# More Accurate ECN Feedback in TCP

## draft-ietf-tcpm-accurate-ecn-04

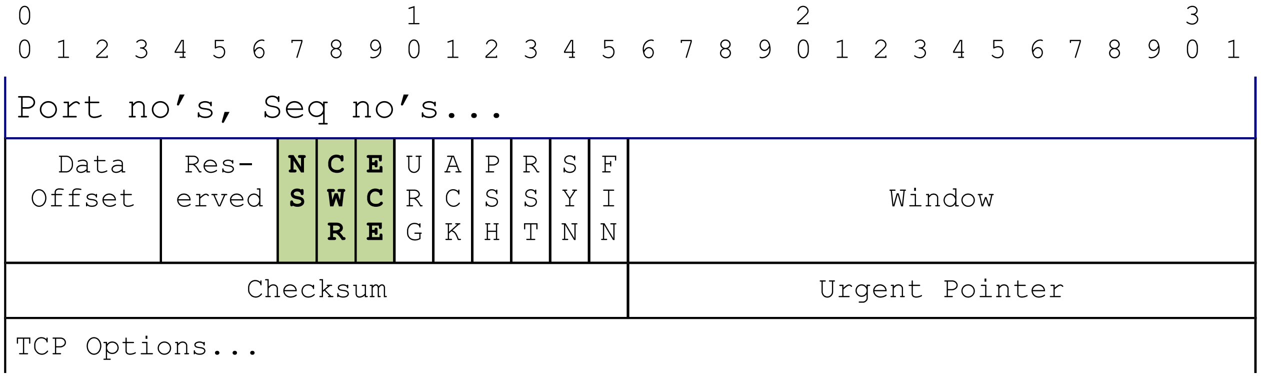
Bob Briscoe, CableLabs
Mirja Kühlewind, ETH Zürich
Richard Scheffenegger, NetApp

IETF-100 Nov 2017

# 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 |

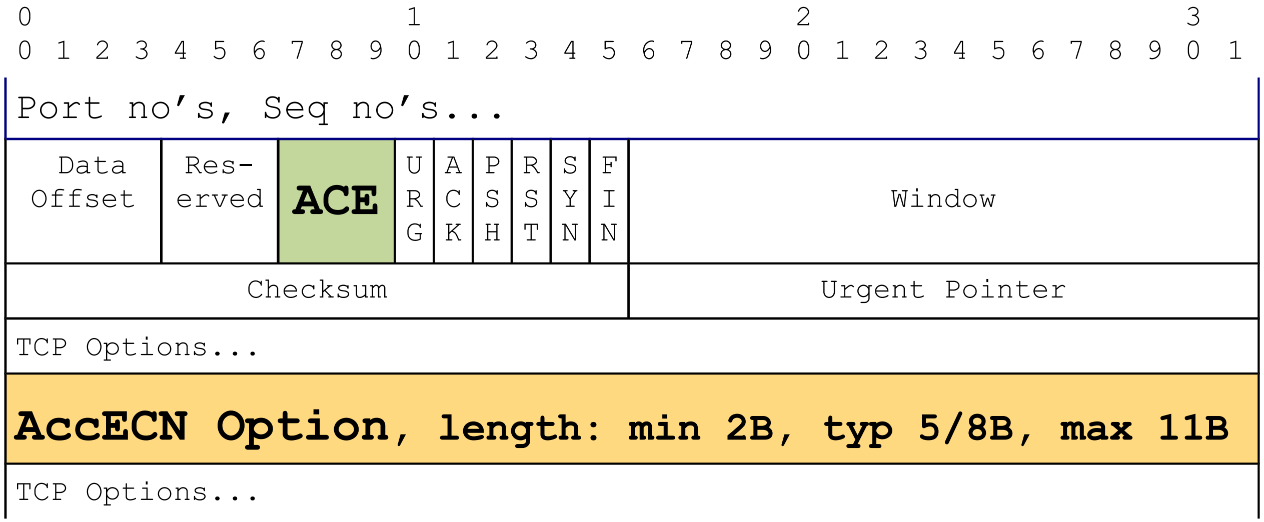


```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Port no's, Seq no's...
Data    Res-   N C E U A P R S F
Offset  erved  S W C R C S S Y I      Window
                 R E G K H T N N
Checksum                    Urgent Pointer
TCP Options...
```

- 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

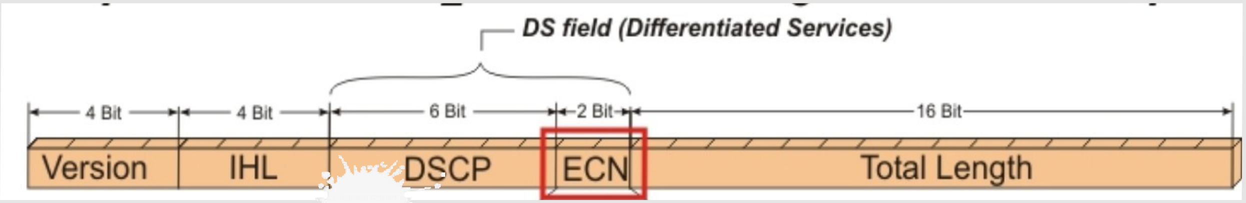


- Key to congestion control for low queuing delay
  - 0.5 ms (vs. 5-15 ms) over public Internet
- Applicability: (see spare slide)

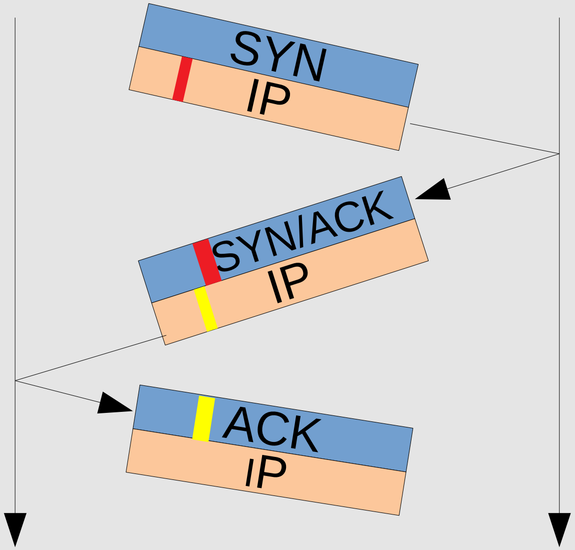
# Fall-back if IP/ECN bleached/mangled

- 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

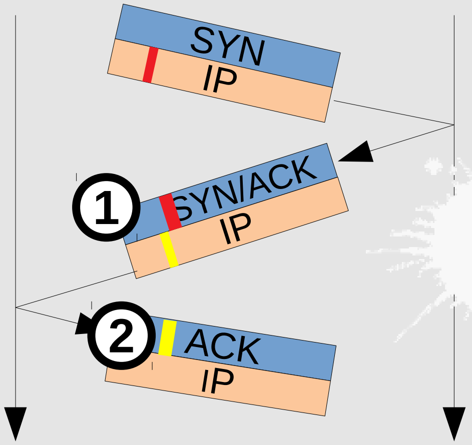


- Solution: Feed back (in the 3 TCP/ECN flags) which of 4 possible IP/ECN codepoints arrived on:
    - – SYN : in SYN-ACK
    - – SYN/ACK : in ACK of 3WHS
        - (With TFO, this ACK is not reliably delivered)
- If mangled, disable ECN for half connection



* 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 |

# Change Triggered ACKs

- SHOULD → “MUST with get-out clause”
- So that receiver can rely on the behaviour
    - e.g. at flow-start when heuristics waste valuable time

> “A concern has been raised that certain offload hardware needed for high performance might not be able to support change-triggered ACKs, although high performance protocols such as DCTCP successfully use change-triggered ACKs.
>
> One possible experimental compromise would be for the receiver to heuristically detect whether the sender is in slow-start, then to implement change-triggered ACKs in software while the sender is in slow-start, and offload to hardware otherwise.
>
> If the operator disables change-triggered ACKs, whether partially like this or otherwise, the operator will also be responsible for ensuring a co-ordinated sender algorithm is deployed;”

# Minor Edits

- Clarified that AccECN is not dependent on ECN (of whatever flavour) in the network
- Experiment success criteria: added “deployed”
- Clarified that ‘Congestion Window Reduced’ signal is not used
- Defined behaviours for all unused values (forward compatibility)

# Status & Next Steps

- Implemented in Linux[1]
- All open issues now closed
  - Appendix B “Alternative Design Choices” DELETED
  - Appendix C “Open Protocol Design Issues” DELETED
- Ready for WGLC

[1] https://github.com/mirjak/linux-accecn

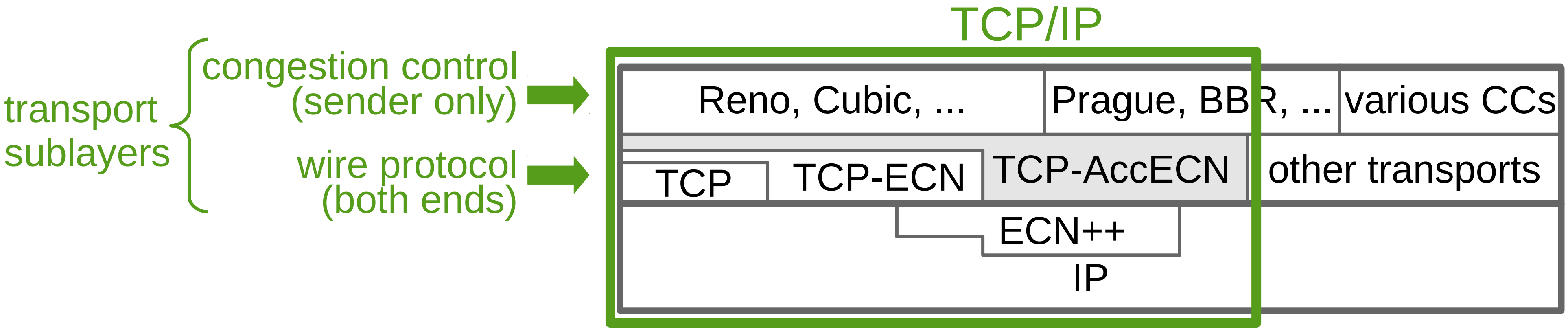
# AccECN

# Q&A

spare slides

# Where AccECN Fits

- Can only enable AccECN if both TCP endpoints support it (1)
  - but no dependency on network changes
- Extends the feedback part of TCP wire protocol
- Foundation for new sender-only changes (and for existing TCP), e.g.
  - congestion controls (TBA):
    - 'TCP Prague' for L4S (2)
    - BBR+ECN
  - Full benefit of ECN-capable TCP control packets (ECN++) (3)



TCP/IP

transport sublayers

congestion control (sender only)

wire protocol (both ends)

Reno, Cubic, ...

Prague, BBR, ...

various CCs

TCP

TCP-ECN

TCP-AccECN

other transports

ECN++

IP

(1) Backwards compatible handshake

- SYN: offer AccECN
  SYN-ACK can accept AccECN, ECN or non-ECN

(2) Low Latency Low Loss Scalable throughput [draft-ietf-tsvwg-l4s-arch]

(3) Without AccECN, benefit of ECN++ excluded from SYN [draft-ietf-tcpm-generalized-ecn]