
Subject: [PATCH net-2.6.25 9/19] [NETNS] Add netns parameter to
inet_(dev_)add_type.

Posted by [den](#) on Wed, 19 Dec 2007 15:24:39 GMT

[View Forum Message](#) <> [Reply to Message](#)

Acked-by: Benjamin Thery <benjamin.thery@bull.net>

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

drivers/s390/net/qeth_main.c | 2 +-
include/net/route.h | 4 ++--
net/atm/clip.c | 2 +-
net/ipv4/af_inet.c | 2 +-
net/ipv4/arp.c | 12 ++++++-----
net/ipv4/fib_frontend.c | 18 ++++++++-----
net/ipv4/fib_semantics.c | 4 ++--
net/ipv4/icmp.c | 4 ++--
net/ipv4/ip_options.c | 4 ++--
net/ipv4/ipvs/ip_vs_core.c | 2 +-
net/ipv4/ipvs/ip_vs_ctl.c | 4 ++--
net/ipv4/netfilter.c | 2 +-
net/ipv4/netfilter/ipt_addrtype.c | 2 +-
net/ipv4/raw.c | 2 +-
net/ipv4/route.c | 2 +-
net/ipv6/af_inet6.c | 2 +-
net/sctp/protocol.c | 2 +-
17 files changed, 36 insertions(+), 34 deletions(-)

diff --git a/drivers/s390/net/qeth_main.c b/drivers/s390/net/qeth_main.c

index ff99ff..f1866a2 100644

--- a/drivers/s390/net/qeth_main.c

+++ b/drivers/s390/net/qeth_main.c

@@ -8340,7 +8340,7 @@ qeth_arp_constructor(struct neighbour *neigh)

neigh->parms = neigh_parms_clone(parms);

rcu_read_unlock();

- neigh->type = inet_addr_type((__be32 *) neigh->primary_key);

+ neigh->type = inet_addr_type(&init_net, (__be32 *) neigh->primary_key);

neigh->nud_state = NUD_NOARP;

neigh->ops = arp_direct_ops;

neigh->output = neigh->ops->queue_xmit;

diff --git a/include/net/route.h b/include/net/route.h

index 76b08c1..b777000 100644

--- a/include/net/route.h

+++ b/include/net/route.h

@@ -117,8 +117,8 @@ extern int ip_route_input(struct sk_buff*, __be32 dst, __be32 src, u8 tos,
stru

extern unsigned short ip_rt_frag_needed(struct iphdr *iph, unsigned short new_mtu);

extern void ip_rt_send_redirect(struct sk_buff *skb);

```

-extern unsigned inet_addr_type(__be32 addr);
-extern unsigned inet_dev_addr_type(const struct net_device *dev, __be32 addr);
+extern unsigned inet_addr_type(struct net *net, __be32 addr);
+extern unsigned inet_dev_addr_type(struct net *net, const struct net_device *dev, __be32 addr);
extern void ip_rt_multicast_event(struct in_device *);
extern int ip_rt_ioctl(unsigned int cmd, void __user *arg);
extern void ip_rt_get_source(u8 *src, struct rtable *rt);
diff --git a/net/atm/clip.c b/net/atm/clip.c
index 741742f..66ff14f 100644
--- a/net/atm/clip.c
+++ b/net/atm/clip.c
@@ -285,7 +285,7 @@ static int clip_constructor(struct neighbour *neigh)
 struct neigh_parms *parms;

 pr_debug("clip_constructor (neigh %p, entry %p)\n", neigh, entry);
- neigh->type = inet_addr_type(entry->ip);
+ neigh->type = inet_addr_type(&init_net, entry->ip);
 if (neigh->type != RTN_UNICAST)
 return -EINVAL;

diff --git a/net/ipv4/af_inet.c b/net/ipv4/af_inet.c
index 19b557a..6fac905 100644
--- a/net/ipv4/af_inet.c
+++ b/net/ipv4/af_inet.c
@@ -444,7 +444,7 @@ int inet_bind(struct socket *sock, struct sockaddr *uaddr, int addr_len)
 if (addr_len < sizeof(struct sockaddr_in))
 goto out;

- chk_addr_ret = inet_addr_type(addr->sin_addr.s_addr);
+ chk_addr_ret = inet_addr_type(&init_net, addr->sin_addr.s_addr);

/* Not specified by any standard per-se, however it breaks too
 * many applications when removed. It is unfortunate since
diff --git a/net/ipv4/arp.c b/net/ipv4/arp.c
index 14dec92..dff4736 100644
--- a/net/ipv4/arp.c
+++ b/net/ipv4/arp.c
@@ -235,7 +235,7 @@ static int arp_constructor(struct neighbour *neigh)
 struct in_device *in_dev;
 struct neigh_parms *parms;

- neigh->type = inet_addr_type(addr);
+ neigh->type = inet_addr_type(&init_net, addr);

rcu_read_lock();
in_dev = __in_dev_get_rcu(dev);
@@ -341,14 +341,14 @@ static void arp_solicit(struct neighbour *neigh, struct sk_buff *skb)

```

```

switch (IN_DEV_ARP_ANNOUNCE(in_dev)) {
default:
case 0: /* By default announce any local IP */
- if (skb && inet_addr_type(ip_hdr(skb)->saddr) == RTN_LOCAL)
+ if (skb && inet_addr_type(&init_net, ip_hdr(skb)->saddr) == RTN_LOCAL)
    saddr = ip_hdr(skb)->saddr;
    break;
case 1: /* Restrict announcements of saddr in same subnet */
    if (!skb)
        break;
    saddr = ip_hdr(skb)->saddr;
- if (inet_addr_type(saddr) == RTN_LOCAL) {
+ if (inet_addr_type(&init_net, saddr) == RTN_LOCAL) {
    /* saddr should be known to target */
    if (inet_addr_onlink(in_dev, target, saddr))
        break;
@@ -479,7 +479,7 @@ int arp_find(unsigned char *haddr, struct sk_buff *skb)

    paddr = ((struct rtable*)skb->dst)->rt_gateway;

- if (arp_set_predefined(inet_addr_type(paddr), haddr, paddr, dev))
+ if (arp_set_predefined(inet_addr_type(&init_net, paddr), haddr, paddr, dev))
    return 0;

    n = __neigh_lookup(&arp_tbl, &paddr, dev, 1);
@@ -807,7 +807,7 @@ static int arp_process(struct sk_buff *skb)
/* Special case: IPv4 duplicate address detection packet (RFC2131) */
if (sip == 0) {
    if (arp->ar_op == htons(ARPOP_REQUEST) &&
-    inet_addr_type(tip) == RTN_LOCAL &&
+    inet_addr_type(&init_net, tip) == RTN_LOCAL &&
        !arp_ignore(in_dev, dev, sip, tip))
        arp_send(ARPOP_REPLY, ETH_P_ARP, sip, dev, tip, sha,
            dev->dev_addr, sha);
@@ -868,7 +868,7 @@ static int arp_process(struct sk_buff *skb)
*/
if (n == NULL &&
    arp->ar_op == htons(ARPOP_REPLY) &&
-    inet_addr_type(sip) == RTN_UNICAST)
+    inet_addr_type(&init_net, sip) == RTN_UNICAST)
    n = __neigh_lookup(&arp_tbl, &sip, dev, 1);
}

diff --git a/net/ipv4/fib_frontend.c b/net/ipv4/fib_frontend.c
index 2c20908..312a728 100644
--- a/net/ipv4/fib_frontend.c
+++ b/net/ipv4/fib_frontend.c
@@ -166,7 +166,8 @@ out:

```

```

* Find address type as if only "dev" was present in the system. If
* on_dev is NULL then all interfaces are taken into consideration.
*/
-static inline unsigned __inet_dev_addr_type(const struct net_device *dev,
+static inline unsigned __inet_dev_addr_type(struct net *net,
+      const struct net_device *dev,
+      __be32 addr)
{
    struct flowi fl = { .nl_u = { .ip4_u = { .daddr = addr } } };
@@ -183,7 +184,7 @@ static inline unsigned __inet_dev_addr_type(const struct net_device
*dev,
    res.r = NULL;
#endif

- local_table = fib_get_table(&init_net, RT_TABLE_LOCAL);
+ local_table = fib_get_table(net, RT_TABLE_LOCAL);
    if (local_table) {
        ret = RTN_UNICAST;
        if (!local_table->tb_lookup(local_table, &fl, &res)) {
@@ -195,14 +196,15 @@ static inline unsigned __inet_dev_addr_type(const struct net_device
*dev,
        return ret;
    }

-unsigned int inet_addr_type(__be32 addr)
+unsigned int inet_addr_type(struct net *net, __be32 addr)
{
- return __inet_dev_addr_type(NULL, addr);
+ return __inet_dev_addr_type(net, NULL, addr);
}

-unsigned int inet_dev_addr_type(const struct net_device *dev, __be32 addr)
+unsigned int inet_dev_addr_type(struct net *net, const struct net_device *dev,
+      __be32 addr)
{
-    return __inet_dev_addr_type(dev, addr);
+    return __inet_dev_addr_type(net, dev, addr);
}

/* Given (packet source, input interface) and optional (dst, oif, tos):
@@ -391,7 +393,7 @@ static int rentry_to_fib_config(int cmd, struct rentry *rt,
    if (rt->rt_gateway.sa_family == AF_INET && addr) {
        cfg->fc_gw = addr;
        if (rt->rt_flags & RTF_GATEWAY &&
-        inet_addr_type(addr) == RTN_UNICAST)
+        inet_addr_type(&init_net, addr) == RTN_UNICAST)
            cfg->fc_scope = RT_SCOPE_UNIVERSE;
    }

```

```
@@ -782,7 +784,7 @@ static void fib_del_ifaddr(struct in_ifaddr *ifa)
    fib_magic(RTM_DELROUTE, RTN_LOCAL, ifa->ifa_local, 32, prim);
```

```
/* Check, that this local address finally disappeared. */
- if (inet_addr_type(ifa->ifa_local) != RTN_LOCAL) {
+ if (inet_addr_type(&init_net, ifa->ifa_local) != RTN_LOCAL) {
    /* And the last, but not the least thing.
       We must flush stray FIB entries.
```

```
diff --git a/net/ipv4/fib_semantics.c b/net/ipv4/fib_semantics.c
index bbd4a24..c1263e2 100644
```

```
--- a/net/ipv4/fib_semantics.c
+++ b/net/ipv4/fib_semantics.c
@@ -531,7 +531,7 @@ static int fib_check_nh(struct fib_config *cfg, struct fib_info *fi,
```

```
    if (cfg->fc_scope >= RT_SCOPE_LINK)
        return -EINVAL;
- if (inet_addr_type(nh->nh_gw) != RTN_UNICAST)
+ if (inet_addr_type(&init_net, nh->nh_gw) != RTN_UNICAST)
    return -EINVAL;
    if ((dev = __dev_get_by_index(&init_net, nh->nh_oif)) == NULL)
        return -ENODEV;
@@ -809,7 +809,7 @@ struct fib_info *fib_create_info(struct fib_config *cfg)
    if (fi->fib_prefsrsrc) {
        if (cfg->fc_type != RTN_LOCAL || !cfg->fc_dst ||
            fi->fib_prefsrsrc != cfg->fc_dst)
- if (inet_addr_type(fi->fib_prefsrsrc) != RTN_LOCAL)
+ if (inet_addr_type(&init_net, fi->fib_prefsrsrc) != RTN_LOCAL)
            goto err_inval;
    }
```

```
diff --git a/net/ipv4/icmp.c b/net/ipv4/icmp.c
index ccdef9a..7df8951 100644
```

```
--- a/net/ipv4/icmp.c
+++ b/net/ipv4/icmp.c
@@ -591,7 +591,7 @@ void icmp_send(struct sk_buff *skb_in, int type, int code, __be32 info)
    if (xfrm_decode_session_reverse(skb_in, &fl, AF_INET))
        goto out_unlock;

- if (inet_addr_type(fl.fl4_src) == RTN_LOCAL)
+ if (inet_addr_type(&init_net, fl.fl4_src) == RTN_LOCAL)
    err = __ip_route_output_key(&rt2, &fl);
    else {
        struct flowi fl2 = {};
@@ -734,7 +734,7 @@ static void icmp_unreach(struct sk_buff *skb)
    */
```

```

if (!sysctl_icmp_ignore_bogus_error_responses &&
-   inet_addr_type(iph->daddr) == RTN_BROADCAST) {
+   inet_addr_type(&init_net, iph->daddr) == RTN_BROADCAST) {
    if (net_ratelimit())
        printk(KERN_WARNING "%u.%u.%u.%u sent an invalid ICMP "
            "type %u, code %u "
diff --git a/net/ipv4/ip_options.c b/net/ipv4/ip_options.c
index 2f14745..4d31515 100644
--- a/net/ipv4/ip_options.c
+++ b/net/ipv4/ip_options.c
@@ -151,7 +151,7 @@ int ip_options_echo(struct ip_options * dopt, struct sk_buff * skb)
    __be32 addr;

    memcpy(&addr, sptr+soffset-1, 4);
-   if (inet_addr_type(addr) != RTN_LOCAL) {
+   if (inet_addr_type(&init_net, addr) != RTN_LOCAL) {
        dopt->ts_needtime = 1;
        soffset += 8;
    }
@@ -400,7 +400,7 @@ int ip_options_compile(struct ip_options * opt, struct sk_buff * skb)
    {
        __be32 addr;
        memcpy(&addr, &optptr[optptr[2]-1], 4);
-   if (inet_addr_type(addr) == RTN_UNICAST)
+   if (inet_addr_type(&init_net, addr) == RTN_UNICAST)
        break;
        if (skb)
            timeptr = (__be32*)&optptr[optptr[2]+3];
diff --git a/net/ipv4/ipvs/ip_vs_core.c b/net/ipv4/ipvs/ip_vs_core.c
index 041f512..963981a 100644
--- a/net/ipv4/ipvs/ip_vs_core.c
+++ b/net/ipv4/ipvs/ip_vs_core.c
@@ -423,7 +423,7 @@ int ip_vs_leave(struct ip_vs_service *svc, struct sk_buff *skb,
    and the destination is RTN_UNICAST (and not local), then create
    a cache_bypass connection entry */
    if (sysctl_ip_vs_cache_bypass && svc->fwmark
-   && (inet_addr_type(iph->daddr) == RTN_UNICAST)) {
+   && (inet_addr_type(&init_net, iph->daddr) == RTN_UNICAST)) {
        int ret, cs;
        struct ip_vs_conn *cp;

diff --git a/net/ipv4/ipvs/ip_vs_ctl.c b/net/ipv4/ipvs/ip_vs_ctl.c
index 693d924..39fd50d 100644
--- a/net/ipv4/ipvs/ip_vs_ctl.c
+++ b/net/ipv4/ipvs/ip_vs_ctl.c
@@ -704,7 +704,7 @@ int __ip_vs_update_dest(struct ip_vs_service *svc,
    conn_flags = udest->conn_flags | IP_VS_CONN_F_INACTIVE;

```

```

/* check if local node and update the flags */
- if (inet_addr_type(udest->addr) == RTN_LOCAL) {
+ if (inet_addr_type(&init_net, udest->addr) == RTN_LOCAL) {
    conn_flags = (conn_flags & ~IP_VS_CONN_F_FWD_MASK)
    | IP_VS_CONN_F_LOCALNODE;
}
@@ -756,7 +756,7 @@ ip_vs_new_dest(struct ip_vs_service *svc, struct ip_vs_dest_user
*udest,

```

EnterFunction(2);

```

- atype = inet_addr_type(udest->addr);
+ atype = inet_addr_type(&init_net, udest->addr);
  if (atype != RTN_LOCAL && atype != RTN_UNICAST)
    return -EINVAL;

```

```

diff --git a/net/ipv4/netfilter.c b/net/ipv4/netfilter.c
index 7bf5e4a..833b9bd 100644

```

```

--- a/net/ipv4/netfilter.c
+++ b/net/ipv4/netfilter.c
@@ -19,7 +19,7 @@ int ip_route_me_harder(struct sk_buff *skb, unsigned addr_type)
    unsigned int hh_len;
    unsigned int type;

```

```

- type = inet_addr_type(iph->saddr);
+ type = inet_addr_type(&init_net, iph->saddr);
  if (addr_type == RTN_UNSPEC)
    addr_type = type;

```

```

diff --git a/net/ipv4/netfilter/ipt_addrtype.c b/net/ipv4/netfilter/ipt_addrtype.c
index 14394c6..8763902 100644

```

```

--- a/net/ipv4/netfilter/ipt_addrtype.c
+++ b/net/ipv4/netfilter/ipt_addrtype.c
@@ -26,7 +26,7 @@ MODULE_DESCRIPTION("iptables addrtype match");
static inline bool match_type(const struct net_device *dev, __be32 addr,
    u_int16_t mask)
{
- return !(mask & (1 << inet_dev_addr_type(dev, addr)));
+ return !(mask & (1 << inet_dev_addr_type(&init_net, dev, addr)));
}

```

static bool

```

diff --git a/net/ipv4/raw.c b/net/ipv4/raw.c
index 1bd188f..7fc88f8 100644

```

```

--- a/net/ipv4/raw.c
+++ b/net/ipv4/raw.c
@@ -618,7 +618,7 @@ static int raw_bind(struct sock *sk, struct sockaddr *uaddr, int addr_len)

```

```

if (sk->sk_state != TCP_CLOSE || addr_len < sizeof(struct sockaddr_in))
    goto out;
- chk_addr_ret = inet_addr_type(addr->sin_addr.s_addr);
+ chk_addr_ret = inet_addr_type(&init_net, addr->sin_addr.s_addr);
ret = -EADDRNOTAVAIL;
if (addr->sin_addr.s_addr && chk_addr_ret != RTN_LOCAL &&
    chk_addr_ret != RTN_MULTICAST && chk_addr_ret != RTN_BROADCAST)
diff --git a/net/ipv4/route.c b/net/ipv4/route.c
index e35076e..601ac85 100644
--- a/net/ipv4/route.c
+++ b/net/ipv4/route.c
@@ -1164,7 +1164,7 @@ void ip_rt_redirect(__be32 old_gw, __be32 daddr, __be32 new_gw,
    if (IN_DEV_SEC_REDIRECTS(in_dev) && ip_fib_check_default(new_gw, dev))
        goto reject_redirect;
    } else {
- if (inet_addr_type(new_gw) != RTN_UNICAST)
+ if (inet_addr_type(&init_net, new_gw) != RTN_UNICAST)
        goto reject_redirect;
    }

diff --git a/net/ipv6/af_inet6.c b/net/ipv6/af_inet6.c
index 34c2053..29ab300 100644
--- a/net/ipv6/af_inet6.c
+++ b/net/ipv6/af_inet6.c
@@ -280,7 +280,7 @@ int inet6_bind(struct socket *sock, struct sockaddr *uaddr, int addr_len)
    /* Check if the address belongs to the host. */
    if (addr_type == IPV6_ADDR_MAPPED) {
        v4addr = addr->sin6_addr.s6_addr32[3];
- if (inet_addr_type(v4addr) != RTN_LOCAL) {
+ if (inet_addr_type(&init_net, v4addr) != RTN_LOCAL) {
        err = -EADDRNOTAVAIL;
        goto out;
    }

diff --git a/net/sctp/protocol.c b/net/sctp/protocol.c
index dc22d71..0bed129 100644
--- a/net/sctp/protocol.c
+++ b/net/sctp/protocol.c
@@ -372,7 +372,7 @@ static int sctp_v4_addr_valid(union sctp_addr *addr,
    /* Should this be available for binding? */
    static int sctp_v4_available(union sctp_addr *addr, struct sctp_sock *sp)
    {
- int ret = inet_addr_type(addr->v4.sin_addr.s_addr);
+ int ret = inet_addr_type(&init_net, addr->v4.sin_addr.s_addr);

    if (addr->v4.sin_addr.s_addr != INADDR_ANY &&
--

```