NAME

ng_macfilter - packet filtering netgraph node using ethernet MAC addresses

SYNOPSIS

#include <sys/types.h>
#include <netgraph/ng_macfilter.h>

DESCRIPTION

The **macfilter** allows routing ethernet packets over different hooks based on the sender MAC address.

This processing is done when traffic flows from the "ether" hook through **macfilter** to one of the outgoing hooks. Outbound hooks can be added to and remove from **macfilter** and arbitrarily named. By default one hook called "default" is present and used for all packets which have no MAC address in the MAC table. By adding MAC addresses to the MAC table traffic coming from this host can be directed out other hooks. **macfilter** keeps track of packets and bytes from and to this MAC address in the MAC table.

Packets are not altered in any way. If hooks are not connected, packets are dropped.

HOOKS

This node type by default has an ether hook, to be connected to the lower hook of the NIC, and a default hook where packets are sent if the MAC adddress is not found in the table. **macfilter** supports up to NG_MACFILTER_UPPER_NUM hooks to be connected to the NIC’s upper hook. Other nodes can be inserted to provide additional processing. All outbound can be combined back into one by using ng_one2many.

CONTROL MESSAGES

This node type supports the generic control messages, plus the following:

NGM_MACFILTER_RESET (**reset**)
Resets the MAC table in the node.

NGM_MACFILTER_DIRECT (**direct**)
Takes the following argument struct:

```
struct ngm_macfilter_direct {
  u_char ether[ETHER_ADDR_LEN];        /* MAC address */
  u_char hookname[NG_HOOKSIZ];         /* Upper hook name*/
};
```

The given ethernet MAC address will be forwarded out the named hook.

NGM_MACFILTER_DIRECT_HOOKID (**directi**)

Takes the following argument struct:

```
struct ngm_macfilter_direct_hookid {
  u_char ether[ETHER_ADDR_LEN];     /* MAC address */
  u_int16_t     hookid;             /* Upper hook hookid */
};
```

The given ethernet MAC address will be forwarded out the hook at id hookid.

NGM_MACFILTER_GET_MACS (**getmacs**)

Returns the list of MAC addresses in the node in the following structure:

```
struct ngm_macfilter_mac {
  u_char ether[ETHER_ADDR_LEN];     /* MAC address */
  u_int16_t     hookid;             /* Upper hook hookid */
  u_int64_t     packets_in;                 /* packets in from downstream */
  u_int64_t     bytes_in;           /* bytes in from upstream */
  u_int64_t     packets_out;                /* packets out towards downstream */
  u_int64_t     bytes_out;                  /* bytes out towards downstream */
};
struct ngm_macfilter_macs {
  u_int32_t   n;                /* Number of entries in macs */
  struct ngm_macfilter_mac macs[];      /* Macs table */
};
```

NGM_MACFILTER_GETCLR_MACS (**getclrmacs**)

Same as above, but will also atomically clear the packets_in, bytes_in, packets_out, and bytes_out fields in the table.

NGM_MACFILTER_CLR_STATS (**clrmacs**)

Will clear the per MAC address packet and byte counters.

NGM_MACFILTER_GET_HOOKS (**gethooks**)

Will return a list of hooks and their hookids in an array of the following struct’s:

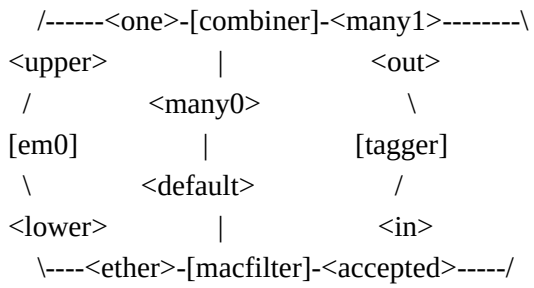
```
struct ngm_macfilter_hook {
  u_char hookname[NG_HOOKSIZ];      /* Upper hook name*/
  u_int16_t     hookid;             /* Upper hook hookid */
  u_int32_t   maccnt;           /* Number of mac addresses associated with hook */
};
```

SHUTDOWN

This node shuts down upon receipt of a NGM_SHUTDOWN control message or when all have been disconnected.

EXAMPLES

The following netgraph configuration will apply ipfw(8) tag 42 to each packet that is routed over the "accepted" hook. The graph looks like the following:



Commands:

```
ngctl mkpeer em0: macfilter lower ether
ngctl name em0:lower macfilter

# Funnel both streams back into ether:upper
ngctl mkpeer em0: one2many upper one
ngctl name em0:upper recombiner
# Connect macfilter:default to recombiner:many0
ngctl connect macfilter: recombiner: default many0
# Connect macfilter:accepted to tagger:in
ngctl mkpeer macfilter: tag accepted in
ngctl name macfilter:accepted tagger
# Connect tagger:out to recombiner:many1
ngctl connect tagger: recombiner: out many1

# Mark tag all traffic through tagger in -> out with an ipfw tag 42
ngctl msg tagger: sethookin '{ thisHook="in" ifNotMatch="out" }'
ngctl msg tagger: sethookout '{ thisHook="out" tag_cookie=1148380143 tag_id=42 }'

# Pass traffic from ether:upper / combiner:one via combiner:many0 on to
# macfilter:default and on to ether:lower.
ngctl msg recombiner: setconfig '{ xmitAlg=3 failAlg=1 enabledLinks=[ 1 1 ] }'
```

Note: The tag_cookie 1148380143 was retrieved from MTAG_IPFW in */usr/include/netinet/ip_var.h*.

The following command can be used to add a MAC address to be output via macfilter:accepted:

```
ngctl msg macfilter: direct '{ hookname="known" ether=08:00:27:92:eb:aa }'
```

The following command can be used to retrieve the packet and byte counters :

```
ngctl msg macfilter: getmacs
```

It will return the contents of the MAC table:

```
Rec'd response "getmacs" (4) from "[54]:":
Args:    { n=1 macs=[ { ether=08:00:27:92:eb:aa hookid=1 packets_in=3571 bytes_in=592631 packets_out=3437 b
```

SEE ALSO

divert(4), ipfw(4), netgraph(4), ng_ether(4), ng_one2many(4), ng_tag(4), ngctl(8)

AUTHORS

The original version of this code was written by Pekka Nikander, and subsequently modified heavily by Nick Hibma <*n_hibma@FreeBSD.org*>.

BUGS

None known.