Enabling Internet-Wide Deployment of Explicit Congestion Notification

Brian Trammell[1], Mirja Kühlewind[1], Damiano Boppart[1], Iain Learmonth[2], Gorry Fairhurst[2], and Richard Scheffenegger[3]

[1] Communication Systems Group, ETH Zurich, Switzerland
[2] University of Aberdeen, Scotland
[3] NetApp, Inc.

Abstract. Explicit Congestion Notification (ECN) is an TCP/IP extension to signal network congestion without packet loss, which has barely seen deployment though it was standardized and implemented more than a decade ago. On-going activities in research and standardization aim to make the usage of ECN more beneficial. This measurement study provides an update on deployment status and newly assesses the marginal risk of enabling ECN negotiation by default on client end-systems. Additionally, we dig deeper into causes of connectivity and negotiation issues linked to ECN. We find that about five websites per thousand suffer additional connection setup latency when fallback per RFC 3168 is correctly implemented; we provide a patch for Linux to properly perform this fallback. Moreover, we detect and explore a number of cases in which ECN brokenness is clearly path-dependent, i.e. on middleboxes beyond the access or content provider network. Further analysis of these cases can guide their elimination, further reducing the risk of enabling ECN by default.

1 Introduction

Explicit Congestion Notification (ECN) [1] is a TCP/IP extension that allows congestion signaling without packet loss. Even though ECN was standardized in 2001, and it is widely implemented in end-systems, it is barely deployed. This is due to a history of problems with severely broken middleboxes shortly after standardization, which led to connectivity failure and guidance to leave ECN disabled. The authors revisited this question in [2], finding an increase in the number of servers which successfully negotiate and use ECN, but with nearly no use of ECN within a national-scale access network.

In this paper we show that server-side support for ECN negotiation has further increased. Unfortunately, server-side support is only the first step. Since TCP clients initiate ECN negotiation, it is client-side support and negotiation by default that is necessary to complete deployment on end-systems. While ECN must also be enabled on routers together with an Active Queue Management (AQM) scheme in order to be useful, the lack of deployment on end-systems reduces the incentive to deploy on routers and vice-versa. In the past two years, there has been increasing deployment of AQM [3] in the Internet; we expect this trend to continue and to drive router support for ECN. We therefore choose to focus on end-system deployment to break this loop, in the context of supporting on-going research in this area [4] to define more beneficial signaling. Specifically, this work aims to answer the following questions:

- What is the marginal risk of enabling ECN by default at the client-side?
- How can we detect and localize connectivity/signaling issues related to ECN?

To do so, we performed the following active measurements on nearly 600,000 popular web servers[4] taken from the Alexa top million list:

- Connectivity dependency: can ECN negotiation cause connectivity issues?
- ECN readiness: how many webservers will negotiate ECN if asked?
- ECN signaling anomalies: is ECN signaling viable to use end-to-end?

Specifically, the key focus of this work is on connectivity issues caused by ECN, in order to provide operational guidance and an answer to our most important question: is it now safe to use ECN-by-default on the client side to drive ECN deployment in the Internet? On this point we conclude that enabling ECN by default on client devices carries with it a low marginal risk of increased connection latency when fallback as recommended in RFC 3168 [1] is properly implemented; more measurement is necessary to localize the rare devices within the Internet that may lead to path-dependent failure of ECN-enabled connections. We provide a patch for Linux at `http://ecn.ethz.ch/ecn-fallback`; work to incorporate fallback into the Linux kernel mainline is ongoing.

All tools used in this study are available as open-source software, as are the raw data and intermediate results listing servers by ECN support status, from `http://ecn.ethz.ch`. We intend this work to introduce an ongoing ECN and middlebox impairment observatory which will support an effort to deploy ECN on an Internet-wide scale.

1.1 Overview of Explicit Congestion Notification (ECN)

ECN uses two bits in the IP header to mark traffic as ECN-capable or as having experienced congestion along the path, and when used with TCP it uses two flags, ECE and CWR, to negotiate the use of ECN in the TCP handshake and subsequently to echo congestion marking back to the sender during the connection. To review, a client sends an initial SYN ECE CWR to the server to negotiate ECN; to confirm negotiation, the server responds SYN ACK ECE, or to deny, simply SYN ACK. Section 6.1.1.1 of RFC 3168 [1] recommends falling back to non-ECN support if the initial SYN ACK ECE connection attempt fails.

After successful negotiation, data packets from each side can be marked using one of the ECN-Capable Transport codepoints (ECT(0)/ECT(1)) in the IP

[4] We examine HTTP in this study for comparison with related work, and because large-scale probing of HTTP is less likely to be regarded as abuse than other services.

header, which is replaced with the CE codepoint if a router's AQM along the path determines the link is congested. This congestion signal is echoed back to the sender marking all acknowledgments with the ECE flag until the sender acknowledges the receipt of the congestion signal with the CWR flag.

This describes the case where everything goes well. The negotiation and signaling in ECN can however go badly for various reasons. First, the two bits in the IPv4 and IPv6 header used for ECN were previously part of the Type of Service (ToS) byte, and there are still middleboxes and firewalls deployed in the Internet that use the old definition of these bits, interfering with ECN signaling. Second, firewalls may be configured to strip the ECN bits in the IP or TCP header, leading to negotiation and signaling errors; or to drop SYN ACK ECE, specifically to disable ECN, leading to connection failure. Third, end hosts and TCP proxies may have design or implementation faults in their handling of the semantics of the ECN bits.

1.2 Related Work

This work follows directly our previous work [2] and from [5], which sought to measure the state of ECN deployment as of August 2014 and September 2011, respectively. Our numbers for ECN capability and non-capability of webservers, being taken from the Alexa top million and using a comparable methodology, are therefore directly comparable to those in [2, 5]. We show that ECN support in webservers continues to increase, and reached the majority of the top million by the middle of 2014. Methodologies for packet mangling and marking are also comparable to those in [5]. More generally, this work follows from the continuing history of measurements of the Internet to estimate the ability to deploy new featues at the endpoints (e.g. Honda et al [6], Medina et al [7]), and contributes a data point to the continuing effort to improve the situation (e.g. the IAB Stack Evolution program[5] [8], or middlebox cooperation schemes such as [9]).

2 Methodology

2.1 Measurement Setup and Data Set

All measurements in this paper were performed from vantage-points running Ubuntu 14.04 (kernel 3.13.0 without SYN retry fallback as in RFC3168 [1]), run by commercial hosting provider Digital Ocean, in London, New York, and Singapore. Initial investigation showed that all ECN signaling works properly on this provider's networks, and all sites have native dual-stack connectivity. We ran trials on three seperate occasions, on 27 August, 4 September, and 9 September 2014.

As with previous work on testing ECN readiness of webservers [2, 5], we select our targets from Alexa's publicly available top million websites list. We then resolve these to at most one IPv4 and one IPv6 address per site. Duplicate

[5] `http://www.iab.org/activities/programs/ip-stack-evolution-program`

IP addresses are eliminated, taking the highest-ranked website for each address. Name resolution was performed on 27 August 2014 from the London vantage point using Google's public DNS server (8.8.8.8), resulting in 581,737 unique IPv4 addresses and 17,029 unique IPv6 addresses.

2.2 ECN-Spider and QoF

We built an active measurement tool atop the operating system's ECN implementation, to test ECN negotiation and negotiation-linked connectivity. This tool, called ECN Spider, is implemented in Python 3. ECN Spider takes as input a list of IP addresses along with the associated domain name and a number as a label to be used in later analysis; in this work, we use the Alexa rank. For each unique address, the tool then simultaneously opens one connection without attempting to negotiate ECN and one connection attempting to negotiate ECN, and reports the connection status for each, along with timing and HTTP status information.

ECN Spider's design is based on utilizing Linux's system-level configuration of ECN negotiation using the `sysctl` facility, using the implemented TCP stack instead of packet injection. For each site, we must therefore:

1. disable ECN using sysctl
2. open a socket to the target (attempts a SYN 3WHS)
3. enable ECN using sysctl
4. open a socket to the target (attempts a SYN ECE CWR 3WHS)
5. perform HTTP requests via both sockets

To make it possible to test a half million websites in a reasonable amount of time, the `sysctl` calls are performed in their own thread, which synchronizes with several hundred worker threads, amortizing the cost (about 10ms) of changing the system-wide setting. Each connection attempt is given 4 seconds to succeed, which can lead to transient connection failures on slower websites, but is necessary to keep slow and disconnected sites from delaying testing.

ECN Spider always tests connectivity without ECN first, in order to eliminate the possibility that sending an ECN negotiation packet down a path changes the result of the non-ECN SYN. When performing HTTP requests, ECN Spider does not follow redirects or otherwise crawl resources on the retrieved page.

While ECN Spider can detect whether or not a connection failed in the presence or absence of ECN negotiation, it cannot detect whether or not ECN was actually negotiated or observe negotiation anomalies, since this information is not available in userland. Therefore, we simultaneously observe the traffic with the QoF [10] flow meter to evaluate the traffic generated by ECN Spider providing TCP flags and ECN signaling information on a per-flow basis.

2.3 IPtables Packet Mangling

We also combined ECN Spider with the Linux iptables connection tracking and packet-mangling facilities in order to test the three following cases:

1. ECE response: mark all outgoing packets with CE to verify that we see ECE
2. CWR response: mark all incoming packets with CE to verify that we see CWR in response to ECE-marked ACKs
3. CE and ECT blackhole testing: mark SYN with CE/ECT(0)/ECT(1) to verify that marked packets are not dropped on path

In all cases the TCP MSS was set to 300 bytes, in order to split HTTP requests into multiple packets. For the ECE and CWR response testing, we used QoF for data analysis; for the CE and ECT blackhole testing, we analyzed ECN Spider's connectivity logs assuming that a path that drops marked SYNs would also drop other marked packets.

3 The Marginal Risk of Enabling ECN By Default

In our previous work [2], we found a multiple order-of-magnitude difference between the proportion of webservers supporting ECN negotiation and marking, and passively-measured flows on a university network actually negotiating and using ECN. Since webserver support is largely driven by the default configuration of the server operating system, the question naturally arises of whether client-side support could be driven by the same mechanism.

This is not a viable strategy if there still exist many paths through the Internet where attempting to negotiate ECN causes connectivity issues. Note that even with RFC3168 fallback, ECN-dependent connectivity can lead to additional connection setup latency, which depends on the client operating system. So we turn our attention to the question of marginal risk: how many additional connectivity issues can we expect if we turn ECN on by default?

3.1 Connectivity Dependency and Anomalies

Table 1 shows that for the vast majority of sites we probed, connectivity is clearly independent of whether ECN is requested or not. 578,433 (99.43%) of IPv4 and 16945 (99.50%) of IPv6[6] exhibit no ECN-dependent connectivity.

In 2443 cases for IPv4 and 16 cases for IPv6, connectivity apparently depends on ECN not being requested.The vast majority of these (2193 IPv4 and 13 IPv6 hosts) exhibit stable connectivity dependency at or near the host itself: every attempt to connect to the host with ECN failed, and every attempt to connect without succeeded.

This leaves us with the anomalous cases. We observe stable ECN dependency on the path in 15 cases for IPv4. Here, every connection attempt requesting ECN fails from one vantage point but succeeds from another. 6 of these sites are within a single AS (26496, GoDaddy.com LLC), and occur on servers used to park domain names. The remaining 9 may be more problematic, as they could represent ECN-disabling devices on path. A further 34 IPv4 and 3 IPv6

[6] Note that the relatively high prevalence of permanent IPv6 connection failure (nearly 10%) indicates continued limited operational experience with IPv6.

Table 1. Connectivity statistics, of 581,737 IPv4 hosts and 17,029 IPv6 hosts, all vantage points, 27 Aug - 9 Sep 2014

IPv4		IPv6		
hosts	pct	hosts	pct	description
553805	95.20%	14889	87.43%	Always connected from all vantage points
3998	0.69%	1594	9.36%	Never connected from any vantage point
8631	1.48%	138	0.81%	Single transient connection failure
11999	2.06%	324	1.90%	Non-ECN-related transient connectivity
578433	**99.43%**	**16945**	**99.50%**	**Total ECN-independent connectivity**
2193	0.38%	13	0.08%	Stable ECN dependency near host
15	0.00%	0	0.00%	Stable ECN dependency on path
34	0.01%	3	0.02%	Potential ECN dependency on path
201	0.03%	0	0.00%	Temporal ECN dependency
2443	**0.42%**	**16**	**0.09%**	**Total apparent ECN-dependent connectivity**
862	0.15%	69	0.41%	Inconclusive transient connectivity

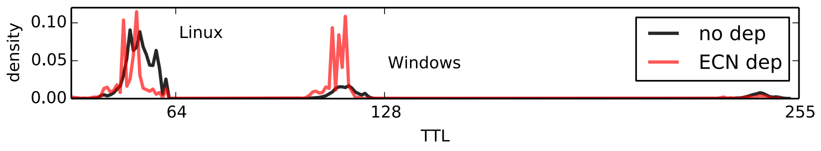

Fig. 1. TTL spectrum of ECN-dependent and -independent connectivity cases

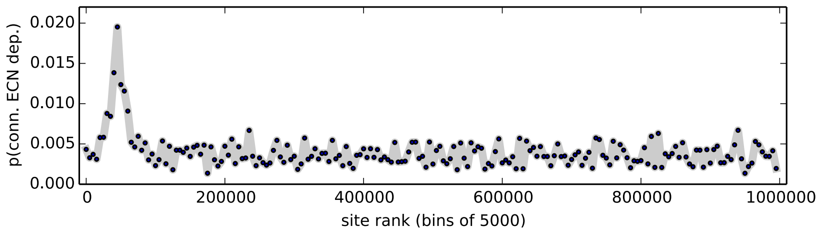

Fig. 2. Proportion of sites failing to connect when ECN negotiation is requested

hosts exhibit *potential* ECN dependency: no connection attempt requesting ECN succeeds from one vantage point, and at least one connection attempt with ECN from another vantage point succeeds, though we cannot rule out transient connectivity effects here. We also observed time-dependent anomalies: 201 cases for IPv4 where connectivity was ECN-independent from all vantage points during one trial, but ECN-dependent during another. This probably represents changes in network or host configuration during the time we ran our trials.

A further 862 cases for IPv4 and 69 for IPv6 cannot be definitively classified as either ECN-dependent or transient, leading us to estimate an upper-bound "blackhole" rate of 0.57% for IPv4 and 0.50% for IPv6. This is comparable to [5], suggesting that boxes that break connectivity when ECN is requested are not being replaced quickly.

Connectivity dependency can be linked to the operating system of the webserver by estimating the initial TTL. As shown in figure 1, sites with initial TTL 64 (Linux) and 128 (Windows) are roughly equally represented among hosts exhibiting ECN-dependent connectivity, while Linux servers are far more common among the majority where connectivity is ECN-independent. ECN-dependent connectivity failure also depends slightly on website rank as shown in figure 2: as many as 2% of websites with an Alexa rank between 50,000 and 55,000 fail to connect when ECN is requested, compared to a background rate of about 0.5%. The distribution of these sites by rank is shown in figure 2.

3.2 RFC 3168 Fallback Testing

Based on our RFC3168 ECN fallback Linux patch applied to single Ubuntu 14.04 machine at ETH Zurich running the 3.13 kernel we reran ECN Spider against the hosts which showed some evidence of connectivity depending on ECN and, as expected, we found that this patch eliminated connection failures attributable to ECN negotiation, at the cost of increased connection setup latency[7]. Therefore the implementation of ECN fallback as the default behavior in all operating systems will restore connectivity and is an important step for wide-scale deployment of ECN.

3.3 Conclusions

Our analysis therefore indicates that enabling ECN by default would lead to connections to about five websites per thousand to suffer additional setup latency with RFC 3168 fallback. This represents an order of magnitude fewer than the about forty per thousand which experience transient or permanent connection failure due to other operational issues. Comparison with [5] indicates this situation is likely unchanged in its magnitude since 2011.

As not all websites are equally popular, failures on five per thousand *websites* does not by any means imply that five per thousand *connection attempts* will fail. While estimation of connection attempt rate by rank is out of scope of this work, we note that the highest ranked website exhibiting stable connection failure has rank 596, and only 13 such sites appear in the top 5000.

4 An Analysis of ECN Signaling

We then analyzed the traces taken from our three ECN Spider runs using QoF to determine the extent of server-side support for ECN, and to investigate the prevalence of the different ways in which the ECN mechanism can fail today in the Internet.

[7] Fallback latency is a function of client implementation. We note anecdotally that additional latency is on the order of seconds on Windows 7, and barely noticeable on Mac OS X Mavericks.

Table 2. ECN negotiation statistics, of 581,711 IPv4 hosts and 17,028 IPv6 hosts, all vantage points, 27 Aug - 9 Sep 2014, compared to previous measurements.

IPv4		IPv6		2011	2012	
hosts	pct	hosts	pct	pct[5]	pct[2]	Description
326743	**56.17%**	**11138**	**65.41%**	11.2%	29.48%	**Capable of negotiating ECN**
324607	55.80%	11121	65.31%	–	–	...and always negotiate
2136	0.37%	17	0.11%	–	–	...sometimes negotiate, of which...
107	0.02%	1	0.01%	–	–	negotiation depends on path
27	0.02%	0	0.00%	–	–	sometimes reflect SYN ACK flags
248791	43.23%	3961	26.23%	82.8%	70.52%	Not capable of negotiating ECN
2013	0.35%	83	0.48%	–	–	...and reflect SYN ACK flags
6177	1.06%	1929	11.33%	–	–	Never connect with ECN (see §3.1)

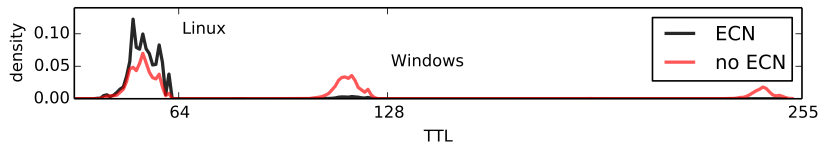

Fig. 3. Comparison of TTL spectrum between ECN-capable and -incapable hosts

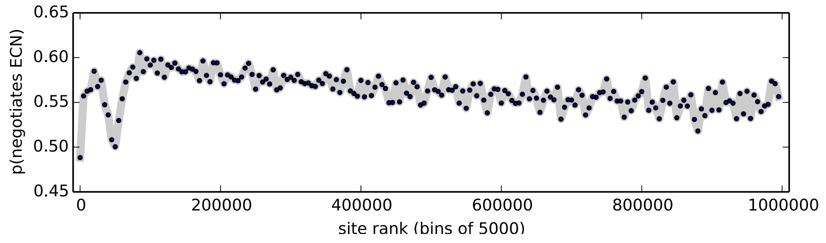

Fig. 4. Proportion of sites negotiating ECN by rank

4.1 ECN Negotiation

As seen in table 2, the majority of the top million web servers (56.17% of those connecting for IPv4, 65.41% for IPv6) are now capable of negotiating ECN, continuing a more or less linear trend since 2008. We attribute this to the decision to negotiate ECN if requested by the client by default in common server operating systems. Indeed, there continue to be large differences in ECN support per operating system, as shown in figure 3: note here that almost no initial-TTL 128 (i.e. Windows) or 255 (Solaris; also Google) hosts negotiate ECN. Considering only initial-TTL 64 (Linux) hosts, 326,720 of 468,555 or 69.73% are ECN capable.

As with connectivity, the proportion of hosts negotiating ECN depends slightly on the rank of the site, as shown in figure 4. The highest ranked website that will negotiate ECN has rank 6 (`www.wikipedia.org`). We note that websites of higher rank generally use custom networking software, and are therefore

Table 3. Relationship between ECN IP and TCP flags (*expected cases in italics*)

	IPv4 (N=581711)			IPv6 (N=17028)		
Marking	ECN	Reflect	No ECN	ECN	Reflect	No ECN
only ECT(0)	*315605*	693	1995	*8998*	1	46
ECT(0) + ECT(1)	0	0	0	4	1	7
ECT(0) on SYN ACK	7780	0	46	89	0	82
only ECT(1)	3	1	17	0	10	12
ECT(1) on SYN ACK	4	0	16	7	0	31
only CE	11	1	7	0	0	48
CE + ECT	5	2	0	23	66	39
CE on SYN ACK	11	0	5	22	0	87
none	6939	1343	*243150*	2013	5	*3694*

not affected by ECN negotiation by default. The top 100,000 sites are less likely to support ECN negotiation than the remaining 900,000.

Troubling are the 107 IPv4 hosts and one single IPv6 host for which ECN negotiation appears to be dependent on the vantage point. This indicates a device on path which mangles the ECN TCP flags. There are also 2029 IPv4 and 16 IPv6 hosts which sometimes negotatiate and sometimes do not, indicating either path or temporal instability in ECN signaling. Further, there are 2047 IPv4 hosts and 83 IPv6 hosts which reflect the ECN TCP flags on the SYN ACK (i.e., answering SYN ECE CWR with SYN ACK ECE CWR), indicating poorly implemented end-host stacks or TCP proxies. Of these, 693 IPv4 hosts and one IPv6 host go on to send ECT(0) marked packets, indicating that the end host may believe it has negotiated ECN correctly.

4.2 IP Signaling Anomalies

Assessing middlebox mangling of IP ECN signaling, we see in table 3 that 315,605 (97.2%) of the IPv4 hosts and 8998 (80.9%) of IPv6 hosts that always negotiate ECN mark all subsequent packets ECT(0) which is the expected signaling; we would expect ECN to work in these cases. On the other hand, there are 6939 (2.1%) IPv4 and 2013 (18.1%) IPv6 hosts which always negotiate ECN but never send an ECT marked packet in any trial from any vantage point. While it is acceptable for hosts which have negotiated ECN not to mark every data packet, this could also indicate a middlebox along the path that does not interfere with the ECN TCP signaling but does with ECP IP signaling. We note that this anomaly is less common for IPv4 than reported in [5], but it is not clear to what to attribute this change.

We can observe various interesting anomalies here which indicate possible mangling. 1995 IPv4 hosts do not appear to negotiate ECN but send ECT(0) marked packets anyway. 46 of these set ECT(0) on the SYN ACK which indicates a middlebox overwriting the former ToS field. The other 1849 cases indicate either a broken TCP stack, or ECN TCP flag mangling on the down-

Table 4. Marking on flows without ECN negotiation attempt

	IPv4 (N=581711)			IPv6 (N=17028)		
Codepoint	Once	Always	SYN ACK	Once	Always	SYN ACK
ECT(0)	4592	104	68	179	2	101
ECT(1)	21	18	18	116	76	39
CE	21	17	17	162	12	94

stream path wherein the server believes ECN has been negotiated, but the client does not, i.e. the ECE bit is cleared from the SYN ACK ECE sent by the server.We note that the magnitude of this anomaly is comparable with that reported in [5], indicating little if any improvement in middlebox mangling of ECN on this point. Conversely, table 4 gives insignt on hosts and paths using the ECN IP bits for non-ECN purposes, showing statistics for ECN marking by servers on connection attempts without ECN negotiation.

There are a few additional anecdotes to take from this analysis. The incidence of IPv6 negotiation anomalies (15.20%) is an order of magnitude higher than in IPv4 (1.93%), indicating that, although negotiation is supported by a higher proportion of IPv6 than IPv4 servers, ECN support in IPv6 in hosts and middleboxes is less mature. Many of these can be traced to specific providers: a single ISP in the Netherlands, for instance, is responsible for all 22 of the hosts that mark CE on the SYN ACK for IPv6 when negotiating ECN. Of five IPv4 hosts which send both CE and ECT marked packets, indicating the potential presence of a CE-marking router, there is only one (www.grandlyon.com, 213.162.51.7, as seen from London on 4 September and Singapore on 9 September 2014) for which we cannot rule out this hypothesis. In neither trial was the connection long enough to observe a CWR acknowledging the resulting ECE.

4.3 IP ECN connectivity and ECN echo tests

To further verify correct ECN signaling end-to-end, we ran CE and ECT blackhole experiments on 24 September, and ECE and CWR response tests on 23 September 2014, both from the London vantage point.

In the blackhole experiment, 4791 (0.82%) IPv4 hosts and 104 (0.61%) IPv6 hosts fail to connect when at least one ECN codepoint is set on the SYN. Of these, 2006 IPv4 and 12 IPv6 hosts are among those which also failed to connect from all vantage points when ECN was requested (see section 3.1). 287 IPv4 and 17 IPv6 hosts fail to connect regardless of the ECN codepoint set. In this experiment, the magnitude of transient failure is comparable to that in section 3.

The ECE response test succeeded for IPv4 in 94.8% (309,842 hosts) of all ECN-enabled cases in table 2. In contrast, the CWR response test succeeded only in 44.3% (144,290 hosts) of the cases. Further, we found 690 IPv4 hosts responding with ECE and 351 hosts responding with CWR even though ECN was not successfully negotiated. There also appears to be significant impairment or implementation error in ECN signaling for IPv6, with only 7 hosts responding ECE and 9 CWR.

Reasons for ECE response test failures include clearing of the CE codepoint along the forward path or clearing of the ECE flag along the reverse path. Reasons for CWR test failures include clearing of the ECE flag along the forward path, clearing of the CWR flag along the reverse path, or termination of the flow before CWR could be sent by the sender. As the median size of responses from hosts that did set CWR in the test was 3168 bytes, while the median size from those that did not was only 864 bytes (i.e., smaller than MSS), we do not consider our CWR results as a reliable indication of impairment on path.

Therefore, while the ECN IP-related connectivity risk is proportional to that related to ECN TCP signaling, the correct handling of ECE and CWR signaling after negotiation seem to be more impaired. Even worse, signaling is significantly more impaired on IPv6 than on IPv4.

5 Conclusions, Outlook, and Future Work

We have shown that while webservers support for ECN continues to increase, there does not appear to have been any appreciable reduction in the proportion of potential connectivity failure linked to ECN since 2011. The vast majority of connectivity problems we found with ECN negotiation were close to the server, i.e., cases in which routing changes during a connection would not lead to connection failure in the middle of an ECN-enabled flow. The fallback behavior defined in RFC 3168 eliminates connectivity risk for these cases, such that enabling ECN by default would lead only to increased connection latency when attempting to connect to about five of every thousand websites.

Verifying and localizing ECN path dependency in the remaining cases proves to be quite difficult. Bauer et al [5] used a tomography-based approach (as in e.g. Tracebox [11]) for localizing ECT mark clearing to an Autonomous System (AS); an approach that is unfortunately not applicable to our case. The ICMP Time Exceeded message contains no TCP flag information, making it impossible to verify path-dependent negotiation failures. Traceroute artifacts and on-path blocking of traceroute make it similarly impossible to differentiate connectivity issues from traceroute issues. Correlation of data-plane and control-plane routing information (e.g. from `http://stat.ripe.net/`) is a promising approach, but in none of our path-dependent connectivity cases did it yield a most-likely AS for the connectivity failure. We therefore leave further investigation of path dependency to future work, potentially leveraging existing wide-area distributed measurement platforms such as RIPE Atlas[8] to probe the set of paths through the Internet more comprehensively, using the volume of data to make up for the drawbacks of the traceroute-based tomography methods.

Our study shows that while it is safe for operating system vendors to activate ECN on the client-side by default presuming they implement RFC 3168 fallback, we cannot yet unreservedly recommend doing so. For a tiny minority of sites (15 of 598,766, or about 1 in 40,000) we cannot rule out path-dependent connectivity

[8] `https://atlas.ripe.net/`

issues. A similar proportion of sites exhibit indiscriminate CE marking, which would cause throughput degradation with use of ECN. These numbers are small enough that targeted collaboration with the operations community based on additional measurement is a viable way forward. We encourage other researchers to use the tools and dataset made available at `http://ecn.ethz.ch` to continue these investigations, and to guide the eventual elimination of ECN-unfriendly middleboxes, in order to move toward full deployment of ECN.

6 Acknowledgments

This work was materially supported by the European Commission though the Seventh Framework Grant Agreements mPlane (FP7-318627) and Reducing Internet Transport Latency (RITE) (FP7-317700); no endorsement of the work by the Commission is implied. Thanks to Stephan Neuhaus for his guidance during the development of ECN Spider, to Daniel Borkmann and Florian Westphal for discussions on Linux kernel modifications for RFC 3168 Fallback, and to Stuart Cheshire for his feedback.

References

1. Ramakrishnan, K., Floyd, S., Black, D.: The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168, IETF (September 2001)
2. Kühlewind, M., Neuner, S., Trammell, B.: On the State of ECN and TCP Options in the Internet. In: Proc. Passive and Active Measurement 2013, Hong Kong SAR, China (March 2013)
3. Baker, F., Fairhurst, G.: IETF Recommendations Regarding Active Queue Management: draft-ietf-aqm-recommendation-08. Internet-draft, IETF (December 2014) (Work in Progress).
4. Kühlewind, M., Wagner, D.P., Espinosa, J.M.R., Briscoe, B.: Using Data Center TCP (DCTCP) in the Internet. In: Proceedings of the third IEEE Globecom Workshop on Telecommunication Standards: From Research to Standards. (2014)
5. Bauer, S., Beverly, R., Berger, A.: Measuring the state of ECN readiness in servers, clients,and routers. In: Proc. of Internet Measurement Conference. (2011)
6. Honda, M., Nishida, Y., Raiciu, C., Greenhalgh, A., Handley, M., Tokuda, H.: Is it still possible to extend TCP? In: Proc. of IMC 2011. IMC '11, New York, NY, USA, ACM (2011) 181–194
7. Medina, A., Allman, M., Floyd, S.: Measuring the evolution of transport protocols in the Internet. SIGCOMM Comput. Commun. Rev. **35**(2) (April 2005) 37–52
8. Trammell, B., Hildebrand, J.: Evolving Transport in the Internet. IEEE Internet Computing (September 2014)
9. Craven, R., Beverly, R., Allman, M.: Middlebox-cooperative TCP for a non end-to-end Internet. In: Proceedings of ACM SIGCOMM 2014 Conference, Chicago, IL, USA (August 2014)
10. Trammell, B., Gugelmann, D., Brownlee, N.: Inline Data Integrity Signals for Passive Measurement. In: Proc. Sixth Int. Wksp. on Traffic Measurement and Analysis, London, England (April 2014)
11. Detal, G., Hesmans, B., Bonaventure, O., Vanaubel, Y., Donnet, B.: Revealing Middlebox Interference with Tracebox. In: Proceedings of the 2013 Internet Measurement Conference. IMC '13, Barcelona, Spain (2013) 1–8