

---

# **MTP: Mesh Transport Protocol**

***Release 0.0.1***

**Ertan Onur**

**Dec 08, 2023**



**CONTENTS:**

<b>1</b>	<b>MTP</b>	<b>1</b>
1.1	Overview . . . . .	1
1.2	Features . . . . .	1
1.3	Headers . . . . .	1
1.4	Authors . . . . .	1
<b>2</b>	<b>Indices and tables</b>	<b>11</b>
	<b>Index</b>	<b>13</b>



## 1.1 Overview

This readme provides background on the Mesh Transport Protocol (MTP).

## 1.2 Features

MTP provides the following **features**:

- MTP separates policy from mechanism. The user space application is able to configure supported features of MTP
- MTP

## 1.3 Headers

## 1.4 Authors

### Authors

Ertan Onur

int **MTP\_load**(void)

invoked when this module is loaded into the Linux kernel

### Parameters

void

no arguments

### Return

0 on success, otherwise a negative errno.

void **\_\_exit MTP\_unload** (void)

invoked when this module is unloaded from the Linux kernel.

### Parameters

void

no arguments

int **MTP\_bind**(struct socket \*sock, struct sockaddr \*addr, int addr\_len)

Implements the bind system call for MTP sockets: associates a well-known service port with a socket. Unlike other AF\_INET6 protocols, there is no need to invoke this system call for sockets that are only used as clients.

### Parameters

**struct socket \*sock**

Socket on which the system call was invoked.

**struct sockaddr \*addr**

Contains the desired port number.

**int addr\_len**

Number of bytes in uaddr.

### Return

0 on success, otherwise a negative errno.

void **MTP\_close**(struct sock \*sk, long timeout)

Invoked when close system call is invoked on a MTP socket.

### Parameters

**struct sock \*sk**

Socket being closed

**long timeout**

??

int **MTP\_shutdown**(struct socket \*sock, int how)

Implements the shutdown system call for MTP sockets.

### Parameters

**struct socket \*sock**

Socket to shut down.

**int how**

Ignored: for other sockets, can independently shut down sending and receiving, but for MTP any shutdown will shut down everything.

### Return

0 on success, otherwise a negative errno.

int **MTP\_disconnect**(struct sock \*sk, int flags)

Invoked when disconnect system call is invoked on a MTP socket.

### Parameters

**struct sock \*sk**

Socket to disconnect

**int flags**

??

### Return

0 on success, otherwise a negative errno.

int **MTP\_ioc\_abort**(struct sock \*sk, unsigned long arg)

The top-level function for the ioctl that implements the MTP\_abort user-level API.

### Parameters

**struct sock \*sk**

Socket for this request.

**unsigned long arg**

Used to pass information from user space.

#### **Return**

0 on success, otherwise a negative errno.

int **MTP\_ioctl**(struct sock \*sk, int cmd, int \*arg)

Implements the ioctl system call for MTP sockets.

#### **Parameters**

**struct sock \*sk**

Socket on which the system call was invoked.

**int cmd**

Identifier for a particular ioctl operation.

**int \*arg**

Operation-specific argument; typically the address of a block of data in user address space.

#### **Return**

0 on success, otherwise a negative errno.

int **MTP\_socket**(struct sock \*sk)

Implements the socket(2) system call for sockets.

#### **Parameters**

**struct sock \*sk**

Socket on which the system call was invoked. The non-MTP parts have already been initialized.

#### **Return**

always 0 (success).

int **MTP\_setsockopt**(struct sock \*sk, int level, int optname, sockptr\_t optval, unsigned int optlen)

Implements the setsockopt system call for MTP sockets.

#### **Parameters**

**struct sock \*sk**

Socket on which the system call was invoked.

**int level**

Level at which the operation should be handled; will always be IPPROTO\_MTP.

**int optname**

Identifies a particular setsockopt operation.

**sockptr\_t optval**

Address in user space of information about the option.

**unsigned int optlen**

Number of bytes of data at **optval**.

#### **Return**

0 on success, otherwise a negative errno.

```
int MTP_getsockopt (struct sock *sk, int level, int optname, char __user *optval,  
int __user *option)
```

Implements the getsockopt system call for MTP sockets.

#### Parameters

**struct sock \*sk**

Socket on which the system call was invoked.

**int level**

??

**int optname**

Identifies a particular setsockopt operation.

**char \_\_user \*optval**

Address in user space where the option's value should be stored.

**int \_\_user \*option**

??.

#### Return

0 on success, otherwise a negative errno.

```
int MTP_sendmsg(struct sock *sk, struct msghdr *msg, size_t length)
```

Send a request or response message on a MTP socket.

#### Parameters

**struct sock \*sk**

Socket on which the system call was invoked.

**struct msghdr \*msg**

Structure describing the message to send; the msg\_control field points to additional information.

**size\_t length**

Number of bytes of the message.

#### Return

0 on success, otherwise a negative errno.

```
int MTP_recvmsg(struct sock *sk, struct msghdr *msg, size_t len, int flags, int *addr_len)
```

Receive a message from a MTP socket.

#### Parameters

**struct sock \*sk**

Socket on which the system call was invoked.

**struct msghdr \*msg**

Controlling information for the receive.

**size\_t len**

Total bytes of space available in msg->msg\_iov; not used.

**int flags**

Flags from system call, not including MSG\_DONTWAIT; ignored.

**int \*addr\_len**

Store the length of the sender address here

#### Return



The length of the message on success, otherwise a negative  
errno.

```
int MTP_sendpage(struct sock *sk, struct page *page, int offset, size_t size, int flags)
    ??.
```

#### Parameters

**struct sock \*sk**  
Socket for the operation

**struct page \*page**  
??

**int offset**  
??

**size\_t size**  
??

**int flags**  
??

#### Return

0 on success, otherwise a negative errno.

```
int MTP_hash(struct sock *sk)
    ??.
```

#### Parameters

**struct sock \*sk**  
Socket for the operation

#### Return

??

```
void MTP_unhash(struct sock *sk)
    ??.
```

#### Parameters

**struct sock \*sk**  
Socket for the operation

```
void MTP_rehash(struct sock *sk)
    ??.
```

#### Parameters

**struct sock \*sk**  
Socket for the operation

```
int MTP_get_port(struct sock *sk, unsigned short snum)
    It appears that this function is called to assign a default port for a socket.
```

#### Parameters

**struct sock \*sk**  
Socket for the operation

**unsigned short snum**  
Unclear what this is.

### Return

Zero for success, or a negative errno for an error.

int **MTP\_diag\_destroy**(struct sock \*sk, int err)  
??.

### Parameters

**struct sock \*sk**  
Socket for the operation

**int err**  
??

### Return

??

int **MTP\_v4\_early\_demux**(struct sk\_buff \*skb)  
Invoked by IP for ??.

### Parameters

**struct sk\_buff \*skb**  
Socket buffer.

### Return

Always 0?

int **MTP\_v4\_early\_demux\_handler**(struct sk\_buff \*skb)  
invoked by IP for ??.

### Parameters

**struct sk\_buff \*skb**  
Socket buffer.

### Return

Always 0?

int **MTP\_softirq**(struct sk\_buff \*skb)  
This function is invoked at SoftIRQ level to handle incoming packets.

### Parameters

**struct sk\_buff \*skb**  
The incoming packet.

### Return

Always 0

int **MTP\_backlog\_rcv**(struct sock \*sk, struct sk\_buff \*skb)  
Invoked to handle packets saved on a socket's backlog because it was locked when the packets first arrived.

### Parameters

**struct sock \*sk**  
MTP socket that owns the packet's destination port.

**struct sk\_buff \*skb**  
The incoming packet. This function takes ownership of the packet (we'll delete it).

**Return**

Always returns 0.

int **MTP\_err\_handler\_v4**(struct sk\_buff \*skb, u32 info)

Invoked by IP to handle an incoming error packet, such as ICMP UNREACHABLE.

**Parameters**

**struct sk\_buff \*skb**

The incoming packet.

**u32 info**

Information about the error that occurred?

**Return**

zero, or a negative errno if the error couldn't be handled here.

int **MTP\_err\_handler\_v6**(struct sk\_buff \*skb, struct inet6\_skb\_parm \*opt, u8 type, u8 code, int offset, \_\_be32 info)

Invoked by IP to handle an incoming error packet, such as ICMP UNREACHABLE.

**Parameters**

**struct sk\_buff \*skb**

The incoming packet.

**struct inet6\_skb\_parm \*opt**

options

**u8 type**

type

**u8 code**

code

**int offset**

offset

**\_\_be32 info**

Information about the error that occurred?

**Return**

zero, or a negative errno if the error couldn't be handled here.

\_\_poll\_t **MTP\_poll**(struct *file* \*file, struct socket \*sock, struct poll\_table\_struct \*wait)

Invoked by Linux as part of implementing select, poll, epoll, etc.

**Parameters**

**struct file \*file**

Open file that is participating in a poll, select, etc.

**struct socket \*sock**

A MTP socket, associated with *file*.

**struct poll\_table\_struct \*wait**

This table will be registered with the socket, so that it is notified when the socket's ready state changes.

**Return**

A mask of bits such as EPOLLIN, which indicate the current state of the socket.

int **MTP\_metrics\_open**(struct *inode* \*inode, struct *file* \*file)

This function is invoked when /proc/net/MTP\_metrics is opened.

### Parameters

**struct inode \*inode**

The inode corresponding to the file.

**struct file \*file**

Information about the open file.

### Return

always 0.

**ssize\_t MTP\_metrics\_read** (struct file \*file, char \_\_user \*buffer, size\_t length, loff\_t \*offset)

This function is invoked to handle read kernel calls on /proc/net/MTP\_metrics.

### Parameters

**struct file \*file**

Information about the file being read.

**char \_\_user \*buffer**

Address in user space of the buffer in which data from the file should be returned.

**size\_t length**

Number of bytes available at **buffer**.

**loff\_t \*offset**

Current read offset within the file.

### Return

the number of bytes returned at **buffer**. 0 means the end of the file was reached, and a negative number indicates an error (-errno).

**loff\_t MTP\_metrics\_lseek**(struct *file* \*file, loff\_t offset, int whence)

This function is invoked to handle seeks on /proc/net/MTP\_metrics. Right now seeks are ignored: the file must be read sequentially.

### Parameters

**struct file \*file**

Information about the file being read.

**loff\_t offset**

Distance to seek, in bytes

**int whence**

Starting point from which to measure the distance to seek.

int **MTP\_metrics\_release**(struct *inode* \*inode, struct *file* \*file)

This function is invoked when the last reference to an open /proc/net/MTP\_metrics is closed. It performs cleanup.

### Parameters

**struct inode \*inode**

The inode corresponding to the file.

**struct file \*file**

Information about the open file.

**Return**

always 0.

**int MTP\_dointvec (struct ctl\_table \*table, int write, void \_\_user \*buffer, size\_t \*lenp, loff\_t \*ppos)**

This function is a wrapper around proc\_dointvec. It is invoked to read and write sysctl values and also update other values that depend on the modified value.

**Parameters**

**struct ctl\_table \*table**

sysctl table describing value to be read or written.

**int write**

Nonzero means value is being written, 0 means read.

**void \_\_user \*buffer**

Address in user space of the input/output data.

**size\_t \*lenp**

Not exactly sure.

**loff\_t \*ppos**

Not exactly sure.

**Return**

0 for success, nonzero for error.

**int MTP\_sysctl\_softirq\_cores (struct ctl\_table \*table, int write, void \_\_user \*buffer, size\_t \*lenp, loff\_t \*ppos)**

This function is invoked to handle sysctl requests for the “gen3\_softirq\_cores” target, which requires special processing.

**Parameters**

**struct ctl\_table \*table**

sysctl table describing value to be read or written.

**int write**

Nonzero means value is being written, 0 means read.

**void \_\_user \*buffer**

Address in user space of the input/output data.

**size\_t \*lenp**

Not exactly sure.

**loff\_t \*ppos**

Not exactly sure.

**Return**

0 for success, nonzero for error.

enum hrtimer\_restart **MTP\_hrtimer**(struct hrtimer \*timer)

This function is invoked by the hrtimer mechanism to wake up the timer thread. Runs at IRQ level.

**Parameters**

**struct hrtimer \*timer**

The timer that triggered; not used.

### Return

Always HRTIMER\_RESTART.

int **MTP\_timer\_main**(void \*transportInfo)

Top-level function for the timer thread.

### Parameters

void \***transportInfo**

Pointer to struct MTP.

### Return

Always 0.

void **MTP\_sock\_init**(struct MTP\_sock \*mtpsk, struct MTP \*mtpinst)

Constructor for MTP\_sock objects. This function initializes only the parts of the socket that are owned by MTP.

### Parameters

struct MTP\_sock \***mtpsk**

Object to initialize.

struct MTP \***mtpinst**

MTP implementation that will manage the socket.

### Return

always 0 (success).

struct MTP \***MTP\_init**(void)

Constructor for MTP object. This function initializes MTP data structure.

### Parameters

void

no arguments

### Return

pointer to the MTP object.

## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`





## INDEX

### M

MTP\_backlog\_rcv (*C function*), 6  
MTP\_bind (*C function*), 1  
MTP\_close (*C function*), 2  
MTP\_diag\_destroy (*C function*), 6  
MTP\_disconnect (*C function*), 2  
MTP\_err\_handler\_v4 (*C function*), 7  
MTP\_err\_handler\_v6 (*C function*), 7  
MTP\_get\_port (*C function*), 5  
MTP\_hash (*C function*), 5  
MTP\_hrtimer (*C function*), 9  
MTP\_init (*C function*), 10  
MTP\_ioc\_abort (*C function*), 2  
MTP\_ioctl (*C function*), 3  
MTP\_load (*C function*), 1  
MTP\_metrics\_lseek (*C function*), 8  
MTP\_metrics\_open (*C function*), 7  
MTP\_metrics\_release (*C function*), 8  
MTP\_poll (*C function*), 7  
MTP\_recvmmsg (*C function*), 4  
MTP\_rehash (*C function*), 5  
MTP\_sendmsg (*C function*), 4  
MTP\_sendpage (*C function*), 5  
MTP\_setsockopt (*C function*), 3  
MTP\_shutdown (*C function*), 2  
MTP\_sock\_init (*C function*), 10  
MTP\_socket (*C function*), 3  
MTP\_softirq (*C function*), 6  
MTP\_timer\_main (*C function*), 10  
MTP\_unhash (*C function*), 5  
MTP\_v4\_early\_demux (*C function*), 6  
MTP\_v4\_early\_demux\_handler (*C function*), 6