TCP Extended Data Offset Option

draft-ietf-tcpm-tcp-edo-01

IETF 92 - Dallas

Joe Touch, USC/ISI

Wes Eddy, MTI Systems (presenter)

3/22/2015

1

Overview

- Why do we need EDO?
- Implementation status and experience
  - Getting code and docs
  - Issues with generic receive offload (GRO)
  - Path forward

EDO Motivation

- TCP option space is limited
- EDO extends the space available for options
  - EDO is carried in the current data offset (DO) space
  - EDO indicates a length that overrides DO during header processing

Implementation status

- Two Linux implementations underway
  - USC/ISI student project
    - http://www.isi.edu/touch/tools
    - Tested using an additional JUNK option
  - Pasi's code
    - https://github.com/PasiSa/linux
    - Tested using NOPs to exceed DO limit

ISI implementation

- Status
  - – Linux 3.13 patches available
  - – ISI tech report available
- Results
  - – Currently supports options up to a total of 272B (including a single option up to the max)
  - – NOT compatible with GRO
  - – Perf. of 932 Mbps (GRO off, EDO on)
    (vs. 940 with GRO on and EDO off,
    with no measurable change in CPU load)

GRO issue

- GRO tries to coalesce similar segments
  - With “same options” (that’s incorrect for EDO)
- GRO caused drops and rexmits
  - Drop from 932 Mbps to 208 Mbps
- Experiment to “trick” GRO failed
  - Even when option types change per-segment, GRO caused packet drops
  - It *appears* that GRO may be ignoring unknown options and coalescing even when options aren’t identical, but we did NOT debug or analyze GRO code

Way forward

- A ‘hardened’ implementation will:
  - Overcome header ‘working space’ limits
  - Determine the root cause of the GRO issue
  - GRO should be considered part of the TCP implementation, and needs to either correctly support coalesced EDO segments (e.g., where the EDO option contents match) or not coalesce EDO segments at all (next doc update should mention this impl. Issue)