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

<table>
<thead>
<tr>
<th>IP-ECN</th>
<th>Codepoint</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>not-ECT</td>
<td>No ECN</td>
</tr>
<tr>
<td>10</td>
<td>ECT(0)</td>
<td>ECN-Capable Transport</td>
</tr>
<tr>
<td>01</td>
<td>ECT(1)</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>CE</td>
<td>Congestion Experienced</td>
</tr>
</tbody>
</table>

- 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

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

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

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

---

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>SYN A-&gt;B</th>
<th>SYN/ACK B-&gt;A</th>
<th>Feedback Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccECN</td>
<td>AccECN</td>
<td>1 1 1</td>
<td>0 1 0</td>
<td>AccECN (Not-ECT on SYN)</td>
</tr>
<tr>
<td>AccECN</td>
<td>AccECN</td>
<td>1 1 1</td>
<td>0 1 1</td>
<td>AccECN (ECT1 on SYN)</td>
</tr>
<tr>
<td>AccECN</td>
<td>AccECN</td>
<td>1 1 1</td>
<td>1 0 0</td>
<td>AccECN (ECT0 on SYN)</td>
</tr>
<tr>
<td>AccECN</td>
<td>AccECN</td>
<td>1 1 1</td>
<td>1 1 0</td>
<td>AccECN (CE on SYN)</td>
</tr>
<tr>
<td>AccECN</td>
<td>Nonce</td>
<td>1 1 1</td>
<td>1 0 1</td>
<td>classic ECN</td>
</tr>
<tr>
<td>AccECN</td>
<td>ECN</td>
<td>1 1 1</td>
<td>0 0 1</td>
<td>classic ECN</td>
</tr>
<tr>
<td>AccECN</td>
<td>No ECN</td>
<td>1 1 1</td>
<td>0 0 0</td>
<td>Not ECN</td>
</tr>
<tr>
<td>AccECN</td>
<td>Broken</td>
<td>1 1 1</td>
<td>1 1 1</td>
<td>Not ECN</td>
</tr>
</tbody>
</table>

2. ACE on ACK of SYN/ACK
   - IP-ECN codepoint on SYN/ACK inferred by server
   - Initial s.scep of server in AccECN mode

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

(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]