Abstract Encoding for Congestion Exposure

Matt Mathis

ConEx WG, IETF 78

We already have a morass

- Major source of complexity is encoding
- Encoding issues often obscure algorithm issues
- Encoding issues bury the simplicity of the ideal
- Assumptions about encoding color our thinking

We need to simplify

- Do the base algorithm design without encoding
  - Understand and inventory potential capabilities
    - Can include variants of the algorithms
- Design encoding as a separate step
  - Choose code points to conflate
  - Can (computationally?) validate:
    - Preserved v lost capabilities
    - Effects of known bugs
      - Remapping codepoints
    - Effects of partial deployment

Congestion Exposure assumptions

- Must support both:
  - ECN based RE-Feedback
  - Loss based RE-Feedback
- Transport protocol does not have to be TCP
  - If I say TCP I really mean transport

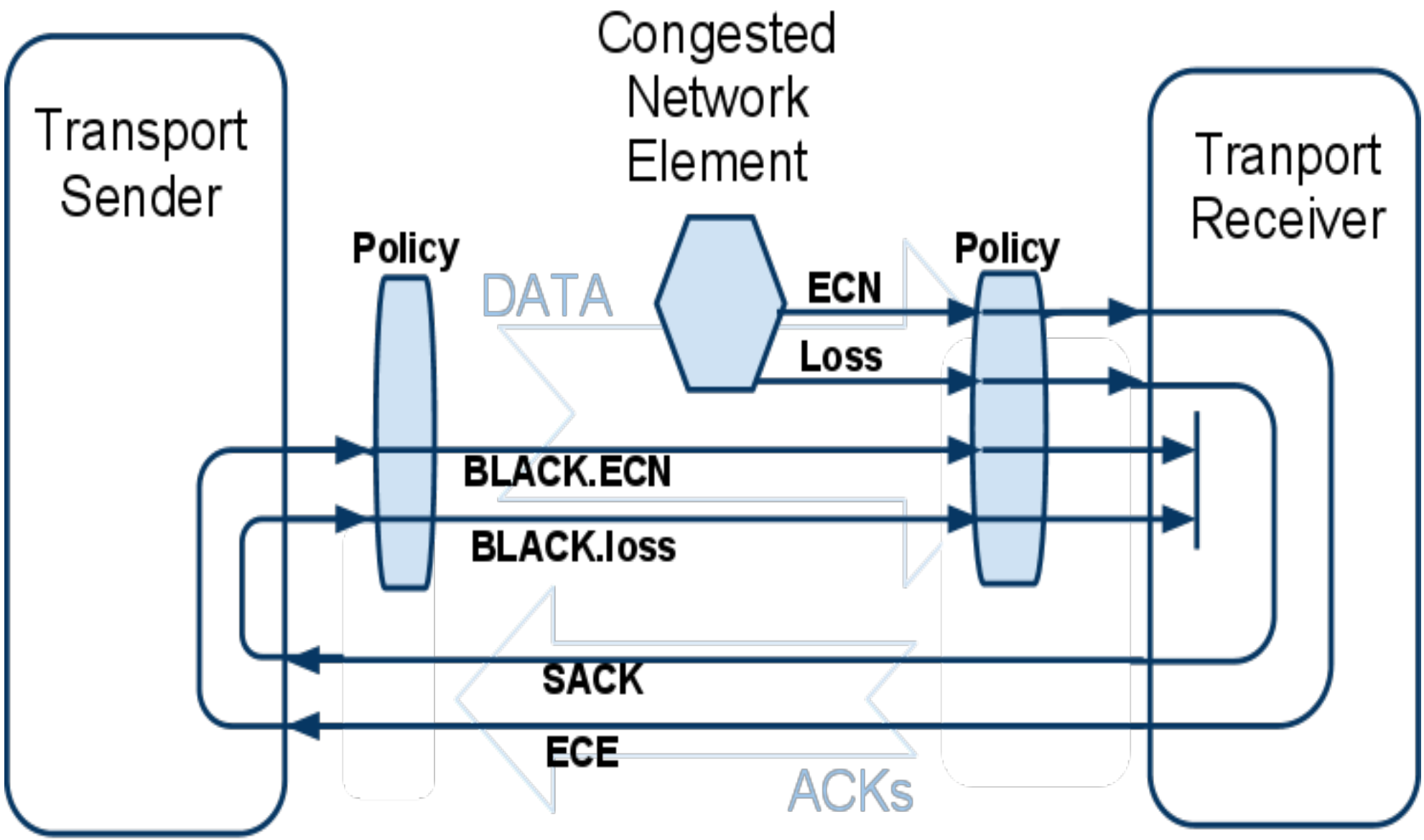
Model Assumptions

- Data flow model
  - Discreet functional building blocks
  - Connected by common signals
  - Complete algorithms built out of assembled blocks
- All signals include explicit "not supported" indication
  - Don't constrain deployment scenarios
- Notation for variants of building blocks:
  BASENAME.varient

Basic signals & functional units

- LOSS - Network bottleneck to transport receiver
  - Default implicit congestion signal
- SACK - Transport receiver to sender loss indications
  - Also include duplicate ACKs
- BLACK.loss - Transport sender to all path elements
  - Exposes retransmissions to the entire path
- ECN - Network bottleneck to Transport receiver
  - Defined by RFC 3581
- ECE - Transport receiver to Transport sender
  - Counter carried by transport
- BLACK.ece - Transport sender to all path elements
  - Exposes ECN marks to the entire path
- GREEN - Transport sender to all path elements
  - Pre credits to facilitate strong enforcement

Basic signals and functional units



LOSS

- Congestion signal Network -> Transport receiver
  - Implicit, default congestion indication
- May use Random Early Detection (LOSS.red)
  - Or drop tail (LOSS.tail)
  - Or something else (LOSS.magic)

SACK (dupACKs)

- Loss information from Transport receiver->sender
  - also include dupACK
  - and any other returned loss signals
- Required part of all reliable protocols
- Required to implement congestion control

BLACK.loss

- Transport sender -> entire path
  - Carries RE-echo'd SACK/loss info
- Also called credit marks
- Indicates the total lossed over the entire path

(Out-of-scope: mental model
  - Mark all retransmissions such that the network
    - Can instrument (count)
    - Can (test) implementing policy

)

ECN: Explicit Congest. Notification

- Network element -> transport receiver
  - Indicates congestion
  - Sometime called Negative, Debit or RED marks
  - The detected congestion is always upstream
- ECN.3168 defined exactly per RFC 3168
  - This fully constrains the encoding
- May consider slightly revising 3168
  - CAUTION greatly raises deployment cost
    - e.g. use different drop probability than losses
    - e.g. redefine one of the ECT code points
    - All others are probably non-starters
      - But should not be forbidden outright

ECE: ECN Echo

- Transport receiver -> transport sender
  - Carries ECN info back to the sender
- ECE.3168 is not really strong enough
  - Only permits one event (signal) per RTT
- Single bit is also too weak
  - Sparse ACKs may not be able to carry enough
  - ACKs might get lost
- More likely implementation:
  - Small ECN counter carried in retuning ACKs
  - Sender can count counter advances
  - Robust to lost ACKs and ACK thinning
    - Up to a point

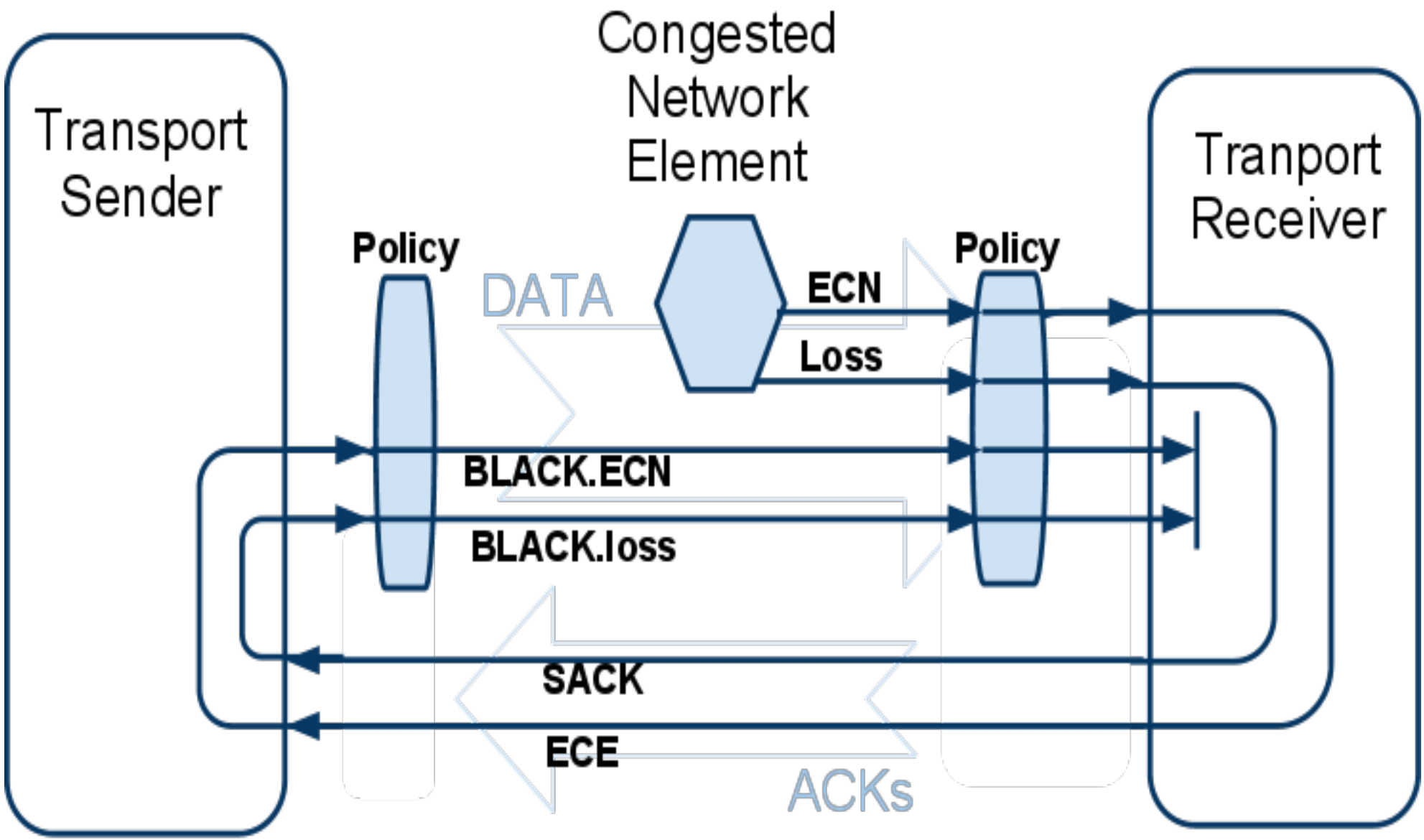
BLACK.ECN

- Transport sender -> entire path
  - RE-echo'd ECN info by way of ECE
- Also called credit
- Indicates the total ECN marks to the entire path
  - But delayed by one RTT

GREEN

- Transport sender -> entire path
  - Similar semantics to BLACK.* aka Credit marks
  - Pre-credits to offset 1 RTT delay in BLACK marks
- GREEN.maxflight
  - Mark every packet that raises MAX(in_flight)
- Assures that GREEN.maxflight+BLACK.ECN-ECN >0
  - For every hop, for all time
  - Strong cheat detection when implemented close to the receiver

Basic signals and functional units



Useful (mid path) observations

- Can compute total congestion for the entire path
  - BLACK.Loss + BLACK.ECN
- Can compute total upstream congestion
  - LOSS (reconstructed state machine) + ECN
- Can compute down stream congestion
  - BLACK.Loss - LOSS (reconstruct state machine)
    - Same as # late duplicate packets
      - e.g. you see both first and retransmit
      - This test is very robust
  - BLACK.ECN - ECN

The encoding problem

- Without any collapsing need 3x3x3x3 code points
  - States: {Unsupported, Off, On}
  - Signals: {ECN, BLACK.ECN, BLACK.Loss, GREEN}
- Key issue is eliminating redundant "unsupported"
  - Simple model:
    - 3 CP to handle ECN
    - 5 CP to handle BLACK* and GREEN
      - All share the same "unsupported"
      - One no credit CP
      - Once CP for each credit
        - Can't represent combined credits
    - Independent "supported" for ECN and RE-echo
    - Still too many bits for IPV4
      - Further conflating possible

Useful deployment observations

- Can make Loss and ECN based systems independent
- Loss based RE-echo may be easy to deploy
  - Just set a "retransmitted" flag in IP layer
    - Tiny patch to existing stacks
    - Auditing cheaters requires reconstructing TCP
      - And may be fragile
    - But good enough to study and validate uses:
      - Instrumentation
      - Policy, etc
  - But what bit? (OFF TOPIC)

Conclusion

- Separate core algorithm design from coding design
  - Core algorithms are really simple
  - Encoding adds huge complexity
    - Tweaking algorithms after encoding hurts
      - Think combinatoric spaghetti
  - Better model:
    - Tweak base algorithms
    - (re)apply encoding

(more, but out of scope)

What bit/CP to tag retransmitted?

- Bit 48 one obvious choice
  - But huge political baggage
- What about redefining ECT(1) as BLACK.Loss?
  - If ECN enabled
    - Send only ECT(0) for ECN enabled
    - Will (rarely) overwrite BLACK.Loss with ECN
      - Only congestion from different bottleneck
  - If ECN disabled
    - Normally send Not-ECT
    - BLACK.Loss looks enabled so ECN might be lost
      - But TCP is already in recovery, so don't care
  - Best part:
    - Hard code (crossing TCP/IP layers) is done!