```
**********************************************************
   79084 Mon May  5 11:11:28 2014
new/usr/src/uts/common/inet/ipf/ip_fil_solaris.c
4787 ipf: remove rate_limit_message
**********************************************************
______unchanged_portion_omitted_

#include <sys/time.h>
#include <sys/varargs.h>

#ifndef _KERNEL
#include <stdio.h>
#endif

#define NULLADDR_RATE_LIMIT 10  /* 10 seconds */


/*
 * Print out warning message at rate-limited speed.
 */
static void rate_limit_message(ipf_stack_t *ifs,
                               int rate, const char *message, ...)
{
        static time_t last_time = 0;
        time_t now;
        va_list args;
        char msg_buf[256];
        int  need_printed = 0;

        now = ddi_get_time();

        /* make sure, no multiple entries */
        ASSERT(MUTEX_NOT_HELD(&(ifs->ifs_ipf_rw.ipf_lk)));
        MUTEX_ENTER(&ifs->ifs_ipf_rw);
        if (now - last_time >= rate) {
                need_printed = 1;
                last_time = now;
        }
        MUTEX_EXIT(&ifs->ifs_ipf_rw);

        if (need_printed) {
                va_start(args, message);
                (void)vsnprintf(msg_buf, 255, message, args);
                va_end(args);
#ifdef _KERNEL
                cmn_err(CE_WARN, msg_buf);
#else
                fprintf(std_err, msg_buf);
#endif
        }
}

/*
 * Return the first IP Address associated with an interface
 * For IPv6, we walk through the list of logical interfaces and return
 * the address of the first one that isn't a link-local interface.
 * We can't assume that it is :1 because another link-local address
 * may have been assigned there.
 */
/*ARGSUSED*/
int fr_ifpaddr(v, atype, ifptr, inp, inpmask, ifs)
int v, atype;
void *ifptr;
struct in_addr  *inp, *inpmask;
ipf_stack_t *ifs;
{
```



```
        struct sockaddr_in6 v6addr[2];
        struct sockaddr_in v4addr[2];
        net_ifaddr_t type[2];
        net_handle_t net_data;
        phy_if_t phyif;
        void *array;

        switch (v)
        {
        case 4:
                net_data = ifs->ifs_ipf_ipv4;
                array = v4addr;
                break;
        case 6:
                net_data = ifs->ifs_ipf_ipv6;
                array = v6addr;
                break;
        default:
                net_data = NULL;
                break;
        }

        if (net_data == NULL)
                return -1;

        phyif = (phy_if_t)ifptr;

        switch (atype)
        {
        case FRI_PEERADDR :
                type[0] = NA_PEER;
                break;

        case FRI_BROADCAST :
                type[0] = NA_BROADCAST;
                break;

        default :
                type[0] = NA_ADDRESS;
                break;
        }

        type[1] = NA_NETMASK;

        if (v == 6) {
                lif_if_t idx = 0;

                do {
                        idx = net_lifgetnext(net_data, phyif, idx);
                        if (net_getlifaddr(net_data, phyif, idx, 2, type,
                                           array) < 0)
                                return -1;
                        if (!IN6_IS_ADDR_LINKLOCAL(&v6addr[0].sin6_addr) &&
                            !IN6_IS_ADDR_MULTICAST(&v6addr[0].sin6_addr))
                                break;
                } while (idx != 0);

                if (idx == 0)
                        return -1;

                return fr_ifpfillv6addr(atype, &v6addr[0], &v6addr[1],
                                        inp, inpmask);
        }

        if (net_getlifaddr(net_data, phyif, 0, 2, type, array) < 0)
                return -1;
```

```
1376         return fr_ifpfillv4addr(atype, &v4addr[0], &v4addr[1], inp, inpmask);
1377 }
```

_____***unchanged_portion_omitted_***