TCP Maintenance and Minor Extensions (tcpm)                   B. Briscoe
Internet-Draft                                                    BT
Intended status: Experimental                                   July 22, 2014
Expires: January 23, 2015

Abstract

This document describes an experimental method to extend the option
space for connection parameters within the initial TCP SYN segment at
the start of a TCP connection. In this method the TCP client sends
two alternative SYN:s: one intended for legacy servers and one
intended for upgraded servers. Once it establishes which type of
server has responded, it continues the connection appropriate to that
server type and aborts the other. The SYN intended for upgraded
servers includes additional options at the end of the payload.

Status of This Memo

This Internet-Draft is submitted in full conformance with the
provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering
Task Force (IETF). Note that other groups may also distribute
working documents as Internet-Drafts. The list of current Internet-
Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on January 23, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the
document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal
Provisions Relating to IETF Documents
(http://trustee.ietf.org/license-info) in effect on the date of
publication of this document. Please review these documents
carefully, as they describe your rights and restrictions with respect
to this document. Code Components extracted from this document must
include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.  Introduction .................................................. 2
   1.1.  Scope ..................................................... 3
   1.2.  Experiment Goals ......................................... 3
   1.3.  Terminology ............................................... 3
2.  Protocol Specification ......................................... 3
3.  Interaction with TCP ........................................... 7
4.  IANA Considerations ........................................... 7
5.  Security Considerations ....................................... 7
6.  Acknowledgements .............................................. 7
7.  References .................................................... 8
   7.1.  Normative References ...................................... 8
   7.2.  Informative References .................................... 8
Appendix A.  Alternative Protocol Specifications ..................... 8
   A.1.  Deterministic Protocol Specification ....................... 8
   A.4.  Comparison of Alternatives ................................ 12
Appendix B.  Protocol Design Issues (to be Deleted before Publication) .............................................. 13
Appendix C.  Change Log (to be Deleted before Publication) ........ 14
Author’s Address .................................................. 15

1.  Introduction

This document describes an experimental method to extend the option space available in the initial SYN segment of a TCP connection (i.e. SYN set and ACK not set) [RFC0793]. This extension is required to support some combinations of TCP options, notably large ones such as TCP AO [RFC5925] (16B), Multipath TCP [RFC6824] (12B), and TCP Fast Open [I-D.ietf-tcpm-fastopen] (6-18B) as well as other options already typically used in TCP connections, such as SACK-ok (2B), Timestamp (10B), Window Scale (3B), MSS (4B).

In this method the TCP client sends two alternative SYNs: one intended for legacy servers and one intended for upgraded servers. Once it establishes which type of server has responded, it continues the connection appropriate to that server type and aborts the other. The SYN intended for upgraded servers includes additional options at the end of the payload.
1.1. Scope

This experimental specification extends the TCP wire protocol. It is independent of the dynamic behaviour of TCP and it is independent of (and thus compatible with) any protocol that encapsulates TCP, including IPv4 and IPv6.

1.2. Experiment Goals

TCP is critical to the robust functioning of the Internet, therefore any proposed modifications to TCP need to be thoroughly tested. The present specification describes an experimental protocol that provides extra option space on the initial TCP SYN segment. The intention is to specify the protocol sufficiently so that more than one implementation can be built in order to test its function, robustness and interoperability (with itself, with previous version of TCP, and with various commonly deployed middleboxes).

Success criteria: The experimental protocol will be considered successful if it satisfies the following requirements in the consensus opinion of the IETF tcpm working group. (ToDo: describe success criteria)

Duration: To be credible, the experiment will need to last at least 12 months from publication of the present specification. If successful, a report on the experiment will be written up. It would then be appropriate to work on a standards track specification, in which the experiment report may be included.

1.3. Terminology

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 [RFC2119]. In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

2. Protocol Specification

This method is termed the non-deterministic method to distinguish it from similar alternative methods specified in Appendix A. In this method the TCP client sends two alternative SYNs: a regular SYN intended for legacy servers and a SYN-UN intended for upgraded servers. The two SYNs will have the same network addresses and the same destination port, but different source ports. Once it establishes which type of server has responded, it continues the connection appropriate to that server type and aborts the other.
SYN intended for upgraded servers (SYN-UN) includes additional options at the end of the payload. The options are placed at the end of the payload to ensure that the SYN-UN is more likely to traverse middleboxes that inspect application-layer headers, which they expect to be at the start of the payload.

Table 1 summarises the TCP 3-way handshake exchange for each of the two SYNs between an upgraded TCP (active opening) client and either i) a legacy server or ii) an upgraded server.

<table>
<thead>
<tr>
<th></th>
<th>Legacy Server</th>
<th>Legacy Server</th>
<th>Upgraded Server</th>
<th>Upgraded Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SYN</td>
<td>SYN-UN</td>
<td>SYN</td>
<td>SYN-UN</td>
</tr>
<tr>
<td>2</td>
<td>SYN/ACK</td>
<td>SYN/ACK</td>
<td>SYN/ACK</td>
<td>SYN/ACK-U</td>
</tr>
<tr>
<td>3</td>
<td>ACK</td>
<td>RST</td>
<td>Wait for ACK</td>
<td>ACK</td>
</tr>
<tr>
<td>4</td>
<td>Cont...</td>
<td>SYN/ACK-U</td>
<td>RST</td>
<td>Cont...</td>
</tr>
</tbody>
</table>

Table 1: Overview of 3-Way Handshakes for the Two Alternative SYNs in Two Server Scenarios: Non-Deterministic Alternative

(ToDo: explain the table long-hand.)

If the client receives a response to the SYN, but not the SYN-UN, it SHOULD retransmit the SYN-UN. In parallel to any retransmission, or instead of any retransmission, the client MAY give up on the SYN-UN connection and complete the 3-way handshake of the other regular connection. If the client receives no response at all, it SHOULD solely retransmit the SYN. It MUST NOT retransmit both segments, because the lack of response could be due to severe congestion.

The SYN-UN is structured as shown in Figure 1. It can be seen that TCP options are placed at the end of the payload at an offset from the start of the payload defined using the Extra Options Offset (EOO) field.

The EOO field is read from a new ‘SYN-OP-SYS’ TCP option defined in this specification. The SynOpSis TCP options MUST be the final TCP option right-aligned at the end of the payload (preceded by padding options if necessary), so that the server can find it (using the length of the whole packet found in the network layer header, e.g. IPv4 or IPv6).
The SynOpSis TCP option has Kind SynOpSis, with a value (TBA) (See Section 4). In general, the SynOpSis TCP option can have different lengths for different purposes. However, in a SYN-UN, the SynOpSis TCP option MUST have Length = 8, so that the server can find where it starts (8 octets before the end of the segment).

The internal structure of the SynOpSis TCP option for a SYN-UN is defined in Figure 2. The SynOpSis TCP option for a SYN-UN MUST have Length = 8. The first 4 octets of the option contain a magic number (TBA) to reduce the chance that arbitrary data within the payload will be mistaken for a SynOpSis TCP option.

Two single octet offset fields are placed at the end of the SynOpSis TCP option for a SYN-UN:

- **The Extra Options Offset (EOO)**: The EOO field defines the offset in 4-octet words from the start of the payload to the start of the first extra TCP option at the end of the payload. If a payload is not required, EOO will be zero.

- **The Extra Prefix Options Offset**: The EPOO field defined an additional offset from the start of the extra TCP options in order to identify any extra TCP options that need to be processed before any regular TCP options in the SYN-U. The EPOO field defines this offset in 4-octet words.

An upgraded server determines whether there is a SynOpSis TCP option at the end of the packet by checking all the following conditions:

- The Kind value is the SynOpSis Kind value;
The length is 8;

The next 4 octets match the magic number;

The sum of the value of the EOO field, and all the length fields found by walking along the TCP options at the end of the payload exactly reaches the end of the packet.

If any of these conditions fails, the server MUST treat the whole payload as user-data.

With this non-deterministic approach, there will be a very small probability (roughly $2^{-48-L}$) that payload data on a regular (non-SynOpSis) SYN will happen to contain a pattern in exactly the right place that matches the magic number, and that the payload data also happens to contain a valid sequence of numbers in exactly the right places to look like a valid string of TCP options (where $L$ is the sum of all the bits in all the TCP option length fields that seem to be in the payload). For instance, if it appears that there are 2 TCP options before the SynOpSis option at the end of the payload, then $L=2*8=16$, and the probability of incorrectly using user-data as TCP options will then be roughly $2^{-64} = 1$ in 18 billion billion (18x10^18).

It is recognised that it is potentially dangerous to use probability to determine whether TCP options are hidden at the end of the payload. This ‘stealth’ approach has been taken in order to maximise the chances of traversing middleboxes; by ensuring that all the TCP headers and options of a SYN-UN are completely indistinguishable from a regular SYN. If this non-deterministic approach is not preferred, an alternative more conventional deterministic protocol designs is provided in Appendix A.

An upgraded server processes the TCP options in a SYN-UN in the following order:

1. The Prefix TCP options (TCP-Opt#1 in Figure 1)

2. The regular TCP options following the main header but before the payload (TCP-Opt#2 in Figure 1);

3. The Suffix TCP options (TCP-Opt#3 in Figure 1)

SYN/ACK-U carries a simple SynOpSis flag TCP option as defined in Figure 3. It solely identifies that the SYN/ACK is from a server that supports SynOpSis TCP options.
3. Interaction with TCP

(ToDo: TCP User Interface, TCP States and Transitions, TCP Segment Processing, Processing and Segment Size Overhead, Connectionless Resets, ICMP Handling. Interaction with EDO, Interactions with Other TCP Variants, Forward-Compatibility, Interaction with TCP assumptions of Middleboxes.)

4. IANA Considerations

This memo will include a request to IANA for a new TCP option kind SynOpSis.

This specification requires IANA to allocate one value from the TCP option Kind name-space, against the name "Sister SYN Options (SynOpSis)"

Early implementation before the IANA allocation MUST follow [RFC6994] and use experimental option 254 and magic number 0xHHHH (16 bits) (ToDo TBA and register this with IANA), then migrate to the new option after the allocation.

5. Security Considerations

Certain cryptographic functions have different coverage rules for the TCP payload and TCP options. Placing some TCP options in the payload could mean that they are treated differently from regular TCP options. This is a deliberate feature of the protocol, but application developers will need to be aware that this is the case.

(ToDo: More)

6. Acknowledgements

The idea of this approach grew out of discussions with Joe Touch while developing draft-touch-tcpmsyn-ext-opt, and with Jana Iyengar and Olivier Bonaventure. The following people provided useful review comments: Joe Touch, Yuchung Cheng.
Bob Briscoe was part-funded by the European Community under its Seventh Framework Programme through the Trilogy 2 project (ICT-317756). The views expressed here are solely those of the authors.

7. References

7.1. Normative References


7.2. Informative References


Appendix A. Alternative Protocol Specifications

This appendix is informative and will be deleted before publication. It documents alternative protocol arrangements that may be considered instead of the non-deterministic protocol in Section 2.

A.1. Deterministic Protocol Specification

This alternative protocol arrangement is termed the Deterministic SynOpSis protocol. It is termed 'deterministic' because it uses a more conventional approach for placement of the SynOpSis TCP option instead of the non-deterministic approach in Section 2. However, it is likely to be less practical, given it uses TCP options in the clear, hoping that they will traverse middleboxes, which will not always be successful.
This method is similar in structure to the non-deterministic method in Section 2. The TCP client still sends two alternative SYNs: SYN-L intended for legacy servers and SYN-UD intended for upgraded servers. The two SYNs will have the same network addresses and the same destination port, but different source ports. Once the client establishes which type of server has responded, it continues the connection appropriate to that server type and aborts the other. The SYN intended for upgraded servers (SYN-UD) includes additional options at the end of the payload. The options are placed at the end of the payload to ensure that the SYN-UD is more likely to traverse middleboxes that inspect application-layer headers, which they expect to be at the start of the payload.

Table 2 summarises the TCP 3-way handshake exchange for each of the two SYNs between an upgraded TCP (active opening) client and either i) a legacy server or ii) an upgraded server.

<table>
<thead>
<tr>
<th></th>
<th>Legacy Server</th>
<th>Legacy Server</th>
<th>Upgraded Server</th>
<th>Upgraded Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SYN-L</td>
<td>SYN-UD</td>
<td>SYN-L</td>
<td>SYN-UD</td>
</tr>
<tr>
<td>2</td>
<td>SYN/ACK</td>
<td>SYN/ACK</td>
<td>RST</td>
<td>SYN/ACK-U</td>
</tr>
<tr>
<td>3</td>
<td>ACK</td>
<td>RST</td>
<td>ACK</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Cont...</td>
<td></td>
<td>Cont...</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Overview of 3-Way Handshakes for the Two Alternative SYNs in Two Server Scenarios: Deterministic Alternative

(ToDo: Explain the table long-hand.)

If the client receives a RST on one connection and no response on the other, it SHOULD retransmit the SYN that got no response. In parallel to any retransmission, or instead of any retransmission, the client MAY send a SYN without any SynOpSis option, in case this is the cause of the black-hole. However, the presence of the RST implies that one of the SYNs with a SynOpSis TCP option probably reached the server, therefore it is more likely (but not certain) that the lack of response on the other connection is due to transmission loss or congestion loss. If the client receives no response at all, it SHOULD fall-back to sending a regular SYN with no SynOpSis option. It MUST NOT retransmit both segments, because the lack of response could be due to severe congestion.

In contrast to the more robust method, the SYN intended for a legacy server is different from a regular SYN, hence it is called a SYN-L. The SYN-L is merely a SYN with with an extra SynOpSis flag option as
shown in Figure 3. It solely identifies that the SYN is from a client that supports SynOpSis TCP options.

The placement of the SynOpSis TCP option in a deterministic SYN-UD segment is more conventional than in the SYN-U of Section 2, as shown in Figure 4. Nonetheless, it can be seen that extra TCP options are still placed at the end of the payload at an offset from the start of the payload defined using the Extra Options Offset (EOO) field.

The EOO field is read from a new ‘SYN-OP-SYS’ TCP option defined in this specification. The SynOpSis TCP options is placed in the regular TCP option space of the SYN-UD.

```
| EOO |
+-----+-------->

+---------+-----------+----------+-----------+---------+-----------+
| TCP hdr | TCP-Opt#1 | SynOpSis | TCP-Opt#3 | Payload | TCP-Opt#2 |
+---------+-----------+----------+-----------+---------+-----------+
```

Figure 4: The Structure of an alternative (deterministic) SYN-UD segment (not to scale)

The SYN-OP-SYS TCP option for a SYN-UD segment MUST have Kind SynOpSis, with a value (TBA) (See Section 4) and Length = 3. In general, the SynOpSis TCP option can have different lengths for different purposes. However, in a SYN-UD, the SynOpSis TCP option has Length = 3, so that it can carry the 1-octet EOO field, which MUST be present in a SYN-UD. The internal structure of the SynOpSis TCP option for a SYN-UD segment is defined in Figure 5.

```
| Kind=SynOpSis | Length=3 | EOO |
+---------------+----------+-----+
```

Figure 5: SynOpSis TCP Option for a deterministic SYN-UD

The Extra Options Offset (EOO) field defines the offset in 4-octet words from the start of the payload to the start of the first extra TCP option at the end of the payload. If space for payload data is not required, EOO will be zero, but it MUST still be present.

An upgraded server processes the TCP options in a SYN-UD in the following order:

1. The regular TCP options following the main header but before the SynOpSis TCP option (TCP-Opt#1 in Figure 4)
2. The TCP options at the end of the payload (TCP-Opt#2 in Figure 4)

3. The regular TCP options following the main header but after the SynOpSis TCP option (TCP-Opt#3 in Figure 4);


The protocol outlined in Table 3 is a hybrid of the two approaches in Section 2 and Appendix A.1.

<table>
<thead>
<tr>
<th></th>
<th>Legacy Server</th>
<th>Legacy Server</th>
<th>Upgraded Server</th>
<th>Upgraded Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SYN-L</td>
<td>SYN-UN</td>
<td>SYN-L</td>
<td>SYN-UN</td>
</tr>
<tr>
<td>2</td>
<td>SYN/ACK</td>
<td>SYN/ACK</td>
<td>RST</td>
<td>SYN/ACK-U</td>
</tr>
<tr>
<td>3</td>
<td>ACK</td>
<td>RST</td>
<td></td>
<td>ACK</td>
</tr>
<tr>
<td>4</td>
<td>Cont...</td>
<td></td>
<td></td>
<td>Cont...</td>
</tr>
</tbody>
</table>

Table 3: Overview of 3-Way Handshakes for the Two Alternative SYNs in Two Server Scenarios: Non-Deterministic Explicit Alternative

The SYN-L might not traverse middleboxes, therefore connections with legacy servers could suffer from the added latency of a retransmission. Nonetheless, if it does reach an upgraded server, the server knows explicitly that the client supports SynOpSis TCP options and can abort the connection without having had to hold any state.

The SYN-UN has the advantage that it is all-but indistinguishable from a regular SYN, therefore it is likely to traverse most middleboxes. However, there is a tiny chance that an upgraded server might mistake some arbitrary payload for a SYN-U, if data happens to match the magic number.

### A.3. Deterministic Implicit Protocol Specification

The protocol outlined in Table 4 is a hybrid of the two approaches in Section 2 and Appendix A.1.
Table 4: Overview of 3-Way Handshakes for the Two Alternative SYNs in Two Server Scenarios: Deterministic Implicit Alternative

The regular SYN will traverse middleboxes so there will be no latency problems reaching legacy servers. However an upgraded server cannot know that it comes from a client that supports SynOpSis TCP options, therefore it has to open connection state for both connections until the client explicitly aborts the one intended for a legacy server.

The SYN-UD has the advantage that an upgraded server cannot occasionally mistake payload data for TCP options. However, this approach could introduce latency reaching an upgraded server, if the SYN-UD does not traverse a middlebox.

A.4. Comparison of Alternatives

The four solutions in sections 2, A.1, A.2 & A.3, each suffer from two of the following four failings:

Risk Delay Accessing Legacy Server: due to the SYN intended for a legacy server being non-normal and therefore at risk of drop by a middlebox;

Risk Delay Accessing Upgraded Server: due to the SYN intended for an upgraded server being non-normal and therefore at risk of drop by a middlebox;

Doubles Upgraded Server State within RTT: because the server has to hold the second connection’s state until the client resets it;

Risk of Upgraded Server Mistaking User-Data for TCP Options: due to use of the probabilistic magic number technique.

All four schemes double up connection state on the legacy server for a round trip.
Table 5 places a tick ('/') against the combinations of SYNs that are better in each respect.

<table>
<thead>
<tr>
<th></th>
<th>Non-Deterministic</th>
<th>Deterministic</th>
<th>Non-Deterministic</th>
<th>Deterministic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SYN + SYN-UN</td>
<td>SYN-L + SYN-UD</td>
<td>SYN-L + SYN-UN</td>
<td>SYN + SYN-UD</td>
</tr>
<tr>
<td>Min Delay to Legacy Svr</td>
<td>/</td>
<td>X</td>
<td>X</td>
<td>/</td>
</tr>
<tr>
<td>Min Delay to Upgraded Svr</td>
<td>/</td>
<td>X</td>
<td>/</td>
<td>X</td>
</tr>
<tr>
<td>Min State on Upgraded Svr</td>
<td>X</td>
<td>/</td>
<td>/</td>
<td>X</td>
</tr>
<tr>
<td>User-Data Unstable</td>
<td>X</td>
<td>/</td>
<td>X</td>
<td>/</td>
</tr>
</tbody>
</table>

Table 5: Comparison of the Four Alternative Solutions

It should be noted that there is no need for the IETF to choose a single pair of SYNs. Once the SynOpSis TCP options are defined, clients can use their own choices of pairs in different circumstances. Initially clients might choose the Non-Deterministic scheme to minimise delays due to middlebox interference. But later, perhaps once more middleboxes support the scheme, clients might choose the Non-Deterministic Explicit scheme, to minimise both state and delay with the upgraded server.

Appendix B. Protocol Design Issues (to be Deleted before Publication)

This appendix is informative, not normative. It records outstanding issues with the protocol design that will need to be resolved before publication.

Reliance on segmentation boundary: The definition of the position of the SynOpSis TCP options depends on where the sender decided to
place a segment boundary. In general, a sender cannot rely on segment boundaries being preserved, e.g. by segmentation offloading hardware. In the case of a SYN, no more payload data is sent in the first round trip, therefore using this segment boundary might be safe. However, it may constrain future attempts to send additional data in the first round.

Appendix C. Change Log (to be Deleted before Publication)

A detailed version history can be accessed at
<http://datatracker.ietf.org/doc/draft-briscoe-tcpm-syn-op-sis/history/>

From briscoe...-00 to briscoe...-01:

Technical changes:

* Added the definition of a SYN/ACK-U

* Deterministic Protocol Spec: Replaced SYN/ACK-L with RST (Joe Touch)

* Added Non-Deterministic Explicit and Deterministic Implicit Protocol Specs in Appendices

* Added Comparison of Alternatives as an Appendix

* Security Considerations: Added note about crypto coverage of TCP options in the payload being different from that of other TCP options.

* Added an appendix to record outstanding Protocol Design Issues, and included segmentation boundary issue (Yuchung Cheng).

Editorial changes:

* Changed TCP option Kind from SYN-OP-SIS to SynOpSis

* Protocol Spec: Explained why the extra TCP options are placed at the end of the payload

* Throughout: avoided the ambiguity in the word payload, now that there are TCP options at the end of the payload. Some might consider these to be within the payload, while others might consider them to be placed beyond the payload.

* Segment structure figures: Clarified that they are not to scale.
* Added placeholder section "Interaction with TCP"

* Acknowledged reviewers

Author’s Address

Bob Briscoe  
BT  
B54/77, Adastral Park  
Martlesham Heath  
Ipswich IP5 3RE  
UK  
Phone: +44 1473 645196  
Email: bob.briscoe@bt.com  
URI:  http://bobbriscoe.net/