Tracing Internet Path Transparency

Author(s):
Kühlewind, Mirja; Walter, Michael; Learmonth, Iain R.; Trammell, Brian

Publication Date:
2018

Permanent Link:
https://doi.org/10.3929/ethz-b-000315299 →

Originally published in:
http://doi.org/10.23919/TMA.2018.8506532 →

Tracing Internet Path Transparency

Mirja Kühlewind
ETH Zurich

Michael Walter
ETH Zurich

Iain R. Learmonth
Univiversity of Aberdeen

Brian Trammell
ETH Zurich

***Abstract*—Investigating Internet Path Transparency means measuring if a network path between two endhosts is impaired by in-network functions on the path. A path is considered transparent if it provides connectivity and the same performance independent of the protocol or protocol stack that is used for the transmission. Unfortunately this is not always the case. Simple firewalls that block e.g. UDP, are an example. Of course such in-network functions are often valuable, like firewalls. However, these middleboxes also, sometimes unintentionally, make assumptions about the traffic passing through them that restricts innovation in the Internet on the higher layers, e.g. the deployment of new UDP-based protocols such as QUIC, to stick with the previous example.**

***PATHspider* is an active measurement tool to test for Path Transparency. In this paper we present a new feature of *PATHspider* that integrates tracebox-based functionality and analysis to not only detect in-transparency but also further locate the origin of the impairment observed. As an example study we show updated and extended measurements on ECN support and connectivity. By using our enhanced ECN *PATHspider* plugin to test network support of the ECN IP codepoint and additional path tracing that is correlated with DSCP testing, we show that most in-network ECN IP codepoint zeroing is due to use of the deprecated definition of the IP ToS field for domain-internal service differentiation, while pure resetting of the ECN IP field is more likely an active inference in border networks.**

I. Introduction

The end-to-end principle [1], on which the basic design of the Internet is based, argues for smart endpoints connected by a simple network: more complex functions should be placed at higher layers whenever possible. Following this principle keeps the upper layers of the Internet protocol stack flexible. Implicitly this principle suggests that connectivity and network treatment should not depend on functions provided by higher layer protocols; i.e., the network should be *transparent* to these higher layer protocols.

This principle is often quoted but increasingly rarely followed. Today, there are a diverse and increasing set of functions deployed in the Internet that go far beyond forwarding. These functions have been introduced for various reasons, such as network address conservation, ensure stable management and operations, supporting troubleshooting, or increasing security. However, independent of their utility or perceived value, these middlebox functions impair the connectivity and treatment of end-to-end traffic, whether intentionally, (e.g., in case of a firewall), or accidentally, as a side effect of desired function.

As a consequence we need to consider impairments introduced by on-path network elements in protocol design and protocol-path-pair selection. Newer protocol extensions such as Multipath TCP [2] are expressly designed to deal with these effects, incorporating path probing and fallback mechanisms that disable optional features if problems have been detected during connection establishment. Emerging approaches in building endpoint transport stacks can even race entire protocols against each other on different paths [3] to work around these impairments

However, as a first step we actually need to know what impairments to expect. Any of the approaches above add complexity and impose cost in form of latency and efficiency. Therefore it is beneficial to cover those cases that are most likely to be observed on the Internet.

PATHspider [4] is an active measurement tool that is designed to address exactly the question whether a certain path on the Internet is transparent to a given protocol or protocol feature under test. It has been developed and extended over the last couple of years and been used in a diverse set of measurements [5], [6].

This collected data provides a view on which impairments exist, and how likely certain impairments are to be expected under certain network conditions. However, *PATHspider* treats the network as a black box, performing A/B testing of connectivity and operation with and without the tested feature. An approach that doesn't give much insight into why and where these impairments are observed. Better understanding the root cause of the observed behavior can help to locate problems and potentially fix them, as well as assist the design of solutions that will not only work around the problem but can also help to support initial intended middlebox functionality correctly.

In this paper we present the integration of route tracing and impairment localization into *PATHspider* based upon the tracebox [7] methodology. Tracebox is a measurement tool that performs a traceroute, comparing the headers returned by each node with the headers originally sent. Through the integration with *PATHspider* we can use information derived from A/B testing to find paths on which a given feature doesn't work, and immediately trace the path on which the feature failed. In contrast to a two-phase approach where one would take the results from *PATHspider* and feeding them into a Tracebox implementation after the full *PATHspider* measurement has been completed, the integrated approach has less delay between the initial detection and the follow-up traceroute measurement and provides therefore more timely and likely more accurate path information. Further, the integrated approach can more easily use a TCP packet with the same characteristics of that which caused the measured failure.

Further, we describe the results of an initial study performed

with a prototype version of the integrated tracing feature, which also extends our previous work on measuring connectivity and support of TCP's Explicit Congestion Notification (ECN) [8] extension [9], [10]. ECN, if negotiated successfully during the TCP handshake, allows endpoints to mark their packets as ECN Capable Transport, using one of two codepoint in the two bit ECN IP header field (ECT0 or ECT1). Subsequently, network devices can signal congestion to the endpoints before packets need to be dropped by setting the Congestion Experienced (CE) codepoint in the ECN IP field. While ECN support is continuously increasing and any severe connectivity problems that hindered initial deployment are overcome, we show that the usability of ECN is still impaired by middlebox mangling.

For this measurement we extend the *PATHspider* ECN-plugin to also test the use of different ECN IP codepoint during the TCP handshake. These cases are important to verify the functionality of an extension to ECN [11] currently proposed within the IETF to allow ECN support for TCP control packets. In addition, we use the trace extension to correlate impairments of the Differentiated Services Code Point (DSCP) [12] with impairments to the ECN codepoints in the IP header. This help us to identify a root cause of our observed problems as the majority of ECN IP mangling along the paths is caused by out-dated middlebox implementations, which treat the byte containing the ECN and DSCP codepoints according its previous definition as the IP Type of Service byte [13]; this definition has been deprecated for two decades. We further found that if the ECN IP field is reset likely independent of the DSCP field, that these bleaching is close to the network border, presumably on purpose by content networks. In case of Google it is known that ECN is not supported by their web servers and as such we could observe active re-setting to Not-ECT in their domain. It should be noted that this case study, we present, does not include measurement from mobile vantage point, where ECN is often not supported due to TCP proxying. These follow-up measurements are currently in progress.

A. Related work

PATHspider's development follows on a body of work on the deployability of TCP extensions, such as Honda et al's [14] options study, or Bauer et al's initial re-examination of ECN readiness [15]. TCP HICCUPS [16] went even further by proposing a build in mechanism in the TCP handshake to detect middlebox mangling. More recent work also focuses on deployability of encryption mechanisms, mainly TLS [17]. *PATHspider* 's HTTP/2 plugin, not further covered in this work, can be used to test HTTPS. Further, there have been a few recent studies that also focus on UDP, e.g. on ECN support [18], complementing our earlier ECN measurement, or our own measurement on UDP differential treatment [19] which was performed together with some of the authors of the tracebox tool [20]. Further, DSCP modification in the IP header was recently examined [21], however, only providing initial results from a small-scale measurement study.

Further, the measurement community has also paid increasing attention to mobile networks [22], as it is known that in-network functions are heavily used in these networks due to their architecture. *PATHspider* is currently under deployment on the MONROE testbed[1], a European-wide measurement platform mostly based on nodes connected to multiple mobile access networks. While we presented initial measurement results recently [6], the public release of a MONROE container for *PATHspider* measurement is work in progress.

II. Integrating tracebox functionality into *PATHspider*

PATHspider [4] is a general-purpose A/B testing tool for Internet path transparency. Each test run by *PATHspider* typically consists of multiple connections performed in quick succession: a control connection that is usually vanilla HTTP over TCP or DNS request using UDP, to verify connectivity; and the experimental connection(s) using the protocol or protocol extension under test. If the experimental flow fails, this indicates protocol-dependent connectivity problems. *PATHspider* also passively observes and analyzes all test traffic and can thus attempt to infer from this any unexpected behaviors or modifications. The current release of *PATHspider*[2] provides support for testing ECN [8], DSCP [12], TCP Fast Open (TFO) [23], HTTP/2 [24], and the Evil Bit [25]; this last test is a test to detect generalized connectivity breakage when setting reserved header fields. As a result *PATHspider* provides **observations** on path transparency. An observation is an assertion that a given **condition** was observed on a given path at a given time; e.g. that ECN was successfully negotiated, or that an experimental TFO cookie was seen.

PATHspider's architecture is designed to be scalable to measure millions of paths in a single campaign. As such it can detect protocol-dependent connectivity problems as well as middlebox mangling, however, further measurements are needed to localize where problem occurs. The tracebox [7] tool, used by the *PATHspider* authors in previous analysis [9], provided inspiration for an automated traceroute function that has been integrated into the *PATHspider* workflow. Instead of running traceroutes manually after a large-scale *PATHspider* campaign, *PATHspider* can now be used to automatically perform traceroutes in close timely succession to the original measurement that detected the problem if a configured failure condition has been observed. Similar as the tracebox tool, *PATHspider* will not only provide the trace information but also an analysis of any mangling or dropping of packets by middleboxes.

Where the end-to-end-only *PATHspider* measurements would infer path conditions from the replies received from targets, the new integrated traceroute functionality relies on routers on the path providing ICMP quotations attached to ICMP Time-To-Live (TTL) exceeded messages. These hop-by-hop measurements produce a finer view of the path and allow for the more precise localisation of any breakage.

[1] see https://www.monroe-project.eu/

[2] https://pathspider.net

When an IPv4 router receives an IPv4 packet whose TTL is going to expire, it returns an ICMPv4 TTL exceeded message that contains a quotation of the original packet. According to [26], the returned ICMP packet should quote the IP header of the original packet and the first 64 bits of the payload of this packet. When the packet contains a TCP segment, these first 64 bits correspond to the source and destination ports and the sequence number. Later, [27] recommended to quote the entire IP packet in the returned ICMP, but this recommendation may not be followed on older routers. For IPv6, the mechanism is similar except that it is an ICMPv6 packet that is returned and the specification for ICMPv6 messages has always been that as much of the original packet should be quoted as possible without exceeding the MTU [28]. By comparing the returned quoted packet to the original packet sent, *PATHspider* is able to detect various header field modifications performed by middleboxes and routers on the path towards a target.

The tradeoff for these more precise measurements is in the scalability as running a traceroute measurement takes significantly longer than performing a single connection. For this reason traceroutes are only performed when breakage has been identified by the traditional end-to-end connection tests. Further, the base end-to-end measurement can already be used to estimate the number of hops between the measurement vantage point and the target by analysing the TTL field as received on reply packets in order to reduce the sending of redundant traceroute probes. Limiting the tracing to only the expected number of hops further speeds up the measurement.

When a traceroute is performed, the output from the measurement additionally includes more specific information on middlebox mangling observed as well as the contents of relevant packet headers as seen by each router that produced a TTL exceeded reply with a quotation where changes compared to the originally sent bits are already highlighted. In later analysis it will be possible to use this data to determine any kind of packet mangling in addition to connectivity issues.

A. PATHspider 2.0 Architecture

An overview of the architecture of the 1.0 series of *PATHspider* is available in [4]. For the 2.0 series of *PATHspider*, the architecture was extended to allow for arbitrary numbers of tests to be performed as opposed to a single A/B test. The new extended architecture is shown in figure 1. This was done through the addition of an extra *Combiner* stage before the output stage that would collect the results from individual tests and then only combine them with the job records to produce path conditions once all the tests had completed for a target. Job records are passed to the combiner stage independent of the results of the workers to reduce memory consumption, as each job record can contain metadata that can assist during analysis but should not be duplicated for each test connection.

In this new architecture, parts of the code can be more easily integrated as the API available to plugins becomes more flexible. Traceroutes can now be implemented as additional connections and the merging of traceroute results can be handled by the combiner. Existing plugins for passive observation

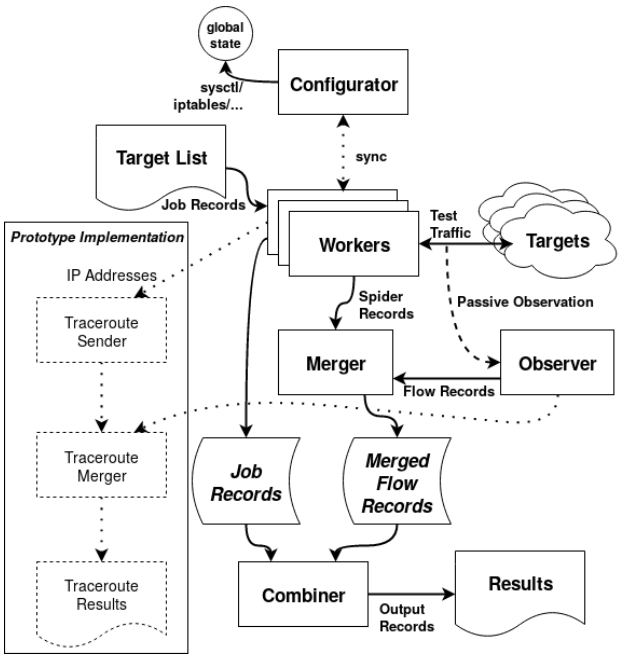

Fig. 1. The extended architecture of the 2.0 series of *PATHspider* supporting arbitrary numbers of tests for each target. The prototype implementation's modules are shown in this diagram with dashed outlines.

can be easily extended to act on the ICMP quotations without the need for this to happen in the merging module.

B. Traceroute Integration

Traceroute integration was developed in parallel with a number of other improvements to *PATHspider*. For this reason the implementation used to perform measurements in this paper is considered a prototype that will inform a final implementation. The prototype source code is available on GitHub[3].

Two new modules were created, a traceroute sender and a traceroute merger. When a measurement for a target has returned a result that warrants further investigation, a traceroute for this target is performed by passing the target details to the traceroute sender. This will use a forged packet specific to the kind of measurement to attempt to make broken behaviour observable. For example, for ECN it may be the case that a middlebox on the path clears the ECN Capable Transport (ECT) field in the IP header and so the traceroute would be performed with this field set to "all ones", Congestion Experienced (CE), and it will be observable when the field is cleared.

If a flow contains ICMP TTL exceeded packets, it is marked for analysis by the traceroute merger module. This module will prepare the traceroute data for output.

The traceroute implementation within *PATHspider* can also be invoked as a standalone tool, i.e., to accept a list of targets as if they had already been found to have interesting conditions associated with them. We use this functionality to

[3]https://github.com/mami-project/pathspider/tree/traceroute-prototype

TABLE I
ECN SUMMARY STATISTICS FOR POPULAR WEB SERVERS SINCE 2012

August 2012 [9] n =77 854		September 2014 [10] n =598 739		June 2016 [4] n =642 345		January 2017 [4] n =765 820		**November 2017** n =751 064		Description
hosts	pct	hosts	pct	hosts	pct	hosts	pct	hosts	pct	
-	-	-	-	11,858	1.85%	16,204	2.12%	30,445	4.05%	Completely failed to connect
22,948	29.5%	337,881	56.4%	452,806	70.5%	581,588	75.9%	589,821	78.5%	Capable of negotiating ECN
-	-	8,106	1.35%	2,076	0.32%	1,874	0.24%	1,608	0.21%	Failed to connect w/ECN

examine the correlation between ECN and DSCP manipulation in section III-C.

C. Enhanced ECN plugin

With the ability to perform more than just the A/B connectivity breakage test, we also performed tests for ECN negotiation with the ECT field deliberately corrupted. For each target we established 4 TCP connections where one does not attempt ECN negotiation and the remaining 3 attempt ECN neogtiation with the IP ECT field clear, set to zero, set to ECT0, and finally set to CE.

The ECN base specification [8] does explicitly exempt TCP control packets, such as the TCP SYN, to indicate ECN support, mainly because the specified ECN feedback scheme for TCP does not implement a way to provide feedback if congestion is observed for these packet. However, with the development of a more accurate ECN feedback scheme [29], a more universal support for ECN including the important-to-not-lose control packets is currently under discussion in standardization [11].

[8] does, however, not specify how to react to the reception of ECT-marked control packet. Given there are known cases where the ToS field was incorrectly rewritten and not bleached at the network border such that all packets crossing certain networks would appears as ECT or CE marked, there are ECN implementations that do not negotiate ECN if the TCP SYN has a non-zero ECN IP field. Other implementations only disable ECN negotiation if the CE codepoint is set in the SYN. While re-writing to ECT could conceal previous CE marks and thereby delay a congestion reaction until packet drops are observed, an erroneous middlebox that marks all packets with CE would indicate permanent congestion leading to sending rate reduction to basically zero at the sender side. In our measurements we show that both implementation are widely deployed which can make universal deployment of ECN more challenging than it already is.

III. DETECTING MIDDLEBOX MANAGLING

For all measurements we created a target list of IP addressees based on the Alexa top million website domains list, obtained on Oct 30, 2017. We resolved each domain name to zero or more IP addresses, taking all IPv4 and IPv6 addresses returned by the resolver and adding them to the target list. After filtering out duplicates, we end up with a target list of 751,064 IP addresses. All measurements were conducted in November 2017 and were made from servers of the cloud infrastructure provider Digital Ocean located in Amsterdam and London.

A. Update on ECN connectivity and support

First we provide a quick update on ECN support and connectivity as measured by the ECN *PATHspider* plugin with respect to measurements we reported earlier [9], [10], [6], as also summarized in Table **??**tab:ecn). Out of the 751,064 target IP address, we could not connect to 30,445 at all. Respectively, for 716,419 (95.39%) addresses we did not observe any connectivity problems, including 92,008 IPv6 addresses. For the remaining hosts we either could not connect when ECN was requested (1608 addresses) or we could connect with ECN negotiation but not in the base TCP test without ECN negotiation attempt (2,592 IP address). As both numbers are in the same range, we assumed that this indicates mainly transient failures. Therefore, we repeated the measurement with the same target list in the subsequent two weeks and found for 66 hosts with TCP SYN packets that had the ECN TCP header flags set to negotiate endpoint ECN support were stably dropped while connectivity without ECN negotiation attempts always succeeded. We subsequently utilized the integrated tracebox functionality to detect that 7 of these hosts drop the packets on reception, and one host sends a TCP RST. For the remaining hosts we were not able to acquire trace information due to unresponsive node on the path. This is however expected, as we have seen in previous measurement that there is a correlation between nodes that perform some kind of packet mangling and unresponsiveness to ICMP.

For ECN support we can report that the observed increase from previous measurement reports is continuing with 500,585 (80.17%) IPv4 addresses and 89,236 (96.99%) IPv6 address negotiating ECN successfully in November 2017. However, we also found 1,858 (0.26%) addresses where the host would reflect the TCP ECN header bits which indicates a broken TCP connection or potentially some proxy in the middle.

B. ECN IP codepoint mangling

Current specifications do not allow TCP control packets, like the SYN, to be ECN enabled, as no feedback is usually provided for this kind of packet, however, there are mechanisms under discussion to enable a fully ECN-supported Internet where all traffic is ECN capable [11]. To test deployment issues with these changes, we repeated the measurements with the extended version of the ECN plugin, performing additional test runs with the ECT0 and CE codepoints set in the IP header of the TCP SYN. For this measurement we limited our target list to the top 50,000 entries. Similar to the first study above, we were able to connect successfully to 95.59%

TABLE II
DSCP AND ECN MANIPULATION PER PATH (N = 201,854)

DSCP treatment	ECN → ECT0 (preserved)		ECN → Non-ECT (rewritten)		**total**	
	n	pct	n	pct	n	pct
→ EF (unchanged)	41850	20.7%	169	0.08%	42019	20.8%
→ 6 (three-bit bleach)	87182	43.2%	101	0.05%	87283	43.2%
→ 0 (bleach)	50031	24.8%	1665	0.82%	51686	25.6%
→ CSx	4883	2.42%	701	0.35%	5584	2.77%
→ AFxx/VA	9951	4.93%	68	0.03%	10019	4.96%
→ undefined value	5182	2.57%	81	0.04%	5263	2.61%
total	199079	98.6%	2775	1.37%	201854	100%

of the target host, independent of the codepoint set, indicating that this sample set is representative.

69.35% of all hosts (33145) only negotiated ECN when the ECN IP codepoint was set to zeros (non-ECT) but not if ETC0 or CE was set. Only 12.79% of the hosts (8280) negotiated ECN no matter what codepoint was set on the TCP SYN. 26 hosts negotiated ECN when ECT0 was set but not when CE was set. While it was expected that some hosts would not negotiate ECN if CE was set, as this is a known fallback mechanism to prevent ECN usage on corrupted path, it is rather surprising how many hosts have implemented this logic and that most host apply the same fallback to ETC0.

However, for the 12.79% of the hosts, that negotiated ECN independent of the codepoint set, we further wanted to test if the codepoint was actually observed at the endhost or potentially erased on the path. Therefore, we performed another measurement study where we used the full-traceroute mode of *PATHspider*from our Vantage point in Amsterdam to a reduced set of the initial target list. For fast convergence, we limited the list of target addresses to IP addresses that have a unique prefix of length 24 for IPv4 or length 64 for IPv6, as other addresses with the same prefix lie in the same collision domain and will thus assumably anyway share the same path, leading to a reduced set of 223,367 unique targets. For this measurement we used TCP SYN packets with ECT0 set but no ECN negotiation attempt (ECE and CWR not set in the TCP header) to avoid unwanted interference. We further set the DSCP codepoint to 46 (Expedited Forwarding) to look at correlation between ECN and DSCP mangling on path, as described in section III-C below.

201,854 (90.37%) hosts gave back a TCP response to the TCP SYN. 263 of these hosts responded with a RST. For these hosts we also tested the response with the DSCP set to zero; this always yielded the same result. For 2,775 (1.37%) of these hosts, the ECT0 codepoint was erased before the TCP SYN received the server, with about 50% of the codepoint removal in the last hop and in more than 90% of the cases in the last 40% of the path. For these hosts we ran an additional measurement with ECT0 set but no SYN flag to test if that was a general misconfiguration or an intended normalization on the TCP SYN and found that only about approximately 2/3 of the last hops still had the ECT0 erased. We also found that 3,252 hosts reflected the ECT0 codepoint in the SYN/ACK even though ECN support was not requested in the SYN, while 22 replied with ECT1 and 18 with CE in the SYN/ACK. Both the erasure of the ECT1 codepoint as well as unexpected codepoints on the SYN/ACK indicate potential deployment of routers that still operate on the ToS field, as further investigated in the following section.

C. Correlation between ECN IP and DSCP manipulation

To better understand potentially root causes of the ECN IP mangling, we now take a closer look at the correlation between ECN IP codepoint and DSCP manipulation along the path.

1) Manipulation per-path: The results per path on the collection of the 201,854 responsive paths we probed are shown in Table II. Here we compare the value of the ECN and DSCP codepoints at the ingress to the last hop before the destination[4]. DSCP manipulation along the path is far more common than ECN manipulation: four of five paths see some DSCP manipulation, while only about one in 75 paths see the ECN codepoints manipulated.

Of note is that the most common single outcome is that ECN is preserved, while DSCP is rewritten to 6. This is indicative of an older interpretation of the bits of the DSCP field, blanking the old IP Precedence bits and leaving the low-order three bits unchanged ("three-bit bleaching"). The next most common occurrence is a DSCP rewrite to 0 ("bleaching") while leaving ECN unchanged; followed by no change of either codepoint along the path. Fortunately both cases do not impact the operation of ECN directly.

Examining only those paths on which the IP ECN codepoint is manipulated, we see a different pattern. The majority of ECN-manipulating paths also bleach the DSCP codepoint. The second most common occurrence is more interesting: here, the ECN codepoint is set to 0 while the DSCP codepoint is set to a CS value. This is also consistent with the treatment of this byte according to the old ToS definition [13]: a device sets a CS value as if it were the ToS byte and zeroes the ECN codepoint as a side-effect.

For the remaining 11.45% we observed in total 51 different code points of the possible 64, indicating that the field was remarked for internal use but not bleached at the network border. Also different than observed with the ECN IP bits, about 70% of all DSCP manipulations happened in the first

[4]as we are using TCP traceroute, we cannot see the values that left the last hop toward the server, as the server will send us a TCP SYN ACK (or a TCP RST) instead of an ICMP Time Exceeded.

TABLE III
DSCP AND ECN MANIPULATION PER EDGE (N = 28,961)

DSCP treatment	ECN → ECT0 (preserved)		ECN → Non-ECT (rewritten)		**total**	
	n	pct	n	pct	n	pct
→ EF	55	0.19%	136	0.47%	191	0.66%
→ 6 (three-bit bleach)	3891	13.4%	88	0.30%	3979	13.7%
→ 0 (bleach)	17469	60.3%	1367	4.72%	18836	65.0%
→ CSx	1179	4.07%	359	1.24%	1538	5.31%
→ AFxx/VA	1770	6.11%	79	0.27%	1849	6.38%
→ undefined value	2496	8.62%	72	0.25%	2568	8.87%
total	26860	92.7%	2101	7.25%	28961	100%

half of the path. As a side note, we similarly observed in the TCP response of the 201,854 hosts 37 of the 64 different codepoints, unsurprising with 89% set to 0.

In summary, there is a high number of DSCP rewriting and bleaching, as expected, and on a path basis, the majority of on-path interference with the ECN codepoint in the Internet appears to be linked to DSCP interference by devices implementing the older interpretation of this byte in the IP header. We now turn to per-edge analysis to verify this hypothesis.

2) Manipulation per-edge and per-node: We next take all the paths from our traceroute measurements and combine them into a single graph with 418,834 distinct edges between any two hops on the measured paths. Of these, 389,873 (93.1%) change neither the DSCP nor the ECN codepoint, 26,965 (6.44%) change the DSCP codepoint but not the ECN codepoint, 1,059 (2.5%) change both, and 937 (2.2%) change only the ECN codepoint. However, we note that the most common ECN changes are associated with zeroing the entire byte containing both codepoints, and on 561 of these 937 edges, the incoming DSCP codepoint is already zero. We can therefore state that the majority – 53.1% of ECN manipulation can be observed on the same edge as the DSCP manipulation with an upper bound for 81.2% when including those cases where DSCP has already been previously bleached and as such no additional observation about ToS bleaching could be made when the ECN IP codepoint was erased.

The outgoing codepoints per edge on the collection of 28,961 edges which change at least one of the DSCP and ECN codepoints are shown in Table III. We note that the most common per-path behavior is three-bit bleaching without ECN manipulation, while the most common per-edge behavior is full DSCP bleaching without ECN manipulation.

To determine the location of the nodes responsible for this manipulation, we first assume that the source node of each edge is responsible for the manipulation; i.e., we assume that DSCP and ECN will not be modified on a packet before the TTL is checked. Grouping our edges by egress node, we find 10,139 distinct nodes. 1,226 of these manipulate ECN, and 9,573 manipulate DSCP.

We see 16 ASNs with a high proportion of ECN interference (i.e. more than half of nodes in our traces that manipulate any codepoint manipulate ECN) representing 14 entities. Five of these are hosting firms, eight are Internet and infrastructure service providers (six of these in the APNIC region), and one is Google. The ISPs see proportionally more DSCP remarking, suggesting that ECN damage in ISPs is more likely due to collateral damage from older DSCP treatment, while the hosting providers and Google are more likely to change ECN markings without DSCP re-marking.

IV. CONCLUSION

In this paper we presented an integration of tracebox functionality into *PATHspider*, an active measurement tool to test Internet path transparency. Our approach of full integration of path tracing and trace analysis in the *PATHspider* architecture makes it possible to automatically trigger path tracing in timely succession when a failure condition has be observed in the base end-to-end path transparency test. This enhances the *PATHspider* measurement tool to not only detect middlebox impairments but also provides further information where the impairment is located and thereby indications about the root cause of the impairment.

We show the applicability of this methodology based on a large-scale measurement campaign using the enhanced ECN plugin of *PATHspider* to also test support of the ECN IP codepoint field as well as the new full-traceroute mode to demonstrate correlation with DSCP rewriting and bleaching. With this measurement we continue our longitudinal study of ECN deployment dating back to 2012, showing a continuous increase 78.5% popular web servers negotiating ECN, with very little remaining connectivity impairment.

However, where connectivity breakage was still observed, we note that it is often linked to ICMP breakage. This can be an indication of middleboxes that are designed or configured to evade ping and traceroute. Moreover, we also observed some remaining challenges for universal ECN deployment, where all packets including TCP control packets are envisioned to be ECN enabled, due to accidental or indented ECN IP codepoint reset. Especially concerning is that 50-80% of ECN IP zeroing is connected to DSCP bleaching, indicating large deployments of outdated router functionality that operates based on the deprecated ToS field. Further, while ToS bleaching was observed on the whole network path, as this function is usually performed on domain borders, active ECN IP rewriting is more commonly performed at edge networks, e.g. when the connected network is known to not support ECN.

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 688421, and was supported by the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number 15.0268. The opinions expressed and arguments employed reflect only the authors' views. The European Commission is not responsible for any use that may be made of that information. Further, the opinions expressed and arguments employed herein do not necessarily reflect the official views of the Swiss Government.

REFERENCES

[1] Saltzer, J.H., Reed, D.P., Clark, D.D.: End-to-end arguments in system design. ACM Trans. Comput. Syst. **2**(4) (1984) 277–288

[2] Ford, A., Raiciu, C., Handley, M., Bonaventure, O.: TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824 (Experimental) (2013)

[3] Pauly, T., Trammell, B., Brunstrom, A., Fairhurst, G., Perkins, C., Tiesel, P., Wood, C.: An architecture for transport services. Internet-Draft draft-ietf-taps-arch-00, IETF (2018)

[4] Learmonth, I., Trammell, B., Kühlewind, M., Fairhurst, G.: PATHspider: A tool for active measurement of path transparency. In: First ACM/IRTF Applied Networking Research Workshop, Berlin, Germany (2016)

[5] Trammell, B., Khlewind, M., Vaere, P.D., Learmonth, I.R., Fairhurst, G.: Tracking transport-layer evolution with pathspider. In: Proceedings of the ACM, IRTF & ISOC Applied Network Research Workshop (ANRW'17). (2017)

[6] Learmonth, I.R., Lutu, A., Fairhurst, G., Ros, D., zg Alay: Path transparency measurements from the mobile edge with pathspider. In: IEEE/IFIP Workshop on Mobile Network Measurement (MNM17). (2017)

[7] Detal, G., Hesmans, B., Bonaventure, O., Vanaubel, Y., Donnet, B.: Revealing Middlebox Interference with Tracebox. In: Proceedings of the 2013 Conference on Internet Measurement Conference. IMC '13, Barcelona, Spain, ACM (2013) 1–8

[8] Ramakrishnan, K., Floyd, S., Black, D.: The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168 (Proposed Standard) (2001) Updated by RFCs 4301, 6040.

[9] Kühlewind, M., Neuner, S., Trammell, B.: On the state of ECN and TCP options on the Internet. In: Passive and Active Measurement Conference, Hong Kong, China (2013) 135–144

[10] Trammell, B., Khlewind, M., Boppart, D., Learmonth, I., Fairhurst, G., R., R.S.: Enabling internet-wide deployment of explicit congestion notification. In: Passive and Active Measurement (PAM). (2015)

[11] Bagnulo, M., Briscoe, B.: ECN++: Adding Explicit Congestion Notification (ECN) to TCP Control Packets. Internet-Draft draft-ietf-tcpm-generalized-ecn-02, IETF (2017)

[12] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W.: An Architecture for Differentiated Services. RFC 2475 (Informational) (1998) Updated by RFC 3260.

[13] Almquist, P.: Type of Service in the Internet Protocol Suite. RFC 1349 (Proposed Standard) (1992) Obsoleted by RFC 2474.

[14] Honda, M., Nishida, Y., Raiciu, C., Greenhalgh, A., Handley, M., Tokuda, H.: Is It Still Possible to Extend TCP? In: Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference. IMC '11, Berlin, Germany, ACM (2011) 181–194

[15] Bauer, S., Beverly, R., Berger, A.: Measuring the state of ecn readiness in servers, clients,and routers. In: Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference. IMC '11, Berlin, Germany, ACM (2011)

[16] Craven, R., Beverly, R., Allman, M.: A middlebox-cooperative tcp for a non end-to-end internet. SIGCOMM Comput. Commun. Rev. **44**(4) (2014) 151–162

[17] Mandalari, A.M., Bagnulo, M., Lutu, A.: Informing protocol design through crowdsourcing: the case of pervasive encryption. In: ACM SIGCOMM Workshop on Crowdsourcing and crowdsharing of Big (Internet) Data (C2B(I)D). (2015)

[18] McQuistin, S., Perkins, C.S.: Is explicit congestion notification usable with udp? In: Proceedings of the 2015 ACM Conference on Internet Measurement Conference. IMC '15, New York, NY, USA, ACM (2015) 63–69

[19] Edeline, K., Khlewind, M., Trammell, B., Donnet, B.: copycat: Testing differential treatment of new transport protocols in the wild. In: Proceedings of the ACM, IRTF & ISOC Applied Network Research Workshop (ANRW'17). (2017)

[20] Detal, G., Hesmans, B., Bonaventure, O., Vanaubel, Y., Donnet, B.: Revealing middlebox interference with tracebox. In: Proceedings of the 2013 Conference on Internet Measurement Conference. IMC '13, New York, NY, USA, ACM (2013) 1–8

[21] Barik, R., Welzl, M., Elmokashfi, A.: How to say that you are special: Can we use bits in the IPv4 header? In: First ACM/IRTF Applied Networking Research Workshop, Berlin, Germany (2016)

[22] Wang, Z., Qian, Z., Xu, Q., Mao, Z., Zhang, M.: An untold story of middleboxes in cellular networks. SIGCOMM Comput. Commun. Rev. **41**(4) (2011) 374–385

[23] Cheng, Y., Chu, J., Radhakrishnan, S., Jain, A.: TCP Fast Open. RFC 7413 (Experimental) (2014)

[24] Belshe, M., Peon, R., Thomson, M.: Hypertext Transfer Protocol Version 2 (HTTP/2). RFC 7540 (Proposed Standard) (2015)

[25] Bellovin, S.: The Security Flag in the IPv4 Header. RFC 3514 (Informational) (2003)

[26] Postel, J.: Internet Control Message Protocol. RFC 792 (Internet Standard) (1981) Updated by RFCs 950, 4884, 6633, 6918.

[27] Baker, F.: Requirements for IP Version 4 Routers. RFC 1812 (Proposed Standard) (1995) Updated by RFCs 2644, 6633.

[28] Conta, A., Deering, S.: Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. RFC 2463 (Draft Standard) (1998) Obsoleted by RFC 4443.

[29] Briscoe, B., Kuehelwind, M., Scheffenegger, R.: More accurate ecn feedback in tcp. Internet-Draft draft-ietf-tcpm-accurate-ecn-05, IETF (2017)