More Accurate ECN Feedback in TCP

draft-ietf-tcpm-accurate-ecn-09+

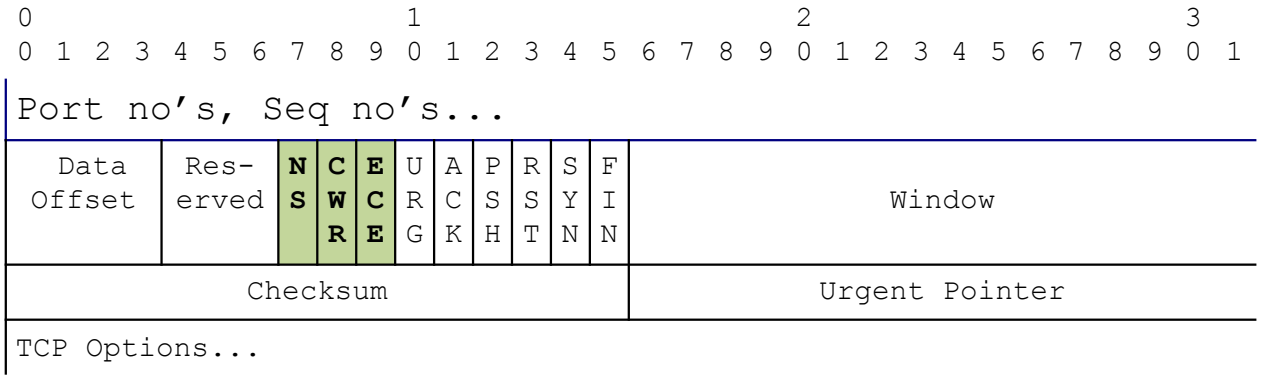
Bob Briscoe, CableLabs
Mirja Kühlewind, ETH Zürich
Richard Scheffenegger, NetApp

IETF-107 Mar 2020

Problem (Recap)
Congestion Existence, not Extent

- Explicit Congestion Notification (ECN)
  - routers/switches mark more packets as load grows
  - RFC3168 added ECN to IP and TCP

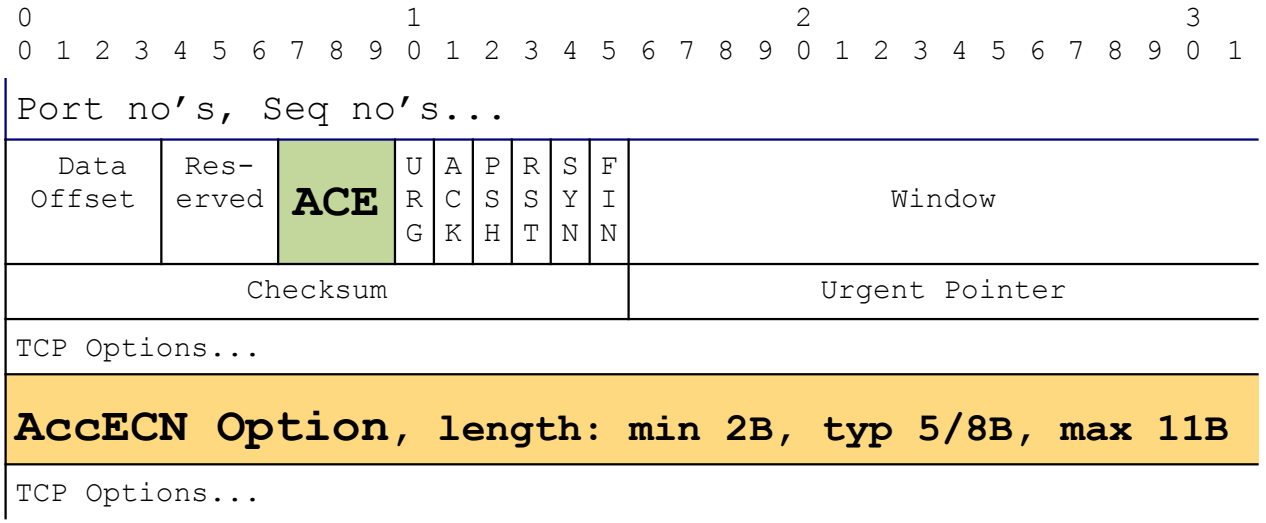
IP-ECN	Codepoint	Meaning
00	not-ECT	No ECN
10	ECT(0)	ECN-Capable Transport
01	ECT(1)	
11	CE	Congestion Experienced



- Problem with RFC3168 ECN feedback:
  - only one TCP feedback per RTT
  - rcvr repeats ECE flag for reliability, until sender's CWR flag acks it
  - suited TCP at the time – one congestion response per RTT

Solution (recap)
Congestion extent, not just existence

- AccECN: Change to TCP wire protocol
  - Repeated count of CE packets (ACE) - essential
  - and CE bytes (AccECN Option) – supplementary

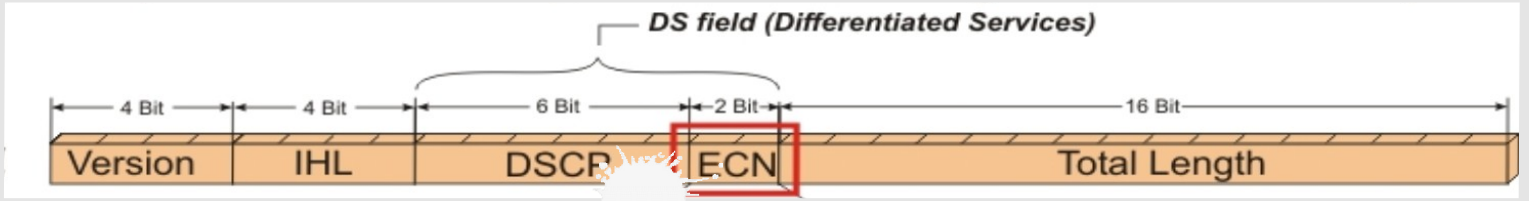


- Key to congestion control for low queuing delay
  - 0.5 ms (vs. 5-15 ms) over public Internet
- Applicability: (see spare slide)

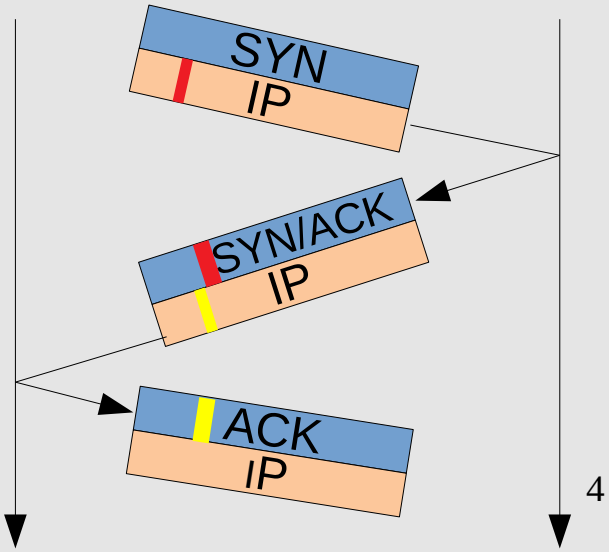
Fall-back if IP/ECN bleached/mangled

- We thought ECN traversal was surprisingly perfect
  ...until the latest measurement study*
  - ~60% of those mobile operators measured bleach upstream ECN by 1st IP hop
  - Prob. prevalent bug that wipes ECN as side effect of Diffserv bleaching

Octets 1-4 of IPv4 header

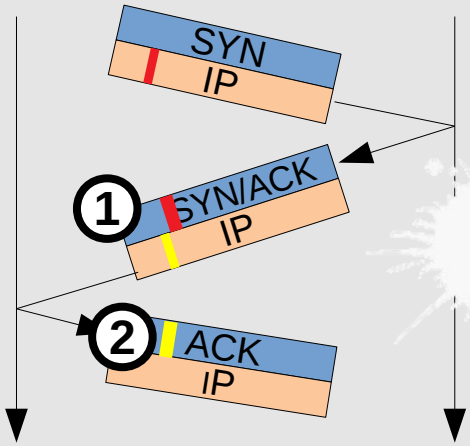


- Solution: Feed back (in the 3 TCP/ECN flags) which of 4 possible IP/ECN codepoints arrived on:
  - SYN : in SYN-ACK
  - SYN/ACK : in ACK of 3WHS
    - (this ACK is not reliably delivered)
- If mangled, disable ECN for half connection



* see ECN++ presentation (IETF-100 tcpm),
or http://www.it.uc3m.es/amandala/ecn++/

Feedback of IP/ECN during 3WHS



①

A	B	SYN A->B			SYN/ACK B->A			Feedback Mode
		AE	CWR	ECE	AE	CWR	ECE	
AccECN	AccECN	1	1	1	0	1	0	AccECN (Not-ECT on SYN)
AccECN	**AccECN**	1	1	1	**0**	**1**	**1**	**AccECN (ECT1 on SYN)**
AccECN	**AccECN**	1	1	1	**1**	**0**	**0**	**AccECN (ECT0 on SYN)**
AccECN	AccECN	1	1	1	1	1	0	AccECN (CE on SYN)
AccECN	Nonce	1	1	1	1	0	1	classic ECN
AccECN	ECN	1	1	1	0	0	1	classic ECN
AccECN	No ECN	1	1	1	0	0	0	Not ECN
:	:	:	:	:	:	:	:	:
AccECN	Broken	1	1	1	1	1	1	Not ECN

- Consumes last 2 combinations of TCP/ECN flags on SYN/ACK

- Same coding on ACK
  - ACE counter in prev. drafts

Notes:

1) Could be TCP bleaching
2) Used by RFC5562 + SYN cookie
3) Currently Unused

②

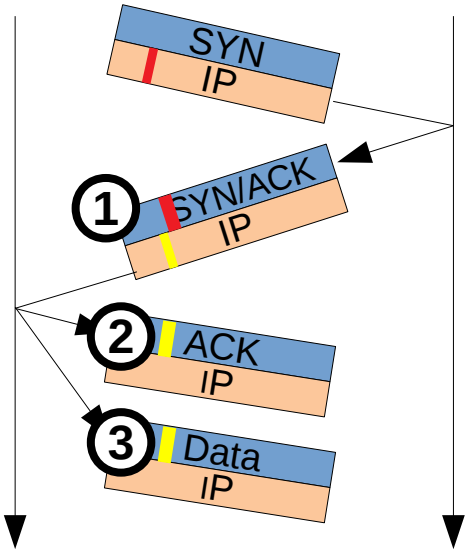
ACE on ACK of SYN/ACK	IP-ECN codepoint on SYN/ACK inferred by server	Initial s.cep of server in AccECN mode
0b000	{Notes 1, 2}	Disable ECN
0b001	{Notes 2, 3}	5
0b010	Not-ECT	5
0b011	ECT(1)	5
0b100	ECT(0)	5
0b101	Currently Unused {Note 3}	5
0b110	CE	**6**
0b111	Currently Unused {Note 3}	5

Changes to handshake reflection subsequent to draft-04

- draft-05

  The encoding in Table 3 is solely applicable on a packet in the client-server direction with an acknowledgement number 1 greater than the Initial Sequence Number (ISN) that was used by the server

  - Reason:
    - repeats reflection on first data packet (if any) because 3rd ACK is not delivered reliably



- draft-09+

  (?) don't increment CE counters within handshake

  - Reasons:
    - redundant given handshake reflection
    - then have to check and handle possible inconsistency
    - avoids burning >1 initial value of CE counters (useful for future extensions)

  (?) considering client only reflecting on 3rd ACK, not data

  - Reasons: (see Ilpo's presentation)

Following slides are to aid discussion
Painful detail not intended for tcpm presentation

Proposal #4; without TFO

- `The encoding in Table 3 is solely applicable on a` **`pure ACK in the client-server direction that does not acknowledge new data`**
- `The CE packet counters (r.cep and s.cep), count the number of packets the host receives with the CE code point in the IP ECN field, including CE marks on control packets without data,` **`but excluding any CE on the SYN`**`.`

  (plus suitable normative requirements consistent with these)

- Reasons:
  - Reflect 4 SYN states reliably in SYN/ACK
  - Too complex to reflect 4 SYN/ACK states reliably
  - So, reflect 4 SYN/ACK states unreliably on 3rd ACK only (but not 1st data)
  - nonetheless, CE feedback delivered reliably
- Trade-off (in client→server direction only):
  - reliable CE feedback vs. extra initial ACE value for future extensibility

Proposal #4; with TFO

- In the section at the end about interaction with other TCP experiments
- **If an AccECN client sent a non-zero TFO cookie on the SYN**

  and if the client enters AccECN mode on receipt of the SYN/ACK, it SHOULD reflect the IP-ECN field of the SYN/ACK in a pure ACK that acknowledges no new data.
- **If there was data on the SYN/ACK, the client can then acknowledge this new data separately, without reflecting the IP/ECN field.**
- **In addition, the server MAY record any failure of the SYN/ACK reflection test in the TFO cookie it gives to the client to initialize a subsequent connection.**

  Note that the server cannot include this information in the first TFO cookie it returns to any particular client, which it sends on the SYN/ACK before it knows the result of the reflection test.

Proposal #4: What if 3rd ACK lost?

- If server
  - does not receive the pure ACK covering no new data
  - and it is not receiving the AccECN TCP Option
  - it cannot test for server-client ECN mangling, during the 3WHS
  - but it does catch up with CE feedback on subsequent packets (if no mangling)
- In environments where ECN mangling is likely,
  - if constant CE feedback from start
    - disable ECN for the half-connection
  - if no 3WHS test was possible, and no CE yet,
    - server SHOULD set CE on at least one early data packet
    - then, if ACE does not increment on that packet's ACK, disable ECN for the half-connection
    - server MAY suppress the congestion response to this CE feedback
  - (this is also a weak test of receiver's feedback integrity)

Proposal #4; what if >1 SYN/ACK?

- If there is CE on the SYN/ACK, the client MUST feed back the CE on the 3rd ACK and increment r.cep so that it is subsequently fed back reliably, via the ACE field.
- **If more than one SYN/ACK arrives at the client, it MUST only increment r.cep once.**
- On receipt of the first packet from the client that feeds back that at least one SYN/ACK was marked CE (whether the 3rd ACK or a subsequent packet from the client), the server increments s.cep and applies the appropriate congestion response.
- **If 3rd ACK indicates CE, but ACE on subsequent first packet still at initial value of 5, server MUST disable ECN for the half-connection.**
- Reasons:
  - Presume same congestion response at server no matter how many SYN/ACK attempts were CE marked. (?)
  - Avoids burning more than 2 initial values of the ACE counter

Proposal #4: Can reflection tests be removed?

- If mangling becomes a non-problem long-term
- Free up codepoints?
  - Would like to reduce from 4 to 2 reflection codes on SYN/ACK & on 3rd ACK
    - possible, but a drawn-out 2-stage process
    - burn another code:
      Not CE = Not ECN || ECT0 || ECT1
      then wait for use of the 3 old codes to subside
- Free up test processing?
  - Either end can just not check for a valid transition but they have to check for the CE transition anyway

Change Triggered ACKs

Minor Edits

Status & Next Steps

- Full implementation in Linux(1)
- Implemented without TCP Option in FreeBSD(2)
- Ready for WGLC

(1) https://
(2) https://

AccECN

Q&A

spare slides