ECN++: Adding ECN to TCP Control Packets
draft-ietf-tcpm-generalized-ecn-03

Bob Briscoe, CableLabs
Marcelo Bagnulo, UC3M

TCPM WG, IETF-103, Nov 2018
## ECN++ Recap

<table>
<thead>
<tr>
<th>TCP packet type</th>
<th>RFC3168</th>
<th>ECN++ [draft-ietf-tcpm-generalized-ecn-03]</th>
<th>congestion response</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AccECN f/b negotiated</td>
<td>RFC3168 f/b negotiated</td>
<td></td>
</tr>
<tr>
<td>SYN¹</td>
<td>not-ECT</td>
<td>ECT</td>
<td>not-ECT</td>
</tr>
<tr>
<td>SYN-ACK</td>
<td>not-ECT</td>
<td>ECT</td>
<td>ECT</td>
</tr>
<tr>
<td>Pure ACK</td>
<td>not-ECT</td>
<td>ECT</td>
<td>not-ECT</td>
</tr>
<tr>
<td>Window probe</td>
<td>not-ECT</td>
<td>ECT</td>
<td>ECT</td>
</tr>
<tr>
<td>FIN</td>
<td>not-ECT</td>
<td>ECT</td>
<td>ECT</td>
</tr>
<tr>
<td>RST</td>
<td>not-ECT</td>
<td>ECT</td>
<td>ECT</td>
</tr>
<tr>
<td>Re-XMT</td>
<td>not-ECT</td>
<td>ECT</td>
<td>ECT</td>
</tr>
<tr>
<td>Data</td>
<td>ECT</td>
<td>ECT</td>
<td>ECT</td>
</tr>
</tbody>
</table>

¹ For SYN, 'negotiated' means requested
² Obviously only in AccECN case
We thought we'd finished...

- Editorial issues:
  1) Separate: AccECN vs. RFC3168 f/b negotiated

- Technical issues:
  2) Response to CE on Pure ACK
  3) New ECN++ measurement study: dire
  4) Widened scope: *receiver* packet validation / acceptance

CE = Congestion Experienced
Dependence of ECN++ on AccECN experiment

• Problem: unclear which parts of ECN++ draft to follow
  – if you choose not to implement AccECN
  – if AccECN expriment evolved to something different

• Proposed solutions ranged across:
  • Split into 2 near-identical drafts
  • Appendix explaining what depends on AccECN

• Solution
  • Divided the SYN & Pure ACK sections for each case
  • Flagged which case at start of each sub-sub-section
Pure ACK Congestion Response (1/2)

• Problem:
  • Now the sender knows about congestion on ACKs, how does it respond?

• Congestion response specifics out of scope
  • Where draft can say 'usual cwnd/IW response' it does (see table)
  • If it can't (Pure ACK), specifics ought to be defined for each congestion control [Reno, Cubic, BBR, DCTCP]
  • But we ought to give some (informational) guidance in this draft
Pure ACK Congestion Response (2/2)

- A CE-marked Pure ACK is part of an aggregate causing congestion; e.g.
  1) other data flow(s) in parallel to the ACKs
  2) data and ACKs interspersed in one flow
  3) or purely Pure ACK congestion

- Suggest two potential responses (informative only):
  - Optionally AckCC [RFC5690]
  - Reduce cwnd proportional to: \((CE\text{-marked header bytes} + CE\text{-marked data bytes})\)
    \((all\ header\ bytes + all\ data\ bytes)\)

- Deals reasonably with all three scenarios:
  - 1) & 2) cwnd reduction scaled down by 40/1500 (say)
  - 1) & 3) cwnd reduction has no effect on the pure ACKs

- Addresses “it's wrong to do nothing” concern
  - even tho current TCP does nothing if a Pure ACK is lost
Network mangling nil; Server mangling 84


- Of the 82% of servers that now support ECN,
  - 84% disable ECN for the connection if they receive an ECT SYN

- Traced to May 2012 **Linux patch** (and other OSs?):
  
  ```
  % RFC3168 : 6.1.1: SYN packets must not have ECT/ECN bits set.
  % If we receive a SYN packet with these bits set,
  % it means a network is playing bad games with TOS bits.
  % In order to avoid possible false congestion notifications,
  % we disable TCP ECN negotiation.
  ```

- The draft calls this the 'Contra-Postel' ECN test...
The Contra-Postel ECN Test – getting code fixed

• Ironic: this form of network mangling of ECN is non-existent, but servers disable ECN in their attempt to detect it
  • drastic action based on 1-ended inference of a codepoint transition
  • and silent – no logging of the 'problem' to get it fixed

• Recommendations
  1) Remove the Contra-Postel ECN test:
     • while deploying AccECN on servers
       – replaces 1-ended with 2-ended test for mangling
     • while deploying ECN++ on servers
     • just remove it from Linux ECN code

  2) Add client cache work-round (next slide)

  3) Fix the specs (subsequent slide)
Workround: client cache of server support for ECT on SYN (size-capped)

If client implements AccECN, three caching strategies:

- S1: Pessimistic ECT and cache successes
- S2a): Optimistic ECT, no cache
- S2b) Optimistic ECT and cache failures

If client doesn't implement AccECN, no ECT on SYN anyway
The Contra-Postel ECN Test – fixing the specs

- **RFC3168**: “A host MUST NOT set ECT on SYN or SYN-ACK packets.”
- **RFC8311 adds**: “…unless otherwise specified by an Experimental RFC…”
- What does a server do if it receives non-zero ECN on SYN?
  - RFC 3168: Silence
  - RFC 8311: Silence
  - Silence → Postel's Robustness Principle: “...be liberal in what you accept”?
- **ECN++ draft adds**: “In order for this experiment to be useful, the following requirements follow from RFC8311:
  - Any TCP implementation SHOULD accept receipt of any valid TCP control packet or retransmission irrespective of its IP/ECN field. If any existing implementation does not, it SHOULD be updated to do so.
  - A TCP implementation taking part in the experiments proposed here MUST accept receipt of any valid TCP control packet or retransmission irrespective of its IP/ECN field.”
Receiver packet validation / acceptance

• Original scope of ECN++ draft:
  • Solely behaviour of sender of a control pkt
  • Some recommended Receiver-side packet validation checks had been muddled in with Sender-side requirements

• Widened scope:
  • Added specific receiver acceptance guidance for ECN on each type of control packet (previous slide)
  • Warranted separating out a Receiver-side section

• ECN++ is still a sender-only deployment
Next Steps

- Really have finished now
- Closed off all open issues
  1) Separate: AccECN vs. RFC3168 f/b negotiated
  2) Response to CE on Pure ACK
  3) Contra-Postel ECN test
  4) Widened scope: receiver packet validation / acceptance
- WGLC