This project has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 688421.

Measurement and Architecture for a Middleboxed Internet

H2020-ICT-688421

Final Report on Measurement and Deployment

Author(s):	ETH	Brian Trammell (ed.), Mirja Kühlewind
	ULg	Benoit Donnet, Korian Edeline
	UNIABDN	Gorry Fairhurst, Ana Custura, Tom Jones, Iain Learmonth
	ZHAW	Stephan Neuhaus
	SRL / TID	Andra Lutu

Document Number:	D1.3
Internal Reviewer:	David Ros
Due Date of Delivery:	31 December 2018
Actual Date of Delivery:	21 December 2018
Dissemination Level:	Public

Disclaimer

The information, documentation and figures available in this deliverable are written by the MAMI consortium partners under EC co-financing (project H2020-ICT-688421) and does not necessarily reflect the view of the European Commission.

The information in this document is provided "as is", and no guarantee or warranty is given that the information is fit for any particular purpose. The user uses the information at its sole risk and liability.

Contents

European Commission | Horizon 2020 European Union funding for Research & Innovation

Executive Summary

This document contains the final report from the MAMI project on its measurement efforts, covering work since previous deliverables D1.1 and D1.2. During this period, we continued direct measurements of Internet path transparency, as well as auxiliary measurements to support other transport evolution goals of the project. In general, these measurements confirm “common knowledge” in the protocol engineering community about the nature and prevalence of manipulation of optional protocol features (i.e., there is a lot of it), but show ways forward for transport evolution in some cases: e.g., ECN is often manipulated as a side effect of outdated DSCP manipulation, but driving ECN deployment by client-side defaults is safe, since there is negligable and easily recoverable connectivity risk associated with ECN negotiation.

1 Introduction

This document is the final report on the work of the MAMI project in WP1. It covers work over all tasks since the last report: on measurement studies, tool development, and tool maintenance since D1.1 (December 2016), and updates to the PTO during maintenance since D1.2 (June 2018).

Section 2 covers path transparency measurement studies done in this timeframe, while section 3 covers measurement studies related to other transport evolution goals of the MAMI project. Many of these studies also appear in peer-reviewed publications produced by the project; in these cases, the text in this document is extracted from the papers, and references to the full paper are given at the top of the section.

Our findings during this period include:

- Most Internet paths are transparent to ECN marking. The majority of on-path interference we found with the ECN codepoint in the Internet appears to be linked to DSCP interference by devices implementing the 1980s-era interpretation of the TOS byte in the IP header.
- DSCP manipulation is widespread in the Internet. Much of the manipulation other than bleaching is also consistent with the treatment of this codepoint as if it were the old TOS byte.
- More than half of the mobile carriers we tested bleach (clear) the ECN field at the first upstream IP hop.
- In general, where ECN works in the Internet, the proposed ECN++, allowing marking of control packets, works as well.
- MSS clamping is prevalent in the Internet. Web servers artificially lower their MSS while client networks clamp MSS to prevent PMTUD failures.
- The Internet is not transparent to UDP Options, there are widespread issues with checksum calculation and UDP.
- The ubiquitous practice of home roaming in mobile networks – tunneling back mobile roaming users' traffic to egress to the Internet from their home carrier's network border, as opposed to the roaming carrier's network – has widespread, mostly negative consequences for Internet performance while roaming.
- Internet RTT is not useful for geolocation in the general case, though bufferbloated Internet access links that do not block ICMP may leave their users open to remote telemetry of network load.

Section 4 describes work we've done on our tools since December 2016 (for PATHspider) and June 2018 (for the PTO), respectively.

2 Path Transparency Measurement Studies

In this section, we detail final work on Internet path transparency measurements for four protocol features: continued work on ECN and DSCP, as well as work on maximum segment size (MSS) and the TCP MSS option, and a new facility for adding transport-layer options to UDP.

2.1 Differentiated Services Codepoint Manipulation

Note: this section is condensed from two publications by UNIABDN [13] [14]; refer to the papers for details.

Several measurements were performed to better understand how DSCP marks are modified in different types of networks. These measurements can be split in several campaigns, by the type of network and tools used.

2.1.1 DSCP in the Internet

PATHspider was used to survey DSCP-related connectivity issues for all 63 codepoint values towards 100k websites taken from the Alexa topsites list, from eight vantage points. This measurement campaign found no connectivity-dependent failures for DSCP, showing it is safe for applications to enable it. The results are presented in Table 1.

Furthermore, we developed PathTrace, a tool to generate packets with increasing TTLs to perform traceroute-style measurements and analyze the ICMP quotations received from the paths to 500 web servers from eight vantage points in the Digital Ocean Data centre. The ECN field of probe packets was set to 11 to infer correlations between ECN and DSCP modification.

Table 1: Number of connections and percentage success/failure measured by PATHspider.

	# Conn. attempts	%
Both succeeded	12M	99.99
Both failed	5,430	0.01
Baseline succeeded	3,430	0.01
Test succeeded	6,430	0.01

Table 2: Percentage for IPv4 DSCP modification pathologies per vantage point, dataset B.2

AMS	BLR	FRA	LON	NYC	SFO	TOR	%
65.3	80	66.3	83.3	77.4	92.1	67.6	Transparent
1.8	5.4	2.6	6.2	6.3	2.5	7	Reset DSCP
24.6	2.6	16.1	3.6	10.1	0.7	15.4	Reset ToS prec.
3.75	3.8	4.8	0.8	2.7	0.3	1.4	Reset ToS prec. CS6/CS7
1.8	3.0	6.1	3.1	0.4	2.2	2.8	DSCP remark
2.1	0.4	3.4	2.2	2.7	1.4	5.6	DSCP multiple remark
320	420	229	354	444	827	71	Total routers

The campaign used both UDP and TCP traffic, but found no differentiation with regards to

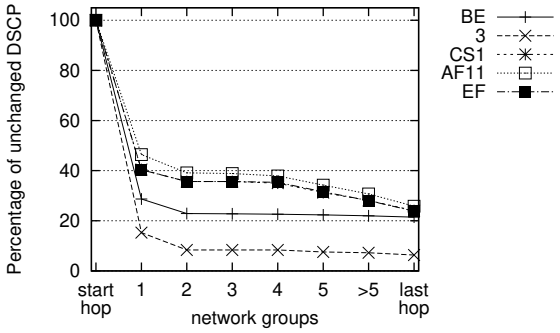


Figure 1: Re-marking on mobile edge networks)

DSCPs between the transports used. DNS resolution was performed from a different location to all vantage points to avoid using pre-assigned mapping, and/or content caching and distribution mechanisms for popular servers (CDNs). Fetching web content is a common service, and it is possible that operators could have specific DiffServ policies in place. The study identified a series of pathologies that exist irrespective of vantage point, as shown in Table 2.

The meaning of each pathology is as follows:

- *DSCP bleaching*: This pathology resets the DSCP field to zero. This remaps all flows to the default traffic class.
- *ToS bleaching*: This pathology resets to zero the upper 3 bits of the DSCP field (the former ToS precedence field), leaving other bits unchanged. This behaviour is distinctive of non-DiffServ aware routers.
- *ToS bleaching except CS6/CS7*: This pathology is a variant of ToS bleaching, where the ToS precedence field is reset only if the ToS is not 110 or 111. These markings correspond to the CS6 (Network Control) and CS7 (Internetwork Control) codepoints, which identify critical Internet traffic. This also is an indication of a non-Diff erv aware router.
- *DSCP remarking and multiple remarking*: This pathology resets the DSCP field to a specific codepoint or to a pool of few codepoints. This may be a result of DiffServ traffic conditioning.

The most important finding of this campaign was that almost one-fourth of IPv4 routers exhibited some pathological behaviour in DSCP remarking, with a significant number of routers implementing ToS bleaching, evidence of routers functioning on the deprecated ToS semantics.

2.1.2 DSCP in Mobile Networks

PathTrace was also used to collect data for 100 web servers from the Alexa top websites list from 107 mobile vantage points within the European MONROE platform, for a total of 9202 different source-destination pairs. A range of DSCP values were sent using both UDP and TCP. The measurements were completed between September 2016 and January 2017.

Table 3: Summary results for the codepoints seen at the last observed hop of the path for the EF codepoint, TCP versus UDP

TCP $n = 581$		**UDP** $n = 581$		**DSCP Observed**
hosts	pct	hosts	pct	
223	38.38%	225	38.73%	BE
281	48.36%	278	47.85%	EF
46	7.92%	49	8.43%	6
14	2.41%	14	2.41%	CS1
7	1.20%	7	1.20%	41
10	1.72%	8	1.38%	Others

Figure 1 shows that most DSCP values are remarked on entry to the mobile network. The percentage of unchanged codepoints falls below 40% immediately after the first network group.

The study also observed remarking at the edge of the network, before packets leave the mobile domain. In most cases, DSCP values were bleached (i.e. reset to zero) when packets left the mobile domain. The two-stage remarking is typical of mobile networks. Table 3 presents a summary for the codepoints seen at the last observed hop on the path.

2.2 Explicit Congestion Notification Support

We have continued to investigate the deployability of ECN, focusing on three aspects in particular: localizing the modification of ECN IP codepoints along paths, the deployability of the ECN++ enhancement to ECN, and support for ECN in the Tor anonymity network.

2.2.1 Localizing ECN Impairment

Note: this section is condensed from Tracing Internet Path Transparency [23], published by ETH and UNIABDN at the 2018 Traffic Measurement and Analysis conference. Refer to the paper for details.

We integrated route tracing and impairment localization into PATHspider based upon the tracebox [15] methodology, in an prototype extension to PATHspider. 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 taking the results from PATHspider and feeding them into a Tracebox implementation, this approach has less delay between the initial detection and the follow-up traceroute measurement, and can easily use a TCP packet with the same characteristics of that which caused the measured failure.

2.2.1.1 Correlating DSCP and ECN Mangling

We use this prototype extension to look for correlation in impairments of the Differentiated Services Codepoint (DSCP) [11] with impairments to the ECN codepoints in the IP header.

Table 4: 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%

This helps 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 to its previous definition as the IP Type of Service byte [5]; this definition has been deprecated for two decades. We further found that if the ECN IP field is reset independent of the DSCP field, then this bleaching is close to the network border, presumably done 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 measurements from mobile vantage points, where ECN is often not supported due to TCP proxying.

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. Here, we send traffic with the Expedited Forwarding (EF) codepoint set, on the assumption that since this represents the ”best” quality of service, it would be the most likely to be manipulated; however, we see no significant evidence that manipulation depends on the codepoint set.

2.2.1.2 Manipulation Per-path

The results per path on the collection of the 201,854 responsive paths we probed are shown in Table 4. Here we compare the value of the ECN and DSCP codepoints at the ingress to the last hop before the destination[1]. 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.

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

[1] 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 5: 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%

DSCP codepoint is set to a CS value. This is also consistent with the treatment of this byte according to the old TOS definition: 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 half of the path. As a side note, we similarly observed in the TCP response of the 201854 hosts 37 of the 64 different codepoints, unsurprisingly with 89% set to 0.

In summary, there is a high number of DCSP 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.

2.2.1.3 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, 1059 (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 5. 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. 1226 of these manipulate ECN, and 9573 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 re-marking, 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.

2.2.2 ECN++

Note: this section is condensed from a paper by SRL [30] published in IEEE Communications Magazine; refer to the paper for details.

A new performance enhancement called ECN++[2] proposes safe ways to remove all the original prohibitions on using ECN on each type of TCP packet. As with any new protocol, ECN++ can experience deployment problems, either because existing networks and servers protect themselves against out-of-the-ordinary behavior, or because optimizations have been built around a narrow and unchanging interpretation of the way protocols work. In particular, DoS attacks were (erroneously) considered more likely with ECN on SYN. Thus, some security devices might block it, impacting the deployment of the ++ enhancement for ECN.

We measured how much existing networks and servers would mangle or block the ECN++ updates to TCP/IP. We tested ECN and ECN++ support in both fixed and mobile networks. In particular, we used both the MONROE platform (mobile carriers) and PlanetLab (wired service providers) to test 18 mobile carriers and collect information for a total of 26 million end-to-end communications, testing 6.5 million different paths. This is, to the best of our knowledge, the largest measurement study of ECN in mobile networks so far (even Apple had only tested three mobile carriers). Although we only set out to measure ECN++ support, our measurements from mobile vantage points challenge the accepted belief that path traversal of ECN itself is free of problems.

2.2.2.1 TCP SYN and TCP SYN/ACK

To test support for ECN++ in the SYN and the SYN/ACK, we design the experiments to exchange packets containing the values used by ECN++ in the ECN field of the IP header and in the TCP ECN flags. We design two types of experiments, namely, client-side experiments and client/server experiments. In the client-side experiments, we only control the client that sends the TCP SYN packets against existing servers in the Internet (Alexa top 100k servers). This allows us to learn information about a large number of paths and about support for ECN++ in both network elements and servers.

While client-side measurements provide a lot of information about support for ECN++ in the SYN packet, it usually provides little information about support for ECN++ in the SYN/ACK packet, since the SYN/ACK packet is generated by a server that is out of our control and may

[2] https://tools.ietf.org/html/draft-bagnulo-tcpm-generalized-ecn-04

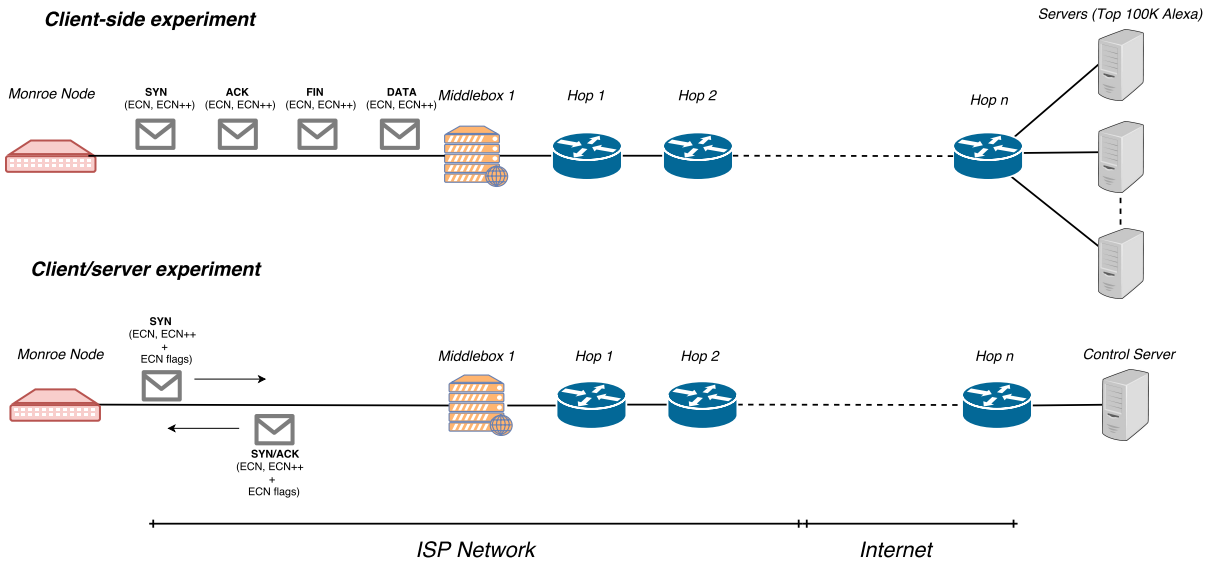


Figure 2: ECN++ support: experimental setup.

not support ECN++.

In the client/server experiments we control both the clients and servers of the connection, but not proxies. This allows us to perform exhaustive testing of all possible combinations of Accurate ECN (AccECN)[3] and ECN++ fields both in the SYN and the SYN/ACK. However, these experiments are limited to a few servers that we control, which also constrains the number of paths traversed.

Figure 2 summarizes the operational setup for the client-side and the client/server experiments.

2.2.2.2 Client-side Experiments

To observe how the ECN field in the IP header is treated by network elements we use Tracebox [15]. Tracebox uses the same principle as traceroute (i.e., sends packets with increasing TTL and receives an ICMP TTL exceeded error message from the router that discards the original packet when the TTL reaches zero). Tracebox uses information about the original packet returned in the ICMP error message to identify any changes in the IP header.

We run Tracebox sending TCP SYN packets with different codepoints in the ECN field of the IP header. We executed these tests with all possible combinations of the CWR and ECE flags. With this test, we check whether the ECN field is modified, and if so at which hop along the path it is modified.

While Tracebox is very powerful for seeing how the fields in the IP header are treated, it cannot detect the changes in the ECN flags in the TCP header. This limitation stems from the fact that most routers implement RFC792 [35] which requires them to return only the first 64 bits of the IP payload of the packet (leaving out all the ECN related flags later in the TCP header) while a few routers implement RFC1812 [8] which requires them to return the full packet if possible. Because of this, we implement a test that directly sends SYN packets from the clients

[3] https://tools.ietf.org/html/draft-ietf-tcpm-accurate-ecn-05

we control to the Alexa top 100k servers with the different values for the ECN codepoints and TCP flags used by ECN and ECN++.

This test enables us to check, through the reception of the SYN/ACK, if the SYN was delivered to the server and the server processed it. We can further identify how many servers use RFC3168 [36] (Classic ECN) and how many servers use RFC5562 [24] (ECN+).

2.2.2.3 Client/Server Experiments

For client/server experiments, we implement both a client and a server side of the test that exchange every allowed ECN++ packet sequence. We also test for the case where ECN is not used. We test for the possible SYN-SYN/ACK packet sequences involving the different codepoint/flag combinations. We measure whether the fields sent arrive at the other end, to check for middlebox interference (e.g. proxies).

2.2.2.4 Data packets, Pure ACKs and FINs

We designed these experiments to show how the ECN++ FIN and pure ACK packets are treated by the network and the servers. We also measure how ECN-enabled data packets are treated to establish a baseline for comparison. Like previous experiments, we perform both client-side and client/server experiments (i.e., we perform these experiments with the Alexa top 100k servers and with our own servers).

In the experiments, the client uses Tracebox with PureACKs, Data packets, and FINs with the different combinations of the ECN codepoints and TCP flags. In all cases, the client establishes a standard ECN TCP connection before sending the test packets.

2.2.2.5 Response to Congestion Signal

We execute a number of client-side experiments to determine how the deployed base of ECN-enabled servers respond to ECN congestion signals. In particular, we want to learn if the congestion response to a CE marked packet echoed through one or more packets with the ECE flag set is equal to the response to three duplicate ACKs. We test for the case when a data packet is marked with CE (regular ECN) and the case when the CE marked packet is the SYN/ACK (ECN+ case). The approach we use is similar to the one used by TBIT [34]. For a given server, we measure the Initial Window (IW) of the TCP connection. After learning this, we establish a new TCP connection to the same server and we pretend that the first data packet sent by the server has experienced congestion (either pretending the packet is lost and by sending 3 duplicated ACKs or by sending the ACK for that packet with the ECE flag set) and we measure the resulting congestion window after such a congestion signal, learning the response to congestion from the server in these two situations (CE mark, or packet loss). In particular, we are able to learn if the response is the same for the two congestion signals. We also perform this test pretending that the packet that encountered congestion is the ACK of the TCP three way handshake (3WHS), allowing us to test the congestion response for servers that support ECN+.

Table 6: Experiments and datasets

Purpose	Tool used	Campaign Name	Date
Collect server MSS	PATHspider	A.1 ”PATHspider”	Jan 2018
Validate server MSS	Ping	A.2 ”Ping”	Feb 2018
Collect wireless/mobile client MSS	Pathtrace	B.1 ”MONROE”	Dec 2016
Collect wired edge client MSS	RIPE Atlas Traceroute	B.2 ”RIPE”	Jan 2018
Explore server PMTUD	Scamper	C.1 ”Scamper”	Jan 2018
Explore edge PMTUD	Netalyzr Traceroute	C.2 ”Netalyzr”	Dec 2017
Inspect ICMP quotations	Pathtrace	D ”ICMP”	Jan 2017

2.2.2.6 Results

Bad news: More than half of the mobile carriers we tested bleach (clear) the ECN field at the first upstream IP hop. This contradicts the impression of hardly any ECN traversal problems that has been reinforced by all recent studies. Nonetheless, our testing with fixed connectivity is consistent with these previous studies.

But not awful news: Bleaching ECN is benign[4], the connection continues, but without the benefits of ECN. We find no evidence of packets carrying the ECN capability being blocked.

Good news: Wherever ECN gets through, we find no problems for ECN++. Although the bad news for the base ECN protocol is also bad for ECN++, at least this suggests that, as ECN traversal problems are fixed, ECN++ traversal should improve in tandem.

Good news: We also find no problems with how servers respond to ECN-marking, but we do find some interesting congestion behaviors unrelated to ECN.

2.3 Maximum Transmission Unit along Internet paths

Note: this section is condensed from a paper by UNIABDN presented at the Traffic Measurement Analysis conference [12]; refer to the paper for details.

A series of measurements were performed to determine the current state of Path MTU Discovery (PMTUD) and mechanisms used to mitigate it. We employ several tools, platforms and experiments to explore various aspects of PMTUD, presented in Table 6.

2.3.1 Campaign A - PATHspider and Ping

PATHspider was used to collect advertised MSS for servers in the Internet core, by sending HTTP requests to the Umbrella top 1M domains and collecting the advertised MSS on the return path. Tests ran from the Janet 2 academic network (Campaign A.1, PATHspider) in December 2017. For the same targets, an ICMP ping test subsequently determined whether or not the target responds to a control ping the size of the server’s advertised MSS as well as a ping 1500 bytes (Campaign A.2, Ping) in size.

[4]Nonetheless, if other links precede the cellular hop (e.g. a home router or bus/train connected over cellular), any CE-marking introduced in the home or vehicle network would be wiped, which would fool ECN sources into overrunning their local network.

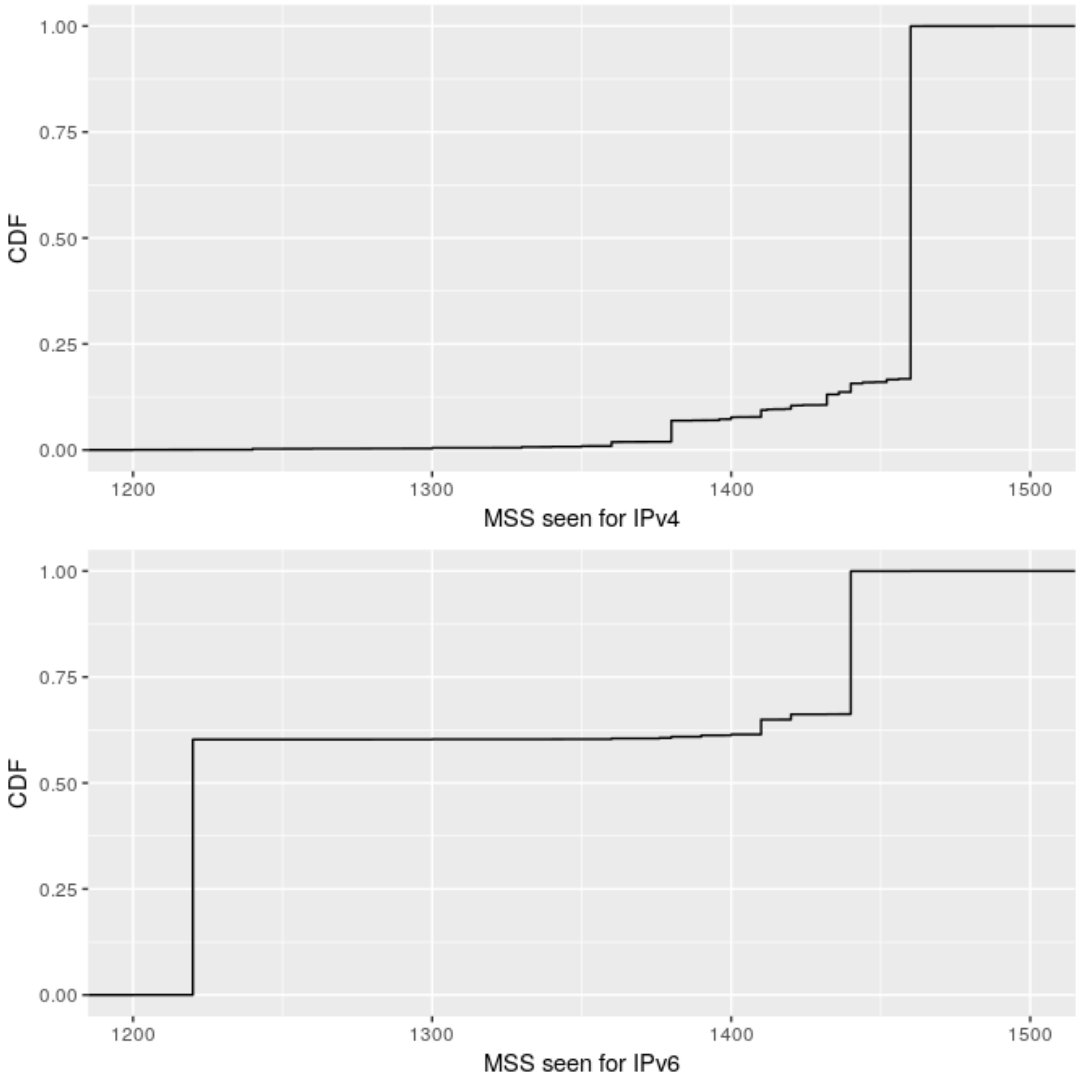


Figure 3: Advertised MSS (in bytes) on TCP SYN/ACK server response seen at Janet academic network

Figure 3 presents the distribution of MSS seen from clients, for IPv4 and IPv6. As expected, 80% of IPv4 clients will advertise an MSS of 1460, a larger proportion than previously found. For IPv6 however, more than 60% of webservers advertised an MSS of 1220 bytes.

In the subsequent ping test, two-thirds (65.8%) of targets were reach- able with our probes, as outlined in table 7. We found 98.4% of reachable servers responded to a ping of 1500 bytes, indicating the path supported this PMTU. Out of the subset of reachable servers which advertised an MSS lower than 1460 (34920 in total), 93% were reachable with a probe of 1500 bytes.

Table 7: Ping experiment results n=295,488

Behaviour	hosts	%
No response to ping from Server	101259	34.2%
Both probes failed with PTB message	657	0.2%
Both probes succeeded	190219	64.3%
1500 byte probe failed	2378	0.8%
1500 byte probe failed with PTB message	568	0.2%

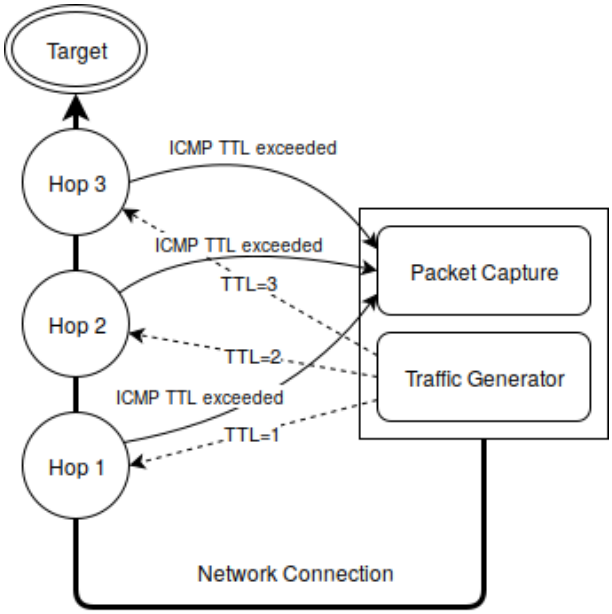


Figure 4: Pathtrace core architecture

2.3.2 Campaign B - MONROE and RIPE

We assessed MSS seen from clients in the mobile edge using traceroute-style measurements on the MONROE platform. The architecture of the tool used can be seen in Figure 4. We send packets without any additional TCP options, and then dissect the quotations returned to us via ICMP TTL exceeded messages. Our tests observed a total of 888 (21%) hops that returned a quotation with an MSS option, evidence of MSS Clamping in the mobile and edge networks. We furthermore sent 1500 byte UDP probes with the DF flag set on ten of the mobile paths that were found to insert an MSS option, and identified some of the networks that insert these options in Table 8.

To evaluate the wired edge, we performed a TCP traceroute from 3000 RIPE Atlas probes towards a server controlled by University of Aberdeen. When performing TCP traceroute tests, RIPE Atlas probes send a TCP SYN packet with no options. Our web server advertises an MSS of 1460 when replying. RIPE Atlas allows us to observe the MSS received from our server over the return path. At the same time, we capture the MSS that was advertised by the probes. This allows us to infer whether MSS manipulations in the path are symmetric.

We find 4.8% of the sent probes arrive at our server carrying an MSS option, inserted by a middlebox in the path. Significantly more (23% or 764) of the MSS values received by the probes

Table 8: Inserted MSS options by mobile network, $n = 10\ paths$

Network	Inserted MSS option
Telenor Norway	1410 bytes
Telia Sweden	1400 bytes
Vodafone Italy	1400 bytes
Wind Italy	1420 bytes

Table 9: IPv4 PMTUD behaviour

	1420 MTU	576 MTU	576 MTU Black-hole
PMTUD Too Small	7.45%	3.7%	0.95%
PMTUD Success	68.2%	63.9%	8.2%
PMTUD Failure	16.4%	19.5%	67.4%
No DF set	12.5%	12.3%	15.2%
Clear DF	2.7%	4.1%	NIL

differ from the value that was sent, suggesting middleboxes are more likely to manipulate this option only if it already exists.

2.3.3 Campaign C - Scamper and Netalyzr

PMTUD behaviour was measured with Scamper [27], from the Janet academic network. The Scamper methodology [32] takes the MTU to be tested as a parameter. If the HTTP response of the target is larger than the tested MTU and has the DF bit set, Scamper sends an ICMP Destination Unreachable or PTB message towards the target. PMTUD is inferred successful if the target reduces the size of its packets, failed if there is no response from the target after four retransmissions. Scamper also records whether the tested server retransmits IPv4 packets clearing the DF flag or if the observed packets did not have the DF flag set. We analyzed PMTUD behaviour for 60,000 domains chosen from the Cisco Umbrella top 100,000. The test MTU values were 1420 and 576 bytes. The test was then repeated, but filtering the local ICMP PTB messages, using Scamper's 'blackhole' option. We have observed up to 20% failure rate in PMTUD for IPv4, while for IPv6 a large percentage of the tested servers did not supply large enough responses to perform the test. Of the servers that did, over 95% were successful in performing PMTUD. Tables 9 and 10 present the full results.

PMTUD from the mobile edge was explored using Netalyzr [22] measurements from 50 vantage points - Campaign C.2, "Netalyzr". This experiment consistently reported a PMTU of 1500 bytes.

Table 10: IPv6 PMTUD behaviour

	1280 MTU	1280 MTU Black-hole
PMTUD Too Small	59.6%	53.1%
PMTUD Success	95.5%	32%
PMTUD Failure	4.5%	67.9%

2.3.4 Campaign D - ICMP

For network tests across the Internet core we analyzed a total of 14257 ICMP and ICMPv6 replies that contained a quotation (Campaign D, “ICMP”), to determine whether quoted packets could be used to validate ICMP PTB messages. For this purpose, we have studied the length and whether the quoted packets match the original probe on the 5-tuple. We found most quotations were observed to have the minimum permitted size by RFC792, 64.5% for IPv4 and 89.8% for IPv6. We found no evidence of quotations smaller than the minimum permitted size, which means all information required to match a probe is returned. We also found evidence of erroneous behaviour, 12 replies which did not match the original probe on the 5 tuple.

Section V-E of [12] contains a complete analysis of these results.

2.4 Path Transparency to UDP Options

UDP Options [42] is an IETF work in progress to add transport options to UDP. The transport option space is created by using a redundancy in length fields when UDP is carried in IP. Normally both the IP payload length and the UDP length refer to the same size, option space is created by varying the IP payload length while not varying the UDP length.

The mechanism UDP Options uses has always been supported on the Internet, however, the redundancy between these fields has led to the majority of implementations using only the IP Payload length when evaluating UDP checksums.

Use of this surplus space is new in the Internet, so we conducted a series of measurements to evaluate how UDP Options datagrams would be treated in the network.

The `tracemore`[5] tool was used to generate UDP Options datagrams and perform measurements. `tracemore` is a flexible traffic generator which performs a ring search by decreasing the IP hop count field to see where in the network packets are modified or dropped.

A first round of measurements were performed to see how UDP Options datagrams worked in the network, from these measurements a set of five middlebox failure pathologies were created. The pathologies describe the main causes for a UDP Options datagram to be dropped:

1. Full Payload Checksum
2. Full Payload Checksum, UDP length Pseudoheader
3. UDP Length Checksum, IP length Pseudoheader
4. Only passes 0s as options space
5. Only passes IP payload length == UDP Length

The first round of measurements discovered that approximately half the paths tested would drop UDP Options datagrams due to a mismatch in the calculation of the UDP checksum. The most common pathology leading to a UDP Options datagram to be dropped was a checksum based on the IP length rather than the UDP Length.

[5] `http://www.middleboxes.org/tracemore/`

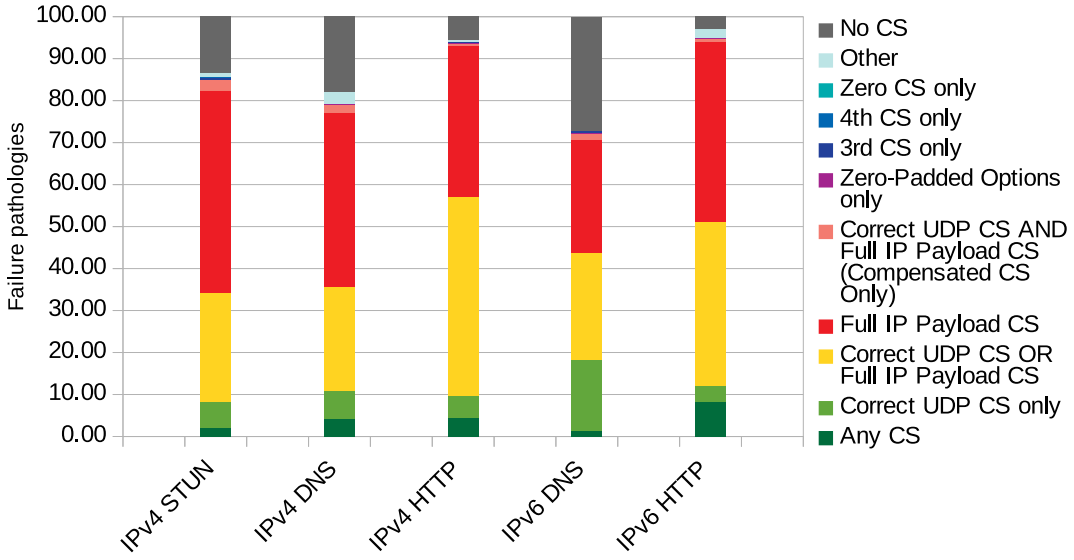


Figure 5: Chance of Failure for UDP Options Datagrams with different measurement packets

A new UDP Options, the Checksum Compensation Option (CCO) [17] was developed to work around this issue. The mechanism of the CCO is described in Section 4 of [17], in short it uses a checksum in the UDP Options space that causes both the correctly calculated checksum based on UDP Length and the incorrectly calculated checksum based on IP Payload Length to have the same result.

To measure the utility of the CCO a `tracemore` test suite was created to test each type of middlebox failure. `tracemore` measurements were performed using several variants of UDP packets and a method was developed to measure against public servers that do not host UDP services.

Lists of public STUN and DNS servers were used as targets for UDP Options datagrams. The Alexa list of web servers was also used for testing to get a wider range of paths. As the Alexa web servers do not host public UDP services these paths were tested by sending datagrams to the host and listening for ICMP Port Unreachable error messages. This error signals that the packet has been processed by the host, but there was no listening server.

A test suite consisting of 11 packets was used to evaluate each path:

1. UDP Packet
2. UDP Packet zero checksum
3. UDP Packet wrong checksum
4. UDP Packet with surplus area padded with zeros
5. UDP Packet with Options
6. UDP Packet with Options 3rd CS (checksum pseudo header using UDP length)
7. UDP Packet with Options 4th CS (checksum pseudo header using IP length)

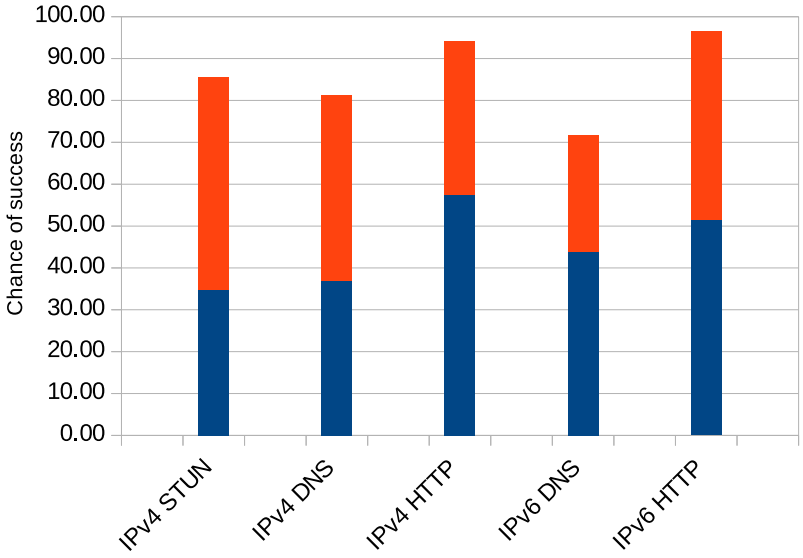


Figure 6: Chance of success for UDP Options Datagrams and UDP Options Datagrams with the CCO Option. The blue bar shows paths which will pass UDP Options Datagrams while the Red bar shows the additional paths made available by using the CCO with UDP Options.

8. UDP Packet with Options, Full IP Payload CS
9. UDP Packet with Options, zero CS
10. UDP Packet with Options, wrong CS
11. UDP Packet with Options and CCO

Packets 1-3 of the test suite evaluate if the path to the test host is valid and working as expected, while tests 4-11 each evaluate one of the failure pathologies discussed earlier.

Figure 5 shows a proportional breakdown of failure pathology for each tested protocol and address family. The packet test suite was able to reveal 11 distinct reasons for why a UDP datagram would be dropped due or passed based on the checksum. The final category 'other' represents measurements where the packet is dropped for some reason in the host UDP stack. The 'No CS' category are paths where no UDP packet was able to work end to end.

Figure 6 shows the chance of success of a UDP Options Datagram making it end to end on a path and the additional paths made available by using the CCO.

The `tracemore` UDP Options measurements show checksum related issues for a large number of paths on the Internet. UDP Options senders without checksum compensation can expect very high failure rates. From these measurements the CCO should be recommended for all UDP Options datagrams.

A paper is being prepared for publication in 2019 to report the findings on UDP Options impediments in the network.

2.5 Detection of NATs in wireless networks

This sections introduces a novel methodology that we devloped for detecting NATs embodied in `Mobile Tracebox` [45], a measurement tool for Android smart devices that detects a wide range of middle- boxes. It analyzes ICMP time-exceeded messages received during traceroute and points at IP and transport checksum inconsistencies in the embedded packets to uncover address translation along a path. We deployed `Mobile Tracebox` through a crowdsourcing approach and used the collected dataset to validate our methodology. Results showed that, in absence of middleboxes breaking traceroute, it can help to detect and locate NATs in the majority of the cases.

2.5.1 Detecting NATs

Standard tracebox methodology is able to detect a wide spectrum of modifications performed by middleboxes. Unfortunately modifications on source address and port – indeed those performed by NAT – are not among them. The reason is given in RFC 5508 [40], stating NAT best current practice with respect to ICMP Error Packet Translation.

When a NAT device receives an ICMP Error packet from the external realm, if the NAT has an active mapping for the embedded payload, RFC5508 prescribes the NAT to do the following prior to forwarding the packet:

1. Revert the IP and Transport headers of the embedded IP packet to their original form, using the matching mapping;
2. Leave the ICMP Error type and code unchanged;
3. Modify the destination IP address of the outer IP header to be same as the source IP address of the embedded packet after translation.

Point 3 is needed for the host whose packet has expired in transit to receive the ICMP error message while Point 1 is needed for the same host to understand which connection (in other words which socket) the error is related to. This means that the NAT'ed address and port of the quoted packet – inside the ICMP message – are restored to the original values making the NAT transparent. To leave no stone unturned, we used at first an experimental version of `Mobile Tracebox` where the core was set to receive any incoming ICMP `time_exceeded` message. In this way, tracebox was able to detect any modification on packets, including source address and port if not properly restored by NAT but, unfortunately, it was not of any help, resulting only in a large amount of inconclusive probes.

After this preliminary survey, we set the tracebox core to correctly retrieve only ICMP packets where the embedded packet matches sent packet 5-tuple (source address, destination address, source port, destination port, transport protocol), and we switched to a smarter methodology. NATs, besides source address and port, have to update both IP and TCP/UDP checksum: even if the NAT does not alter the port, the transport checksum has to be updated since IP source address is part of the *pseudoheader*. We have explained earlier how NATs restore the initial values of source address and port, the next question is: are they restoring also IP and transport checksum? Looking back to RFC5508 [40], it states that NAT should "revert the IP

Table 11: Checksum error coherence with address/port offset.

	Address+Port	Address only	None	All
IP	–	2	5	7
TCP	7	2	3	12
UDP	3	5	3	12

and transport headers of the embedded IP packet to their original form", thus including checksums even if not explicitly asserted. On the other hand, a first deployment of `Mobile Tracebox` showed evidence of ICMP messages carrying an embedded packet with wrong IP or transport checksum, especially the latter. There is a good reason for tolerating wrong transport checksum inside ICMPv4 messages and it is related to quoting mechanism: although RFC1812 [8] recommends to include as much as possible of the original packet, the previous RFC792 [35] recommended only 8 bytes of Layer-4 data to be included and, in most cases, only a part of the transport data is reported: in this cases the checksum is, of necessity, inconsistent. RFC5508 suggests incorrect IP and transport checksums to be treated differently: IP checksum should be validated and a packet with wrong IP checksum should be dumped while transport checksum should not be validated. While analyzing data as they were collected from users, we realized that about 5% of routers crossed by tracebox showed checksum inconsistencies in Layer-4 checksum and about 1% in IP checksum.

The reasons behind these errors can be diverse: from malfunctioning in forwarding packets to any mix of modifications occurring on downstream and upstream nodes that do not compute the checksum accordingly. Mobile Tracebox abstains from evaluating transport checksum when the entire packet is not available, thus partial quoting can be excluded as a cause for detected checksum inconsistencies. Inspecting those errors, we saw evidence that they can be related to NATs:

- errors appear at some hop and persist after that hop until the destination;
- for repeated probes, errors always show up at the same hop (even with different values);
- packets within a single connection show the same offset, for instance TCP `SYN` and the following `ACK` packet, although having different sent and received transport checksums, turn out to have the same offset from the respective correct values;
- packets belonging to different connections show the same IP checksum offset.

All these observations are compatible with NATs.

We analyzed probes from 22 networks, both cellular and Wi-Fi, and checked for coherence between IP and transport checksum errors detected through tracebox and modifications on source address and port detected at the server.

Results presented in Table 11 show how checksum errors can be linked to NATs. In these cases, the error detected via tracebox matches exactly the address (and port) translation offset. Surprisingly, a few transport checksum errors (especially for UDP) match only the offset between private address and NAT'ed address, even when the probe to our server confirmed the presence of a NAPT. IP checksum errors can only match addresses offset because IP checksum does not include port information.

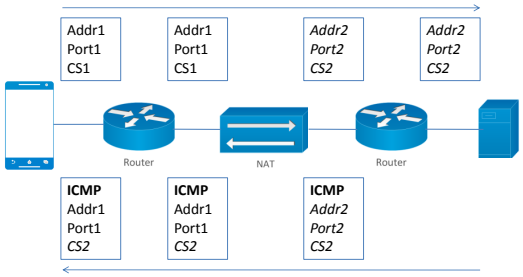


Figure 7: Detecting NATs through checksum errors.

In Figure 7 a typical scenario where our methodology works is shown. The NAT is translating a client's private address and port and updating the transport checksum on the client's outgoing packet. The packet is received at the server with the fields `Addr2`, `Port2`, `Checksum2`. When an incoming ICMP `time_exceeded` message reaches the NAT, it restores `Addr1` and `Port1` in the embedded packet but does not update the checksum: the packet retrieved through `traceroute` has the fields `Addr1`, `Port1`, `Checksum2`. The client then computes the offset between `Checksum1` and `Checksum2`.

Errors that do not match NAT offsets can be due to other middeboxes modifications that are performed upstream and restored downstream (without restoring checksum) but we did not find evidence of correlation with other fields alterations detected at the server. They can be still related to NAT in a more subtle way and must be further investigated. Two networks showed at the same hop a transport checksum error matching the offset and an IP checksum error not matching the offset: we cannot exclude that even the IP checksum error is due to NAT. We must clarify that the test above should not succeed when there is more than one NAT and only one is not updating checksums. In this case, the error is related to one NAT while the offset is due to multiple address translations. We had physical access to one of the networks that showed no coherence between checksum error and address/port offset and verified that it had a double level of NAT.

2.5.2 Deployment

We show now results of the analysis of IPv4 probes. We will analyze IPv6 probes in the last part of this section. Fig. 8 displays how many probes are showing IP and transport checksum errors: a significant portion of paths presents TCP or UDP checksum errors. IP checksum errors are less frequent as NATs are probably less negligent in manipulating IP header of embedded packets.

Results in Fig. 8 can underestimate the extent of this methodology as they are based on the entirety of probes: not all probes in fact are supposed to exhibit that previous behavior. To gain a better perspective we have to refine the dataset with respect to three features: the address owned by the host, the presence of other middleboxes that break traceroute, and the partial embedding of Layer-4 data in ICMP error messages. First of all, we have to differentiate probes in which a host has a private address and probes in which the host has a public address: while in the first case the presence of NAT is implied, in the second instead it is quite unlikely. Then, we have to exclude the probes where other middleboxes break traceroute. Due to the presence of a proxy or a firewall blocking ICMP messages many probes exhibits no or just a

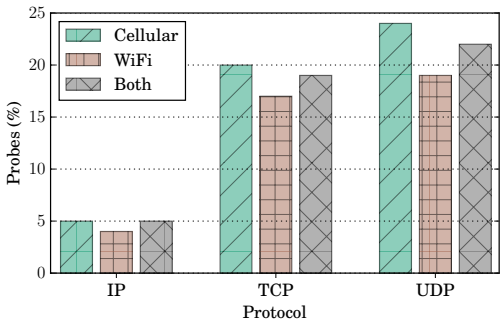


Figure 8: Checksum errors raw percent on probes.

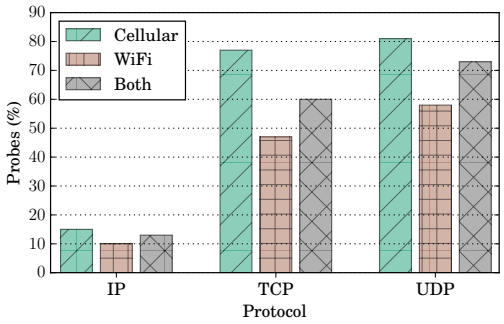


Figure 9: Checksum errors refined percent on probes.

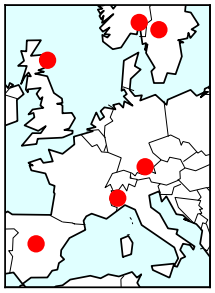
few intermediate hops: as we linked errors to NAT, a probe should show in the path at least one router with a public address, thus residing outside the private network. Another consideration concerns the amount of Layer-4 data encapsulated in ICMP messages. Although RFC1812 recommends to quote the entire packets, many IPv4 routers quote only the first 8 bytes of transport layer, as originally required by RFC792: `Mobile Tracebox` cannot validate a Layer-4 checksum if the packet is not available in its entirety. This leads us to restrict the eligible probes to: probes with at least a public address router with regard to IP checksum, probes with at least a public address router quoting the full original packet with regard to transport checksum.

Fig. 9 displays the refined statistics for checksum errors. It shows how, under the circumstances discussed before, the majority of probes that are supposed to cross a NAT in traceroute path show checksum inconsistencies.

We excluded probes where the host has a public IP address from last statistics. We analyzed separately 320 probes collected from 8 networks but did not find any evidence of checksum errors. This is not surprising: even if still possible, the presence of a NAT is quite unlikely in those cases.

All the statistics provided till now are based only on IPv4 probes. We explored also IPv6 probes in the dataset looking for checksum errors: 160 probes belonging to 5 networks did not show any inconsistency in transport checksum (IPv6 has no checksum at IP layer). Again it is entirely in line with the relation between checksum errors and address translation since a host owning an IPv6 public address is not supposed to cross a NAT.

Results demonstrate that, when no other middleboxes (e.g., proxies, firewall blocking ICMP, etc) are obstructing traceroute, `Mobile Tracebox`, an extension to tracebox developed by the MAMI project, can detect and locate NATs in the majority of cases.



	Mobile Network Operators (MNOs)		
NO	Telia NO	Telenor NO	
SE	Telia SE	Telenor SE	3 SE
UK	Vodafone UK	EE	
DE	Vodafone DE	T-Mobile	O2
ES	Vodafone ES	Movistar	Orange
IT	Vodafone IT	TIM	3 IT

Figure 10: The distribution (left) of the MONROE-Roaming nodes in six countries and (right) SIMs for 16 MNOs we measure across Europe. Each country deploys two MONROE-Roaming nodes and one measurement server.

3 Related Measurements

This section presents measurements of phenomena which are not directly relevant to Internet path transparency, but which support the wider goals of the MAMI project to study and help enable deployable evolution of the Internet's transport layer.

3.1 Implications of International Data Roaming in Europe

Note: this section is condensed from a paper by TID, SRL and UNIABDN [31] presented at the ACM MobiCom 2018 conference; refer to the paper for details.

To measure the implications of international data roaming in Europe, we design and build MONROE-Roaming, a dedicated hardware platform for roaming measurements in Europe. The main blocks include measurement nodes distributed in six different EU countries (NO, SE, DE, UK, IT, ES), the backend system, several measurement servers and a scheduler, all of which we detail next. To build the MONROE-Roaming platform we adapted the open source software provided by MONROE. To understand the roaming ecosystem in Europe, we focus on diversity of the Mobile Network Operators (MNOs). In other words, we aim to cover a large number of Subscriber Identity Modules (SIMs) rather than running measurements from a large number of vantage points. To this end, we deployed two MONROE-Roaming nodes in each of the six European countries to measure a total of 16 MNOs that operate their own network, as illustrated in Fig. 10.

For each MNO, we bought six SIMs that support roaming in Europe and we distributed one SIM in each of the countries we cover. For example, in Germany, we bought six Vodafone DE SIMs that support roaming. We kept one Vodafone DE SIM as the *home SIM* in the home country (i.e., Germany). Then, we distributed five *roaming SIMs* from Vodafone DE to the other five countries (i.e, Sweden, Norway, UK, Italy and Spain). Each roaming SIM connects to (or *camps on*) a local roaming partner (or visited network) native to the visited country. For example, Vodafone DE in Germany is a roaming partner of Telenor SE in Sweden. Therefore, Telenor SE serves Vodafone DE's customers roaming in Sweden by allowing Vodafone DE users to camp on Telenor SE's network. For each roaming SIM, we identify the corresponding visited network (e.g., Telenor SE in Sweden for Vodafone DE) and, when available, activate

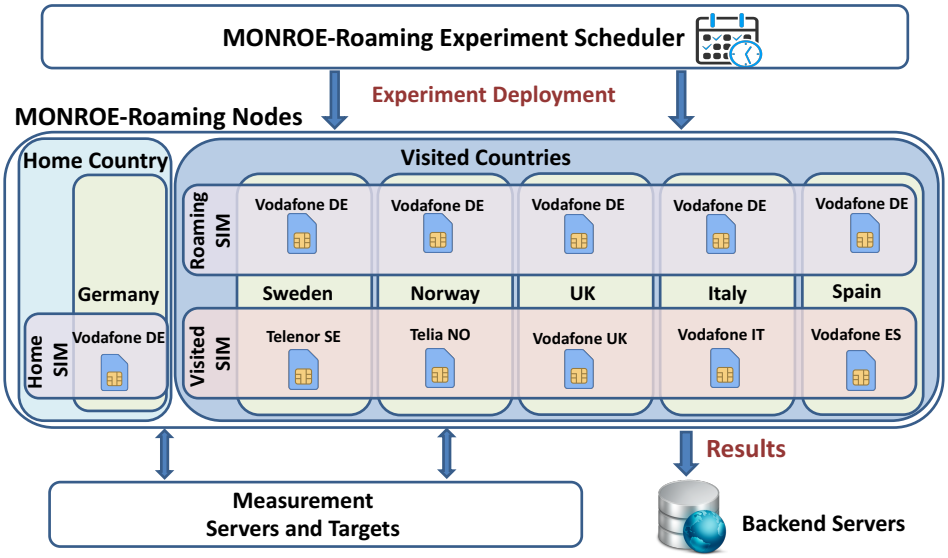


Figure 11: MONROE-Roaming platform and experimental setup. We exemplify our setup for Vodafone DE. We have five Vodafone DE SIMs in international roaming nodes and another SIM in the home country nodes. For each roaming Vodafone DE SIM, we insert the SIM corresponding to the local roaming partner for the MNO. For example, in Sweden we use the Telenor SE SIM which corresponds to the network on which the Vodafone DE SIM is camping.

the corresponding native SIM from the visited network (which we hereinafter denote by *visited SIM*). We illustrate this configuration in the experimental setup in Fig. 11.

3.1.1 Roaming Setup and Performance

We run a series of measurements that enable us to identify the roaming setup for each MNO, infer the network configuration for the 16 MNOs that we measure and quantify the end-user performance for the roaming configurations which we detect. We run `traceroute` for path discovery, `dig` for Domain Name Service (DNS) lookups and `curl` for testing data transfers with popular URLs. We complement this analysis with metadata (e.g., radio access technology, signal strength parameters) collected from each node. For each MNO, we measure in parallel the roaming user, the home user and the visited user through the MONROE-Roaming scheduler. In this way, we are able to capture potential performance penalties that might result, for example, from roaming internationally under a home-routed configuration.

`traceroute`: We run periodic `traceroute` measurements against all the servers we deploy in each country as measurement responders. We repeat the measurements ten times towards each target.

`dig`: We run the `dig` utility for DNS lookups against a list of 180 target Fully Qualified Domain Names (FQDNs) mapped to advertisement services.

`curl`: We run `curl` towards a set of 10 target popular webpages over HTTP1.1/TLS. We repeat the measurements towards each URL at least 10 times (increasing the sample size if the SIM data quota allows it). We store various metrics, including the download speed, the size of the

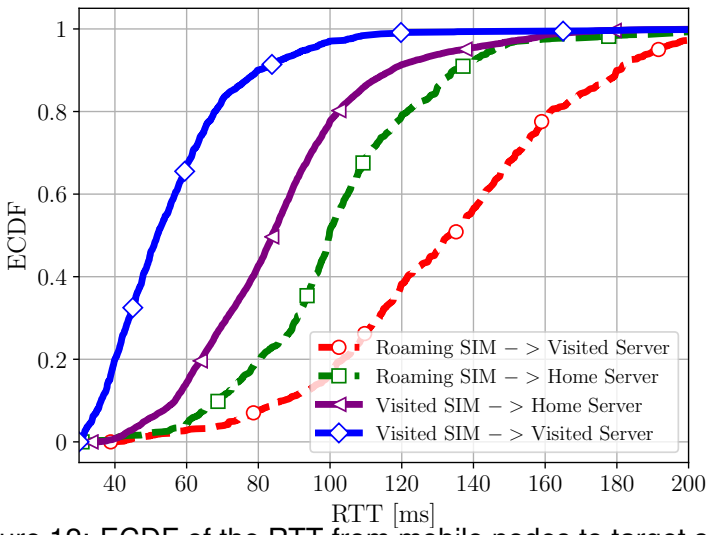


Figure 12: ECDF of the RTT from mobile nodes to target servers.

download, the total time of the test, the time to first byte, the name lookup time (query time) and the handshake time.

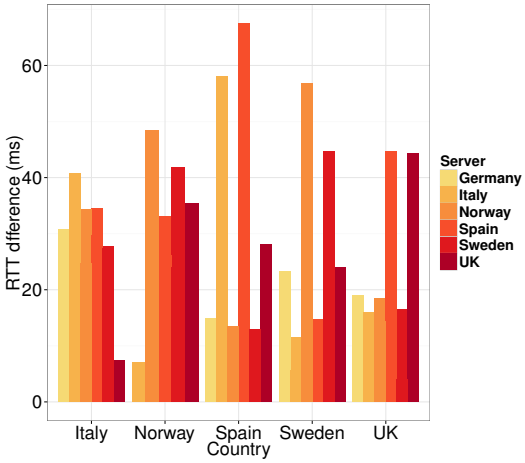
`metadata`: We collect contextual information from the nodes, including the visited network Mobile Country Code (MCC) / Mobile Network Code (MNC) for each roaming SIM and the radio technology. This allows us to verify which visited network each roaming SIM uses as well as to identify and separate the collected data by radio technology.

`targets`: We target the following 10 popular web pages: `www.httpvshttps.com`, `facebook.com/telia/`, `en.wikipedia.org/wiki/Timeline_of_the_far_future`, `linkedin.com/company/facebook`, `www.yahoo.com/movies`, `instagram.com/leomessi/`, `google.com/search?q=iPhone+7`, `youtube.com/watch?v=xGJ5a7uIZ1g`, `ebay.com/globaldeals`, `nytimes.com`, `theguardian.com.uk/lifeandstyle`.

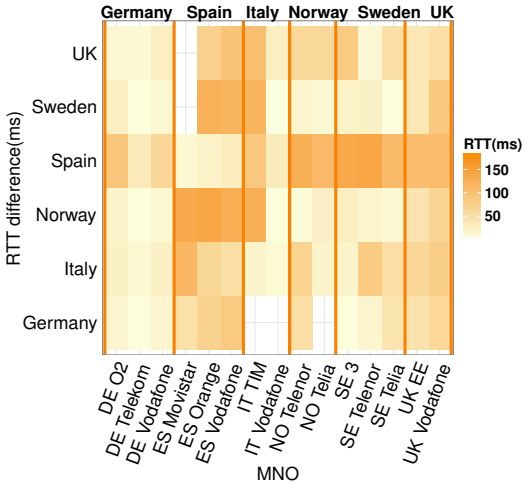
Our initial goal is to determine the roaming setup for each MNO (i.e., whether it uses local breakout (LBO), home-routed roaming (HR) or IPX hub breakout (IHBO)). For this, we determine the MNO that allocates the public IP address of the roaming SIM. Our results show that *HR was used by all 16 MNOs from all the different roaming locations we capture*. We further corroborate this result by retrieving the first hop replying with a public IP address along the data path from a roaming SIM to each server and identifying the MNO that owns it. We find that the first hop with a public IP address along the path lies in the original home network of each roaming SIM, which is consistent with HR.

3.1.2 Home-Routed Roaming Implications

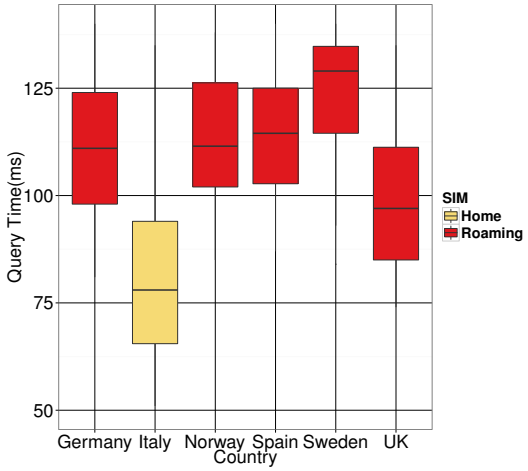
Delay implications The HR data implies that the roaming user's exit point to the Internet is always in the original home network. Thus, the data that the roaming user consumes always flow through the home network. Depending on the location of the server, this translates to a potential delay penalty. Fig. 12 shows the ECDF of the RTT we measured between the roaming SIMs and the target servers located in the visited or home networks (red and green



(a) RTT difference from the visited country to all servers for Vodafone DE.



(b) RTT difference per operator.



(c) DNS Query time to all FQDNs for TIM IT.

Figure 13: Delay penalty of HR.

curves, respectively). To compare the HR with the LBO configuration, we also include the RTT measurements between the visited SIMs against the same targets in the visited or home networks (blue and purple curves, respectively). The RTTs experienced by the visited SIMs serve as estimates of the best RTTs that one could expect with a LBO configuration, since LBO relies on access to local infrastructure with no need for tunnelling back to the home network. We note that the largest delay penalty occurs when the roaming user tries to access a server located in the visited country. This is because the packets must go back and forth from the home network. Surprisingly, we note that the HR configuration also impacts the case when the roaming user accesses a target server located in the home network. That is, the GTP tunnel is slower than the native Internet path. In this case, the median value of the delay penalty considering all the MNOs is approximately 17ms. This varies across MNOs and in some cases we observe very low penalties (e.g., just 0.2ms for O2 Germany).

We investigate this performance impact further and calculate the estimated delay penalty between LBO and HR when the target is in the visited network. In more detail, we compute the delay penalty as the difference between the median delay to reach a given server when roaming, and the median delay to reach the same server from home. Fig. 13a exemplifies these median values for Vodafone Germany. We note that, in general, the delay penalty varies widely with the geographical location of the roaming users and the target servers. For example, when a German SIM roams in Spain, the difference in terms of RTT is higher if the server is in the visited country (i.e., Spain) (red curve in Fig. 12). If the German SIM roams in Spain or Italy and the target server is in Norway or Sweden the delay penalty of the roaming is smaller, since to go to Norway or Sweden the data path would anyway likely pass through Germany (and this is similar to the delay one would have because of the HR configuration).

We then evaluate the RTT difference between the roaming SIM and the visited SIM towards the same target and we group them per MNO. Fig. 13b shows the median value of the delay penalty of an MNO (on the `x` axis of the tile plot) while roaming against each of the six different servers (on the `y` axis of the tile plot, marked by country). We note that the delay penalty varies as a function of the location of the home country. For example, German SIMs experience a lower delay penalty, which is potentially due to them being in an advantageous position in the center of Europe.

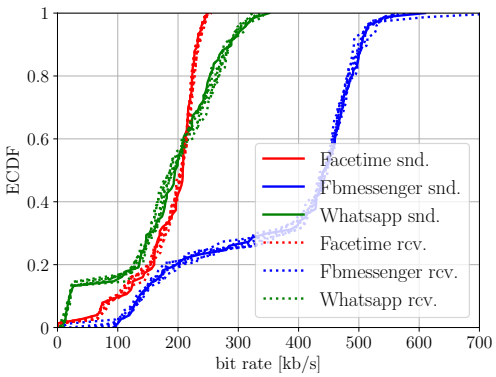
DNS implications The results of the `dig` measurements show that the DNS server offered to a roaming user is the same as the one offered when at home. This is again consistent with the use of HR. We verify whether this translates into an inflated query time for the roaming user. Fig. 13c presents the distribution of DNS query times for all the SIMs of TIM IT. We note that for the home user the query time is significantly lower in average than for the other five roaming users. This is consistent for all the 16 MNOs we measured. This further translates into implications in terms of CDN replica selection: the roaming user would be likely redirected to CDN content at its home network, and will not be able to access the same content from a local cache (which would in any case result in facing a higher delay due to the home routing policy).

HTTP performance implications Similar to the delay and DNS implications, international roaming affects HTTP and HTTPS performance. We quantify this penalty by considering the handshake time between each SIM and the target web servers. The median value of the handshake time from the visited SIMs towards all the targets we measure is 170ms, while the median value for the roaming SIMs is 230ms. This leads to a delay penalty of approximately 60ms. As in the cases before, some MNOs are affected more by this roaming effect than others.

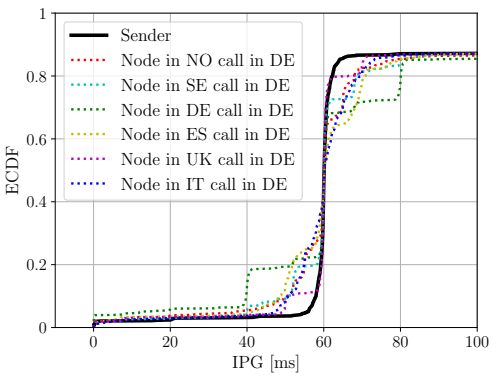
3.1.3 VoIP Calls

This section investigates potential traffic differentiation policies (such as blocking or throttling) that may hamper Voice over IP (VoIP) communications for a roaming user in comparison to a home user. We focus on three popular VoIP applications: FaceTime [1], Facebook Messenger, and Whatsapp [2].

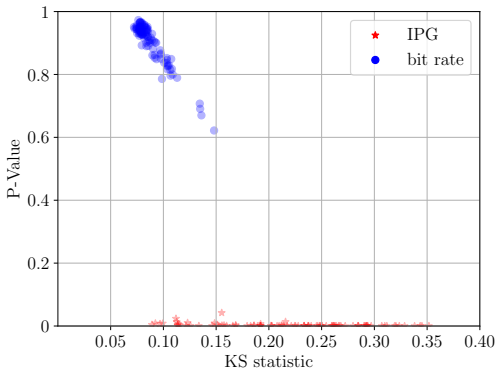
The experiment makes three audio and video calls using each application running on a regular mobile phone connected using an instrumented WLAN access point (AP) in our lab. Packet traces were recorded using `tcpdump` resulting in 18 traces, each with duration between



(a) Bitrate for operator O2 DE.



(b) IPG for operator O2 DE.



(c) KS and P-Value test.

Figure 14: A sample of VoIP results and statistical similarity of bitrate and IPG.

[60,80] s. We verify the call-setup phase, which used a complex mix of TCP, STUN [39], and custom protocols to setup the end-to-end communication. From each trace, we then extract the actual audio/video streams. In the next step, we create a Docker container with pre-loaded traces, which we replay using `tcpreplay` to properly edit them so that packets are directed toward dedicated receivers hosted in our premises.

All applications run SRTP [9] on top of UDP, enabling easy adjustment of the packet timing and updating of the source and destination IP addresses. The dedicated server in each country acts as a UDP receiver with a custom signalling TCP connection to log the status of the node (visited network, node identifier, metadata, experiment type, etc.) and the experiment associated with the receiver. In each test, the mobile node sequentially replays the pre-recorded traces with two receivers: a call to a destination in the home country, and a call to a destination in the visited country. For each call, we record `pcap` traces on both the client and server sides. We post-process these traces to check for traffic differentiation.

We do not observe any traffic differentiation on any of the 16 MNOs we measure. However, the additional delay of HR could impair real-time applications. This is an old issue (typically referred to as tromboning) which has been solved in GSM networks, but persists for 3G/4G VoIP applications.

We analyze well-known Quality of Service (QoS) metrics for real-time VoIP applications: packet loss, instantaneous bit rate, and Inter-Packet Gap (IPG), the time difference between two consecutive packets, to detect traffic differentiation. The results show that the packet loss ratio is less than 1% in all experiments. We conclude that no operators introduce artificial packet loss during our tests. MNOs could do so if desired to reduce the quality of calls for these applications and enforce traffic policing.

For each trace, we compare the bit rate we observe at the sender side and at the receiver side. Fig. 14a presents this as an ECDF for the three applications. Solid/dashed lines indicate the sender/receiver side when calling a receiver in the home or visited country (operator O2 DE). The applications use different audio/video codec combinations with different bit rate requirements. We observe no differences when using the tested network and the home network.

Fig. 14b shows a mobile user of the O2 DE operator making a Facetime call to our server in Germany while roaming. The periodic 60 ms long IPG is typical of low rate audio codecs that modern VoIP applications use. We observe some differences when comparing measurements at the sender (solid line) and the receiver (dashed lines – one for each visited country). Some gaps are compressed (a smaller IPG), while others become expanded (a larger IPG). We observe this in all experiments with all operators when the sender is in its home country. We ascribe this to the modulation of the IPG by 3G/4G access mechanisms. Given the IPG is bounded to less than 80 ms, we conclude that this would not hamper voice quality, and expect these variations to be absorbed by the receiver playout buffer [10].

All experiments give very similar results, pointing to no evidence of traffic manipulation. We summarize these findings using the well-known KS Test [3] and P-Value [44] to determine whether the ECDFs differ between the sender (our reference) and the receiver. If statistically similar, the KS would be close to 0 while P-Value would be close to 1. If significantly different, the KS would be greater than 0, and the P-Value close to 0. Fig. 14c shows the scatter plot of the (KS, P-Value) points. Our results confirm that the receiver throughput is statistically identical to the sender throughput in all experiments. The IPG statistics are affected by the 3G/4G access mechanisms that alter the distribution (albeit not impairing the VoIP quality).

Finally, while QoS in terms of IPG and throughput are good, the total end-to-end delay could be significantly higher when roaming because of the HR solution. The one-way-delay could thus grow excessively large when two roaming SIMs call each other, making the interactive voice conversation difficult. The same effect was faced in GSM networks, and fixed by anti-tromboning [6] solutions (e.g., allowing local breakout for voice traffic).

3.1.4 Content Discrimination

In this section, we evaluate the availability of content when roaming, in particular whether MNOs filter website content and apply geographical restrictions. There are many reasons operators could have content filtering, which include complying with government guidelines or following court orders, e.g, to restrict access to file-sharing websites in the UK [38], or the use of 'opt-out' parental filters. We refer to any differences in the availability of websites and their content due to network interference as "content discrimination". When this difference is attributed to geographical location rather than the studied network, it is known as "content geo-restriction".

The Open Observatory of Network Interference (OONI) [18] provides software tests for detecting censorship, surveillance and traffic manipulation in the Internet, using open source soft-

ware. We ran two measurement campaigns using OONI's tool `ooniprobe` to detect network interference in home and roaming scenarios, geared at both content discrimination and content geo-restriction.

The `ooniprobe` web connectivity test performs the following steps over both the network of interest (tested network, using both home and roaming SIMs) and the Tor network [26]: resolver identification, DNS lookup, TCP connect and HTTP GET requests. First, ooniprobe performs DNS queries to disclose the IP endpoint of the DNS resolver in the tested network, and records the response, alongside a control response returned by Google's DNS resolver. Then, a TCP connection on port 80 (or port 443 for URLs that support TLS) is attempted using the list of IP endpoints identified in the DNS responses. HTTP GET requests are sent towards a list of URLs over both the tested network and over the Tor network and the responses are recorded. Differences in the results for the two tests are indicative of network interference. The results are made available to the public via the OONI API[1]. Our first set of tests considers 50 randomly selected websites from `ooniprobe`'s default global censorship list contributed by users, seeking to identify content discrimination for roaming users. For the second set of measurements, we provide a list of 15 websites known to be available locally in the tested countries, but geo-restricted abroad. Beside network interference profile matching for the home and visited results, we additionally searched HTTP responses for known geo-restriction indicators and warnings. The high latency of the Tor network and data quota limitations limited us to test only a small number of websites.

We found no evidence of content discrimination and geo-restriction for users in roaming scenarios, and the experience of browsing websites was the same in roaming and at home. However, there are clear differences between the experiences of a user of a network and a user visiting the same network, including the inability to retrieve geo-restricted content and the availability of different content.

3.2 Measurement Study of Mobile Cloud Services

Note: this section is condensed from a paper by SRL [33] that was presented at IEEE INFOCOM 2018; refer to the paper for details.

The line distinguishing a CDN from a cloud computing provider can be blurry at times: third-party service providers may simultaneously offer cloud computing and CDN services using the same domain names and IP blocks. Due to this classification challenge, in this study we analyze them together using the term *Cloud Service Providers (CSP)*.

We empirically analyze the web of relationships between mobile apps, CSPs, and MNOs. In particular, we aim to answer the following questions:

- Which are the most dominant CSPs enabling the mobile Internet?
- How well are these CSPs interconnected with MNOs at a topological level?
- What is the performance of these services (i.e., as perceived by end-users) when accessed from commercial MNOs?

[1] https://api.ooni.io/

We conducted the first comprehensive study of its kind, combining different measurement techniques and vantage points to fully capture the synergies between the entities forming this complex ecosystem. As a starting point, we use traffic logs that we collected with Lumen Privacy Monitor [37], a mobile privacy and transparency tool. Lumen's rich traffic logs allow us to accurately identify the most prevalent CSPs providing on-line infrastructure to 8,281 mobile apps in the wild. Then, we run a purpose-specific month-long measurement campaign using the MONROE platform for mobile broadband measurements to capture the interactions between ten commercial MNOs from four European countries and the most popular CSPs, as well as their transport- and application-layer performance. Specifically, we focus on analyzing the effect of replica selection, the role of the DNS subsystem, and the impact of in-path TCP splitting proxies, as well as routing- and peering-level effects on transport-layer performance. Our study also includes mobile subscriptions roaming internationally.

Our analysis reveals that six CSPs— Amazon Web Services (AWS), Google, Facebook, Akamai, Amazon CloudFront, and Highwinds — provide infrastructure and online support to 85% of the apps that we measure with Lumen. We captured interesting multi-CSP strategies that 687 second level domains (15% of domains) use to increase their geographical coverage and reliability. We also tracked the integration and collaboration strategies between the top CSPs identified through Lumen and the MNOs available in the MONROE platform. In particular, Akamai's strategic alliances with multiple MNOs stand out. The varying degrees of collaboration between MNOs and CSPs translates into notable performance differences, which we actively measured and analyzed. Namely, the tight integration between MNOs and CSPs results in lower latency and connection times: Google's relationship with various MNOs has resulted in 15% lower connection times on average compared to other similarly performing CSPs. We detect various levels of EDNS adoption among the studied operators, which, however, does not seem to have any significant impact on performance. Finally, international roaming may add significant delays, especially in the case of well provisioned CSPs, defeating their attempts to put content close to the user.

3.3 Revisiting Privacy in Latency Data

Note: this section is condensed from a paper [43] published by ETH at the 2018 Passive and Active Measurement conference. Refer to the paper for more details.

As part of the MAMI project's work on the QUIC spin bit (an explicit signal for exposing end-to-end RTT information to on-path measurement devices; see Deliverable 3.3 for details), ETH, together with the IETF QUIC working group RTT Design Team, designed an active measurement study leveraging existing RIPE Atlas and MONROE data to determine the extent to which explicitly providing end-to-end latency data as a transport feature may pose a geoprivacy threat.

We begin by considering the components of observed RTT RTT_{obs} in eq. (3.1), for f hops in one direction and r hops in the opposite direction, where D_{prop} is propagation delay on a link, D_{queue} is queueing delay at a forwarding node, D_{proc} is processing delay at a forwarding node, D_{stack} is stack delay at the remote endpoint (the time it takes for a packet to make it from the network interface to the application and back, including acknowledgment delay [16] when traffic is unidirectional), and D_{app} is application delay at that endpoint.

$$RTT_{obs} = \sum_{n=0}^{f}(D_{prop_{n\to n+1}} + D_{queue_n} + D_{proc_n}) + \sum_{m=0}^{r}(D_{prop_{m\to m+1}} + D_{queue_m} + D_{proc_m}) + D_{stack} + D_{app} \tag{3.1}$$

This study was possible because both Atlas and MONROE provide latency measurements between vantage points with known location toward targets with known location: Atlas through its anchoring measurements, and MONROE via periodic pings toward the MONROE collection infrastructure. This allows us to compare distance derived from eq. (3.2) with actual (reported) distance:

$$dist < \frac{\sum_{n=0}^{f} D_{prop_{n\to n+1}} + \sum_{m=0}^{r} D_{prop_{m\to m+1}}}{2} \times c_{internet} \tag{3.2}$$

Here $c_{internet}$ is the "speed of light in the Internet", specifically the speed of light in an idealized, single-link optical fiber network, of about $\frac{c}{1.4677}$[2], and $dist$ represents the idealized point-to-point distance, ignoring optical fiber routing and the curvature of the Earth.

Our results, detailed in section 3.3.1 and section 3.3.2, confirm the network operations rule of thumb that 1ms of RTT is 100km of distance, and the findings of previous studies that RTT measurement can provide location accuracy on the order of 30km to 100km. The sensitivity of RTT measurements for geoprivacy is therefore related to the minimum RTT represented by those measurements. However, RTT measurement is less accurate than IP geolocation using even the most basic, free databases. We therefore recommend care in dissemination of RTT measurement datasets in those cases where the datasets themselves are dominated by samples on the order of less than 10ms, and/or where one (but not both) IP addresses are anonymized.

We also consider the question of load telemetry, as detailed in section 3.3.3. Here we hold distance-related components of the RTT to be constant, and observe eq. (3.3):

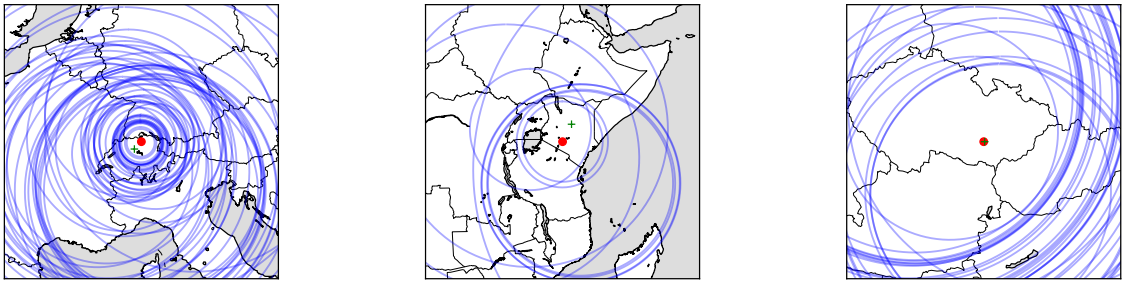
$$load \propto D_{stack} + D_{app} \tag{3.3}$$

3.3.1 Exclusion

We start by using Atlas anchoring measurements to attempt geolocation by exclusion, exploiting the inequality for $dist$, the observation that the RTT between two endpoints cannot be less than the speed of light in the medium of the Internet multiplied by twice the distance between those endpoints. We looked at all anchoring ICMP traceroute measurements on Monday 2 October 2017 to a set of 39 anchors, and filter out any measurements from probe-anchor pairs with reported locations less than 500m from each other[3]. We assume that each of our RIPE

[2] taken from the index of refraction in a datasheet for a typical optical fiber, see `http://ece466.groups.et.byu.net/notes/smf28.pdf`

[3] Here, the reasoning is that such pairs are either colocated in the same rack, or possible connected to the same local- or metropolitan-area network, and as such do not accurately reflect Internet RTT measurement.



(a) Glattbrugg, Switzerland. (b) Nairobi, Kenya. (c) Brno, Czechia.

Figure 15: Exclusion circles around selected anchors (red dot) and associated MaxMind Geolite City geolocation result (green cross)

Atlas Anchor targets is unicasted. This yields a total of 9.61 million individual measurements over 22,072 probe-anchor pairs. We then took the minimum end-to-end RTT measurement for each probe-anchor pair, taking this to be the best measurement for exclusion purposes.

We then draw exclusion circles corresponding to each probe's minimum RTT at that probe's location, and examine the intersection of these circles. We note that for 35 of the 39 anchors, intersection gives no additional information; i.e. the closest probe's exclusion circle is completely covered by that of the next closest probe. RTT location via exclusion is therefore largely a matter of luck of the location of the known vantage point. An illustration of this most common case is shown in Figure 15a. In the other cases, either the refinement to the exclusion area is insignificant, or the location estimate covers a large region with or without intersection. Figure 15b shows an example of this case; note here that both the location estimate and IP geolocation yield national-scale results

Though sometimes comparatively remote probes can refine each others' exclusion circles, in no case did we find such a refinement resulting in a reasonably accurate location estimate: the uncertainty in RTT simply grows too quickly with distance. Figure 15c illustrates this. Here, estimates from Prague and Vienna yield an area roughly the size of Czechia, but do exclude Prague.

When the IP address of the target is known, IP address geolocation can also be used to estimate its location. We therefore attempt to geolocate each anchor based on its IPv4 address in the freely-available MaxMind GeoLite City database[4], which we take as a worst-case IP geolocation result, noting that better IP geolocation databases will yield better results [20]. We compare the error between the geolocation result and the anchor's declared location with the uncertainty circle for each probe. Here we find that only 140 of 22079 of our anchor-probe pairs have less uncertainty than IP geolocation error, and only 14 of the 39 anchors, generally in areas with a very high probe density, have a measurement from at least one such probe. This further underscores the role of luck in vantage point selection.

We take the RIPE Atlas anchoring measurement dataset to be representative of Internet RTT measurements. Given that opportunities for location area reduction by intersection are not

[4] As retrieved from `https://stat.ripe.net` on 10 October 2017.

significant in this dataset, we now make a simplifying assumption that the best estimate for the location of an anchor is the center of the uncertainty circle of the probe with the lowest minimum RTT, and therefore that the error in the best estimate is simply the distance from that probe to the anchor. Median error in the Atlas dataset is 39km while the median IP geolocation error is 16km. We note that even though our methodology is far simpler than those described in the literature, it achieves comparable accuracy, underscoring the finding that skill (or luck) in vantage point placement is the dominant factor in accuracy in geolocation by exclusion using RTT measurements.

Atlas measurements are largely from residential or infrastructure networks toward infrastructure networks. Recent work by Bajpai et al. [7] shows our findings also to be applicable to the location of residential subscribers. This analysis of Atlas and SamKnows measurements of last-mile latency finds latency to depend on provider, technology, and point of presence, with median (two-way) latencies per provider between 5 and 20ms. Last-mile latency is therefore responsible, on its own, for an exclusion radius between about 500km and 2000km. We therefore take the location of residential endpoints to be more challenging than location by exclusion of Atlas anchors.

Note that while landmark selection is a challenge for active measurement, when RTT information is observed passively, e.g. during a transport or application-layer handshake, or using passive TCP measurement [41], the increased flatness of the Internet topology [4] implies that there is a decent chance to observe active communications between a client and a nearby content server.

3.3.2 Linear Distance Modeling

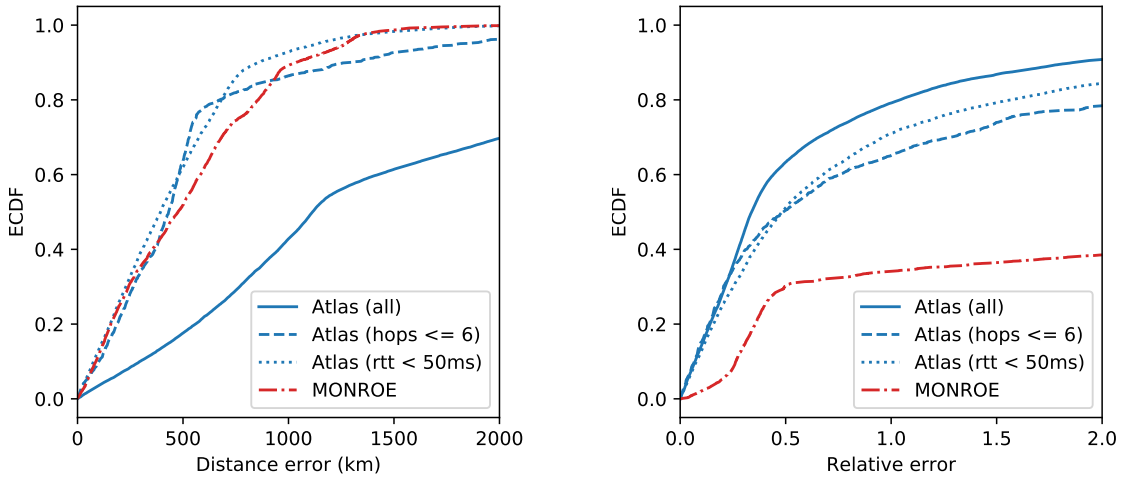
We also attempted trilateration through the creation of a linear model relating RTT to distance; i.e. $dist_{est} = f(RTT_{obs})$, based both on Atlas and MONROE measurements. The linear models we derived from our measurements (Atlas: $RTT = 0.0190 \times dist + 22.317$ with $r = 0.86$; MONROE[5]: $RTT = 0.0154 \times dist + 37.0735$ with $r = 0.78$, for RTT in milliseconds and distance in kilometers) are too imprecise to use as a basis for trilateration, with variance and last-mile latency making distance estimation even less feasible on mobile networks.

However, in examining the absolute (fig. 16a) and relative (fig. 16b) error in these models, a guideline for using RTT measurement for location estimation emerges. Restricting RTT data in ways that are possible using only simple inference or measurement can lead to better models with less error. Figure 16 also shows error results for models based on subsets of the Atlas RTT data, considering only pairs with a minimum RTT less than 50 ms, or considering only short paths (with less than 6 hops).

3.3.3 Remote Load Telemetry

RTT measurement is of interest to in-network operations precisely because it can be used to gain insight into the functioning and malfunctioning of network devices. An unusual D_{queue_n} or D_{proc_n} is often indicative of a fault to be corrected. This is the basic insight behind the mea-

[5]MONROE nodes provide GPS metadata for mobile nodes for location ground truth. We split MONROE data from 1 September 2017 into 5 minute bins (300 pings) and associated the geographic average GPS location with the minimum RTT in each bin to yield 3,863 samples from 45 nodes.



(a) Absolute.

(b) Relative.

Figure 16: Distance error for linear models

surement of in-network buffering performed by Netalyzr [22] and other measurement platforms, and has been used more recently in the inference of congestion at network interconnects [28]. Load on the endpoint can also be visible in RTT measurements, as shown by Holterbach et al [21], who showed in a study of the load dependent accuracy of the Atlas platform that several milliseconds of RTT error could be induced and measured by varying the load on RIPE Atlas probes.

This utility, however, has a flipside, as it necessarily exposes information about D_{queue_n} or D_{proc_n} to any device on path which can use active or passive measurement of RTT. More precisely, we now consider a threat model where the attacker knows an IP address associated with a given target, and wants to estimate activity on that target's network. Here, the attacker can leverage four assumptions about common characteristics of residential access networks to successfully determine activity on a residential customer's network:

- The access link is usually the bottleneck link for residential access, so latency variation is due to D_{queue} on the two directions of this link.
- Residential access links are frequently "bufferbloated" [19, 22]; i.e., modems have overdimensioned buffers that lead to high D_{queue} under load.
- In many markets, a single customer has a single public IP address at a single point in time, so an ICMP ping to a given address will traverse the access link. Note that this assumption does not hold when carrier-grade NAT is used to conserve IP addresses [29].
- ICMP packets, if not blocked, will generally share a queue with other packets, and can therefore be used to measure D_{queue} induced by other traffic.

In other words, remote buffer measurement is possible, and can be used to infer activity on residential networks. The basic methodology consists of repeatedly pinging the public IP ad-

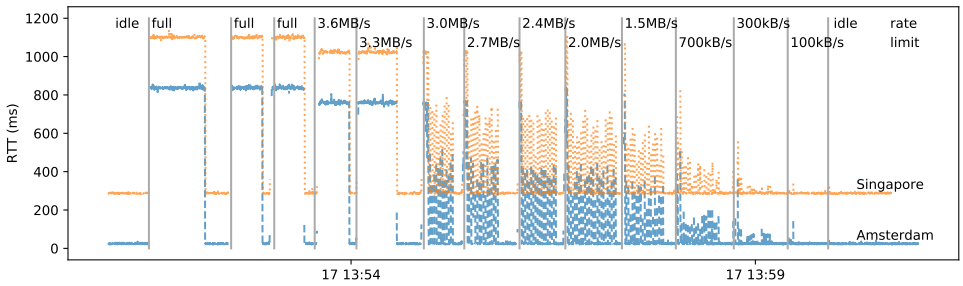


Figure 17: RTT time series from Amsterdam and Singapore toward a typical residential cable access network in Zürich with a 30Mbit download link, during downloads from that network with various rate limits.

dress of a target, and comparing the series of measured RTTs with the minumum. We ran a proof of concept of this methodology, pinging a residential IP address belonging to an author of this study in Zurich ten times a second from vantage points in Amsterdam and Singapore, while subjecting the inbound network link to varying load through TCP downloads using `curl` at various rate limits. The results are shown in figure 17. Idle and active periods are clearly visible due to RTT change from baseline from both the near and far vantage point, even down to a 300kB/s downstream rate limit, one tenth of the capacity of the link. This author's home access network is also home to a RIPE Atlas probe. Indeed, examining the 5-minute RTT series from second-hop latency measurements from this probe on a different day show clear diurnal peaks in maximum RTT measured during morning and evening weekday network activity, indicating that load telemetry is even possible with very limited RTT data.

We made an automated version of this load telemetry measurement available as an online tool[6]. Analysis of data collected during 2018[7], taken fom 66 distinct access networks, both terrestrial and mobile, remote load telemetry is possible on 9 (14%). Of the others, 33 (50%) block ICMP, and 24 (33%) allow pings, but show no evidence of load-linked RTT, due to small buffers and/or aggregation behind a single IP address.

[6] `https://pingme.pto.mami-project.eu`; source code at `https://github.com/mami-project/pingme`

[7] Note these figures update those from the published paper; see the MAPRG presentation at `https://datatracker.ietf.org/meeting/102/materials/slides-102-maprg-is-bufferbloat-a-privacy-issue-brian-trammell-00`

4 Tool Development and Maintenance

During this reporting period development work continued on the PATHspider active measurement tool and the Path Transparenct Observatory (PTO), and these tools were transitioned to maintenance after submission of D1.2. PATHspider and the PTO have grown to complement each other: PATHspider now exports path and condition information directly encoded in its raw data output, to reduce normalization effort at the PTO, and includes an integrated tool for generating metadata and uploading results to a PTO instance.

4.1 PATHspider

Since last reporting in D1.2, the PATHspider test suite has been further expanded to provide self-verification of vantage points used in measurement campaigns. This work has uncovered bugs in the underlying libraries that are in use, python-libtrace and Scapy, and these have been reported and fixed upstream so that others building tools with these libraries may also benefit from their discovery.

Further, automated metadata generation and upload of PATHspider results to the PTO can now be performed with PATHspider offering tight integration.

Since the last release we have added support for dynamic test configurations where the number of tests to run can be specified at runtime. For example, when testing for protocol-dependent failures when using DiffServ codepoints, multiple codepoints can now be tested in a single run.

We have also added a new plugin to test for the negotiation of several TCP options: timestamps, window scaling and selective acknowledgements.

Lastly, we have added support for two new deployment options[1]. The first of these is virtualenv where we now document how to install PATHspider and its dependencies as a non-privileged user which can be useful for application layer tests not requiring raw socket access, or simply to keep your development system tidy. The second is using cloud-init, a popular system used by cloud orchestration platforms.

4.1.1 Software Maintenance

Software maintenance of PATHspider has included fixing issues that arise from external factors, such as library dependencies.

Throughout the project, Scapy has presented issues due to our use of version 3 of the Python language for implementation of PATHspider. Originally the developers of Scapy did not support Python 3 and so we needed to use a fork of the library. Over time, this fork grew apart from the more active upstream development and bugs went unfixed. The upstream developers then did decide to support Python 3 and that support is now mature. In the next release of PATHspider, 2.1.0, we will depend on the upstream Scapy distribution which includes many fixes for issues that we've come across during PATHspider's development.

[1] The documentation for these options can be found at: `https://pathspider.readthedocs.io/en/latest/`.

4.1.2 Lasting Influence

At the MAMI Active Measurement Hackathon held at the University of Aberdeen in December 2018, researchers from Tor Project met with researchers from MAMI to exchange experiences from working on their Internet measurements. Tor Project uses Internet measurement in their sub-project: the Open Observatory of Network Interference[2] (OONI). This project aims to measure Internet censorship on a global scale.

During this hackathon three of the tests performed by PATHspider for protocol-dependent connectivity failure were incorporated into the test specifications[3] for OONI to be implemented in a future release of their measurement tool. Support for the tests was also added to their underlying library known as measurement-kit.

Additionally, experimental support for uploading results to the OONI Observatory was implemented. This would allow for the OONI measurements, where there are thousands of vantage points but each only performs tests with a small number of targets, to be compared with PATHspider measurements where there are less vantage points but they are assumed to be hosted in "clean" networks and can test greater numbers of targets.

The PATHspider website will continue to run at `https://pathspider.net/` beyond the end of the project. Minor issues reported via GitHub along with any patches will be reviewed on a best-effort basis with new releases being made as appropriate. Other projects, such as Tor Metrics, have expressed interest in using PATHspider for active measurements [25].

4.2 Path Transparency Observatory

The final design of the Path Transparency Observatory (PTO) is covered in Deliverable 1.2. Since the completion of the design effort, work has continued on improving the public front-end to the PTO, maintenance and refinement of the PTO implementation, and planning for continued operation of the PTO after the end of the project.

4.2.1 PTO Implementation and Deployment Notes

The current PTO instance is is implemented in the Go programming language, and is split among four repositories in GitHub:

- `https://github.com/mami-project/pto3-go` contains the PTO core.
- `https://github.com/mami-project/pto3-ecn` contains normalizers for ECN data from previous ECN campaigns, as well as a generalized normalizer for PATHspider data.
- `https://github.com/mami-project/pto3-trace` contains normalizers for tracebox data.
- `https://github.com/mami-project/pto3-web` contains a basic web front-end for the PTO.

[2] The homepage for this project can be found at: `https://ooni.torproject.org/`.

[3] The specifications can be found at: `https://github.com/ooni/spec/tree/master/techniques`. The PATHspider-derived tests are labelled as `tq-031`, `tq-032`, and `tq-033`.

Since the report in D1.2, we have made the following maintenance changes to the PTO:

- Revamped the web front-end in the repository at `https://github.com/mami-project/pto3-web`.
- Added the ability to serve static content, allowing an instance of the PTO to serve both the API as well as a web front-end for interacting with the front-end. Our instance uses this feature to present the revamped front-end.
- Added the `autonorm` utility, which automatically normalizes all the files in a specified set of raw data campaigns not yet stored as observation sets in the database.
- Added an optimized, parallelized framework for writing normalizers in Go, and reimplemented our normalizers in terms of this framework.
- Changed the permission system slightly to make it possible to grant finer-grained permissions.

Our current deployment of the PTO is on a Linux server at ETH Zrich, with 20 cores, 256G of RAM, and 32T of disk, which it shares with a few other measurement research tasks. The observation database is stored in PostgreSQL. Our current data load includes about 1.5T of raw files (many of which are currently uncompressed) and the PostgreSQL observation database currently takes about 250G of storage.

We continue to load and normalize data into the PTO; current efforts focus on DSCP studies in 2017 and MSS studies in 2018.

4.2.2 Post-Project Maintenance of the PTO

After the end of the MAMI project, stewardship of the PTO will transition to MAMI partner ZHAW, as the hardware the PTO currently runs on will transition to use by another group at ETH. A machine with roughly equivalent hardware specifications has been purchased and installed on the ZHAW premises in Winterthur, and will continue to be operated and maintained by ZHAW past the end of the project. This maintenance will not consist of the addition of new features to the PTO.

As the core concept of the PTO allows the regeneration of observations from the raw data given the provenance information in the PTO itself, our migration of data from ETH to ZHAW consists of copying the raw data, then re-running normalization and analysis on the ZHAW instance.

The archival instance of the PTO will be accessible at `https://pto.mami-project.eu`.

5 Conclusion

This deliverable concludes the work of the MAMI project's WP1, covering our measurement activities. Measurement within the MAMI project has been both more extensive in scope and slightly smaller in scale than envisioned at the project's inception. During this period, we extended measurements to the MONROE testbed, and transitioned some measurements to longer-term collection for longitudinal study.

Our original vision was limited to path transparency measurement – determining, at a relatively low level, the extent to which a given transport protocol or transport feature can be safely deployed in the Internet. While we have answered this question for the Internet core and, to a limited extent, for mobile access networks for a variety of transport features, our measurements have had a wider impact on the project's goals and on transport stack evolution in general.

Internet measurement work performed by the MAMI project has had the following results:

- Confirmation of the viability of UDP as a substrate for transport layer evolution, whether through a path layer protocol such as PLUS, or directly as in the case of QUIC (see section 3.2 of D1.1).
- Review of the privacy implications of Internet round-trip time (RTT) data, based on RIPE Atlas and MONROE data, confirming that explicit exposure of RTT data (as with the PLUS PSN/PSE facility, or with the QUIC spin bit advocated by the project) is fundamentally safe to deploy in the Internet with respect to geoprivacy (see section 3.3).
- A survey of the types and prevalences of header manipulation in the Internet on a per-hop basis, as input to efforts to build a middlebox bestiary and simulator in WP2 (see section 3.5 of D1.1).
- Determination that ECN is safe to deploy by default on the client side, which in turn provided support for Apple's decision to do so, leading to the first significant movement toward enabling ECN on a large proportion of Internet flows since its definition two decades ago; as well as further study of specific impairments to ECN (see section 3.1 of D1.1, as well as section 2.2).
- Understanding that ECN impairment is a good proxy metric for other forms of TCP and IP manipulation, such as TCP-interfering censorship activities and treatment of DSCP+ECN according to obsolete semantics (section 2.2).
- Confirmation of the deployability of UDP options and the ECN++ extension (see section 2.2.2).
- Improvement of the PATHspider active measurement tool for path transparency, adding the ability to measure a larger set of protocol features, as well as improving the tool's performance and extensibility (see section 2.1.1 of D1.1, as well as section 4.1).
- Creation of a path transparency observatory, both to store our measurement data and as a proof of concept of the applicability of normalization and metadata primacy in general to Internet measurement studies (see D1.2 in its entirety, as well as section 4.2).

These results, including those details of measurement work performed over the past two years covered in the deliverable, represent a key contribution of the project toward the wider (and ongoing) goal of evolving the Internet's transport layer to meet present and future challenges.

References

[1] iOS 11: iOS Security Guide. https://www.apple.com/business/docs/iOS_Security_Guide.pdf [Online; accessed 06-March-2018].

[2] WhatsApp Encryption Overview. https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf [Online; accessed 06- March-2018].

[3] *The Concise Encyclopedia of Statistics*, chapter Kolmogorov–Smirnov Test, pages 283–287. Springer New York, New York, NY, 2008.

[4] B. Ager, N. Chatzis, A. Feldmann, N. Sarrar, S. Uhlig, and W. Willinger. Anatomy of a Large European IXP. In *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '12, pages 163–174, Helsinki, Finland, 2012. ACM.

[5] P. Almquist. Type of service in the internet protocol suite. RFC 1349, RFC Editor, July 1992. http://www.rfc-editor.org/rfc/rfc1349.txt.

[6] C. Aoun and S. Sen. Identifying intra-realm calls and avoiding media tromboning. Internet-Draft draft-aoun-midcom-intrarealmcalls, Feb. 2002. https://tools.ietf.org/html/draft-aoun-midcom-intrarealmcalls-00.

[7] V. Bajpai, S. J. Eravuchira, and J. Schönwälder. Dissecting last-mile latency characteristics. *SIGCOMM Comput. Commun. Rev.*, 47(5):25–34, Oct. 2017.

[8] F. Baker. Requirements for IP version. RFC 1812, Internet Engineering Task Force, June 1995.

[9] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman. The Secure Real-time Transport Protocol (SRTP). RFC 3711, RFC Editor, March 2004. http://www.rfc-editor.org/rfc/rfc3711.txt.

[10] R. Birke, M. Mellia, M. Petracca, and D. Rossi. Experiences of VoIP traffic monitoring in a commercial ISP. *International Journal of Network Management*, 20(5):339–359, 2010.

[11] S. Blake, D. L. Black, M. A. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. RFC 2475, RFC Editor, December 1998. http://www.rfc-editor.org/rfc/rfc2475.txt.

[12] A. Custura, G. Fairhurst, and I. Learmonth. Exploring usable Path MTU in the Internet. In *Network Traffic Measurement and Analysis Conference, TMA 2018*. IFIP Open Digital Library, 4 2018.

[13] A. Custura, R. Secchi, and G. Fairhurst. "Exploring DSCP modification pathologies in the Internet". *Computer Communications*, 127:86 – 94, 2018.

[14] A. Custura, A. Venne, and G. Fairhurst. Exploring DSCP modification pathologies in mobile edge networks. In *2017 Network Traffic Measurement and Analysis Conference (TMA)*, pages 1–6, June 2017.

[15] G. Detal, B. Hesmans, O. Bonaventure, Y. Vanaubel, and B. Donnet. Revealing middlebox interference with tracebox. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, pages 1–8, Barcelona, Spain, 2013. ACM.

[16] H. Ding and M. Rabinovich. TCP stretch acknowledgements and timestamps: Findings and implications for passive RTT measurement. *SIGCOMM Comput. Commun. Rev.*, 45(3):20–27, July 2015.

[17] G. Fairhurst, T. Jones, and R. Zullo. Checksum compensation options for UDP Options. Internet-Draft draft-fairhurst-udp-options-cco-00, IETF Secretariat, October 2018. http://www.ietf.org/internet-drafts/draft-fairhurst-udp-options-cco-00.txt.

[18] A. Filastò and J. Appelbaum. OONI: Open Observatory of Network Interference. In *USENIX FOCI 2012*.

[19] J. Gettys and K. Nichols. Bufferbloat: Dark buffers in the Internet. *Queue*, 9(11):40:40–40:54, Nov. 2011.

[20] M. Gharaibeh, A. Shah, B. Huffaker, H. Zhang, R. Ensafi, and C. Papadopoulos. A Look at Router Geolocation in Public and Commercial Databases. In *Internet Measurement Conference (IMC)*, Nov 2017.

[21] T. Holterbach, C. Pelsser, R. Bush, and L. Vanbever. Quantifying interference between measurements on the RIPE Atlas platform. In *Proceedings of the 2015 Internet Measurement Conference*, IMC '15, pages 437–443, Tokyo, Japan, 2015. ACM.

[22] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson. Netalyzr: Illuminating the edge network. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC '10, pages 246–259, Melbourne, Australia, 2010.

[23] M. Kühlewind, M. Walter, I. Learmonth, and B. Trammell. Tracing Internet path transparency. In *Traffic Measurement and Analysis Conference*, Vienna, Austria, 2018.

[24] A. Kuzmanovic, A. Mondal, S. Floyd, and K. Ramakrishnan. Adding Explicit Congestion Notification (ECN) Capability to TCP's SYN/ACK Packets. RFC 5562, Internet Engineering Task Force, June 2009.

[25] I. R. Learmonth and K. Loesing. Towards modernising data collection and archive for the Tor network. Technical Report 2018-12-001, The Tor Project, December 2018. `https://research.torproject.org/techreports/modern-collector-2018-12-19.pdf`.

[26] K. Loesing, S. J. Murdoch, and R. Dingledine. A Case Study on Measuring Statistical Data in the Tor Anonymity Network. In *Proceedings of the Workshop on Ethics in Computer Security Research (WECSR 2010)*, LNCS, pages 203–215. Springer Berlin Heidelberg, Jan. 2010.

[27] M. Luckie. Scamper: A Scalable and Extensible Packet Prober for Active Measurement of the Internet. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement (IMC '10)*, pages 239–245, New York, NY, USA, 2010. ACM.

[28] M. Luckie, A. Dhamdhere, D. Clark, B. Huffaker, and k. claffy. Challenges in inferring Internet interdomain congestion. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, IMC '14, pages 15–22, Vancouver, BC, Canada, 2014.

[29] A. Lutu, M. Bagnulo, A. Dhamdhere, and k. claffy. NAT Revelio: Detecting NAT444 in the ISP. In *Passive and Active Network Measurement Workshop (PAM)*, Mar 2016.

[30] A. M. Mandalari, A. Lutu, B. Briscoe, M. Bagnulo, and O. Alay. Measuring ECN++: Good news for ++, bad news for ECN over mobile. *IEEE Communications Magazine*, 56(3):180–186, March 2018.

[31] A. M. Mandalari, A. Lutu, A. Custura, A. Safari Khatouni, O. Alay, M. Bagnulo, V. Bajpai, A. Brunstrom, J. Ott, M. Mellia, and G. Fairhurst. Experience: Implications of roaming in Europe. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, MobiCom '18, pages 179–189, 2018.

[32] A. Medina, M. Allman, and S. Floyd. Measuring interactions between transport protocols and middleboxes. In *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement (IMC '04)*, pages 336–341, New York, NY, USA, 2004. ACM.

[33] F. Michelinakis, H. Doroud, A. Razaghpanah, A. Lutu, N. Vallina-Rodriguez, P. Gill, and J. Widmer. The cloud that runs the mobile Internet: A measurement study of mobile cloud services. In *IEEE INFOCOM 2018*, pages 1619–1627, April 2018.

[34] J. Pahdye and S. Floyd. On inferring TCP behavior. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '01, pages 287–298, New York, NY, USA, 2001. ACM.

[35] J. Postel. Internet Control Message Protocol. RFC 792, Internet Engineering Task Force, September 1981.

[36] K. Ramakrishnan, S. Floyd, and D. Black. The addition of explicit congestion notification (ECN) to IP. RFC 3168, RFC Editor, September 2001. http://www.rfc-editor.org/rfc/rfc3168.txt.

[37] A. Razaghpanah, N. Vallina-Rodriguez, S. Sundaresan, C. Kreibich, P. Gill, M. Allman, and V. Paxson. Haystack: In situ mobile traffic analysis in user space. *CoRR*, abs/1510.01419, 2015.

[38] E. Rosati. 2015: the year of blocking injunctions? *Journal of Intellectual Property Law & Practice*, 10(3):147, 2015.

[39] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing. Session Traversal Utilities for NAT (STUN). RFC 5389, RFC Editor, October 2008. http://www.rfc-editor.org/rfc/rfc5389.txt.

[40] P. Srisuresh, B. Ford, S. Sivakumar, and S. Guhg. NAT behavioral requirements for ICMP. RFC 5508, Internet Engineering Task Force, April 2009.

[41] S. D. Strowes. Passively measuring TCP round-trip times. *Commun. ACM*, 56(10):57–64, Oct. 2013.

[42] J. Touch. Transport options for UDP. Internet-Draft draft-ietf-tsvwg-udp-options-05, IETF Secretariat, July 2018. http://www.ietf.org/internet-drafts/draft-ietf-tsvwg-udp-options-05.txt.

[43] B. Trammell and M. Kühlewind. Revisiting the privacy implications of two-way Internet latency data. In *Passive and Active Measurement*, pages 73–84, Cham, 2018. Springer International Publishing.

[44] R. L. Wasserstein and N. A. Lazar. The ASA's statement on p-values: Context, process, and purpose. *The American Statistician*, 70(2):129–133, 2016.

[45] R. Zullo, A. Pescapé, K. Edeline, and B. Donnet. Hic sunt NATs: Uncovering address translation with a smart traceroute. In *Proc. IEEE/IFIP Workshop on Mobile Network Measurement (MNM)*, June 2017.