Describing TCP with Augmented Packet Header Diagrams
draft-mcquistin-augmented-tcp-example-00

Abstract

   This document describes TCP, and a number of its extensions, using
   Augmented Packet Header Diagrams. This document is an example of the
   Augmented Packet Header Diagram language: it is not intended as a
   contribution to any ongoing or future work on maintaining or
   extending TCP.

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 https://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 6 May 2021.

Copyright Notice

Table of Contents

1. Introduction

   This document uses Augmented Packet Header Diagrams
   [AUGMENTED-DIAGRAMS] to describe TCP [RFC793], and is intended to
   further discussion about the design and implementation of the
   Augmented Packet Header Diagram language and tooling. Given this
   purpose, this document is not intended as a contribution to any
   ongoing or future work on maintaining or extending TCP. Further,
   this document does not necessarily reflect TCP, and its extensions,
   as presently standardised.

2. TCP Header

   This document describes the TCP protocol. The TCP protocol uses TCP
   Headers.

   A TCP Header is formatted as follows:

```
    0                   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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          Source Port          |       Destination Port        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                        Sequence Number                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Acknowledgment Number                      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Data  |       |C|E|U|A|P|R|S|F|                               |
   | Offset| Rsrvd |W|C|R|C|S|S|Y|I|          Window Size           |
   |       |       |R|E|G|K|H|T|N|N|                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           Checksum            |        Urgent Pointer         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           [Options]                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               :
   :                           Payload                             :
   :                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

Source Port: 16 bits. The source port number.

Destination Port: 16 bits. The destination port number.

Sequence Number: 32 bits. The sequence number of the first data
   octet in this segment (except when the SYN flag is set). If SYN
   is set the sequence number is the initial sequence number (ISN)
   and the first data octet is ISN+1.

Acknowledgment Number: 32 bits. If the ACK control bit is set, this
   field contains the value of the next sequence number the sender of
   the segment is expecting to receive. Once a connection is
   established, this is always sent.

Data Offset (DOffset): 4 bits. 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.

Reserved (Rsrvd): 4 bits. A set of control bits reserved for future
   use. Must be zero in generated segments and must be ignored in
   received segments, if corresponding future features are
   unimplemented by the sending or receiving host.

   CWR: 1 bit. Congestion Window Reduced

   ECE: 1 bit. ECN-Echo

   URG: 1 bit. Urgent Pointer field significant

   ACK: 1 bit. Acknowledgment field significant.

   PSH: 1 bit. Push Function (see the Send Call description)

   RST: 1 bit. Reset the connection

   SYN: 1 bit. Synchronize sequence numbers

   FIN: 1 bit. No more data from sender.

   Window Size: 16 bits. The number of data octets beginning with the
      one indicated in the acknowledgment field that the sender of this
      segment is willing to accept.

      The window size MUST be treated as an unsigned number, or else
      large window sizes will appear like negative windows and TCP will
      not work (MUST-1). It is RECOMMENDED that implementations will
      reserve 32-bit fields for the send and receive window sizes in the
      connection record and do all window computations with 32 bits
      (REC- 1).

   Checksum: 16 bits. The checksum field is the 16 bit one's complement
      of the one's complement sum of all 16 bit words in the header and
      text. The checksum computation needs to ensure the 16-bit
      alignment of the data being summed. If a segment contains an odd
      number of header and text octets, alignment can be achieved by
      padding the last octet with zeros on its right to form a 16 bit
      word for checksum purposes. The pad is not transmitted as part of
      the segment. While computing the checksum, the checksum field
      itself is replaced with zeros.

   Urgent Pointer: 16 bits. This field communicates the current value
      of the urgent pointer as a positive offset from the sequence
      number in this segment. The urgent pointer points to the sequence
      number of the octet following the urgent data. This field is only
      be interpreted in segments with the URG control bit set.

   Options: [TCP Option]; Options#Size == (DOffset-5)*32; present
   only when DOffset > 5. Options may occupy space at the end of the
      TCP header and are a multiple of 8 bits in length. All options
      are included in the checksum.

   Payload. Payload.

3. TCP Options

   A TCP Option is one of: a EOL Option, a NOOP Option, a Maximum
   Segment Size Option, a Window Scale Factor Option, a Timestamp
   Option, or a SACK Permitted Option.

   An EOL Option is formatted as follows:

```
0                   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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       0       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   where:

   Option Kind (Kind): 1 byte; Kind == 0. This option code indicates
      the end of the option list.

   A NOOP Option is formatted as follows:

```
0                   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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       1       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   where:

   Option Kind (Kind): 1 byte; Kind == 1. This option code can be used
      between options, for example, to align the beginning of a
      subsequent option on a word boundary.

   A Maximum Segment Size Option is formatted as follows:

```
0                   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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       2       |     Length    |    Maximum Segment Size (MSS)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   where:

   Option Kind (Kind): 1 byte; Kind == 2. If this option is present,
      then it communicates the maximum receive segment size at the TCP
      endpoint that sends this segment.

   Option Length (Length): 1 byte; Length == 4. Option length.

   Maximum Segment Size (MSS): 2 bytes. The maximum segment size
      allowed.

   A Window Scale Factor Option is formatted as follows:

```
0                   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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       3       |     Length    |      Window Scale Factor      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   where:

   Option Kind (Kind): 1 byte; Kind == 3. If present, this option
      carries the window scale factor.

   Option Length (Length): 1 byte; Length == 3. Option length.

   Window Scale Factor: 1 byte. Window scale factor.

   A Timestamp Option is formatted as follows:

```
0                   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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       8       |     Length    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Timestamp value (TSval)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Timestamp echo reply (TSecr)                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
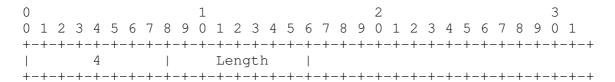
   where:

   Option Kind (Kind): 1 byte; Kind == 8. If present, this option
      carries a timestamp and an echoed timestamp.

   Option Length (Length): 1 byte; Length == 10. Option length.

   Timestamp value (TSval): 4 bytes. TSval.

   Timestamp echo reply (TSecr): 4 bytes. TSecr.

   A SACK Permitted Option is formatted as follows:

```
 0                   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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       4       |     Length    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

Option Kind (Kind): 1 byte; Kind == 4. If present, this option
   indicates that SACK is permitted.

Option Length (Length): 1 byte; Length == 2. Option length.

4. Comparison with draft-ietf-tcpm-rfc793bis

   While the purpose of this document is to give an example use of the
   Augmented Packet Header Diagrams, and not to contribute to ongoing or
   future TCP standardisation efforts, it is instructive to draw
   comparisons with documents that describe the same protocol. In
   particular, the descriptions of the TCP header (Section 2) and TCP
   options (Section 3) from this document can be compared with
   Section 3.1 of [draft-ietf-tcpm-rfc793bis]. We have adopted the same
   field descriptions to help with this comparison.

   Beyond the broad similarities, there are a number of elements where
   this document describes the protocol differently. For example, the
   TCP header as defined in [draft-ietf-tcpm-rfc793bis] contains a
   variable-length padding field which is not present in the description
   contained in this document. The Padding field is used to pad the TCP
   header to a 32-bit boundary, and to ensure that the Payload begins on
   a 32-bit boundary. The length of the Padding field is the difference
   between the value of the Data Offset field and the length of the
   Options field. However, the length of the Options field is itself
   dependent on the value of the Data Offset field and the particular
   set of options contained within an instance of the TCP header. The
   Augmented Packet Header Diagram format cannot express this logic: the
   lengths of both the Padding and Payload fields would be variable. To
   overcome this, as shown in Section 2, the description of TCP given in
   this document relies upon End of List options being syntactically
   identical to Padding (i.e., being all zeroes). This highlights the
   limitations of the Augmented Packet Header Diagram format: prose text
   is much more expressive.

However, the precision required by the Augmented Packet Header
Diagram can be beneficial. For example, the description of the
Options field in this document sets out an explicit length and
presence constraint, where [draft-ietf-tcpm-rfc793bis] indicates that
it is "variable". Similar constraints exist in the description given
in [draft-ietf-tcpm-rfc793bis], but they are contained with the prose
description of the field.

In summary, the most striking comparison to be made is that the
description of TCP given in this document is largely similar to that
given in [draft-ietf-tcpm-rfc793bis]. By retaining the overall
structure of a packet header diagram and an accommpanying list of
descriptions for each field, the Augmented Packet Header Diagram
format is close to that already in use. This is encouraging: it
indicates that the Augmented Packet Header Diagram language can be
used to describe realistic protocols, without introducing a format
that is unfamiliar or difficult to write.

5. IANA Considerations

   This document contains no actions for IANA.

6. Security Considerations

   The security implications of the Augmented Packet Header Diagrams
   format are considered in [AUGMENTED-DIAGRAMS].

7. Acknowledgements

   This work has received funding from the UK Engineering and Physical
   Sciences Research Council under grant EP/R04144X/1.

8. Informative References

   [AUGMENTED-DIAGRAMS]
              McQuistin, S., Band, V., Jacob, D., and C. S. Perkins,
              "Describing Protocol Data Units with Augmented Packet
              Header Diagrams", Work in Progress, Internet-Draft, draft-
              mcquistin-augmented-ascii-diagrams-07, 2 November 2020,
              <http://www.ietf.org/internet-drafts/draft-mcquistin-
              augmented-ascii-diagrams-07.txt>.

   [RFC793]   Postel, J., "Transmission Control Protocol", RFC 793,
              September 1981, <https://www.rfc-editor.org/info/rfc793>.

   [draft-ietf-tcpm-rfc793bis]
              Eddy, W., "Transmission Control Protocol (TCP)
              Specification", Work in Progress, Internet-Draft, draft-

                ietf-tcpm-rfc793bis-19, 27 October 2020,
                <http://www.ietf.org/internet-drafts/draft-ietf-tcpm-
                rfc793bis-19.txt>.

Appendix A. Source code repository

    The source code for tooling that can be used to parse this document,
    and generate parser code for the protocol it describes, is available
    from https://github.com/glasgow-ipl/ips-protodesc-code.

Authors' Addresses

    Stephen McQuistin
    University of Glasgow
    School of Computing Science
    Glasgow
    G12 8QQ
    United Kingdom

    Email: sm@smcquistin.uk

    Vivian Band
    University of Glasgow
    School of Computing Science
    Glasgow
    G12 8QQ
    United Kingdom

    Email: vivianband0@gmail.com

    Dejice Jacob
    University of Glasgow
    School of Computing Science
    Glasgow
    G12 8QQ
    United Kingdom

    Email: d.jacob.1@research.gla.ac.uk

   Colin Perkins
   University of Glasgow
   School of Computing Science
   Glasgow
   G12 8QQ
   United Kingdom

   Email: csp@csperkins.org