

TCPM
Internet-Draft
Intended Status: Experimental
File: draft-borman-tcpm-tcp4way-01.txt
Expires: September 9, 2015

D. Borman
Quantum Corporation
March 9, 2015

TCP Four-Way Handshake

Abstract

One of the limitations of TCP is that it has limited space for TCP options, only 40 bytes. Many mechanisms have been proposed for extending the TCP option space, but the biggest challenge has been to get additional option space in the initial SYN packet.

This memo presents a optional four-way TCP handshake to allow extended option space to be used in SYN packets in both directions.

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 September 9, 2015.

Copyright

Copyright (c) 2015 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	3
2. Terminology	3
3. Motivation For this Approach	3
4. TCP Four-Way Handshake	4
4.1. Overview	5
4.2. Changes to the TCP state diagram	6
4.3. Three-Way or Four-Way Handshake?	7
4.3.1. Using the 4WAY Bit	7
4.3.2. Non Four-Way Client Sets 4WAY Bit	7
4.3.3. Non Four-Way Server Sets 4WAY Bit	8
4.4. PRE-ESTABLISHED State	8
5. Negotiating Non-directional vs. Directional TCP Options	8
5.1. Caching EDO support	9
6. TCP Connection State Diagram	9
7. Using TCP Options in the Four-Way Handshake	12
7.1. New TCP Options in the SYN/ACK	12
7.2. Handling Unknown TCP Options	13
7.2.1. Unknown Option Option	13
7.3. TCP options	14
7.3.1. RFC defined options	14
7.3.1.1. End of Option List	14
7.3.1.2. No-Operation	14
7.3.1.3. Maximum Segment Size	14
7.3.1.4. Window Scale	14
7.3.1.5. SACK Permitted	14
7.3.1.6. SACK	14
7.3.1.7. Timestamps	15
7.3.1.8. MD5 Signature Option	15
7.3.1.9. Quick-Start Response	15
7.3.1.10. User Timeout Option	15
7.3.1.11. TCP Authentication Options (TCP-AO)	15
7.3.1.12. Multipath TCP (MPTCP)	15
7.3.1.13. RFC3692-style Experiment	15
7.3.1.14. TCP Fast Open Option	15
7.3.1.15. T/TCP Options	16
8. IANA Considerations	16
9. Security Considerations	16
10. References	16
10.1. Normative References	16
10.2. Informative References	17
Appendix A. First Response of the Four-Way Handshake	19
Appendix B. Communicating Four-Way Handshake Support	20
Acknowledgments	20
Contributors	20
Author's Address	21

1. Introduction

The TCP packet format has 40 bytes for adding TCP options. The most common method to extend TCP is to define new options, but the limited TCP option space can make that difficult as the number of potential options grow. Support for various TCP options is typically negotiated during the three-way handshake, in the packets that contain the SYN. If both sides send and receive a given option in a packet with the SYN bit set, then both sides know that the option is supported. Examples of this are the Window Scale and Timestamps options [RFC7323]. Note that RFC 7323 is not clear on this point, its description is Three-Way handshake centric, stating that the Timestamps option is enabled by its presence in the <SYN> and <SYN,ACK> packets, rather than the more general definition that the option is enabled by both sending and receiving the option in a packet that contains the SYN bit. It is a subtle difference that doesn't matter for the Three-Way handshake, but is important for a simultaneous open and the Four-Way handshake.

The majority of TCP sessions begin with three-way handshake, the exception to that is a simultaneous open.

The ideas presented in this memo were first hinted at in a message to the TCPM mailing list [Borman14].

2. 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 RFC 2119. [RFC2119]

3. Motivation For this Approach

The problem of expanding the TCP option space in the initial SYN packet has vexed designers for years. The main issue is maintaining compatibility with legacy TCP implementations, which don't understand the expanded TCP option space. When the initial SYN is sent, there is no knowledge as to whether or not the remote side can understand the extended option space. Various approaches have been considered:

- 1) Send dual SYNs, with and without the extended options, and arrange that the extended SYN will be considered invalid and dropped by legacy implementations. [Yourtchenko11] [Briscoe14]
- 2) Send an additional out of band packet along with the SYN to contain additional options. [Touch14]
- 3) Send additional options that didn't fit into the SYN in additional packets using a new TCP option. [Eddy08]
- 4) Send an initial SYN with extended options that a legacy server will fail, and then fall back to a new SYN without extended

options. [Kohler04]

[Ramaiah12] contains additional analysis of proposed ways to expand the TCP option space.

Expanding the TCP option space in the initial SYN is a case of the more general issue: How can you change the fixed TCP header in the initial SYN packet and still maintain compatibility with legacy implementations? The TCP Window Scale option [RFC7323] redefined the Window field, but only in non-SYN packets. In the case of expanding the TCP option space, it involves redefining or overriding the Data Offset (DO) field.

The most straight forward method for dealing with modifying the initial SYN packet is to add an initial packet exchange so that the client can find out what the server supports, and then it knows, for example, if the server supports extended TCP option space. The problem with this approach is that it adds an additional RTT to connection startup, and most people are looking for ways to shorten, not lengthen, the initial connection setup, for example "TCP Fast Open" [TFO]. Though to be clear, the extra RTT is really not a concern about connection setup, but about when data can be first delivered to the application.

An alternative is to send the additional data in the initial SYN such that a legacy TCP will ignore it. This is most commonly done by sending the information in a TCP option, which legacy TCP would ignore. But the TCP option space is only 40 bytes, and by definition an expanded TCP option space won't fit in the legacy TCP option space. So, the additional data needs to be sent by some other mechanism, e.g. in a second SYN or in an additional non-SYN packet. Challenges with this approach include the SYNs being routed to different destination machines, the order of the packets being reversed, as well as a server needing to wait some amount of time to decide whether or not the additional packet will be arriving.

The goal of this proposal is to integrate discovery of server capabilities into the connection setup, while still allowing for data to be delivered in a timely manner.

4. TCP Four-Way Handshake

The TCP Four-Way handshake extends the traditional Three-Way handshake by changing the clients final ACK to a SYN/ACK, and adding a final ACK from the Server. This gives the client a second packet with the SYN bit set in which it can make use of the EDO option to send TCP options that didn't fit into the original SYN, with some limitations.

4.1. Overview

For a connection with ISS (Initial Send Sequence) values of ISSA from the client and ISSB from the server, the normal three-way TCP handshake is:

```

Enter SYN-SENT
SYN(seq=ISSA) ->

Enter SYN-RECEIVED
<- SYN(seq=ISSB)/ACK(ISSA)

Enter ESTABLISHED
ACK(ISSB) ->

Enter ESTABLISHED

```

A simultaneous open is:

```

Enter SYN-SENT
SYN(seq=ISSA) ->

Enter SYN-SENT
<- SYN(seq=ISSB)

Enter SYN-RECEIVED
SYN(seq=ISSA)/ACK(ISSB) ->

Enter SYN-RECEIVED
<- SYN(seq=ISSB)/ACK(ISSA)

Enter ESTABLISHED

Enter ESTABLISHED

```

See [RFC793] page 68 and [RFC1122] page 86.

The normal scenario for the proposed four-way handshake is:

```

Enter SYN-SENT
SYN(seq=ISSA) ->

Enter SYN-SENT
<- SYN(seq=ISSB)/ACK(ISSA)

Enter PRE-ESTABLISHED
SYN(seq=ISSA)/ACK(ISSB) ->

Enter ESTABLISHED
<- ACK(ISSA)

Enter ESTABLISHED

```

There are other options for the initial server response in the four-way handshake. Those are discussed in Appendix A as well as the reasons they weren't chosen.

4.2. Changes to the TCP state diagram

The changes can be described entirely as new new state transitions and some additional decisions:

```

LISTEN -> rcv SYN,
    if (allow4way)
        passive4way=1, snd SYN,ACK -> SYN-SENT
    else
        passive4way=0, snd SND,ACK -> SYN-RCVD
SYN-SENT -> rcv ACK
    if (passive4way == 1)
        -> ESTABLISHED
    else
        normal error processing

CLOSED -> active OPEN, create TCB, snd SYN,
        active4way=1 -> SYN-SENT

SYN-SENT -> rcv SYN,ACK
    if (active4way == 1 && (continue4way))
        snd SYN,ACK -> PRE-ESTABLISHED
    else
        snd ACK -> ESTABLISHED

```

The "allow4way" and "continue4way" decisions are based on the contents of the inbound packet.

Instead of overloading the SYN-SENT state and burying the decisions in the existing LISTEN and SYN-SENT states, the state diagram is expanded with two new states, SYN-ACK-SENT and PRE-ESTABLISHED, and two transitional states, ALLOW-4WAY and CONTINUE-4WAY. These are transitional because once entered, an immediate decision is made and then they are exited to a new state.

The LISTEN -> SYN-RCVD transition is replaced by:

```

LISTEN -> rcv SYN -> ALLOW-4WAY

ALLOW-4WAY(YES) -> snd SYN,ACK -> SYN-ACK-SENT
ALLOW-4WAY(NO) -> snd SYN,ACK -> SYN-RCVD

SYN-ACK-SENT -> rcv SYN,ACK, snd ACK -> ESTABLISHED
SYN-ACK-SENT -> rcv ACK of SYN, x -> ESTABLISHED

```

and the SYN-SENT -> ESTABLISHED transition is replaced by:

SYN-SENT -> rcv SYN,ACK -> CONTINUE-4WAY

CONTINUE-4WAY(YES) -> snd SYN,ACK -> PRE-ESTABLISHED

CONTINUE-4WAY(NO) -> snd ACK -> ESTABLISHED

PRE-ESTABLISHED -> rcv RST -> LISTEN

PRE-ESTABLISHED -> rcv ACK of SYN -> ESTABLISHED

PRE-ESTABLISHED -> CLOSE/snd FIN -> FIN WAIT-1

4.3. Three-Way or Four-Way Handshake?

There are two new decision points for for handling a four-way handshake. First, when a connection in LISTEN state receives a SYN packet, it has to decide based on the contents of that packet whether or not the remote side understands the four-way handshake. This is accomplished through the allocation of one of the unused bits in the TCP header, the 4WAY bit.

Note: There are other ways to convey support for the four-way handshake instead of using an unused header bit. These are discussed in Appendix B.

4.3.1. Using the 4WAY Bit

The client sets the 4WAY bit in the initial SYN. If the server receives a 4WAY bit in the initial SYN, then it will set the 4WAY bit in the SYN/ACK. If the client receives a SYN/ACK without the 4WAY bit set, it proceeds with the normal three-way handshake. If it receives a SYN/ACK with the 4WAY bit set, then based on the options in the SYN/ACK it can chose to either proceed with the normal three-way handshake, or to continue with the four-way handshake.

If a packet is received with the 4WAY bit set, but not the SYN bit, the 4WAY bit is ignored. When sending a packet without the SYN bit set, the 4WAY bit must not be set.

[RFC3168] notes TCP interoperability issues with the CWR and ECE bits, but the 4WAY bit does not have the same issues.

4.3.2. Non Four-Way Client Sets 4WAY Bit

In this case, the server might enter SYN-ACK-SENT state. It will respond with a SYN-ACK. Because this looks like the same ACK generated in SYN-RCVD state, it will look to the client like a normal SYN/ACK packet, other than the 4WAY bit, and it will respond with a normal ACK, and the connection will complete with the normal three-way handshake.

4.3.3. Non Four-Way Server Sets 4WAY Bit

If the client decides to not continue a four-way handshake, then it will respond with an ACK and complete the normal three-way handshake. If the client decides that it does want to continue with a four-way exchange, it'll send a SYN/ACK. When the server receives the packet, the normal TCP processing will strip off the SYN, and continue processing as a normal three-way handshake.

4.4. PRE-ESTABLISHED State

When compared to the three-way handshake, the four-way handshake adds an additional RTT before the client side enters ESTABLISHED state. At first glance, this would imply that there will be an additional delay before user data can be delivered. The PRE-ESTABLISHED state addresses this issue.

From [RFC793]:

Several examples of connection initiation follow. Although these examples do not show connection synchronization using data-carrying segments, this is perfectly legitimate, so long as the receiving TCP doesn't deliver the data to the user until it is clear the data is valid (i.e., the data must be buffered at the receiver until the connection reaches the ESTABLISHED state). The three-way handshake reduces the possibility of false connections. It is the implementation of a trade-off between memory and messages to provide information for this checking.

The PRE-ESTABLISHED state is identical to the SYN-RCVD state, except that in PRE-ESTABLISHED state we know that the connection is valid, and hence data can now be delivered to the user. It is the sending of a packet with a SYN and receiving an ACK of that SYN that provides the assurance that the connection is valid, not the transition to ESTABLISHED state. With the three-way handshake, it just happens that this corresponds exactly with entering ESTABLISHED state. With the four-way handshake, the PRE-ESTABLISHED state also corresponds with having sent a SYN and received an ACK of that SYN, so once a connection has entered PRE-ESTABLISHED state it is also safe to deliver user data.

For the socket interface, this means that a connect() call will return upon entering either PRE-ESTABLISHED or ESTABLISHED state, and a subsequent read() on that socket will return any data that has been received. The final completion of the four-way handshake runs in parallel with delivering user data.

5. Negotiating Non-directional vs. Directional TCP Options

TCP options that are negotiated in the initial SYN exchange can be classified as either non-directional or directional. An example of a

non-directional option is the TCP Window Scale option. Negotiating a non-directional TCP option falls naturally into the Four-Way handshake, but allows for more options to be negotiated than will fit into the initial SYN packet when using expanded TCP option space. In order to allow this, the SYN/ACK from the server, with the TCP Extended Data option (EDO) [EDO], can contain initial negotiation for TCP options that weren't received in the initial SYN, which the client can then acknowledge in its SYN/ACK, using the EDO option. Because the options are non-directional, it doesn't matter which side presents it first.

Directional options do not fall as cleanly into the extended four-way handshake. A directional option is one which is originated in the initial SYN, and the server's response in the SYN/ACK is determined in direct response to the inbound option. For example, assume an option FOO that has 100 variants, where servers typically have support for all 100 variants, but clients usually only a small number. The client sends option FOO with a short list of variants that it supports, and then the server chooses which one of those to use, and responds with that variant. If instead the server initiates the option in the SYN/ACK, it'd have to include all 100 variants and let the client choose from that list. In the future, new TCP options would need to be designed to work in the context of the four-way handshake. For existing directional options, it would not be unreasonable to require that they be included in the initial SYN, and other non-directional options would be deferred and negotiated in the SYN/ACK exchange.

5.1. Caching EDO support

One reason for using the Four-Way handshake is to allow use of the EDO option from the client, by giving the client a second chance to send TCP options, and avoid sending the EDO option in a SYN packet to a machine that doesn't understand the EDO option. An implementation could choose to cache information about peers that support the EDO option, allowing successive TCP connections to the same peer to use the EDO option in the initial SYN of a standard Three-Way handshake.

6. TCP Connection State Diagram

The following diagram is modified from the diagram in RFC 793 [RFC793]. In addition to adding the "ALLOW 4WAY?", "CONTINUE 4WAY?" and "SYN-ACK SENT" states, it also includes the three changes listed in RFC 1122 [RFC1122]:

"(a) The arrow from SYN-SENT to SYN-RCVD should be labeled with "snd SYN,ACK", to agree with the text on page 68 and with Figure 8.

(b) There could be an arrow from SYN-RCVD state to LISTEN

state, conditioned on receiving a RST after a passive open (see text page 70).

- (c) It is possible to go directly from FIN-WAIT-1 to the TIME-WAIT state (see page 75 of the spec)."

The SYN-RCVD and PRE-ESTABLISHED states are presented in the same box to simplify the diagram, since the transitions out of SYN-RCVD and PRE-ESTABLISHED are identical.

[illegible]

7. Using TCP Options in the Four-Way Handshake

7.1. New TCP Options in the SYN/ACK

The TCP protocol [RFC793] does not restrict what options may appear in which segments. The TCP protocol [RFC793] only specifies three TCP options: End of option list (EOL), No-Operation (NOP) and Maximum Segment Size (MSS), and all TCP implementations are required to support these options. "Requirements for Internet Hosts -- Communication Layers" [RFC1122] further requires that:

A TCP MUST be able to receive a TCP option in any segment.
A TCP MUST ignore without error any TCP option it does not implement, assuming that the option has a length field (all TCP options defined in the future will have length fields).

Instead, any restrictions on how an individual TCP option is to be used is specified in the definition of each individual TCP option. For example, the Maximum Segment Size option is only sent in packets with the SYN bit set [RFC793]. Prior to TCP Extensions for High Performance [RFC1072], there were no TCP options that were sent in non-SYN packets. At that time there was concern that legacy TCP implementations might not be prepared to process TCP options in non-SYN packets. To preclude that, a method of option enablement was devised: the Window Scale, Echo, and SACK Permitted options had to be exchanged in SYN packets before they could be used in non-SYN packets. (Echo and Echo Reply were replaced by Timestamps [RFC1323][RFC7323].) Additional optimization for this style of option enablement was to specify that for the Three-Way handshake, if an option wasn't received in the SYN, there was no point in putting it into the SYN/ACK since it would never be enabled. Other TCP options have different requirements.

Because option enablement that is non-directional, the Four-Way handshake allows these options to have a second chance to be enabled. The server can include additional options in its SYN/ACK that weren't present in the inbound SYN, and the client can then enable any of those additional options with its SYN/ACK. When generating the SYN/ACKs, both sides know whether or not the other side supports EDO, allowing EDO to be used to include more options than would fit in the original 40 byte option space.

Note that in the Four-Way handshake the client's SYN/ACK can omit RFC7323 style options that were in the initial SYN; if the servers SYN/ACK contained them, those options are enabled, since they were sent and received in a SYN packet. There is no way to retract an offer to enable an option, so when retransmitting a packet with a SYN, it must contain the same set of TCP options that were contained

in the original transmission of that packet, except if the option is otherwise defined. But because the clients SYN and SYN/ACK are separate packets, they can have different sets of TCP options; it is the union of the options in those two packets, intersected with the options in the servers SYN/ACK, that determine which RFC7323 style options will be enabled.

7.2. Handling Unknown TCP Options

One of the concerns since RFC 1072 [RFC1072] has been whether or not legacy TCP implementations will properly ignore unknown TCP options. As has already been stated, this has been a requirement for over 20 years, but people continue to worry about bad interactions.

Any implementation which indicates support for the Four-Way Handshake is also indicating that it properly handles unknown TCP options. This includes not only ignoring TCP options unknown to the implementation, but also properly handling known options that are received at unexpected points in the TCP stream, or that have been explicitly disabled for a specific connection.

7.2.1. Unknown Option Option

One challenge with sending new options that were not enabled during the initial SYN exchange is how to decide whether or not the other side supports the option, since the other side will just silently ignore any options it doesn't understand, so the sender has to infer non-support by the absence of any response.

The Unknown Option option is an advisory only option, that allows a receiving TCP to give explicit indication that it has received a TCP option that it does not understand.

Kind: TBD

Length: N >= 2

```

+-----+-----+-----+ - - - +
| Kind=TBD|Length=N | option1 | ...   |
+-----+-----+-----+ - - - +

```

When a TCP receives TCP options that are not supported for this connection, in addition to silently ignoring the options [RFC1122], the TCP MAY include an Unknown Option option in the next packet it sends. It MAY generate a new ACK-only packet to send the Unknown Option option, or it MAY piggy-back it on the next packet it sends.

When a TCP receives an Unknown Option option, it SHOULD NOT send the indicated options in future packets, provided the definition

of the indicated option allows for that action. For example, when TCP-AO is being used, it is sent in every packet, so receiving an Unknown Option option indicating that TCP-AO is not supported MUST be ignored. A counter example is the UTO option [RFC5482]. If an Unknown Option option is received indicating that UTO is not supported, then the sending TCP SHOULD NOT send any more UTO options.

A TCP MUST NOT assume that a given option that it sent is supported, solely by not receiving an Unknown Option option.

7.3. TCP options

7.3.1. RFC defined options

TCP options that are defined in RFC documents include:

7.3.1.1. End of Option List

The End of Option List option [RFC793] may be present in any packet, all implementations must support EOL.

7.3.1.2. No-Operation

The No-Operation [RFC793] option may be present in any packet, all implementations must support NOP.

7.3.1.3. Maximum Segment Size

The MSS option [RFC793] only appears in SYN packets and is not negotiated, all implementations are required to support the MSS option.

7.3.1.4. Window Scale

The Window Scale option [RFC7323] uses RFC7323 style enablement, and is only sent in SYN packets. If not enabled, its contents are ignored.

7.3.1.5. SACK Permitted

The SACK Permitted option [RFC2018] uses RFC7323 style enablement, and is only sent in SYN packets. It enables use of the SACK option in non-SYN packets.

7.3.1.6. SACK

The SACK option [RFC2018] is only sent in non-SYN packets if enabled by the SACK Permitted option.

7.3.1.7. Timestamps

The Timestamps option [RFC7323] uses RFC7323 style enablement; once enabled, it is included in every packet.

7.3.1.8. MD5 Signature Option

Use of the MD5 Signature Option [RFC2385] is not negotiated. It is sent in every packet, its absence in the SYN/ACK does not disable the option, but causes the SYN/ACK to be silently dropped.

7.3.1.9. Quick-Start Response

Use of the Quick-Start Response option [RFC4782] is not negotiated. The receipt of the IP Quick-Start option implies support for the TCP Quick-Start Response option.

7.3.1.10. User Timeout Option

Though the User Timeout Option (UTO) option [RFC5482] may be exchanged in SYN packets, it is not negotiated, and may still be sent in non-SYN packets if the application has requested UTO. It relies on the fact that unknown TCP options are to be ignored [RFC1122].

7.3.1.11. TCP Authentication Options (TCP-AO)

The TCP-AO option [RFC5925] is not negotiated. The application specifies that TCP-AO is to be use and the Master Key Tuple (MKT) configuration controls when TCP-AO is to be used, and how to handle inbound packets that arrive either without TCP-AO or with an unknown MKT.

7.3.1.12. Multipath TCP (MPTCP)

The MPTCP option [RFC6824] uses a variation of RFC7323 style enablement. Each side includes the MPTCP option with its own Key for the connection in the SYN packets, if both sent and received, the final ACK contains an MPTCP option with both keys.

7.3.1.13. RFC3692-style Experiment

Two TCP options [RFC4727] are reserved for experimentation, and so whether or not they are negotiated is determined by the experiment being run.

7.3.1.14. TCP Fast Open Option

The TCP Fast Open [TFO] option spans multiple TCP connections, and is uni-directional. The first instance of it without a Cookie is used by the client to get a Cookie from the server, which is saved and can

then be used in subsequent TCP connections to allow data in the SYN-only packet to be delivered to the application before the 3WHS has been completed.

Since the Cookie information is being saved, the TCP can also save with the Cookie whether or not the other side supports the EDO option, and if so, future connections to that host can safely make use of the EDO option in the initial SYN-only packet, since it is known that the other side supports it.

In addition, the client might not include a TFO=INIT option in its first SYN-only packet due to option space limitations. But because the client has indicated support for the Four-Way handshake, the server can still safely send back in its SYN/ACK a TFO=Cookie option, allowing for TFO initialization in the client.

7.3.1.15. T/TCP Options

T/TCP [RFC1644] defines three TCP options: CC (connection count), CC.NEW and CC.ECHO. Though now obsolete [RFC6247], this is a good example of a directional TCP option. The CC or CC.NEW option is sent in the initial SYN, and the responding SYN/ACK has a CC.ECHO option that contains the connection count from the received CC or CC.NEW option. This is not an RFC7323 style enablement, and only fits into the Four-Way handshake if the CC or CC.NEW option is included in the initial SYN packet, the servers SYN/ACK can't contain a CC.ECHO if it didn't receive a CC option. However, T/TCP is aimed at reducing the Three-Way handshake to a two packet exchange, and needs to keep state about which hosts can utilize T/TCP. As such, the client could keep state as to which servers support the EDO option, and then be able make use of the EDO option in its initial SYN for future connections to those servers.

8. IANA Considerations

A Kind value needs to be allocated for the Unknown Option Option.

9. Security Considerations

TBD

10. References

10.1. Normative References

[RFC793] Postel, J., "Transmission Control Protocol - DARPA Internet Program Protocol Specification", RFC 793, DARPA, September 1981.

- [RFC1122] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989, <<http://www.rfc-editor.org/info/rfc1122>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.

10.2. Informative References

- [Borman14] Borman, D., "Re: [tcpm] New Version Notification for draft-touch-tcpm-tcp-edo-01.txt", message to the TCPM mailing list, 22 May 2014, <<http://www.ietf.org/mail-archive/web/tcpm/current/msg08804.html>>.
- [RFC1072] Jacobson, V., and R.T. Braden, "TCP extensions for long-delay paths", RFC 1072, October 1988, <<http://www.rfc-editor.org/info/rfc1072>>.
- [RFC1323] Jacobson, V., Braden, R., and D. Borman. "TCP Extensions for High Performance." RFC 1323, May 1992, <<http://www.rfc-editor.org/info/rfc1323>>.
- [RFC1644] R. Braden, "T/TCP -- TCP Extensions for Transactions Functional Specification", RFC 1644, July 1994, <<http://www.rfc-editor.org/info/rfc1644>>.
- [RFC2018] Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, "TCP Selective Acknowledgment Options", RFC 2018, October 1996, <<http://www.rfc-editor.org/info/rfc2018>>.
- [RFC2385] A. Heffernan, "Protection of BGP Sessions via the TCP MD5 Signature Option", RFC 2385, August 1998, <<http://www.rfc-editor.org/info/rfc2385>>.
- [RFC4727] B. Fenner, "Experimental Values In IPv4, IPv6, ICMPv4, ICMPv6, UDP, and TCP Headers", RFC 4727, November 2006, <<http://www.rfc-editor.org/info/rfc4727>>.
- [RFC4782] Floyd, S., Allman, M., Jain, A., and P. Sarolahti, "Quick-Start for TCP and IP", RFC 4782, January 2007, <<http://www.rfc-editor.org/info/rfc4782>>.
- [RFC5482] Eggert, L., and F. Gont, "TCP User Timeout Option", RFC 5482, March 2009, <<http://www.rfc-editor.org/info/rfc5482>>.
- [RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, June 2010, <<http://www.rfc-editor.org/info/rfc5925>>.

- [RFC6247] L. Eggert, "Moving the Undeployed TCP Extensions RFC 1072, RFC 1106, RFC 1110, RFC 1145, RFC 1146, RFC 1379, RFC 1644, and RFC 1693 to Historic Status", RFC 6247, May 2011, <<http://www.rfc-editor.org/info/rfc6247>>.
- [RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, January 2013, <<http://www.rfc-editor.org/info/rfc6824>>.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001, <<http://www.rfc-editor.org/info/rfc3168>>.
- [RFC7323] Borman, D., Braden, R., Jacobson, V., and R. Scheffenegger, Ed., "TCP Extension for High Performance", RFC 7323, September 2014, <<http://www.rfc-editor.org/info/rfc7323>>.
- [TFO] Cheng, Y., Jhu, J., Radhakrishnan, S., and A. Jain, "TCP Fast Open", RFC 7413, December 2014, <<http://www.rfc-editor.org/info/rfc7413>>.
- [EDO] Joe Touch, J., and W. Eddy, "TCP Extended Data Offset Option", Work in Progress, draft-ietf-tcpm-tcp-edo-01.txt, October 2014.
- [Kohler04] Kohler, E, "Extended Option Space for TCP" Work in Progress, draft-kohler-tcpm-extopt-00.txt, September 2004.
- [Touch14] Touch, J., Briscoe, B., and T. Faber, "TCP SYN Extended Option Space in the Payload of a Supplementary Segment", Work in Progress, draft-touch-tcpm-tcp-syn-ext-opt-01.txt, September 2014.
- [Eddy08] Eddy, W., and A. Langley, "Extending the Space Available for TCP Options", Work in Progress, draft-eddy-tcp-loo-04, July 2008.
- [Yourtchenko11] Yourtchenko, A., "Introducing TCP Long Options by Invalid Checksum", Work in Progress, draft-yourtchenko-tcp-loic-00.txt, April 2011.
- [Ramaiah12] Ramaiah, A., "TCP option space extension", Work in Progress, draft-ananth-tcpm-tcptext-00.txt, March 2012.
- [Briscoe14] Briscoe, B., "Extended TCP Option Space in the Payload of an Alternative SYN", Work in Progress, draft-briscoe-tcpm-syn-op-sis-02, September 2014.

Appendix A. First Response of the Four-Way Handshake

For a connection with ISS values of ISSA from the client and ISSB from the server, three different options for the first server response were considered:

- (1) SYN(seq=ISSB)
- (2) SYN(seq=ISSB)/ACK(seq=ISSA-1)
- (3) SYN(Seq=ISSB)/ACK(seq=ISSA)

SYN(seq=ISSB)

The original idea for the four-way handshake was to have the server do a simple turn-around of the TCP three-way handshake, by responding to the initial SYN with another bare SYN. Because it had already received a SYN and knows that the client supports the four-way handshake, it could respond with a plain SYN, making use of header modifying options that the client had indicated it supported. This is similar to a simultaneous open, except the server is able to transition from SYN-SENT to ESTABLISHED instead of going through SYN-RECEIVED state.

Enter SYN-SENT
SYN(seq=ISSA) ->

Enter SYN-SENT
<- SYN(seq=ISSB)

Enter SYN-RECEIVED
SYN(seq=ISSA)/ACK(ISSB) ->

Enter ESTABLISHED
<- ACK(ISSA)

Enter ESTABLISHED

The problems with this approach are that it forces the full four-way handshake, and a middle-box in the path might block the returning bare SYN.

SYN(seq=ISSB)/ACK(seq=ISSA-1)

This response also turns the three-way handshake into something that looks a lot like a simultaneous open, since the ACK does not acknowledge the SYN. The disadvantage is that it also forces a full four-way handshake, since it does not acknowledge the initial SYN. However, this should work better for getting through a middle-box since it is not a bare SYN. But if the middle-box is digging into the TCP packet and tries to verify the ACK field, it might still block this packet since it is not the expected ACK field of the normal three-way handshake.

SYN(seq=ISSB)/ACK(seq=ISSA)

This response looks like the normal three-way handshake response, which gives the client the ability to choose whether to complete the three-way handshake by sending an ACK(ISSB), or continue the four-way handshake by responding with SYN(seq=ISSA)/ACK(ISSB). The advantage of this option is that it doesn't always force the four-way handshake, and to a middle-box the packets look like the normal TCP packets that it expects to see.

The third option offers the least possibility that middle-boxes will block the packets, and also leaves the flexibility for deciding on a three-way or four-way handshake up to the client. Because it is to the client's benefit to have a four-way handshake, it should be the one to decide whether or not the four-way handshake is needed for a particular handshake.

Appendix B. Communicating Four-Way Handshake Support

Besides allocating a 4WAY bit in the TCP header, two other options were considered for communicating support for the four-way handshake:

Create a new 4WAY TCP option

This does not have the interoperability issues that the 4WAY TCP bit has, because it is assumed that connections will not send unknown TCP options. The disadvantage of this is that it requires two more bytes out of the TCP option space.

Implied support by other TCP options

The primary motivation for the four-way handshake is to give the client a second chance to send TCP options in a SYN. This is intended for use with the new TCP EDO option, and the presence of the EDO option could imply support for the four-way handshake. This allows the client to send additional TCP options using the TCP EDO option in a SYN/ACK packet.

Acknowledgments

TBD

Contributors

TBD

Author's Address

David Borman
Quantum Corporation
1155 Centre Pointe Drive, Suite 1
Mendota Heights, MN 55120

Phone: (651) 688-4394
Email: david.borman@quantum.com

