```
**********************************************************
   94925 Thu Apr  4 14:09:51 2019
new/usr/src/uts/common/inet/ip/spdsock.c
10687 Service routine cast changes need smatch fixes
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 2001, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright (c) 2012 Nexenta Systems, Inc. All rights reserved.
 * Copyright 2019, Joyent, Inc.
 * Copyright 2017 Joyent, Inc.
 */

#include <sys/param.h>
#include <sys/types.h>
#include <sys/stream.h>
#include <sys/strsubr.h>
#include <sys/strsun.h>
#include <sys/stropts.h>
#include <sys/zone.h>
#include <sys/vnode.h>
#include <sys/sysmacros.h>
#define _SUN_TPI_VERSION 2
#include <sys/tihdr.h>
#include <sys/timod.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/mkdev.h>
#include <sys/debug.h>
#include <sys/kmem.h>
#include <sys/cmn_err.h>
#include <sys/suntpi.h>
#include <sys/policy.h>
#include <sys/dls.h>

#include <sys/socket.h>
#include <netinet/in.h>
#include <net/pfkeyv2.h>
#include <net/pfpolicy.h>

#include <inet/common.h>
#include <netinet/ip6.h>
#include <inet/ip.h>
#include <inet/ip6.h>
#include <inet/mi.h>
#include <inet/proto_set.h>
#include <inet/nd.h>
```

```
#include <inet/ip_if.h>
#include <inet/optcom.h>
#include <inet/ipsec_impl.h>
#include <inet/spdsock.h>
#include <inet/sadb.h>
#include <inet/iptun.h>
#include <inet/iptun/iptun_impl.h>

#include <sys/isa_defs.h>

#include <c2/audit.h>

/*
 * This is a transport provider for the PF_POLICY IPsec policy
 * management socket, which provides a management interface into the
 * SPD, allowing policy rules to be added, deleted, and queried.
 *
 * This effectively replaces the old private SIOC*IPSECONFIG ioctls
 * with an extensible interface which will hopefully be public some
 * day.
 *
 * See <net/pfpolicy.h> for more details on the protocol.
 *
 * We link against drv/ip and call directly into it to manipulate the
 * SPD; see ipsec_impl.h for the policy data structures and spd.c for
 * the code which maintains them.
 *
 * The MT model of this is QPAIR with the addition of some explicit
 * locking to protect system-wide policy data structures.
 */

static vmem_t *spdsock_vmem;            /* for minor numbers. */

#define ALIGNED64(x) IS_P2ALIGNED((x), sizeof (uint64_t))

/* Default structure copied into T_INFO_ACK messages (from rts.c...) */
static struct T_info_ack spdsock_g_t_info_ack = {
        T_INFO_ACK,
        T_INFINITE,     /* TSDU_size. Maximum size messages. */
        T_INVALID,      /* ETSDU_size. No expedited data. */
        T_INVALID,      /* CDATA_size. No connect data. */
        T_INVALID,      /* DDATA_size. No disconnect data. */
        0,              /* ADDR_size. */
        0,              /* OPT_size. No user-settable options */
        64 * 1024,      /* TIDU_size. spdsock allows maximum size messages. */
        T_COTS,         /* SERV_type. spdsock supports connection oriented. */
        TS_UNBND,       /* CURRENT_state. This is set from spdsock_state. */
        (XPG4_1)        /* Provider flags */
};
______unchanged_portion_omitted_

/*
 * Write-side service procedure, invoked when we defer processing
 * if another message is received while a dump is in progress.
 */
int
spdsock_wsrv(queue_t *q)
{
        spdsock_t *ss = q->q_ptr;
        mblk_t *mp;
        ipsec_stack_t *ipss = ss->spdsock_spds->spds_netstack->netstack_ipsec;

        if (ss->spdsock_dump_req != NULL) {
                qenable(OTHERQ(q));
                return (0);
        }
	while ((mp = getq(q)) != NULL) {
		if (ipsec_loaded(ipss)) {
			(void) spdsock_wput(q, mp);
			spdsock_wput(q, mp);
			if (ss->spdsock_dump_req != NULL)
				return (0);
		} else if (!ipsec_failed(ipss)) {
			(void) putq(q, mp);
		} else {
			spdsock_error(q, mp, EPFNOSUPPORT, 0);
		}
	}
	return (0);
}
```

______unchanged_portion_omitted_

```
**********************************************************
   65228 Thu Apr  4 14:09:51 2019
new/usr/src/uts/common/io/ppp/sppp/sppp.c
10687 Service routine cast changes need smatch fixes
**********************************************************
/*
 * sppp.c - Solaris STREAMS PPP multiplexing pseudo-driver
 *
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 * Copyright (c) 2016 by Delphix. All rights reserved.
 * Copyright 2019, Joyent, Inc.
 *
 * Permission to use, copy, modify, and distribute this software and its
 * documentation is hereby granted, provided that the above copyright
 * notice appears in all copies.
 *
 * SUN MAKES NO REPRESENTATION OR WARRANTIES ABOUT THE SUITABILITY OF
 * THE SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED
 * TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE, OR NON-INFRINGEMENT.  SUN SHALL NOT BE LIABLE FOR
 * ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR
 * DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES
 *
 * Copyright (c) 1994 The Australian National University.
 * All rights reserved.
 *
 * Permission to use, copy, modify, and distribute this software and its
 * documentation is hereby granted, provided that the above copyright
 * notice appears in all copies.  This software is provided without any
 * warranty, express or implied. The Australian National University
 * makes no representations about the suitability of this software for
 * any purpose.
 *
 * IN NO EVENT SHALL THE AUSTRALIAN NATIONAL UNIVERSITY BE LIABLE TO ANY
 * PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES
 * ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF
 * THE AUSTRALIAN NATIONAL UNIVERSITY HAS BEEN ADVISED OF THE POSSIBILITY
 * OF SUCH DAMAGE.
 *
 * THE AUSTRALIAN NATIONAL UNIVERSITY SPECIFICALLY DISCLAIMS ANY WARRANTIES,
 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE.  THE SOFTWARE PROVIDED HEREUNDER IS
 * ON AN "AS IS" BASIS, AND THE AUSTRALIAN NATIONAL UNIVERSITY HAS NO
 * OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS,
 * OR MODIFICATIONS.
 *
 * This driver is derived from the original SVR4 STREAMS PPP driver
 * originally written by Paul Mackerras <paul.mackerras@cs.anu.edu.au>.
 *
 * Adi Masputra <adi.masputra@sun.com> rewrote and restructured the code
 * for improved performance and scalability.
 */

#define RCSID   "$Id: sppp.c,v 1.0 2000/05/08 01:10:12 masputra Exp $"

#include <sys/types.h>
#include <sys/debug.h>
#include <sys/param.h>
#include <sys/stat.h>
#include <sys/stream.h>
#include <sys/stropts.h>
#include <sys/sysmacros.h>
#include <sys/errno.h>
#include <sys/time.h>
#include <sys/cmn_err.h>
```

```
#include <sys/kmem.h>
#include <sys/conf.h>
#include <sys/dlpi.h>
#include <sys/ddi.h>
#include <sys/kstat.h>
#include <sys/strsun.h>
#include <sys/ethernet.h>
#include <sys/policy.h>
#include <sys/zone.h>
#include <net/ppp_defs.h>
#include <net/pppio.h>
#include "sppp.h"
#include "s_common.h"

/*
 * This is used to tag official Solaris sources.  Please do not define
 * "INTERNAL_BUILD" when building this software outside of Sun Microsystems.
 */
#ifdef INTERNAL_BUILD
/* MODINFO is limited to 32 characters. */
const char sppp_module_description[] = "PPP 4.0 mux";
#else /* INTERNAL_BUILD */
const char sppp_module_description[] = "ANU PPP mux";

/* LINTED */
static const char buildtime[] = "Built " __DATE__ " at " __TIME__
#ifdef DEBUG
" DEBUG"
#endif
"\n";
#endif /* INTERNAL_BUILD */

static void     sppp_inner_ioctl(queue_t *, mblk_t *);
static void     sppp_outer_ioctl(queue_t *, mblk_t *);
static queue_t  *sppp_send(queue_t *, mblk_t **, spppstr_t *);
static queue_t  *sppp_recv(queue_t *, mblk_t **, spppstr_t *);
static void     sppp_recv_nondata(queue_t *, mblk_t *, spppstr_t *);
static queue_t  *sppp_outpkt(queue_t *, mblk_t **, int, spppstr_t *);
static spppstr_t *sppp_inpkt(queue_t *, mblk_t *, spppstr_t *);
static int      sppp_kstat_update(kstat_t *, int);
static void     sppp_release_pkts(sppa_t *, uint16_t);

/*
 * sps_list contains the list of active per-stream instance state structures
 * ordered on the minor device number (see sppp.h for details). All streams
 * opened to this driver are threaded together in this list.
 */
static spppstr_t *sps_list = NULL;
/*
 * ppa_list contains the list of active per-attachment instance state
 * structures ordered on the ppa id number (see sppp.h for details). All of
 * the ppa structures created once per PPPIO_NEWPPA ioctl are threaded together
 * in this list. There is exactly one ppa structure for a given PPP interface,
 * and multiple sps streams (upper streams) may share a ppa by performing
 * an attachment explicitly (PPPIO_ATTACH) or implicitly (DL_ATTACH_REQ).
 */
static sppa_t *ppa_list = NULL;

static const char *kstats_names[] = { SPPP_KSTATS_NAMES };
static const char *kstats64_names[] = { SPPP_KSTATS64_NAMES };

/*
 * map proto (which is an IANA defined ppp network protocol) to
 * a bit position indicated by NP_* in ppa_npflag
 */
static uint32_t
```

```
sppp_ppp2np(uint16_t proto)
{
        switch (proto) {
        case PPP_IP:
                return (NP_IP);
        case PPP_IPV6:
                return (NP_IPV6);
        default:
                return (0);
        }
}
```

______unchanged_portion_omitted_

```
/*
 * sppp_lrsrv()
 *
 * MT-Perimeters:
 *    exclusive inner, shared outer.
 *
 * Description:
 *    Lower read-side service procedure.  This is run once after the I_LINK
 *    occurs in order to clean up any packets that came in while we were
 *    transferring in the lower stream.  Otherwise, it’s not used.
 */
int
sppp_lrsrv(queue_t *q)
{
        mblk_t *mp;

        while ((mp = getq(q)) != NULL)
                (void) sppp_lrput(q, mp);
                sppp_lrput(q, mp);
        return (0);
}
```

______unchanged_portion_omitted_

```
**********************************************************
   15750 Thu Apr  4 14:09:51 2019
new/usr/src/uts/common/io/softmac/softmac_dev.c
10687 Service routine cast changes need smatch fixes
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 * Copyright 2019, Joyent, Inc.
 */

#include <sys/types.h>
#include <inet/common.h>
#include <sys/stropts.h>
#include <sys/modctl.h>
#include <sys/dld.h>
#include <sys/softmac_impl.h>

dev_info_t              *softmac_dip = NULL;
static kmem_cache_t     *softmac_upper_cachep;

/*
 * This function is a generic open(9E) entry point into the softmac for
 * both the softmac module and the softmac driver.
 */
static int softmac_cmn_open(queue_t *, dev_t *, int, int, cred_t *);

/*
 * The following softmac_mod_xxx() functions are (9E) entry point functions for
 * the softmac module.
 */
static int softmac_mod_close(queue_t *, int, cred_t *);
static int softmac_mod_rput(queue_t *, mblk_t *);
static int softmac_mod_wput(queue_t *, mblk_t *);
static int softmac_mod_wsrv(queue_t *);

/*
 * The following softmac_drv_xxx() functions are (9E) entry point functions for
 * the softmac driver.
 */
static int softmac_drv_open(queue_t *, dev_t *, int, int, cred_t *);
static int softmac_drv_close(queue_t *, int, cred_t *);
static int softmac_drv_wput(queue_t *, mblk_t *);
```

```
static int softmac_drv_wsrv(queue_t *);

static int softmac_attach(dev_info_t *, ddi_attach_cmd_t);
static int softmac_detach(dev_info_t *, ddi_detach_cmd_t);
static int softmac_info(dev_info_t *, ddi_info_cmd_t, void *, void **);

static struct module_info softmac_modinfo = {
        0,
        SOFTMAC_DEV_NAME,
        0,
        INFPSZ,
        65536,
        1024
};
______unchanged_portion_omitted_

static int
softmac_drv_wsrv(queue_t *wq)
{
        softmac_upper_t *sup = dld_str_private(wq);

        ASSERT(wq->q_next == NULL);

        mutex_enter(&sup->su_mutex);
        if (sup->su_mode != SOFTMAC_FASTPATH) {
                /*
                 * Bump su_tx_inprocess so that su_mode won't change.
                 */
                sup->su_tx_inprocess++;
                mutex_exit(&sup->su_mutex);
                (void) dld_wsrv(wq);
                dld_wsrv(wq);
                mutex_enter(&sup->su_mutex);
                if (--sup->su_tx_inprocess == 0)
                        cv_signal(&sup->su_cv);
        } else if (sup->su_tx_busy && SOFTMAC_CANPUTNEXT(sup->su_slp->sl_wq)) {
                /*
                 * The flow-conctol of the dedicated-lower-stream is
                 * relieved. If DLD_CAPAB_DIRECT is enabled, call tx_notify
                 * callback to relieve the flow-control of the specific client,
                 * otherwise relieve the flow-control of all the upper-stream
                 * using the traditional STREAM mechanism.
                 */
                if (sup->su_tx_notify_func != NULL) {
                        sup->su_tx_inprocess++;
                        mutex_exit(&sup->su_mutex);
                        sup->su_tx_notify_func(sup->su_tx_notify_arg,
                            (mac_tx_cookie_t)sup);
                        mutex_enter(&sup->su_mutex);
                        if (--sup->su_tx_inprocess == 0)
                                cv_signal(&sup->su_cv);
                }
                ASSERT(sup->su_tx_flow_mp == NULL);
                VERIFY((sup->su_tx_flow_mp = getq(wq)) != NULL);
                sup->su_tx_busy = B_FALSE;
        }
        mutex_exit(&sup->su_mutex);
        return (0);
}
______unchanged_portion_omitted_
```

```
**********************************************************
   34200 Thu Apr  4 14:09:52 2019
new/usr/src/uts/common/io/softmac/softmac_fp.c
10687 Service routine cast changes need smatch fixes
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 * Copyright 2019, Joyent, Inc.
 */

/*
 * Softmac data-path switching:
 *
 * - Fast-path model
 *
 * When the softmac fast-path is used, a dedicated lower-stream
 * will be opened over the legacy device for each IP/ARP (upper-)stream
 * over the softMAC, and all DLPI messages (including control messages
 * and data messages) will be exchanged between the upper-stream and
 * the corresponding lower-stream directly. Therefore, the data
 * demultiplexing, filtering and classification processing will be done
 * by the lower-stream, and the GLDv3 DLS/MAC layer processing will be
 * no longer needed.
 *
 * - Slow-path model
 *
 * Some GLDv3 features requires the GLDv3 DLS/MAC layer processing to
 * not be bypassed to assure its function correctness. For example,
 * softmac fast-path must be disabled to support GLDv3 VNIC functionality.
 * In this case, a shared lower-stream will be opened over the legacy
 * device, which is responsible for implementing the GLDv3 callbacks
 * and passing RAW data messages between the legacy devices and the GLDv3
 * framework.
 *
 * By default, the softmac fast-path mode will be used to assure the
 * performance; MAC clients will be able to request to disable the softmac
 * fast-path mode to support certain features, and if that succeeds,
 * the system will fallback to the slow-path softmac data-path model.
 *
 *
 * The details of the softmac data fast-path model is stated as below
 *
 * 1. When a stream is opened on a softMAC, the softmac module will takes
 *    over the DLPI processing on this stream;
 *
 * 2. For IP/ARP streams over a softMAC, softmac data fast-path will be
 *    used by default, unless fast-path is disabled by any MAC client
 *    explicitly. The softmac module first identifies an IP/ARP stream
 *    by seeing whether there is a SIOCSLIFNAME ioctl sent from upstream,
 *    if there is one, this stream is either an IP or an ARP stream
 *    and will use fast-path potentially;
 *
 * 3. When the softmac fast-path is used, an dedicated lower-stream will
 *    be setup for each IP/ARP stream (1-1 mapping). From that point on,
 *    all control and data messages will be exchanged between the IP/ARP
 *    upper-stream and the legacy device through this dedicated
 *    lower-stream. As a result, the DLS/MAC layer processing in GLDv3
 *    will be skipped, and this greatly improves the performance;
 *
 * 4. When the softmac data fast-path is disabled by a MAC client (e.g.,
 *    by a VNIC), all the IP/ARP upper streams will try to switch from
 *    the fast-path to the slow-path. The dedicated lower-stream will be
 *    destroyed, and all the control and data-messages will go through the
 *    existing GLDv3 code path and (in the end) the shared lower-stream;
 *
 * 5. On the other hand, when the last MAC client cancels its fast-path
 *    disable request, all the IP/ARP streams will try to switch back to
 *    the fast-path mode;
 *
 * Step 5 and 6 both rely on the data-path mode switching process
 * described below:
 *
 * 1) To switch the softmac data-path mode (between fast-path and slow-path),
 *    softmac will first send a DL_NOTE_REPLUMB DL_NOTIFY_IND message
 *    upstream over each IP/ARP streams that needs data-path mode switching;
 *
 * 2) When IP receives this DL_NOTE_REPLUMB message, it will bring down
 *    all the IP interfaces on the corresponding ill (IP Lower level
 *    structure), and bring up those interfaces over again; this will in
 *    turn cause the ARP to "replumb" the interface.
 *
 *    During the replumb process, both IP and ARP will send downstream the
 *    necessary DL_DISABMULTI_REQ and DL_UNBIND_REQ messages and cleanup
 *    the old state of the underlying softMAC, following with the necessary
 *    DL_BIND_REQ and DL_ENABMULTI_REQ messages to setup the new state.
 *    Between the cleanup and re-setup process, IP/ARP will also send down
 *    a DL_NOTE_REPLUMB_DONE DL_NOTIFY_CONF messages to the softMAC to
 *    indicate the *switching point*;
 *
 * 3) When softmac receives the DL_NOTE_REPLUMB_DONE message, it either
 *    creates or destroys the dedicated lower-stream (depending on which
 *    data-path mode the softMAC switches to), and change the softmac
 *    data-path mode. From then on, softmac will process all the succeeding
 *    control messages (including the DL_BIND_REQ and DL_ENABMULTI_REQ
 *    messages) and data messages based on new data-path mode.
 */

#include <sys/types.h>
#include <sys/disp.h>
#include <sys/callb.h>
#include <sys/sysmacros.h>
#include <sys/file.h>
#include <sys/vlan.h>
#include <sys/dld.h>
#include <sys/sockio.h>
#include <sys/softmac_impl.h>
#include <net/if.h>
```

```
static kmutex_t         softmac_taskq_lock;
static kcondvar_t       softmac_taskq_cv;
static list_t           softmac_taskq_list;     /* List of softmac_upper_t */
boolean_t               softmac_taskq_quit;
boolean_t               softmac_taskq_done;

static void             softmac_taskq_dispatch();
static int              softmac_fastpath_setup(softmac_upper_t *);
static mac_tx_cookie_t  softmac_fastpath_wput_data(softmac_upper_t *, mblk_t *,
                            uintptr_t, uint16_t);
static void             softmac_datapath_switch_done(softmac_upper_t *);

void
softmac_fp_init()
{
        mutex_init(&softmac_taskq_lock, NULL, MUTEX_DRIVER, NULL);
        cv_init(&softmac_taskq_cv, NULL, CV_DRIVER, NULL);

        softmac_taskq_quit = B_FALSE;
        softmac_taskq_done = B_FALSE;
        list_create(&softmac_taskq_list, sizeof (softmac_upper_t),
            offsetof(softmac_upper_t, su_taskq_list_node));
        (void) thread_create(NULL, 0, softmac_taskq_dispatch, NULL, 0,
            &p0, TS_RUN, minclsyspri);
}
______unchanged_portion_omitted_

/*
 * Process the non-data mblk.
 */
static void
softmac_wput_single_nondata(softmac_upper_t *sup, mblk_t *mp)
{
        softmac_t *softmac = sup->su_softmac;
        softmac_lower_t *slp = sup->su_slp;
        unsigned char   dbtype;
        t_uscalar_t     prim;

        dbtype = DB_TYPE(mp);
        sup->su_is_arp = 0;
        switch (dbtype) {
        case M_CTL:
                sup->su_is_arp = 1;
                /* FALLTHROUGH */
        case M_IOCTL: {
                uint32_t        expected_mode;

                if (((struct iocblk *)(mp->b_rptr))->ioc_cmd != SIOCSLIFNAME)
                        break;

                /*
                 * Nak the M_IOCTL based on the STREAMS specification.
                 */
                if (dbtype == M_IOCTL)
                        miocnak(sup->su_wq, mp, 0, EINVAL);
                else
                        freemsg(mp);

                /*
                 * This stream is either IP or ARP. See whether
                 * we need to setup a dedicated-lower-stream for it.
                 */
                mutex_enter(&softmac->smac_fp_mutex);

                expected_mode = DATAPATH_MODE(softmac);
                if (expected_mode == SOFTMAC_SLOWPATH)
```

```
                        sup->su_mode = SOFTMAC_SLOWPATH;
                list_insert_head(&softmac->smac_sup_list, sup);
                mutex_exit(&softmac->smac_fp_mutex);

                /*
                 * Setup the fast-path dedicated lower stream if fast-path
                 * is expected. Note that no lock is held here, and if
                 * smac_expected_mode is changed from SOFTMAC_FASTPATH to
                 * SOFTMAC_SLOWPATH, the DL_NOTE_REPLUMB message used for
                 * data-path switching would already be queued and will
                 * be processed by softmac_wput_single_nondata() later.
                 */
                if (expected_mode == SOFTMAC_FASTPATH)
                        (void) softmac_fastpath_setup(sup);
                return;
        }
        case M_PROTO:
        case M_PCPROTO:
                if (MBLKL(mp) < sizeof (t_uscalar_t)) {
                        freemsg(mp);
                        return;
                }
                prim = ((union DL_primitives *)mp->b_rptr)->dl_primitive;
                switch (prim) {
                case DL_NOTIFY_IND:
                        if (MBLKL(mp) < sizeof (dl_notify_ind_t) ||
                            ((dl_notify_ind_t *)mp->b_rptr)->dl_notification !=
                            DL_NOTE_REPLUMB) {
                                freemsg(mp);
                                return;
                        }
                        /*
                         * This DL_NOTE_REPLUMB message is initiated
                         * and queued by the softmac itself, when the
                         * sup is trying to switching its datapath mode
                         * between SOFTMAC_SLOWPATH and SOFTMAC_FASTPATH.
                         * Send this message upstream.
                         */
                        qreply(sup->su_wq, mp);
                        return;
                case DL_NOTIFY_CONF:
                        if (MBLKL(mp) < sizeof (dl_notify_conf_t) ||
                            ((dl_notify_conf_t *)mp->b_rptr)->dl_notification !=
                            DL_NOTE_REPLUMB_DONE) {
                                freemsg(mp);
                                return;
                        }
                        /*
                         * This is an indication from IP/ARP that the
                         * fastpath->slowpath switch is done.
                         */
                        freemsg(mp);
                        softmac_datapath_switch_done(sup);
                        return;
                }
                break;
        }

        /*
         * No need to hold lock to check su_mode, since su_mode updating only
         * operation is is serialized by softmac_wput_nondata_task().
         */
        if (sup->su_mode != SOFTMAC_FASTPATH) {
                (void) dld_wput(sup->su_wq, mp);
                dld_wput(sup->su_wq, mp);
                return;
        }

        /*
         * Fastpath non-data message processing. Most of non-data messages
         * can be directly passed down to the dedicated-lower-stream, aside
         * from the following M_PROTO/M_PCPROTO messages.
         */
        switch (dbtype) {
        case M_PROTO:
        case M_PCPROTO:
                switch (prim) {
                case DL_BIND_REQ:
                        softmac_bind_req(sup, mp);
                        break;
                case DL_UNBIND_REQ:
                        softmac_unbind_req(sup, mp);
                        break;
                case DL_CAPABILITY_REQ:
                        softmac_capability_req(sup, mp);
                        break;
                default:
                        putnext(slp->sl_wq, mp);
                        break;
                }
                break;
        default:
                putnext(slp->sl_wq, mp);
                break;
        }
}
```

______***unchanged_portion_omitted_***

```
void
softmac_wput_data(softmac_upper_t *sup, mblk_t *mp)
{
        /*
         * No lock is required to access the su_mode field since the data
         * traffic is quiesce by IP when the data-path mode is in the
         * process of switching.
         */
        if (sup->su_mode != SOFTMAC_FASTPATH)
                (void) dld_wput(sup->su_wq, mp);
                dld_wput(sup->su_wq, mp);
        else
                (void) softmac_fastpath_wput_data(sup, mp, NULL, 0);
}
```

______***unchanged_portion_omitted_***