. RFC 3775, Internet Engineering Task Force (June 2004)
33. Kent, S.: IP authentication header. RFC 4302, Internet Engineering Task Force (December 2005)
34. Kent, S., Atkinson, R.: IP encapsulating security payload (ESP). RFC 2406, Internet Engineering Task Force (November 1998)
35. Léas, R., Iurman, J., Vyncke, E., Donnet, B.: Measuring IPv6 extension headers survivability with james. In: Proc. ACM Internet Measurement Conference (IMC), Poster Session (October 2022)
36. Linux: ping6(8) - Linux man page, https://linux.die.net/man/8/ping6, [Last Accessed: May 21th, 2024]
37. LWN.net: net, sched: add clsact qdisc (January 2016), https://lwn.net/Articles/671458/, [Last Accessed: June 1st, 2024]
38. Marechal, E., Donnet, B.: Network fingerprinting: Routers under attack. In: Proc. International Workshop on Traffic Measurements for Cybersecurity (WTMC) (September 2020)
39. Moskowitz, R., Nikander, P., Jokela, P., Henderson, T.: Host identity protocol. RFC 5201, Internet Engineering Task Force (April 2008)
40. Netcat project: The GNU Netcat project, https://netcat.sourceforge.net, [Last Accessed: May 21th, 2024]
41. Nordmark, E., Bagnulo, M.: Shim6: Level 3 multihoming shim protocol for IPv6. RFC 55533201, Internet Engineering Task Force (June 2009)
42. Ouellette, K.: IPv6 Hop-by-hop and Destination Options Forwarding In Routers. Internet Draft (Work in Progress) draft-ouellette-v6ops-eh-router-forwarding-00, Internet Engineering Task Force (March 2024)

43. Scapy Community: Scapy, https://scapy.net, [Last Accessed: May 7th, 2024]
44. tc: tc(8) – linux manual page, see https://man7.org/linux/man-pages/man8/tc.8.html
45. The Linux Foundation: eBPF (October 2021), https://ebpf.io
46. Tuxology: An entertaining eBPF XDP adventure (May 2017), https://suchakra.files.wordpress.com/2017/05/cls-xdp1.png?w=696, [Last Accessed: May 7th, 2024]
47. V.-A Padurean, Gasser, O., Bush, R., Feldmann, A.: SRv6: Is there anybordy out there? In: Proc. International Workshop on Traffic Measurements for Cybersecurity (WTMC) (June 2022)
48. Vyncke, E., Donnet, B., Iurman, J.: Attribution of Internet Probes. RFC 9511, Internet Engineering Task Force (November 2023)
49. Vyncke, E., Léas, R., Iurman, J.: Just another measurement of extension header survivability (JAMES). Internet Draft (Work in Progress) draft-vyncke-v6ops-james-02, Internet Engineering Task Force (July 2022)