

Network Working Group  
Internet Draft  
Intended status: Experimental  
Expires: June 2022

G. Lencse  
Szechenyi Istvan University  
Sz. Szilagyi  
University of Debrecen  
F. Fejes  
Eotvos Lorand University  
M. Georgescu  
RCS&RDS  
December 15, 2021

MPT Network Layer Multipath Library  
draft-lencse-tsvwg-mpt-09.txt

## Abstract

Although several contemporary IT devices have multiple network interfaces, communication sessions are restricted to use only one of them at a time due to the design of the TCP/IP protocol stack: the communication endpoint is identified by an IP address and a TCP or UDP port number. The simultaneous use of these multiple interfaces for a communication session would improve user experience through higher throughput and improved resilience to network failures.

MPT is a network layer multipath solution, which provides a tunnel over multiple paths using the GRE-in-UDP specification, thus being different from both MPTCP and Huawei's GRE Tunnel Bonding Protocol.

MPT can also be used as a router, routing the packets among several networks between the tunnel endpoints, thus establishing a multipath site-to-site connection.

The version of tunnel IP and the version of path IP are independent from each other, therefore MPT can also be used for IPv6 transition purposes.

## Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Draft MPT Network Layer Multipath Library

December 2021

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

The list of current Internet-Drafts can be accessed at

<http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at

<http://www.ietf.org/shadow.html>

This Internet-Draft will expire on June 15, 2009.

## Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the [Trust Legal Provisions](#) and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	<a href="#">Introduction .....</a>	<a href="#">3</a>
<a href="#">1.1.</a>	<a href="#">Design Assumptions .....</a>	<a href="#">3</a>
<a href="#">1.2.</a>	<a href="#">MPT in the Networking Stack .....</a>	<a href="#">3</a>
<a href="#">1.3.</a>	<a href="#">Terminology .....</a>	<a href="#">4</a>
<a href="#">1.4.</a>	<a href="#">MPT Concept .....</a>	<a href="#">5</a>
<a href="#">2.</a>	<a href="#">Conventions Used in this Document .....</a>	<a href="#">6</a>
<a href="#">3.</a>	<a href="#">Operation Overview .....</a>	<a href="#">6</a>
<a href="#">4.</a>	<a href="#">MPT Control .....</a>	<a href="#">8</a>
<a href="#">4.1.</a>	<a href="#">Configuration Information .....</a>	<a href="#">8</a>
<a href="#">4.1.1.</a>	<a href="#">General Information for the MPT Server .....</a>	<a href="#">8</a>
<a href="#">4.1.2.</a>	<a href="#">Connection Specifications .....</a>	<a href="#">9</a>
<a href="#">4.2.</a>	<a href="#">MPT Configuration Commands .....</a>	<a href="#">13</a>
<a href="#">5.</a>	<a href="#">Possible Mappings of the Tunnel Traffic to Paths .....</a>	<a href="#">14</a>

<a href="#">5.1. Per Packet Based Mapping .....</a>	<a href="#">17</a>
<a href="#">5.2. Flow Based Mapping .....</a>	<a href="#">18</a>
<a href="#">5.3. Combined Mapping .....</a>	<a href="#">19</a>
<a href="#">6. Packet Reordering .....</a>	<a href="#">19</a>

<a href="#">7. Why MPT is Considered Experimental? .....</a>	<a href="#">20</a>
<a href="#">7.1. Published Results .....</a>	<a href="#">21</a>
<a href="#">7.1.1. MPT Concept and First Implementation .....</a>	<a href="#">21</a>
<a href="#">7.1.2. Estimation of the Channel Aggregation Capabilities .....</a>	<a href="#">21</a>
<a href="#">7.1.3. Demonstrating the Resilience of an MPT Connection .....</a>	<a href="#">21</a>
<a href="#">7.2. Open questions .....</a>	<a href="#">22</a>
<a href="#">7.2.1. Parameters.....</a>	<a href="#">22</a>
<a href="#">7.2.2. Development of Further Mapping Algorithms.....</a>	<a href="#">22</a>
<a href="#">7.2.3. Performance Issues .....</a>	<a href="#">22</a>
<a href="#">8. Security Considerations .....</a>	<a href="#">22</a>
<a href="#">9. IANA Considerations .....</a>	<a href="#">23</a>
<a href="#">10. Conclusions .....</a>	<a href="#">23</a>
<a href="#">11. References .....</a>	<a href="#">23</a>
<a href="#">11.1. Normative References .....</a>	<a href="#">23</a>
<a href="#">11.2. Informative References .....</a>	<a href="#">23</a>
<a href="#">12. Acknowledgments .....</a>	<a href="#">26</a>
<a href="#">Appendix A. Sample C code for packet reordering .....</a>	<a href="#">27</a>

## [1. Introduction](#)

MPT is a multipath extension of the GRE-in-UDP encapsulation [[RFC8086](#)].

### [1.1. Design Assumptions](#)

MPT is intended to be used as a preconfigured tunnel and the application of MPT does not require any modifications to the applications using the TCP/IP socket interface API.

### [1.2. MPT in the Networking Stack](#)

The layer architecture of MPT is shown in Fig. 1. MPT extends the GRE-in-UDP [[RFC8086](#)] architecture by allowing multiple physical paths. To that end it can be compared to MPTCP [[RFC6824](#)], but unlike MPTCP, MPT uses UDP in the underlying layer, builds on GRE-in-UDP, and provides a tunnel IP layer, over which both UDP and TCP can be

used. The aim of Huawei's GRE tunnel bonding protocol [RFC8157] is also similar to that of MPT: it targets to bonded access to wired and wireless network in customer premises. However, it uses GRE (not GRE-in-UDP) which is less supported in ISP networks than UDP, and it seems to limit the number of physical interfaces to two. For the comparison of MPT with other multipath solutions, please refer to [Alm2017].

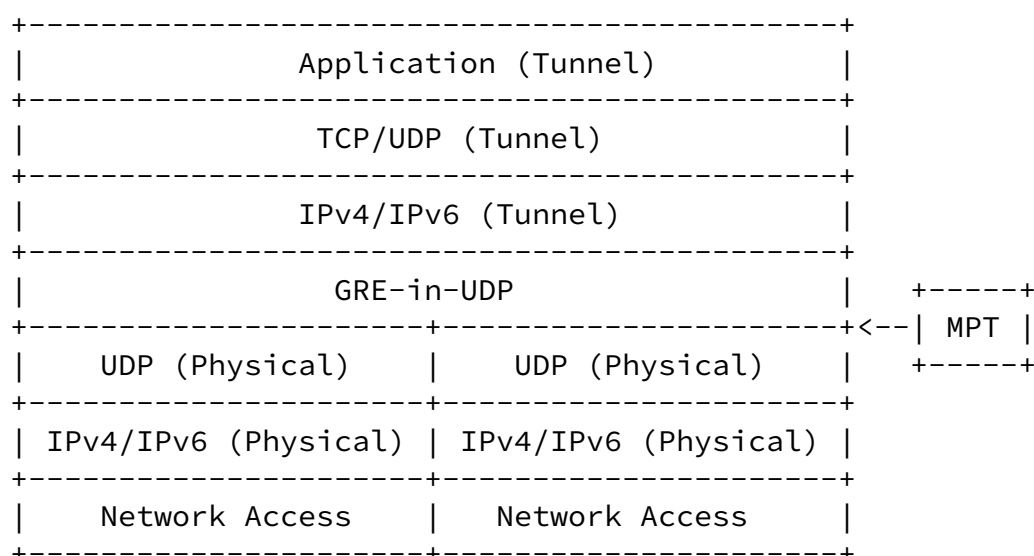


Figure 1: MPT Layer Architecture

### 1.3. Terminology

This document uses a number of terms that are either MPT specific or have defined meaning in the context of MPT as follows:

**MPT server:** An MPT server is a software that implements network layer multipath communication by providing an UDP tunnel (named "connection" in the MPT terminology) over several underlying "paths".

**MPT client:** An MPT client is a software tool, which is used to control the local MPT server (e.g. start/stop connections, add paths to connections, etc.).

**Connection:** An MPT connection (also referred as communication

session) is an UDP tunnel between two MPT servers, which can be used to carry user data. A connection can be established over one or more paths. A connection is initiated on the basis of a "connection specification".

**Path:** A path is used to refer to the pair of the network cards of the end nodes (identified by the pair of IP addresses of the cards). Using a specified path, the packet transmission runs between the given pair of network cards.

**Connection specification:** A connection specification is stored in a configuration file and it is used by an MPT server to establish an MPT connection with another MPT server. It contains all the configuration information for the connection (e.g. endpoint IP

versions and addresses, number of paths and configuration information for all paths). The precise definition of the connection specification can be found in [Section 4.1.2](#).

**Data port:** Data port means the GRE-in-UDP port defined in [[RFC8086](#)] as 4754. It is used for transmitting data packets.

**Local command port:** An MPT server accepts commands from the MPT client at the local command port.

**Remote command port:** An MPT server MAY accept commands from other MPT servers at the remote command port.

**Data plane:** The parts and functions of MPT, which are responsible for handling user data packets.

**Control plane:** All parts and functions of MPT except the data plane. E.g.: handling connections and paths, all the communication through local or remote command ports, etc.

#### [1.4](#). MPT Concept

When an MPT server is started, it reads its configuration files, and depending on its contents, it MAY wait for and accept connection(s) initiated by other MPT server(s) and/or it MAY initiate one or more MPT connection(s) with other MPT server(s). In the simplest case, the MPT server uses the connection specifications described in its configuration files for initiating connections. In addition to that,

new connections MAY be established, connections MAY be closed, the parameters of the connections MAY be changed later (e.g. some paths may be switched on and off) dynamically by issuing the appropriate commands using an MPT client.

MPT connections between MPT servers implement tunnels. The traffic comes from the tunnel interface is distributed over the active paths of the MPT connection by the MPT server. There are three possible mappings (see [Section 5](#) for details and illustrative examples):

- o per packet based mapping, where a mapping decision is made for every single packet
- o flow based mapping, where the flows, identified by the usual five tuple, are always mapped to a given path.
- o combined mapping, where the flows, identified by the usual five tuple, are always mapped to a given connection and a mapping decision is made for every single packet of each connections.

The peer MPT server receives and de-encapsulates the traffic from the different paths and restores the tunnel traffic using the optional GRE sequence numbers for packet reordering if necessary.

## [2.](#) Conventions Used in this Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#) [[RFC2119](#)].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying significance described in [RFC 2119](#).

## [3.](#) Operation Overview

In this Section, we describe the operation of the data plane, whereas the operation of the control plane can be found in [Section 4](#).

The data packet transmission and receive mechanism of MPT is summarized in Fig. 2. Now, we shall follow the route and processing of data packets.



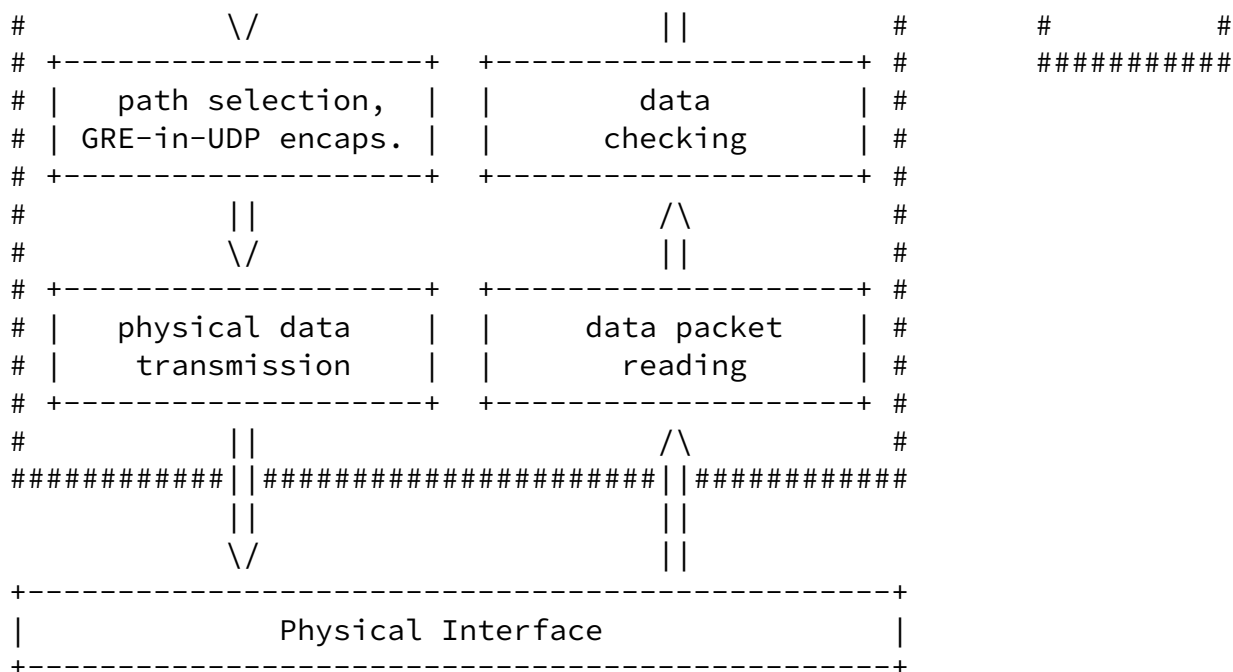


Figure 2: Conceptual architecture of MPT working mechanism

When a packet is read from the physical interface, its destination UDP port number is the 4754 GRE-in-UDP port. MPT reads the packet, identifies the connection the packet belongs to (by the source and destination IP addresses of the tunnel IP header) and runs checking mechanisms (e.g. connection validity check, GRE sequence number check or GRE Key value check, if present). If all the checking mechanisms finish successfully and no reordering is necessary, then the packet is promptly transmitted to the Transport and Application

Layers through the tunnel interface. If reordering is on and GRE sequence number indicates that one or more data unit(s) are missing, then the packet is placed into a buffer array for reordering purposes. (Reordering is discussed in [Section 6](#).)

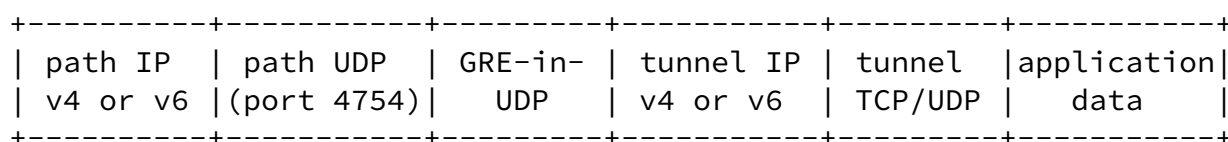


Figure 3: PDU encapsulation of the MPT data communication

#### 4. MPT Control



A connection can be established between two MPT servers in two ways:

1. When the MPT server is started, it establishes the connection on the basis of a connection specification from the configuration files. In this case, the connection specification contains all the necessary parameters. MPT client commands still can be used to modify the parameters, switch off and on paths, etc. as described in [Section 4.2](#).
2. The connection is established by using MPT client commands. In this case the command line arguments of the MPT commands and configuration files contain the necessary parameters.

#### [4.1](#). Configuration Information

The MPT configuration files contain various pieces of information. They can be divided into two groups:

1. general information for the MPT server
2. connections specification(s)

##### [4.1.1](#). General Information for the MPT Server

The MPT configuration file is made up of sections. The "general" section MUST be present and it contains general information for the operation of the MPT server, whereas there MAY several sections follow, each of which describes a different tunnel.

The general section MUST contain the following elements:

- o tunnel number: the number of tunnels to create (they are to be described in separate sections)

- o accept remote: it is a key (yes/no) whether this MPT server should accept commands from other MPT servers to build up connections, which are not defined in the local configuration files
- o local command port: the port number on which the local MPT client software can give commands to the MPT server

- o command timeout: the timeout value for the MPT client

For each tunnel, a separate section is to be used to describe the following parameters:

- o name: The name is used by the operating system to access to the interface.
- o MTU: The maximum transmission unit of the tunnel interface. For the Ethernet environment, the value should be set between 1436 and 1468 (depending on the additional header sizes, used by the actual system). It can be calculated as:  $1500 - \text{Path\_IP\_header\_size} - \text{UDP\_header\_size} - \text{GRE\_header\_size}$ .
- o ipv4\_addr: IPv4 address and mask
- o ipv6\_addr: IPv6 address and mask

Note that both ipv4\_addr and ipv6\_addr MAY be present. At least one of them MUST be present.

It is important that the same tunnel may be used by several connections. A connection can be uniquely identified by the IP addresses of the two endpoints, which have to be of the same type (IPv4 or IPv6).

#### [4.1.2](#). Connection Specifications

A connection specification is made up of sections. The "connection" section contains parameters that are to be specified only ones for each connection. The "paths" section contains one or more path definitions. The optional "networks" section contains network definitions for routing purposes.

The general section (called "connection") MUST contain the following elements:

- o name: The unique name of the connection. If we use multiple connections, the name must uniquely identify the connection.

- o permissions: There MAY be SEND and RECEIVE permissions, which allow sending and receiving connection updates. The term SEND means that the local MPT environment is allowed to start

configuration change to the peer. The term RECEIVE means that the peer is allowed to start a configuration change, and the local MPT environment will accept it. (The actual execution of the requested change depends on further conditions, e.g. successful authentication.)

- o IP version: its possible values are 4 or 6.
- o local IP address: must be of the IP version specified above and must be the same as defined for the tunnel.
- o remote IP address: the IP address of the remote peer, must be of the IP version specified above
- o local data port number: used for data communication, SHOULD be set to the 4754 GRE-in-UDP port number
- o remote data port number: used for data communication, SHOULD be set to the 4754 GRE-in-UDP port number
- o remote command port number: The UDP port number of the peer, which is used to accept control commands. If the local MPT client starts an MPT command (e.g. turning off a path usage), the MPT server will communicate this action to the peer by using the remote command port number as the destination port number.
- o path count: The key is an integer  $P$ , denoting the number of paths defined for this connection. The minimum value is 1, the maximum value is implementation dependent, e.g. 20. This configuration file MUST have  $P$  sections (usually named [path\_n]), where  $0 \leq n < P$ , describing all paths of the connection.
- o network count: The MPT environment can be used to establish a tunnel between networks (i.e. not only the tunnel peers can use the tunnel for communication). The minimum value is 0, the maximum value is implementation dependent, e.g. 20. This key is an integer  $L$ , denoting the number of networks on which the actual connection is able to route. This configuration file MUST have  $L$  sections (usually named [net\_n]), where  $0 \leq n < L$ , describing all networks that belong to the connection.
- o status: The key indicates the initial status of the connection. The value 0 means OK.

- o authentication type: The MPT system uses control communication between the tunnel endpoints. The control communication can be requested to use authentication. The value 0 means no authentication.

The following elements are OPTIONAL:

- o reorder window: The reorder window value specifies the length (or size) of the buffer-array for the optional packet reordering on the basis of the GRE sequence numbers, see [Section 6](#) for more information. The value of 0 (which is the default value when omitting the key) means, that no sorting will be performed at the receiver (the packets are transferred to the tunnel interface immediately when they arrive).
- o maximum buffer delay: This key is used only if we require ordered packet transmission (i.e. reorder window > 0). If ordered packet transmission is required, maximum buffer delay specifies the maximum time (in milliseconds) while the packet may be stored in the buffer-array.
- o authentication key: The authentication key contains the key value of the control communication authentication. Some algorithms do not need authentication keys. In this case the specification of the authentication key is not necessary, or will be ignored.

A path definition section MUST contain the following elements:

- o interface name: The value is the name of the physical interface used by the given path for packet forwarding (e.g. eth0, wlan0).
- o IP version: Specifies the version of IP used by the path. The value can be 4 or 6.
- o public IP address: Specifies the public IP address of the interface used for the tunnel communication. If the host is placed into the Global Address Realm, the public IP address is the IP address of the interface, otherwise (i.e. when the host is behind a NAT-Box) it is the public address assigned by the NAT-Box to the tunnel communication session. If the path uses IPv4 and NAT, then the special address value of 0.0.0.0 can be used to force the MPT server program to determine the public IP address automatically.
- o remote IP address: Indicates the public IP address of the remote endpoint.

- o gateway IP address: The IP address of the gateway, used to reach the peer (i.e. remote IP address) using the given path. If the operating system uses the Network Manager (nmcli) software for network configuration, then the value of 0.0.0.0 can be used to find the gateway of the named interface automatically.
- o weight out: This is the "weight of the path" in the system expressing the estimated transmission capacity of the path. The MPT server program distributes the outgoing packets between the available paths according to their weights, if per packet based mapping is used. The value must be between 1 and 10,000.
- o status: This key means the initial state of the path after starting the MPT server. The value "up" means that the path is usable (working), and the state of the path is OK. If required, may be set initially as "down".

A path definition section MAY contain the following elements:

- o private IP address: The IP address of the physical interface. Can be omitted, if the public IP address is assigned directly to the interface. When using IPv4 and NAT, the special value of 0.0.0.0 can be used to force the MPT server application to read and use the first IPv4 address assigned to the interface.
- o keepalive time: The MPT system monitors the availability of each path by sending keepalive messages regularly. The key specifies the frequency (i.e. the time between the keepalive messages in seconds) that the MPT server uses for sending keepalives. The value of zero (which is the default value) means switching off the keepalive mechanism.
- o dead time: If the keepalive mechanism is active, and the host does not receive any keepalive message on the given path from the peer for dead time seconds, then the path is considered as "dead" and will not be used for data transmission. (The default value is 3\*keepalive time.)
- o weight in: This field is used at the "mpt path up" command (see [Section 4.2](#)) to set the outgoing weight of the corresponding path at the peer. The default value is 1.
- o command default: This key can be used to specify one path as the

default path for control command communication. In the case of receiving the control command of "create connection", the system will use this path for the control communication.

The optional "networks" section contains network definitions for routing purposes. Each network definition begins with its name in the [net\_n] format and contains the following parameters:

- o IP version: Specifies the version of IP used in the network definitions. The value can be 4 or 6.
- o source address: specifies the source network and its prefix length in the CIDR notation.
- o destination address: specifies the destination network and its prefix length in the CIDR notation.

The network configuration can also be used to provide multipath Internet connection by specifying 0.0.0.0/0 as destination address and prefix length. (The source is our tunnel address in this case.)

#### [4.2.](#) MPT Configuration Commands

The same control interface is used for the local administration of the MPT server (by the MPT client accessing the MPT server at the local command port through the loopback interface) and for the communication of the local MPT server with the remote MPT server (accessing it at its remote command port).

Now, some client commands will follow. Although some of the syntax of our MPT implementation will be used, the focus is not on their syntax, which may be implementation dependent, but rather on their functionalities. The execution of these commands may also involve communication between the local MPT server and a/the remote MPT server.

```
mpt address {add|del} IPADDRESS/PREFIX dev INTERFACE
```

An IPv4 or IPv6 address can be added to or deleted from a (local) interface.

```
mpt interface INTERFACE {up|down}
```

The specified interface is turned up or down plus all the paths, that are based on the given local physical interface are also turned on or off by starting the "mpt path {up|down}" command (see below) for each considered path.

```
mpt path PATH {up|down}
```

This command can be used to turn on or off a specified path. If the path status is changed to down, then it is not used by the connection, (i.e. no data is sent through that path by the MPT software).

```
mpt connection CONNECTION {create|delete}
```

This command can be used to establish or tear down a connection between the local and a remote MPT server. (The parameters are taken from local configuration files.) If the remote server is configured so, then it accepts the parameters of the connection from the local server.

```
mpt save [FILENAME]
```

The current configuration can be changed during runtime by remote peers. (This can be enabled with the accept remote key and with the permissions key.) This command is used to write these connection changes to the configuration files, so the new settings will remain after server startup or after mpt reload.

```
mpt reload [FILENAME]
```

Warm restart: the MPT server build up its connections according to its configuration files. (Our implementation only establishes, but it does not tears down connections.)

## 5. Possible Mappings of the Tunnel Traffic to Paths

The data packets coming from the tunnel interface must be forwarded through one of the active paths of the connection. Three possible mapping solutions are proposed:

- o Per packet based mapping means that the tunnel traffic is distributed among the paths on the basis of the parameters of the paths only, and regardless of what network flow a given packet belongs to.
- o Flow based mapping means that packets which belong to a given network flow, identified by the usual five tuple of source IP address, destination IP address, source port number, destination port number, and protocol number (TCP or UDP), or three tuple of source IP address, destination IP address, and protocol number (TCP, UDP or ICMP), are always mapped to the same path.
- o Combined mapping means the combinations of the two above in the way that packets which belong to a given network flow, identified by the way described above, are always mapped to the same connection. And the packets that belong to a connection are distributed among the paths of that connection by per packet decisions on the basis of the parameters of the paths of the connection.

We illustrate the three mapping solutions by examples.

Definitions for the examples:

Computers A and B are interconnected by 3 different paths:

path\_1: 100Base-TX Ethernet



path\_2: 802.11g WiFi

path\_3: LTE

Connection\_1 has 3 paths with the following weight out values:

path\_1: 5

path\_2: 2

path\_3: 3

Example 1 (Per packet based mapping)

All the traffic between the two computers is distributed among the three paths of Connection\_1 proportionally to their weight out

values. A decision is made about every single packet as described in [Section 5.1](#), regardless of the fact what application it belongs to.

Advantage: The transmission capacity of all the paths can be utilized.

Disadvantage: There is no possibility to use different mappings for different applications.

Example 2 (Per flow based mapping)

Based on the destination port number or port range, the traffic of different applications are mapped to paths as follows:

HTTP, VoD: path\_1

FTP, Bit-Torrent: path\_2

VoIP: path3

Advantage: Application can be differentiated: e.g. the delay

critical VoIP can use LTE, whereas the free WiFi is satisfactory for the non-mission critical Bit-Torrent.

Disadvantage: The mapping of the traffic is too rigid, all the traffic of applications of a given type is mapped to a single path, therefore, the applications (and thus their users) do not experience the benefits of multipath transmission.

### Example 3 (Combined mapping)

We define further two connections:

Connection\_2

path\_1: 5

path\_2: 2

Connection\_2

path\_1: 5

path\_3: 3

Based on the destination port number or port range, the traffic of different applications are mapped to paths as follows:

HTTP: connection\_1

FTP, Bit-Torrent: connecton\_2

VoIP, VoD: connection\_3

Advantage: The applications may benefit from the multipath transmission, whereas each types of applications use those paths, which are beneficial and affordable for them.

Disadvantage: The price of the above resilience is the time and computational complexity of the execution of both algorithms.

Conclusion: The appropriate choice of the mapping algorithm depends on the expectations of the user.

### [5.1.](#) Per Packet Based Mapping

The aim of the "per packet based" mapping is to distribute the tunnel traffic to the paths proportionally to their transmission capacity. This mapping facilitates the aggregation of the transmission capacities of the paths.

In MPT, the transmission capacity of the paths is represented by their WEIGT\_OUT parameter.

The following algorithm calculates the sending vector, which contains the indices of the paths in the order they are to be used for transmission.

ALGORITHM calculate\_sending\_vector

INPUT:  $W[i]$  ( $1 \leq i \leq N$ ), the vector of the weights of the paths.

(Note: We have  $N$  paths with indices  $(1, \dots, N)$ )

OUTPUT:  $O[j]$  ( $1 \leq j \leq M$ ), the sending vector containing the indices of the paths; where  $M$  is the length of the sending cycle.

$lcm :=$  Least Common Multiple for  $(W[1], \dots, W[N])$

$M := 0$

$s[i] := 0$ , for all  $i$  ( $1 \leq i \leq N$ )

(Note:  $s[i]$  will store the sum of the increments for path  $i$ , where the increment is  $lcm/W[i]$ )

WHILE TRUE DO

$z := \min(s[1]+lcm/W[1], \dots, s[N]+lcm/W[N])$

```

    k := The smallest index i, for which  $z == s[i] + lcm/W[i]$ 

    M := M+1

    s[k] := z

    O[M] := k

    IF s[i] == z for all i (1 <= i <= N) THEN RETURN

DONE

END

```

A sample C code can be found in the Appendix.

## [5.2.](#) Flow Based Mapping

The aim of the flow based mapping is to be able to distinguish the packets belong to different network flows and map them to the path that was set for them. (E.g. WiFi is used for Torrent traffic and LTE is used for VoIP calls.)

Our current implementation realizes a port-based flow mapping. It is possible to select the interface for the outgoing traffic based on transport protocol and port. For communication between two MPT servers, you can precisely specify which flow mapped to which path.

The configuration of the mechanism is simple. Four new values can be added for the definition of paths:

tcp\_dst - TCP destination port matches

tcp\_src - TCP source port matches

udp\_dst - UDP destination port matches

udp\_src - UDP source port matches

All of these are optional, they can be listed in many ports. Ports that are not defined will continue to be per-packet based. The

current implementation of the MPT with flow based mapping can be found on [\[MptFlow\]](#)

In the example below, each outgoing TCP packet with destination port 80, 443, and 8080 and UDP packet with destination port 5901 will be sent on path\_1. TCP packets with source ports 7880 and 56000 will be sent on path\_2.

Example (flow based mapping configuration snippet)

```
[path_1]
...
tcp_dst      = 80 443 8080
udp_dst      = 5901
```

```
[path_2]

...

tcp_src      = 7880 56000
```

### [5.3.](#) Combined Mapping

TBD

## [6.](#) Packet Reordering

As the delay of the different paths can be different, packet reordering may appear in a packet sequence transmission. The MPT environment offers an optional feature to ensure the right ordered packet transmission for the tunnel communication. If this feature is enabled, the receiver uses a buffer-array to store the incoming (unordered) packets. Then the packets are sorted according to the GRE sequence numbers, so ensuring the ordered transmission to the receiver's tunnel interface.

There are two parameters aimed to control the reordering. The reorder window parameter specifies the length of the buffer array used for reordering. The maximum buffer delay parameter specifies

the maximum time (in milliseconds) while the packet is stored in the buffer-array. If the packet is delayed in the buffer-array for the specified time, it will be transmitted to the tunnel interface, even in the case, when some packets are missing before the considered packet. The missing packets are considered as lost packets (i.e. we will not wait more for a lost packet). The arrived packets are transferred to the tunnel interface according to their GRE sequence number, so the ordered delivery will be kept also in the case of packet loss.

How to set the values of these parameters?

As for maximum buffer delay, if its value is too small, then MPT may incorrectly consider a sequence number as lost, and if it arrives later, MPT has to drop it to keep on the order-right delivery. If its value is too large, then the packet loss will be determined too late, and thus the communication performance may decrease. Our experience shows that a feasible choice could be: a few times the RTT (Round-Trip Time) of the slowest path.

As for reorder window, it MUST be large enough to store packets arriving at maximum line rate from all the active paths of the given connection during a maximum buffer delay interval.

The appropriate choice of these parameters is still subject of research.

## [7.](#) Why MPT is Considered Experimental?

We view MPT as a research area rather than a solution which is ready for deployment. We have an MPT implementation, which is workable, but it contains only the "per packet based" mapping of the tunnel traffic to the paths. One of our aims of writing this Internet Draft is to enable others to write MPT implementations. It is our hope that the experience gained with preparing other implementations as well as the results of their testing and performance analysis will lead to a better MPT specification, which may then serve as a standard track specification of an improved MPT, which will be ready for deployment.

In this section, we summarize the most important results as well as the open questions of the MPT related research.

## [7.1.](#) Published Results

### [7.1.1.](#) MPT Concept and First Implementation

The conceptual architecture of MPT, comparison with other multipath solutions, some details of the first implementation and some test results are available in [[Alm2017](#)].

The user manual of the first MPT implementation and the precompiled MPT libraries for Linux (both i386 and amd64) and Raspbian are available from [[Mpt2017](#)].

### [7.1.2.](#) Estimation of the Channel Aggregation Capabilities

The channel aggregation capabilities of an early MPT implementation, which did not use GRE-in-UDP, were analyzed up to twelve 100Mbps links in [[Len2015](#)].

Some of the above tests were repeated with the current GRE-in-UDP based MPT implementation, and the path aggregation capabilities of MPT were compared to that of MPTCP in [[Kov2016](#)] and [[Szi2018](#)].

Measurements were performed also using two 1Gbps links [[Szi2018b](#)] and four 1Gbps links [[Szi2019](#)].

The performance of MPT and MPTCP was compared using two 10Gbps links [[Szi2019b](#)].

To test MPT in a more realistic environment than the previous simple laboratory testbeds, an emulated WAN environment was used for throughput aggregation of two 100Mbps links [[Szi2021](#)] and [[Szi2022](#)].

### [7.1.3.](#) Demonstrating the Resilience of an MPT Connection

The resilience property of the early MPT implementation, which did not use GRE-in-UDP, was demonstrated in [[Alm2014](#)] and in [[Alm2015](#)].

The fast connection recovery of the GRE-in-UDP based MPT implementation was demonstrated in [[Fej2016](#)].

Playout buffer length triggered path switching algorithm was developed for the GRE-in-UDP based MPT, and its effectiveness was demonstrated by the elimination of the stalling events on YouTube video playback [[Fej2017](#)].

#### [7.1.4.](#) Examining the Effect of Different Congestion Control Algorithms

The throughput of MPT and MPTCP using various TCP congestion control algorithms (TCP Cubic, Highspeed, Illinois, Reno, Scalable, Vegas, Veno) were compared in a test environment containing four 100Mbps communication channels in [[Szi2020](#)].

### [7.2.](#) Open questions

#### [7.2.1.](#) Parameters

The optimal (or good enough) choice of the reorder window size and maximum buffer delay parameters are important questions, which should be solved before MPT can be deployed.

#### [7.2.2.](#) Development of Further Mapping Algorithms

The current MPT implementation [[Mpt2017](#)] includes only the per packet base mapping. For a precise specification of the further two mapping algorithms, we would like to use our experiences with them. There are some open questions e.g. how to handle the traffic that is neither TCP nor UDP?

#### [7.2.3.](#) Performance Issues

The current MPT implementation [[Mpt2017](#)] works in user space. Thus, it is not surprising, that multipath transmission of the same amount of traffic by MPT results in higher CPU load than its multipath transmission by MPTCP [[Kov2019](#)]. How much CPU power could a kernel space MPT implementation save?

It was also pointed out by [[Kov2019](#)], that MPT is not able to utilize the computing power of more than two CPU cores. It is so, because MPT uses only two threads (one for each direction). This is not a serious issue, when MPT is used on personal computers. However, when MPT is used to connect several networks, it is an important question, how MPT could utilize the computing power of the modern CPUs with several cores.

### [7.3.](#) Implementation

A sample implementation of the MPT software is available from [[MptSrc](#)] under GPLv3 license. It is intended for research and



experimentation purposes only, as it has not been sufficiently tested to be used for commercial purposes.

## [8](#). Security Considerations

Threats that apply to GRE-in-UDP tunneling, apply here, too. For the security considerations of GRE-in-UDP, please refer to [Section 11 of \[RFC8086\]](#).

If an MPT server is configured so, its peer is allowed to build up connections. It may lead to resource exhaustion and thus successful DoS (Denial of Service) attacks.

Authentication between MPT servers is optional, which may lead to security issues.

## [9](#). IANA Considerations

Port numbers may be reserved for local command port and remote command port.

## [10](#). Conclusions

Hereby we publish the specifications of the MPT network layer multipath library in the hope, that it can be made better by the review and comments of the WG members and, after answering several open questions, one day MPT can mature to be a production tool. We seek for interested volunteers for a different implementation and we would be happy to take part in research cooperation. We welcome all kinds of feedback from anyone to make MPT better.

## [11](#). References

### [11.1](#). Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.

[RFC8086] Young, L. (Editor), Crabbe, E., Xu, X., and T. Herbert, "GRE-in-UDP Encapsulation", [RFC 8086](#), DOI: 10.17487/RFC8086, March 2017.

## 11.2. Informative References

- [Alm2014] Almasi, B., "A solution for changing the communication interfaces between WiFi and 3G without packet loss", in Proc. 37th Int. Conf. on Telecommunications and Signal Processing (TSP 2014), Berlin, Germany, Jul. 1-3, 2014, pp. 73-77

- [Alm2015] Almasi, B., Kosa, M., Fejes, F., Katona, R., and L. Pusok, "MPT: a solution for eliminating the effect of network breakdowns in case of HD video stream transmission", in: Proc. 6th IEEE Conf. on Cognitive Infocommunications (CogInfoCom 2015), Gyor, Hungary, 2015, pp. 121-126, doi: 10.1109/CogInfoCom.2015.7390576 .
- [Alm2017] Almasi, B., Lencse, G., and Sz. Szilagyi, "Investigating the Multipath Extension of the GRE in UDP Technology", Computer Communications (Elsevier), vol. 103, no. 1, (2017.) pp. 29-38, DOI: 10.1016/j.comcom.2017.02.002
- [Fej2016] Fejes, F., Katona, R., and L. Pusok, "Multipath strategies and solutions in multihomed mobile environments", in: Proc. 7th IEEE Conf. on Cognitive Infocommunications (CogInfoCom 2016), Wroclaw, Poland, 2016, pp. 79-84, doi: 10.1109/CogInfoCom.2016.7804529
- [Fej2017] Fejes, F., Racz, S., and G. Szabo, "Application agnostic QoE triggered multipath switching for Android devices", In: Proc. 2017 IEEE International Conference on Communications (IEEE ICC 2017), Paris, France, 21-25 May 21-25, 2017. pp. 1585-1591.
- [Kov2016] Kovacs, A., "Comparing the aggregation capability of the MPT communications library and multipath TCP", in: Proc. 7th IEEE Conf. on Cognitive Infocommunications (CogInfoCom 2016), Wroclaw, Poland, 2016, pp. 157-162, doi: 10.1109/CogInfoCom.2016.7804542
- [Kov2019] Kovacs, A., "Evaluation of the Aggregation Capability of the MPT Communications Library and Multipath TCP", Acta Polytechnica Hungarica, vol. 16, no. 6, 2019, pp. 129-147.

- [Len2015] Lencse, G. and A. Kovacs, "Advanced Measurements of the Aggregation Capability of the MPT Multipath Communication Library", International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems, vol. 4. no. 2. (2015.) pp 41-48. DOI: 10.11601/ijates.v4i2.112
- [Szi2018] Szilagyi, Sz., Fejes, F. and R. Katona, "Throughput Performance Comparison of MPT-GRE and MPTCP in the Fast Ethernet IPv4/IPv6 Environment", Journal of Telecommunications and Information Technology, vol. 3. no. 2. (2018.) pp 53-59. DOI: 10.26636/jtit.2018.122817

- [Szi2018b] Szilagyi, Sz., Bordan, I., Harangi, L. and B. Kiss, "MPT-GRE: A Novel Multipath Communication Technology for the Cloud", in: Proc. 9th IEEE Conf. on Cognitive Infocommunications (CogInfoCom 2018), Budapest, Hungary, 2018, pp. 81-86, doi: 10.1109/CogInfoCom.2018.8639941
- [Szi2019] Szilagyi, Sz., Bordan, I., Harangi, L. and B. Kiss, "Throughput Performance Comparison of MPT-GRE and MPTCP in the Gigabit Ethernet IPv4/IPv6 Environment", Journal of Electrical and Electronics Engineering, vol. 12. no. 1. (2019.), pp. 57-60. ISSN: 1844-6035
- [Szi2019b] Szilagyi, Sz., Bordan, I., Harangi, L. and B. Kiss, "Throughput Performance Analysis of the Multipath Communication Technologies for the Cloud", Journal of Electrical and Electronics Engineering, Vol. 12, No. 2, (2019.), pp. 69-72, ISSN: 1844-6035
- [Szi2020] Szilagyi, Sz., Bordan, I., "The Effects of Different Congestion Control Algorithms over Multipath Fast Ethernet IPv4/IPv6 Environments", In: Proceedings of the 11th International Conference on Applied Informatics (ICAI 2020), Eger, Hungary, 29-31 January, 2020. pp. 341-349. <http://ceur-ws.org/Vol-2650/paper35.pdf>
- [Szi2021] Szilagyi, Sz., Bordan, I., "Throughput Performance Measurement of the MPT-GRE Multipath Technology in Emulated WAN Environment", In: Proceedings of the 1st

Conference on Information Technology and Data Science (CITDS 2020), Debrecen, Hungary, 6-8 November, 2020. pp. 187-195.

<http://ceur-ws.org/Vol-2874/short17.pdf>

[Szi2022] Szilagyi, Sz., Bordan, I., "Investigating the Throughput Performance of the MPT-GRE Network Layer Multipath Library in Emulated WAN Environment", Acta Technica Jaurinensis, in press.

<https://doi.org/10.14513/actatechjaur.00639>

[Mpt2017] MPT - Multipath Communication Library,  
<https://irh.inf.unideb.hu/~szilagyi/index.php/en/mpt/>

[MptFlow] "MPT - Multi Path Tunnel", source code version with flow based packet to path mapping feature,

[https://github.com/spyff/mpt/tree/flow\\_mapping](https://github.com/spyff/mpt/tree/flow_mapping)

Lencse et al.

Expires June 15, 2022

[Page 25]

---

Internet-Draft    MPT Network Layer Multipath Library

December 2021

[MptSrc] "MPT - Multi Path Tunnel", source code,  
<https://github.com/spyff/mpt>

[RFC6824] Ford, A., Raiciu, C, Handley, M., and O Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", [RFC 6824](#), DOI: 10.17487/RFC6824, January, 2013.

[RFC8157] Leymann, N., Heidemann, C., Zhang, M., Sarikaya, B, and M. Cullen, "Huawei's GRE Tunnel Bonding Protocol", [RFC 8157](#), DOI: 10.17487/RFC8157, May, 2017

## [12](#). Acknowledgments

The MPT Network Layer Multipath Library was invented by Bela Almasi, the organizer and original leader of the MPT development team.

This document was prepared using 2-Word-v2.0.template.dot.

[Appendix A](#). Sample C code for calculating the packet sending order

```
<CODE BEGINS>
void calculate_pathselection(connection_type *con) {
    long long lcm;
    long gcd, min_inc, cinc;
    int i,j, min_idx;
    path_type *p;

    con->pathselectionlength = 0;
    gcd = con->mpath[0].weight_out;
    lcm = gcd;
    for (i = 0; i < M; i++)
        O[i] = NULL;

    for (i = 0; i < con->path_count; i++) {
        gcd = CALCULATE_GCD(gcd, con->mpath[i].weight_out);
        lcm = (lcm * con->mpath[i].weight_out) / gcd;
        con->mpath[i].selection_increment = 0;
```

```

    }

    for (j = 0; j < M; j++) {
        min_idx = 0;
        min_inc = lcm + 1;
        for (i = 0; i < con->path_count; i++) {
            p = &con->mpath[i];
            cinc = p->selection_increment + (lcm / p->weight_out);
            if ((p->weight_out) && (cinc < min_inc)) {
                min_idx = i;
                min_inc = cinc;
            }
        }
        O[j] = &con->mpath[min_idx];
        con->mpath[min_idx].selection_increment = min_inc;

        for (i = 0; i < con->path_count; i++) // check if ready
            if (con->mpath[i].selection_increment != min_inc)
                goto NEXT_SELECTION;
        break;

    NEXT_SELECTION:
        continue;
    }
    con->path_index = 0;
    con->pathselectionlength = j + 1;
}
<CODE ENDS>

```

Copyright (c) 2021 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in [Section 4.c](#) of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).

#### Authors' Addresses

Gabor Lencse  
Szechenyi Istvan University  
Egyetem ter 1  
H-9026 Győr  
Hungary

Phone: +36 1 613 665  
Email: lencse@sze.hu

Szabolcs Szilagyi  
University of Debrecen  
Egyetem ter 1.  
H-4032 Debrecen  
Hungary

Phone: +36 52 512 900 / 75013  
Email: szilagyi.szabolcs@inf.unideb.hu

Ferenc Fejes  
Eotvos Lorand University  
Egyetem ter 1-3.  
H-1053 Budapest  
Hungary

Phone: +36 70 545 48 07  
Email: fejes@inf.elte.hu

Marius Georgescu  
RCS&RDS  
Strada Dr. Nicolae D. Staicovici 71-75  
Bucharest 030167  
Romania

Phone: +40 31 005 0979  
Email: marius.georgescu@rcs-rds.ro