Robust ECN Signaling with Nonces

Neil Spring, David Wetherall, David Ely
University of Washington
IEEE CCW
October, 2001

ECN gives receivers power

Old congestion signals:

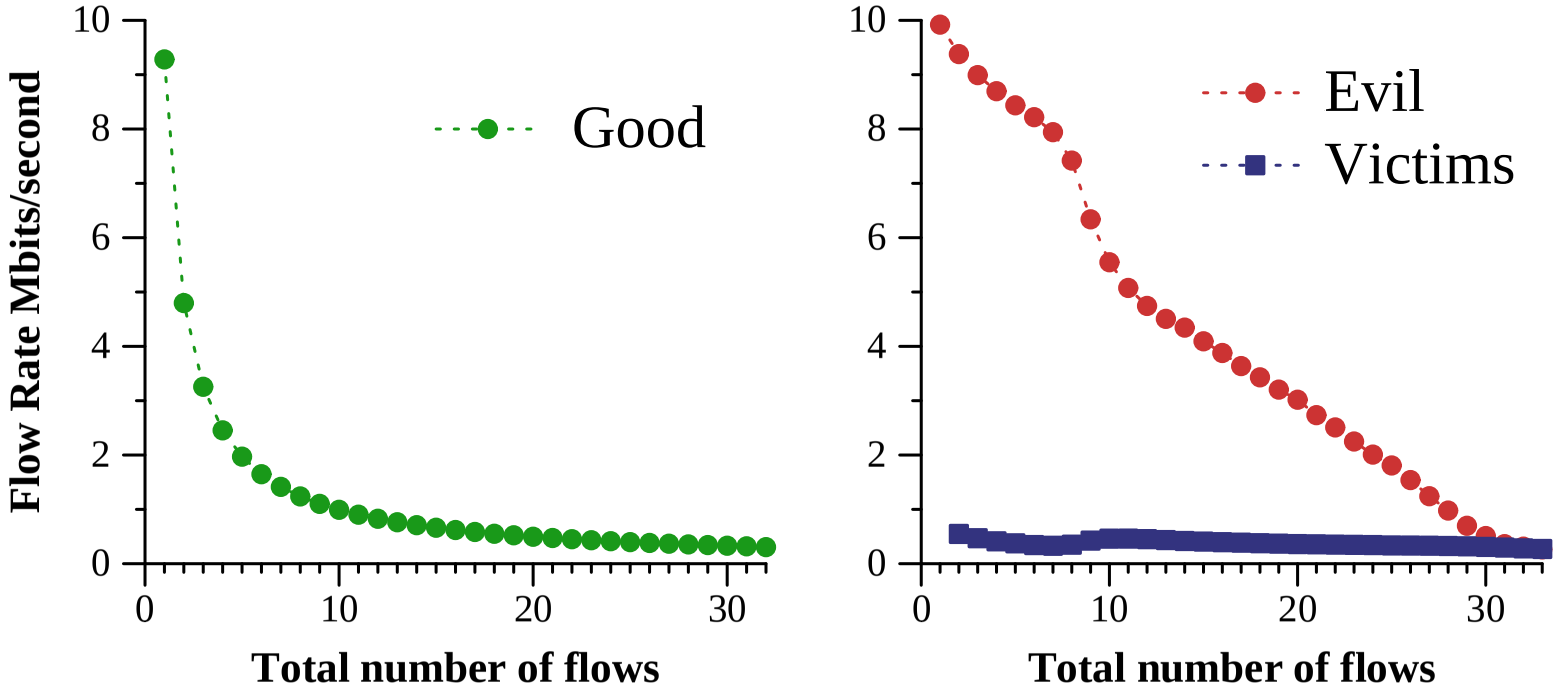
- Receivers claim lost packets for reliable delivery.
- Senders slow down when retransmitting.

ECN congestion signals:

- Good receivers return Congestion Experienced as ECN Congestion Echo.
- Senders slow down when ECE seen.
- Evil receivers have no reason to set ECE.

Evil is: greed, malice, or software bug.

Good vs. Evil



Evil receivers

- Get better performance than Good receivers
- Reduce performance of competing flows (Victims)

The Challenge

Can we have both:

- The benefits of ECN, and
- Protection against evil receivers?

Can we *detect* evil?

Can we *discourage* evil?

Outline

- ECN-nonce basics
- Header bits: IP and TCP
- Walkthroughs: Good and Evil receivers
- Endpoint memory requirements
- Resynchronization after loss/mark
- Policy: giving the ECN-nonce teeth
- Strange implementation cases
- Further reading

Nonces revoke receiver power

Sender attaches random “nonce” using ECT field

Receivers return the sum (parity) of nonces

Correct nonce sum depends on unmarked packets.

Receiver’s sum only incorrect when:

- ECT cleared by CE.
- Retransmission lacks ECT

Sender can detect an Evil receiver

ECN bits to code-points

bits	RFC 2481	RFC 3168
0 0	not ECN capable	same
0 1	unused	ECT (1)
1 0	ECN-Capable Transport (ECT)	ECT (0)
1 1	Congestion Experienced	same

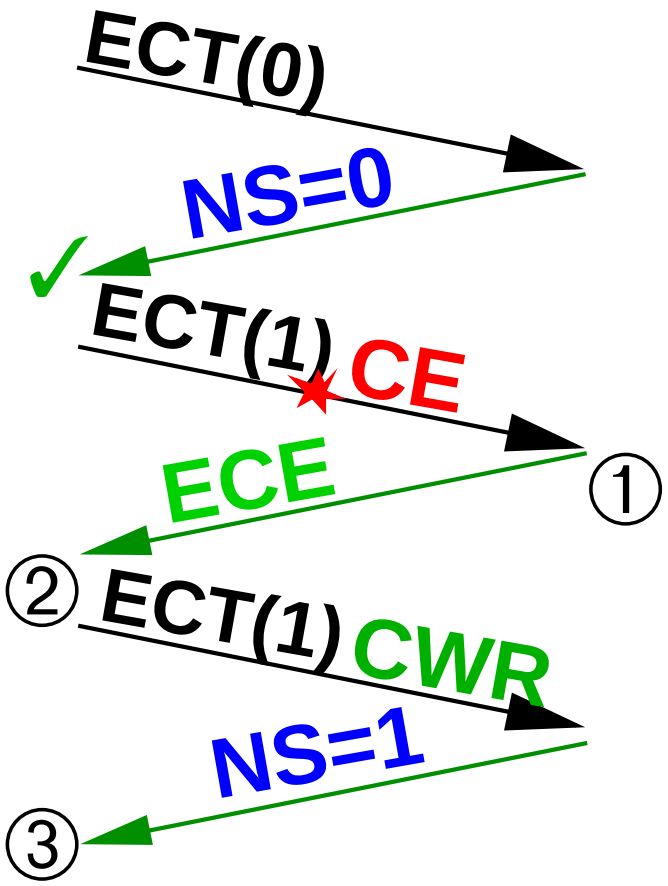
Transition to CE removes original ECT(0) or ECT(1).

Nonce Sum TCP header bit

4 bit header length	reserved (3 bits)	NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN

- Not yet standardized
- Defined whenever ACK is set
- Initial sum is 1.

ECN-Nonces with good receivers

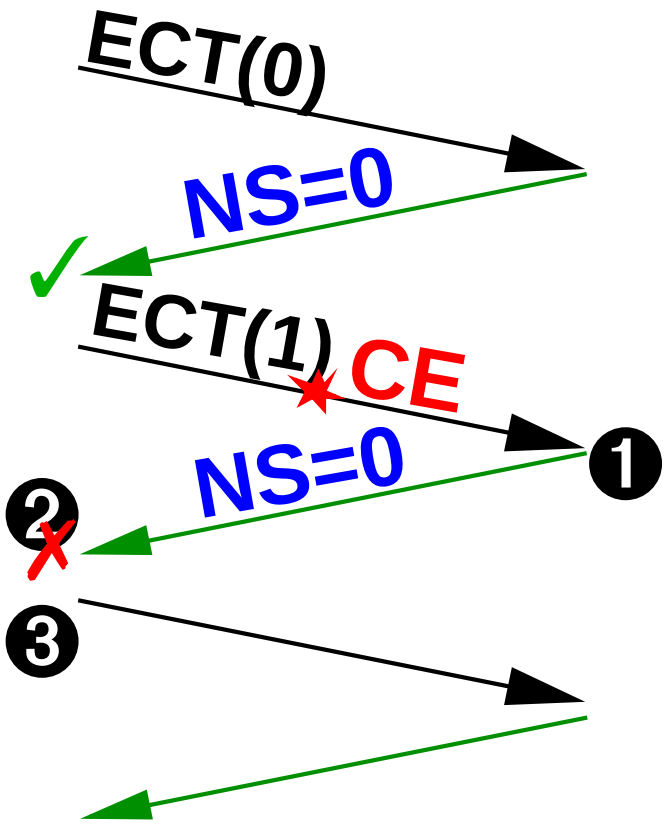


① ECN properly echoed

② Nonce sum (**NS**) ignored

③ Synch. **NS** after **CWR**

ECN-Nonces with an evil reciever



1. CE improperly hidden.
2. Guessed **NS** is wrong.
3. Sender disables ECN.

Memory requirements

Senders store:

- Expected nonce sum for ack of each packet in retransmission buffer.
- The sequence number of the last CWR sent.
- A bit set when the expected nonce sum is wrong. (next slide)

Receivers store:

- Nonce Sum sent in ACKs.
- Nonces of unacknowledged packets.

Resynchronization

After ACK-of-CWR, NS may be “wrong.”

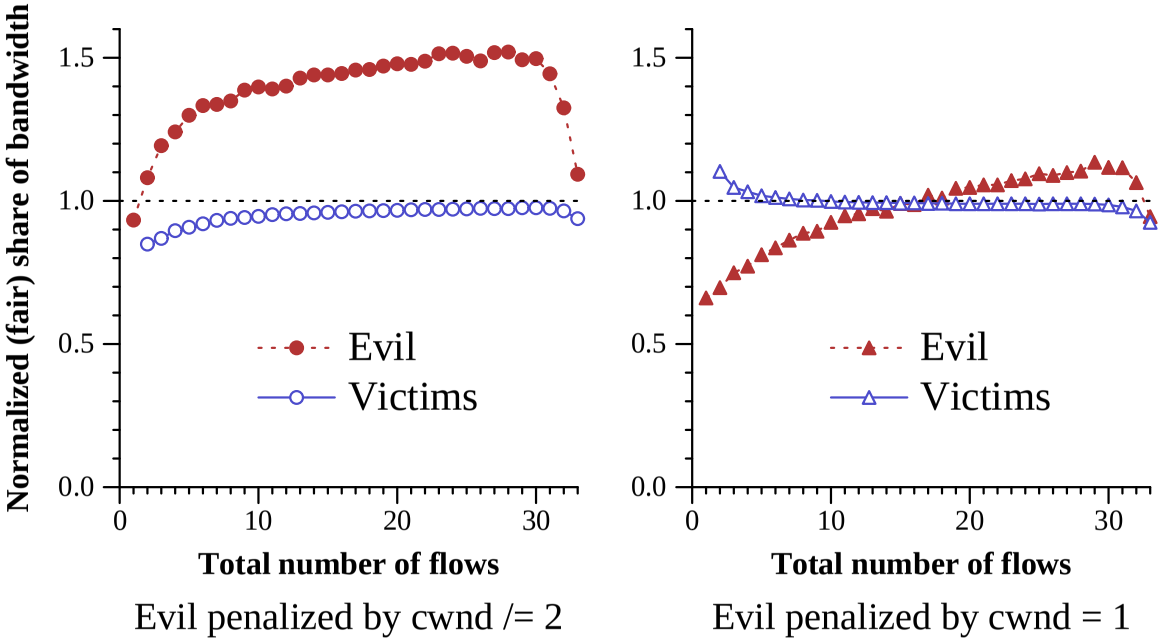
If so, set a flag: NS will continue to be “wrong” until another congestion signal

Keeps protocol complexity at sender.

Making detection sufficient (policy)

- Encourage nonce support
  - Preferred treatment
  - Non-ECN Optimizations
- Discourage misbehavior: On an incorrect nonce:
  - Stop sending ECT
  - Reduce cwnd, ssthresh to 1.
  - Alternatives: RST, limit send window, blacklist

Why cwnd = ssthresh = 1?



Tested long-term behavior: sender continues to set ECT.

cwnd /= 2 left an advantage that cwnd = 1 removed.

Stop sending ECT to be sure

Strange Implementation Cases

Path MTU discovery:

retransmission without a congestion signal.

Fragmentation:

ECN-nonce doesn’t detect faulty reassembly.

Expected Nonce Sum storage:

Send buffer can change size: no static allocation.

Fast paths:

Linux ECN left fast path on ECE or CWR.

ECN-nonce verification is an every-packet thing.

Further Reading

Linux implementation (incomplete)

ICNP 2001 Paper

draft-ietf-tsvwg-tcp-nonce-01

nspring@cs.washington.edu

http://www.cs.washington.edu/homes/nspring/