More Accurate ECN Feedback in TCP (AccECN)
draft-kuehlewind-tcpm-accurate-ecn-03

Bob Briscoe, BT
Richard Scheffenegger, NetApp
Mirja Kühlewind, Stuttgart Uni
IETF-90, Jul’14
Purpose of Talk

• Introduce latest AccECN protocol spec
  – awesome protocol design (IMHO)
  – satisfies numerous conflicting requirements
    • except not as simple as we’d have liked 🙁

• seeking adoption, expert review and opinions
  – intent: Experimental
  – full spec (38pp) plus pseudocode examples, design alternatives & outstanding issues (+17pp)
  – consensus prior to implementation
The Problem (Recap)

Congestion Extent, not just Existence

• Current ‘classic’ ECN feedback in TCP [RFC3168]
  if (any packet marked in RTT) \{signal 1\}
  else \{signal 0\}

• \(<\text{ironic}>\) Imagine using a 128b field for 2 addresses
  if (any bit set) \{address = 1\}
  else \{address = 0\}
  \(<\text{/ironic}>\)

• Only ECN-for-TCP is so clunky
  – TCP widely uses SACK to identify individual drops
  – modern transports (DCCP, SCTCP, RTP/UDP etc) feed back extent of ECN
  – need to update TCP, in its role as 1 of 2 transport protocols that work

• DCTCP feedback scheme would be nice, but:
  1. new wire protocol but no negotiation
  2. greatly confused by ACK loss
  3. higher congestion → more ACKs

\[\begin{array}{c|c}
0 & 1 \\
\hline
012345678012345678901234567801 & 2 \\
\hline
01234567890123456789012345678901 & 3 \\
\end{array}\]

\[\{0 \mid 1\}\]
a new problem: feedback of bleached ECN

• erasure of ECN field to Not-ECT (00) in transit
  • RFC3168 notes that this could happen
  • and says it would be very bad
  • but doesn’t say what to do about it

• if Not-ECT arrives at a classic ECN receiver
  • it does nothing, and can do nothing

• some tests show that bleaching ECN is common

• AccECN now includes Not-ECT feedback
Where to find spare bits?

• Satisfied requirements with zero extra bits
  – **essential** part: overloaded 3 existing ECN flags in main TCP header
  – **supplementary** part: overloaded 15b in Urgent Pointer when redundant

• Non-Zero Urgent Pointer when TCP URG flag = 0?
  – middlebox traversal
    • seems better than for new TCP options in initial tests*
  – opportunistic – not available when URG = 1
    • not useful for most other protocols that need more bits

---

* Perhaps because earlier Windows versions did not zero the Urgent Pointer when URG=0
2 complementary signals

- After successful capability negotiation

1. cumulative counters of the 3 ECN codepoints
2. the sequence of ECN codepoints covered by each delayed ACK

- note: packet-based not byte-based counters
- note: pure ACKs are not counted
  (there are deep questions behind both these points)
Protocol Design

Capability Negotiation

• AccECN is a change to TCP wire protocol
  • only to be used if both ends support it
• client negotiates support on initial SYN
  • using the 3 ECN-related TCP flags
  • server sets the 3 flags accordingly on the SYN/ACK
    – or it replies as the latest variant it recognises
  • if nec. client downgrades to match the server

• supp. field not used until 3rd leg of handshake
  • consumes no TCP option space on SYN
  • if at any time supp. field = 0 → middlebox interference
Cumulative ECN Codepoint Counters

after SYN/ACK

- Data receiver counts arriving CE, ECT(1) & Not-ECT (11, 01 & 00)*
- Selects one counter to feed back in each ACK
  - encodes in the ACE field, overloading the 3 ECN flags
  - encoding fits a base 4, base 3 and base 1 counter in 3 bits!

- includes 4 most significant bits of the selected counter in the supp. field

* ECT(0) found from remainder and from sequence field if available
ECN Sequence covered by each Delayed ACK

- **ECN Sequence (ESQ) field**
  - encodes 2 Run-Longths of SPaces, each ending in one possibly different Mark

- **Value of ACE selects MK2 (no need to encode in ESQ)**

- **Receiver sends a Delayed ACK on any of these events:**
  a) Max delayed ACK coverage is reached (e.g. 2 full-sized segments)
  b) Delayed ACK timer expires (e.g. 500ms)
  c) Pattern becomes too complex to encode

- **in one ACK, it is possible to encode a sequence of:**
  - up to 15 segments for typical marking patterns
  - 3 segments for any possible marking pattern
# AccECN Protocol Features Summary

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Classic ECN</th>
<th>ECN Nonce</th>
<th>DCTCP</th>
<th>AccECN Urg-Ptr</th>
<th>AccECN TCP opt</th>
<th>AccECN essential</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resilience</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>o</td>
</tr>
<tr>
<td>Timeliness</td>
<td>o</td>
<td>o</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Integrity</td>
<td>-</td>
<td>o</td>
<td>+*</td>
<td>+*</td>
<td>+*</td>
<td>+*</td>
</tr>
<tr>
<td>Accuracy</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Ordering</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Complexity</td>
<td>++</td>
<td>+</td>
<td>o</td>
<td>-</td>
<td>-</td>
<td>o</td>
</tr>
<tr>
<td>Overhead</td>
<td>++</td>
<td>o</td>
<td>o</td>
<td>+</td>
<td>0</td>
<td>++</td>
</tr>
<tr>
<td>Compatibility</td>
<td>o</td>
<td>o</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>o</td>
</tr>
</tbody>
</table>

* = compatible with an independent zero-overhead integrity solution
Opportunistic but not Presumptuous?

- Presumptuous to reassign Urgent Pointer experimentally?
- While experimental:
  - use a TCP option for the supplementary part
  - Reserved 15b in Urgent Pointer
    - to use if this progresses to standards track
  - Experimental implementations required to recognise either location
- AccECN still ‘works’ if TCP option is cleared or discarded
Interaction with other TCP variants

- Server can use AccECN with SYN Cookies
  - capability negotiation can be inferred
- AccECN compatible with main TCP options:
  - Max Segment Size (MSS)
  - Timestamp
  - Window Scaling
  - Selective ACKs (SACK)
  - Authentication Option (TCP-AO)
  - TCP Fast Open (TFO)
  - Multipath TCP (MPTCP)
- AccECN consumes no option space on the SYN
  — even when deployed experimentally as a TCP option
Open Design Issues

1. Could simplify by removing sequence (ESQ) feedback entirely?
   – Instead require the receiver to disable delayed ACKs?
     • during slow-start (Linux receiver does this heuristically)?
     • requested by the sender?
   – But, is ACKing every segment acceptable?

2. Could simplify by using Urgent Pointer for experimental protocol?
   • See Appendix C of draft, for these and 7 other more detailed issues
Alternative Design Choices

Roughly highest importance first

- Earlier ECN feedback (on SYN/ACK)
- Remote Delayed ACK Control
- Earlier ECN fall-back (on SYN/ACK)
- Shave 1 bit off ECN sequence field

See Appendix B of draft
summary & next steps

- awesome protocol design (IMHO)
  - capability negotiation and 3 counters in 7b
    - even works in 3b, if middlebox clears other 4b
  - sequence of up to 15 x 4 codepoints in 10b
    - most likely of $2^{30}$ combinations in a $2^{10}$ space
  - zero (extra) header bits

- still room for improvement
  - draft written to support consensus process
  - fully specified protocol, but also...
  - a container for design alternatives & issues

- adoption call please
More Accurate ECN Feedback in TCP (AccECN)

Requirements

draft-ietf-tcpm-accecn-reqs-06

Proposed Protocol Spec

draft-kuehlewind-tcpm-accurate-ecn-03

Q&A

spare slides
ECN Sequence covered by each Delayed ACK

- **SPace or MarK1** can be any of:
  - N: Not-ECT (00)
  - 0: ECT(0) (10)
  - 1: ECT(1) (01)
  - C: CE (11)

- **Examples**

  a) \[1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \]
  
  b) \[0 \ 0 \ C \ C \ C \]
  
  c) \[0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \]
  
  d) \[C \ 0 \ 0 \ 0 \ 0 \ 0 \ C \]
  
  e) \[N \ N \]
Protocol Features

detailed explanations

• Resilience
  • DCTCP confused by ACK loss

• Timeliness
  • Classic ECN: only timely once per RTT
  • DCTCP is always 1 transition behind

• Integrity
  • ECN nonce: relies on receiver incriminating itself
  • DCTCP & AccECN compatible with approach in draft-moncaster-tcpm-rcv-cheat

• Accuracy
  • DCTCP lack of resilience impacts accuracy

• Ordering
  • ‘AccECN essential’ is the fall-back when a middlebox clears the sequence field

• Complexity
  • Hard to quantify

• Overhead
  • ECN Nonce marked down because it consumes the last ECN-IP codepoint
  • AccECN Urg-Ptr marked down because it prevents others using the Urgent Pointer

• Compatibility
  • Class ECN has had continuing problems with middlebox traversal
  • DCTCP is unsafe to interoperate with other TCP variants
  • ‘AccECN Urg-Ptr’ seems good at traversal, but more experiments needed
  • ‘AccECN TCP opt’ unlikely to traverse middleboxes that wipe TCP options

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Classic ECN</th>
<th>ECNNonce</th>
<th>DC TCP</th>
<th>AccECN Urg-Ptr</th>
<th>AccECN TCP opt</th>
<th>AccECNessential</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resilience</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>o</td>
</tr>
<tr>
<td>Timeliness</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Integrity</td>
<td>-</td>
<td>0</td>
<td>+*</td>
<td>+*</td>
<td>+*</td>
<td>+*</td>
</tr>
<tr>
<td>Accuracy</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Ordering</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Complexity</td>
<td>++</td>
<td>+</td>
<td>o</td>
<td>-</td>
<td>-</td>
<td>o</td>
</tr>
<tr>
<td>Overhead</td>
<td>++</td>
<td>0</td>
<td>o</td>
<td>+</td>
<td>0</td>
<td>++</td>
</tr>
<tr>
<td>Compatibility</td>
<td>o</td>
<td>o</td>
<td>-</td>
<td>o</td>
<td>-</td>
<td>o</td>
</tr>
</tbody>
</table>