TCP Option space Extension

draft-ananth-tcpm-tcpointext-00.txt

Anantha Ramaiah
Agenda (Hope)

• Motivation (briefly)
• Summary of known issues with TCP option extension
• Existing proposals and other ideas (briefly)
• What needs to be done moving forward?
Introduction/Motivation

• TCP option space is limited to 40 bytes due to the 4 bit data offset field in the TCP header.
• Several TCP options have been proposed as a means of TCP extension to offer some functionality (TCP-AO, UTO, Multipath TCP options, various experimental options etc.).
• TCP requires that these options need to be “negotiated” during the TCP 3 way handshake.
• During data exchange, there is already a limitation of how much TCP options one can pack in a segment (e.g.; # of SACK blocks)
• Hence, this is not a “solution in search of a problem” anymore. (as it was deemed many years back!)
General issues

• End Host compatibility
  – Needs to be backward compatible and graceful fallback [H1]
  – TCP option negotiation time [H2]

• Middlebox awareness
  – TCP PEP’s (TCP connection termination) [M1]
  – TCP payload scanner/modifiers (Security apps, NAT ALG) [M2]
  – Features like “TCP intercept” [M3]
  – TCP options stripping middleboxes [M4]
  – Middleboxes resegmenting TCP data [M5]
  – Middleboxes dropping packets with new (unknown) TCP options [M6]
  – Maybe other uncommon behaviors/bugs
Existing Proposals (Overview)

- **TCP LO/SLO**
  - Redefines the standard DO field and uses the TCP data area to add extra TCP options.
  - M2 and M5 is an issue. M3 may be an issue.

- **TCP “Extended segments” (DO field overload)**
  - Redefines TCP DO field by using the currently invalid values (i.e., values < 5).
  - Not a clean solution since the TCP option space length would be limited to 5 fixed values.
  - Doesn’t address H2 well. (SYN would get retransmitted etc.,)
  - Exhibits the same issues as TCP LO/SLO as far as the middleboxes goes. The protection against M5 is difficult.
Existing proposals (contd)

• **TCP X2 (Double TCP header size)**
  - Proposes doubling of the TCP header (all fields), so the TCP option space becomes 1020 bytes.
  - Defines a new IP protocol number.
  - Not a long term proposal, since everything has to change. Has the *same* issues in the network (includes middlebox) as any new IP protocol being deployed.

• **TCP LOIC. (Long options with invalidated checksum)**
  - Sends 2 SYN segments (one with deliberate checksum error) and other one containing the LO option. The main aim is to pack all extra TCP options in the “deliberate SYN”.
  - Checksum overload is not always reliable (checksum rollover). Same issues as other proposals.
Additional thoughts

• TCP multiple segments with continuation.
  – Always honor TCP DO existing semantics. Send extra segments (duplicate) to convey extra TCP options.
  – Increases TCP option exchange delay.

• TCP “option cookies”
  – Idea is to compress or encode TCP options in the SYN segment, possibly by having a TCP template or header.

• Reuse/Overload of other TCP fields.
  – Urgent pointer could be used to convey the TCP extended offset. (just like TCP checksum and DO field in the earlier schemes)
Summary

• Good problem to solve, however no solution is perfect.
• Ok, that doesn’t mean one should not move forward, may be it is ok to pick or devise a solution that works well with large set of scenarios.
• We need to do something for TCP options as they are a key element for TCP extensibility.
Next steps

• Please read the draft – provide comments.
  – Is the draft structure ok?
  – Should we just list the motivations alone, proposals in separate draft?
• Any other thoughts?
• Maybe we should add the TCP option space issue to the TCPM charter.

THANK YOU