Abstract

Multipath TCP extends the plain, singlepath limited, TCP towards the capability of multipath transmission. This greatly improves the reliability and performance of TCP communication. For backwards compatibility reason the Multipath TCP was designed to setup successfully an initial path first, after which subsequent paths can be added for multipath transmission. For that reason the Multipath TCP has the same limitations as the plain TCP during connection setup, in case the selected path is not functional.

RobE provides the ability to simultaneously use multiple paths for connection setup. This document presents with RobE a set of extensions to Multipath TCP to overcome its singlepath limitation during connection initialisation and extends it to a full fledged multipath protocol. RobE ensures connectivity if at least one functional path out of a bunch of paths is given and offers beside that the opportunity to significantly improve loading times of Internet services.

Status of This Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 9, 2020.
1. Introduction

Multipath TCP Robust Session Establishment (MPTCP RobE) is a set of extensions to regular MPTCP [RFC6824] and its next upcoming version [I-D.ietf-mptcp-rfc6824bis], which releases single path limitations during the initial connection setup. Several scenarios require and benefit from a reliable and immediate connection setup which is not covered by [RFC6824] and [I-D.ietf-mptcp-rfc6824bis] so far. MPTCP was designed to be compliant with the TCP standard [RFC0793] and introduced the concept of an initial TCP flow and adding
subsequent flows after successful multipath negotiation on the initial path. While fulfilling its purpose, MPTCP is however fully dependent on the transmission characteristics of the communication link selected for initiating MPTCP.

RobE aims to overcome the limitations of [RFC6824] and [I-D.ietf-mptcp-rfc6824bis] using one initial flow and introduces the concept of potential initial flows triggered simultaneously. Potential initial flows give the freedom to use more than one path to request multipath connectivity and select the initial flow at a later point. Potential initial flow mechanisms and the gain of robustness and performance over the traditional MPTCP connection setup are evaluated in [RobE_slides] and [RobE_paper].

Two mechanisms of RobE, a simple and an extended version are specified in this document and are further referred to as RobE_SIM and RobE_EXT. RobE_SIM is a break-before-make mechanism, guaranteeing at least the robust connection establishment, however the RobE_EXT reuses every potential initial flow request to combine it with less overhead and accelerated multipath availability, leveraging a new MPTCP option MP_JOIN_CAP. From a standardization perspective, the RobE_SIM is fully compliant with [RFC6824] and [I-D.ietf-mptcp-rfc6824bis] and is herein more of a descriptive and procedural nature. The RobE_EXT requires a new MPTCP option but with the potential to significantly improve the MPTCP experience.

Table 1 summarizes the impact of RobE_SIM and RobE_EXT compared to [RFC6824] and [I-D.ietf-mptcp-rfc6824bis]. Considered are the scenarios where the initial path (IP) is disturbed and the impact on connection setup duration for the IP and multipath (MP) availability. In particular both setup durations are a mean to significantly improve the quality of experience.
### Table 1: Overview RobE features during initial connection setup

<table>
<thead>
<tr>
<th>Scenario</th>
<th>MPTCP</th>
<th>MPTCP + RobE_SIM</th>
<th>MPTCP + RobE_EXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP packet loss</td>
<td>Delayed connection</td>
<td>No impact</td>
<td>No impact</td>
</tr>
<tr>
<td>IP broken</td>
<td>No connection</td>
<td>No impact</td>
<td>No impact</td>
</tr>
<tr>
<td>IP setup duration</td>
<td>default route</td>
<td>fastest path</td>
<td>fastest path</td>
</tr>
<tr>
<td>MP availability duration</td>
<td>MP_CAPABLE + MP_JOIN</td>
<td>MP_CAPABLE + MP_JOIN</td>
<td>max(MP_CAPABLE_1, MP_CAPABLE_2)</td>
</tr>
</tbody>
</table>

IP: Initial Path; MP: Multi-Path

This document presents the protocol, procedures and changes required to extend MPTCP by a robust session setup; specifically, those for signaling and setting up multiple paths ("subflows")

#### 1.1. Terminology

This document makes use of a number of terms that are either MPTCP-specific or have defined meaning in the context of MPTCP, as follows:

Path: A sequence of links between a sender and a receiver, defined in this context by a 4-tuple of source and destination address/port pairs.

Subflow: A flow of TCP segments operating over an individual path, which forms part of a larger MPTCP connection. A subflow is started and terminated similar to a regular TCP connection.

#### 1.2. Comparison of RobE and traditional MPTCP

Figure 1 shows the traditional way of MPTCP handshaking with a MP_CAPABLE first, followed when successful by additional flows engaging MP_JOIN. MPTCP v0 [RFC6824] and MPTCP v1
[I-D.ietf-mptcp-rfc6824bis] differ in that a Key-A is sent with the first MP_CAPABLE or not.

<table>
<thead>
<tr>
<th></th>
<th>Host A</th>
<th>Host B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address A1</td>
<td></td>
<td>Address B1</td>
</tr>
<tr>
<td>Address A2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYN + MP_CAPABLE(Key-A[*])</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYN/ACK + MP_CAPABLE(Key-B)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACK + MP_CAPABLE(Key-A, Key-B)</td>
</tr>
<tr>
<td></td>
<td>SYN + MP_JOIN(Token-B, R-A)</td>
<td>SYN/ACK + MP_JOIN(HMAC-B, R-B)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACK + MP_JOIN(HMAC-A)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACK</td>
</tr>
</tbody>
</table>

[*] Key-A in the first MP-capable is related to MPTCP v0 only and does not exist in MPTCP v1.

Figure 1: MPTCP connection setup

Figure 2 changes the concept of the subsequent establishment in Figure 1 towards simultaneous requests (potential initial flows). For that, on several paths the MP_CAPABLE is used, like independent initial flow establishment. The final decision which path is selected as the main one and the handling of the remaining flow(s) differs in SA6.1 when RobE_SIM is applied or instead SA6.2 RobE_EXT.
1.3. Simple RobE (RobE_SIM)

Sender only implementation, no negotiation required, robust according to Table 1, connection setup benefits from fastest path.

1.4. Extended RobE (RobE_EXT)

Extends RobE_SIM by reusing the potential initial flows. This eliminates the overhead from RobE_SIM and accelerates the transmission speed by early availability of multiple paths. Further it relaxes the dependency on a reliable third ACK of the 3-way handshake in MPTCP v1.

2. Operation Overview

[*] Key-A in the first MP-capable is related to MPTCP v0 only and does not exist in MPTCP v1.

Figure 2: MPTCP RobE_SIM and RobE_EXT connection setup
2.1. MP_JOIN_CAP option

```
+---------------+---------------+-------+-------+---------------+
|     Kind      |    Length     |Subtype|       |    ADDR_ID    |
+---------------+---------------+-------+-------+---------------+
|                        Sender's Key-A (64 bits)                          |
+---------------------------------------------------------------+
|                                                               |
|                                                               |
|                        HMAC (>=96 bits)                          |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
+---------------------------------------------------------------+
```

Key-B_fast_hash = crc16(Key-B) & 0x3FF -> (10bit)
HMAC_keys = HMAC(Key-A+Key-B+Key-B') -> (>=96bit)
HMAC = (HMAC_keys & ~0x3FF) | Key-B_fast_hash -> (size HMAC_keys)

Figure 3: MP_JOIN_CAP

The ADDR_ID field of the MP_JOIN_CAP in Figure 3 is set according to the definition of MP_JOIN in [RFC6824] [I-D.ietf-mptcp-rfc6824bis].

2.2. RobE_EXT negotiation

[Tbd]

2.3. States of operation

[Tbd], according to Table 1 describe for MPTCPv0/v1 the different states which can occur; Section has DRAFT status

2.3.1. RobE_SIM

[Tbd]

2.3.2. RobE_EXT

1. In the flow diagram Figure 2, A1<>B1 is assumed to be the initial flow. A2<>B1 shall be recycled and the ACK is sent with MP_JOIN_CAP. Furthermore, the MP_CAPABLE arrives first at Host B (SB5) and the MP_JOIN_CAP afterwards (SB6.2). When the MP_JOIN_CAP is received, Host B has to iterate over the connection list once (like MP_JOIN) and check for Key-A availability. If a Key-A connection is found, this one is
validated against the HMAC value. The validation has two reasons: first, several Key-A can exist, because different hosts may choose the same Key-A by accident. Furthermore, no one can join a connection by just recording/brute-forcing Key-A and duplicating the request.

2. Like above, but MP_JOIN_CAP arrives before last MP_CAPABLE at Host B

* [I-D.ietf-mptcp-rfc6824bis]; Based on Key-A, Host B will iterate over the connection list, but it will not find a match, because Key-A of the previous selected initial flow (SA3, SA5) has not arrived yet. So it will continue with a fast iteration only over the connections which are still in establishment phase using the 10 bit Key-B fast hash (crc16(Key-B) & 0x3FF). If it matches against a (precomputed) existing Key-B_fast_hash in the connection list, it will validate the request using the HMAC(Key-A+B+B’) to ensure legitimation. If successful, both, the initial flow and the MP_JOIN_CAP flow, can be immediately established. This is true, because without the knowledge of Key-B, Host A could not calculate the HMAC. So it is clear, that Host A had received the SYN/ACK (SB3). This also mitigates the exchange of a reliable ACK during the handshake process. MPTCP sends the Key-A only with the last ACK and therefore prevents subsequent flow establishment until successful reception at Host B. Using RobE_EXT, the reception of a MP_JOIN_CAP ([I-D.ietf-mptcp-rfc6824bis]) is sufficient to establish both, the path carrying Key-B and Key-B’.

* [RFC6824]; Can match based on Key-A, same effort as for a MP JOIN.

3. A2<->B1 is selected as initial flow, because the respective SYN/ACK returns earlier at Host A. It is the same as above, just the other way round.

2.4. Fallback mechanism

[Tbd], e.g. one path does not support RobE or MP-TCPv0/v1

2.5. TFO support

[Tbd]
3. Practical implications

3.1. Performance compared to MP-TCP

[Tbd]

4. Security Considerations

[Tbd]

5. Acknowledgments

[Tbd]

6. IANA Considerations

[Tbd]

7. Informative References

[I-D.ietf-mptcp-rfc6824bis]


Author’s Address

Markus Amend
Deutsche Telekom
Deutsche-Telekom-Allee 7
64295 Darmstadt
Germany

Email: Markus.Amend@telekom.de