

# An Enabler for New E2E Transport Behaviours: More Accurate ECN Feedback Reflector (AccECN)

draft-kuehlewind-tcpm-accurate-ecn-03



Bob Briscoe, BT

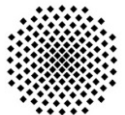


Richard Scheffenegger, NetApp



Mirja Kühlewind, Stuttgart Uni

IETF-90, Jul'14

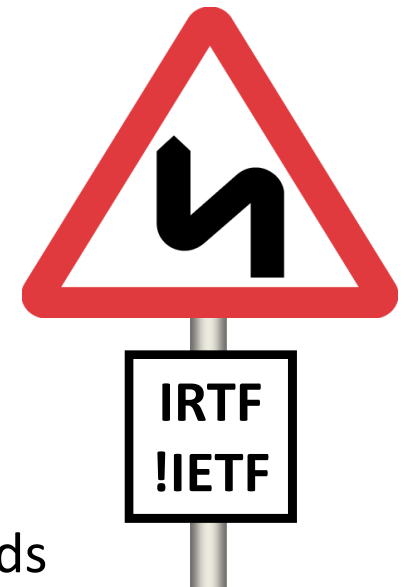


Universität  
Stuttgart

Bob Briscoe's work is part-funded by the European Community under its Seventh Framework Programme through the Reducing Internet Transport Latency (RITE) project (ICT-317700) and through the Trilogy 2 project (ICT-317756)

# Purpose of Talk

- Introduce feedback reflector for ECN in TCP
  - awesome protocol design (IMHO)
- Status of AccECN in IETF
  - Requirements draft finished w-g last call 21-Jul-14
- Once-in-a-decade (?) chance for change
  - TCP feedback meant to be generic, not just for known needs
  - Once rcvr deployed, new sndr behaviours can be deployed unilaterally
  - Would it meet the needs of your (research) ideas?
  - Would it suit your ideas better with some tweaks?



# The Problem (Recap)

## Congestion Extent, not just Existence

- Current 'classic' ECN feedback in TCP [RFC3168]

```

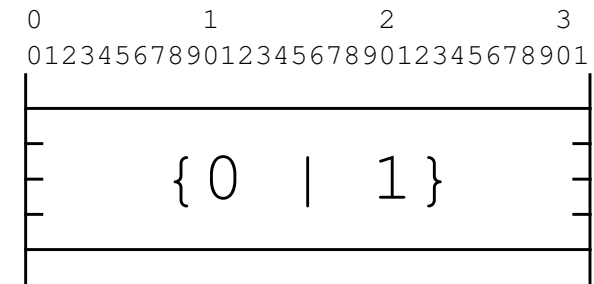
if (any packet marked in RTT)    {signal 1}
else                               {signal 0}
  
```

- **<ironic>** Imagine using a 128b field for 2 addresses

```

if (any bit set)    {address = 1}
else                {address = 0}
  
```

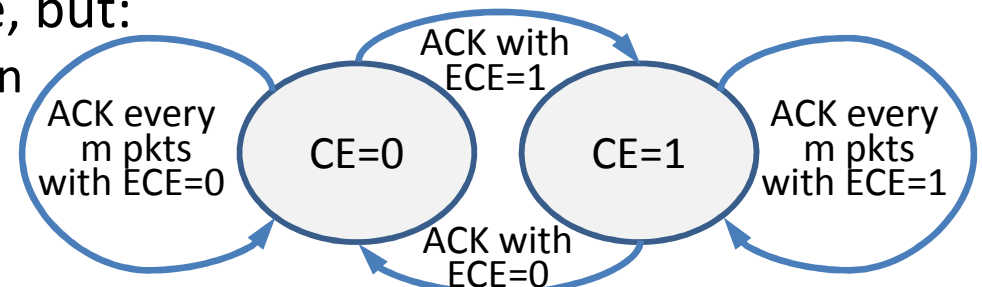
**</ironic>**



- 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

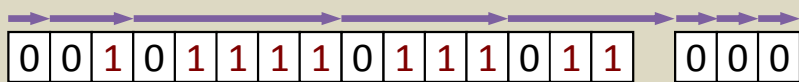


# Usefulness

- Low latency queuing signals:

1. Data Center TCP (DCTCP) [widely used in private networks]
  - existing feedback scheme fails if ACKs dropped
2. Congestion Exposure (ConEx) [IETF]
  - existing TCP sufficient for drop feedback, but not for ECN
3. Variable-structure congestion Control Protocol (VCP) [SIGCOMM'05]
4. Chirping – TCP Rapid [Infocom'09] →
5. Very low threshold ECN marking [research to appear]
6. Queue View (QV) [research to appear]

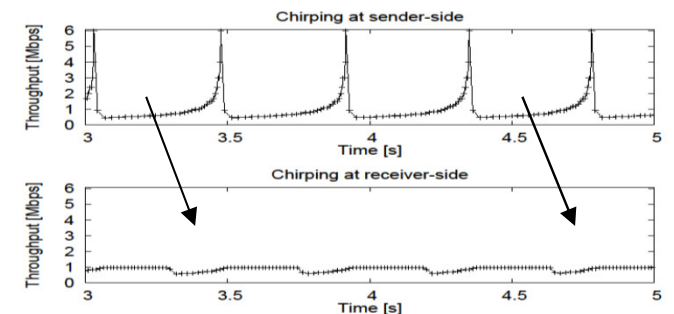
Spacing between zeros is inst. queue length



Legend: 0 ECT(0) 1 ECT(1)

- Allows sender to safely set CE as a test

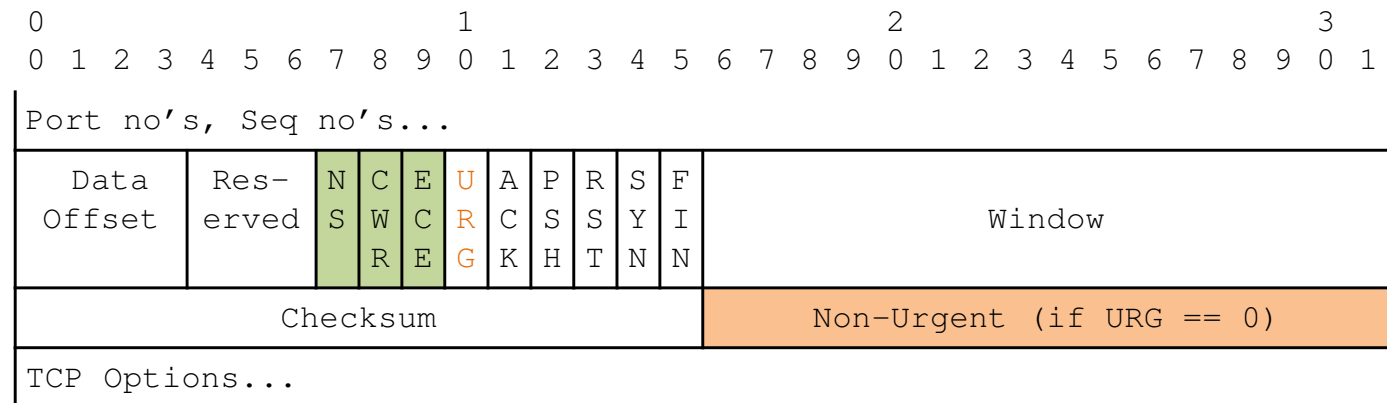
7. ECN path testing [draft-kuehlewind-tcpm-ecn-fallback]
8. Receiver cheat detection [draft-moncaster-tcpm-rcv-cheat]
  - Otherwise, with classic ECN, each test would conceal a whole window



# Protocol Design I

## 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

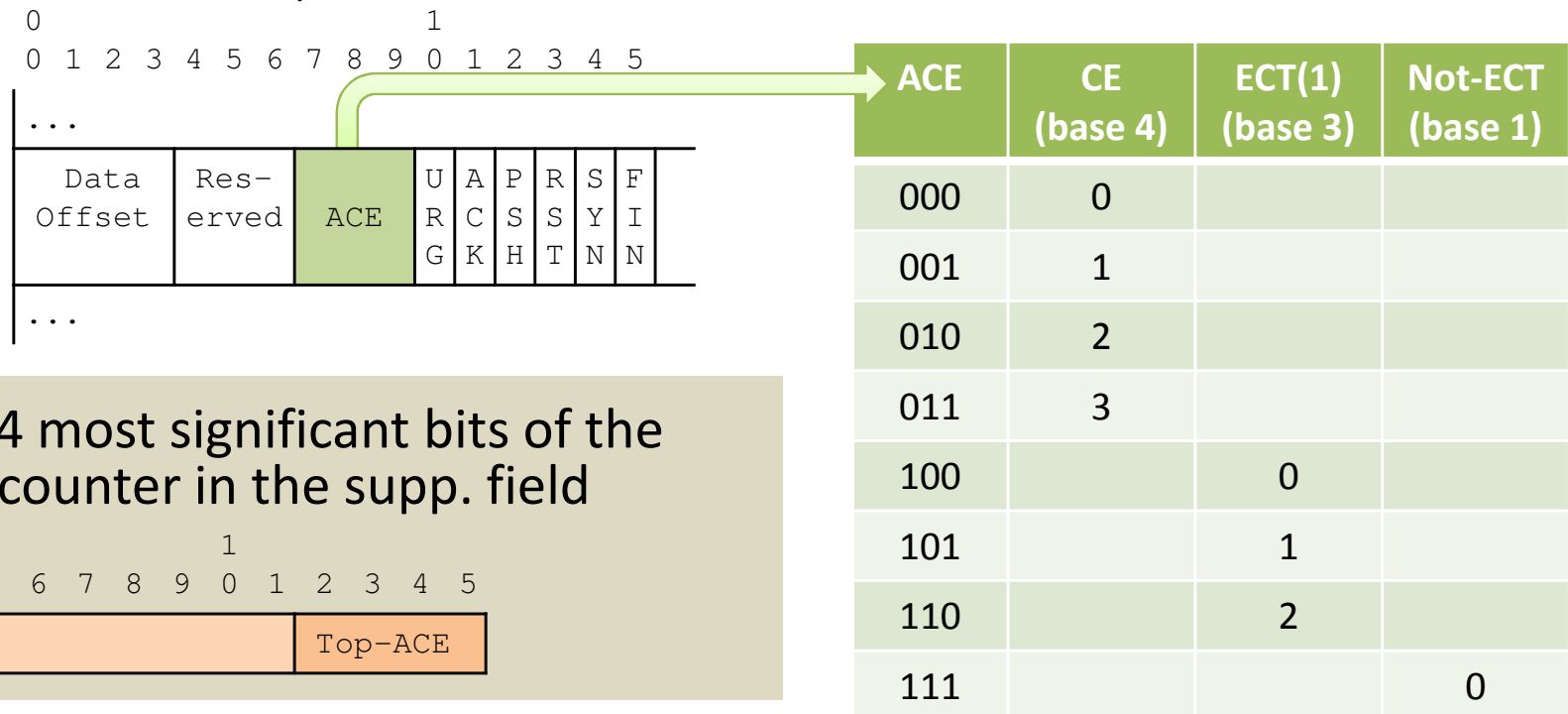
## 2 complementary signals

- 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)

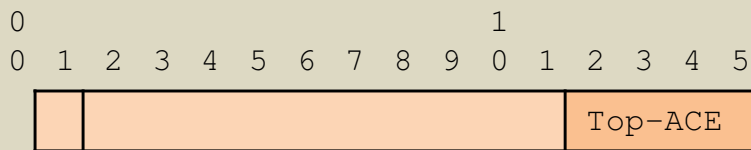
## Cumulative ECN Codepoint Counters

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!

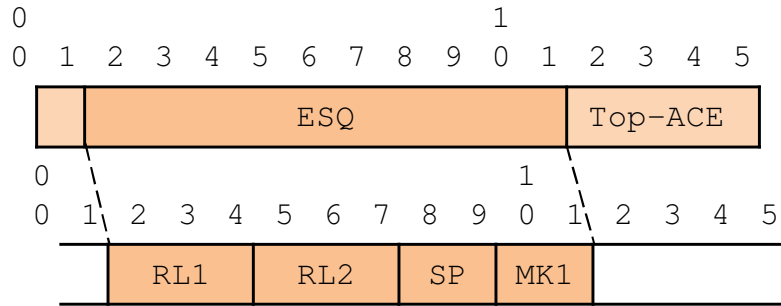


– includes 4 most significant bits of the selected counter in the supp. field

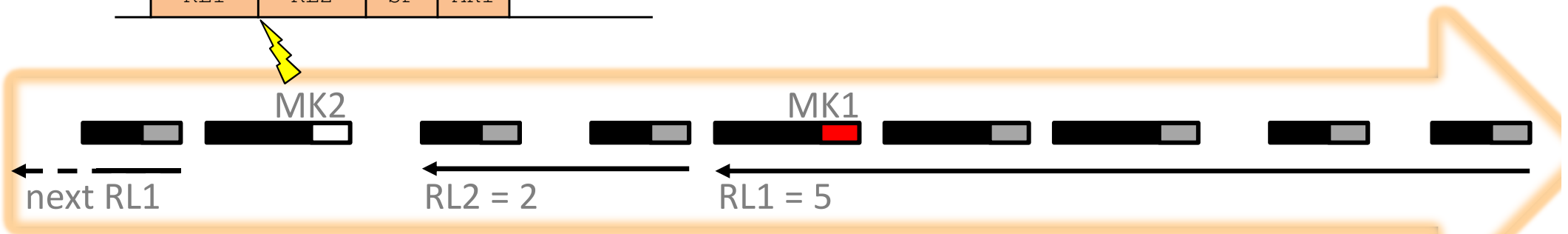


\* ECT(0) found from remainder and from sequence field if available

## ECN Sequence covered by each Delayed ACK



- ECN Sequence (ESQ) field
- encodes 2 **R**un-**L**engths of **S**Paces, each ending in one possibly different **M**ark



- 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

Examples





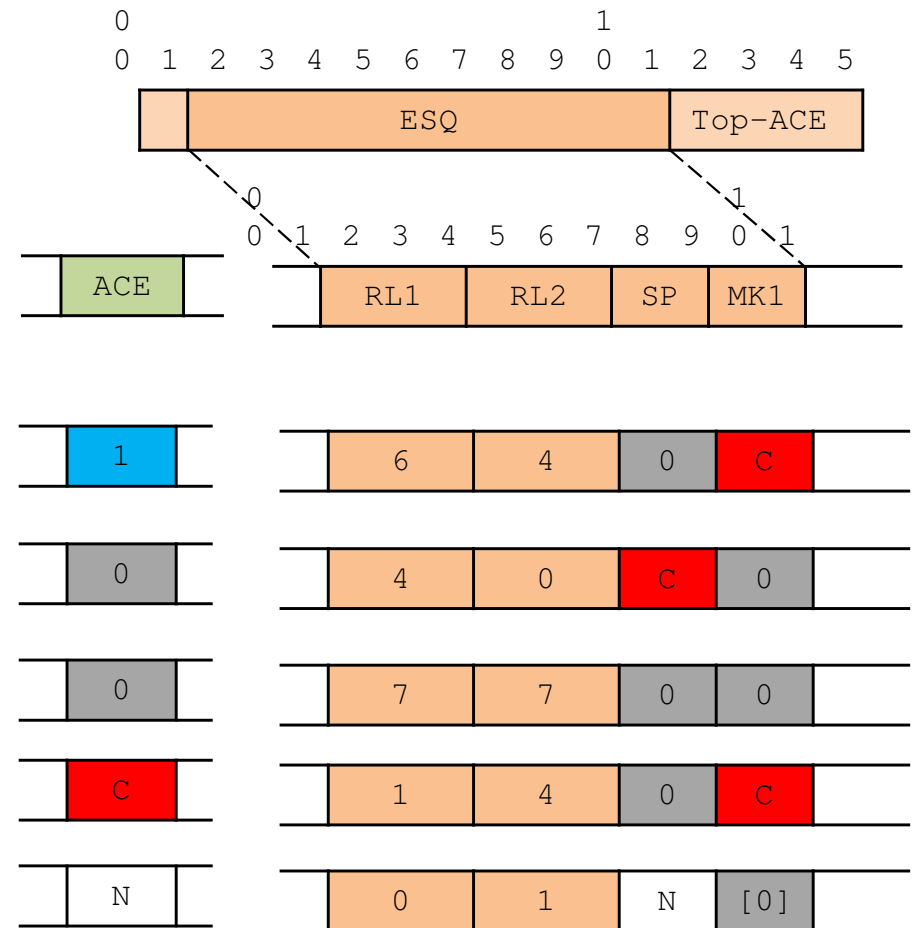
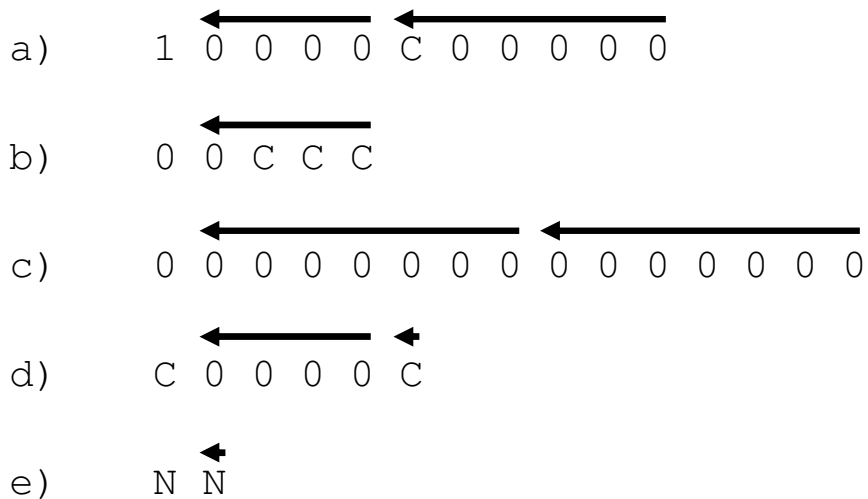
# Protocol Design VI

## 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



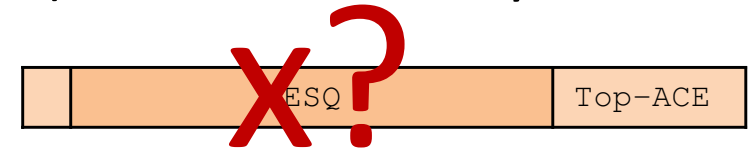
# AccECN Protocol Features Summary

| Requirement   | Classic ECN | ECN Nonce | DCTCP | AccECN Urg-Ptr | AccECN TCP opt | AccECN essential |
|---------------|-------------|-----------|-------|----------------|----------------|------------------|
| Resilience    | +           | +         | -     | +              | +              | 0                |
| Timeliness    | 0           | 0         | -     | +              | +              | +                |
| Integrity     | -           | 0         | -     | +*             | +*             | +*               |
| Accuracy      | -           | -         | -     | +              | +              | +                |
| Ordering      | -           | -         | +     | +              | +              | -                |
| Complexity    | ++          | +         | 0     | -              | -              | 0                |
| Overhead      | ++          | 0         | 0     | +              | 0              | ++               |
| Compatibility | 0           | 0         | -     | 0              | -              | 0                |

\* = compatible with an independent zero-overhead integrity solution

# Open Design Issues

1. Could simplify by removing sequence (ESQ) feedback entirely?

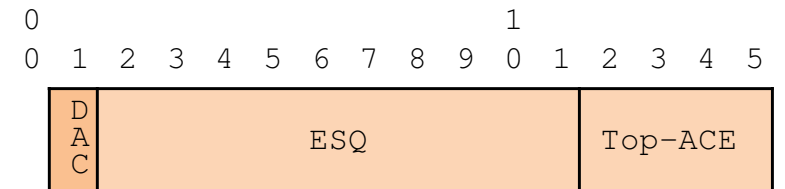


- Instead require the receiver to disable delayed ACKs?
    - during slow-start (Linux receiver does this heuristically)?
    - requested by the sender?
  - But, is ACKing every segment acceptable?
2. if supp. field unavailable, certain counter(s) could rarely be seen
    - if they are rarely last in a delayed ACK (maybe due to unfortunate synch.)
- 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



where to draw the line?

See Appendix B of draft

# summary & next steps

| Requirement   | AccECN<br>Urg-Ptr |
|---------------|-------------------|
| Resilience    | +                 |
| Timeliness    | +                 |
| Integrity     | +                 |
| Accuracy      | +                 |
| Ordering      | +                 |
| Complexity    | -                 |
| Overhead      | +                 |
| Compatibility | 0                 |

- awesome protocol design (IMHO)
  - zero (extra) header bits
- still room for improvement
  - pls ponder design alternatives & issues (Appendices B & C)
  - simple enough for high performance?
  - generic enough for your research ideas?

# An Enabler for New E2E Transport Behaviours: More Accurate ECN Feedback Reflector (AccECN)

Requirements

[draft-ietf-tcpm-accecn-reqs-06](#)

Proposed Protocol Spec

[draft-kuehlewind-tcpm-accurate-ecn-03](#)

Q&A

spare slides

## 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 3<sup>rd</sup> leg of handshake
  - consumes no TCP option space on SYN
  - if at any time supp. field = 0 → middlebox interference

SYN

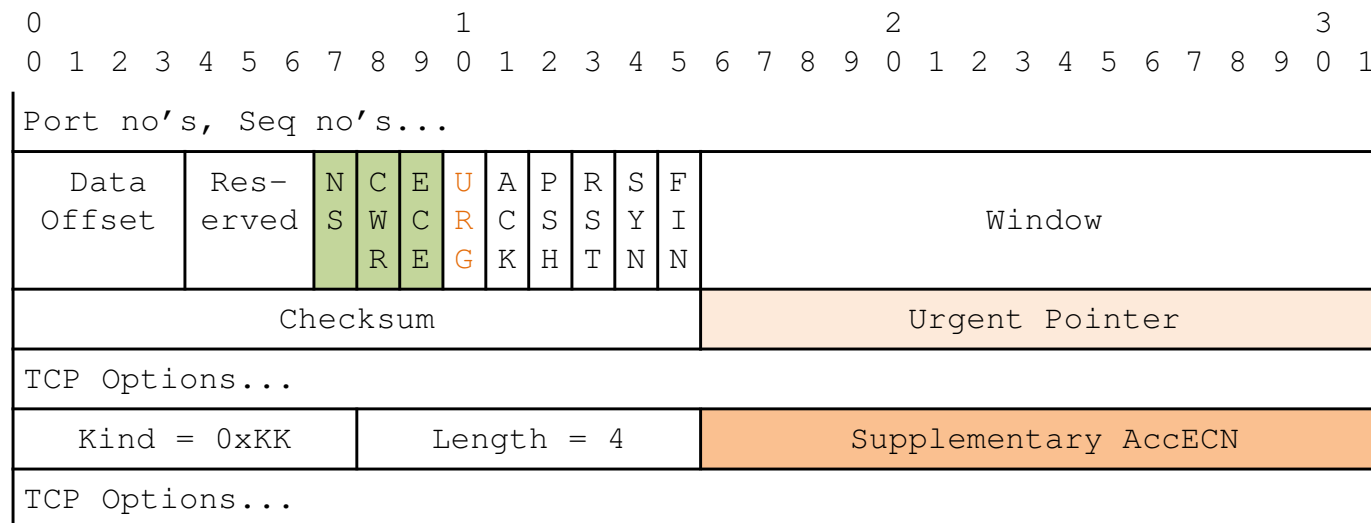
|   |   |   |
|---|---|---|
| N | C | E |
| S | W | C |
| = | = | = |
| 1 | 1 | 1 |

SYN/ACK

|   |   |   |
|---|---|---|
| N | C | E |
| S | W | C |
| = | = | = |
| 0 | 1 | 0 |

# 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

# Protocol Features

## detailed explanations

| Requirement   | Classic ECN | ECN Nonce | DC TCP | AccECN Urg-Ptr | AccECN TCP opt | AccECN essential |
|---------------|-------------|-----------|--------|----------------|----------------|------------------|
| Resilience    | +           | +         | -      | +              | +              | 0                |
| Timeliness    | 0           | 0         | -      | +              | +              | +                |
| Integrity     | -           | 0         | +*     | +*             | +*             | +*               |
| Accuracy      | -           | -         | -      | +              | +              | +                |
| Ordering      | -           | -         | +      | +              | +              | -                |
| Complexity    | ++          | +         | 0      | -              | -              | 0                |
| Overhead      | ++          | 0         | 0      | +              | 0              | ++               |
| Compatibility | 0           | 0         | -      | 0              | -              | 0                |

- 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