IGMPv3

IGMPv3 for IPv4 allows receiver applications to specify source filters on multicast group memberships, so that applications only receive multicast packets from specified sources. The source filters can exclude or include source addresses. This capability is desired in the case of one-to-many type multicast applications.
To support source filtering of multicast packets new socket APIs have been added.
The current IP Multicast APIs allow a receiver application to specify the multicast group address (destination) and (optionally) the local interface.
The new source filter socket APIs provide the same functionality but also allow receiver multicast applications to:

- Specify zero or more unicast (source) address(es) in a source filter.
- Determine whether the source filter describes an inclusive or exclusive list of sources.

Two modes of setting source filters are provided: “include”, and “exclude” modes. The user cannot mix include and exclude modes. At any given time, the source filter list for a given socket, interface, and multicast group address is either inclusive, or exclusive.

Source filter mode	*Comments*
include	For a given multicast group address, and interface, the user accepts multicasts on the socket from sources addresses on the list
exclude	For a given multicast group address, and interface, the user accepts multicasts on the socket from all source addresses except the ones on the list.

Two sets of APIs are provided as specified in RFC 3678:

- Basic (delta based) APIs using the **setsockopt** BSD socket API.
- Advanced (Full-State) APIs which allow the user to define a complete source-filter comprised of zero or more source-addresses, and replace the previous filter with a new one.

API	*Type of APIs*	*Comments*
setsockopt	Basic	Cannot do getsockopt. Add filter addresses one at a time. Can use exclusive or inclusive mode. Must leave the group to switch between inclusive and exclusive mode.
setipv4sourcefilter	Advanced	Can specify complete source filter list with one call. Can replace a source

		filter list with a new one. Can switch between inclusive and exclusive mode without leaving the group.
getipv4sourcefilter	Advanced	Application can retrieve source filter list.
tfIoctl	Advanced	Same functionality as setipv4sorucefilter, and getipv4sourcefilter. Obsoleted by setipv4sourcefilter and getipv4sourcefilter. Provided for backward compatibility.

Implementation Notes

- Because IGMPv3 is added, and MLDv2 is not, only the IPv4 APIs as specified in RFC 3678 have been added.
- The host (but not the router) side of IGMPv3 is implemented.
- When TM_USE_IGMPV3 is defined, the IGMPv3 protocol is supported along with IGMPv2 and IGMPv1 for backward compatibility.

System Wide Macros

Macro name	*Default value*	*Comment*
TM_USE_IGMPV3	Not defined	Add #define TM_USE_IGMPV3 to trsystem.h
TM_IP_MAX_SOURCE_FILTER	128	Maximum number of source addresses in the filter per multicast group. The user can redefine this macro with another value in trsystem.h.
TM_SO_MAX_SOURCE_FILTER	64	Maximum number of source addresses in the filter per socket for a given multicast group. The user can redefine this macro with another value in trsystem.h.

tfSetTreckOptions/tfInitTreckOptions

The user can change the per context default values for the maximum source filter per group and/or per socket at run time using either the **tfSetTreckOptions**, or **tfInitTreckOptions** APIs.

Option name	*Comment*
TM_OPTION_IGMP_SO_MAX_SRC_FILTER	Maximum Source Filter per socket.
TM_OPTION_IGMP_IP_MAX_SRC_FILTER	Maximum Source Filter per multicast group.

tfInterfaceSetOptions

To prevent DOS attacks, RFC 3376 states that "Hosts SHOULD ignore v2 or v3 Queries without the Router-Alert option." Unfortunately some routers do send IGMPv2 Queries without the router alert option. The stack follows the RFC, but allows the user to set an interface option to alter the behavior so that the stack can process IGMPv2 queries even when the local routers do not set the router alert option in the IGMPv2 queries.

A new option TM_DEV_OPTIONS_NO_IGMPV2_RA has been added to **tfInterfaceSetOptions**. Please refer to the Treck User Manual for the **tfInterfaceSetOptions** description.

Option name	*Data type*	*Comment*	
TM_DEV_OPTIONS_NO_IGMPV2_RA	Unsigned char	0	Drop incoming IGMPv2 queries that do not carry the router alert option. (Default per RFC 3376.)
		1	Accept IGMPv2 queries that do not carry the router alert option.

Example:
To prevent the stack from dropping IGMPv2 queries that do not carry the router alert option, call **tfInterfaceSetOptions** as follows:

```
Unsigned char optionValue;
optionValue = 1;
errorCode = tfInterfaceSetOptions(interfaceHandle,
TM_DEV_OPTIONS_NO_IGMPV2_RA, &optionValue, sizeof(unsigned char));
```

tfInterfaceGetOptions

The user can check whether the TM_DEV_OPTIONS_NO_IGMPV2_RA is on or off, by calling **tfInterfaceGetOptions**. For example:

Unsigned char optionValue;
errorCode = tfInterfaceGetOptions(interfaceHandle,
TM_DEV_OPTIONS_NO_IGMPV2_RA, &optionValue, sizeof(unsigned char));

setsockopt

New options have been added to **setsockopt** at the **SOL_SOCKET** level to allow the user to specify source filters. See **setsockopt** section of the Treck User's manual for **setsockopt** parameters, and usage.

Data structures

Two data structures are used by **setsockopt** with the IGMP multicast options; struct ip_mreq is an existing structure; struct ip_mreq_source has been added to support source filtering.

Data structure	*Usage*
struct ip_mreq	Currently used by the user to specify the multicast group address (destination) and optionally the local interface. It allows any source address
struct ip_mreq_source	This new data structure allows the user to specify a source address in addition to the multicast group address (destination) and local interface.

struct ip_mreq	{	
struct in_addr	imr_multiaddr	Ipv4 multicast address of group
struct in_addr	imr_interface	Ipv4 address of interface. Note that a zero ip address is valid if the user has either designated a default multicast interface for the socket (**IP_MULTICAST_IF**), or designated a default multicast interface for the system (**tfSetMcastInterface**).
};		

struct ip_mreq_source	{	
struct in_addr	imr_multiaddr	Ipv4 multicast address of group
struct in_addr	imr_sourceaddr	IPv4 address of source
struct in_addr	imr_interface	Ipv4 address of interface.

		Note that a zero ip address is valid if the user has either designated a default multicast interface for the socket (**IP_MULTICAST_IF**), or designated a default multicast interface for the system (**tfSetMcastInterface**).
};		

Options

IP_ADD_MEMBERSHIP, and IP_DROP_MEMBERSHIP are existing options. All other options have been added to support source filtering.

Note: ***For a given socket, interface, multicast group address, the user has to either use the exclude mode, or the include mode, i.e. the user has to pick options from one set and one set only, i.e. the user has to pick options in the IPv4 Any Source Multicast APIs set, or in the IPv4 Source Specific Multicast APIs set. To switch between include mode and exclude mode the user has to leave the group (IP_DROP_MEMBERSHIP).***

1. IPv4 Any Source Multicast API (exclude mode):

By default, all sources are accepted. Individual sources may be turned off and back on as needed over time. This is also known as "exclude" mode, since the source filter contains a list of excluded sources.

Option	*Associated structure*	*Comment*
IP_ADD_MEMBERSHIP	struct ip_mreq	Join the specified multicast group, regardless of the source address
IP_DROP_MEMBERSHIP	struct ip_mreq	Leave the specified multicast group, regardless of the source address.
IP_BLOCK_SOURCE	struct ip_mreq_source	Block data from a given source to a given multicast group (mute)
IP_UNBLOCK_SOURCE	struct ip_mreq_source	Unblock data from a given source to a given multicast group (un-mute)

2. IPv4 Source Specific Multicast API (include mode):

Only sources in a given list are allowed. The list may change over time. This is also known as "include" mode, since the source filter contains a list of included sources. IPv4 Source Specific Multicast API would be used, for example, by "single-source" applications such as audio/video broadcasting. It would also be used for logical multi-source sessions where each source independently allocates its own Source-Specific Multicast group address.

Option	*Associated structure*	*Comment*
IP_ADD_SOURCE_MEMBERSHIP	struct ip_mreq_source	Join a source-specific group
IP_DROP_SOURCE_MEMBERSHIP	struct ip_mreq_source	Leave a source-specific group.
IP_DROP_MEMBERSHIP	struct ip_mreq	Provided as a convenience. Drop all sources which have been joined for a particular group and interface.

Return

Upon success the **setsockopt** API returns 0. Upon failure the **setsockopt** API returns -1. In case of failure the user can get the error code by calling **tfGetSocketError**(sd);

errorCode	*Meaning*
TM_EBADF	Invalid socket descriptor.
TM_EPROTOTYPE	The socket is not of type SOCK_DGRAM or SOCK_RAW
TM_EINVAL	• Operation is not legal on the multicast group. (i.e., when trying a Source-Specific option on a group after doing IP_ADD_MEMBERSHIP, or when trying an Any-Source option without doing IP_ADD_MEMBERSHIP). Please see table below. • The option length is not valid. For example when trying to use struct ip_mreq_source instead of struct ip_mreq with IP_DROP_MEMBERSHIP or IP_ADD_MEMBERSHIP.
TM_EADDRNOTAVAIL	Address is invalid. For example when the interface address is not a valid configured

	address, or when the group address is not multicast, or when the source address is invalid, or when trying to add a source on a multicast local group, or when trying to block a source that is already blocked on the socket, or when trying to drop an un-joined multicast group. Please see table below.
TM_EADDRINUSE	Address already in use. Please see table below.
TM_ENOBUFS	No memory for operation, most probably because the maximum number of filters has been reached.

The following table shows what sequence of requests are allowed or disallowed, and if disallowed what the error code is.

Previous Request	*Next Request*	*errorCode*
IP_ADD_MEMBERSHIP	IP_ADD_MEMBERSHIP	TM_EADDRINUSE
IP_ADD_MEMBERSHIP	IP_DROP_MEMBERSHIP	0
IP_ADD_MEMBERSHIP	IP_ADD_SOURCE_MEMBERSHIP	TM_EINVAL
IP_ADD_MEMBERSHIP	IP_DROP_SOURCE_MEMBERSHIP	TM_EINVAL
IP_ADD_MEMBERSHIP	IP_BLOCK_SOURCE	0
IP_ADD_SOURCE_MEMBERSHIP	IP_ADD_MEMBERSHIP	TM_EADDRINUSE
IP_ADD_SOURCE_MEMBERSHIP	IP_DROP_MEMBERSHIP	0
IP_ADD_SOURCE_MEMBERSHIP	IP_ADD_SOURCE_MEMBERSHIP	TM_EADDRNOTAVAIL if source address is same
		0 otherwise
IP_ADD_SOURCE_MEMBERSHIP	IP_DROP_SOURCE_MEMBERSHIP	0 if source address is same
		TM_EADDRNOTAVAIL otherwise
IP_ADD_SOURCE_MEMBERSHIP	IP_BLOCK_SOURCE	TM_EINVAL
IP_ADD_SOURCE_MEMBERSHIP	IP_UNBLOCK_SOURCE	TM_EINVAL
IP_BLOCK_SOURCE	IP_ADD_MEMBERSHIP	TM_EADDRINUSE
IP_BLOCK_SOURCE	IP_DROP_MEMBERSHIP	0
IP_BLOCK_SOURCE	IP_ADD_SOURCE_MEMBERSHIP	TM_EINVAL
IP_BLOCK_SOURCE	IP_DROP_SOURCE_MEMBERSHIP	TM_EINVAL
IP_BLOCK_SOURCE	IP_BLOCK_SOURCE	TM_EADDRNOTAVAIL

		if source address is same
		0 otherwise
IP_BLOCK_SOURCE	IP_UNBLOCK_SOURCE	0 if source address is same
		TM_EADDRNOTAVAIL otherwise

getsockopt

None of those options are supported in the **getsockopt** socket API

Return

The **getsockopt** API returns -1 if called with any of these options. In that case **tfGetSocketError**(sd) will return TM_EOPNOTSUPP

errorCode	*Meaning*
TM_EOPNOTSUPP	Operation is not supported.

setipv4sourcefilter

#include <trsocket.h>

int **setipv4sourcefilter**	(
int	sd,
struct in_addr	interfaceAddr,
struct in_addr	group,
ttUser32Bit	fmode,
ttUser32Bit	numsrc,
struct in_addr *	slist
);	

The **setipv4sourcefilter** API allows the user to set a source address filter list in either exclude mode or include mode for a given socket, interface, and multicast source address.

Calling **setipv4sourcefilter** in include mode with numsrc set to zero is the same as dropping the multicast membership.

Note: ***For a given socket, interface, multicast group address, the user has to either use the exclude mode, or the include mode. The user can switch between the include mode and the exclude mode. The user can only specify one source list. A new call replaces the source list.***

Parameters

Parameter	*Comment*
sd	Socket Descriptor
interfaceAddr	Local IPv4 address of the corresponding configured interface. Note that a zero ip address is valid if the user has either designated a default multicast interface for the socket (**IP_MULTICAST_IF**), or designated a default multicast interface for the system (**tfSetMcastInterface**).
group	Destination multicast group address
fmode	Filter mode. Either MCAST_INCLUDE, or MCAST_EXCLUDE
numsrc	Number of source addresses in the slist array.
slist	Points to an array of IPv4 source addresses to include or exclude based on the value of the fmode argument.

Return

Upon success the **setipv4sourcefilter** API returns 0. Upon failure the **setipv4sourcefilter** API returns -1. In case of failure the user can get the error code by calling **tfGetSocketError**(sd);

ErrorCode	*Meaning*
TM_EBADF	Socket Descriptor is not valid.
TM_EPROTOTYPE	The socket is not of type SOCK_DGRAM or SOCK_RAW
TM_EINVAL	Operation is not legal on the group. For example fmode contains a value other than MCAST_INCLUDE, or MCAST_EXCLUDE.
TM_EADDRNOTAVAIL	Address is invalid. For example when the interface address is not a valid configured address, or when a source address is invalid, or when trying to add a source on a local multicast group, or when the group address is not multicast, or when trying to drop an un-joined multicast group (by setting numsrc to 0 in include mode for an unjoined multicast group).
TM_ENOBUFS	No memory for operation, most probably because the maximum number of filters has been reached.

getipv4sourcefilter

#include <trsocket.h>

int **getipv4sourcefilter**	(
int	sd,
struct in_addr	interfaceAddr,
struct in_addr	group,
ttUser32BitPtr	fmodePtr,
ttUser32BitPtr	numsrcPtr,
struct in_addr *	slist
);	

The **getipv4sourcefilter** API allows the user to retrieve the source address filter list for a given socket, interface, and multicast source address.

Parameters

Parameter	*Comment*
sd	Socket Descriptor
interfaceAddr	Local IPv4 address of the corresponding configured interface. Note that a zero ip address is valid if the user has either designated a default multicast interface for the socket (**IP_MULTICAST_IF**), or designated a default multicast interface for the system (**tfSetMcastInterface**).
group	Destination multicast group address
fmodePtr	points to a 32-bit integer that will contain the filter mode on a successful return. The value of this field will be either MCAST_INCLUDE or MCAST_EXCLUDE
numsrcPtr	On input, the numsrcPtr argument holds the number of source addresses that will fit in the slist array. On output, the numsrcPtr argument will hold the total number of sources in the filter. If the application does not know the size of the source list beforehand, it can make a reasonable guess (e.g., 0), and if upon completion, numsrcPtr points to a larger value, the operation can be repeated with a large enough buffer.That is, on return,

	numsrcPtr is always updated to point to the total number of sources in the filter, while slist will hold as many source addresses as fit, up to the minimum of the array size passed in as the original numsrc value and the total number of sources in the filter.
slist	The slist argument points to a buffer into which an array of IPv4 addresses of included or excluded (depending on the filter mode) sources will be written. If numsrcPtr pointed to 0 on input, a NULL pointer may be supplied.

Return

Upon success the **getipv4sourcefilter** API returns 0. Upon failure the **getipv4sourcefilter** API returns -1. In case of failure the user can get the error code by calling **tfGetSocketError**(sd);

ErrorCode	*Meaning*
TM_EBADF	Socket Descriptor is not valid.
TM_EPROTOTYPE	The socket is not of type SOCK_DGRAM or SOCK_RAW
TM_EINVAL	Operation is not legal on the group. For example fmodePtr points to a value other than MCAST_INCLUDE, or MCAST_EXCLUDE.
TM_EADDRNOTAVAIL	Address is invalid. For example when the interface address is not a valid configured address, or when one of the source addresses is invalid, or when the group address is not multicast, or when the group address is not joined.
TM_ENOBUFS	No memory for operation, most probably because the maximum number of filters has been reached.

tfIoctl

New **tfIoctl** options SIOCSIPMSFILTER, and SIOCGIPMSFILTER have been added to allow the user to specify source filters. See **tfIoctl** section of the Treck User's manual for **tfIoctl** parameters, and usage.

Note: setipv4sourcefilter, and getipv4sourcefilter obsolete the tfIoctl options SIOCSIPMSFILTER, and SIOCGIPMSFILTER. Those tfIoctl options are provided

for backward compatibility.

Data structure

One data structure is used by **tfIoctl** to support adding/getting source filter lists.

Data structure	*Usage*
struct ip_msfilter	This new data structure allows the user to specify a source address list, and the mode (exclude or include) for a given the multicast group address (destination) and local interface. The **tfIoctl** third parmeter points to such a structure, when the second parameter is set to either SIOCSIPMSFILTER, or SIOCGIPMSFILTER.

struct ip_msfilter	{	
struct in_addr	imsf_multiaddr;	Ipv4 multicast address of group
struct in_addr	imsf_interface;	Ipv4 address of interface. Note that a zero ip address is valid if the user has either designated a default multicast interface for the socket (**IP_MULTICAST_IF**), or designated a default multicast interface for the system (**tfSetMcastInterface**).
ttUser32Bit	imsf_fmode;	Filter Mode (include, or exclude)
ttUser32Bit	imsf_numsrc;	Number of sources in imsf_slist[]
struct in_addr	imsf_slist[1];	Start of IPv4 source list
};		

Macros

The following macros are defined in trsocket.h:

Macro name	*Usage*	*Comment*
MCAST_INCLUDE	imsf_fmode	Initialize imsf_fmode with MCAST_INCLUDE when the mode is inclusive

MCAST_EXCLUDE	imsf_fmode	Initialize imsf_fmode with MCAST_EXCLUDE when the mode is exclusive
IP_MSFILTER_SIZE(x)	Allocation size of the ip_msfilter structure.	When x is the number of sources as set in imsf_numsrc, this macro gives the allocation size of the ip_msfilter structure.

Options

***tfIoctl** option*	*Argument type*	*Comment*
SIOCSIPMSFILTER	Struct ip_msfilter	set or modify the source filter content (e.g., unicast source address list) or mode (exclude or include).
SIOCGIPMSFILTER	Struct ip_msfilter	retrieve the list of source addresses that comprise the source filter along with the current filter mode.

*Option (second parameter of **tfIoctl**)*	*Detailed Usage*
SIOCSIPMSFILTER	The third parameter of the **tfIoctl** API points to an ip_msfilter structure. This allocated structure length must be at least IP_MSFILTER_SIZE(0) bytes long, and the imsf_numsrc parameter should be set so that IP_MSFILTER_SIZE(imsf_numsrc) indicates the buffer length. The imsf_fmode should be set to MCAST_INCLUDE or MCAST_EXCLUDE. Calling **i**n include mode with imsf_numsrc set to zero is the same as dropping the multicast membership.
SIOCGIPMSFILTER	If the application does not know the size of the source list beforehand, it can make a reasonable guess (e.g., 0), and if upon completion, the imsf_numsrc field holds a larger value, the operation can be repeated

	with a large enough buffer. That is, on return from SIOCGIPMSFILTER, imsf_numsrc is always updated to be the total number of sources in the filter, while imsf_slist will hold as many source addresses as fit, up to the minimum of the array size passed in as the original imsf_numsrc value and the total number of sources in the filter.

Note: ***For a given socket, interface, multicast group address, the user has to either use the exclude mode, or the include mode. The user can switch between the include mode and the exclude mode. The user can only specify one source list. A new call replaces the source list.***

Return

Upon success the **tfIoctl** API returns 0. Upon failure the **tfIoctl** API returns -1. In case of failure the user can get the error code by calling **tfGetSocketError**(sd);

errorCode	*Meaning*
TM_EBADF	Invalid socket descriptor.
TM_EPROTOTYPE	The socket is not of type SOCK_DGRAM or SOCK_RAW
TM_EINVAL	Operation is not legal on the group. For example imsf_fmode contains a value other than MCAST_INCLUDE, or MCAST_EXCLUDE.
TM_EADDRNOTAVAIL	Address is invalid. For example when the interface address is not a valid configured address, or when a source address is invalid, or when trying to add a source on a local multicast group, or when the group address is not multicast, or when trying to get the source list from an unjoined multicast group, or when trying to drop an un-joined multicast group (by setting imsf_numsrc to 0 in include mode for an unjoined multicast group).
TM_ENOBUFS	No memory for operation, most probably because the maximum number of filters has been reached.