MTP: Mesh Transport Protocol

Release 0.0.1

Ertan Onur

Dec 08, 2023

CONTENTS:

CHAPTER

ONE

MTP

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 getsockopt 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.

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

INDEX