TCPM Working Group   O. Bonaventure, Ed.
Internet-Draft   Tessares
Intended status: Experimental   M. Boucadair, Ed.
Expires: April 21, 2019   Orange
                                                          S. Gundavelli
                                                                  Cisco
                                                                S. Seo
                                                         Korea Telecom
                                                      October 18, 2018

                         0-RTT TCP Convert Protocol
                      draft-ietf-tcpm-converters-03

Abstract

   This document specifies an application proxy, called Transport
   Converter, to assist the deployment of TCP extensions such as
   Multipath TCP. This proxy is designed to avoid inducing extra delay
   when involved in a network-assisted connection (that is, 0-RTT).
   This specification assumes an explicit model, where the proxy is
   explicitly configured on hosts.

   -- Editorial Note (To be removed by RFC Editor)

   Please update these statements with the RFC number to be assigned to
   this document:
   [This-RFC]

   Please update TBA statements with the port number to be assigned to
   the Converter Protocol.

Status of This Memo

Table of Contents

1. Introduction

Transport protocols like TCP evolve regularly [RFC7414]. TCP has
been improved in different ways. Some improvements such as changing
the initial window size [RFC6928] or modifying the congestion control
scheme can be applied independently on clients and servers. Other
improvements such as Selective Acknowledgements [RFC2018] or large
windows [RFC7323] require a new TCP option or to change the semantics
of some fields in the TCP header. These modifications must be
deployed on both clients and servers to be actually used on the
Internet. Experience with the latter TCP extensions reveals that
their deployment can require many years. [Fukuda2011] reports
results of a decade of measurements showing the deployment of
Selective Acknowledgements, Window Scale and TCP Timestamps.
[ANRW17] describes measurements showing that TCP Fast Open [RFC7413]
(TFO) is still not widely deployed.

There are some situations where the transport stack used on clients
(resp. servers) can be upgraded at a faster pace than the transport
stack running on servers (resp. clients). In those situations,
clients would typically want to benefit from the features of an
improved transport protocol even if the servers have not yet been
upgraded and conversely. In the past, Performance Enhancing Proxies
have been proposed and deployed [RFC3135] as solutions to improve TCP
performance over links with specific characteristics.

Recent examples of TCP extensions include Multipath TCP
[RFC6824][I-D.ietf-mptcp-rfc6824bis] or TCPINC
[I-D.ietf-tcpinc-tcpcrypt]. Those extensions provide features that
are interesting for clients such as wireless devices. With Multipath
TCP, those devices could seamlessly use WLAN and cellular networks,
for bonding purposes, faster handovers, or better resiliency.
Unfortunately, deploying those extensions on both a wide range of
clients and servers remains difficult.

More recently, experimentation of 5G bonding, which has very scarce
coverage, has been conducted into global range of the incumbent 4G
(LTE) connectivity in newly devised clients using Multipath TCP
proxy. Even if the 5G and the 4G bonding by using Multipath TCP
increases the bandwidth to data transfer, it is as well crucial to
minimize latency for all the way between endhosts regardless of
whether intermediate nodes are inside or outside of the mobile core.
In order to handle uRLLC (Ultra-Reliable Low-Latency Communication)
for the next generation mobile network, Multipath TCP and its proxy
mechanism must be optimised to reduce latency.

This document specifies an application proxy, called Transport
Converter. A Transport Converter is a function that is installed by
a network operator to aid the deployment of TCP extensions and to
provide the benefits of such extensions to clients. A Transport
Converter may support conversion service for one or more TCP
extensions. This service is provided by means of the Converter
Protocol (Convert), that is an application layer protocol which uses
TBA TCP port number (Section 8).

The Transport Converter adheres to the main principles as drawn in
[RFC1919]. In particular, the Converter achieves the following:

o Listen for client sessions;

o Receive from a client the address of the final target server;

o Setup a session to the final server;

o Relay control messages and data between the client and the server;

o Perform access controls according to local policies.

The main advantage of network-assisted Converters is that they enable
new TCP extensions to be used on a subset of the end-to-end path,
which encourages the deployment of these extensions. The Transport
Converter allows the client and the server to directly negotiate TCP
options.

The Convert Protocol is a generic mechanism to provide 0-RTT
conversion service. As a sample applicability use case, this
document specifies how the Convert Protocol applies for Multipath
TCP. It is out of scope of this document to provide a comprehensive
list of potential all conversion services; separate documents may be
edited in the future for other conversion services upon need.

This document does not assume that all the traffic is eligible to the
network-assisted conversion service. Only a subset of the traffic
will be forwarded to a Converter according to a set of policies.
Furthermore, it is possible to bypass the Converter to connect to the
servers that already support the required TCP extension.

This document assumes that a client is configured with one or a list
of Converters (e.g., [I-D.boucadair-tcpm-dhc-converter]).
Configuration means are outside the scope of this document.

This document is organized as follows. We first provide a brief
explanation of the operation of Transport Converters in Section 3.
We describe the Converter Protocol in Section 4. We discuss in
Section 5 how Transport Converters can be used to support different
TCP options. We then discuss the interactions with middleboxes
(Section 6) and the security considerations (Section 7).

Appendix A provides a comparison with SOCKS proxies that are already
used to deploy Multipath TCP in some cellular networks.

2. Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and
"OPTIONAL" in this document are to be interpreted as described in
[RFC2119] [RFC8174] when, and only when, they appear in all capitals,
as shown here.

3. Architecture

3.1. Functional Elements

The architecture considers three types of endhosts:

o Client endhosts;

o Transport Converters;

o Server endhosts.

A Transport Converter is a network function that relays all data
exchanged over one upstream connection to one downstream connection
and vice versa (Figure 1). The Converter, thus, maintains state that
associates one upstream connection to a corresponding downstream
connection.

A connection can be initiated from both sides of the Transport
Converter (Internet-facing interface, client-facing interface).

```
                    +------------+
   <--- upstream --->| Transport  |<--- downstream --->
                    | Converter  |
                    +------------+
```

Figure 1: A Transport Converter relays data between pairs of TCP
                          connections

Transport Converters can be operated by network operators or third
parties. Nevertheless, this document focuses on the single
administrative deployment case where the entity offering the
connectivity service to a client is also the entity which owns and
operates the Transport Converter.

A Transport Converter can be embedded in a standalone device or be
activated as a service on a router. How such function is enabled is
deployment-specific (Figure 2).

```
             +-+     +-+     +-+
   Client -  |R| -- |R| -- |R| - - -  Server
             +-+     +-+     +-+
                      |
                  Transport
                  Converter
```

Figure 2: A Transport Converter can be installed anywhere in the
                            network

The architecture assumes that new software will be installed on the
Client hosts and on Transport Converters. Further, the architecture
allows for making use of TCP new extensions if those are supported by
a given server.

The Client is configured, through means that are outside the scope of
this document, with the names and/or the addresses of one or more
Transport Converters.

The architecture does not mandate anything on the server side.

Similar to address sharing mechanisms, the architecture does not interfere with end-to-end TLS connections between the client and the server.

One of the benefits of this design is that different transport protocol extensions can be used on the upstream and the downstream connections. This encourages the deployment of new TCP extensions until they are widely supported by servers.

3.2. Theory of Operation

At a high level, the objective of the Transport Converter is to allow the Client to use a specific extension, e.g. Multipath TCP, on a subset of the end-to-end path even if the Server does not support this extension. This is illustrated in Figure 3 where the Client initiates a Multipath TCP connection with the Converter (Multipath packets are shown with "===") while the Converter uses a regular TCP connection with the Server.

The packets belonging to the pair of connections between the Client and Server passing through a Transport Converter may follow a different path than the packets directly exchanged between the Client and the Server. Deployments should minimize the possible additional delay by carefully selecting the location of the Transport Converter used to reach a given destination.

```
                              Transport
Client                        Converter                       Server
     ======================>

                              -------------------->

                              <--------------------

     <======================
       Multipath TCP packets     Regular TCP packets
```

           Figure 3: Different TCP variants can be used on the Client-Converter
                   path and on the Converter-Server path

When establishing a connection, the Client can, depending on local policies, either contact the Server directly (e.g., by sending a TCP SYN towards the Server) or create the connection via a Transport Converter. In the latter case, which is the case we consider in this document, the Client initiates a connection towards the Transport Converter and indicates the IP address and port number of the

ultimate Server inside the connection establishment packet. Doing so
enables the Transport Converter to immediately initiate a connection
towards that Server, without experiencing an extra delay. The
Transport Converter waits until the confirmation that the Server
agrees to establish the connection before confirming it to the
Client.

The client places the destination address and port number of the
target Server in the payload of the SYN sent to the Converter by
leveraging TCP Fast Open [RFC7413]. In accordance with [RFC1919],
the Transport Converter maintains two connections that are combined
together:

o   the upstream connection is the one between the Client and the
    Transport Converter.

o   the downstream connection is between the Transport Converter and
    the remote Server.

Any user data received by the Transport Converter over the upstream
(resp., downstream) connection is relayed over the downstream (resp.,
upstream) connection.

Figure 4 illustrates the establishment of a TCP connection by the
Client through a Transport Converter. The information shown between
brackets is part of the Converter Protocol described later in this
document.

```
                             Transport
Client                       Converter                         Server
     -------------------->
      SYN TFO [->Server:port]

                               -------------------->
                                     SYN

                               <--------------------
                                   SYN+ACK
     <--------------------
       SYN+ACK [ ]
```

      Figure 4: Establishment of a TCP connection through a Converter

The Client sends a SYN destined to the Transport Converter. This SYN
contains a TFO cookie and inside its payload the addresses and ports
of the destination Server. The Transport Converter does not reply
immediately to this SYN. It first tries to create a TCP connection

towards the destination Server. If this second connection succeeds,
the Transport Converter confirms the establishment of

the connection to the Client by returning a SYN+ACK and the first
bytes of the bytestream contain information about the TCP options
that were negotiated with the final Server. This information is sent
at the beginning of the bytestream, either directly in the SYN+ACK or
in a subsequent packet. For graphical reasons, the figures in this
section show that the Converter returns this information in the
SYN+ACK packet. An implementation could also place this information
in a packet that it sent shortly after the SYN+ACK.

The connection can also be established from the Internet towards a
Client via a Transport Converter. This is typically the case when
the Client embeds a server (video server, for example).

The procedure described in Figure 4 assumes that the Client has
obtained a TFO cookie from the Transport Converter. This is part of
the Bootstrap procedure which is illustrated in Figure 5. The Client
sends a SYN with a TFO request option to obtain a valid cookie from
the Converter. The Converter replies with a TFO cookie in the
SYN+ACK. Once this connection has been established, the Client sends
a Bootstrap message to request the list of TCP options for which the
Transport Converter provides a conversion service.

```
                          Transport
Client                    Converter                        Server
      -------------------->
       SYN TFO(empty)

      <--------------------
        SYN+ACK TFO(cookie)

      -------------------->
          [Bootstrap]

      <--------------------
        [Supported TCP Extension Services]
```

Figure 5: Bootstrapping a Client connection to a Transport Converter

Note that the Converter may rely on local policies to decide whether
it can service a given requesting Client. That is, the Converter
will not return a cookie for that Client. How such policies are
supplied to the Converter are out of scope.

Also, the Converter may behave in a cookie-less mode when appropriate
means are enforced at the Converter and the network in-between to
protect against attacks such as spoofing and SYN flood. Under such
deployments, the use of TFO is not required.

3.3. Sample Examples of Outgoing Converter-Assisted Multipath TCP Connections

As an example (Figure 6), let us consider how the Convert protocol
can help the deployment of Multipath TCP [RFC6824]. We assume that
both the Client and the Transport Converter support Multipath TCP,
but consider two different cases depending whether the Server
supports Multipath TCP or not. A Multipath TCP connection is created
by placing the MP_CAPABLE (MPC) option in the SYN sent by the Client.

Figure 6 describes the operation of the Transport Converter if the
Server does not support Multipath TCP.

```
                             Transport
Client                       Converter                        Server
    -------------------->
    SYN, MPC [->Server:port]

                                  -------------------->
                                        SYN, MPC

                                  <--------------------
                                        SYN+ACK
    <--------------------
      SYN+ACK,MPC [ ]

    -------------------->
        ACK,MPC
                                  -------------------->
                                          ACK
```

Figure 6: Establishment of a Multipath TCP connection through a
Converter

The Client tries to initiate a Multipath TCP connection by sending a
SYN with the MP_CAPABLE option (MPC in Figure 6). The SYN includes
the address and port number of the final Server and the Transport
Converter attempts to initiate a Multipath TCP connection towards
this Server. Since the Server does not support Multipath TCP, it
replies with a SYN+ACK that does not contain the MP_CAPABLE option.
The Transport Converter notes that the connection with the Server

does not support Multipath TCP and returns the TCP options received
from the Server to the Client.

Figure 7 considers a Server that supports Multipath TCP. In this
case, it replies to the SYN sent by the Transport Converter with the
MP_CAPABLE option. Upon reception of this SYN+ACK, the Transport
Converter confirms the establishment of the connection to the Client
and indicates to the Client that the Server supports Multipath TCP.
With this information, the Client has discovered that the Server
supports Multipath TCP natively. This will enable it to bypass the
Transport Converter for the next Multipath TCP connection that it
will initiate towards this Server.

```
                               Transport
   Client                      Converter                       Server
        -------------------->
        SYN, MPC [->Server:port]

                                   -------------------->
                                        SYN, MPC

                                   <--------------------
                                        SYN+ACK, MPC
        <--------------------
          SYN+ACK, MPC [ MPC supported ]

        -------------------->
            ACK, MPC
                                   -------------------->
                                        ACK, MPC
```

        Figure 7: Establishment of a Multipath TCP connection through a
                               converter

3.4. Sample Example of Incoming Converter-Assisted Multipath TCP Connection

An example of an incoming Converter-assisted Multipath TCP connection
is depicted in Figure 8. In order to support incoming connections
from remote hosts, the Client may use PCP [RFC6887] to instruct the
Converter to create dynamic mappings. Those mappings will be used by
the Converter to intercept an incoming TCP connection destined to the
Client and convert it into a Multipath TCP connection.

```
                           Transport
    Client                 Converter                      Remote Host
                                         <-------------------
                                             SYN

         <-------------------
         SYN, MPC[Remote Host:port]

         --------------------->
                  SYN+ACK, MPC
                                         --------------------->
                                                 SYN+ACK

                                         <---------------------
                                                  ACK
         <-------------------
                 ACK, MPC
```

       Figure 8: Establishment of an Incoming TCP Connection through a
                                  Converter

[4](#). The Converter Protocol (Convert)

   This section describes in details the messages that are exchanged
   between a Client and a Transport Converter. The Converter Protocol
   (Convert, for short) leverages the TCP Fast Open extension [[RFC7413](#)].

   The Converter Protocol uses a 32 bits long fixed header that is sent
   by both the Client and the Transport Converter. This header
   indicates both the version of the protocol used and the length of the
   Convert message.

[4.1](#). The Convert Fixed Header

   The Fixed Header is used to exchange information about the version
   and length of the messages between the Client and the Transport
   Converter.

   The Client and the Transport Converter MUST send the fixed-sized
   header shown in Figure 9 as the first four bytes of the bytestream.

```
                             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
        +---------------+---------------+-------------------------------+
        |    Version    | Total Length  |           Unassigned          |
        +---------------+---------------+-------------------------------+
```

        Figure 9: The fixed-sized header of the Converter protocol

   The Version is encoded as an 8 bits unsigned integer value. This
   document specifies version 1. Version 0 is reserved by this document
   and MUST NOT be used.

   The Total Length is the number of 32 bits word, including the header,
   of the bytestream that are consumed by the Converter protocol
   messages. Since Total Length is also an 8 bits unsigned integer,
   those messages cannot consume more than 1020 bytes of data. This
   limits the number of bytes that a Transport Converter needs to
   process. A Total Length of zero is invalid and the connection MUST
   be reset upon reception of such a header.

   The Unassigned field MUST be set to zero in this version of the
   protocol. These bits are available for future use [RFC8126].

4.2. Convert TLVs

4.2.1. Generic Convert TLV Format

   The Convert protocol uses variable length messages that are encoded
   using the generic TLV format depicted in Figure 10. All TLV fields
   are encoded using the network byte order.

```
                             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
        +---------------+---------------+-------------------------------+
        |     Type      |    Length     |      (optional) Value  ...    |
        +---------------+---------------+-------------------------------+
        |              ...         (optional)  Value                    |
        +---------------------------------------------------------------+
```

                  Figure 10: Converter Generic TLV Format

   A given TLV LUST only appear once on a connection. If two or more
   copies of the same TLV are exchanged over a Converter connection, the
   associated TCP connections MUST be closed. All fields are encoded
   using the network byte order. The length field is the number of 32
   bits words.

4.2.2. **Summary of Supported Convert TLVs**

   This document specifies the following Convert TLVs:

```
+------+-----+----------+-------------------------------------------+
| Type | Hex |  Length  | Description                               |
+------+-----+----------+-------------------------------------------+
|   1  | 0x1 |    1     | Bootstrap TLV                             |
|  10  | 0xA | Variable | Connect TLV                               |
|  20  | 0x14| Variable | Extended TCP Header TLV                   |
|  21  | 0x15| Variable | Supported TCP Extension Services TLV      |
|  30  | 0x1E| Variable | Error TLV                                 |
+------+-----+----------+-------------------------------------------+
```

             Figure 11: The TLVs used by the Converter protocol

   To establish a connection via a Transport Converter, a Client MUST
   first obtain a valid TFO cookie from that Converter. This is the
   bootstrap procedure during which the Client opens a connection to the
   Transport Converter with an empty TFO option. According to
   [RFC7413], the Transport Converter returns its cookie in the SYN+ACK.
   Then the Client sends a Bootstrap TLV (Section 4.2.3) to which the
   Transport Converter replies with the Supported TCP Extension Services
   TLV described in Section 4.2.4.

   With the TFO cookie of the Transport Converter, the Client can
   request the establishment of connections to remote servers with the
   Connect TLV (see Section 4.2.5). If the connection can be
   established with the final server, the Transport Converter replies
   with the Extended TCP Header TLV and returns an Error TLV inside a
   RST packet (see Section 4.2.7).

   When the Transport Converter receives an incoming connection
   establishment from a Client, it MUST process the TCP options found in
   the SYN and the Connect TLV. In general, the Transport Converter
   MUST add to the proxied SYN the TCP options that were included in the
   Connect TLV. It SHOULD add to the proxied SYN the TCP options that
   were included in the incoming SYN provided that it supports the
   corresponding TCP extension.

   There are some exceptions to these rules given the semantics of some
   TCP options. First, TCP options with Kinds 0 (EOL), 1 (NOP), 2
   (MSS), and 3 (WS) MUST be used according to the configuration of the
   TCP stack of the Transport Converter. The Timestamps option
   (Kind=10) SHOULD be used in the proxied SYN if it was present in the
   incoming SYN, but the contents of the option in the proxied SYN
   SHOULD be set by the Converter's stack. The MP_CAPABLE option SHOULD
   be added to the proxied SYN if it was present in the incoming SYN,

but the content of the option in the proxied SYN SHOULD be set by the
Converter's stack. The TCP Fast Open cookie option SHOULD be handled
as described in Section 6.

As a general rule, when an error is encountered an Error TLV with the
appropriate error code MUST be returned.

4.2.3. The Bootstrap TLV

The Bootstrap TLV (Figure 12 is sent by a Client to request the TCP
extensions that are supported by a Transport Converter and for which
it provides a conversion service. It is typically sent on the first
connection that a Client establishes with a Transport Converter to
learn its capabilities. Assuming a Client is entitled to invoke the
Converter, this latter replies with the Supported TCP Extensions
Services TLV described in Section 4.2.4.

```
                     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
+---------------+---------------+-------------------------------+
|     Type      |    Length     |             Zero              |
+---------------+---------------+-------------------------------+
```

Figure 12: The Bootstrap TLV

4.2.4. Supported TCP Extension Services TLV

The Supported TCP Extension Services TLV (Figure 13) is used by a
Converter to announce the TCP options for which it provides a
conversion service. Each supported TCP option is encoded with its
TCP option Kind listed in the "TCP Parameters" registry maintained by
IANA.

```
                     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
+---------------+---------------+-------------------------------+
|     Type      |    Length     |          Unassigned           |
+---------------+---------------+-------------------------------+
|    Kind #1    |    Kind #2     |            ...                |
+---------------+---------------+-------------------------------+
/                              ...                               /
/                                                                /
+----------------------------------------------------------------+
```

Figure 13: The Supported TCP Extension Services TLV

TCP option Kinds 0, 1, and 2 defined in [RFC0793] are supported by
all TCP implementations and thus MUST NOT appear in this list.

The list of Supported TCP Extension Services is padded with 0 to end
on a 32 bits boundary.

Typically, if the Converter only supports Multipath TCP conversion
service, solely Kind=30 will be present in the Supported TCP
Extension Services TLV returned by the Converter to a requesting
Client.

4.2.5. Connect TLV

The Connect TLV (Figure 14) is used to request the establishment of a
connection via a Transport Converter.

The 'Remote Peer Port' and 'Remote Peer IP Address' fields contain
the destination port number and IP address of the target server for
an outgoing connection towards a server located on the Internet. For
incoming connections destined to a client serviced via a Converter,
these fields convey the source port and IP address.

The Remote Peer IP Address MUST be encoded as an IPv6 address. IPv4
addresses MUST be encoded using the IPv4-Mapped IPv6 Address format
defined in [RFC4291].

The optional 'TCP Options' field is used to specify how specific TCP
Options should be advertised by the Transport Converter to the final
destination of a connection. If this field is not supplied, the
Transport Converter MUST use the default TCP options that correspond
to its local policy.

The Connect TLV could be designed to be generic to include the DNS
name of the remote peer instead of its IP address as in SOCKS
[RFC1928]. However, that design was not adopted because it induces
both an extra load and increased delays on the Converter to handle
and manage DNS resolution requests.

```
                              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
      +---------------+---------------+-------------------------------+
      |     Type      |    Length     |        Remote Peer Port       |
      +---------------+---------------+-------------------------------+
      |                                                               |
      |            Remote Peer IP Address (128 bits)                  |
      |                                                               |
      |                                                               |
      +---------------------------------------------------------------+
      |                     TCP Options (Variable)                    |
      |                              ...                              |
      +---------------------------------------------------------------+
```
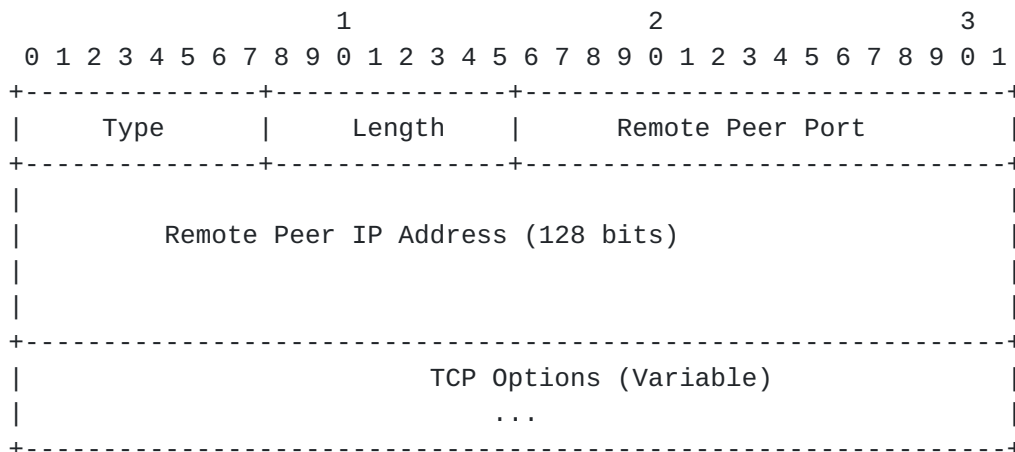
                       Figure 14: The Connect TLV

   The 'TCP Options' field is a variable length field that carries a
   list of TCP option fields (Figure 15). Each TCP option field is
   encoded as a block of 2+n bytes where the first byte is the TCP
   option Type and the second byte is the length of the TCP option as
   specified in [RFC0793]. The minimum value for the TCP option Length
   is 2. The TCP options that do not include a length subfield, i.e.,
   option types 0 (EOL) and 1 (NOP) defined in [RFC0793] cannot be
   placed inside the TCP options field of the Connect TLV. The optional
   Value field contains the variable-length part of the TCP option. A
   length of two indicates the absence of the Value field. The TCP
   options field always ends on a 32 bits boundary after being 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
      +---------------+---------------+---------------+---------------+
      |  TCPOpt type  | TCPOpt Length | Value  (opt)  |  ....         |
      +---------------+---------------+---------------+---------------+
      |                             ....                              |
      +---------------------------------------------------------------+
      |                              ...                              |
      +---------------------------------------------------------------+
```

                     Figure 15: The TCP Options field

   If a Transport Converter receives a Connect TLV with a non-empty TCP
   options field, and the Converter acceptss to process the request, it
   SHALL present those options to the destination peer in addition to
   the TCP options that it would have used according to its local
   policies. For the TCP options that are listed without an optional
   value, the Converter MUST generate its own value. For the TCP
   options that are included in the 'TCP Options' field with an optional

value, it SHALL copy the entire option for use in the connection with
the destination peer. This feature is required to support TCP Fast
Open.

The Converter may discard a Connect TLV request for many reasons
(e.g., bad TFO cookie, authorization failed, out of resources). An
error message indicating the encountered error is returned to the
requesting Client Section 4.2.7. In order to prevent denial-of-
service attacks, error messages sent to a Client SHOULD be rate-
limited.

4.2.6. Extended TCP Header TLV

The Extended TCP Header TLV (Figure 16) is used by the Transport
Converter to send to the Client the extended TCP header that was
returned by the Server in the SYN+ACK packet. This TLV is only sent
if the Client sent a Connect TLV to request the establishment of a
connection.

```
                        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
   +---------------+---------------+-------------------------------+
   |     Type      |    Length     |           Unassigned          |
   +---------------+---------------+-------------------------------+
   |                 Returned Extended TCP header                  |
   |                            ...                                |
   +---------------------------------------------------------------+
```

Figure 16: The Extended TCP Header TLV

The Returned Extended TCP header field is a copy of the extended
header that was received in the SYN+ACK by the Transport Converter.

The Unassigned field MUST be set to zero by the transmitter and
ignored by the receiver. These bits are available for future use
[RFC8126].

4.2.7. Error TLV

The optional Error TLV (Figure 17) can be used by the Transport
Converter to provide information about some errors that occurred
during the processing of a request to convert a connection. This TLV
appears after the Convert header in a RST segment returned by the
Transport Converter if the error is fatal and prevented the
establishment of the connection. If the error is not fatal and the
connection could be established with the final destination, then the
error TLV will be carried in the payload.

```
                      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
 +---------------+---------------+---------------+--------------+
 |     Type      |     Length    |     Error     |    Value     |
 +---------------+---------------+---------------+--------------+
```

                     Figure 17: The Error TLV
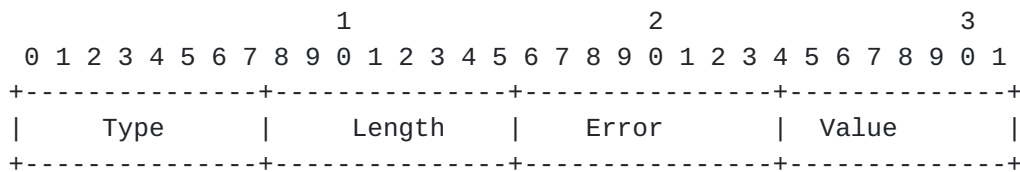
   Different types of errors can occur while processing Convert
   messages. Each error is identified by a code represented as an
   unsigned integer. Four classes of errors are defined:

   o Message validation and processing errors (0-31 range): returned
      upon reception of an an invalid message (including valid messages
      but with invalid or unknown TLVs).

   o Client-side errors (32-63 range): the Client sent a request that
      could not be accepted by the Converter (e.g., unsupported
      operation).

   o Converter-side errors (64-95 range) : problems encountered on the
      Converter (e.g., lack of resources) which prevent it from
      fulfilling the Client's request.

   o Errors caused by destination server (96-127 range) : the final
      destination could not be reached or it replied with a reset
      message.

   The following error codes are defined in this document:

   o Unsupported Version (0): The version number indicated in the fixed
      header of a message received from a peer is not supported.

      This error code MUST be generated by a Converter when it receives
      a request having a version number that it does not support.

      The value field MUST be set to the version supported by the
      Converter. When multiple versions are supported by the Converter,
      it includes the list of supported version in the value field; each
      version is encoded in 8 bits.

      Upon receipt of this error code, the client checks whether it
      supports one of the versions returned by the Converter. The
      highest common supported version MUST be used by the client in
      subsequent exchanges with the Converter.

   o Malformed Message (1): This error code is sent to indicate that a
      message can not be successfully parsed.

To ease troubleshooting, the value field MUST echo the received
message. The Converter and the Client MUST send a RST containing
this error upon reception of a malformed message.

o Unsupported Message (2): This error code is sent to indicate that
   a message type is not supported by the Converter.

   To ease troubleshooting, the value field MUST echo the received
   message. The Converter and the Client MUST send a RST containing
   this error upon reception of an unsupported message.

o Not Authorized (32): This error code indicates that the Converter
   refused to create a connection because of a lack of authorization
   (e.g., administratively prohibited, authorization failure, etc.).
   The Value field MUST be set to zero.

   This error code MUST be sent by the Converter when a request
   cannot be successfully processed because the authorization failed.

o Unsupported TCP Option (33): A TCP option that the Client
   requested to advertise to the final Server cannot be safely used
   jointly with the conversion service.

   The Value field is set to the type of the unsupported TCP option.
   If several unsupported TCP options were specified in the Connect
   TLV, only one of them is returned in the Value.

o Resource Exceeded (64): This error indicates that the Transport
   Converter does not have enough resources to perform the request.

   This error MUST be sent by the Converter when it does not have
   sufficient resources to handle a new connection.

o Network Failure (65): This error indicates that the Converter is
   experiencing a network failure to relay the request.

   The Converter MUST send this error code when it experiences
   forwarding issues to relay a connection.

o Connection Reset (96): This error indicates that the final
   destination responded with a RST packet. The Value field MUST be
   set to zero.

o Destination Unreachable (97): This error indicates that an ICMP
   destination unreachable, port unreachable, or network unreachable
   was received by the Converter. The Value field MUST echo the Code
   field of the received ICMP message.

This error message MUST be sent by the Converter when it receives an error message that is bound to a message it relayed previously.

Figure 18 summarizes the different error codes.

```
+-------+------+-----------------------------------------------+
| Error | Hex  | Description                                   |
+-------+------+-----------------------------------------------+
|     0 | 0x00 | Unsupported Version                           |
|     1 | 0x01 | Malformed Message                             |
|     2 | 0x02 | Unsupported Message                           |
|    32 | 0x20 | Not Authorized                                |
|    33 | 0x21 | Unsupported TCP Option                        |
|    64 | 0x40 | Resource Exceeded                             |
|    65 | 0x41 | Network Failure                               |
|    96 | 0x60 | Connection Reset                              |
|    97 | 0x61 | Destination Unreachable                       |
+-------+------+-----------------------------------------------+
```

Figure 18: Convert Error Values

5. Compatibility of Specific TCP Options with the Conversion Service

In this section, we discuss how several standard track TCP options can be supported through the Converter. The non-standard track options and the experimental options will be discussed in other documents.

5.1. Base TCP Options

Three TCP options were initially defined in [RFC0793] : End-of-Option List (Kind=0), No-Operation (Kind=1) and Maximum Segment Size (Kind=2). The first two options are mainly used to pad the TCP extended header. There is no reason for a client to request a Converter to specifically send these options towards the final destination.

The Maximum Segment Size option (Kind=2) is used by a host to indicate the largest segment that it can receive over each connection. This value is function of the stack that terminates the TCP connection. There is no reason for a Client to request a Converter to advertise a specific MSS value to a remote server.

A Converter MUST ignore options with Kind=0, 1 or 2 if they appear in a Connect TLV. It MUST NOT announce them in a Bootstrap TLV.

5.2. Window Scale (WS)

The Window Scale option (Kind=3) is defined in [RFC7323]. As for the
MSS option, the window scale factor that is used for a connection
strongly depends on the TCP stack that handles the connection. When
a Converter opens a TCP connection towards a remote server on behalf
of a Client, it SHOULD use a WS option with a scaling factor that
corresponds to the configuration of its stack. A local configuration
MAY allow for WS option in the proxied message to be function of the
scaling factor of the incoming connection.

There is no benefit from a deployment viewpoint in enabling a Client
of a Converter to specifically request the utilisation of the WS
option (Kind=3) with a specific scaling factor towards a remote
Server. For this reason, a Converter MUST ignore option Kind=3 if it
appears in a Connect TLV. It MUST NOT announce it in a Bootstrap
TLV.

5.3. Selective Acknowledgements

Two distinct TCP options were defined to support selective
acknowledgements in [RFC2018]. This first one, SACK Permitted
(Kind=4), is used to negotiate the utilisation of selective
acknowledgements during the three-way handshake. The second one,
SACK (Kind=5), carries the selective acknowledgements inside regular
segments.

The SACK Permitted option (Kind=4) MAY be advertised by a Transport
Converter in the Bootstrap TLV. In this case, Clients connected to
this Transport Converter MAY include the SACK Permitted option in the
Connect TLV.

The SACK option (Kind=5) cannot be used during the three-way
handshake. For this reason, a Transport Converter MUST ignore option
Kind=5 with if it appears in a Connect TLV. It MUST NOT announce it
in a Bootstrap TLV.

5.4. Timestamp

The Timestamp option was initially defined in [RFC1323] which has
been replaced by [RFC7323]. It can be used during the three-way
handshake to negotiate the utilisation of the timestamps during the
TCP connection. It is notably used to improve round-trip-time
estimations and to provide protection against wrapped sequence
numbers (PAWS). As for the WS option, the timestamps are a property
of a connection and there is limited benefit in enabling a client to
request a Converter to use the timestamp option when establishing a
connection to a remote server. Furthermore, the timestamps that are

used by TCP stacks are specific to each stack and there is no benefit
in enabling a client to specify the timestamp value that a Converter
could use to establish a connection to a remote server.

A Transport Converter MAY advertise the Timestamp option (Kind=8) in
the Bootstrap TLV. The clients connected to this Converter MAY
include the Timestamp option in the Connect TLV but without any
timestamp.

5.5. Multipath TCP

The Multipath TCP options are defined in [RFC6824]. [RFC6824]
defines one variable length TCP option (Kind=30) that includes a
subtype field to support several Multipath TCP options. There are
several operational use cases where clients would like to use
Multipath TCP through a Converter [IETFJ16]. However, none of these
use cases require the Client to specify the content of the Multipath
TCP option that the Converter should send to a remote server.

A Transport Converter which supports Multipath TCP conversion service
MUST advertise the Multipath TCP option (Kind=30) in the Bootstrap
TLV. Clients serviced by this Converter may include the Multipath
TCP option in the Connect TLV but without any content.

5.6. TCP Fast Open

The TCP Fast Open cookie option (Kind=34) is defined in [RFC7413].
There are two different usages of this option that need to be
supported by Transport Converters. The first utilisation of the Fast
Open cookie is to request a cookie from the server. In this case,
the option is sent with an empty cookie by the client and the server
returns the cookie. The second utilisation of the Fast Open cookie
is to send a cookie to the server. In this case, the option contains
a cookie.

A Transport Converter MAY advertise the TCP Fast Open cookie option
(Kind=34) in the Bootstrap TLV. If a Transport Converter has
advertised the support for TCP Fast Open in its Bootstrap TLV, it
needs to be able to process two types of Connect TLV. If such a
Transport Converter receives a Connect TLV with the TCP Fast Open
cookie option that does not contain a cookie, it MUST add an empty
TCP Fast Open cookie option in the SYN sent to the remote server. If
such a Transport Converter receives a Connect TLV with the TCP Fast
Open cookie option that contains a cookie, it MUST copy the TCP Fast
Open cookie option in the SYN sent to the remote server.

5.7. TCP User Timeout

   The TCP User Timeout option is defined in [RFC5482]. The associated
   TCP option (Kind=28) does not appear to be widely deployed.

   Editor's Note: Feedback requested for the utilisation of this option
   by deployed TCP stacks.

5.8. TCP-AO

   TCP-AO [RFC5925] provides a technique to authenticate all the packets
   exchanged over a TCP connection. Given the nature of this extension,
   it is unlikely that the applications that require their packets to be
   authenticated end-to-end would want their connections to pass through
   a converter. For this reason, we do not recommend the support of the
   TCP-AO option by Transport Converters. The only use cases where is
   makes sense to combine TCP-AO and the solution in this document are
   those where the TCP-AO-NAT extension [RFC6978] is in use.

   A Converter MUST NOT advertise the TCP-AO option (Kind=29) in the
   Bootstrap TLV. If a Converter receives a Connect TLV that contains
   the TCP-AO option, it MUST reject the establishment of the connection
   with error code set to "Unsupported TCP Option", except if the TCP-
   AO-NAT option is used.

5.9. TCP Experimental Options

   The TCP Experimental options are defined in [RFC4727]. Given the
   variety of semantics for these options and their experimental nature,
   it is impossible to discuss them in details in this document.

6. Interactions with Middleboxes

   The Converter Protocol was designed to be used in networks that do
   not contain middleboxes that interfere with TCP. We describe in this
   section how a Client can detect middlebox interference and stop using
   the Transport Converter affected by this interference.

   Internet measurements [IMC11] have shown that middleboxes can affect
   the deployment of TCP extensions. In this section, we only discuss
   the middleboxes that modify SYN and SYN+ACK packets since the
   Converter Protocol places its messages in such packets.

   Let us first consider a middlebox that removes the TFO Option from
   the SYN packet. This interference will be detected by the Client
   during the bootstrap procedure discussed in Section 4.2.3. A Client
   should not use a Transport Converter that does not reply with the TFO
   option during the Bootstrap.

Consider a middlebox that removes the SYN payload after the bootstrap
procedure. The Client can detect this problem by looking at the
acknowledgement number field of the SYN+ACK returned by the Transport
Converter. The Client should stop to use this Transport Converter
given the middlebox interference.

As explained in [RFC7413], some carrier-grade NATs can affect the
operation of TFO if they assign different IP addresses to the same
end host. Such carrier-grade NATs could affect the operation of the
TFO Option used by the Converter Protocol. See also the discussion
in Section 7.1 of [RFC7413].

7. Security Considerations

7.1. Privacy & Ingress Filtering

The Converter may have access to privacy-related information (e.g.,
subscriber credentials). The Converter MUST NOT leak such sensitive
information outside a local domain.

Given its function and its location in the network, a Transport
Converter has access to the payload of all the packets that it
processes. As such, it MUST be protected as a core IP router (e.g.,
[RFC1812]).

Furthermore, ingress filtering policies MUST be enforced at the
network boundaries [RFC2827].

This document assumes that all network attachments are managed by the
same administrative entity. Therefore, enforcing anti-spoofing
filters at these network ensures that hosts are not sending traffic
with spoofed source IP addresses.

7.2. Authorization

The Converter Protocol is intended to be used in managed networks
where end hosts can be identified by their IP address. Thanks to the
Bootstrap procedure, the Transport Converter can verify that the
Client correctly receives packets sent by the Converter. Stronger
authentication schemes MUST be defined to use the Converter Protocol
in more open network environments; such schemes are out of scope of
this document.

See below for authorization considerations that are specific for
Multipath TCP.

7.3. Denial of Service

   Another possible risk is the amplification attacks since a Transport
   Converter sends a SYN towards a remote Server upon reception of a SYN
   from a Client. This could lead to amplification attacks if the SYN
   sent by the Transport Converter were larger than the SYN received
   from the Client or if the Transport Converter retransmits the SYN.
   To mitigate such attacks, the Transport Converter SHOULD rate limit
   the number of pending requests for a given Client. It SHOULD also
   avoid sending to remote Servers SYNs that are significantly longer
   than the SYN received from the Client. In practice, Transport
   Converters SHOULD NOT advertise to a Server TCP options that were not
   specified by the Client in the received SYN. Finally, the Transport
   Converter SHOULD only retransmit a SYN to a Server after having
   received a retransmitted SYN from the corresponding Client.

   Upon reception of a SYN that contains a valid TFO cookie and a
   Connect TLV, the Transport Converter attempts to establish a TCP
   connection to a remote Server. There is a risk of denial of service
   attack if a Client requests too many connections in a short period of
   time. Implementations SHOULD limit the number of pending connections
   from a given Client. Means to protect against SYN flooding attacks
   MUST also be enabled [RFC4987].

7.4. Traffic Theft

   Traffic theft is a risk if an illegitimate Converter is inserted in
   the path. Indeed, inserting an illegitimate Converter in the
   forwarding path allows traffic interception and can therefore provide
   access to sensitive data issued by or destined to a host. Converter
   discovery and configuration are out of scope of this document.

7.5. Multipath TCP-specific Considerations

   Multipath TCP-related security threats are discussed in [RFC6181] and
   [RFC6824].

   The operator that manages the various network attachments (including
   the Converters) can enforce authentication and authorization policies
   using appropriate mechanisms. For example, a non-exhaustive list of
   methods to achieve authorization is provided hereafter:

   o The network provider may enforce a policy based on the
      International Mobile Subscriber Identity (IMSI) to verify that a
      user is allowed to benefit from the aggregation service. If that
      authorization fails, the Packet Data Protocol (PDP) context/bearer
      will not be mounted. This method does not require any interaction
      with the Converter.

o The network provider may enforce a policy based upon Access
   Control Lists (ACLs), e.g., at a Broadband Network Gateway (BNG)
   to control the hosts that are authorized to communicate with a
   Converter. These ACLs may be installed as a result of RADIUS
   exchanges, e.g. [I-D.boucadair-radext-tcpm-converter]. This
   method does not require any interaction with the Converter.

o A device that embeds the Converter may also host a RADIUS client
   that will solicit an AAA server to check whether connections
   received from a given source IP address are authorized or not
   [I-D.boucadair-radext-tcpm-converter].

A first safeguard against the misuse of Converter resources by
illegitimate users (e.g., users with access networks that are not
managed by the same provider that operates the Converter) is the
Converter to reject Multipath TCP connections received on its
Internet-facing interfaces. Only Multipath PTCP connections received
on the customer-facing interfaces of a Converter will be accepted.

8. IANA Considerations

8.1. Convert Service Port Number

IANA is requested to assign a TCP port number (TBA) for the Converter
Protocol from the "Service Name and Transport Protocol Port Number
Registry" available at https://www.iana.org/assignments/service-
names-port-numbers/service-names-port-numbers.xhtml.

8.2. The Converter Protocol (Convert) Parameters

IANA is requested to create a new "The Converter Protocol (Convert)
Parameters" registry.

The following subsections detail new registries within "The Converter
Protocol (Convert) Parameters" registry.

8.2.1. Convert Versions

IANA is requested to create the "Convert versions" sub-registry. New
values are assigned via Standards Action.

The initial values to be assigned at the creation of the registry are
as follows:

```
+---------+-------------------------------------+------------+
| Version | Description                         | Reference  |
+---------+-------------------------------------+------------+
|    0    | Reserved by this document           | [This-RFC] |
|    1    | Assigned by this document           | [This-RFC] |
+---------+-------------------------------------+------------+
```

8.2.2. Convert TLVs

IANA is requested to create the "Convert TLVs" sub-registry. The
procedure for assigning values from this registry is as follows:

o The values in the range 1-127 can be assigned via Standards
   Action.

o The values in the range 128-191 can be assigned via Specification
   Required.

o The values in the range 192-255 can be assigned for Private Use.

The initial values to be assigned at the creation of the registry are
as follows:

```
+---------+-------------------------------------+------------+
|  Code   | Name                                | Reference  |
+---------+-------------------------------------+------------+
|    0    | Reserved                            | [This-RFC] |
|    1    | Bootstrap TLV                       | [This-RFC] |
|   10    | Connect TLV                         | [This-RFC] |
|   20    | Extended TCP Header TLV             | [This-RFC] |
|   22    | Supported TCP Extension Services TLV | [This-RFC] |
|   30    | Error TLV                           | [This-RFC] |
+---------+-------------------------------------+------------+
```

8.2.3. Convert Error Messages

IANA is requested to create the "Convert Errors" sub-registry. Codes
in this registry are assigned as a function of the error type. Four
types are defined; the following ranges are reserved for each of
these types:

o Message validation and processing errors: 0-31

o Client-side errors: 32-63

o Converter-side errors: 64-95

o Errors caused by destination server: 96-127

The procedure for assigning values from this sub-registry is as
follows:

o 0-191: Values in this range are assigned via Standards Action.

o 192-255: Values in this range are assigned via Specification
   Required.

The initial values to be assigned at the creation of the registry are
as follows:

```
+-------+------+---------------------------------+----------+
| Error | Hex  | Description                     | Reference |
+-------+------+---------------------------------+----------+
|     0 | 0x00 | Unsupported Version             | [This-RFC]|
|     1 | 0x01 | Malformed Message               | [This-RFC]|
|     2 | 0x02 | Unsupported Message             | [This-RFC]|
|    32 | 0x20 | Not Authorized                  | [This-RFC]|
|    33 | 0x21 | Unsupported TCP Option          | [This-RFC]|
|    64 | 0x40 | Resource Exceeded               | [This-RFC]|
|    65 | 0x41 | Network Failure                 | [This-RFC]|
|    96 | 0x60 | Connection Reset                | [This-RFC]|
|    97 | 0x61 | Destination Unreachable         | [This-RFC]|
+-------+------+---------------------------------+----------+
```

Figure 19: The Convert Error Codes

9. Acknowledgements

Although they could disagree with the contents of the document, we
would like to thank Joe Touch and Juliusz Chroboczek whose comments
on the MPTCP mailing list have forced us to reconsider the design of
the solution several times.

We would like to thank Raphael Bauduin, Stefano Secci, Benjamin
Hesmans and Anandatirtha Nandugudi for their help in preparing this
document. Nandini Ganesh provided valuable feedback about the
handling of TFO and the error codes. Thanks to them.

This document builds upon earlier documents that proposed various
forms of Multipath TCP proxies [I-D.boucadair-mptcp-plain-mode],
[I-D.peirens-mptcp-transparent] and [HotMiddlebox13b].

From [I-D.boucadair-mptcp-plain-mode]:

Many thanks to Chi Dung Phung, Mingui Zhang, Rao Shoaib, Yoshifumi
Nishida, and Christoph Paasch for their valuable comments.

Thanks to Ian Farrer, Mikael Abrahamsson, Alan Ford, Dan Wing, and
Sri Gundavelli for the fruitful discussions in IETF#95 (Buenos
Aires).

Special thanks to Pierrick Seite, Yannick Le Goff, Fred Klamm, and
Xavier Grall for their inputs.

Thanks also to Olaf Schleusing, Martin Gysi, Thomas Zasowski, Andreas
Burkhard, Silka Simmen, Sandro Berger, Michael Melloul, Jean-Yves
Flahaut, Adrien Desportes, Gregory Detal, Benjamin David, Arun
Srinivasan, and Raghavendra Mallya for the discussion.

9.1. Contributors

Bart Peirens contributed to an early version of the document.

As noted above, this document builds on two previous documents.

The authors of [I-D.boucadair-mptcp-plain-mode] were:

o Mohamed Boucadair

o Christian Jacquenet

o Olivier Bonaventure

o Denis Behaghel

o Stefano Secci

o Wim Henderickx

o Robert Skog

o Suresh Vinapamula

o SungHoon Seo

o Wouter Cloetens

o Ullrich Meyer

o Luis M. Contreras

o Bart Peirens

The authors of [I-D.peirens-mptcp-transparent] were:

o Bart Peirens

o Gregory Detal

o Sebastien Barre

o Olivier Bonaventure

10. Change Log

This section to be removed before publication.

o 00 : initial version, designed to support Multipath TCP and TFO
   only

o 00 to -01 : added section Section 5 describing the support of
   different standard tracks TCP options by Transport Converters,
   clarification of the IANA section, moved the SOCKS comparison to
   the appendix and various minor modifications

o 01 to -02 : Minor modifications

o 02 to -03 : Minor modifications

11. References

11.1. Normative References

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7,
           RFC 793, DOI 10.17487/RFC0793, September 1981,
           <https://www.rfc-editor.org/info/rfc793>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
           Requirement Levels", BCP 14, RFC 2119,
           DOI 10.17487/RFC2119, March 1997, <https://www.rfc-
           editor.org/info/rfc2119>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing
           Architecture", RFC 4291, DOI 10.17487/RFC4291, February
           2006, <https://www.rfc-editor.org/info/rfc4291>.

[RFC4727] Fenner, B., "Experimental Values In IPv4, IPv6, ICMPv4,
           ICMPv6, UDP, and TCP Headers", RFC 4727,
           DOI 10.17487/RFC4727, November 2006, <https://www.rfc-
           editor.org/info/rfc4727>.

   [RFC4987] Eddy, W., "TCP SYN Flooding Attacks and Common
              Mitigations", RFC 4987, DOI 10.17487/RFC4987, August 2007,
              <https://www.rfc-editor.org/info/rfc4987>.

   [RFC5482] Eggert, L. and F. Gont, "TCP User Timeout Option",
              RFC 5482, DOI 10.17487/RFC5482, March 2009,
              <https://www.rfc-editor.org/info/rfc5482>.

   [RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP
              Authentication Option", RFC 5925, DOI 10.17487/RFC5925,
              June 2010, <https://www.rfc-editor.org/info/rfc5925>.

   [RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure,
              "TCP Extensions for Multipath Operation with Multiple
              Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013,
              <https://www.rfc-editor.org/info/rfc6824>.

   [RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP
              Fast Open", RFC 7413, DOI 10.17487/RFC7413, December 2014,
              <https://www.rfc-editor.org/info/rfc7413>.

   [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for
              Writing an IANA Considerations Section in RFCs", BCP 26,
              RFC 8126, DOI 10.17487/RFC8126, June 2017,
              <https://www.rfc-editor.org/info/rfc8126>.

   [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC
              2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,
              May 2017, <https://www.rfc-editor.org/info/rfc8174>.

11.2. Informative References

   [ANRW17]   Trammell, B., Kuhlewind, M., De Vaere, P., Learmonth, I.,
              and G. Fairhurst, "Tracking transport-layer evolution with
              PATHspider", Applied Networking Research Workshop 2017
              (ANRW17) , July 2017.

   [Fukuda2011]
              Fukuda, K., "An Analysis of Longitudinal TCP Passive
              Measurements (Short Paper)", Traffic Monitoring and
              Analysis. TMA 2011. Lecture Notes in Computer Science, vol
              6613. , 2011.

   [HotMiddlebox13b]
              Detal, G., Paasch, C., and O. Bonaventure, "Multipath in
              the Middle(Box)", HotMiddlebox'13 , December 2013,
              <http://inl.info.ucl.ac.be/publications/
              multipath-middlebox>.

[I-D.arkko-arch-low-latency]
          Arkko, J. and J. Tantsura, "Low Latency Applications and
          the Internet Architecture", draft-arkko-arch-low-
          latency-02 (work in progress), October 2017.

[I-D.boucadair-mptcp-plain-mode]
          Boucadair, M., Jacquenet, C., Bonaventure, O., Behaghel,
          D., stefano.secci@lip6.fr, s., Henderickx, W., Skog, R.,
          Vinapamula, S., Seo, S., Cloetens, W., Meyer, U.,
          Contreras, L., and B. Peirens, "Extensions for Network-
          Assisted MPTCP Deployment Models", draft-boucadair-mptcp-
          plain-mode-10 (work in progress), March 2017.

[I-D.boucadair-radext-tcpm-converter]
          Boucadair, M. and C. Jacquenet, "RADIUS Extensions for
          0-RTT TCP Converters", draft-boucadair-radext-tcpm-
          converter-00 (work in progress), April 2018.

[I-D.boucadair-tcpm-dhc-converter]
          Boucadair, M., Jacquenet, C., and R. K, "DHCP Options for
          0-RTT TCP Converters", draft-boucadair-tcpm-dhc-
          converter-00 (work in progress), April 2018.

[I-D.ietf-mptcp-rfc6824bis]
          Ford, A., Raiciu, C., Handley, M., Bonaventure, O., and C.
          Paasch, "TCP Extensions for Multipath Operation with
          Multiple Addresses", draft-ietf-mptcp-rfc6824bis-12 (work
          in progress), October 2018.

[I-D.ietf-tcpinc-tcpcrypt]
          Bittau, A., Giffin, D., Handley, M., Mazieres, D., Slack,
          Q., and E. Smith, "Cryptographic protection of TCP Streams
          (tcpcrypt)", draft-ietf-tcpinc-tcpcrypt-13 (work in
          progress), September 2018.

[I-D.olteanu-intarea-socks-6]
          Olteanu, V. and D. Niculescu, "SOCKS Protocol Version 6",
          draft-olteanu-intarea-socks-6-04 (work in progress),
          August 2018.

[I-D.peirens-mptcp-transparent]
          Peirens, B., Detal, G., Barre, S., and O. Bonaventure,
          "Link bonding with transparent Multipath TCP", draft-
          peirens-mptcp-transparent-00 (work in progress), July
          2016.

[IETFJ16] Bonaventure, O. and S. Seo, "Multipath TCP Deployment",
          IETF Journal, Fall 2016 , n.d..

   [IMC11]   Honda, K., Nishida, Y., Raiciu, C., Greenhalgh, A.,
              Handley, M., and T. Hideyuki, "Is it still possible to
              extend TCP ?", Proceedings of the 2011 ACM SIGCOMM
              conference on Internet measurement conference , 2011.

   [RFC1323] Jacobson, V., Braden, R., and D. Borman, "TCP Extensions
              for High Performance", RFC 1323, DOI 10.17487/RFC1323, May
              1992, <https://www.rfc-editor.org/info/rfc1323>.

   [RFC1812] Baker, F., Ed., "Requirements for IP Version 4 Routers",
              RFC 1812, DOI 10.17487/RFC1812, June 1995,
              <https://www.rfc-editor.org/info/rfc1812>.

   [RFC1919] Chatel, M., "Classical versus Transparent IP Proxies",
              RFC 1919, DOI 10.17487/RFC1919, March 1996,
              <https://www.rfc-editor.org/info/rfc1919>.

   [RFC1928] Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., and
              L. Jones, "SOCKS Protocol Version 5", RFC 1928,
              DOI 10.17487/RFC1928, March 1996, <https://www.rfc-
              editor.org/info/rfc1928>.

   [RFC2018] Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, "TCP
              Selective Acknowledgment Options", RFC 2018,
              DOI 10.17487/RFC2018, October 1996, <https://www.rfc-
              editor.org/info/rfc2018>.

   [RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering:
              Defeating Denial of Service Attacks which employ IP Source
              Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827,
              May 2000, <https://www.rfc-editor.org/info/rfc2827>.

   [RFC3135] Border, J., Kojo, M., Griner, J., Montenegro, G., and Z.
              Shelby, "Performance Enhancing Proxies Intended to
              Mitigate Link-Related Degradations", RFC 3135,
              DOI 10.17487/RFC3135, June 2001, <https://www.rfc-
              editor.org/info/rfc3135>.

   [RFC6181] Bagnulo, M., "Threat Analysis for TCP Extensions for
              Multipath Operation with Multiple Addresses", RFC 6181,
              DOI 10.17487/RFC6181, March 2011, <https://www.rfc-
              editor.org/info/rfc6181>.

   [RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with
              Dual-Stack Hosts", RFC 6555, DOI 10.17487/RFC6555, April
              2012, <https://www.rfc-editor.org/info/rfc6555>.

   [RFC6887] Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and
              P. Selkirk, "Port Control Protocol (PCP)", RFC 6887,
              DOI 10.17487/RFC6887, April 2013, <https://www.rfc-
              editor.org/info/rfc6887>.

   [RFC6928] Chu, J., Dukkipati, N., Cheng, Y., and M. Mathis,
              "Increasing TCP's Initial Window", RFC 6928,
              DOI 10.17487/RFC6928, April 2013, <https://www.rfc-
              editor.org/info/rfc6928>.

   [RFC6978] Touch, J., "A TCP Authentication Option Extension for NAT
              Traversal", RFC 6978, DOI 10.17487/RFC6978, July 2013,
              <https://www.rfc-editor.org/info/rfc6978>.

   [RFC7323] Borman, D., Braden, B., Jacobson, V., and R.
              Scheffenegger, Ed., "TCP Extensions for High Performance",
              RFC 7323, DOI 10.17487/RFC7323, September 2014,
              <https://www.rfc-editor.org/info/rfc7323>.

   [RFC7414] Duke, M., Braden, R., Eddy, W., Blanton, E., and A.
              Zimmermann, "A Roadmap for Transmission Control Protocol
              (TCP) Specification Documents", RFC 7414,
              DOI 10.17487/RFC7414, February 2015, <https://www.rfc-
              editor.org/info/rfc7414>.

Appendix A. Differences with SOCKSv5

   The description above is a simplified description of the Converter
   protocol. At a first glance, the proposed solution could seem
   similar to the SOCKS v5 protocol [RFC1928]. This protocol is used to
   proxy TCP connections. The Client creates a connection to a SOCKS
   proxy, exchanges authentication information and indicates the
   destination address and port of the final server. At this point, the
   SOCKS proxy creates a connection towards the final server and relays
   all data between the two proxied connections. The operation of an
   implementation based on SOCKSv5 is illustrated in Figure 20.

```
   Client                      SOCKS Proxy                      Server
        ------------------->
             SYN
        <-------------------
             SYN+ACK
        ------------------->
             ACK

        ------------------->
        Version=5, Auth Methods
        <-------------------
             Method
        ------------------->
           Auth Request (unless "No auth" method negotiated)
        <-------------------
           Auth Response
        ------------------->
        Connect Server:Port        ------------------->
                                            SYN

                                   <-------------------
                                            SYN+ACK
        <-------------------
             Succeeded

        ------------------->
             Data1
                                   ------------------->
                                            Data1

                                   <-------------------
                                            Data2
        <-------------------
              Data2
```
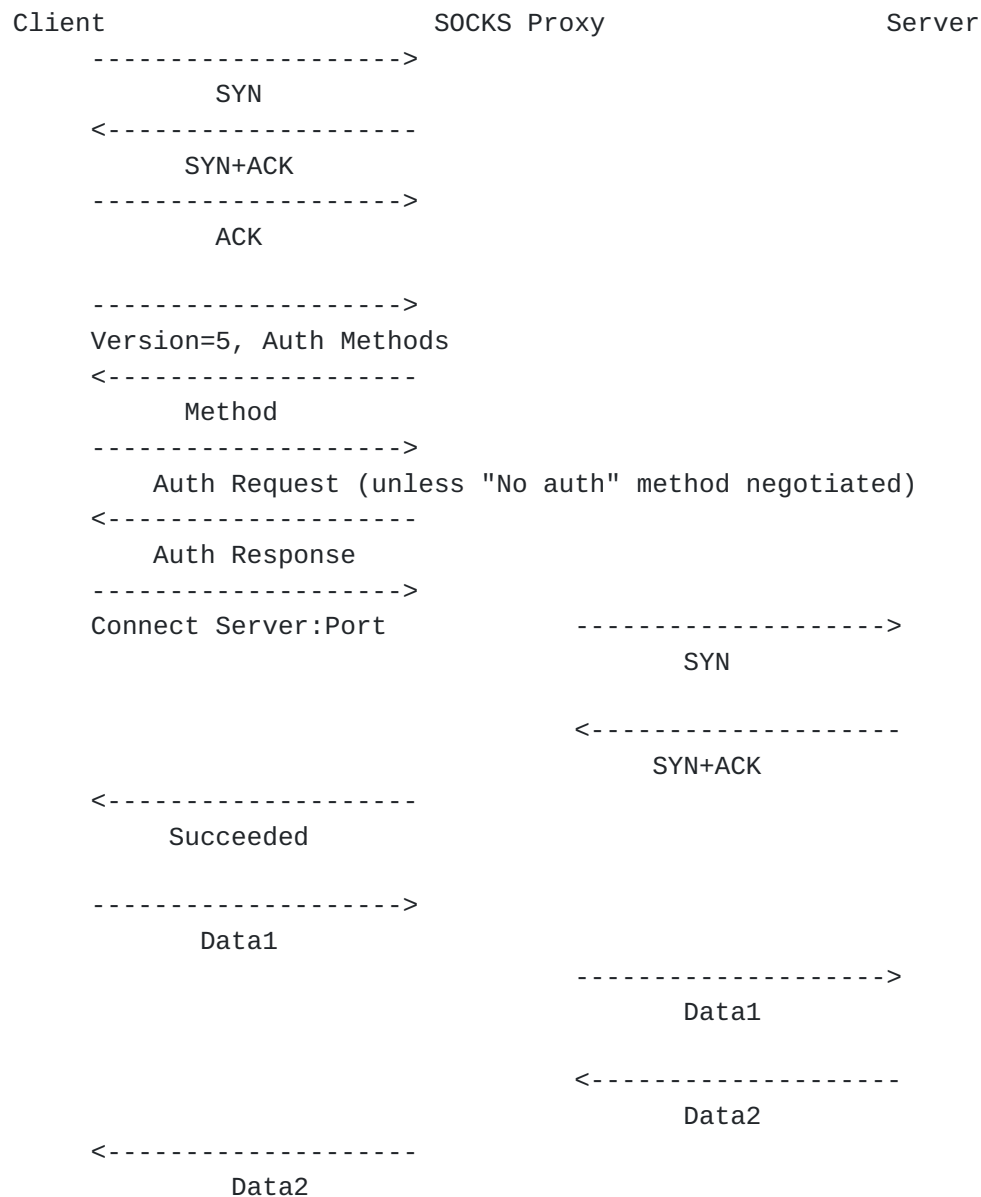
        Figure 20: Establishment of a TCP connection through a SOCKS proxy
                           without authentication

   The Converter protocol also relays data between an upstream and a
   downstream connection, but there are important differences with
   SOCKSv5.

   A first difference is that the Converter protocol leverages the TFO
   option [RFC7413] to exchange all control information during the
   three-way handshake. This reduces the connection establishment delay
   compared to SOCKS that requires two or more round-trip-times before
   the establishment of the downstream connection towards the final
   destination. In today's Internet, latency is a important metric and

various protocols have been tuned to reduce their latency
[I-D.arkko-arch-low-latency]. A recently proposed extension to SOCKS
also leverages the TFO option [I-D.olteanu-intarea-socks-6].

A second difference is that the Converter protocol explicitly takes
the TCP extensions into account. By using the Converter protocol,
the Client can learn whether a given TCP extension is supported by
the destination Server. This enables the Client to bypass the
Transport Converter when the destination supports the required TCP
extension. Neither SOCKS v5 [RFC1928] nor the proposed SOCKS v6
[I-D.olteanu-intarea-socks-6] provide such a feature.

A third difference is that a Transport Converter will only accept the
connection initiated by the Client provided that the downstream
connection is accepted by the Server. If the Server refuses the
connection establishment attempt from the Transport Converter, then
the upstream connection from the Client is rejected as well. This
feature is important for applications that check the availability of
a Server or use the time to connect as a hint on the selection of a
Server [RFC6555].

Authors' Addresses

   Olivier Bonaventure (editor)
   Tessares

   Email: Olivier.Bonaventure@tessares.net

   Mohamed Boucadair (editor)
   Orange

   Email: mohamed.boucadair@orange.com

   Sri Gundavelli
   Cisco

   Email: sgundave@cisco.com

   SungHoon Seo
   Korea Telecom

   Email: sh.seo@kt.com