

Network Working Group
Internet-Draft
Expires: January 2, 2009

W. Eddy
NASA GRC/Verizon FNS
A. Langley
Google Inc
July 1, 2008

Extending the Space Available for TCP Options
draft-eddy-tcp-loo-04

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with [Section 6 of BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/1id-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on January 2, 2009.

Internet-Draft

TCP Long Options

July 2008

Abstract

This document describes a method for increasing the space available for TCP options. Two new TCP options (LO and SLO) are detailed which reduce the limitations imposed by the TCP header's Data Offset field. The LO option provides this extension after connection establishment, and the SLO option aids in transmission of lengthy connection initialization and configuration options.

Table of Contents

1.	Requirements Notation	3
2.	Introduction	4
3.	The Long Options (LO) Option	6
4.	The SYN Long Options (SLO) Option	7
5.	Middlebox Interactions	9
6.	Comparison to Extended Segments	10
7.	Security Considerations	12
8.	IANA Considerations	13
9.	Acknowledgements	14
10.	References	15
10.1.	Normative References	15
10.2.	Informative References	15
Appendix A.	Changes	16
	Authors' Addresses	17
	Intellectual Property and Copyright Statements	18

1. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#) [[RFC2119](#)].

2. Introduction

Every TCP segment's header contains a 4-bit Data Offset (DO) field that implies the length of that segment's TCP header. The DO field has been specified as: "The number of 32-bit words in the TCP Header. This indicates where the data begins. The TCP header (even one including options) is an integral number of 32 bits long" [[RFC0793](#)]. For a TCP implementation, this means that the boundary separating TCP control data and application data is always exactly $DO * 4$ bytes from the beginning of the TCP header.

As a 4-bit unsigned integer, DO's value is bounded between 0 and 15. This allows for a maximum TCP header length of 60 bytes ($15 * 4$ bytes). The required fields in a TCP header occupy a fixed 20 bytes, leaving 40 bytes as the maximum amount of space for use by TCP options.

While 40 bytes is a reasonable amount of space, sufficient for the concurrent use of several presently defined TCP options, there are cases where more space might be useful. For example, the Selective Acknowledgement (SACK) option [[RFC2018](#)] uses a fixed 2 bytes for its kind and length fields, and requires an additional 8 bytes per SACK block. Thus, the maximum number of SACK blocks a TCP acknowledgement may carry is limited to 4 (with 6 bytes left over). Since SACK is commonly used with the Timestamp option [[RFC1323](#)], which uses 10 bytes, this further limits the number of SACK blocks that may be carried to 3. For specific scenarios involving large windows and combinations of data and acknowledgement loss, additional capacity for SACK blocks is known to be useful [[more-sack](#)].

Creation of new TCP options is also hindered by the lack of space left over after currently-used options are accounted for. For long options that must be present at connection-startup time, this is a particular problem, as all negotiable options need to share 40 bytes of space in a SYN segment. One method that has been used to get around this limitation is overloading the Timestamp bytes in the SYN segments [[migrate](#)]. There are other header fields that might be similarly overloaded (e.g. the urgent pointer), but this approach is of obviously limited utility, as it does not address the fundamental limitation imposed by the DO field, and there are a finite number of overloadable header bits.

This document specifies two new TCP options, LO and SLO. The Long Options (LO) option allows two hosts to negotiate for the ability to use TCP headers longer than 60 bytes (and thus options space of greater than 40 bytes) on subsequent segments. This is accomplished by ignoring the DO field's value and adding a 16-bit field at a fixed location in the header's options to replace it. The format and usage

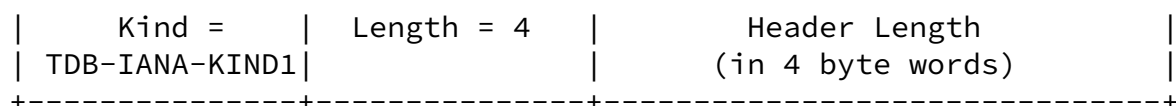
of the LO option is detailed in [Section 3](#).

Attempting to process initial SYN segments with greater than 60 bytes of TCP headers might cause errors if received by hosts that consider anything past the DO-specified boundary to be application data. For backwards compatibility reasons, the maximum length of options on a connection-initiating SYN segment remains 40. The SYN Long Options (SLO) option is used in the case where these 40 bytes are not enough space to carry the desired startup configuration options, and negotiates for later reliable delivery of the left-off options. [Section 4](#) describes the format and usage of the SLO option.

[3.](#) The Long Options (LO) Option

A host might implement some set of TCP options allowing it to predict that greater than 40 bytes of TCP options space may be useful (for example SACK, Timestamps, alternate checksums, etc). In this case, a host MAY implement the LO option. When initiating connections through an active open, hosts implementing the LO option SHOULD place a LO option of the form shown in Figure 1 somewhere in the SYN segment's options. The 16-bit field labelled "Header Length" should be filled in with the same value as the DO field in the required portion of the TCP header, left-padded with zeros.

1																2																3															
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1																
+-----+-----+-----+-----+																																															



TCP Long Options (LO) Option

Figure 1

Receipt of an acknowledgement covering the SYN and also containing an LO option means that future segments MAY include an LO option which expands the length of the TCP header beyond the limit of the DO field. The LO option MUST be the first option and the DO field MUST be set to 6. The value 6 represents the length of the required portions of the TCP header plus the LO option.

An LO option SHOULD NOT be used when not required by the options in a given segment. A host MUST reject any non-SYN segment containing an LO option if the DO field is not equal to 6.

Since a LO option's Header Length field has greater range than the IP header's Total Length field [[RFC0791](#)], this allows TCP options to consume an entire maximum-sized IP datagram's length (minus the IP header and required TCP header fields). No matter what size the options section of a TCP header is, it must still be appended with zero-padding to make the total header a multiple of 32 bits, per [RFC 793](#) [[RFC0793](#)].

Listening hosts that implement the LO option, after reception of a SYN segment with the LO option present, SHOULD reply with a LO option in their SYN-ACK. It can be seen that in both the normal case where one host passively opens and another actively opens, and the more rare case where two hosts simultaneously initiate active opens, the LO option's use can be successfully negotiated.

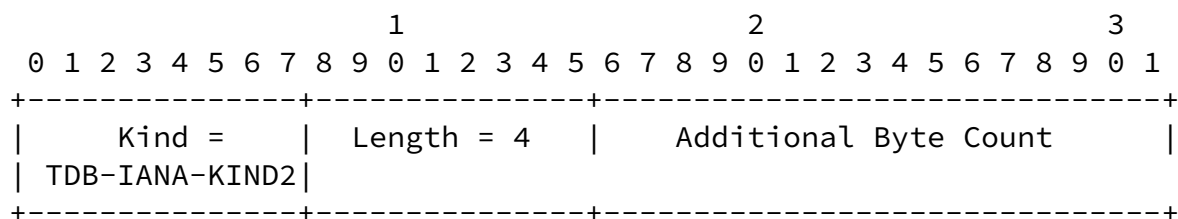
[4.](#) The SYN Long Options (SLO) Option

If the LO option has been successfully negotiated, an active-opening host that has more bytes of initialization options than would fit in the SYN, can use the SYN Long Options (SLO) option. If a host supports the LO option, then it MUST support the SLO option.

Any option bytes transmitted using the SLO option will be treated as

if they were carried on the SYN segment. Since there is no guarantee that the L0 option will be successfully negotiated, the additional 36 bytes left over aside from the 4 byte L0 option on a SYN segment should be filled with the most important remaining options that will fit, as determined by the particular implementation. A host issuing a passive open, MUST NOT use the SLO option, as it can use the L0 option on SYN-ACK segments if it needs to send long initialization options. The SLO option only serves the needs of an active-opening host that, for backwards compatibility reasons, could not send more than 40 bytes of options on the SYN segment.

After successful L0 negotiation, if a host has any options that did not fit on the SYN, then additional data or acknowledgement segments MUST carry a SLO option until the first data byte has been acknowledged. The SLO option's format is shown in figure Figure 2. The trailing 2 bytes hold a 16-bit unsigned count of the additional bytes that would have been in the SYN segment's options, if they had been possible to include. This represents an offset from the end of the SLO option, to the last byte that should be considered a SYN option. The next "Additional Byte Count"-number of bytes trailing the SLO option MUST be the ones that did not fit in the SYN segment. The SLO option should always immediately follow the L0 option, followed by the additional SYN options, and then by normal options, and finally application data.



TCP SYN Long Options (SLO) Option

Figure 2

Since TCP connection establishment is often concluded by a pure acknowledgement (carrying no data), only placing the SLO option and additional SYN options in such a single, unreliable segment would be risky. This is why a host MUST continue transmitting SLO options on

all segments until its first byte of sent data is acknowledged.

Acknowledgement of the first data-byte implicitly covers the SLO and trailing options, as these must have been received end-to-end with the first data byte.

If a host does not send any data bytes, but if by some means (perhaps through the received options) it is possible to derive either an explicit or implicit acknowledgement of even a single option transmitted in a SLO-carrying segment (for example via a Timestamp echo), then a host MAY choose to stop transmitting the SLO data. This special case overrides the previously specified MUST condition.

A host SHOULD NOT continue sending SLO options after it has received acknowledgement of the first data byte, nor should a host process incoming SLO options other than on the first valid segment it receives that carries them.

5. Middlebox Interactions

The large number of middleboxes (firewalls, proxies, protocol scrubbers, etc) currently present in the Internet pose some difficulty for deploying new TCP options. Some firewalls may block segments that carry unknown options. For instance, if the L0 option is not understood by a firewall, incoming SYNs advertising L0 support may be dropped, preventing connection establishment. This is similar to the ECN blackhole problem, where certain faulty hosts and routers throw away packets with ECN bits set [[RFC3168](#)]. Some recent results indicate that for new TCP options, this may not be a significant threat, with only 0.2% of web requests failing when carrying an unknown option [[transport-middlebox](#)].

More problematic, are the implications of TCP connection-splitting middleboxes and protocol scrubbers that do not understand the L0 option. Since such middleboxes may operate on a packet's contents (aggregating application data between multiple segments, rewriting sequence numbers, etc), if the L0 option is not understood, then there may be a mangling of the data passed to the application, as control data could end up inter-mingled with the application data. Such errors could be difficult to detect at the transport layer, and many applications might not perform their own integrity checks. An encouraging fact is that some of these devices reset connection attempts when they see TCP options that they do not understand. Hosts that implement the TCP options described in this document MAY retry connection attempts without L0 options on the SYNs, if their first attempt with L0 options fails.

6. Comparison to Extended Segments

Another proposal that solves the same problem as the LO and SLO options is that of TCP "extended segments" [[ex-segs](#)]. The extended segments technique was proposed following the initial introduction and discussion of the LO and SLO options within the IETF's TCP Maintenance and Minor Extensions working group. The two methods solve the same problem in rather different ways, and have several minor comparative advantages and disadvantages.

The LO and SLO options are designed using the philosophy of using the TCP options space to compensate for insufficiency of the standard header. This is in keeping with the way that several currently-used options work. For example, the Window Scale option deals with the limited space in the advertised receive window field, and the Selective Acknowledgement option solves the lack of information in the cumulative acknowledgement field. Extended segments approach overloads the meaning of the standard Data Offset field, keeping its original meaning for values of 5 and greater, but redefining it for values less than 5. This is seen as acceptable since values less than 5 are currently impossible, illegal, and unusable. Extended segments avoid the need for new options by changing the way that the existing standard header is parsed.

A key advantage of the extended segments approach is that it does not increase the TCP header size, whereas the LO option adds 4 bytes of space to TCP headers. The severity or triviality of this bloat in header overhead depends entirely upon the network properties and application traffic for particular use cases.

It is also not altogether clear that extended segments will always save space in comparison to LO options. The granularity of option lengths that extended segments can support is limited to the number of unusable Data Offset values (5, 0 through 4). Currently, the extended segments proposal defines 4 fixed lengths, and one "infinite" length that means the entire segment is options, with no application data. The fixed option lengths are 48, 64, 128, and 256 bytes. If the required per-data-segment options space for some extension or combination of extensions does not map to exactly these

values, then padding bytes are required. If 129 bytes of options are required on a data segment, then a length of 256 must be used, and 127 bytes of useless padding are added. The L0 option has a single-byte granularity and avoids the need for all wasteful padding, aside from that mandated to make the header a perfect multiple of 4-bytes. It is possible that the overhead on a single extended segment could be more than that of several segments using the L0 option.

Some networkers have found the SLO mechanism that is required for

processing of long initialization options to be somewhat "ugly". Extended segments avoid this by sending long initialization options on the initial SYN and SYN-ACK segments. If the other side does not support extended segments, this adds needless confusion and delay in connection setup. The protocol dance to negotiate use of extended segments is arguably much worse than using SLO. If an extended SYN is not understood, a non-reliably transmitted RST segment signals the initiating host to retry without extended segments. Such a retry mechanism is not commonly found in existing TCP implementations. If the L0 option is not understood, a SYN-ACK is still immediately generated and the connection goes on uninterrupted, without any additional retry mechanisms. Furthermore, extended SYN-ACKs may be sent in response to non-extended SYNs. This complicates the recovery procedure even more, if not understood, and goes against the way that all current negotiable TCP extensions operate (only used on SYN-ACK if advertised on SYN).

Over-zealous middleboxes are immensely troublesome for the deployment of most transport layer extensions. It is unclear whether L0 and extended segments have any real difference in robustness in the presence of different types of middleboxes. Both types of segments may appear as invalid to some middleboxes, and both may be mangled if rewritten by a middlebox.

[7.](#) Security Considerations

The TCP options presented in this document open no additional vulnerabilities that we are aware of.

[8.](#) IANA Considerations

This document does not create any new registries or modify the rules for any existing registries managed by IANA.

This document requires IANA to update values in its registry of TCP options numbers to assign two new entries, referred herein as "TBD-IANA-KIND1" and "TBD-IANA-KIND2".

[9.](#) Acknowledgements

This document benefitted specifically from discussions with Josh Blanton and Shawn Ostermann. Some comments from Eddie Kohler motivated the discussion of middlebox interactions. Valuable feedback was obtained from Mark Allman and other participants in the TCP Maintenance and Minor Extensions (TCPM) Working Group.

[10.](#) References

[10.1.](#) Normative References

- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, [RFC 793](#), September 1981.
- [RFC0791] Postel, J., "Internet Protocol", STD 5, [RFC 791](#), September 1981.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.

[10.2](#). Informative References

- [RFC1323] Jacobson, V., Braden, B., and D. Borman, "TCP Extensions for High Performance", [RFC 1323](#), May 1992.
- [RFC2018] Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, "TCP Selective Acknowledgment Options", [RFC 2018](#), October 1996.
- [more-sack] Srijith, K., Jacob, L., and A. Ananda, "Worst-case Performance Limitation of TCP SACK and a Feasible Solution", Proceedings of 8th IEEE International Conference on Communications Systems (ICCS), November 2002.
- [migrate] Snoeren, A. and H. Balakrishnan, "An End-to-End Approach to Host Mobility", Proc. of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking, August 2000.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", [RFC 3168](#), September 2001.
- [transport-middlebox] Medina, A., Allman, M., and S. Floyd, "Measuring Interactions Between Transport Protocols and Middleboxes", ACM SIGCOMM/USENIX Internet Measurement Conference, October 2004.
- [ex-segs] Kohler, E., "Extended Option Space for TCP", Internet Draft (work in progress), September 2004.

[Appendix A](#). Changes

To be removed by RFC Editor before publication

Changes since 03

1. Change the option numbers specified to placeholders:
"TBD-IANA-KIND1" and "TBD-IANA-KIND2".
2. Change the requirement that all segments include the L0 option, if negotiated, to a SHOULD NOT unless the options require it. The reasoning behind the initial requirement was for implementation ease but, having implemented it myself, the ability to use the fast path processing for L0 connections outweighs that.
3. Change the units of the L0 option from bytes to words. This was ambiguous in the 03 draft and, since padding to four bytes was required anyway, it seemed best to remove one extra way that the option could be invalid.

Internet-Draft

TCP Long Options

July 2008

Authors' Addresses

Wesley M. Eddy
NASA GRC/Verizon FNS
21000 Brookpark Rd, MS 54-5
Cleveland, OH 44135

Phone: 216-433-6682
Email: weddy@grc.nasa.gov

Adam Langley
Google Inc

Email: agl@imperialviolet.org

Full Copyright Statement

Copyright (C) The IETF Trust (2008).

This document is subject to the rights, licenses and restrictions contained in [BCP 78](#), and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in [BCP 78](#) and [BCP 79](#).

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at

<http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.