Measuring the current state of ECN support in servers, clients, and routers

Steven Bauer and Robert Beverly

MIT CSAIL and NPS

{bauer@mit.edu, rbeverly@nps.edu}

Outline

1. **Why new ECN measurements are important**
2. **ECN refresher**
3. **ECN measurement methodology is more exciting than you might think**
4. **Interesting preliminary results**
5. **Future Work**

ECN is a hot topic again

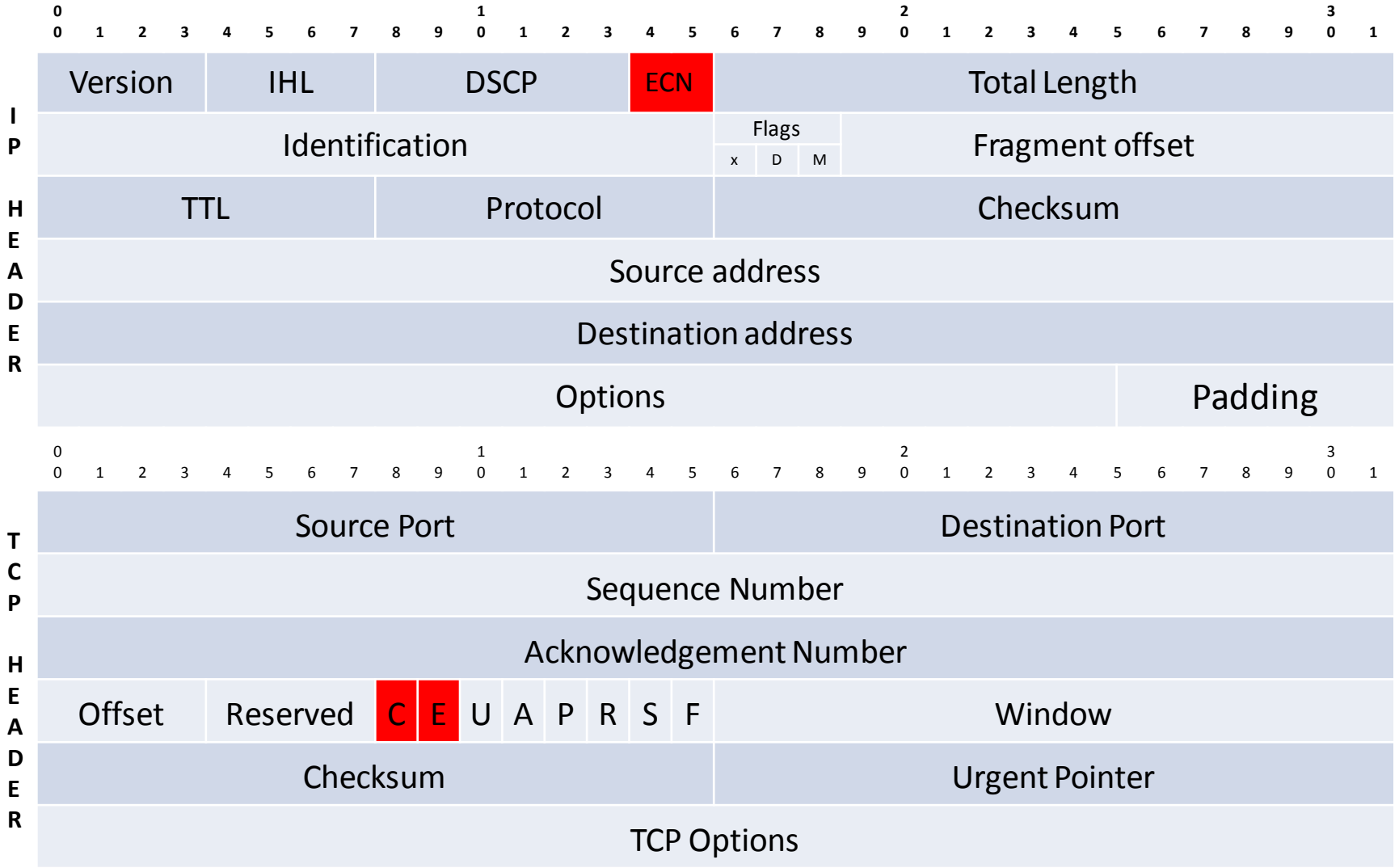
Recent technical discussions involving ECN

- Data Center TCP (DCTCP)
- IETF Congestion Exposure (conex) working group
  - Briscoe's re-ecn
- One proposed solution to latencies introduced by overly large buffers
  - "Buffer bloat", "Big buffer problem"
  - http://gettys.wordpress.com/category/bufferbloat/

Recent economic and policy discussions where ECN is an alternative solution

- Traffic volume is increasingly being challenged as the basis for interconnection and peering agreements
  - Level 3 / Comcast dispute
- Volume caps in broadband plans are increasingly being attacked for not necessarily relating to actual congestion
  - Canadian ISPs volume caps
  - Time Warner Cable

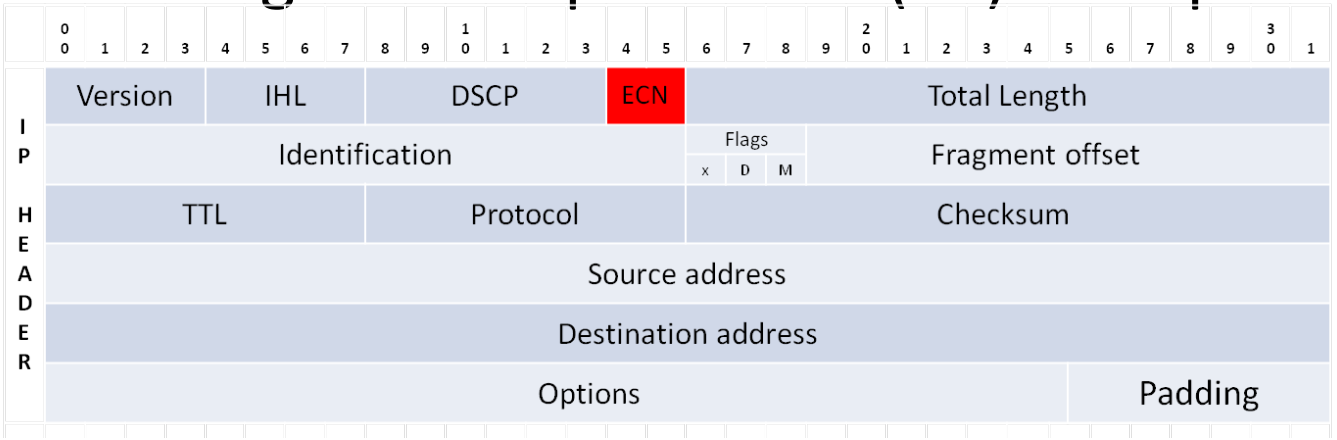
Four ECN bits in the TCP/IP header



ECN in a nutshell (1)

Marking in IP header:

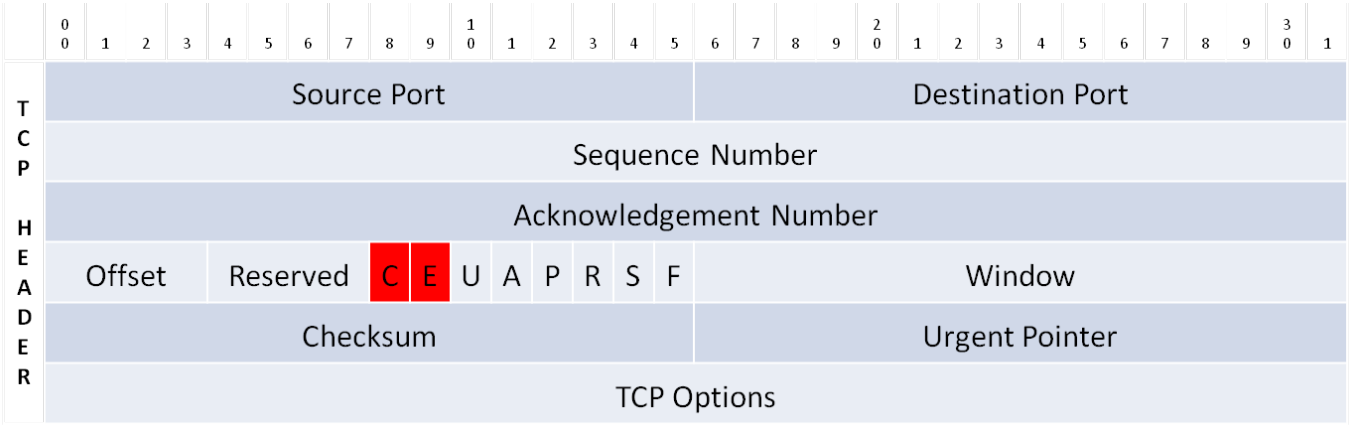
- IP packets in an ECN TCP flow set the ECN capable Transport (ECT) code point **0x10 (or 0x01)**
- If a router detects congestion, it marks the packet with Congestion Experienced (CE) code point **0x11**



ECN in a nutshell (2)

Negotiation and signaling in TCP header:

- ECN is negotiated as part of TCP 3-way handshake
- Upon receiving a packet with CE marked in IP header, destination host marks the TCP ECN Echo (ECE) bit in packets it sends to source host until…
- Source host receiving an ECE reduces its congestion window and sends a Congestion Window Reduced (CWR) market packet

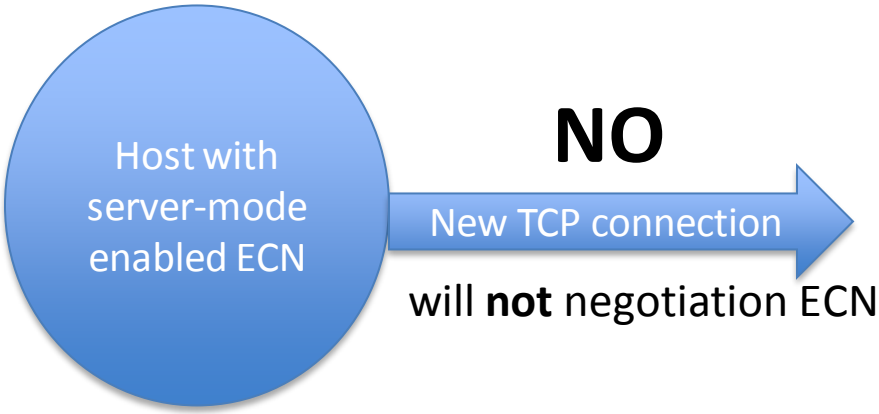


CSAIL

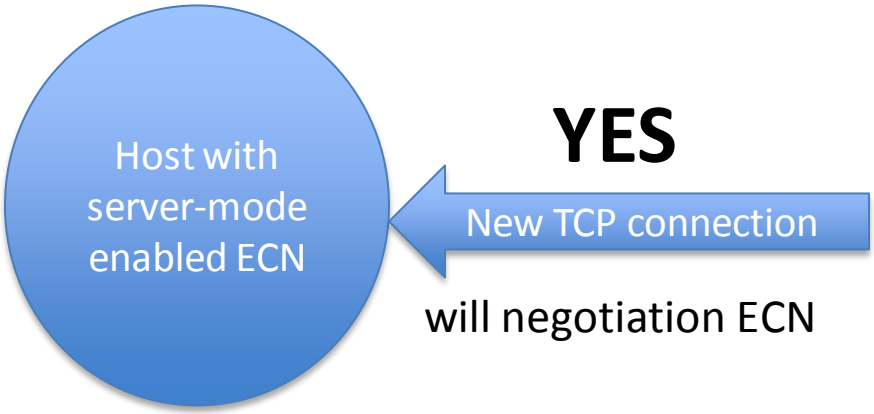
NPS PRAESTANTIA PER SCIENTIAM

Server-mode ECN

- Host will not negotiate ECN for outgoing TCP connections



- Host will negotiate ECN for incoming TCP connections



Chicken and egg problem of incremental ECN deployment answered: server side is enabling first

- Linux
  - Linux 2.3 router code for ECN. May 1999
  - Linux 2.4 full ECN support. January 2001.
  - Linux 2.6.31 server-mode **enabled by default** on kernel. Sept 2009
  - Important because of prevalence of Linux in server side architectures
- Windows
  - Vista ECN support
  - Windows 7 ECN support server mode **enabled by default?**
  - Server 2008 ECN support server mode **enabled by default?**
- Mac
  - OS X versions > =10.5 implement ECN
  - Full or server mode configurable
- Freebsd
  - ECN implemented in version 8.0 and later
- NetBSD
  - ECN support added by Google Summer of Code project in 2006.
- Mobile operating systems
  - Linux kernel of Android has ECN support but no easy way for users to enable (that I can figure out)

* Not personally verified. Info cribbed from Wikipedia, Sally Floyd's ECN page, commit logs, and other web pages

Chicken and egg problem of incremental ECN deployment answered: server side is enabling first

- Linux
  - Linux 2.3 router code for ECN. May 1999
  - Linux 2.4 full ECN support. January 2001.
  - Linux 2.6.31 server-mode **enabled by default** on kernel. Sept 2009
  - Important because of prevalence of Linux in server side architectures
- Windows
  - Vista ECN support
  - Windows 7 ECN support server mode **enabled by default**
  - Server 2008 ECN support server mode **enabled by default**
- Mac
  - OS X versions > =10.5 implement ECN
  - Full or server mode configurable
- Freebsd
  - ECN implemented in version 8.0 and later
- NetBSD
  - ECN support added by Google Summer of Code project in 2006.
- Mobile operating systems
  - Linux kernel of Android has ECN support but no easy way for users to enable (that I can figure out)

* Not personally verified. Info cribbed from Wikipedia, Sally Floyd's ECN page, commit logs, and other web pages

Chicken and egg problem of incremental ECN deployment answered: server side is enabling first

- Linux
  - Linux 2.3 router code for ECN. May 1999
  - Linux 2.4 full ECN support. January 2001.
  - Linux 2.6.31 server-mode **enabled by default** on kernel. Sept 2009
  - Important because of prevalence of Linux in server side architectures
- Windows
  - Vista ECN support
  - Windows 7 ECN support server mode **enabled by default?**
  - Server 2008 ECN support server mode **enabled by default?**
- Mac
  - OS X versions > =10.5 implement ECN
  - Full or server mode configurable
- Freebsd
  - ECN implemented in version 8.0 and later
- NetBSD
  - ECN support added by Google Summer of Code project in 2006
- Mobile operating systems
  - Linux kernel of Android has ECN support but no easy way for users to enable (that I can figure out)

Interest here is because operators control both the handset and proxies and thus are in a position to turn on ECN on both sides

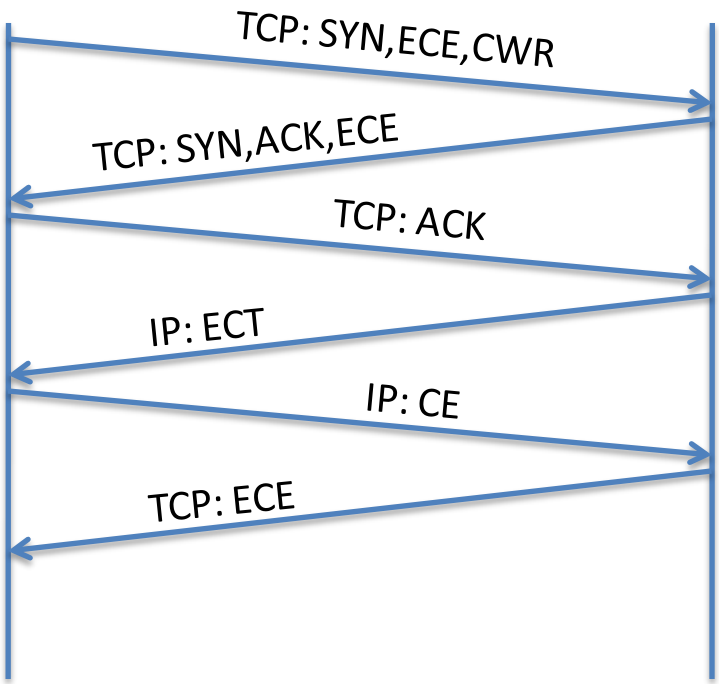
* Not personally verified. Info cribbed from Wikipedia, Sally Floyd's ECN page, commit logs, and other web pages

Updated and expanded ECN measurements needed

- Langley (2008) was the last study of ECN support before the deployment of server-mode ECN was default enabled in some OSes
- Maier (2009) observes “only a handful” of hosts using ECN in observations of 20,000 DSL customers
- Important to test more than just the web server population
  - Broadband networks
  - Video and CDN networks
  - University networks
  - Web servers

Testing ECN support

- Lots of questions to ask:



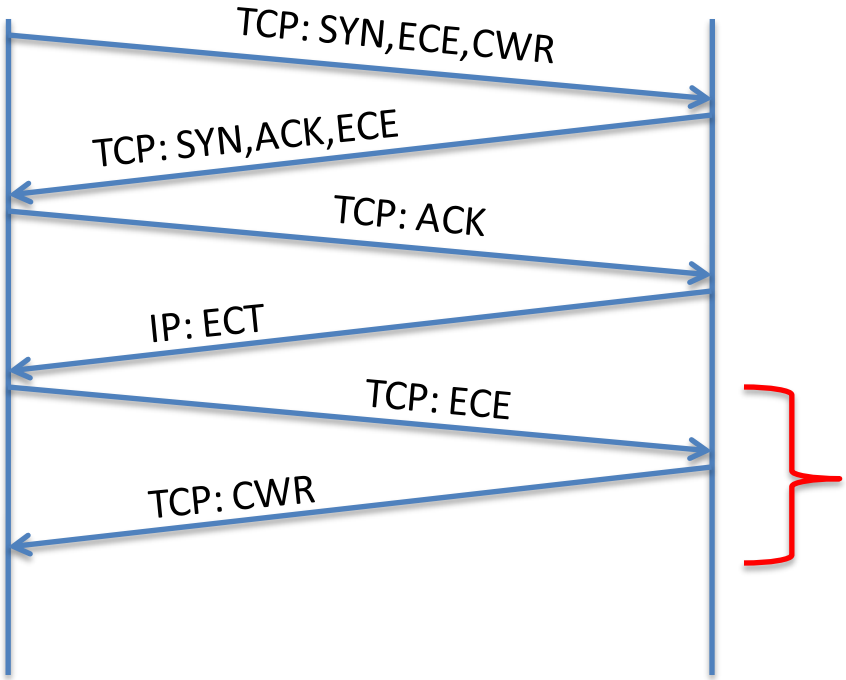
Q1: Fraction of hosts that negotiate ECN?

Q2: When TCP negotiated, is connection marked as ECN capable at IP?

Q3: Send artificial IP congestion signal. Is the corresponding TCP congestion echo observed?

Testing ECN support

- Lots of questions to ask:



Q4: Send artificial TCP congestion echo. Is the corresponding TCP congestion window reduced seen? Does the sender reduce the congestion window?

Networks are improperly clearing the ECN fields

- Compromises a carefully designed congestion feedback loop
  - Potentially raises concerns about the congestion safety or fairness of using ECN if senders don't back off
  - If CWR is cleared the receiver keeps sending ECE killing TCP throughput
- Hard for us to miss the cleared ECT bits :
  - My MIT lab cleared ECT on all connections
  - Home broadband provider cleared ECT on outbound path
- Naturally raised the question how much more wide spread this problem is
- Medina (2004) mentions some network paths may clear the ECT bits
- Also other potential barriers to ECN usage exist
  - Middleboxes that improperly drop TCP SYN with ECN

Server ECN support test populations

- Alexa top 1 million websites
  - Motivation: the largest number of flows
- Infrastructure of video and CDN providers
  - Motivation: the largest number of bytes
- University and college websites (8600 worldwide)
  - Motivation: we identified network ECN problems first at MIT

Testing server ECN Support

Basic methodology

- Start packet capture
- Retrieve whole page at <hostname>
- Analyze resulting pcap file and http headers returned

ECN tests

- Negotiated ECN at TCP layer
- ECT received at IP layer
- If ECN capable:
  - Set IP CE and wait for TCP ECE
  - Set TCP ECE and wait for TCP CWR

iptables trick

- Instead of a modifying a user-space TCP to implement the somewhat complex ECN rules…
- Leveraging iptables mangling, coupled with connection tracking and filters, provides a simple solution
- Sets CE on outgoing packets
  - iptables -t mangle -A OUTPUT -p tcp -m ecn --ecn-ip-ect 2 -m connbytes --connbytes 3:10 –connbytes-dir original –connbytes-mode packets –j TOS –or-tos 0x01
- Sets CE on incoming packets so the TCP stack will then handle sending ECE until a CWR is received
  - Iptables –t mangle –A INPUT -p tcp -m ecn –ecn-ip-ect 2 –m connbytes –connbytes 2:4 –connbytes-dir reply –connbytes-mode packets –j TOS --or-tos 0x01

Server population ECN results

	Langley 2008	Alexa			Universities/Colleges		
Aggregate	IP	host	IP	/24	host	IP	/24
Contact count	1,349,71	961,789	542,466	144,617	7,690	7,228	6,867
ECN successfully negotiated	1.07%	15.7%	12.7%	12.9%	9.4%	9.7%	9.8%

- This is a single test run
  - different runs show slightly different results perhaps due to load balancing?

Server population ECN results

	Langley 2008	Alexa			Universities/Colleges		
Aggregate	IP	host	IP	/24	host	IP	/24
Contact count	1,349,71	961,789	542,466	144,617	7,690	7,228	6,867
ECN successfully negotiated	1.07%	15.7%	12.7%	12.9%	9.4%	9.7%	9.8%

- A significant increase in ECN capability on the server side since 2008
- Note, these are different test populations

Server population ECN brokeness

	Alexa			Universities/Colleges		
Aggregate	host	IP	/24	host	IP	/24
ECN successfully negotiated	149,756	68,282	18,467	717	697	668
IP ECT broken	4,897	2,547 (3.7%)	1,551 (8.3%)	198	194 (27.8%)	192 (28.7%)
ECE broken ECT not broken	1,550	1,105	654	32	32	32
ECT not broken ECE not broken CWR broken	355	153	116	4	4	4

- Inbound IP ECN broken

Server population ECN results

	Alexa			Universities/Colleges		
Aggregate	**host**	**IP**	**/24**	**host**	**IP**	**/24**
ECN successfully negotiated	149,756	68,282	18,467	717	697	668
IP ECT broken	4,897	2,547	1,551	198	194	192
ECE broken ECT not broken	1,550	1,105	654	32	32	32
ECT not broken ECE not broken CWR broken	355	153	116	4	4	4

- Asymmetric: outbound to server broken at IP level, inbound to MIT not broken at IP level
- Current test implementation did not test if ECT was broken but ECE not broken since we never received a packet that indicated ECT

Server population ECN results

	Alexa			Universities/Colleges		
Aggregate	host	IP	/24	host	IP	/24
ECN successfully negotiated	149,756	68282	18,467	717	697	668
IP ECT broken	4,897	2,547	1,551	198	194	192
ECE broken ECT not broken	1,550	1,105	654	32	32	32
ECT not broken ECE not broken CWR broken	355	153	116	4	4	4

- Hosts that fail to send CWR
- Some manually inspected traces look like window is actually reduced but we just don't get the CWR
- Other traces clearly indicate the server does not receive our ECE

Where along the path from sender to receiver are the ECN bits getting cleared?

- Methodological insight is to leverage traceroute as the IP header returned inside the TTL-expired ICMP packet has the ECN field visible
- Which hop actually cleared the bit, the router that returned the ICMP packet or the one before it?
  - Based upon the cases where we know the answer, the previous hop is the best device to finger as the culprit
  - Possible other routers are different…

Where along the path from sender to receiver are the ECN bits getting cleared?

- Only able to diagnosis one direction
  - Obviously asymmetric paths are possible
  - But even without asymmetric paths, configurations which break ECN in only one direction are possible (already have an example of this)
- ICMP, UDP, TCP SYN, TCP ACK traceroutes are all possible.
  - Only existing standard is for ECT in TCP packets with data
  - We tested with all the above traceroute types but didn't find any apparent ECN related differences
  - ICMP finds more hops so we are currently leveraging it

ECN traceroute

- Current vantage point is MIT only
  - Leverage ARK for the next iteration, just waiting on an ARK change so that we can get the TOS field
- **If routers ever turned on ECN marking, we could use method to find what routers were congested**
  - Assuming our traceroute packets were occasionally marked
  - Assuming which hop is congested doesn't vary too much

Scamper results

	Count	Total
Paths with ECT cleared	27,263	542,466
Unique IP **at hop before** router that returned ICMP packet with ECT cleared	1,749	27,263
Unique IP **at hop** that returned ICMP packet with ECT cleared	3,566	27,263

Testing ECN support of some of the largest sources of network traffic

- Infrastructure of video and CDN providers
  - i.e. “hyper giants” responsible for large fractions of traffic volume on the Internet
  - Could have a big impact if they turned on ECN

Testing ECN support of some of the largest sources of network traffic

- Methodological challenges
  - Content providers, for instance Netflix, uses multiple CDNs
  - Where content is hosted changes over time
  - CDNs have heterogeneous infrastructures
  - Tests need to actually exchange traffic, not just do 3-way handshake with a server.
  - Requires valid URLs of content to fetch
  - But content can be restricted to paying members (e.g. Netflix videos) or require complex multi-stage processes (e.g. to get cookies set properly)
  - Video players that don't work under Linux

Comments on preliminary results of testing video/CDN providers

- Manual testing by browsing sites while wireshark is running
- Inspection of packet traces for all ECN enabled TCP connections
- No ECN capable server actually delivering content yet found
  - Some infrastructure where log files from a video player were POSTed were ECN capable
  - Heterogeneous infrastructures
- None of the CDN infrastructure for non-video content we have tested so far enable ECN
- **In need of feedback for how to make this more systematic and comprehensive…**

Client ECN Support

- Most previous work investigates server-side ECN support
- What about client side?
  - Our own broadband network had a problem
- As Maier (2009) observed, “only a handful” of hosts initiate ECN capable TCP in broadband networks
- Idea:
  - Find a way to initiate ECN connection with clients

Client ECN Support

- To obtain a large set of potential ECN-capable servers located at access edge, we turn to P2P (where clients are also servers)
- Use ion-stumbler [Stuzbach 2009] crawler on ECN enabled server
- Use aforementioned iptables tricks
- Capture packets

Client ECN preliminary results

Measure	Count	Total	Percent
ECN successfully negotiated	121	200,138	0.06%
ECN RST	464	200,138	0.23%
ECT broken	53	121	42.8%
ECE broken	18	116	15.5%
CWR broken	17	17	100%

Client ECN preliminary results

Measure	Count	Total	Percent
ECN successfully negotiated	121	200,138	0.06%
ECN RST	464	200,138	0.23%
ECT broken	53	121	42.8%
ECE broken	18	116	15.5%
CWR broken	17	17	100%

- A very small percentage

Client ECN preliminary results

Measure	Count	Total	Percent
ECN successfully negotiated	121	200,138	0.06%
RST with ECE	464	200,138	0.23%
ECT broken	53	121	42.8%
ECE broken	18	116	15.5%
CWR broken	17	17	100%

- Reset packet received in response to SYN had ECE on… not sure how to interpret that. Maybe a ECN capable box that is simply not listening on the port any longer?

Client ECN preliminary results

Measure	Count	Total	Percent
ECN successfully negotiated	121	200,138	0.06%
ECN RST	464	200,138	0.23%
ECT broken	53	121	42.8%
ECE broken	18	116	15.5%
CWR broken	17	17	100%

- Significantly higher than percentage seen in server populations… but a small sample size

Client ECN preliminary results

Measure	Count	Total	Percent
ECN successfully negotiated	121	200,138	0.06%
ECN RST	464	200,138	0.23%
ECT broken	53	121	42.8%
ECE broken	18	116	15.5%
CWR broken	17	17	100%

- We are not sure how to interpret this…
- Needs more validation

What is clearing the ECN bits?

Known causes

- Switches
  - Configuration designed to copy 802.1p field from Ethernet to DSCP was overwriting all 8 bits of the TOS field
  - This was a problem at MIT
- Cable broadband network CMTS
  - Intention was to clear the diffserv field
  - We worked with provider to fix the problem

Possible causes

- NATs and home routers
- Load balancers
- Middle-boxes

Our measurements have prompted changes already

- Documenting problems gives us leverage to fix them
  - MIT's CSAIL network
  - large broadband provider
- Fairly quick fixes in both cases after the right folks were sent traces demonstrating the issue

Future Work

1. Tests to determine if servers fail to respond to SYNs with ECN bit
   - Langley study recorded a 0.56% failure rate
2. Resolve measurement ambiguities:
   - Home modems and other layer 2 rewriting we can't detect
   - Inconsistencies (load balancing?)
3. Deploy on Caida's Ark infrastructure:
   - More vantage points, explore more paths in network
4. Test whether remote side actually reduces congestion window, not just signals that they have
5. Additional measurements of "client side" support of ECN
6. Improved methodology for testing "hyper giants"
7. Website for users to test their ECN and path
   - http://test-ecn.csail.mit.edu

EXTRA SLIDES

Langley study methodology

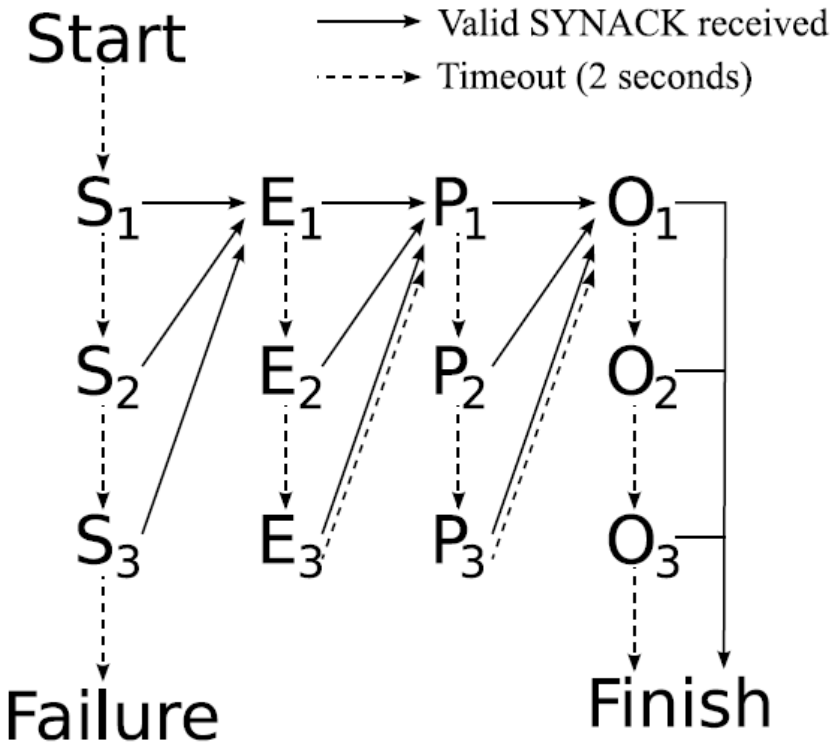


Figure 1: Prober state machine

Medina (2004)

ECN fields in data packets	Number	% of total
ECN-capable servers	1765	100%
Received packets w/ ECT 00 (Not-ECT)	758	42%
Received packets w/ ECT 01 (ECT(1))	0	0%
Received packets w/ ECT 10 (ECT(0))	1167	66%
Received packets w/ ECT 11 (CE)	0	0%
Received packets w/ ECT 00 and ECT 10	174	10%

Table 3: Codepoints in data packets from ECN-Capable Servers

Medina (2004)

Year:	2000		2004	
ECN Status	**Number**	**%**	**Number**	**%**
Number of Servers	24030	100%	84394	100%
I. Classified Servers	21879	91%	80498	95.4%
I.A. Not ECN-capable	21602	90%	78733	93%
I.B. ECN-Capable	277	1.1%	1765	2.1%
I.B.1. no ECN-Echo	255	1.1%	1302	1.5%
I.B.2. ECN-Echo	22	0.1%	463	0.5%
I.C. Bad SYN/ACK	0		183	0.2%
II. Errors	2151	9%	3896	4.6%
II.A. No Connection	2151	9%	3194	3.8%
II.A.1. only with ECN	2151	9%	814	1%
II.A.2. without ECN	0		2380	2.8%
II.B. HTTP Error	–		336	0.4%
II.C. No Data Received	–		54	0%
II.D. Others	–		312	0.4%

Table 2: ECN Test Results

RFC 3168

6.1.5. Retransmitted TCP packets

- This document specifies ECN-capable TCP implementations MUST NOT set either ECT codepoint (ECT(0) or ECT(1)) in the IP header for retransmitted data packets, and that the TCP data receiver SHOULD ignore the ECN field on arriving data packets that are outside of the receiver's current window. This is for greater security against denial-of-service attacks, as well as for robustness of the ECN congestion indication with packets that are dropped later in the network.

RFC 3168

6.1.4. Congestion on the ACK-path

- For the current generation of TCP congestion control algorithms, pure acknowledgement packets (e.g., packets that do not contain any accompanying data) MUST be sent with the not-ECT codepoint. Current TCP receivers have no mechanisms for reducing traffic on the ACK-path in response to congestion notification. Mechanisms for responding to congestion on the ACK-path are areas for current and future research. (One simple possibility would be for the sender to reduce its congestion window when it receives a pure ACK packet with the CE codepoint set). For current TCP implementations, a single dropped ACK generally has only a very small effect on the TCP's sending rate.