TCPM Working Group
Internet-Draft
Intended status: Experimental
Expires: August 20, 2018

O. Bonaventure
Tessares
M. Boucadair
Orange
B. Peirens
Proximus
S. Seo
Korea Telecom
A. Nandugudi
Tessares

February 16, 2018

0-RTT TCP Converter
draft-ietf-tcpm-converters-00

Abstract

This document specifies an application proxy, called Transport Converter, to assist the deployment of 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.

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 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. Fukuda reports in [Fukuda2011] results of a decade of measurements showing the deployment of Selective Acknowledgements, Window Scale and TCP Timestamps. Trammel et al. provide in [ANRW17] 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] 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.

This document specifies an application proxy, called Transport Converter (TC). 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 supports one or more TCP extensions. The Converter Protocol (CP) is an application layer protocol that uses a TCP port number (see IANA section). The Transport Converter adheres to the main principles as drawn in [RFC1919]. In particular, the Converter achieves the following:

- Listen for client sessions;

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

- Setup a session to the final server;
- Relay control messages and data between the client and the server;
- 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 some options between the endpoints. This document focuses on Multipath TCP [RFC6824] and TCP Fast Open [RFC7413]. The support for other TCP extensions will be discussed in other documents.

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 transport converters. 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 compare them in Section 3.3 with SOCKS proxies that are already used to deploy Multipath TCP in cellular networks [IETFJ16]. We then describe the Converter Protocol in Section 4. We then discuss the interactions with middleboxes (Section 5) and the security considerations (Section 6).

2. Applicability Scope

This specification is designed with Multipath TCP [RFC6824][I-D.ietf-mptcp-rfc6824bis] and TCP Fast Open [RFC7413] in mind. That is, the specification draws how network-assisted Multipath TCP connections can be established even if the remote server is not Multipath TCP-capable without inducing extra connection delays (0-RTT proxy). Further, the specification allows the client for end-to-end Multipath TCP connections with or without proxy involvement. Assessing the applicability of the solution to other use cases and other TCP extensions such as [I-D.ietf-tcpinc-tcpcrypt] is outside the scope of this document. Future documents are required to specify the exact behavior when the converter is deployed in other contexts than Multipath TCP.

3. Architecture

The architecture considers three types of endhosts:

- Client endhosts;
- Transport Converters;
- Server endhosts.

It does not mandate anything on the server side. 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.

A Transport Converter is a network function that relays all data exchanged over one upstream connection to one downstream connection and vice versa. A connection can be initiated from both interfaces of the transport converter (Internet-facing interface, client-facing interface). The converter, thus, maintains state that associates one upstream connection to a corresponding downstream connection. 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 supported by many servers.

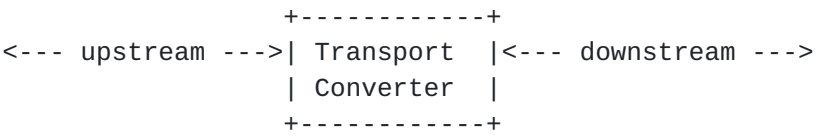

Figure 1: A Transport Converter relays data between pairs of TCP connections

Transport converters can be operated by network operators or third parties. 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 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.

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

deployement-specific.

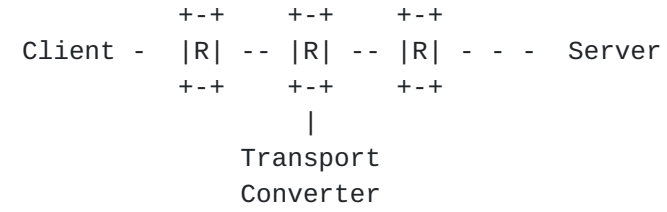

Figure 2: A Transport Converter can be installed anywhere in the network

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

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.

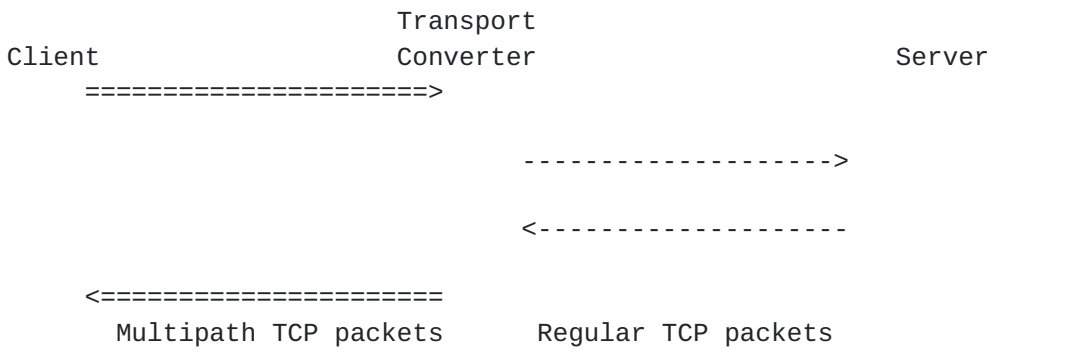

Figure 3: Different TCP variants can be used on the Client-Converter path and on the Converter-Server path

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.

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.

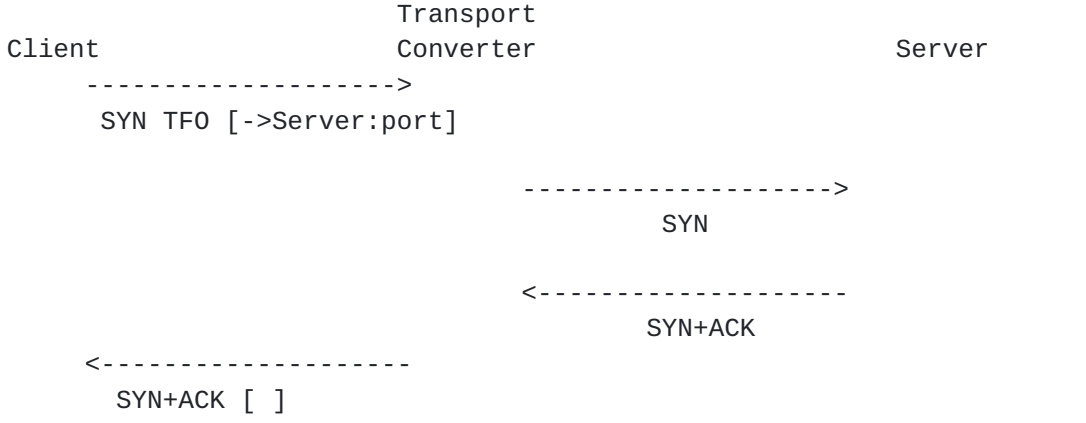

Figure 4: Establishment of a TCP connection through a Converter

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 supported by the Transport Converter. Thanks to this procedure, the Client knows which TCP options are supported by a given Transport Converter.

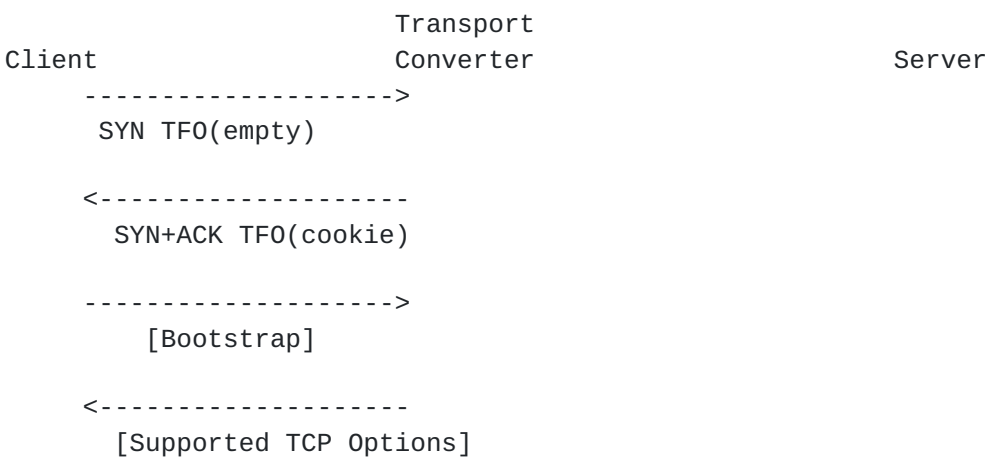

Figure 5: Bootstrapping a Client connection to a Transport Converter

Note that the Converter may rely on local policies to decide whether it can service a given requesting client. That is, the Converter may not return a cookie for that client.

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.

3.1. Sample Examples of Converter-Assisted Multipath TCP Connections

As an example, let us consider how such a 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.

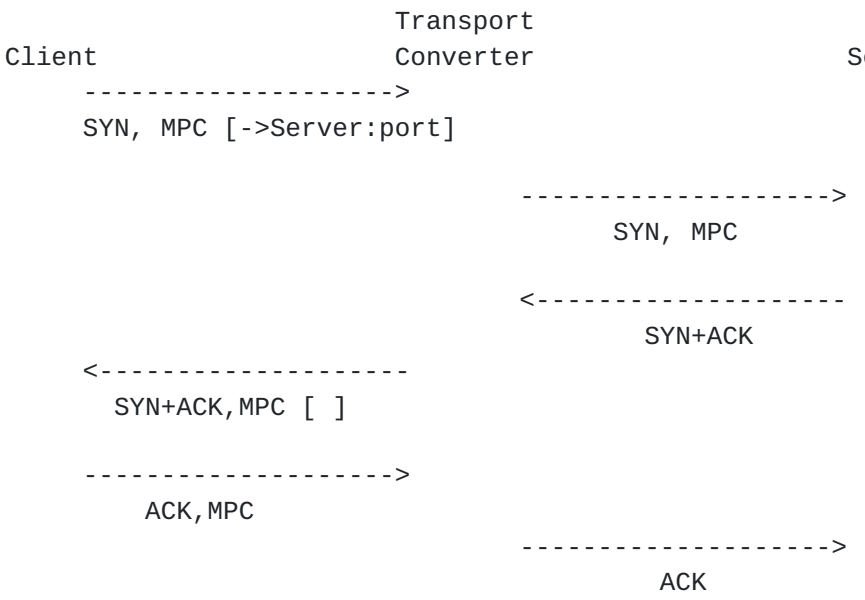

Figure 6: Establishment of a Multipath TCP connection through a Converter

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.

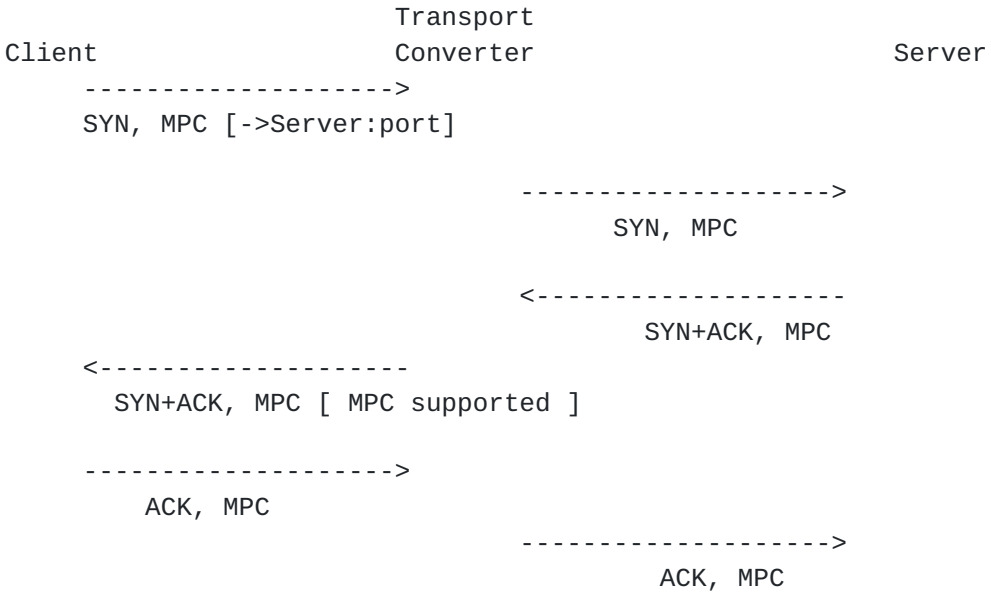

Figure 7: Establishment of a Multipath TCP connection through a converter

3.2. 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.

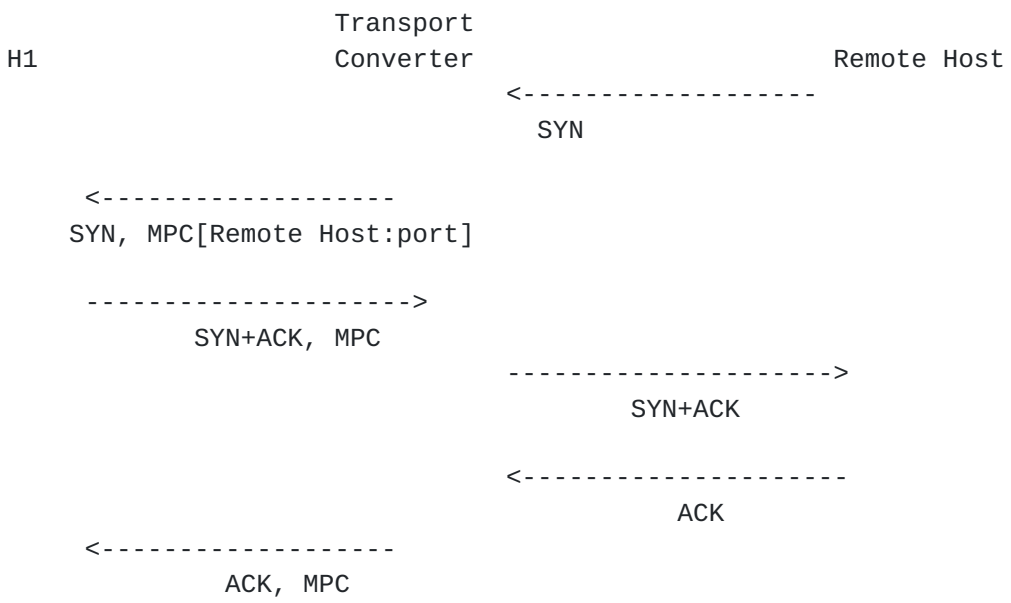

Figure 8: Establishment of an Incoming TCP Connection through a Converter

3.3. 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 9.

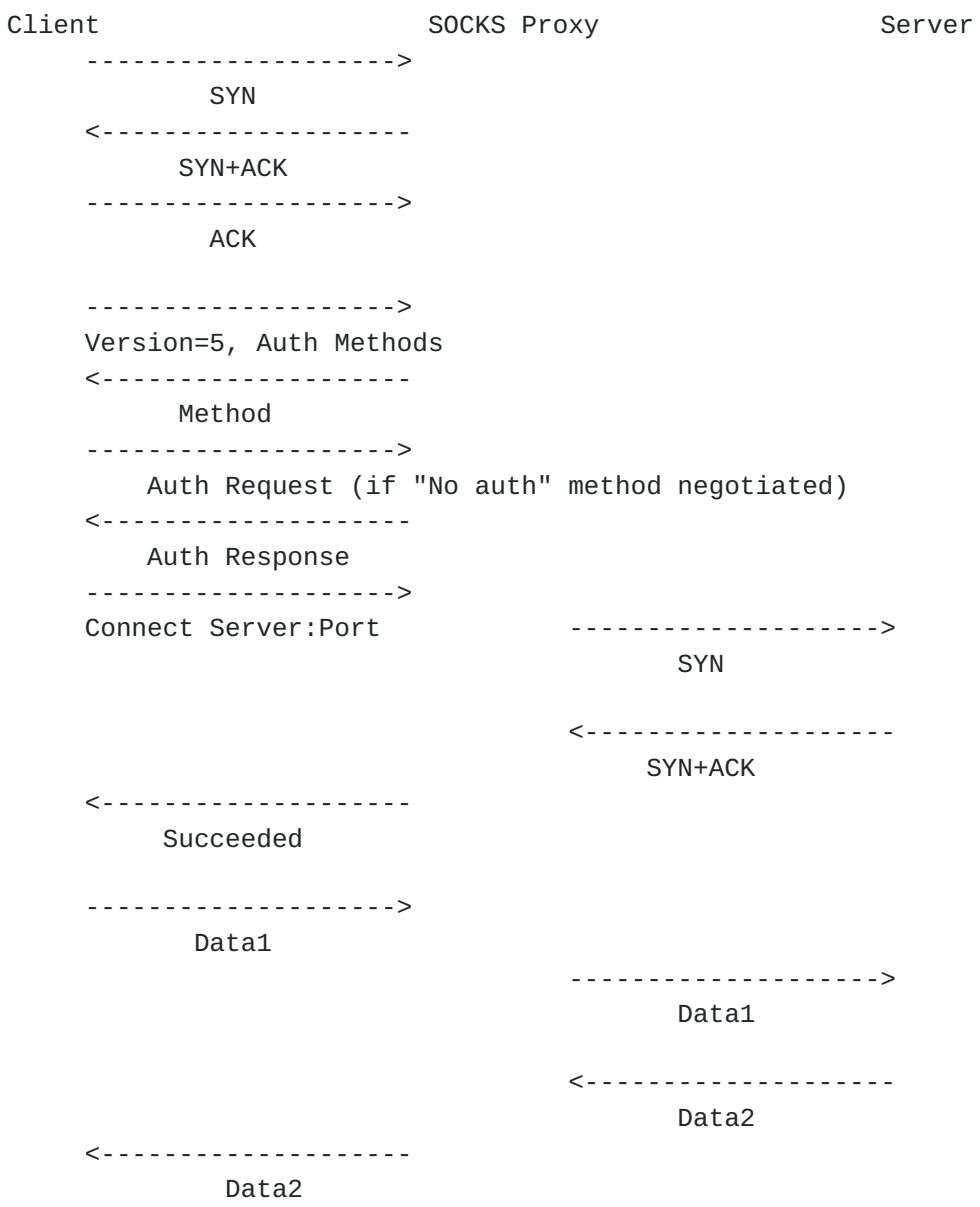

Figure 9: Establishment of a TCP connection through a SOCKS proxy without authentication

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].

4. The Converter Protocol

We now describe in details the messages that are exchanged between a Client and a Transport Converter. The Converter Protocol (CP) leverages the TCP Fast Open extension defined in [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 CP message.

4.1. 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.

4.2. The 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 10 as the first four bytes of the bytestream.

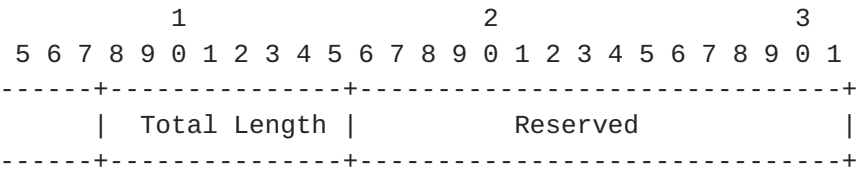

Figure 10: The fixed-sized header of the Converter protocol

The Version is encoded as an 8 bits unsigned integer value. This document specifies version 1. 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 Reserved field MUST be set to zero in this version of the protocol.

4.3. Transport Converter TLVs

The Converter protocol uses variable length messages that are encoded using a TLV format to simplify the parsing of the messages and leave room to extend the protocol in the future. A given TLV can 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.

Five TLVs are defined in this document. They are listed in Table 1.

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 Options TLV
30	0x1E	Variable	Error TLV

Table 1: The TLVs used by the Converter protocol

To use a given Transport Converter, a Client MUST first obtain a valid TFO cookie from it. 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 and the Transport Converter replies with the Supported TCP Options TLV that lists the TCP options that it supports (section Section 4.3.5).

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 4.3.1). 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 Section 4.3.3).

4.3.1. Connect TLV

This TLV (Figure 11) 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 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.

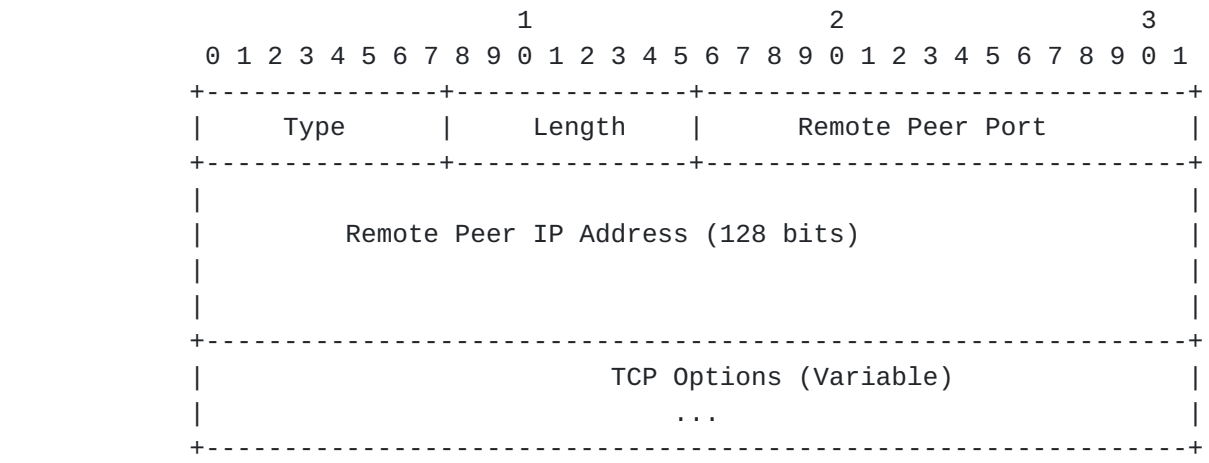

Figure 11: The Connect TLV

The 'TCP Options' field is a variable length field that carries a list of TCP Option fields (Figure 12). 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.

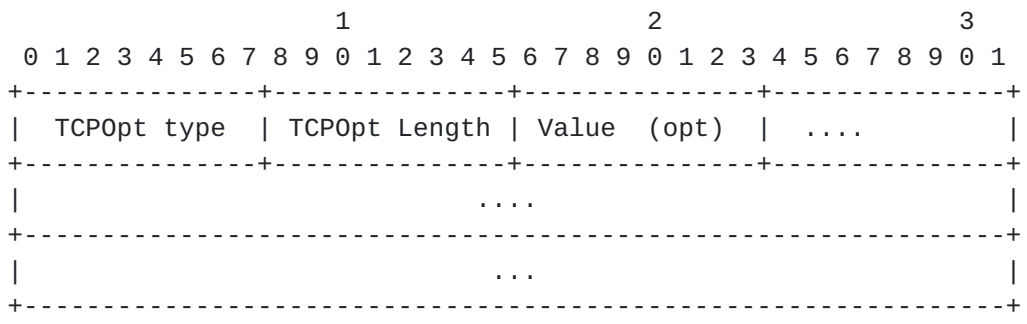

Figure 12: The TCP Options field

If a Transport Converter receives a Connect TLV with a non-empty TCP Options field, 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.

4.3.2. Extended TCP Header TLV

The Extended TCP Header TLV 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.

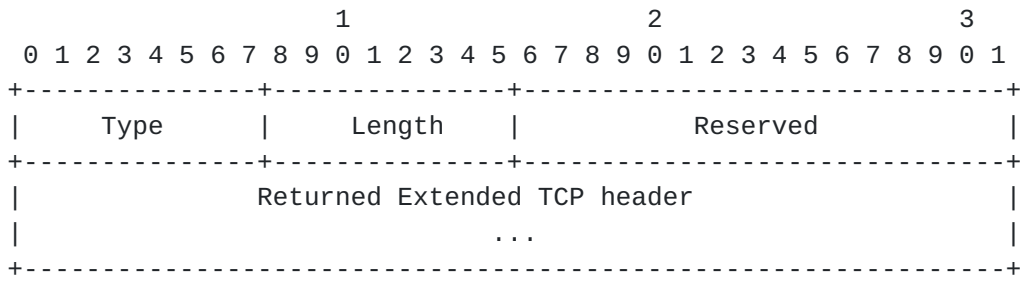

Figure 13: The Extended TCP Header TLV

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 Reserved field is set to zero by the transmitter and ignored by the receiver.

4.3.3. Error TLV

This optional TLV 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 Converter 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.

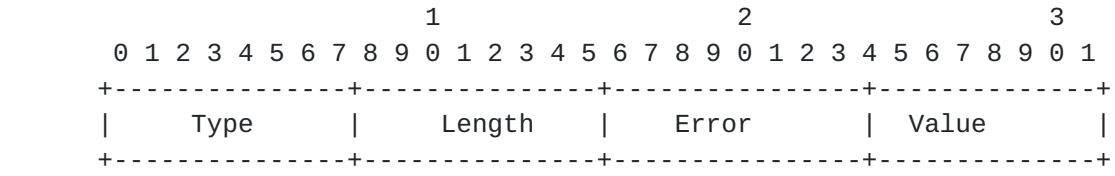

Figure 14: The Error TLV

Different types of errors can occur while processing Converter protocol messages. Each error is identified by a code represented as an unsigned integer. Four classes of errors are defined:

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

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

- Converter-side errors (64<= error code <96) : problems encountered on the Converter (e.g., lack of ressources) which prevent it from fulfilling the Client's request.

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

The following errors are defined in this document:

- 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.

- 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.

- 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.

- 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 is set to zero. This error code MUST be sent by the Converter when a request cannot be successfully processed because the authorization failed.

- Unsupported TCP Option (33). A TCP Option that the Client requested to advertise to the final Server is not supported by the Transport Converter. 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.

- 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.

- 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.

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

- Destination Unreachable (97): This error indicates that an ICMP destination unreachable, port unreachable, or network unreachable was received by the Converter. The Value field contains 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.

Table 2 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

Table 2: The different error codes

4.3.4. The Bootstrap TLV

The Bootstrap TLV is sent by a Client to request the TCP Extensions that are supported by a Transport Converter. It is typically sent on the first connection that a Client establishes with a Transport Converter to learn its capabilities. The Transport Converter replies with the Supported TCP Options TLV described in Section 4.3.5.

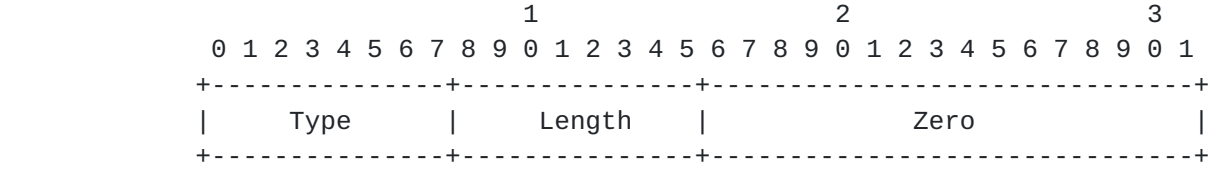

Figure 15: The Bootstrap TLV

4.3.5. Supported TCP Options TLV

The Supported TCP Options TLV is used by a Converter to announce the TCP options that it supports. Each supported TCP Option is encoded with its TCP option Kind listed in the TCP Parameters registry maintained by IANA. TCP option Kinds 0, 1, and 2 defined in [RFC0793] are supported by all TCP implementations and thus cannot appear in this list. The list of supported TCP Options is padded with 0 to end on a 32 bits boundary.

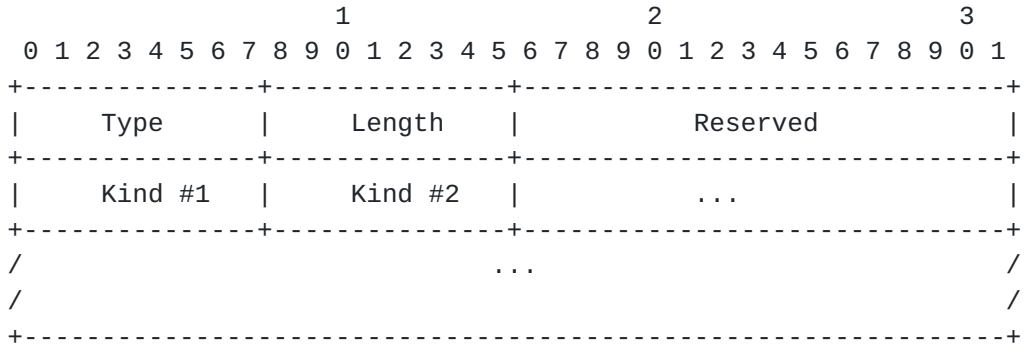

Figure 16: The Supported TCP Options TLV

5. 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 shown in Figure 5. 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].

6. Security Considerations

6.1. 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.

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

This document assumes that all network attachements 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.

6.2. 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 (Figure 5), the Transport Converter can verify that the Client correctly receives packets sent by the Converter. Stronger authentication schemes should be defined to use the Converter protocol in more open network environments.

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

6.3. 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].

6.4. 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.

6.5. 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:

- 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.

- 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-mptcp-radius]. This method does not require any interaction with the Converter.

- 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-mptcp-radius].

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.

7. IANA Considerations

This document requests the allocation of a reserved service name and port number for the converter protocol at https://www.iana.org/assignments/service-names-port-numbers/ service-names-port-numbers.xhtml.

This documents specifies version 1 of the Converter protocol. Five types of Converter messages are defined:

- 1: Bootstrap TLV
- 10: Connect TLV
- 20: Extended TCP Header TLV
- 21: Supported TCP Options TLV
- 30: Error TLV

Furthermore, it also defines the following 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

Table 3: The different error codes

8. 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 and Stefano Secci for their help in preparing this draft. Sri Gundavelli and 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.

8.1. Contributors

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

The authors of [I-D.boucadair-mptcp-plain-mode] were: - Mohamed Boucadair - Christian Jacquenet - Olivier Bonaventure - Denis Behaghel - Stefano Secci - Wim Henderickx - Robert Skog - Suresh Vinapamula - SungHoon Seo - Wouter Cloetens - Ullrich Meyer - Luis M. Contreras - Bart Peirens

The authors of [I-D.peirens-mptcp-transparent] were: - Bart Peirens - Gregory Detal - Sebastien Barre - Olivier Bonaventure

9. References

9.1. 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>.

[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>.

[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>.

[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>.

9.2. 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-mptcp-radius]
Boucadair, M. and C. Jacquenet, "RADIUS Extensions for Network-Assisted Multipath TCP (MPTCP)", draft-boucadair-mptcp-radius-05 (work in progress), October 2017.

[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-09 (work in progress), July 2017.

[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-11 (work in progress), November 2017.

[I-D.olteanu-intarea-socks-6]
Olteanu, V. and D. Niculescu, "SOCKS Protocol Version 6", draft-olteanu-intarea-socks-6-01 (work in progress), October 2017.

[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.

[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>.

[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>.

Authors' Addresses

Olivier Bonaventure
Tessares

Email: Olivier.Bonaventure@tessares.net

Mohamed Boucadair
Orange

Email: mohamed.boucadair@orange.com

Bart Peirens
Proximus

Email: bart.peirens@proximus.com

SungHoon Seo
Korea Telecom

Email: sh.seo@kt.com

Anandatirtha Nandugudi
Tessares

Email: anand.nandugudi@tessares.net