IGMPv3-MLDv2 Multicast Technical Reference

Interniche Legacy Document

Version 1.00

Date: 17-May-2017 16:24

Table of Contents

1 Introduction

The IGMPv3 and MLDv2 modules are platform independent implementations of the Internet Group Management Protocol (IGMP) specification version 3 (RFC 3376) and the Multicast Listener Discovery Protocol (MLD) specification (RFC3810) with the most significant difference being that IGMP is used to manage membership in IPv4 multicast groups, and MLD is used to manage membership in IPv6 multicast groups.

These protocols are used to IP multicast group memberships to neighboring multicast routers. IGMPv3 and MLDv2 provide support for source filtering, the ability for a system to report interest in receiving packets "only" from specific source addresses, or from "all but" specific source addresses, sent to a particular multicast address. This information is used by multicast routing protocols to avoid delivering multicast packets from specific sources to networks where there are no interested receivers.

The reader of this document can assume that unless explicitly stated otherwise, terminology, usage and descriptions are pertinent to both IGMPv3 and MLDv2.

1.1 Features

- Implements IGMP version 3 and MLD version 2 host-side functionality
- Implements multicast socket option APIs as described in RFCs 3493 and 3678
- IGMPv3 is backward compatible with IGMP version 1 and IGMP version 2 and will automatically "step down" to those specifications in order to be compatible with routers supporting older versions of the IGMP protocol.
- MLDv2 is backward compatible with MLD version 1 (RFC 2710), and will automatically operate in that mode to be compatible with routers that support older MLD versions.
- Implements source filtering
- Supports IP and IPv6 router alert options (RFC2113 and RFC2711, respectively)

2 Configuration

For IGMPv3, the `ipport.h` file should enable the following compile time options:

- `USE_IGMPv3`
- `IP_MULTICAST`
- `USE_IGMPV1`
- `USE_IGMPV2`
- `IGMP_MENUS`
- `UDP_SKIP_LCL_ADDR_CHECK`
- `ETHMCAST`
- `IGMP_DEBUG`

For MLDv2, the `ipport.h` file should enable the following compile time options:

- `USE_MLDV2`
- `IP_MULTICAST`
- `USE_MLD`
- `UDP6_SKIP_LCL_ADDR_CHECK`
- `MLD_DEBUG`

3 User API

3.1 Data Structures

The following is a list of data structures that are used as parameters to the **t_setsockopt()** call when configuring multicast socket options:

```
/* IP_ADD_MEMBERSHIP and IP_DROP_MEMBERSHIP (IPv4) */
struct ip_mreq
{
   ip_addr imr_multiaddr;               /* IP multicast address of group */
   ip_addr imr_interface;               /* local IP address of interface */
};

/* IP_ADD_SOURCE_MEMBERSHIP, IP_DROP_SOURCE_MEMBERSHIP, IP_BLOCK_SOURCE,
 * and IP_UNBLOCK_SOURCE (IPv4)
 */
struct ip_mreq_source
{
   ip_addr imr_multiaddr;               /* IP multicast address of group */
   ip_addr imr_sourceaddr;              /* source address of group */
   ip_addr imr_interface;               /* local IP address of interface */
};
/* IP_MSFILTER (IPv4) */
struct msfilter
{
   ip_addr intfaddr;                    /* Interface address */
   ip_addr multaddr;                    /* Multicast group address */
   uint32_t fmode;                      /* Requested mode */
   uint32_t numsrc;                     /* Number of sources in source list */
   ip_addr *slist;                      /* Source List */
};

/* IPV6_JOIN_GROUP and IPV6_LEAVE_GROUP (IPv6) */
struct ipv6_mreq
{
   struct in6_addr ipv6mr_multiaddr;    /* IPv6 multicast addr */
   unsigned int    ipv6mr_interface;    /* interface index */
};

/* MCAST_JOIN_GROUP and MCAST_LEAVE_GROUP (IPv4 and IPv6) */
struct group_req
{
   u_long gr_interface;                 /* interface index */
   struct sockaddr_storage gr_group;    /* group address */
};

/* MCAST_JOIN_SOURCE_GROUP, MCAST_LEAVE_SOURCE_GROUP, MCAST_BLOCK_SOURCE,
 * and MCAST_UNBLOCK_SOURCE (IPv4 and IPv6)
 */
struct group_source_req
{
```

```
   u_long gsr_interface;                /* interface index */
   struct sockaddr_storage gsr_group;   /* group address */
   struct sockaddr_storage gsr_source;  /* source address */
};

/* IP_MSFILTER_PI (IPv4 and IPv6) */
struct group_filter
{
   uint32_t               gf_interface; /* interface index */
   struct sockaddr_storage gf_group;    /* multicast address */
   uint32_t               gf_fmode;     /* filter mode */
   uint32_t               gf_numsrc;    /* number of sources */
   struct sockaddr_storage *gf_slist;   /* source address */
};
```

Note: All data structures that use a numeric interface index require one-based indices.

3.2 Functions and Socket Options

getipv4sourcefilter

Name

`getipv4sourcefilter - Get Multicast Group Information`

Syntax

```
int
getipv4sourcefilter(int s, struct in_addr iface, struct in_addr multaddr,
                    unsigned int *fmode, unsigned int *numsrcp,
                    struct in_addr *slist);
```

Parameters

s	socket id
iface	IPv4 address of ingress interface
multaddr	IPv4 multicast group address
fmode	pointer to location where filter mode will be copied
numsrcp	pointer to location where the number of sources in the filter will be copied
slist	pointer to location where the source addresses in the filter will be copied

Description

This function is used to obtain information about a socket's membership in the specified IPv4 multicast group on the specified interface.

Returns

This function returns `SOCKET_ERROR` (-1) if it encounters an error; otherwise, it returns ESUCCESS (0).

getsourcefilter

Name

`getsourcefilter - Get Multicast Group Information`

Syntax

```
int
getsourcefilter(int s, unsigned int iface, struct sockaddr *multaddr,
                int multaddrlen, unsigned int *fmode, unsigned int *numsrcp,
                struct sockaddr_storage *slist);
```

Parameters

s	socket id
iface	interface identifier (one-based)
multaddr	pointer to multicast group address
multaddrlen	length of the multicast group address
fmode	pointer to location where filter mode will be copied
numsrcp	pointer to location where the number of sources in the filter will be copied
slist	pointer to location where the source addresses in the filter will be copied

Description

This function is used to obtain information about a socket's membership in the specified (IPv4 or IPv6) multicast group on the specified interface.

Returns

This function returns SOCKET_ERROR (-1) if it encounters an error; otherwise, it returns ESUCCESS (0).

setipv4sourcefilter

Name

`setipv4sourcefilter - Configure membership in a multicast group`

Syntax

```
int
setipv4sourcefilter(int s, struct in_addr iface, struct in_addr multaddr,
                    unsigned int fmode, unsigned int numsrc,
                    struct in_addr *slist);
```

Parameters

s	socket id
iface	IPv4 address of ingress interface
multaddr	IPv4 multicast group address
fmode	filter mode (MCAST_INCLUDE or MCAST_EXCLUDE)
numsrc	number of source addresses in filter
slist	pointer to location where the source addresses in the filter are stored

Description

This function is used to configure a socket's membership in the specified IPv4 multicast group on the specified interface to receive or drop packets from the specified list of source addresses.

Returns

This function returns SOCKET_ERROR (-1) if it encounters an error; otherwise, it returns ESUCCESS (0).

setsourcefilter

Name

`setsourcefilter - Configure membership in a multicast group`

Syntax

```
int
setsourcefilter(int s, unsigned int iface, struct sockaddr *multaddr,
                int multaddrlen, unsigned int fmode, unsigned int numsrc,
                struct sockaddr_storage *slist);
```

Parameters

s	socket id
iface	interface identifier (one-based)
multaddr	pointer to multicast group address
multaddrlen	length of the multicast group address
fmode	filter mode (MCAST_INCLUDE or MCAST_EXCLUDE)
numsrc	number of source addresses in filter
slist	pointer to location where the source addresses in the filter are stored

Description

This function is used to configure a socket's membership in the specified IPv4 or IPv6 multicast group on the specified interface to receive or drop packets from the specified list of source addresses.

Returns

This function returns SOCKET_ERROR (-1) if it encounters an error; otherwise, it returns ESUCCESS (0).

t_setsockopt(), t_getsockopt()

In addition to the set of socket options supported by the main NicheStack code, the following options are supported:

Option	Get	Set	"optval"	Description	Address Family
IP_ADD_MEMBERSHIP	No	Yes	ip_mreq	Join the socket to the supplied multicast group on the specified interface. A single socket can join several multicast groups. The limit to this is `IP_MAX_MEMBERSHIPS` (i.e. 20).	IPv4
IP_DROP_MEMBERSHIP	No	Yes	`ip_mreq`	Leaves the specified multicast group from the specified interface.	IPv4
IP_ADD_SOURCE_ MEMBERSHIP	No	Yes	`ip_mreq_source`	Join the supplied multicast group on the given interface and accept data sourced from the supplied source address.	IPv4
IP_DROP_SOURCE_ MEMBERSHIP	No	Yes	`ip_mreq_source`	Drop membership to the given multicast group, interface, and source address.	IPv4
IP_BLOCK_SOURCE	No	Yes	`ip_mreq_source`	Block multicast packets whose source address matches the specified source address. The specified group must be joined previously using `IP_ADD_MEMBERSHIP` or `MCAST_JOIN_GROUP` .	IPv4

Option	Get	Set	"optval"	Description	Address Family
IP_UNBLOCK_SOURCE	No	Yes	`ip_mreq_source`	Unblock (begin receiving) multicast packets which were previously blocked using `IP_BLOCK_SOURCE`.	IPv4
MCAST_JOIN_GROUP	No	Yes	`group_req`	Join a multicast group. Functionally equivalent to `IP_ADD_MEMBERSHIP`.	IPv4 and IPv6
MCAST_LEAVE_GROUP	No	Yes	`group_req`	Leave a multicast group. Functionally equivalent to `IP_DROP_MEMBERSHIP`	IPv4 and IPv6
MCAST_BLOCK_SOURCE	No	Yes	`group_source_req`	Block multicast packets whose source address matches the given source address. The specified group must be joined previously using `IP_ADD_MEMBERSHIP` or `MCAST_JOIN_GROUP` .	IPv4 and IPv6
MCAST_UNBLOCK_SOURCE	No	Yes	`group_source_req`	Unblock (begin receiving) multicast packets which were previously blocked using `MCAST_BLOCK_SOURCE` .	IPv4 and IPv6
MCAST_JOIN_SOURCE_ GROUP	No	Yes	`group_source_req`	Begin receiving packets for the given multicast group whose source address matches the specified address.	IPv4 and IPv6

Option	Get	Set	"optval"	Description	Address Family
MCAST_LEAVE_SOURCE_GROUP	No	Yes	`group_source_req`	Stop receiving packets for the given multicast group whose source address matches the specified address.	IPv4 and IPv6
IP_MULTICAST_IF	Yes	Yes	`ip_addr`	Specify egress interface for outgoing multicast datagrams	IPv4
IP_MULTICAST_LOOP	Yes	Yes	`u_char`	Enable or disable loopback of outgoing multicast datagrams	IPv4
IP_MULTICAST_TTL	Yes	Yes	`u_char`	Specify the Time to Live value for use in IPv4 header of outgoing multicast datagram	IPv4
IPV6_MULTICAST_HOPS	Yes	Yes	int	Specify the Hop Limit value for use in IPv6 header of outgoing multicast datagram (as per RFC3493)	IPv6
IPV6_MULTICAST_IF	Yes	Yes	unsigned int	Specify (ones-based) egress interface for outgoing multicast datagrams (as per RFC3493)	IPv6
IPV6_MULTICAST_LOOP	Yes	Yes	unsigned int	Enable or disable loopback of outgoing multicast datagrams (as per RFC3493)	IPv6
IPV6_JOIN_GROUP	No	Yes	ipv6_mreq	Join a multicast group on the specified (ones-based) interface (as per RFC3493)	IPv6
IPV6_LEAVE_GROUP	No	Yes	ipv6_mreq	Leave a multicast group on the specified (ones-based) interface (as per RFC3493)	IPv6

Option	Get	Set	"optval"	Description	Address Family
IP_MSFILTER	Yes	Yes	msfilter	Specify the complete set of sender (IPv4) addresses whose datagrams to the specified (IPv4) multicast address are either accepted or blocked (reference RFC3678)	IPv4
IP_MSFILTER_PI	Yes	Yes	group_filter	Specify the complete set of sender (IPv4 or IPv6) addresses whose datagrams to the specified (IPv4 or IPv6) multicast address are either accepted or blocked	IPv4 and IPv6

Note: All of the socket options described in this section require the 'level' parameter in `t_setsockopt()` or `t_getsockopt()` to be set to `IPPROTO_IP`.

Please see the manual entries for `t_setsockopt()` and `t_getsockopt()` in the main NicheStack Reference manual.

3.3 Multicast Menu Commands

IGMPv3 Commands

igmp add

Command Name

`igmp add - join an IP multicast group`

Syntax

```
igmp add -g <multicast group> -i <interface name or id> -s <socket> [-x] [-y] [-m] [-a] [-o <IPv4 address(es)>]
```

Parameters

`-a`	Indicates that the addresses specified are allowed (when present), or blocked (when absent).
`-g`	Multicast group address.
`-i`	Interface identifier.
`-m`	Use MCAST_xxx set of socket options for multicast membership changes.
`-o`	List of allowed or blocked source IPv4 addresses.
`-s`	Socket identifier.
`-x`	use setipv4sourcefilter() API to configure multicast membership information.
`-y`	use setsourcefilter() API to configure multicast membership information.

Description

This command is used to join a IPv4 multicast group on an interface via a (previously created) UDP/IPv4 socket. The source filtering features are only available if IGMPv3 is enabled in the system.

Notes/Status

- The multicast group address must be specified in dotted-decimal notation.
- The interface can be specified either via its name or its (one-based) index.
- The IGMP protocol that will operate on the link must be configured via the "igmp mode" CLI command.
- The socket identifier is obtained from the output of a "igmp sock -c" CLI command.
- The use of the -m, -x, and -y options is optional. In the absence of these options, the membership will be updated via the IP_xxx series of socket options. Only one of the -m, -x, and -y options can be used in one invocation of this command.

Example invocation sequences

- 1. use IP_ADD_MEMBERSHIP to join 224.0.0.77 on interface 1 via socket 0xba02d4

```
igmp add -g 224.0.0.77 -i 1 -s 0xba02d4
```

2. use IP_ADD_SOURCE_MEMBERSHIP to join 224.0.0.77 on interface 1 via socket 0xba02d4, and accept packets from 10.0.0.77

```
igmp add -g 224.0.0.77 -i 1 -s 0xba044c -a -o 10.0.0.77
```

3. use MCAST_JOIN_GROUP to join 224.0.0.77 on interface 1 via socket 0xba044c

```
igmp add -g 224.0.0.77 -i 1 -s 0xba044c -m
```

4. use MCAST_JOIN_SOURCE_GROUP to start accepting packets from 10.0.0.77 and destined to 224.0.0.77 on interface 1

```
igmp add -g 224.0.0.77 -i 1 -s 0xba02d4 -a -o 10.0.0.77 -m
```

5. use MCAST_BLOCK_SOURCE to block reception of packets from 10.0.0.77 and destined to 224.0.0.77 on interface 1 (after joining the group in EXCLUDE(none) mode earlier (e.g., via `MCAST_JOIN_GROUP`))

```
igmp add -g 224.0.0.77 -i 1 -s 0xba02d4 -o 10.0.0.77 -m
```

6. use setipv4sourcefilter() to join 224.0.0.77 on interface 1, and accept packets from two sources (10.0.0.77 and 10.0.0.88)

```
igmp add -g 224.0.0.77 -i 1 -s 0xba02d4 -a -o "10.0.0.77 10.0.0.88" -x
```

7. use setsourcefilter() to join 224.0.0.77 on interface 1, and accept packets from two sources (10.0.0.77 and 10.0.0.88)

```
igmp add -g 224.0.0.77 -i 1 -s 0xba02d4 -a -o "10.0.0.77 10.0.0.88" -y
```

Location

This command is provided by the `IGMP` module when `IP_MULTICAST` and at least one of `USE_IGMPV1`, `USE_IGMPV2`, or `USE_IGMPV3` is defined.

igmp cfg

Command Name

`igmp cfg - configure IGMPv3 protocol parameters`

Syntax

```
igmp cfg -i <interface name or id> {-r <#transmissions> | -u <#seconds>}
```

Parameters

`-r`	configure Robustness Variable.
`-u`	configure Unsolicited Report Interval.

Description

This command is used to configure parameters for the IGMPv3 protocol.

Notes/Status

- The existing implementation does not support the configuration of Robustness Variable and Unsolicited Report Interval for IGMPv2.

Location

This command is provided by the `IGMP` module when `IP_MULTICAST` and `USE_IGMPV3` are defined.

igmp drop

Command Name

`igmp drop - unsubscribe from an IP multicast group`

Syntax

```
igmp add -g <multicast group> -i <interface name or id> -s <socket> [-m] [-a] [-o <IPv4 address(es)>]
```

Parameters

`-a`	Indicates that the addresses specified are allowed (when present), or blocked (when absent).
`-g`	Multicast group address.
`-i`	Interface identifier.
`-m`	Use MCAST_xxx set of socket options for multicast membership changes.
`-o`	List of allowed or blocked source IPv4 addresses.
`-s`	Socket identifier.

Description

This command is used to drop out of a IPv4 multicast group on an interface via a UDP/IPv4 socket. The source filtering features are only available if IGMPv3 is enabled in the system.

Notes/Status

- The multicast group address must be specified in dotted-decimal notation.
- The interface can be specified either via its name or its (one-based) index.
- The socket identifier is obtained from the output of a "igmp sock -c" CLI command.
- In the absence of the -m option, the membership will be updated via the IP_xxx series of socket options.

Example invocation sequences

- 1. use IP_DROP_MEMBERSHIP to stop receiving packets destined to 224.0.0.77 on interface 1 via socket 0xba02d4

     ```
     igmp drop -g 224.0.0.77 -i 1 -s 0xba02d4
     ```

  2. use IP_DROP_SOURCE_MEMBERSHIP to stop receiving packets from 10.0.0.77 and destined to 224.0.0.77 on interface 1 via socket 0xba02d4

     ```
     igmp drop -g 224.0.0.77 -i 1 -s 0xba02d4 -a -o 10.0.0.77
     ```

  3. use MCAST_LEAVE_GROUP to stop receiving packets destined to 224.0.0.77 on interface 1 via socket 0xba044c

     ```
     igmp drop -g 224.0.0.77 -i 1 -s 0xba044c -m
     ```

  4. use MCAST_LEAVE_SOURCE_GROUP to stop accepting packets from 10.0.0.77 and destined to 224.0.0.77 on interface 1

     ```
     igmp drop -g 224.0.0.77 -i 1 -s 0xba02d4 -a -o 10.0.0.77 -m
     ```

  5. use MCAST_UNBLOCK_SOURCE to unblock reception of packets from 10.0.0.77 and destined to 224.0.0.77 on interface 1

     ```
     igmp drop -g 224.0.0.77 -i 1 -s 0xba02d4 -o 10.0.0.77 -m
     ```

Location

This command is provided by the `IGMP` module when `IP_MULTICAST` and at least one of `USE_IGMPV1`, `USE_IGMPV2`, or `USE_IGMPV3` is defined.

igmp mode

Command Name

`igmp mode - specify IGMP version`

Syntax

```
igmp mode {-i <interface> -v <1 | 2 | 3>}
```

Parameters

`-i`	Specify interface whose IGMP administrative mode (version) is being set.
`-v`	Specify IGMP version to run.

Description

This command sets the IGMP administrative mode (version) of a particular interface.

Notes/Status

- The interface can be specified either via its name or its (one-based) index.
- The operational mode of an interface may be lower than its administratively-specified mode.

Location

This command is provided by the `IGMP` module when `IP_MULTICAST` and at least one of `USE_IGMPV1`, `USE_IGMPV2`, or `USE_IGMPV3` is defined.

igmp netstat

Command Name

`igmp netstat - display IGMP statistics and status`

Syntax

```
igmp netstat
```

Parameters

None

Description

This command displays statistics associated with the IGMP module.

Notes/Status

- The statistics are not collected on a per-interface basis.

Location

This command is provided by the `IGMP` module when `IP_MULTICAST` and at least one of `USE_IGMPV1`, `USE_IGMPV2`, or `USE_IGMPV3` is defined.

igmp sock

Command Name

`igmp sock - create, delete, or display multicast information for a UDP/IPv4 socket`

Syntax

```
igmp sock {-c | {{-d | -p} -s <socket>} | {{-x | -y} -g <multicast group> -i <interface name or id> -s <socket>}}
```

Parameters

`-c`	create a UDP/IPv4 socket.
`-d`	delete a socket.
`-p`	print multicast-related information for socket.
`-x`	use getipv4sourcefilter() API to print multicast membership information.
`-y`	use getsourcefilter() API to print multicast membership information.

Description

This command is used to create a UDP/IPv4 socket (for use in subsequent multicast operations), delete a socket, or display multicast-related information for a socket.

Notes/Status

- When used with the -c option, this commands prints out an identifier for the newly created socket. This identifier is used in subsequent "igmp add" and "igmp drop" CLI commands.

Example invocation sequences

- 1. create UDP/IPv4 socket

     ```
     igmp sock -c
     ```

  2. delete UDP/IPv4 socket 0xba02d4

     ```
     igmp sock -d -s 0xba02d4
     ```

  3. print membership information for socket 0xba02d4

     ```
     igmp sock -p -s 0xba02d4
     ```

  4. print membership information (via getipv4sourcefilter()) for 224.0.0.77 on interface 1 for socket 0xba02d4

     ```
     igmp sock -p -g 224.0.0.77 -i 1 -s 0xba02d4 -x
     ```

  5. print membership information (via getsourcefilter()) for 224.0.0.77 on interface 1 for socket 0xba02d4

     ```
     igmp sock -p -g 224.0.0.77 -i 1 -s 0xba02d4 -y
     ```

Location

This command is provided by the `IGMP` module when `IP_MULTICAST` and at least one of `USE_IGMPV1`, `USE_IGMPV2`, or `USE_IGMPV3` is defined.

MLDv2 Commands

mld

Command Name

mld - Configure the Multicast Listener Discovery protocol (MLD (RFC 2710) and MLDv2 (RFC 3810))

Syntax

```
mld {-c}
    {-d -s <socket id>}
    {-e -i <interface id> {{-r <# of transmissions>} | {-t 0 | 1} | {-u <time (s

    {-j -g <IPv6 address%scopeid> -s <socket id> {[-m [[-a] -o <IPv6 address>]]

    {-l -g <IPv6 address%scopeid> -s <socket id> [-m [[-a] -o <IPv6 address>]]}
    {-p -s <socket id> [-y -g <IPv6 address%scopeid>]}
    {-z}
```

Parameters

`-a`	Indicates that the addresses specified are allowed when present, or blocked when absent (for MLDv2 only).
`-c`	Create a UDP/IPv6 socket for use in subsequent multicast test operations.
`-d`	Delete a UDP/IPv6 socket.
`-e`	Configure MLD version or parameter for the specified interface.
`-g`	Specify the multicast group address and interface identifier.
`-i`	Interface identifier.
`-j`	Add membership in specified multicast group on interface to socket.
`-l`	Delete membership in specified multicast group on interface from socket.
`-m`	Add or drop multicast membership via the MCAST_xxx set of socket options when present, or via IPV6_xxx options or setsourcefilter() when absent.
`-o`	List of allowed or blocked source IPv4 addresses (for MLDv2 only).
`-p`	Print multicast information for socket.
`-r`	Robustness Variable (for MLDv2 only).
`-s`	Socket identifier.
`-t`	Enable (1) or disable (0) the optimization related to the transmission of the Multicast Listener Done message.
`-u`	Configure Unsolicited Report Interval (in units of seconds).
`-v`	MLD version number (1 for MLD, 2 for MLDv2).
`-y`	use getsourcefilter() to obtain multicast membership information, or use setsourcefilter() to update membership information (for MLDv2 only).
`-z`	Display MLD statistics.

Description

This command can be used to (1) configure the MLD protocol, (2) display status and statistics information for the protocol, and (3) perform test operations (e.g., add group or leave group) to initiate MLD signaling (e. g., send Multicast Listener Report or Multicast Listener Done message).

Example invocation sequences

- 1. configure MLD version number on interface 'nd0' to MLDv2

     ```
     mld -e -i nd0 -v 2
     ```

  2. create UDP/IPv6 socket

     ```
     mld -c
     ```

  3. delete UDP/IPv6 socket

     ```
     mld -d -s 0xba144c
     ```

  4. use IPV6_JOIN_GROUP to join ff02::7777 on interface 1 via socket 0xba144c

     ```
     mld -j -g ff02::7777%1 -s 0xba144c
     ```

  5. use IPV6_LEAVE_GROUP to leave ff02::7777 on interface 1 from socket 0xba144c

     ```
     mld -l -g ff02::7777%1 -s 0xba144c
     ```

  6. use setsourcefilter() to join ff02::7777 on interface 1 via socket 0xba144c

     ```
     mld -j -g ff02::7777%1 -s 0xba144c -y
     ```

  7. use setsourcefilter() to join ff02::7777 on interface 1, and accept packets from two sources (fe80::240:f4ff:feed:8b77 and fe80::211:2fff:fe1a:5518)

     ```
     mld -j -g ff02::7777%1 -s 0xba144c -a -o "fe80::240:f4ff:feed:8b77 fe8
     ```

  8. use MCAST_JOIN_GROUP to join ff02::7777 on interface 1 via socket 0xba144c

     ```
     mld -j -g ff02::7777%1 -s 0xba144c -m
     ```

  9. use MCAST_LEAVE_GROUP to leave ff02::7777 on interface 1 via socket 0xba144c

     ```
     mld -l -g ff02::7777%1 -s 0xba144c -m
     ```

  10. use MCAST_JOIN_SOURCE_GROUP to start accepting packets from fe80::240:f4ff:feed:8b77 and destined to ff02::7777 on interface 1

      ```
      mld -j -g ff02::7777%1 -s 0xba144c -a -o fe80::240:f4ff:feed:8b77 -m
      ```

  11. use MCAST_LEAVE_SOURCE_GROUP to stop accepting packets from fe80::240:f4ff:feed:8b77 and destined to ff02::7777 on interface 1

      ```
      mld -l -g ff02::7777%1 -s 0xba144c -a -o fe80::240:f4ff:feed:8b77 -m
      ```

12. join ff02::7777 in EXCLUDE(none) mode via IPV6_JOIN_GROUP

    ```
    mld -j -g ff02::7777%1 -s 0xba144c
    ```

    or, use MCAST_JOIN_GROUP to join ff02::7777 in EXCLUDE(none) mode

    ```
    mld -j -g ff02::7777%1 -s 0xba144c -m
    ```

    having joined by either of the preceding methods, use MCAST_BLOCK_SOURCE to block reception of packets from fe80::240:f4ff:feed:8b77 and destined to ff02::7777 on interface 1

    ```
    mld -j -g ff02::7777%1 -s 0xba144c -o fe80::240:f4ff:feed:8b77 -m
    ```

    use MCAST_UNBLOCK_SOURCE to unblock reception of packets from fe80::240:f4ff:feed:8b77 and destined to ff02::7777 on interface 1

    ```
    mld -l -g ff02::7777%1 -s 0xba144c -o fe80::240:f4ff:feed:8b77 -m
    ```

13. print membership information for ff02::7777 on interface 1 for socket 0xba144c

    ```
    mld -p -g ff02::7777%1 -s 0xba144c -y
    ```

14. print membership information for socket 0xba144c

    ```
    mld -p -s 0xba144c
    ```

Location

This command is provided by the `IPv6` module when `IP_V6`, `IP_MULTICAST`, and `USE_MLD` are defined. Some options are specific to MLDv2, and require `USE_MLDV2` to be defined. (MLDv2 requires MLD to be enabled for correct operation.)

4 Usage and Product Notes

- `ETHMCAST` - To map an IPv4 multicast destination address to its corresponding Ethernet multicast address (`01-00-5e-xx-xx-xx`) [RFC 1112], `ETHMCAST` must be enabled in the stack's configuration header file. Otherwise, packets with an IPv4 multicast destination address are transmitted with the Ethernet destination address set to the broadcast address (`ff-ff-ff-ff-ff-ff`).
- `UDP_SKIP_LCL_ADDR_CHECK and UDP6_SKIP_LCL_ADDR_CHECK` - Enabling the UDP_SKIP_LCL_ADDR_CHECK orvUDP6_SKIP_LCL_ADDR_CHECK compile-time directives will cause the TCP/IP stack to skip checking that the address being bound is the address of a local interface, thereby allowing the application to bind to specific multicast group addresses.
- Applications intending to restrict multicasting to the local network need only enable `IP_MULTICAST` in `ipport.h`. This will allow the applications to register (for reception) multicast addresses of interest on a particular link with the corresponding device driver for that link. This #define also provides the capability to send packets to IPv4 or IPv6 multicast destination addresses.
- To send and receive IPv4 or IPv6 multicast packets to and from off-link destinations in addition to hosts on the local network, IGMP or MLD must be enabled on that link.

5 Examples

The IPv4 directory ('ip') contains the 'ip4mctest.c' file. This file contains two functions that illustrate the reception (udp4mcrx()) and transmission (udp4mctx()) of multicast UDP/IPv4 datagrams.

The IPv6 directory ('ipv6') contains the 'ip6mctest.c' file. This file contains a sample UDP/IPv6 multicast application. It receives multicast UDP/IPv6 datagrams addressed to ff12::7777 (mcgrp_addr) and port number U6DSTPORT (0x1001). It then sends them out to ff12::8888 (fwd_addr) and port number U6FWDPORT (0x4001).

Additional examples on how to use the source filtering APIs are available in ipmc/igmp_nt.c and ipv6/ipv6_nt.c.