Stream: Internet Engineering Task Force (IETF)
RFC: 9897
Category: Standards Track
Published: January 2026
ISSN: 2070-1721
Authors:
M. Amend, Ed. *DT*
A. Brunstrom *Karlstad University*
A. Kassler *Karlstad University*
V. Rakocevic *City St George's, University of London*
S. Johnson *BT*

RFC 9897 Datagram Congestion Control Protocol (DCCP) Extensions for Multipath Operation with Multiple Addresses

Abstract

Datagram Congestion Control Protocol (DCCP) communications, as defined in RFC 4340, are inherently restricted to a single path per connection, despite the availability of multiple network paths between peers. The ability to utilize multiple paths simultaneously for a DCCP session can enhance network resource utilization, improve throughput, and increase resilience to network failures, ultimately enhancing the user experience.

Use cases for Multipath DCCP (MP-DCCP) include mobile devices (e.g., handsets and vehicles) and residential home gateways that maintain simultaneous connections to distinct network types such as cellular and Wireless Local Area Networks (WLANs) or cellular and fixed access networks. Compared to existing multipath transport protocols, such as Multipath TCP (MPTCP), MP-DCCP is particularly suited for latency-sensitive applications with varying requirements for reliability and in-order delivery.

This document specifies a set of protocol extensions to DCCP that enable multipath operations. These extensions maintain the same service model as DCCP while introducing mechanisms to establish and utilize multiple concurrent DCCP flows across different network paths.

Status of This Memo

This is an Internet Standards Track document.

Copyright Notice

Table of Contents

1. Introduction

The Datagram Congestion Control Protocol (DCCP) [RFC4340] is a transport protocol that provides bidirectional unicast connections of congestion-controlled unreliable datagrams. DCCP communications are restricted to one single path. Other fundamentals of the DCCP protocol are summarized in Section 1 of [RFC4340] such as the reliable handshake process in Section 4.7 of [RFC4340] and the reliable negotiation of features in Section 4.5 of [RFC4340]. These are an important basis for this document. These fundamentals also apply to the DCCP sequencing scheme, which is packet-based (Section 4.2 of [RFC4340]), and the principles for loss and retransmission of features as described in more detail in Section 6.6.3 of [RFC4340]. This document specifies a set of protocol changes that add multipath support to DCCP, specifically support for signaling and setting up multiple paths (a.k.a., "subflows"), managing these subflows, the reordering of data, and the termination of sessions.

Multipath DCCP (MP-DCCP) enables a DCCP connection to simultaneously establish a flow across multiple paths. This can be beneficial to applications that transfer large amounts of data, by utilizing the capacity/connectivity offered by multiple paths. In addition, the multipath extensions enable the trade-off of timeliness and reliability, which is important for low-latency applications that do not require guaranteed delivery services such as Audio/Video streaming.

In addition to the integration into DCCP services, implementers or future specifications could choose MP-DCCP for other use cases such as 3GPP 5G multi-access solutions (e.g., Access Traffic Steering, Switching, and Splitting (ATSSS) specified in [TS23.501]) or hybrid access networks. ATSSS combines 3GPP and non-3GPP access between the user equipment and an operator network, while hybrid access combines fixed and cellular access between a residential gateway and an operator network. MP-DCCP can be used in these scenarios for load balancing, seamless session handover, and bandwidth aggregation when non-DCCP traffic such as IP, UDP, or TCP is encapsulated into MP-DCCP. More details on potential use cases for MP-DCCP are provided in [MP-DCCP.Site], [IETF105.Slides], and [MP-DCCP.Paper]. All of these use cases profit from an Open Source Linux reference implementation provided under [MP-DCCP.Site].

The encapsulation of non-DCCP traffic (e.g., UDP or IP) in MP-DCCP to enable the above-mentioned use cases is not considered in this specification. Also out of scope is the encapsulation of DCCP traffic in UDP to pass middleboxes (e.g., NATs, firewalls, proxies, intrusion detection systems (IDSs), etc.) that do not support DCCP. However, a possible method is defined in [RFC6773] and considered in [U-DCCP] to achieve the same with less overhead.

MP-DCCP is based exclusively on the lean concept of DCCP. For traffic that is already encrypted or does not need encryption, MP-DCCP is an efficient choice as it does not apply its own encryption mechanisms. Also, the procedures defined by MP-DCCP, which allow subsequent reordering of traffic and efficient traffic scheduling, improve performance, as shown in [MP-DCCP.Paper], and take into account the interaction of the protocol with the further elements required for multipath transport.

1.1. Multipath DCCP in the Networking Stack

MP-DCCP provides a set of features to DCCP; Figure 1 illustrates this layering. MP-DCCP is designed to be used by applications in the same way as DCCP with no changes to the application itself.

```
                             +-------------------------------+
                             |           Application         |
+---------------+            +-------------------------------+
|  Application  |            |            MP-DCCP            |
+---------------+            + - - - - - - - + - - - - - - - +
|      DCCP     |            |Subflow (DCCP) |Subflow (DCCP) |
+---------------+            +-------------------------------+
|      IP       |            |      IP       |      IP       |
+---------------+            +-------------------------------+
```

Figure 1: Comparison of Standard DCCP and MP-DCCP Protocol Stacks

A command-line interface (CLI) at the endpoint (or another method) could be used to configure and manage the DCCP connections. This could be extended to also support MP-DCCP, but this specification does not define it.

1.2. Terminology

This document uses terms that are either specific for multipath transport as defined in [RFC8684] or defined in the context of MP-DCCP, as follows:

Path: A sequence of links between a sender and a receiver, defined in this context by a 4-tuple of the source and destination address and the source and destination ports. This definition follows [RFC8684] and is illustrated in the following two examples for IPv6 and IPv4, which each show a pair of sender IP-address:port and a pair of receiver IP-address:port, which together form the 4-tuple:

- IPv6: [2001:db8:3333:4444:5555:6666:7777:8888]:1234, [2001:db8:3333:4444:cccc:dddd:eeee:ffff]:4321
- IPv4: 203.0.113.1:1234, 203.0.113.2:4321

Subflow: A DCCP flow that is transmitted by using a specific path (4-tuple of source and destination address/port pairs) that forms one of the multipath flows used by a single connection.

(MP-DCCP) Connection: A set of one or more subflows, over which an application can communicate between two hosts. The MP-DCCP connection is exposed as a single DCCP socket to the application.

Connection Identifier (CI): A unique identifier that is assigned to a multipath connection by the host to distinguish several multipath connections locally. The CIs must therefore be locally unique per host and do not have to be the same across the peers.

Host: An end host that operates an MP-DCCP implementation and either initiates or accepts an MP-DCCP connection.

'+': The plus symbol means the concatenation of values.

In addition to these terms, within the framework of MP-DCCP, the interpretation of, and effect on, regular single-path DCCP semantics is discussed in Section 3.

1.3. Requirements Language

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Operation Overview

DCCP transmits congestion-controlled unreliable datagrams over a single path. Various congestion control mechanisms have been specified to optimize DCCP performance for specific traffic types in terms of profiles denoted by a Congestion Control IDentifier (CCID). However, DCCP does not provide built-in support for managing multiple subflows within one DCCP connection. The extension of DCCP for Multipath DCCP (MP-DCCP) is described in detail in Section 3.

At a high level of MP-DCCP operation, the data stream from a DCCP application is split by the MP-DCCP operation into one or more subflows that can be transmitted via different paths, for example, using paths via different links. The corresponding control information allows the receiver to optionally reassemble and deliver the received data in the originally transmitted order to the recipient application. This may be necessary because DCCP does not guarantee in-order delivery. The details of the transmission scheduling mechanism and optional reordering mechanism are up to the sender and receiver, respectively, and are outside the scope of this document.

An MP-DCCP connection provides a bidirectional connection of datagrams between two hosts exchanging data using DCCP. It does not require any change to the applications. MP-DCCP enables the hosts to use multiple paths with different 4-tuples to transport the packets of an MP-DCCP connection. MP-DCCP manages the request, set-up, authentication, prioritization, modification, and removal of the DCCP subflows on different paths as well as the exchange of performance parameters.

The number of DCCP subflows can vary during the lifetime of an MP-DCCP connection. The details of the path management decisions for when to add or remove subflows are outside the scope of this document.

The multipath capability for MP-DCCP is negotiated with a new DCCP feature, as specified in Section 3.1. Once negotiated, all subsequent MP-DCCP operations for that connection are signaled with a variable length multipath-related option, as described in Section 3. All MP-DCCP operations are signaled by Multipath Options described in Section 3.2. Options that require confirmation from the remote peer are retransmitted by the sender until confirmed or until confirmation is no longer considered relevant.

The sections that follow define MP-DCCP behavior in detail.

2.1. MP-DCCP Concept

Figure 2 provides a general overview of the MP-DCCP working mode, whose main characteristics are summarized in this section.

```
           Host A                               Host B
------------------------             ------------------------
Address A1    Address A2             Address B1    Address B2
----------    ----------             ----------    ----------
  |             |                      |             |
  |         (DCCP subflow setup)       |             |
  |----------------------------------->|             |
  |<-----------------------------------|             |
  |             |                      |             |
  |             |  (DCCP subflow setup)|             |
  |             |--------------------->|             |
  |             |<---------------------|             |
  | merge individual DCCP subflows to one MP-DCCP connection
  |             |                      |             |
```

Figure 2: Example MP-DCCP Usage Scenario

- An MP-DCCP connection begins with a 4-way handshake between two hosts. In Figure 2, an MP-DCCP connection is established between addresses A1 and B1 on Hosts A and B. In the handshake, a Multipath Capable Feature is used to negotiate multipath support for the connection. Host-specific keys are also exchanged between Host A and Host B during the handshake. The details of the MP-DCCP handshake procedure is described in Section 3.3. MP-DCCP does not require both peers to have more than one address.
- When additional paths and corresponding addresses/ports are available, additional DCCP subflows can be created on these paths and attached to the existing MP-DCCP connection. An MP_JOIN option is used to connect a new DCCP subflow to an existing MP-DCCP connection. It contains a Connection Identifier (CI) during the setup of the initial subflow and is exchanged in the 4-way handshake for the subflow together with the Multipath Capable

Feature. The example in Figure 2 illustrates the creation of an additional DCCP subflow between Address A2 on Host A and Address B1 on Host B. The two subflows continue to provide a single connection to the applications at both endpoints.

- MP-DCCP identifies multiple paths by the presence of multiple addresses/ports at hosts. Combinations of these multiple addresses/ports indicate the additional paths. In the example, other potential paths that could be set up are A1<->B2 and A2<->B2. Although the additional subflow in the example is shown as being initiated from A2, an additional subflow could alternatively have been initiated from B1 or B2.
- The discovery and setup of additional subflows is achieved through a path management method including the logic and details of the procedures for adding/removing subflows. This document describes the procedures that enable a host to initiate new subflows or to signal available IP addresses between peers. However, the definition of a path management method, in which sequence and when subflows are created, is outside the scope of this document. This method is subject to a corresponding policy and the specifics of the implementation. If an MP-DCCP peer host wishes to limit the maximum number of paths that can be maintained (e.g., similar to that discussed in Section 3.4 of [RFC8041]), the creation of new subflows from that peer host is omitted when the threshold of maximum paths is exceeded and incoming subflow requests **MUST** be rejected.
- Through the use of Multipath Options, MP-DCCP adds connection-level sequence numbers and the exchange of Round-Trip Time (RTT) information to enable optional reordering features. As a hint for scheduling decisions, a Multipath Option that allows a peer to indicate its priorities for which path to use is also defined.
- Subflows are terminated in the same way as regular DCCP connections, as described in Section 8.3 of [RFC4340]. MP-DCCP connections are closed by including an MP_CLOSE option in subflow DCCP-CloseReq or DCCP-Close messages. An MP-DCCP connection may also be reset through the use of an MP_FAST_CLOSE option. Key Data from the initial handshake is included in MP_CLOSE and MP_FAST_CLOSE to protect from an unauthorized shutdown of MP-DCCP connections.

3. MP-DCCP Protocol

The DCCP protocol feature list (Section 6.4 of [RFC4340]) is extended in this document by adding a new Multipath Feature with Feature Number 10, as shown in Table 1.

Number	Meaning	Rec'n Rule	Initial Value	Req'd
10	Multipath Capable	SP	0	N

Table 1: Multipath Feature

Rec'n Rule: The reconciliation rule used for the feature. SP indicates the server-priority as defined in Section 6.3 of [RFC4340].

Initial Value: The initial value for the feature. Every feature has a known initial value.

Req'd: This column is "Y" if and only if every DCCP implementation **MUST** understand the feature. If it is "N", then the feature behaves like an extension, and it is safe to respond to Change options for the feature with empty Confirm options.

This specification adds a DCCP protocol option as defined in Section 5.8 of [RFC4340], providing a new multipath-related variable-length option with option type 46, as shown in Table 2.

Type	Option Length	Meaning	DCCP-Data?
46	variable	Multipath	Y

Table 2: Multipath Option Set

3.1. Multipath Capable Feature

A DCCP endpoint negotiates the Multipath Capable Feature to determine whether multipath extensions can be enabled for a DCCP connection.

The Multipath Capable Feature (MP_CAPABLE) has Feature Number 10 and follows the structure for features given in Section 6 of [RFC4340]. Beside the negotiation of the feature itself, one or several values can also be exchanged. The value field specified here for the Multipath Capable Feature has a Length of one byte and can be repeated several times within the DCCP option for feature negotiation. This can be, for example, required to announce support of different versions of the protocol. For that, the leftmost four bits in Figure 3 specify the compatible version of the MP-DCCP implementation and **MUST** be set to 0 following this specification. The four bits following the Version field are unassigned in version 0 and **MUST** be set to zero by the sender and **MUST** be ignored by the receiver.

```
  0   1   2   3    4   5   6   7
+-----------+------------+
|  Version   | Unassigned |
+-----------+------------+
```

Figure 3: Format of the Multipath Capable Feature Value Field

The setting of the Multipath Capable Feature **MUST** follow the server-priority reconciliation rule described in Section 6.3.1 of [RFC4340]. This allows multiple versions to be specified in order of priority.

The negotiation **MUST** be a part of the initial handshake procedure described in Section 3.3. No subsequent renegotiation of the Multipath Capable Feature is allowed for the same MP-DCCP connection.

Clients **MUST** include a Change R option (Section 6 of [RFC4340]) during the initial handshake request to supply a list of supported MP-DCCP protocol versions, ordered by preference.

Servers **MUST** include a Confirm L option (Section 6 of [RFC4340]) in the subsequent response to agree on an MP-DCCP version to be used from the Client list, followed by its own supported version(s), ordered by preference. Any subflow added to an existing MP-DCCP connection **MUST** use the version negotiated for the first subflow.

If no agreement is found, the Server **MUST** reply with an empty Confirm L option with Feature Number 10 and no values.

An example of successful version negotiation is shown hereafter and follows the negotiation example shown in Section 6.5 of [RFC4340]. For better understanding, this example uses the unspecified MP-DCCP versions 1 and 2 in addition to the MP-DCCP version 0 specified in this document:

```
Client                                               Server
------                                               ------
DCCP-Req + Change R(MP_CAPABLE, 1 0)
               ------------------------------------>

                DCCP-Resp + Confirm L(MP_CAPABLE, 1, 2 1 0)
      <------------------------------------

           * agreement on version = 1 *
```

Figure 4: Example of MP-DCCP Support Negotiation Using MP_CAPABLE

This example illustrates the following:

1. The Client indicates support for both MP-DCCP versions 1 and 0, with a preference for version 1.
2. The Server agrees on using MP-DCCP version 1 indicated by the first value and supplies its own preference list with the subsequent values.
3. MP-DCCP is then enabled between the Client and Server with version 1.

Unlike the example in Figure 4, this document only allows the negotiation of MP-DCCP version 0. Therefore, per successful negotiation of MP-DCCP as defined in this document, the Client and the Server **MUST** both support MP-DCCP version 0.

If the version negotiation fails or the Multipath Capable Feature is not present in the DCCP-Request or DCCP-Response packets of the initial handshake procedure, the MP-DCCP connection either **MUST** fall back to regular DCCP or **MUST** close the connection. Further details are specified in Section 3.6.

3.2. Multipath Option

MP-DCCP uses one single option to signal various multipath-related operations. The format of this Multipath Option is shown in Figure 5.

```
            1          2          3
 01234567 89012345 67890123 45678901 23456789
+--------+--------+--------+--------+--------+
|00101110| Length | MP_OPT | Value(s) ...
+--------+--------+--------+--------+--------+
 Type=46
```

Figure 5: Multipath Option Format

The fields used by the Multipath Option are described in Table 3. MP_OPT refers to a Multipath Option.

Type	Option Length	MP_OPT	Meaning
46	var	0 =MP_CONFIRM	Confirm reception and processing of an MP_OPT option
46	12	1 =MP_JOIN	Join subflow to an existing MP-DCCP connection
46	var	2 =MP_FAST_CLOSE	Close an MP-DCCP connection unconditionally
46	var	3 =MP_KEY	Exchange key material for MP_HMAC
46	9	4 =MP_SEQ	Multipath sequence number
46	23	5 =MP_HMAC	Hash-based message authentication code for MP-DCCP
46	12	6 =MP_RTT	Transmit RTT values and calculation parameters
46	var	7 =MP_ADDADDR	Advertise one or more additional addresses/ports
46	8	8 =MP_REMOVEADDR	Remove one or more addresses/ports
46	4	9 =MP_PRIO	Change subflow priority
46	var	10 =MP_CLOSE	Close an MP-DCCP connection
46	var	11 =MP_EXP	Experimental option for private use

Type	Option Length	MP_OPT	Meaning
46	TBD	>11	(available for future Multipath Options)

Table 3: MP_OPT Option Types

Future Multipath Options could be defined in a later version of or extension to this specification.

These operations are largely inspired by the signals defined in [RFC8684]. The procedures for handling faulty or unknown Multipath Options are described in Section 3.6.

3.2.1. MP_CONFIRM

Some Multipath Options require confirmation from the remote peer (see Table 4) for which MP_CONFIRM is specified.

```
             1          2          3           4          5
 01234567 89012345 67890123 45678901 23456789 01234567 89012345
+--------+--------+--------+--------+--------+--------+--------+
|00101110|  var   |00000000| List of confirmations ...
+--------+--------+--------+--------+--------+--------+--------+
 Type=46   Length  MP_OPT=0
```

Figure 6: Format of the MP_CONFIRM Option

Multipath Options that require confirmation will be retransmitted by the sender until an MP_CONFIRM is received or the confirmation of options is considered irrelevant because the data contained in the options has already been replaced by newer information.

This can happen, for example, with an MP_PRIO option if the path prioritization is changed while the previous prioritization has not yet been confirmed. The further processing of the Multipath Options in the receiving host is not the subject of MP_CONFIRM.

Multipath Options could arrive out of order; therefore, Multipath Options defined in Table 4 **MUST** be sent in a DCCP datagram with MP_SEQ (see Section 3.2.5). This allows a receiver to identify whether Multipath Options are associated with obsolete datasets (information carried in the option header) that would otherwise conflict with newer datasets. In the case of MP_ADDADDR or MP_REMOVEADDR, the same dataset is identified based on AddressID, whereas the same dataset for MP_PRIO is identified by the subflow in use. An outdated multipath Option is detected at the receiver if a previous Multipath Option referring to the same dataset contained a higher sequence number in the MP_SEQ. An MP_CONFIRM **MAY** be generated for Multipath Options that are identified as outdated.

Similarly, an MP_CONFIRM could arrive out of order. The associated MP_SEQ received **MUST** be echoed to ensure that the most recent Multipath Option is confirmed. This protects from inconsistencies that could occur, e.g., if three MP_PRIO options are sent one after the other on one path in order to first set the path priority to 0, then to 1, and finally to 0 again. Without an associated MP_SEQ, a loss of the third MP_PRIO option and a loss of the MP_CONFIRM of the second update and the third update would cause the sender to incorrectly interpret that the priority value was set to 0 without recognizing that the receiver has applied priority value 1.

The length of the MP_CONFIRM option and the path over which the option is sent depend on the confirmed Multipath Options and the received MP_SEQ, which are both copied verbatim and appended as a list of confirmations. The list is structured by first listing the received MP_SEQ followed by the related Multipath Option or options to confirm. The same rules apply when Multipath Options with different MP_SEQs are confirmed at once. This could happen if the following are received in short succession: a datagram with MP_PRIO and a first MP_SEQ_1 and another datagram with MP_ADDADDR and a second MP_SEQ_2. In this case, the structure described above is concatenated resulting in MP_SEQ_2 + MP_ADDADDR + MP_SEQ_1 + MP_PRIO. The order of the confirmed Multipath Options in the list of confirmations **MUST** reflect the incoming order at the host who sends the MP_CONFIRM, with the most recent suboption received listed first. This could allow the host receiving the MP_CONFIRM to verify that the options were applied in the correct order and to take countermeasures if they were not, e.g., if an MP_REMOVEADDR overtakes an MP_ADDADDR that refers to the same dataset.

Type	Option Length	MP_OPT	MP_CONFIRM Sending Path
46	var	7 =MP_ADDADDR	Any available
46	4	8 =MP_REMOVEADDR	Any available
46	4	9 =MP_PRIO	Any available

Table 4: Multipath Options Requiring Confirmation

An example to illustrate the MP-DCCP confirm procedure for the MP_PRIO option is shown in Figure 7. Host A sends a DCCP-Request on path A2-B2 with an MP_PRIO option with value 1 and an associated sequence number of 1. Host B replies on the same path in this instance (any path can be used) with a DCCP-Response containing the MP_CONFIRM option and a list containing the original sequence number (1) together with the associated option (MP_PRIO).

```
          Host A                                    Host B
------------------------                   ------------------------
Address A1    Address A2                   Address B1    Address B2
----------    ----------                   ----------    ----------
     |             |                            |            |
     |             | DCCP-Request(seqno 1) + MP_PRIO(1)|      |
     |             |------------------------------------------>|
     |             |                            |            |
     |             | DCCP-Response +            |            |
     |             |<---- MP_CONFIRM(seqno 1, MP_PRIO) --------|
     |             |                            |            |
```

Figure 7: Example MP_CONFIRM Procedure

A second example that illustrates the same MP-DCCP confirm procedure but where an out-of-date option is also delivered is shown in Figure 8. Here, the first DCCP-Data is sent from Host A to Host B with option MP_PRIO set to 4. Host A subsequently sends the second DCCP-Data with option MP_PRIO set to 1. In this case, the delivery of the first MP_PRIO is delayed in the network between Host A and Host B and arrives after the second MP_PRIO. Host B ignores this second MP_PRIO as the associated sequence number is earlier than the first. Host B sends a DCCP-Ack with sequence number 2 to confirm the receipt of the MP_PRIO(1).

```
          Host A                                      Host B
------------------------                     ------------------------
Address A1    Address A2                     Address B1    Address B2
----------    ----------                     ----------    ----------
     |             |                              |             |
     |             | DCCP-Data(seqno 1) +  MP_PRIO(4)  |        |
     |             |------------                  |             |
     |             |            \                 |             |
     |             | DCCP-Data(seqno 2) +  MP_PRIO(1)  |        |
     |             |--------------\---------------------------->|
     |             |               \              |             |
     |             |                ------------------------------>|
     |             |                              |             |
     |             | DCCP-Ack +                   |             |
     |             |<---- MP_CONFIRM(seqno 2, MP_PRIO) --------|
     |             |                              |             |
```

Figure 8: Example MP_CONFIRM Procedure with an Outdated Suboption

3.2.2. MP_JOIN

The MP_JOIN option is used to add a new subflow to an existing MP-DCCP connection, and a successful establishment of the first subflow using MP_KEY is **REQUIRED**.

```
           1          2          3
 01234567 89012345 67890123 45678901
+--------+--------+--------+--------+
|00101110|00001100|00000001| Addr ID|
+--------+--------+--------+--------+
| Connection Identifier             |
+--------+--------+--------+--------+
| Nonce                             |
+--------+--------+--------+--------+
 Type=46  Length=12 MP_OPT=1
```

Figure 9: Format of the MP_JOIN Suboption

The CI is the one from the peer host, which was previously exchanged with the MP_KEY option. MP_HMAC **MUST** be set when using MP_JOIN within a DCCP-Response packet; see Section 3.2.6 for details. Similar to the setup of the first subflow, MP_JOIN also exchanges the Multipath

Capable Feature MP_CAPABLE as described in Section 3.1. This procedure includes the DCCP Confirm principle and thus ensures a reliable exchange of the MP_JOIN in accordance with Section 6.6.4 of [RFC4340].

The MP_JOIN option includes an "Addr ID" (Address ID) generated by the sender of the option, which is used to identify the source address of this packet, even if the IP header was changed in transit by a middlebox. The value of this field is generated by the sender and **MUST** map uniquely to a source IP address for the sending host. The Address ID allows address removal (Section 3.2.9) without the need to know the source address at the receiver, thus allowing address removal through NATs. The Address ID also allows correlation between new subflow setup attempts and address signaling (Section 3.2.8), to prevent setting up duplicate subflows on the same path, if an MP_JOIN and MP_ADDADDR are sent at the same time.

The Address IDs of the subflow used in the initial DCCP Request/Response exchange of the first subflow in the connection are implicit and have the value zero. A host **MUST** store the mappings between Address IDs and addresses for both itself and the remote host. An implementation will also need to know which local and remote Address IDs are associated with which established subflows for when addresses are removed from a local or remote host. An Address ID **MUST** always be unique over the lifetime of a subflow and can only be reassigned if sender and receiver no longer have them in use.

The Nonce is a 32-bit random value locally generated for every MP_JOIN option. Together with the derived key from both hosts' Key Data (as described in Section 3.2.4), the Nonce value builds the basis to calculate the Hash-based Message Authentication Code (HMAC) used in the handshake process (as described in Section 3.3) to avoid replay attacks.

If the CI cannot be verified by the receiving host during a handshake negotiation, the new subflow **MUST** be closed, as specified in Section 3.6.

3.2.3. MP_FAST_CLOSE

DCCP can send a Close or Reset signal to abruptly close a connection. Using MP-DCCP, a regular Close or Reset only has the scope of the subflow over which a signal was received. As such, it will only close the subflow and does not affect other remaining subflows or the MP-DCCP connection (unless it is the last subflow). This permits break-before-make handover between subflows.

In order to provide an MP-DCCP-level "reset" and thus allow the abrupt closure of the MP-DCCP connection, the MP_FAST_CLOSE suboption can be used.

```
           1          2          3
 01234567 89012345 67890123 45678901 23456789
+--------+--------+--------+--------+--------+
|00101110|  var   |00000010| Key Data ...
+--------+--------+--------+--------+--------+
 Type=46   Length  MP_OPT=2
```

Figure 10: Format of the MP_FAST_CLOSE Suboption

When Host A wants to abruptly close an MP-DCCP connection with Host B, it will send out the MP_FAST_CLOSE. The MP_FAST_CLOSE suboption **MUST** be sent from Host A on all subflows using a DCCP-Reset packet with Reset Code 13. The requirement to send the MP_FAST_CLOSE on all subflows increases the probability that Host B will receive the MP_FAST_CLOSE to take the same action. To protect from an unauthorized shutdown of an MP-DCCP connection, the selected Key Data of the peer host during the handshake procedure is carried by the MP_FAST_CLOSE option.

After sending the MP_FAST_CLOSE on all subflows, Host A **MUST** tear down all subflows, and the MP-DCCP connection immediately terminates.

Upon reception of the first MP_FAST_CLOSE with successfully validated Key Data, Host B will send a DCCP-Reset packet response on all subflows to Host A with Reset Code 13 to clean potential middlebox states. Host B **MUST** then tear down all subflows and terminate the MP-DCCP connection.

3.2.4. MP_KEY

MP-DCCP protects against some on-path attacker as further outlined in Section 4. The basis of this protection is laid by an initial exchange of keys during the MP-DCCP connection setup, for which MP_KEY is introduced. The basis of this protection is laid by an initial exchange of keys during the MP-DCCP connection setup, for which MP_KEY is introduced.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
|0 0 1 0 1 1 1 0|      var      |0 0 0 0 0 0 1 1|     resvd     |
+---------------+---------------+---------------+---------------+
|                     Connection Identifier                     |
+---------------+---------------+---------------+---------------+
|  Key Type (1) |  Key Data (1) |  Key Type (2) |  Key Data (2) |
+---------------+---------------+---------------+---------------+
|  Key Type (3) | ...
+---------------+---------------+
    Type=46          Length          MP_OPT=3
```

Figure 11: Format of the MP_KEY Suboption

The MP_KEY suboption is used to exchange a CI and key material between hosts (Host A and Host B) for a given connection. The CI is a unique number in the host for each multipath connection and is generated for inclusion in the first exchange of a connection with MP_KEY. With the CI, it is possible to connect other DCCP subflows to an MP-DCCP connection with MP_JOIN (Section 3.2.2). Its size of 32 bits also defines the maximum number of simultaneous MP-DCCP connections in a host to 2^{32}. According to the Key-related elements of the MP_KEY suboption, the Length varies between 17 and 73 bytes for a single-key message and up to 82 bytes when all specified Key Types 0 and 255 are provided. The Key Type field specifies the type of the following Key Data. The set of Key Types are shown in Table 5.

Key Type	Key Length (bytes)	Meaning
0 =Plain Text	8	Plain text Key
1-254		(available for future Key Types)
255 =Experimental	64	For private use only

Table 5: MP_KEY Key Types

Plain Text:

Key Data is exchanged in plain text between hosts (Host A and Host B), and the respective key parts (KeyA and KeyB) are used by each host to generate the derived key (d-key) by concatenating the two parts with the local key in front. That is,

Host A: d-keyA=(KeyA+KeyB)

Host B: d-keyB=(KeyB+KeyA)

Experimental:

This Key Type allows the use of other Key Data and can be used to validate other key exchange mechanisms for a possible future specification.

Multiple keys are only permitted in the DCCP-Request message of the handshake procedure for the first subflow. This allows the hosts to agree on a single Key Type to be used, as described in Section 3.3

It is possible that not all hosts will support all Key Types, and this specification does not recommend or enforce the announcement of any particular Key Type within the MP_KEY option as this could have security implications. However, at least Key Type 0 (Plain Text) **MUST** be supported for interoperability tests in implementations of MP-DCCP. If the Key Type cannot be agreed in the handshake procedure, the MP-DCCP connection **MUST** fall back to not using MP-DCCP, as indicated in Section 3.6.

3.2.5. MP_SEQ

DCCP [RFC4340] defines a packet sequencing scheme that continues to apply to the individual DCCP subflows within an MP-DCCP connection. However, for the operation of MP-DCPP, the order of packets within an MP-DCCP connection **MUST** be known before assigning packets to subflows to apply the received Multipath Options in the correct order or to recognize whether delayed Multipath Options are obsolete. Therefore, MP_SEQ is introduced and can also be used to reorder data packets on the receiver side.

```
            1          2          3           4          5
 01234567 89012345 67890123 45678901 23456789 01234567 89012345
+--------+--------+--------+--------+--------+--------+--------+
|00101110|00001001|00000100| Multipath Sequence Number
+--------+--------+--------+--------+--------+--------+--------+
                  |
+--------+--------+
 Type=46  Length=9 MP_OPT=4
```

Figure 12: Format of the MP_SEQ Suboption

The MP_SEQ suboption is used for end-to-end 48-bit datagram-based sequence numbers of an MP-DCCP connection. The initial data sequence number (IDSN) **SHOULD** be set randomly [RFC4086]. As with the standard DCCP sequence number, the data sequence number should not start at zero but at a random value to make blind session hijacking more difficult; see also Section 7.2 of [RFC4340].

The MP_SEQ number space is independent of the path individual sequence number space and **MUST** be sent with all DCCP-Data and DCCP-DataACK packets.

When the sequence number space is exhausted, the sequence number **MUST** be wrapped. [RFC7323] provides guidance on selecting an appropriately sized sequence number space according to the Maximum Segment Lifetime (MSL) of TCP. 64 bits is the recommended size for TCP to avoid the sequence number space going through within the segment lifetime. For DCCP, the MSL is the same as that of TCP as specified in Section 3.4 of [RFC4340]. Compared to TCP, the sequence number for DCCP is incremented per packet rather than per byte transmitted. For this reason, the 48 bits chosen in MP_SEQ are considered sufficiently large per the current globally routable maximum packet size (MPS) of 1500 bytes, which corresponds to roughly 375 pebibytes (PiBs) of data within the sequence number space.

3.2.6. MP_HMAC

MP-DCCP protects against some on-path attacker as further outlined in Section 4. Once an MP-DCCP connection has been established, the MP_HMAC option introduced here provides further protection based on the key material exchanged with MP_KEY when the connection is established.

```
            1          2          3           4
 01234567 89012345 67890123 45678901 23456789 01234567
+--------+--------+--------+--------+--------+--------+
|00101110|00010111|00000101| HMAC-SHA256 (20 bytes) ...
+--------+--------+--------+--------+--------+--------+
 Type=46  Length=23 MP_OPT=5
```

Figure 13: Format of the MP_HMAC Suboption

The MP_HMAC suboption is used to provide authentication for the MP_ADDADDR and MP_REMOVEADDR suboptions. In addition, it provides authentication for subflows joining an existing MP_DCCP connection, as described in the second and third step of the handshake of a subsequent subflow in Section 3.3. For this specification of MP-DCCP, the HMAC code is generated according to [RFC2104] in combination with the SHA-256 hash algorithm described in [RFC6234], with the output in big-endian format truncated to the leftmost 160 bits (20 bytes). It is possible that other versions of MP-DCCP will define other hash algorithms in the future.

The "Key" used for the HMAC computation is the derived key (d-keyA for Host A or d-KeyB for Host B) described in Section 3.2.4, while the HMAC "Message" for MP_JOIN, MP_ADDADDR, and MP_REMOVEADDR must be calculated in both hosts in order to protect the Multipath Option when sending and to validate the Multipath Option when receiving; it is a concatenation of:

- For MP_JOIN: The Nonces of the MP_JOIN messages for which authentication shall be performed. Depending on whether Host A or Host B performs the HMAC-SHA256 calculation, it is carried out as follows:
  - MP_HMAC(A) = HMAC-SHA256(Key=d-keyA, Msg=RA+RB)
  - MP_HMAC(B) = HMAC-SHA256(Key=d-keyB, Msg=RB+RA)

A usage example is shown in Figure 21.

- For MP_ADDADDR: The Address ID and Nonce with an associated IP address and a port, if defined; otherwise, 2 bytes of value 0. The IP address and port **MUST** be used in network byte order (NBO). Depending on whether Host A or Host B performs the HMAC-SHA256 calculation, it is carried out as follows:
  - MP_HMAC(A) = HMAC-SHA256(Key=d-keyA, Msg=Address ID+Nonce+NBO(IP)+NBO(Port))
  - MP_HMAC(B) = HMAC-SHA256(Key=d-keyB, Msg=Address ID+Nonce+NBO(IP)+NBO(Port))
- For MP_REMOVEADDR: Solely the Address ID. Depending on whether Host A or Host B performs the HMAC-SHA256 calculation, it is carried out as follows:
  - MP_HMAC(A) = HMAC-SHA256(Key=d-keyA, Msg=Address ID+Nonce)
  - MP_HMAC(B) = HMAC-SHA256(Key=d-keyB, Msg=Address ID+Nonce)

MP_JOIN, MP_ADDADDR, and MP_REMOVEADDR can coexist or be used multiple times within a single DCCP packet. All these Multipath Options require an individual MP_HMAC option. This ensures that the MP_HMAC is correctly associated. Otherwise, the receiver cannot validate multiple MP_JOIN, MP_ADDADDR, or MP_REMOVEADDR options. Therefore, an MP_HMAC **MUST** directly follow its associated Multipath Option. In the likely case of sending an MP_JOIN together with an MP_ADDADDR, this results in concatenating MP_JOIN + MP_HMAC_1 + MP_ADDADDR + MP_HMAC_2, whereas the first MP_HMAC_1 is associated with the MP_JOIN and the second MP_HMAC_2 is associated with the MP_ADDADDR suboption.

On the receiver side, the HMAC validation of the suboptions **MUST** be carried out according to the sending sequence in which the associated MP_HMAC follows a suboption. If the suboption cannot be validated by a receiving host because the HMAC validation fails (HMAC is wrong or missing), the subsequent handling depends on which suboption was being verified. If the suboption to be authenticated was either MP_ADDADDR or MP_REMOVEADDR, the receiving host **MUST** silently ignore it (see Sections 3.2.8 and 3.2.9). If the suboption to be authenticated was MP_JOIN, the subflow **MUST** be closed (see Section 3.6).

In the event that an MP_HMAC cannot be associated with a suboption, this MP_HMAC **MUST** be ignored, unless it is a single MP_HMAC that was sent in a DCCP-Ack corresponding to a DCCP response packet with MP_JOIN (see the penultimate arrow in Figure 21).

3.2.7. MP_RTT

The MP_RTT suboption is used to transmit RTT values and Age (represented in milliseconds) that belong to the path over which this information is transmitted. This information is useful for the receiving host to calculate the RTT difference between the subflows and to estimate whether missing data has been lost.

```
            1          2          3           4          5
 01234567 89012345 67890123 45678901 23456789 01234567 89012345
+--------+--------+--------+--------+--------+--------+--------+
|00101110|00001100|00000110|RTT Type| RTT
+--------+--------+--------+--------+--------+--------+--------+
         | Age                               |
+--------+--------+--------+--------+--------+
 Type=46  Length=12 MP_OPT=6
```

Figure 14: Format of the MP_RTT Suboption

The RTT and Age information is a 32-bit integer. This covers a period of approximately 1193 hours.

The Field RTT type indicates the type of RTT estimation, according to the following description:

Raw RTT (=0)
: Raw RTT value of the last Datagram round trip.

Min RTT (=1)
: Min RTT value over a given period.

Max RTT (=2)
: Max RTT value over a given period.

Smooth RTT (=3)
: Averaged RTT value over a given period.

Each CCID specifies the algorithms and period applied for their corresponding RTT estimations. The availability of the above-described types, to be used in the MP_RTT option, depends on the CCID implementation in place.

Age: The Age parameter defines the time difference between now -- the creation of the MP_RTT option -- and the conducted RTT measurement in milliseconds. If no previous measurement exists, e.g., when initialized, the value is 0.

An example of a flow showing the exchange of path individual RTT information is provided in Figure 15. RTT1 refers to the first path and RTT2 to the second path. The RTT values could be extracted from the sender's congestion control algorithm and are conveyed to the receiving host using the MP_RTT suboption. With the reception of RTT1 and RTT2, the receiver is able to calculate the path_delta that corresponds to the absolute difference of both values. In the case where the path individual RTTs are symmetric in the down-link and up-link directions and there is no jitter, packets with missing sequence number MP_SEQ, e.g., in a reordering process, can be assumed lost after path_delta/2.

```
MP-DCCP                    MP-DCCP
Sender                     Receiver
+--------+  MP_RTT(RTT1)  +-------------+
|   RTT1 |----------------|             |
|        |                | path_delta= |
|        |  MP_RTT(RTT2)  | |RTT1-RTT2| |
|   RTT2 |----------------|             |
+--------+                +-------------+
```

Figure 15: Exemplary Flow of MP_RTT Exchange and Usage

3.2.8. MP_ADDADDR

The MP_ADDADDR suboption announces additional addresses (and, optionally, port numbers) by which a host can be reached. This can be sent at any time during an existing MP-DCCP connection, when the sender wishes to enable multiple paths and/or when additional paths become available. Multiple instances of this suboption within a packet can simultaneously advertise new addresses.

The Length is variable depending on the address family (IPv4 or IPv6) and whether a port number is used. This field is in the range between 12 and 26 bytes.

The Nonce is a 32-bit random value that is generated locally for each MP_ADDADDR option and is used in the HMAC calculation process to prevent replay attacks.

The final 2 bytes optionally specify the DCCP port number to use, and their presence can be inferred from the length of the option. Although it is expected that the majority of use cases will use the same port pairs as used for the initial subflow (e.g., port 80 remains port 80 on all subflows, as does the ephemeral port at the client), there could be cases (such as port-based load balancing) where the explicit specification of a different port is required. If no port is specified, the receiving host **MUST** assume that any attempt to connect to the specified address uses the port already used by the subflow on which the MP_ADDADDR signal was sent.

Along with the MP_ADDADDR option, an MP_HMAC option **MUST** be sent for authentication. The truncated HMAC parameter present in this MP_HMAC option is the leftmost 20 bytes of an HMAC, negotiated and calculated as described in Section 3.2.6. Similar to MP_JOIN, the key for the HMAC algorithm will be d-KeyA when the message is transmitted by Host A and d-KeyB when transmitted by Host B. These are the keys that were exchanged and selected in the original MP_KEY handshake. The message for the HMAC is the Address ID, Nonce, IP address, and port number that precede the HMAC in the MP_ADDADDR option. If the port number is not present in the MP_ADDADDR option, the HMAC message will include 2 bytes of value zero. The rationale for the HMAC is to prevent unauthorized entities from injecting MP_ADDADDR signals in an attempt to hijack a connection. Additionally, note that the presence of this HMAC prevents the address from being changed in flight unless the key is known by an intermediary. If a host receives an MP_ADDADDR option for which it cannot validate the HMAC, it **MUST** silently ignore the option.

The presence of an MP_SEQ (Section 3.2.5) **MUST** be ensured in a DCCP datagram in which MP_ADDADDR is sent, as described in Section 3.2.1.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+-------+-------+---------------+
|0 0 1 0 1 1 1 0|      var      |0 0 0 0 0 1 1 1|  Address ID   |
+---------------+---------------+-------+-------+---------------+
|                             Nonce                             |
+-------------------------------+-------------------------------+
|          Address (IPv4 - 4 bytes / IPv6 - 16 bytes)           |
+-------------------------------+-------------------------------+
|   Port (2 bytes, optional)    | + MP_HMAC option
+-------------------------------+
     Type=46         Length         MP_OPT=7
```

Figure 16: Format of the MP_ADDADDR Suboption

Each address has an Address ID that could be used for uniquely identifying the address within a connection for address removal. Each host maintains a list of unique Address IDs, and it manages these as it wishes. The Address ID is also used to identify MP_JOIN options (see Section 3.2.2) relating to the same address, even when address translators are in use. The Address ID **MUST** uniquely identify the address for the sender of the option (within the scope of the connection); the mechanism for allocating such IDs is implementation specific.

All Address IDs learned via either MP_JOIN or MP_ADDADDR can be stored by the receiver in a data structure that gathers all the Address-ID-to-address mappings for a connection (identified by a CI pair). In this way, there is a stored mapping between the Address ID, the observed source address, and the CI pair for future processing of control information for a connection. Note that an implementation **MAY** discard incoming address advertisements. Reasons for this are, for example:

- to avoid the required mapping state, or

- because advertised addresses are of no use to it.

Possible scenarios in which this applies are the lack of resources to store a mapping or when IPv6 addresses are advertised even though the host only supports IPv4. Therefore, a host **MUST** treat address announcements as soft state. However, a sender **MAY** choose to update the announcements periodically to overcome temporary limitations.

A host **MAY** advertise private addresses, e.g., because there is a NAT on the path. It is desirable to allow this as there could be cases where both hosts have additional interfaces on the same private network. The advertisement of broadcast or multicast IP addresses **MUST** be ignored by the recipient of this option, as it is not permitted according to the unicast principle of the basic DCCP.

The MP_JOIN handshake used to create a new subflow (Section 3.2.2) provides mechanisms to minimize security risks. The MP_JOIN message contains a 32-bit CI that uniquely identifies a connection to the receiving host. If the CI is unknown, the host **MUST** send a DCCP-Reset.

Further security considerations around the issue of MP_ADDADDR messages that accidentally misdirect, or maliciously direct, new MP_JOIN attempts are discussed in Section 4. If a sending host of an MP_ADDADDR knows that no incoming subflows can be established at a particular address, an MP_ADDADDR **MUST NOT** announce that address unless the sending host has new knowledge about the possibility to do so. This information can be obtained from local firewall or routing settings, knowledge about availability of an external NAT or a firewall, or connectivity checks performed by the host/application.

The reception of an MP_ADDADDR message is acknowledged using MP_CONFIRM (Section 3.2.1). This ensures a reliable exchange of address information.

A host that receives an MP_ADDADDR but finds that the IP address and port number is unsuccessful at connection setup **SHOULD NOT** perform further connection attempts to this address/port combination for this connection to save resources. However, if a sender wishes to trigger a new incoming connection attempt on a previously advertised address/port combination, they can refresh the MP_ADDADDR information by sending the option again.

A host **MAY** send an MP_ADDADDR message with an already-assigned Address ID using the IP address previously assigned to this Address ID. The new MP_ADDADDR could have the same port number or a different port number. The receiver **MUST** silently ignore the MP_ADDADDR if the IP address is not the same as that previously assigned to this Address ID. A host wishing to replace an existing Address ID **MUST** first remove the existing one (Section 3.2.9).

3.2.9. MP_REMOVEADDR

If, during the lifetime of an MP-DCCP connection, a previously announced address becomes invalid (e.g., if an interface disappears), the affected host **SHOULD** announce this. The peer can remove a previously added address with an Address ID from a connection using the Remove Address (MP_REMOVEADDR) suboption. This will terminate any subflows currently using that address.

MP_REMOVEADDR is only used to close already-established subflows that have an invalid address. Functional flows with a valid address **MUST** be closed with a DCCP Close exchange (as with regular DCCP) instead of using MP_REMOVEADDR. For more information see Section 3.5.

The Nonce is a 32-bit random value that is generated locally for each MP_REMOVEADDR option and is used in the HMAC calculation process to prevent replay attacks.

Along with the MP_REMOVEADDR suboption, an MP_HMAC option **MUST** be sent for authentication. The truncated HMAC parameter present in this MP_HMAC option is the leftmost 20 bytes of an HMAC, negotiated and calculated as described in Section 3.2.6. Similar to MP_JOIN, the key for the HMAC algorithm will be d-KeyA when the message is transmitted by Host A and d-KeyB when transmitted by Host B. These are the keys that were exchanged and selected in the original MP_KEY handshake. The message for the HMAC is the Address ID.

The rationale for using an HMAC is to prevent unauthorized entities from injecting MP_REMOVEADDR signals in an attempt to hijack a connection. Additionally, note that the presence of this HMAC prevents the address from being modified in flight unless the key is known by an intermediary. If a host receives an MP_REMOVEADDR option for which it cannot validate the HMAC, it **MUST** silently ignore the option.

A receiver **MUST** include an MP_SEQ (Section 3.2.5) in a DCCP datagram that sends an MP_REMOVEADDR. Further details are given in Section 3.2.1.

The reception of an MP_REMOVEADDR message is acknowledged using MP_CONFIRM (Section 3.2.1). This ensures a reliable exchange of address information. To avoid inconsistent states, the sender releases the Address ID only after MP_REMOVEADDR has been confirmed.

The sending and receiving of this message **SHOULD** trigger the closing procedure described in [RFC4340] between the client and the server on the affected subflow(s), if possible. This helps remove middlebox state before removing any local state.

Address removal is done by the Address ID to allow the use of NATs and other middleboxes that rewrite source addresses. If there is no address at the requested Address ID, the receiver will silently ignore the request.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
|0 0 1 0 1 1 1 0|0 0 0 0 0 1 0 0|0 0 0 0 1 0 0 0|   Address ID  |
+---------------+---------------+---------------+---------------+
|                             Nonce                             |
+-------------------------------+-------------------------------+
     Type=46        Length=8         MP_OPT=8

-> followed by the MP_HMAC option
```

Figure 17: Format of the MP_REMOVEADDR Suboption

3.2.10. MP_PRIO

The path priority signaled with the MP_PRIO option provides hints for the packet scheduler when making decisions about which path to use for payload traffic. When a single specific path from the set of available paths is treated with higher priority compared to the others when making scheduling decisions for payload traffic, a host can signal such change in priority to the peer. This could be used when there are different costs for using different paths (e.g., Wi-Fi is free while cellular has a limit on volume, and 5G has higher energy consumption). The priority of a path could also change, for example, when a mobile host runs out of battery, and the usage of only a single path may be the preferred choice of the user.

The MP_PRIO suboption, shown below, can be used to set a priority value for the subflow over which the suboption is received.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
|0 0 1 0 1 1 1 0|0 0 0 0 0 1 0 0|0 0 0 0 1 0 0 1|(resvd)| prio  |
+---------------+---------------+---------------+---------------+
    Type=46         Length=4        MP_OPT=9
```

Figure 18: Format of the MP_PRIO Suboption

The following values are available for the Prio field:

- 0: Do not use. The path is not available.
- 1: Standby: Do not use this path for traffic scheduling if another path (secondary or primary) is available. The path will only be used if other secondary or primary paths are not established.
- 2: Secondary: Do not use this path for traffic scheduling if the other paths are good enough. The path will be used occasionally for increasing the available capacity temporarily, e.g., when primary paths are congested or are not available. This is the recommended setting for paths that have costs or data caps as these paths will be used less frequently then primary paths.
- 3 - 15: Primary: The path can be used for packet scheduling decisions. The priority number indicates the relative priority of one path over the other for primary paths. Higher numbers indicate higher priority. The peer should consider sending traffic first over higher priority paths. This is the recommended setting for paths that do not have a cost or data caps associated with them as these paths will be frequently used.

Example use cases include:

1. Setting the Wi-Fi path to Primary and Cellular path to Secondary. In this case, Wi-Fi will be used and Cellular will be used only if the Wi-Fi path is congested or not available. Such setting results in using the Cellular path only temporally, if more capacity is needed than the Wi-Fi path can provide, indicating a clear priority of the Wi-Fi path over the Cellular due to, e.g., cost reasons.
2. Setting the Wi-Fi path to Primary and Cellular path to Standby. In this case, Wi-Fi will be used and Cellular will be used only if the Wi-Fi path is not available.
3. Setting the Wi-Fi path to Primary and Cellular path to Primary. In this case, both paths can be used when making packet scheduling decisions.

If not specified, the default behavior is to always use a path for packet scheduling decisions (MP_PRIO=3), when the path has been established and added to an existing MP-DCCP connection. At least one path ought to have an MP_PRIO value greater than or equal to one for it to be allowed to send on the connection. It is **RECOMMENDED** to update at least one path to a non-zero MP_PRIO value when an MP-DCCP connection enters a state where all paths remain with an MP_PRIO value of zero. This helps an MP-DCCP connection to schedule when the multipath scheduler strictly respects MP_PRIO value 0. To ensure reliable transmission, the MP_PRIO suboption **MUST** be acknowledged via an MP_CONFIRM (see Table 4).

The relative ratio of the primary path values 3-15 depends on the path usage strategy, which is described in more detail in Section 3.11. In the case of path mobility (Section 3.11.1), only one path can be used at a time and **MUST** have the highest available priority value. That also includes the prio numbers 1 and 2. In the other case of concurrent path usage (Section 3.11.2), the definition is up to the multipath scheduler logic.

An MP_SEQ (Section 3.2.5) **MUST** be present in a DCCP datagram in which the MP_PRIO suboption is sent. Further details are given in Section 3.2.1.

3.2.11. MP_CLOSE

The mechanism available in DCCP [RFC4340] for closing a connection cannot give an indication for closing an MP-DCCP connection, which typically contains several DCCP subflows; therefore, one cannot conclude from the closing of a subflow to the closing of an MP-DCCP connection. This is solved by introducing MP_CLOSE.

```
           1          2          3
 01234567 89012345 67890123 45678901 23456789
+--------+--------+--------+--------+--------+
|00101110|  var   |00001010| Key Data ...
+--------+--------+--------+--------+--------+
 Type=46   Length  MP_OPT=10
```

Figure 19: Format of the MP_CLOSE Suboption

An MP-DCCP connection can be gracefully closed by sending an MP_CLOSE to the peer host. On all subflows, the regular termination procedure described in [RFC4340] **MUST** be initiated using MP_CLOSE in the initial packet (either a DCCP-CloseReq or a DCCP-Close). When a DCCP-CloseReq is used, the following DCCP-Close **MUST** also carry the MP_CLOSE to avoid keeping a state in the sender of the DCCP-CloseReq. At the initiator of the DCCP-CloseReq, all sockets, including the MP-DCCP connection socket, transition to CLOSEREQ state. To protect from unauthorized shutdown of a multipath connection, the selected Key Data of the peer host **MUST** be included in the MP_CLOSE option during the handshake procedure and **MUST** be validated by the peer host. Please note that the Key Data sent in DCCP-CloseReq will not be the same as the Key Data sent in DCCP-Close as these originate from different ends of the connection.

On reception of the first DCCP-CloseReq carrying an MP_CLOSE with valid Key Data, or due to a local decision, all subflows transition to the CLOSING state before transmitting a DCCP-Close carrying MP_CLOSE. The MP-DCCP connection socket on the host sending the DCCP-Close reflects the state of the initial subflow during the handshake with MP_KEY option. If the initial subflow no longer exists, the state moves immediately to CLOSED.

Upon reception of the first DCCP-Close carrying an MP_CLOSE with valid Key Data at the peer host, all subflows, as well as the MP-DCCP connection socket, move to the CLOSED state. After this, a DCCP-Reset with Reset Code 1 **MUST** be sent on any subflow in response to a received DCCP-Close containing a valid MP_CLOSE option.

When the MP-DCCP connection socket is in CLOSEREQ or CLOSED state, new subflow requests using MP_JOIN **MUST** be ignored.

Contrary to an MP_FAST_CLOSE (Section 3.2.3), no single-sided abrupt termination is applied.

3.2.12. Experimental Multipath Option MP_EXP for Private Use

This section reserves a Multipath Option to define and specify any experimental additional feature for improving and optimizing the MP-DCCP protocol. This option could be applicable to specific environments or scenarios according to potential new requirements and is meant for private use only. MP_OPT Feature Number 11 is specified with an exemplary description as below:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
|0 0 1 0 1 1 1 0|      var      |0 0 0 0 1 0 1 1|     Data      |
+---------------+---------------+---------------+---------------+
|   ...
+---------------------------------------------------------------+
     Type=46          Length         MP_OPT=11
```

Figure 20: Format of the MP_EXP Suboption

The Data field can carry any data according to the foreseen use by the experimenters with a maximum Length of 252 bytes.

3.3. MP-DCCP Handshake Procedure

An example MP-DCCP handshake procedure is shown in Figure 21.

```
          Host  A                                        Host  B
------------------------                              ----------
Address A1    Address A2                              Address B1
----------    ----------                              ----------
     |             |                                       |
     |           DCCP-Request + Change R (MP_CAPABLE,...)  |
     |----- MP_KEY(CI-A + KeyA(1), KeyA(2),...) ---------->|
     |<------------------- MP_KEY(CI-B + KeyB) ------------|
     |       DCCP-Response +  Confirm L (MP_CAPABLE, ...)  |
     |             |                                       |
     |   DCCP-Ack  |                                       |
     |---------------------------------------------------->|
     |<----------------------------------------------------|
     |   DCCP-Ack  |                                       |
     |             |                                       |
     |             |DCCP-Request + Change R(MP_CAPABLE,...)|
     |             |--- MP_JOIN(CI-B,RA) ----------------->|
     |             |<------MP_JOIN(CI-A,RB) + MP_HMAC(B)---|
     |             |DCCP-Response+Confirm L(MP_CAPABLE,...)|
     |             |                                       |
     |             |DCCP-Ack                               |
     |             |-------- MP_HMAC(A) ------------------>|
     |             |<--------------------------------------|
     |             |DCCP-Ack                               |
```

Figure 21: Example MP-DCCP Handshake

The basic initial handshake for the first subflow is as follows:

1. Host A sends a DCCP-Request with the Multipath Capable Feature change request and the MP_KEY option with a Host-specific CI-A and a KeyA for each of the supported Key Types as described in Section 3.2.4. CI-A is a unique identifier during the lifetime of an MP-DCCP connection.
2. Host B sends a DCCP-Response with a Confirm feature for MP-Capable and the MP_Key option with a unique Host-specific CI-B and a single Host-specific KeyB. The type of the key is chosen from the list of supported types from the previous request.
3. Host A sends a DCCP-Ack to confirm the proper key exchange.
4. Host B sends a DCCP-Ack to complete the handshake and set both connection ends to the OPEN state.

It should be noted that DCCP is protected against corruption of DCCP header data (Section 9 of [RFC4340]), so no additional mechanisms beyond the general confirmation are required to ensure that the header data has been properly received.

Host A waits for the final DCCP-Ack from Host B before starting any establishment of additional subflow connections.

The handshake for subsequent subflows, based on a successful initial handshake, is as follows:

1. Host A sends a DCCP-Request with the Multipath Capable Feature change request and the MP_JOIN option with Host B's CI-B, obtained during the initial handshake. Additionally, a random Nonce RA is transmitted with the MP_JOIN.
2. Host B computes the HMAC of the DCCP-Request and sends a DCCP-Response with a Confirm feature option for MP-Capable and the MP_JOIN option with the CI-A and a random Nonce RB together with the computed MP_HMAC. As specified in Section 3.2.6, the HMAC is calculated by taking the leftmost 20 bytes from the SHA-256 hash of an HMAC code that is created by using the Nonce received with MP_JOIN(A) and the local Nonce RB as the Message and the derived key as the Key, as described in Section 3.2.4:

   MP_HMAC(B) = HMAC-SHA256(Key=d-keyB, Msg=RB+RA)
3. Host A sends a DCCP-Ack with the HMAC computed for the DCCP-Response. As specified in Section 3.2.6, the HMAC is calculated by taking the leftmost 20 bytes from the SHA-256 hash of an HMAC code created by using the local Nonce RA and the Nonce received with MP_JOIN(B) as message and the derived key described in Section 3.2.4 as key:

   MP_HMAC(A) = HMAC-SHA256(Key=d-keyA, Msg=RA+RB)
4. Host B sends a DCCP-Ack to confirm the HMAC and to conclude the handshake.

3.4. Address Knowledge Exchange

3.4.1. Advertising a New Path (MP_ADDADDR)

When a host (Host A) wants to advertise the availability of a new path, it should use the MP_ADDADDR option (Section 3.2.8) as shown in the example in Figure 22. The MP_ADDADDR option passed in the DCCP-Data contains the following parameters:

- an identifier (id 2) for the new IP address, which is used as a reference in subsequent control exchanges
- a Nonce value to prevent replay attacks
- the IP address of the new path (A2_IP)
- a pair of bytes specifying the port number associated with this IP address. The value of 00 here indicates that the port number is the same as that used for the initial subflow address A1_IP.

According to Section 3.2.8, the following options are required in a packet carrying MP_ADDADDR:

- the leftmost 20 bytes of the HMAC(A) generated during the initial handshake procedure described in Sections 3.3 and 3.2.6
- the MP_SEQ option with the sequence number (seqno 12) for this message, according to Section 3.2.5

Host B acknowledges receipt of the MP_ADDADDR message with a DCCP-Ack containing the MP_CONFIRM option. The parameters supplied in this response are as follows:

- an MP_CONFIRM containing the MP_SEQ number (seqno 12) of the packet carrying the option that we are confirming together with the MP_ADDADDR option
- the leftmost 20 bytes of the HMAC(B) generated during the initial handshake procedure (Section 3.3)

```
          Host A                                         Host B
------------------------                              -----------
Address A1    Address A2                               Address B1
----------    ----------                              -----------
     |              |                                       |
     |   DCCP-Data +  MP_ADDADDR(id 2, Nonce, A2_IP, 00) + |
     |------- MP_HMAC(A) + MP_SEQ(seqno 12) --------------->|
     |              |                                       |
     |   DCCP-Ack + MP_HMAC(B) +                            |
     |<----- MP_CONFIRM(seqno 12, MP_ADDADDR) -------------|
```

Figure 22: Example MP_ADDADDR Procedure

3.4.2. Removing a Path (MP_REMOVEADDR)

When a host (Host A) wants to indicate that a path is no longer available, it should use the MP_REMOVEADDR option (Section 3.2.9) as shown in the example in Figure 23. The MP_REMOVEADDR option passed in the DCCP-Data contains the following parameters:

- an identifier (id 2) for the IP address to remove (A2_IP) and that was specified in a previous MP_ADDADDR message
- a Nonce value to prevent replay attacks

According to Section 3.2.9, the following options are required in a packet carrying MP_REMOVEADDR:

- the leftmost 20 bytes of the HMAC(A) generated during the initial handshake procedure described in Sections 3.3 and 3.2.6
- the MP_SEQ option with the sequence number (seqno 33) for this message, according to Section 3.2.5

Host B acknowledges receipt of the MP_REMOVEADDR message with a DCCP-Ack containing the MP_CONFIRM option. The parameters supplied in this response are as follows:

- an MP_CONFIRM containing the MP_SEQ number (seqno 33) of the packet carrying the option that we are confirming, together with the MP_REMOVEADDR option
- the leftmost 20 bytes of the HMAC(B) generated during the initial handshake procedure (Section 3.3)

```
          Host A                                         Host B
------------------------                               -----------
Address A1    Address A2                                Address B1
----------    ----------                               -----------
     |             |                                        |
     |   DCCP-Data +  MP_REMOVEADDR(id 2, Nonce) +          |
     |------- MP_HMAC(A) + MP_SEQ(seqno 33) --------------->|
     |             |                                        |
     |   DCCP-Ack + MP_HMAC(B) +                            |
     |<----- MP_CONFIRM(seqno 33, MP_REMOVEADDR) ----------|
```

Figure 23: Example MP_REMOVEADDR Procedure

3.5. Closing an MP-DCCP Connection

When a host wants to close an existing subflow but not the whole MP-DCCP connection, it **MUST** initiate the regular DCCP connection termination procedure as described in Section 5.6 of [RFC4340], i.e., it sends a DCCP-Close/DCCP-Reset on the subflow. This may be preceded by a DCCP-CloseReq. In the event of an irregular termination of a subflow, e.g., during subflow establishment, it **MUST** use an appropriate DCCP-Reset Code as specified by IANA [DCCP-PARAMETERS] for DCCP operations. This could be, for example, sending Reset Code 5 (Option Error) when an MP-DCCP option provides invalid data or Reset Code 9 (Too Busy) when the maximum number of maintainable paths is reached. Note that receiving a Reset Code 9 for secondary subflows **MUST NOT** impact already existing active subflows. If necessary, these subflows are terminated in a subsequent step using the procedures described in this section.

A host terminates an MP-DCCP connection using the DCCP connection termination specified in Section 5.5 of [RFC4340] on each subflow with the first packet on each subflow carrying MP_CLOSE (see Section 3.2.11).

```
Host A                                   Host B
------                                   ------
                                  <-     Optional DCCP-CloseReq +
                                         MP_CLOSE [A's key]
                                         [on all subflows]
DCCP-Close + MP_CLOSE             ->
[B's key] [on all subflows]
                                  <-     DCCP-Reset
                                         [on all subflows]
```

Additionally, an MP-DCCP connection may be closed abruptly using the "fast close" procedure described in Section 3.2.3, where a DCCP-Reset is sent on all subflows, each carrying the MP_FAST_CLOSE option.

```
Host A                                    Host B
------                                    ------
DCCP-Reset + MP_FAST_CLOSE       ->
[B's key] [on all subflows]
                                 <-       DCCP-Reset
                                          [on all subflows]
```

3.6. Fallback

When a subflow fails to operate following the intended behavior of the MP-DCCP, it is necessary to proceed with a fallback. This may be either falling back to regular DCCP [RFC4340] or removing a problematic subflow. The main reasons for a subflow failing include: no MP support at the peer host, failure to negotiate the protocol version, loss of Multipath Options, faulty/non-supported MP-DCCP options, or modification of payload data.

At the start of an MP-DCCP connection, the handshake ensures the exchange of the MP-DCCP feature and options and thus ensures that the path is fully MP-DCCP capable. If during the handshake procedure it appears that DCCP-Request or DCCP-Response messages do not carry the Multipath Capable Feature, the MP-DCCP connection will not be established and the handshake **SHOULD** fall back to regular DCCP. If this is not possible, the connection **MUST** be closed.

If the endpoints fail to agree on the protocol version to use during the Multipath Capable Feature negotiation, the connection **MUST** either be closed or fall back to regular DCCP. This is described in Section 3.1. The protocol version negotiation distinguishes between negotiation for the initial connection establishment and the addition of subsequent subflows. If protocol version negotiation is not successful during the initial connection establishment, the MP-DCCP connection will fall back to regular DCCP.

The fallback procedure for regular DCCP **MUST** also be applied if the MP_KEY (Section 3.2.4) Key Type cannot be negotiated.

If a subflow attempts to join an existing MP-DCCP connection but MP-DCCP options or the Multipath Capable Feature are not present or are faulty in the handshake procedure, that subflow **MUST** be closed. This is the case especially if a different MP_CAPABLE version than the originally negotiated version is used. Reception of a non-verifiable MP_HMAC (Section 3.2.6) or an invalid CI used in MP_JOIN (Section 3.2.2) during flow establishment **MUST** cause the subflow to be closed.

The subflow closing procedure **MUST** also be applied if a final ACK carrying MP_KEY with the wrong KeyA/KeyB is received or the MP_KEY option is malformed.

Another relevant case is when payload data is modified by middleboxes. DCCP uses a checksum to protect the data, as described in Section 9 of [RFC4340]. A checksum will fail if the data has been changed in any way. All data from the start of the segment that failed the checksum onwards cannot be considered trustworthy. As defined by DCCP, if the checksum fails, the receiving endpoint **MUST** drop the application data and report that data as dropped due to corruption using a Data Dropped option (Drop Code 3, Corrupt). If data is dropped due to corruption for an MP-DCCP connection, the affected subflow **MAY** be closed. The same procedure applies if the Multipath Option is unknown.

3.7. State Diagram

The MP-DCCP per subflow state transitions follow the state transitions defined for DCCP in [RFC4340] to a large extent, with some modifications due to the MP-DCCP 4-way handshake and fast close procedures. The state diagram below shows the most common state transitions. The diagram is illustrative. For example, there are arcs (not shown) from several additional states to TIMEWAIT, contingent on the receipt of a valid DCCP-Reset.

When the state moves from CLOSED to OPEN during the 4-way handshake, the transitioned states remain the same as for DCCP, but it is no longer possible to transmit application data while in the REQUEST state. The fast close procedure can be triggered by either the client or the server and results in the transmission of a Reset packet. The fast close procedure moves the state of the Client and Server directly to TIMEWAIT and CLOSED, respectively.

```
+----------------------------+    +------------------------------+
|                            v    v                              |
|                         +----------+                           |
|           +-------------+  CLOSED   +-------------+             |
|           | passive     +----------+  active      |             |
|           |  open                      open       |             |
|           |                          snd Request |             |
|           v                                      v             |
|     +-----------+                          +----------+       |
|     |  LISTEN   |                          | REQUEST  |       |
|     +-----+-----+                          +----+-----+       |
|           | rcv Request             rcv Response |             |
|           | snd Response             snd Ack     |             |
|           v                                      v             |
|     +-----------+                          +----------+       |
|     |  RESPOND  |                          | PARTOPEN |       |
|     +-----+-----+                          +----+-----+       |
|           | rcv Ack             rcv Ack/DataAck  |             |
|           | snd Ack                              |             |
|           |             +-----------+            |             |
|           +------------>|   OPEN    |<-----------+             |
|                         +--+-+-+-+--+                          |
|        server active close | | | |   active close             |
|            snd CloseReq    | | | | or rcv CloseReq             |
|                            | | | |    snd Close                |
|                            | | | |                             |
|     +-----------+          | | | |            +----------+     |
|     | CLOSEREQ  |<---------+ | | +----------->| CLOSING  |     |
|     +-----+-----+            | |              +----+-----+     |
|           | rcv Close        | |         rcv Reset |           |
|           | snd Reset        | |                   |           |
|           |                  | | active FastClose  |           |
|<----------+        rcv Close | | or rcv FastClose  v           |
|   or server active FastClose | | snd Reset    +----+-----+     |
|      or server rcv FastClose | +------------->| TIMEWAIT |     |
|                    snd Reset |                +----+-----+     |
+------------------------------+                     |           |
                                                     +-----------+
                                                2MSL timer expires
```

Figure 24: Most Common State Transitions of an MP-DCCP Subflow

3.8. Congestion Control Considerations

Senders **MUST** manage per-path congestion status and avoid sending more data on a given path than congestion control allows for each path.

3.9. Maximum Packet Size Considerations

A DCCP implementation maintains the maximum packet size (MPS) during operation of a DCCP session. This procedure is specified for single-path DCCP in Section 14 of [RFC4340]. Without any restrictions, this is adopted for MP-DCCP operations, in particular the Path MTU (PMTU) measurement and the Sender Behavior. The DCCP application interface **SHOULD** allow the application to discover the current MPS. This reflects the current largest size supported for the data stream that can be used across the set of all active MP-DCCP subflows.

3.10. Maximum Number of Subflow Considerations

MP-DCCP does not support any explicit procedure to negotiate the maximum number of subflows between endpoints. However, in practical scenarios, there will be resource limitations on the host or use cases that do not benefit from additional subflows.

It is **RECOMMENDED** to limit the number of subflows in implementations and to reject incoming subflow requests with a DCCP-Reset using the Reset Code "too busy" according to [RFC4340] if the resource limit is exceeded or it is known that the multipath connection will not benefit from further subflows. Likewise, it is **RECOMMENDED** that the host that wants to create the subflows considers the available resources and possible gains.

To avoid further inefficiencies with subflows due to short-lived connections, it **MAY** be useful to delay the start of additional subflows. The decision on the initial number of subflows can be based on the occupancy of the socket buffer and/or the timing.

While in the socket-buffer-based approach the number of initial subflows can be derived by opening new subflows until their initial windows cover the amount of buffered application data, the timing-based approach delays the start of additional subflows based on a certain time period, load, or knowledge of traffic and path properties. The delay-based approach also provides resilience for low-bandwidth but long-lived applications. All this could also be supported by advanced APIs that signal application traffic requests to the MP-DCCP.

3.11. Path Usage Strategies

MP-DCCP can be configured to realize one of several strategies for path usage via selecting one DCCP subflow out of the multiple DCCP subflows within an MP-DCCP connection for data transmission. This can be a dynamic process further facilitated by the means of DCCP and MP-DCCP-defined options such as path preference using MP-PRIO; adding or removing DCCP subflows using MP_REMOVEADDR, MP_ADDADDR, or DCCP-Close/DCCP-Reset; and path metrics such as packet loss rate, congestion window (CWND), or RTT provided by the congestion control algorithm. Selecting an appropriate method can allow MP-DCCP to realize different path utilization strategies that make MP-DCCP suitable for end-to-end implementation over the Internet or in controlled environments such as Hybrid Access or 5G ATSSS.

3.11.1. Path Mobility

The path mobility strategy provides the use of a single path with a seamless handover function to continue the connection when the currently used path is deemed unsuitable for service delivery. Some of the DCCP subflows of an MP-DCCP connection might become inactive due to either the occurrence of certain error conditions (e.g., DCCP timeout, packet loss threshold, RTT threshold, and closed/removed) or adjustments from the MP-DCCP user. When there is outbound data to send and the primary path becomes inactive (e.g., due to failures) or deprioritized, the MP-DCCP endpoint **SHOULD** try to send the data through an alternate path with a different source or destination address (depending on the point of failure), if one exists. This process **SHOULD** respect the path priority configured by the MP_PRIO suboption; otherwise, if the path priority is not available, pick the most divergent source-destination pair from the originally used source-destination pair.

> Note: Rules for picking the most appropriate source-destination pair are an implementation decision and are not specified within this document. Path mobility is supported in the current Linux reference implementation [MP-DCCP.Site].

3.11.2. Concurrent Path Usage

Different from a path mobility strategy, the selection between MP-DCCP subflows is a per-packet decision that is a part of the multipath scheduling process. This method would allow multiple subflows to be simultaneously used to aggregate the path resources to obtain higher connection throughput.

In this scenario, the selection of congestion control, per-packet scheduling, and a potential reordering method determines a concurrent path utilization strategy and result in a particular transport characteristic. A concurrent path usage method uses a scheduling design that could seek to maximize reliability, maximize throughput, minimize latency, etc.

Concurrent path usage over the Internet can have implications. When an MP-DCCP connection uses two or more paths, there is no guarantee that these paths are fully disjoint. When two (or more) subflows share the same bottleneck, using a standard congestion control algorithm could result in an unfair distribution of the capacity with the multipath connection using more capacity than competing single-path connections.

Multipath TCP uses the coupled congestion control Linked Increases Algorithm (LIA) specified in an experimental specification [RFC6356] to solve this problem. This scheme could also be specified for MP-DCCP. The same applies to other coupled congestion control algorithms that have been proposed for Multipath TCP such as the Opportunistic Linked Increases Algorithm [OLIA].

The specification of scheduling for concurrent multipath and related congestion control algorithms and reordering methods for use in the general Internet are outside the scope of this document. If, and when, the IETF specifies a method for concurrent usage of multiple paths for the general Internet, the framework specified in this document could be used to provide an IETF-recommended method for MP-DCCP.

4. Security Considerations

Similar to DCCP, MP-DCCP does not provide cryptographic security guarantees inherently. Thus, if applications need cryptographic security (integrity, authentication, confidentiality, access control, and anti-replay protection), the use of IPsec, DTLS over DCCP [RFC5238], or other end-to-end security is recommended; the Secure Real-time Transport Protocol (SRTP) [RFC3711] is one candidate protocol for authentication. Integrity would be provided if using SRTP together with the encryption of header extensions described in [RFC6904].

DCCP [RFC4340] provides protection against hijacking and limits the potential impact of some denial-of-service attacks, but DCCP provides no inherent protection against an on-path attacker snooping on data packets. Regarding the security of MP-DCCP compared to regular DCCP, no additional risks should be introduced. The security objectives for MP-DCCP are:

- Provide assurance that the parties involved in an MP-DCCP handshake procedure are identical to those in the original DCCP connection.
- Before a path is used, verify that the new advertised path is valid for receiving traffic.
- Provide replay protection, i.e., ensure that a request to add/remove a subflow is 'fresh'.
- Allow a party to limit the number of subflows that it allows.

To achieve these goals, MP-DCCP includes a hash-based handshake algorithm documented in Sections 3.2.4, 3.2.6, and 3.3. The security of the MP-DCCP connection depends on the use of keys that are shared once at the start of the first subflow and are never sent again over the network. Depending on the security requirements, different Key Types can be negotiated in the handshake procedure or must follow the fallback scenario described in Section 3.6. If there are security requirements that go beyond the capabilities of Key Type 0, then it is **RECOMMENDED** that Key Type 0 not be enabled to avoid downgrade attacks that result in the key being exchanged as plain text. To ease demultiplexing while not revealing cryptographic material, subsequent subflows use the initially exchanged CI information. The keys exchanged once at the beginning are concatenated and used as keys for creating HMACs used on subflow setup, in order to verify that the parties in the handshake of subsequent subflows are the same as in the original connection setup. This also provides verification that the peer can receive traffic at this new address. Replay attacks would still be possible when only keys are used; therefore, the handshakes use single-use random numbers (Nonces) for both parties -- this ensures that the HMAC will never be the same on two handshakes. Guidance on generating random numbers suitable for use as keys is given in [RFC4086]. During normal operation, regular DCCP protection mechanisms (such as the header checksum to protect DCCP headers against corruption) is designed to provide the same level of protection against attacks on individual DCCP subflows as exists for regular DCCP.

As discussed in Section 3.2.8, a host may advertise its private addresses, but these might point to different hosts in the receiver's network. The MP_JOIN handshake (Section 3.2.2) is designed to ensure that this does not set up a subflow to the incorrect host. However, it could still create unwanted DCCP handshake traffic. This feature of MP-DCCP could be a target for denial-of-service exploits, with malicious participants in MP-DCCP connections encouraging the recipient to target other hosts in the network. Therefore, implementations should consider heuristics at both the sender and receiver to reduce the impact of this.

As described in Section 3.9, an MPS is maintained for an MP-DCCP connection. If MP-DCCP exposes a minimum MPS across all paths, any change to one path impacts the sender for all paths. To mitigate attacks that seek to force a low MPS, MP-DCCP could detect an attempt to reduce the MPS to less than a minimum MPS and then stop using these paths.

5. Interactions with Middleboxes

Issues from interaction with on-path middleboxes such as NATs, firewalls, proxies, IDSs, and others have to be considered for all extensions to standard protocols; otherwise, unexpected reactions of middleboxes may hinder its deployment. DCCP already provides means to mitigate the potential impact of middleboxes, in comparison to TCP (see Section 16 of [RFC4340]). When both hosts are located behind a NAT or firewall entity, specific measures have to be applied such as the simultaneous-open technique specified in [RFC5596] that updates the asymmetric connection-establishment procedures for DCCP. Further standardized technologies addressing middleboxes operating as NATs are provided in [RFC5597].

[RFC6773] specifies UDP encapsulation for NAT traversal of DCCP sessions, similar to other UDP encapsulations such as the Stream Control Transmission Protocol (SCTP) [RFC6951]. Future specifications by the IETF could specify other methods for DCCP encapsulation.

The security impact of MP-DCCP-aware middleboxes is discussed in Section 4.

6. Implementation

The approach described above has been implemented in open source across different testbeds, and a new scheduling algorithm has been extensively tested. Also, demonstrations of a laboratory setup have been executed and published; see [MP-DCCP.Site].

7. IANA Considerations

This section provides guidance to the Internet Assigned Numbers Authority (IANA) regarding the registration of values related to the MP extension of the DCCP protocol in accordance with the RFC Required policy in Section 4.7 of [RFC8126]. This document defines one new value that has been allocated in the IANA "DCCP Feature Numbers" registry and creates three new registries that have been added in the "Datagram Congestion Control Protocol (DCCP) Parameters" registry group.

7.1. New Multipath Capable DCCP Feature

Per this document, IANA has assigned a new DCCP feature parameter for negotiating the support of multipath capability for DCCP sessions between hosts as described in Section 3. The following entry in Table 6 has been added to the "Feature Numbers" registry in the DCCP registry group according to Section 19.4 of [RFC4340].

Number	Description/Meaning	Reference
10	Multipath Capable	RFC 9897

Table 6: Addition to DCCP Feature Numbers Registry

7.2. New MP-DCCP Versions Registry

Section 3.1 specifies the new 1-byte entry above that includes a 4-bit part to specify the version of the used MP-DCCP implementation. IANA has created a new "MP-DCCP Versions" registry in the DCCP registry group to track the MP-DCCP version. The initial content of this registry is as follows:

Version	Value	Reference
0	0000	RFC 9897
1-15	Unassigned	

Table 7: MP-DCCP Versions Registry

Future MP-DCCP versions 1 to 15 will be assigned from this registry using the RFC Required policy (Section 4.7 of [RFC8126]).

7.3. New Multipath Option Type and Registry

IANA has assigned value 46 in the DCCP "Option Types" registry, as described in Section 3.2.

IANA has created a new "Multipath Options" registry within the DCCP registry group. The following entries in Table 8 have been added to the new "Multipath Options" registry. The registry has an upper boundary of 255 in the numeric value field.

Multipath Option	Name	Description	Reference
MP_OPT=0	MP_CONFIRM	Confirm reception/processing of an MP_OPT option	Section 3.2.1
MP_OPT=1	MP_JOIN	Join subflow to an existing MP-DCCP connection	Section 3.2.2
MP_OPT=2	MP_FAST_CLOSE	Close an MP-DCCP connection unconditionally	Section 3.2.3
MP_OPT=3	MP_KEY	Exchange key material for MP_HMAC	Section 3.2.4
MP_OPT=4	MP_SEQ	Multipath sequence number	Section 3.2.5
MP_OPT=5	MP_HMAC	Hash-based message authentication code for MP-DCCP	Section 3.2.6

Multipath Option	Name	Description	Reference
MP_OPT=6	MP_RTT	Transmit RTT values and calculation parameters	Section 3.2.7
MP_OPT=7	MP_ADDADDR	Advertise one or more additional addresses/ports	Section 3.2.8
MP_OPT=8	MP_REMOVEADDR	Remove one or more addresses/ports	Section 3.2.9
MP_OPT=9	MP_PRIO	Change subflow priority	Section 3.2.10
MP_OPT=10	MP_CLOSE	Close an MP-DCCP connection	Section 3.2.11
MP_OPT=11	MP_EXP	Experimental option for private use	Section 3.2.12
MP_OPT=12-255	Unassigned		

Table 8: Multipath Options Registry

Future Multipath Options with MP_OPT>11 will be assigned from this registry using the RFC Required policy (Section 4.7 of [RFC8126]).

7.4. New DCCP-Reset Code

IANA has assigned a new DCCP-Reset Code, value 13, in the "Reset Codes" registry, with the description "Abrupt MP termination". Use of this Reset Code is defined in Section 3.2.3.

7.5. New Multipath Key Type Registry

IANA has created a new "Multipath Key Type" registry for this version of the MP-DCCP protocol that contains two different suboptions to the MP_KEY option to identify the MP_KEY Key types in terms of 8-bit values as specified in Section 3.2.4. See the initial entries in Table 9 below. Values in the range 1-254 (decimal) inclusive remain unassigned in this specified version 0 of the protocol and will be assigned via the RFC Required policy [RFC8126] in potential future versions of the MP-DCCP protocol.

Type	Name	Meaning	Reference
0	Plain Text	Plain text Key	Section 3.2.4
1-254	Unassigned		

Type	Name	Meaning	Reference
255	Experimental	For private use only	Section 3.2.4

Table 9: Multipath Key Type Registry with the MP_KEY Key Types for Key Data Exchange on Different Paths

8. References

8.1. Normative References

[DCCP-PARAMETERS] IANA, "Datagram Congestion Control Protocol (DCCP) Parameters", <https://www.iana.org/assignments/dccp-parameters>.

[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>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, DOI 10.17487/RFC4340, March 2006, <https://www.rfc-editor.org/info/rfc4340>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/info/rfc6234>.

[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>.

8.2. Informative References

[IETF105.Slides] Amend, M., "MP-DCCP for enabling transfer of UDP/IP traffic over multiple data paths in multi-connectivity networks", IETF 105 Proceedings, July 2019, <https://datatracker.ietf.org/meeting/105/materials/slides-105-tsvwg-sessa-62-dccp-extensions-for-multipath-operation-00>.

[MP-DCCP.Paper] Amend, M., Bogenfeld, E., Cvjetkovic, M., Rakocevic, V., Pieska, M., Kassler, A., and A. Brunstrom, "A Framework for Multiaccess Support for Unreliable Internet Traffic using Multipath DCCP", 2019 IEEE 44th Conference on Local Computer Networks (LCN), pp. 316-323, DOI 10.1109/LCN44214.2019.8990746, October 2019, <https://doi.org/10.1109/LCN44214.2019.8990746>.

[MP-DCCP.Site] "Multipath extension for DCCP", <https://multipath-dccp.org/>.

[MULTIPATH-REORDERING] Amend, M. and D. Von Hugo, "Multipath sequence maintenance", Work in Progress, Internet-Draft, draft-amend-iccrg-multipath-reordering-03, 25 October 2021, <https://datatracker.ietf.org/doc/html/draft-amend-iccrg-multipath-reordering-03>.

[OLIA] Khalili, R., Gast, N., Popovic, M., Upadhyay, U., and J. Le Boudec, "MPTCP is not pareto-optimal: performance issues and a possible solution", CoNEXT '12: Proceedings of the 8th international conference on Emerging networking experiments and technologies, pp. 1-12, DOI 10.1145/2413176.2413178, December 2012, <https://dl.acm.org/doi/10.1145/2413176.2413178>.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <https://www.rfc-editor.org/info/rfc3711>.

[RFC5238] Phelan, T., "Datagram Transport Layer Security (DTLS) over the Datagram Congestion Control Protocol (DCCP)", RFC 5238, DOI 10.17487/RFC5238, May 2008, <https://www.rfc-editor.org/info/rfc5238>.

[RFC5596] Fairhurst, G., "Datagram Congestion Control Protocol (DCCP) Simultaneous-Open Technique to Facilitate NAT/Middlebox Traversal", RFC 5596, DOI 10.17487/RFC5596, September 2009, <https://www.rfc-editor.org/info/rfc5596>.

[RFC5597] Denis-Courmont, R., "Network Address Translation (NAT) Behavioral Requirements for the Datagram Congestion Control Protocol", BCP 150, RFC 5597, DOI 10.17487/RFC5597, September 2009, <https://www.rfc-editor.org/info/rfc5597>.

[RFC6356] Raiciu, C., Handley, M., and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols", RFC 6356, DOI 10.17487/RFC6356, October 2011, <https://www.rfc-editor.org/info/rfc6356>.

[RFC6773] Phelan, T., Fairhurst, G., and C. Perkins, "DCCP-UDP: A Datagram Congestion Control Protocol UDP Encapsulation for NAT Traversal", RFC 6773, DOI 10.17487/RFC6773, November 2012, <https://www.rfc-editor.org/info/rfc6773>.

[RFC6904] Lennox, J., "Encryption of Header Extensions in the Secure Real-time Transport Protocol (SRTP)", RFC 6904, DOI 10.17487/RFC6904, April 2013, <https://www.rfc-editor.org/info/rfc6904>.

[RFC6951] Tuexen, M. and R. Stewart, "UDP Encapsulation of Stream Control Transmission Protocol (SCTP) Packets for End-Host to End-Host Communication", RFC 6951, DOI 10.17487/RFC6951, May 2013, <https://www.rfc-editor.org/info/rfc6951>.

[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>.

[RFC8041] Bonaventure, O., Paasch, C., and G. Detal, "Use Cases and Operational Experience with Multipath TCP", RFC 8041, DOI 10.17487/RFC8041, January 2017, <https://www.rfc-editor.org/info/rfc8041>.

[RFC8684] Ford, A., Raiciu, C., Handley, M., Bonaventure, O., and C. Paasch, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 8684, DOI 10.17487/RFC8684, March 2020, <https://www.rfc-editor.org/info/rfc8684>.

[RFC9293] Eddy, W., Ed., "Transmission Control Protocol (TCP)", STD 7, RFC 9293, DOI 10.17487/RFC9293, August 2022, <https://www.rfc-editor.org/info/rfc9293>.

[TS23.501] 3GPP, "System architecture for the 5G System; Stage 2; Release 16", Version 16.7.0, Release 16, December 2020, <https://www.3gpp.org/ftp//Specs/archive/23_series/23.501/23501-g70.zip>.

[U-DCCP] Amend, M., Brunstrom, A., Kassler, A., and V. Rakocevic, "Lossless and overhead free DCCP - UDP header conversion (U-DCCP)", Work in Progress, Internet-Draft, draft-amend-tsvwg-dccp-udp-header-conversion-01, 8 July 2019, <https://datatracker.ietf.org/doc/html/draft-amend-tsvwg-dccp-udp-header-conversion-01>.

Appendix A. Differences from Multipath TCP

This appendix is informative.

MP-DCCP is similar to Multipath TCP [RFC8684] in that it extends the related basic DCCP transport protocol [RFC4340] with multipath capabilities in the same way as Multipath TCP extends TCP [RFC9293]. However, because of the differences between the underlying TCP and DCCP protocols, the transport characteristics of MPTCP and MP-DCCP are different.

Table 10 compares the protocol characteristics of TCP and DCCP, which are by nature inherited by their respective multipath extensions. A major difference lies in the delivery of the payload, which for TCP is an exact copy of the generated byte stream. DCCP behaves differently and does not guarantee the delivery of any payload nor the order of delivery. Since this is mainly affecting the receiving endpoint of a TCP or DCCP communication, many similarities on the sender side can be identified. Both transport protocols share the 3-way initiation of a communication and both employ congestion control to adapt the sending rate to the path characteristics.

Feature	TCP	DCCP
Full-Duplex	yes	yes
Connection-Oriented	yes	yes
Header option space	40 bytes	< 1008 bytes or PMTU
Data transfer	reliable	unreliable
Packet-loss handling	retransmission	report only
Ordered data delivery	yes	no
Sequence numbers	one per byte	one per PDU
Flow control	yes	no
Congestion control	yes	yes
ECN support	yes	yes
Selective ACK	yes	depends on congestion control
Fix message boundaries	no	yes
Path MTU discovery	yes	yes
Fragmentation	yes	no
SYN flood protection	yes	no
Half-open connections	yes	no

Table 10: TCP and DCCP Protocol Comparison

Consequently, the multipath characteristics shown in Table 11 are the same, supporting volatile paths that have varying capacities and latency, session handovers, and path aggregation capabilities. All of these features profit by the existence of congestion control.

Feature	MPTCP	MP-DCCP
Volatile paths	yes	yes
Session handover	yes	yes
Path aggregation	yes	yes
Data reordering	yes	optional

Feature	MPTCP	MP-DCCP
Expandability	limited by TCP header	flexible

Table 11: MPTCP and MP-DCCP Protocol Comparison

Therefore, the sender logic is not much different between MP-DCCP and MPTCP.

The receiver side for MP-DCCP has to deal with the unreliable delivery provided by DCCP. The multipath sequence numbers included in MP-DCCP (see Section 3.2.5) facilitates adding optional mechanisms for data stream packet reordering at the receiver. Information from the MP_RTT Multipath Option (Section 3.2.7), DCCP path sequencing, and the DCCP Timestamp Option provide further means for advanced reordering approaches, e.g., as proposed in [MULTIPATH-REORDERING]. However, such mechanisms do not affect interoperability and are not part of the MP-DCCP protocol. Many applications that use unreliable transport protocols can also inherently process out-of-sequence data (e.g., through adaptive audio and video buffers), so additional reordering support might not be necessary. The addition of optional reordering mechanisms are likely to be needed when the different DCCP subflows are routed across paths with different latencies. In theory, applications using DCCP are aware that packet reordering could occur, because DCCP does not provide mechanisms to restore the original packet order.

In contrast to TCP, the receiver processing for MPTCP adopted a rigid "just wait" approach, because TCP guarantees reliable in-order delivery.

Acknowledgments

[RFC8684] defines Multipath TCP and provides important inputs for this specification.

The authors gratefully acknowledge significant input into this document from Dirk von Hugo, Nathalie Romo Moreno, Omar Nassef, Mohamed Boucadair, Simone Ferlin, Olivier Bonaventure, Gorry Fairhurst, and Behcet Sarikaya.

Authors' Addresses

Markus Amend (EDITOR)
Deutsche Telekom
Deutsche-Telekom-Allee 9
64295 Darmstadt
Germany
Email: Markus.Amend@telekom.de

Anna Brunstrom
Karlstad University
Universitetsgatan 2
SE-651 88 Karlstad
Sweden
Email: anna.brunstrom@kau.se

Andreas Kassler
Karlstad University
Universitetsgatan 2
SE-651 88 Karlstad
Sweden
Email: andreas.kassler@kau.se

Veselin Rakocevic
City St George's, University of London
Northampton Square
London
United Kingdom
Email: veselin.rakocevic.1@city.ac.uk

Stephen Johnson
BT
Adastral Park
Martlesham Heath
IP5 3RE
United Kingdom
Email: stephen.h.johnson@bt.com