Measuring ECN++
Good News for ++, Bad News for ECN over Mobile

Anna Maria Mandalari
University Carlos III of Madrid
Email: amandala@it.uc3m.es

Andra Lutu
Simula Research Laboratory
Email: andra@simula.no

Bob Briscoe
Simula Research Laboratory
Email: bob@simula.no

Marcelo Bagnulo
University Carlos III of Madrid
Email: marcelo@it.uc3m.es

Özgü Alay
Simula Research Laboratory
Email: ozgu@simula.no

***Abstract*—After Explicit Congestion Notification (ECN) was first added to the Internet Protocol in 2001, it was hit by a succession of deployment problems. Studies in recent years have concluded that path traversal of ECN has become close to universal. In this article, we test whether the performance enhancement called ECN++ would face a similar deployment struggle to base ECN. For this, we assess the feasibility of ECN++ deployment over mobile as well as fixed networks. In the process, we discovered bad news for the base ECN protocol: contrary to accepted beliefs, more than half the mobile carriers we tested wipe the ECN field at the first upstream hop. All packets still get through and congestion control still functions, just without the benefits of ECN. This throws into question whether previous studies used representative vantage points. This article also reports the good news that, wherever ECN gets through, we found no deployment problems for the '++' enhancement to ECN. The article includes the results of other in-depth tests that check whether servers that claim to support ECN actually respond correctly to explicit congestion feedback.**

Those interested can access the raw measurement data online.

INTRODUCTION

Explicit Congestion Notification (ECN) [1] is a way to mark packets to indicate that the capacity of a link is approaching exhaustion. ECN was standardized as a straight replacement for loss signals, but increasingly ECN is also being recognized as critical for low delay. Despite early evidence of its positive impact [2], a succession of unfortunate incidents stalled ECN deployment for 15 years: some firewalls treated all TCP/ECN connection attempts as port scanning attacks [3]; TCP/ECN connection attempts were mistakenly discarded by certain home router models and one popular model crashed; and when routinely wiping the DiffServ field between networks, a bug also wiped the IP/ECN field.

In recent years, ECN adoption on end-systems has accelerated, with the majority of servers supporting ECN [4]. Currently, motivated by an increasing need to reduce queuing delay in modern networks, solutions such as DCTCP [5] in data centers or L4S[1] in public networks are further driving interest in ECN. In 2016, Apple enabled client negotiation of TCP ECN in a random subset of iOS and macOS devices. In March 2017 they presented a measurement study[2] showing almost universal path traversal support for ECN. Other recent measurement studies [4], [6] report consistent findings.

The original ECN specification for TCP [1] prohibited retransmitted packets and control packets from using ECN. This was unfortunate, because TCP performance, particularly short-flow completion time, is much more sensitive to loss of certain control packets, such as the SYN at the start of a connection [7].

A new performance enhancement called ECN++[3] 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.

This study sets out to measure how much existing networks and servers would mangle or block the ECN++ updates to TCP/IP. We test ECN and ECN++ support in both fixed and mobile networks. In particular, we 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.

The article offers the following main contributions:

Bad news: More than half of the mobile carriers we tested bleach (clear) the ECN field at the first upstream IP hop.

[1] https://tools.ietf.org/html/draft-briscoe-tsvwg-l4s-arch (Accessed on 2017-11-27).

[2] https://www.ietf.org/proceedings/98/slides/slides-98-maprg-tcp-ecn-experience-with-enabling-ecn-on-the-internet-padma-bhooma-00.pdf (Accessed on 2017-11-27).

[3] https://tools.ietf.org/html/draft-bagnulo-tcpm-generalized-ecn-04 (Accessed on 2017-11-27).

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.

ECN Background

ECN [1] can be deployed in any buffer in any network element. A bottleneck buffer will mark the standardized ECN field increasingly often as it detects early signs of increasing load. These ECN markings then propagate to all the hosts receiving data through the buffer. In turn, the receivers feed back these signals to the respective hosts that are sending them data. The aim is that data senders will use this feedback to regulate the load on the bottleneck buffer.

ECN is intimately tied to Active Queue Management (AQM) technology. With AQM, buffers drop a small proportion of packets at the first sign of queue growth to fool TCP senders into thinking the buffer is full and backing off. ECN allows AQM to keep the queue short without dropping packets.

ECN in IP

Codepoint	IP/ECN field	Meaning
not-ECT	00	Not ECN-capable transport
ECT(0)	10	ECN-capable transport
ECT(1)	01	
CE	11	Congestion experienced

TABLE I: The ECN field in IP.

The ECN field has two bits, hence it supports the 4 codepoints in Table I. Before using ECN, the standard requires a data sender to have checked that the receiver has logic for feeding back ECN markings. Each transport protocol does this differently. If at least one end does not support ECN, there is not an 'ECN-capable transport' (ECT) so the sender must set not-ECT in all packets, which is also what legacy hosts send by default. According to [1], ECT(0) and ECT(1) are equivalent and either can be set by the sender to signify the packet is ECT, which means "ECN-capable but not ECN-marked".

Network elements have to check the ECN field before using ECN. If it is not-ECT (00), they must use drop rather than the CE codepoint to indicate congestion. As subsequent packets arrive where the ECN field is nonzero, an ECN-capable buffer can set the CE codepoint (11) increasing the probability to indicate increasing congestion.

ECN in TCP

Three flags in the main TCP header are assigned to ECN. They are called NS, CWR and ECE for Nonce Sum, Congestion Window Reduced and Echo Congestion Experienced. These names are not descriptive of their usage in most circumstances, so this article will represent them as a 3-bit codepoint, as shown in the 'flags' column of Table II.

Table II succinctly supports the following explanations of standard ECN (this section), Accurate ECN and ECN++. For the present explanation of standard ECN, ignore the AccECN rows at the bottom and the ECN++ column on the right. Also, the NS TCP flag is not used for standard ECN (always zero).

For the example of an ECN client contacting an ECN server, the table should be read down the rows as follows. The client sends an 'ECN setup SYN' with TCP ECN flags 011 and the server responds with an 'ECN setup SYN/ACK' with TCP ECN flags 001. Only 00 is allowed in the IP ECN field of both these handshake packets, as shown in the 'Allowed IP ECN' column.

Once the TCP connection is established, ECN feedback proceeds independently in either direction for the two half-connections. Regular data packets have 000 in the TCP ECN flags and, according to the IP ECN column, the data sender can set either ECT codepoint (XX means theoretically any of the 4 values in Table I are allowed). Then, if congestion is experienced along the path, the buffer will set the CE codepoint on some of these packets.

The data receiver will then feed this back by setting the Echo CE (ECE) flag in the TCP header of acknowledgement (ACK). The data sender then confirms receipt of a new ECE flag by setting the CWR flag on the next segment, which confirms that it has reduced its congestion window (cwnd). For reliability against loss of an ECE message, the data receiver is required to set the ECE flag repeatedly on every ACK until it receives the CWR.

Accurate ECN

Because the original ECN scheme repeats the ECE flag for a whole round trip for reliability, more than one CE mark within a round trip cannot induce any more feedback. Since 2010, it has become well-known [5] that queuing delay can be reduced to extremely low levels if more accurate feedback gives the extent, not just the existence, of CE-marking. AccECN[5] adds more accurate ECN feedback to TCP. We introduce it in our testing, but we do not expect to find it in the wild yet because standardization is not yet complete.

The handshake for a client and server supporting AccECN feedback can be seen in the AccECN setup SYN and SYN/ACK rows of Table II. We will not step through the table again, except to highlight differences. The 110 combination of TCP

[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.

[5] https://tools.ietf.org/html/draft-ietf-tcpm-accurate-ecn-05 (Accessed on 2017-11-27).

Feedback mode	Packet	TCP ECN flags (NS:CWR:ECE)		Allowed IP ECN	
		Description	flags	ECN	ECN++
Non-ECN	All	ECN disabled	000	00	00
ECN	SYN	ECN setup SYN	011	00	00
	SYN/ACK	ECN setup SYN/ACK	001	00	XX
	Data	Regular	000	XX	XX
		Echo CE	001	XX	XX
		CWnd Reduced	010	XX	XX
	Control & RTX	Same as data packet		00	XX
AccECN	SYN	AccECN setup SYN	111	00	XX
	SYN/ACK	AccECN setup SYN/ACK	010	00	XX
		—Ditto— with CE Echo	110	00	XX
	Data	CE counter	XXX	XX	XX
	Control & RTX			00	XX

TABLE II: Allowed IP ECN field for all types of TCP packet in all three feedback modes. X means 0 or 1.

ECN flags allows feedback on the SYN/ACK in the event that the SYN arrives at the server with a CE mark, otherwise the SYN/ACK uses 010. This enables the SYN to be ECN-capable (see ECN++ below). Once an AccECN connection is established, an AccECN data receiver uses the ECN TCP flags as a 3-bit counter to continually repeat feedback of a count of how many CE-marked packets it has received over the half-connection.

ECN++

When ECN was first standardized, SYNs and SYN/ACKs were precluded from being ECN-capable at the IP layer. IP/ECN was similarly prohibited for pure ACKs, window probes, and retransmissions (termed ‘Control and RTX’ in Table II). In 2005, the IETF sanctioned an experiment allowing IP/ECN on the SYN/ACK, termed ECN+ [7].

The ECN++ proposal is more ambitious than ECN+. It has found ways to safely allow each type of TCP control packet or retransmission to use IP/ECN, including FIN (finish) and RST (reset) packets as seen in the rightmost column of Table II. ECN++ can be used with either standard ECN feedback or AccECN feedback. Except, the SYN can only be ECN-capable in the IP header if it requests AccECN feedback in the TCP flags (111). This is because only AccECN provides space in the SYN/ACK for feedback in the event that the SYN gets CE-marked.[6]

RELATED WORK

The extent to which the Internet deploys and supports ECN has been a topic visited repeatedly over the past 15 years [4], [6], [8].

Measurements of server-side support usually focus on the population of web servers as ranked by Alexa. While quantifying ECN server-side support in 2004, Medina et al [8] found that as little as 2.1percent of the servers tested supported ECN. Ten years later, Trammell et al [4] report an acceleration in the deployment of ECN-capable servers, finding 56.17percent of ECN-capable servers. In this article, we corroborate the acceleration of ECN deployment for wired networks and Alexa servers, and we extend the study to mobile networks and ECN++.

Regarding support of ECN by the network elements on the end-to-end path, in 2013, Kühlewind et al [6] tested 22,487 hosts and reported that in 0.9percent of cases ECN was not usable due to middleboxes along the path. Later, in 2015, Trammell et al [4] finds that when testing 326,743 hosts capable of negotiating ECN, for 0.02percent of them (107 hosts) a device on the path mangles the TCP/ECN flags. We note that all these above-mentioned efforts report on measurements performed mostly in fixed networks. Only recently, in March 2017, Apple reported 100percent positive results after testing from vantage points connected to ‘a few’ mobile carriers[7], which on further investigation meant three carriers.

EXPERIMENTS

The goal of our experiments is to test how the ECN++ modifications to TCP/IP just described above would be treated in the current Internet, particularly over mobile networks. In order to do that, we designed two series of experiments. The first series of tests explores how the different ECN++ related fields are treated by the currently deployed base of network elements and servers. The second series of tests measures how the congestion control algorithms running in deployed servers react to ECN congestion signals. In both cases, we design our experiments to measure how equivalent non-ECN and ECN packets are treated in order to compare them with the ECN++ measurement results.

ECN++ Support

The experiments test support for ECN++ exchange TCP control packets and pure ACKs with different values in the ECN-related fields and check how those packets are treated by the network and by servers. We next describe the tests performed for each type of packet.

[6] If the server does not support AccECN at all, for safety the client has to behave as if it had received feedback of a CE on the SYN.

[7] https://www.ietf.org/proceedings/98/slides/slides-98-maprg-tcp-ecn-experience-with-enabling-ecn-on-the-internet-padma-bhooma-00.pdf (Accessed on 2017-11-27).

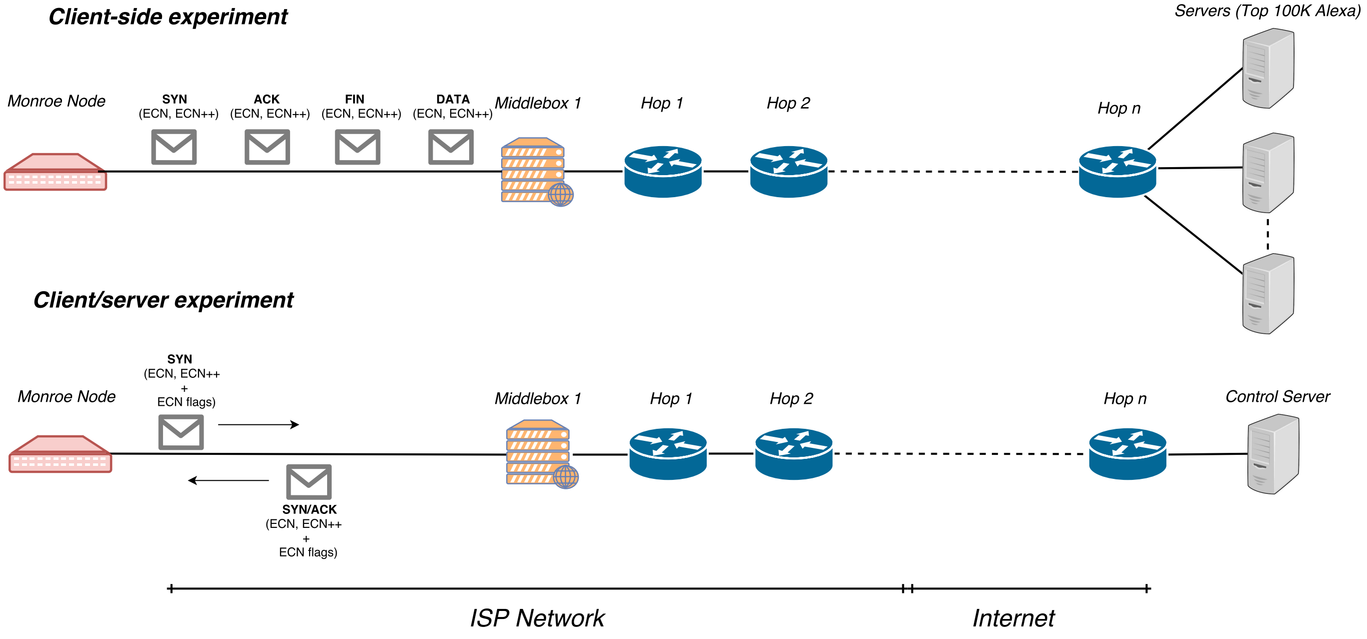

Fig. 1: ECN++ support: experimental setup.

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 does 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 AccECN 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 1 summarizes the operational setup for the client-side and the client/server experiments.

a) Client-side experiments: To observe how the ECN field in the IP header is treated by network elements we use Tracebox [9]. 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 enumerated in Table II. 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 [10] 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 [11] 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 we control to the Alexa top 100k servers with the different values for the ECN codepoints and TCP flags used by ECN and ECN++ as described in the rows corresponding to SYN packets in Table II.

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 [1] (Classic ECN) and how many servers use RFC5562 [7] (ECN+).

b) Client/server experiments: In these experiments we control the client and the server side. 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 described in the SYN and SYN/ACK rows of Table II. We measure whether the fields sent arrive at the other end, to check for middlebox interference (e.g. proxies).

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 included in the rows describing data packets, and 'Control and RTX' in Table II. In all cases, the client establishes a standard ECN TCP connection before sending the test packets.

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 [12]. 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+.

Experimental Setup

We perform all the experiments we designed and described above between January and May 2017. For the client-side experiments, we use both the MONROE platform (mobile carriers) and PlanetLab (wired service providers). The MONROE platform [13] is the first open source and open access hardware-based platform for independent, multihomed, large-scale experimentation in commercial mobile environments. The platform comprises hundreds of nodes multihomed to three of the mobile providers in each of 4 EU countries (Spain, Italy, Sweden, Norway). For the purpose of this study, we instrument 11 MONROE nodes distributed in all the countries with MONROE coverage. We measure the following mobile carriers: Vodafone (IT), TIM (IT), WIND (IT), Orange (ES), Yoigo (ES), Movistar (ES), Telia (SE), Telenor (SE), Three (SE), Telia (NO), Telenor (NO). To test wired providers, we instrument 54 PlanetLab [14] nodes distributed in 25 networks over 22 countries.

All experiments test two TCP ports, namely port 80 and port 443. We execute all the tests both from the MONROE and Planetlab nodes. In all cases, we test only IPv4 hosts. We run the experiments we describe for TCP SYN, TCP SYN/ACK and Data packets, pure ACKs and FINs towards the Alexa top 100k web servers and to our own servers. We run the experiments for the response to congestion signal towards the Alexa top 500k web servers, allowing us to also measure support for ECN in the wild. MONROE nodes resolve the Alexa top-N websites using Google's public DNS resolver; not the mobile carrier's default resolver. This enables us to build a vast and complex dataset, which we open to the community.[8] We collect information for a total of 26 million end-to-end communications, testing 6.5 million different paths.

FINDINGS

In this section, we describe the main findings of our measurement study.

Bad news: 7.5 out of 11 Mobile operators tested bleach ECN in outgoing packets. Using Tracebox in the MONROE nodes we find that 7 out of 11 Mobile operators bleach the ECN field in the IP header for all ECN codepoints in all packets going from the client to the Internet, for all types of packets tested (SYN, data packets, pure ACKS and FINs) and for both ports (80 and 443).

Table III shows the results for ECN and ECN++ bleaching. The operators bleaching ECN are: Yoigo (ES), Orange (ES), Vodafone (IT), TIM (IT), Wind (IT), Telia (NO), Telia (SE). In all the bleaching cases, we observe that the ECN field is cleared in the first hop (i.e., the mobile operator is clearing it in its radio access).

One mobile operator (Movistar (ES)) bleaches the IP/ECN field only for port 80 and not for port 443. Using the client/server experiments, we learn that Movistar is using a web proxy that does not support ECN.[9] We find that two other providers, TIM and Vodafone, also use a proxy on port 80.

For the subset of vantage points connected to the three mobile access networks that do not wipe the outgoing IP/ECN field (Telenor(NO), Telenor(SE) and 3 (SE)), we observe that 0.53percent of paths to the Alexa 100k servers encounter bleaching deeper into the network - always more than 5 hops away. We find similar results in the experiments using the PlanetLab clients (0.23percent bleaching), which is consistent with other measurements (see Related Work).

Regarding incoming packets (towards the client), through client/server experiments we find that all providers that do not

[8] You can access the dataset at http://www.it.uc3m.es/amandala/ecn++/ (Accessed on 2017-11-27).

[9] We observe that the TCP 3WHS is established between the client and the proxy first and when the HTTP GET is issued, the proxy establishes a connection with the server.

Access operator	Country	ECN bleaching (%)		ECN++ bleaching (%)		Proxy on port 80
		1st hop	>5th hop	1st hop	>5th hop	
Vodafone	Italy	100	100	100	100	YES
TIM	Italy	100	100	100	100	YES
WIND	Italy	100	100	100	100	NO
Orange	Spain	100	100	100	100	NO
Yoigo	Spain	100	100	100	100	NO
Movistar	Spain	100 (port 80) 0 (port 443)	100 (port 80) 0.53 (port 443)	100 (port 80) 0 (port 443)	100 (port 80) 0.53 (port 443)	YES
Telia	Sweden	100	100	100	100	NO
Telenor	Sweden	0	0.53	0	0.53	NO
3	Sweden	0	0.53	0	0.53	NO
Telia	Norway	100	100	100	100	NO
Telenor	Norway	0	0.53	0	0.53	NO

TABLE III: ECN and ECN++ bleaching for mobile operators.

use a proxy honour the value in the ECN field. We verified this by sending ECT marked packets from our servers to all the MONROE clients in the different ISPs. We cannot reliably conclude that there is not a fractionalpercentage of bleaching in the incoming direction, because the number of servers we have under our control limits the number of incoming paths we can test.

Given the majority of mobile carriers that bleach the IP/ECN codepoint, we complement the set of mobile networks we measure with 7 additional operators active in 4 countries other than the ones with MONROE coverage (i.e., Elisa (FI), DNA (FI), Vodafone (DE), Blau (DE), Vodafone (UK), O2 (UK), Base (BE)). For these, we run a limited set of measurements only against four servers (two that we control and two that run in the wild). We find that out of the seven mobile carriers, 3 of them (DNA (FI), Vodafone (UK), O2 (UK)) present the same IP/ECN bleaching behavior in the first hop as the other 7.5 carriers we identified in MONROE.

Good news: ECN fields do not cause packet drop. Through our client-side, end-to-end tests, we find that, irrespective of the TCP/ECN flags and IP/ECN codepoint combination, for the 6.5M paths tested, the packets sent using different TCP/ECN value combinations are acknowledged by the other end. We find this to be true for SYNs (we receive a SYN/ACK back), for data packets and FINs (we received ACKs back). It is true both for mobile and wired operators. This does not means the ECN related fields arrive at the other end unchanged though (we know it is not the case for the ECN field in the IP header, as per the previous finding). Through the client/server experiments, we observe that while the IP/ECN field is frequently bleached, all tested paths forward the TCP flags unchanged, except for the paths going through the mobile operators that use a non-ECN-capable proxy on port 80 (i.e., Movistar (ES) and TIM(IT)). Vodafone (IT) uses an ECN-capable proxy that honors the TCP flags. As previously mentioned, Tracebox is unsuited for measuring traversal of the TCP/ECN flags because the deployed router base implements RFC792. In our dataset, we find that only 2.4percent of routers (11k out of 468k) along the paths we tested are RFC1812-compliant.

Good news: ECN++ support is as good as ECN support. As described in the two earlier findings, we observe that in the 6.5M paths tested, ECN++ packets are treated by the network in the same way as ECN packets. In other words, ECN bleaching occurs as often in SYN, PureACKs and FIN packets as in data packets. The combinations of the ECN related fields used by ECN++ are not discarded or bleached more often than when used in ECN. Furthermore, AccECN negotiation in the TCP header traverses the paths we tested, except for the case on in-path proxies.

Good news: 61percent of Alexa top 500k server supports ECN. We used the results of the experiments we describe above, which we ran against Alexa top 500k web servers. Our finding corroborates the acceleration of ECN deployment reported in previous work [4], [6]. We also observe that 3.51percent of servers (10,709) support ECN in the SYN/ACK, a slight increase over the 1.3percent reported in [4]. Five of them answer with ECT(1) in the SYN/ACK, while the others use ECT(0).

Good news: All the ECN-capable top-500k Alexa servers we were able to test have the same response to ECE as to 3 DupACKs. Out of the 305k Alexa servers that support ECN, we were able to test 158k of them for their congestion response to ECE marks. We found that their response to an ECE mark is the same to the response to 3 DupACKs. We were unable to test the congestion response of 147k servers for various reasons, including that we were unable to find enough content to fill the IW; or that they were redirected.

Good news: At least, 51percent of the Alexa top-500k servers support IW of 10 segments. Additionally, out of the Alexa top-500k, 9.2percent support IW of 2 segments and 9.3percent support IW of 4 segments. We were unable to measure the IW of 74k (14percent) of the servers because we were unable to find enough data to fill IW. Though these results are not related to ECN, they are incidental to the measurement technique we employ for testing the congestion response.

Bad News: 0.4percent of the Alexa top-500k servers use IW larger than 10. The 0.4percent we observe account for 1,745 servers, all using an IW larger than 10. Out of these, 1,121 servers deliver the whole file in the first RTT (the largest IW observed is 585 packets of 100 Bytes). Similar behavior was also previously reported in [12].

Bad news: ECN+-enabled servers do not respond to

congestion in the SYN/ACK. For the 3.51percent of servers of the Alexa top-500K that respond with a SYN/ACK with the ECT codepoint, they all show the same odd behaviour, which superficially seems like ECN+. However, none of them respond to an ECE flag in the ACK of the 3WHS, none respond with a second non-ECT SYN/ACK, and none reduce their initial congestion window, all contrary to the ECN+ RFC. Instead, they all enter Congestion Avoidance phase (i.e. in the second RTT the congestion window is 1 MSS larger than IW).

Conclusions

This article opens another chapter in the sorry tale of ECN deployment. It was not all bad news. We found good news for the deployment of ECN++, which was the original subject of the study. And we can confirm that ECN adoption is proceeding well over fixed networks. However, we found bad news over mobile. More than half of the 18 mobile carriers tested routinely wipe the ECN field of all packets from clients. Fortunately, wiping the ECN field at the first hop only denies the benefits of ECN to the connection; the session otherwise proceeds as normal.

ECN problems in mobile have not surfaced before because this is the first ECN study to have extended to a broad enough set of mobile vantage points. This is due to the considerable work needed to build infrastructure like MONROE, which makes it feasible to measure the effect of kernel level changes over a wide range of mobile networks.

Acknowledgments

The work of Anna Maria Mandalari has been funded by the EU FP7 METRICS (607728) project.

The work of Marcelo Bagnulo has been performed in the framework of the H2020-ICT-2014-2 project 5G NORMA and the 5G-City project funded by MINECO.

This work was partially supported by the EU H2020 research and innovation program under grant agreement No. 644399 (MONROE) and grant agreement No. 688421 (MAMI). The opinions expressed and arguments employed reflect only the authors' view. The European Commission is not responsible for any use that may be made of that information.

References

[1] K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," RFC 3168 (Proposed Standard), Internet Engineering Task Force, Sep. 2001, accessed on 2017-11-27. [Online]. Available: http://www.ietf.org/rfc/rfc3168.txt

[2] J. H. Salim and U. Ahmed, "Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks," RFC 2884 (Informational), Internet Engineering Task Force, Jul. 2000, accessed on 2017-11-27. [Online]. Available: http://www.ietf.org/rfc/rfc2884.txt

[3] S. Floyd, "Inappropriate TCP Resets Considered Harmful," RFC 3360 (Best Current Practice), Internet Engineering Task Force, Aug. 2002, accessed on 2017-11-27. [Online]. Available: http://www.ietf.org/rfc/rfc3360.txt

[4] B. Trammell *et al.*, "Enabling Internet-Wide Deployment of Explicit Congestion Notification," in *PAM 2015*, vol. 8995. Springer, 2015, pp. 193–205.

[5] M. Alizadeh *et al.*, "Data Center TCP (DCTCP)," *Proc. ACM SIGCOMM'10, Computer Communication Review*, vol. 40, no. 4, pp. 63–74, Oct. 2010.

[6] M. Kühlewind, S. Neuner, and B. Trammell, "On the State of ECN and TCP Options on the Internet," in *PAM 2013*, vol. 7799. Springer, 2013, pp. 135–144.

[7] A. Kuzmanovic *et al.*, "Adding Explicit Congestion Notification (ECN) Capability to TCP's SYN/ACK Packets," RFC 5562 (Experimental), Internet Engineering Task Force, Jun. 2009, accessed on 2017-11-27. [Online]. Available: http://www.ietf.org/rfc/rfc5562.txt

[8] A. Medina, M. Allman, and S. Floyd, "Measuring the Evolution of Transport Protocols in the Internet," *SIGCOMM Computer Communication Review*, vol. 35, no. 2, pp. 37–52, Apr. 2005.

[9] G. Detal *et al.*, "Revealing Middlebox Interference with Tracebox," in *Proceedings of the 2013 Conference on Internet Measurement Conference*, ser. IMC '13. ACM, 2013, pp. 1–8.

[10] J. Postel, "Internet Control Message Protocol," RFC 792 (INTERNET STANDARD), Internet Engineering Task Force, Sep. 1981, updated by RFCs 950, 4884, 6633, 6918, accessed on 2017-11-27. [Online]. Available: http://www.ietf.org/rfc/rfc792.txt

[11] F. Baker, "Requirements for IP Version 4 Routers," RFC 1812 (Proposed Standard), Internet Engineering Task Force, Jun. 1995, updated by RFCs 2644, 6633, accessed on 2017-11-27. [Online]. Available: http://www.ietf.org/rfc/rfc1812.txt

[12] J. Padhye and S. Floyd, "On Inferring TCP Behavior," *Proc. ACM SIGCOMM'01, Computer Communication Review*, vol. 31, no. 4, pp. 287–298, Oct. 2001.

[13] O. Alay *et al.*, "Experience: An Open Platform for Experimentation with Commercial Mobile Broadband Networks," in *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '17. ACM, 2017, pp. 70–78.

[14] B. Chun *et al.*, "PlanetLab: An Overlay Testbed for Broad-coverage Services," *SIGCOMM Computer Communication Review*, vol. 33, no. 3, pp. 3–12, Jul. 2003.

Biographies

Anna Maria Mandalari (amandala@it.uc3m.es) completed her M.Sc. in Telecommunications Engineering at Mediterranea University of Reggio Calabria (Italy) in 2014 and her M.Sc. in Telematics Engineering at UC3M (Spain) in 2015. She is a Ph.D. candidate in the Telematics Engineering Ph.D. programme at University Carlos III of Madrid. She is doing her Ph.D. in the context of the Marie Curie project METRICS on large scale measurements of the effects of middleboxes in the Internet.

Dr. **Andra Lutu** (andra@simula.no) is a PostDoctoral Fellow at Simula Reseach Laboratory in Norway, working in on network performance measurements, network modeling and analysis, interdomain routing and mobile broadband networks. She previously obtained her Ph.D. and M.Sc. in 2014 and 2010, respectively, from IMDEA Networks Institute and University Carlos III of Madrid, Spain (UC3M) and her B.Sc. from 'Politehnica' University of Bucharest, Romania.

Dr. **Bob Briscoe** (research@bobbriscoe.net) currently consults for CableLabs. From 2015 to mid-2017 he was a Chief Research Scientist with Simula. Prior to that, he was BT's Chief Researcher in Network Infrastructure. Over his career, he has led a number of industry initiatives or start-ups to bring research to market. For instance the IETF has taken up four of his own research outputs as work areas, the latest being L4S.

Dr. **Marcelo Bagnulo** (marcelo@it.uc3m.es) received an Electrical Engineering degree in 1999 from the University of Uruguay, and a Ph.D. in Telecommunications in 2005 from the University Carlos III of Madrid, Spain. In 2000 he joined the University Carlos III of Madrid (UC3M), where he has been an associate professor since 2006. He has published over 60 papers in technical journals, magazines, and conferences and he is co-author of 18 IETF RFCs.

Dr. **Özgü Alay** (ozgu@simula.no) received the B.S. and M.Sc. degrees from Middle East Technical University, Turkey, and Ph.D. degree in Electrical and Computer Engineering at Tandon School of Engineering at New York University. Currently, she is a senior research scientist at Simula Research Laboratory, Norway and Associate Professor in University of Oslo, Norway. Her research interests lie in the areas of mobile networks, mobile applications and multi-path protocols.

Codepoint	IP/ECN field	Meaning
not-ECT	00	Not ECN-capable transport
ECT(0)	10	ECN-capable transport
ECT(1)	01	
CE	11	Congestion experienced

TABLE I: The ECN field in IP.

Feedback mode	Packet	TCP ECN flags (NS:CWR:ECE)		Allowed IP ECN	
		Description	flags	ECN	ECN++
Non-ECN	All	ECN disabled	000	00	00
ECN	SYN	ECN setup SYN	011	00	00
	SYN/ACK	ECN setup SYN/ACK	001	00	XX
	Data	Regular	000	XX	XX
		Echo CE	001	XX	XX
		CWnd Reduced	010	XX	XX
	Control & RTX	Same as data packet		00	XX
AccECN	SYN	AccECN setup SYN	111	00	XX
	SYN/ACK	AccECN setup SYN/ACK	010	00	XX
		—Ditto— with CE Echo	110	00	XX
	Data	CE counter	XXX	XX	XX
	Control & RTX			00	XX

TABLE II: Allowed IP ECN field for all types of TCP packet in all three feedback modes. X means 0 or 1.

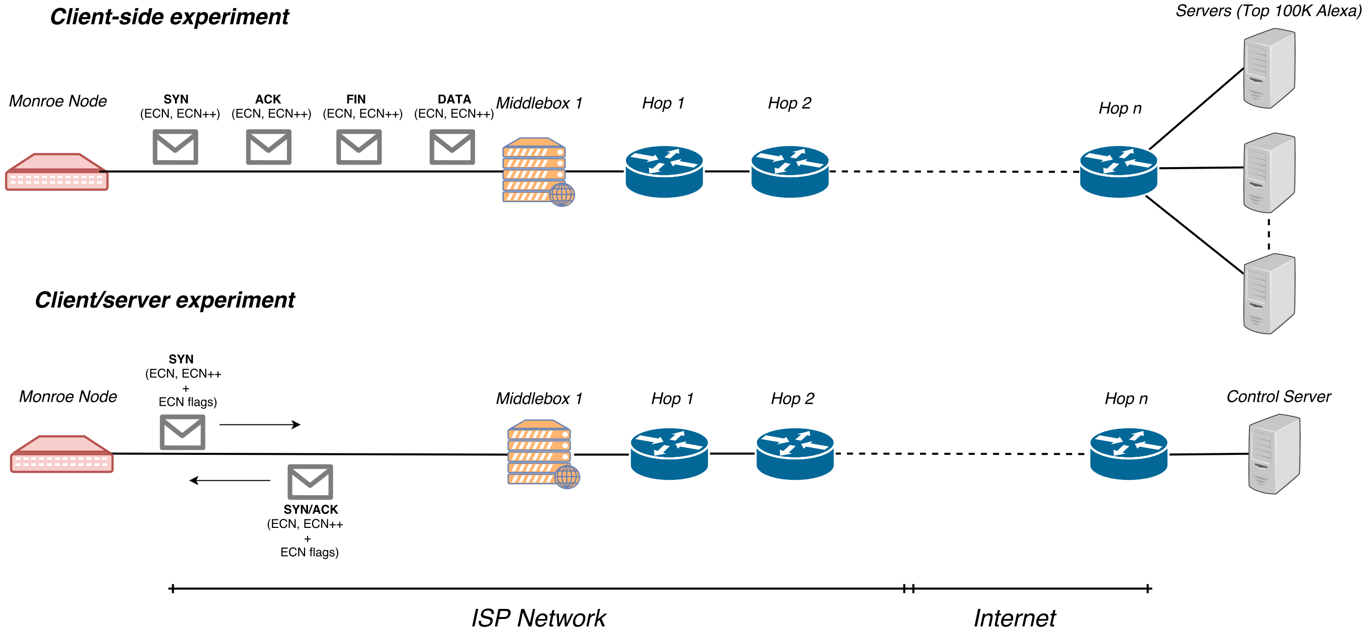

Fig. 1: ECN++ support: experimental setup.

Access operator	Country	ECN bleaching (%)		ECN++ bleaching (%)		Proxy on port 80
		1st hop	>5th hop	1st hop	>5th hop	
Vodafone	Italy	100	100	100	100	YES
TIM	Italy	100	100	100	100	YES
WIND	Italy	100	100	100	100	NO
Orange	Spain	100	100	100	100	NO
Yoigo	Spain	100	100	100	100	NO
Movistar	Spain	100 (port 80) 0 (port 443)	100 (port 80) 0.53 (port 443)	100 (port 80) 0 (port 443)	100 (port 80) 0.53 (port 443)	YES
Telia	Sweden	100	100	100	100	NO
Telenor	Sweden	0	0.53	0	0.53	NO
3	Sweden	0	0.53	0	0.53	NO
Telia	Norway	100	100	100	100	NO
Telenor	Norway	0	0.53	0	0.53	NO

TABLE III: ECN and ECN++ bleaching for mobile operators.