
Subject: [PATCH 2.6.25 2/6] net: Make rtnetlink infrastructure network namespace aware (v3)

Posted by [den](#) on Thu, 15 Nov 2007 15:57:41 GMT

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

After this patch none of the netlink callback support anything except the initial network namespace but the rtnetlink infrastructure now handles multiple network namespaces.

Changes from v2:

- IPv6 addrlabel processing

Changes from v1:

- no need for special rtnl_unlock handling
- fixed IPv6 ndisc

Signed-off-by: Denis V. Lunev <den@openvz.org>

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

```
include/linux/rtnetlink.h | 8 +++---
include/net/net_namespace.h | 3 ++
net/bridge/br_netlink.c | 4 +-
net/core/fib_rules.c | 4 +-
net/core/neighbour.c | 4 +-
net/core/rtnetlink.c | 63 ++++++-----
net/decnet/dn_dev.c | 4 +-
net/decnet/dn_route.c | 2 +-
net/decnet/dn_table.c | 4 +-
net/ipv4/devinet.c | 4 +-
net/ipv4/fib_semantics.c | 4 +-
net/ipv4/ipmr.c | 4 +-
net/ipv4/route.c | 2 +-
net/ipv6/addrconf.c | 14 +++++-----
net/ipv6/addrlabel.c | 2 +-
net/ipv6/ndisc.c | 5 +-
net/ipv6/route.c | 6 +-
net/sched/act_api.c | 8 +++---
net/sched/cls_api.c | 2 +-
net/sched/sch_api.c | 4 +-
net/wireless/wext.c | 5 +++-
21 files changed, 102 insertions(+), 54 deletions(-)
```

diff --git a/include/linux/rtnetlink.h b/include/linux/rtnetlink.h

index e20dcc8..b014f6b 100644

--- a/include/linux/rtnetlink.h

+++ b/include/linux/rtnetlink.h

```
@@ -620,11 +620,11 @@ extern int __rtattr_parse_nested_compat(struct rtattr *tb[], int maxattr,
({ data = RTA_PAYLOAD(rta) >= len ? RTA_DATA(rta) : NULL; \
```

```

__rtattr_parse_nested_compat(tb, max, rta, len); })

-extern int rtnetlink_send(struct sk_buff *skb, u32 pid, u32 group, int echo);
-extern int rtnl_unicast(struct sk_buff *skb, u32 pid);
-extern int rtnl_notify(struct sk_buff *skb, u32 pid, u32 group,
+extern int rtnetlink_send(struct sk_buff *skb, struct net *net, u32 pid, u32 group, int echo);
+extern int rtnl_unicast(struct sk_buff *skb, struct net *net, u32 pid);
+extern int rtnl_notify(struct sk_buff *skb, struct net *net, u32 pid, u32 group,
    struct nlmsg_hdr *nlh, gfp_t flags);
-extern void rtnl_set_sk_err(u32 group, int error);
+extern void rtnl_set_sk_err(struct net *net, u32 group, int error);
extern int rtnetlink_put_metrics(struct sk_buff *skb, u32 *metrics);
extern int rtnl_put_cacheinfo(struct sk_buff *skb, struct dst_entry *dst,
    u32 id, u32 ts, u32 tsage, long expires,
diff --git a/include/net/net_namespace.h b/include/net/net_namespace.h
index 5dd6d90..90802a6 100644
--- a/include/net/net_namespace.h
+++ b/include/net/net_namespace.h
@@ -10,6 +10,7 @@

struct proc_dir_entry;
struct net_device;
+struct sock;
struct net {
    atomic_t count; /* To decided when the network
        * namespace should be freed.
@@ -29,6 +30,8 @@ struct net {
    struct list_head dev_base_head;
    struct hlist_head *dev_name_head;
    struct hlist_head *dev_index_head;
+
+ struct sock *rtnl; /* rtnetlink socket */
};

#ifdef CONFIG_NET
diff --git a/net/bridge/br_netlink.c b/net/bridge/br_netlink.c
index a4ffa2b..f5d6933 100644
--- a/net/bridge/br_netlink.c
+++ b/net/bridge/br_netlink.c
@@ -97,10 +97,10 @@ void br_ifinfo_notify(int event, struct net_bridge_port *port)
    kfree_skb(skb);
    goto errout;
}
- err = rtnl_notify(skb, 0, RTNLGRP_LINK, NULL, GFP_ATOMIC);
+ err = rtnl_notify(skb, &init_net, 0, RTNLGRP_LINK, NULL, GFP_ATOMIC);
errout:
    if (err < 0)
- rtnl_set_sk_err(RTNLGRP_LINK, err);

```

```

+ rtnl_set_sk_err(&init_net, RTNLGRP_LINK, err);
}

/*
diff --git a/net/core/fib_rules.c b/net/core/fib_rules.c
index 3b20b6f..0af0538 100644
--- a/net/core/fib_rules.c
+++ b/net/core/fib_rules.c
@@ -599,10 +599,10 @@ static void notify_rule_change(int event, struct fib_rule *rule,
    kfree_skb(skb);
    goto errout;
}
- err = rtnl_notify(skb, pid, ops->nlggroup, nlh, GFP_KERNEL);
+ err = rtnl_notify(skb, &init_net, pid, ops->nlggroup, nlh, GFP_KERNEL);
errout:
    if (err < 0)
- rtnl_set_sk_err(ops->nlggroup, err);
+ rtnl_set_sk_err(&init_net, ops->nlggroup, err);
}

static void attach_rules(struct list_head *rules, struct net_device *dev)
diff --git a/net/core/neighbour.c b/net/core/neighbour.c
index 29f0a4d..a8b72c1 100644
--- a/net/core/neighbour.c
+++ b/net/core/neighbour.c
@@ -2467,10 +2467,10 @@ static void __neigh_notify(struct neighbour *n, int type, int flags)
    kfree_skb(skb);
    goto errout;
}
- err = rtnl_notify(skb, 0, RTNLGRP_NEIGH, NULL, GFP_ATOMIC);
+ err = rtnl_notify(skb, &init_net, 0, RTNLGRP_NEIGH, NULL, GFP_ATOMIC);
errout:
    if (err < 0)
- rtnl_set_sk_err(RTNLGRP_NEIGH, err);
+ rtnl_set_sk_err(&init_net, RTNLGRP_NEIGH, err);
}

#ifdef CONFIG_ARPD
diff --git a/net/core/rtnetlink.c b/net/core/rtnetlink.c
index 219bb31..be8e10c 100644
--- a/net/core/rtnetlink.c
+++ b/net/core/rtnetlink.c
@@ -60,7 +60,6 @@ struct rtnl_link
};

static DEFINE_MUTEX(rtnl_mutex);
-static struct sock *rtnl;

```

```

void rtnl_lock(void)
{
@@ -455,8 +454,9 @@ size_t rtattr_strncpy(char *dest, const struct rtattr *rta, size_t size)
    return ret;
}

-int rtnetlink_send(struct sk_buff *skb, u32 pid, unsigned group, int echo)
+int rtnetlink_send(struct sk_buff *skb, struct net *net, u32 pid, unsigned group, int echo)
{
+ struct sock *rtnl = net->rtnl;
    int err = 0;

    NETLINK_CB(skb).dst_group = group;
@@ -468,14 +468,17 @@ int rtnetlink_send(struct sk_buff *skb, u32 pid, unsigned group, int
echo)
    return err;
}

-int rtnl_unicast(struct sk_buff *skb, u32 pid)
+int rtnl_unicast(struct sk_buff *skb, struct net *net, u32 pid)
{
+ struct sock *rtnl = net->rtnl;
+
    return nlmsg_unicast(rtnl, skb, pid);
}

-int rtnl_notify(struct sk_buff *skb, u32 pid, u32 group,
+int rtnl_notify(struct sk_buff *skb, struct net *net, u32 pid, u32 group,
    struct nlmsghdr *nlh, gfp_t flags)
{
+ struct sock *rtnl = net->rtnl;
    int report = 0;

    if (nlh)
@@ -484,8 +487,10 @@ int rtnl_notify(struct sk_buff *skb, u32 pid, u32 group,
    return nlmsg_notify(rtnl, skb, pid, group, report, flags);
}

-void rtnl_set_sk_err(u32 group, int error)
+void rtnl_set_sk_err(struct net *net, u32 group, int error)
{
+ struct sock *rtnl = net->rtnl;
+
    netlink_set_err(rtnl, 0, group, error);
}

@@ -1198,7 +1203,7 @@ static int rtnl_getlink(struct sk_buff *skb, struct nlmsghdr* nlh, void
*arg)

```

```

    kfree_skb(nskb);
    goto errout;
}
- err = rtnl_unicast(nskb, NETLINK_CB(skb).pid);
+ err = rtnl_unicast(nskb, net, NETLINK_CB(skb).pid);
errout:
    dev_put(dev);

@@ -1249,10 +1254,10 @@ void rtmsg_ifinfo(int type, struct net_device *dev, unsigned change)
    kfree_skb(skb);
    goto errout;
}
- err = rtnl_notify(skb, 0, RTNLGRP_LINK, NULL, GFP_KERNEL);
+ err = rtnl_notify(skb, &init_net, 0, RTNLGRP_LINK, NULL, GFP_KERNEL);
errout:
    if (err < 0)
-    rtnl_set_sk_err(RTNLGRP_LINK, err);
+    rtnl_set_sk_err(&init_net, RTNLGRP_LINK, err);
}

/* Protected by RTNL semaphore. */
@@ -1263,6 +1268,7 @@ static int rtattr_max;

static int rtnetlink_rcv_msg(struct sk_buff *skb, struct nlmsghdr *nlh)
{
+ struct net *net = skb->sk->sk_net;
    rtnl_doit_func doit;
    int sz_idx, kind;
    int min_len;
@@ -1291,6 +1297,7 @@ static int rtnetlink_rcv_msg(struct sk_buff *skb, struct nlmsghdr *nlh)
    return -EPERM;

    if (kind == 2 && nlh->nlmsg_flags & NLM_F_DUMP) {
+ struct sock *rtnl;
        rtnl_dumpit_func dumpit;

        dumpit = rtnl_get_dumpit(family, type);
@@ -1298,6 +1305,7 @@ static int rtnetlink_rcv_msg(struct sk_buff *skb, struct nlmsghdr *nlh)
    return -EOPNOTSUPP;

    __rtnl_unlock();
+ rtnl = net->rtnl;
    err = netlink_dump_start(rtnl, skb, nlh, dumpit, NULL);
    rtnl_lock();
    return err;
@@ -1370,6 +1378,40 @@ static struct notifier_block rtnetlink_dev_notifier = {
    .notifier_call = rtnetlink_event,
};

```

```

+
+static int rtnetlink_net_init(struct net *net)
+{
+ struct sock *sk;
+ sk = netlink_kernel_create(net, NETLINK_ROUTE, RTNLGRP_MAX,
+   rtnetlink_rcv, &rtnl_mutex, THIS_MODULE);
+ if (!sk)
+   return -ENOMEM;
+
+ /* Don't hold an extra reference on the namespace */
+ put_net(sk->sk_net);
+ net->rtnl = sk;
+ return 0;
+}
+
+static void rtnetlink_net_exit(struct net *net)
+{
+ struct sock *sk = net->rtnl;
+ if (sk) {
+   /* At the last minute lie and say this is a socket for the
+    * initial network namespace. So the socket will be safe to
+    * free.
+    */
+   sk->sk_net = get_net(&init_net);
+   sock_put(sk);
+   net->rtnl = NULL;
+ }
+}
+
+static struct pernet_operations rtnetlink_net_ops = {
+ .init = rtnetlink_net_init,
+ .exit = rtnetlink_net_exit,
+};
+
+void __init rtnetlink_init(void)
+{
+   int i;
@@ -1382,10 +1424,9 @@ void __init rtnetlink_init(void)
+   if (!rta_buf)
+     panic("rtnetlink_init: cannot allocate rta_buf\n");

- rtnl = netlink_kernel_create(&init_net, NETLINK_ROUTE, RTNLGRP_MAX,
-   rtnetlink_rcv, &rtnl_mutex, THIS_MODULE);
- if (rtnl == NULL)
+ if (register_pernet_subsys(&rtnetlink_net_ops))
   panic("rtnetlink_init: cannot initialize rtnetlink\n");
+

```

```

netlink_set_nonroot(NETLINK_ROUTE, NL_NONROOT_RECV);
register_netdevice_notifier(&rtnetlink_dev_notifier);

diff --git a/net/decnet/dn_dev.c b/net/decnet/dn_dev.c
index a92f3f9..b665640 100644
--- a/net/decnet/dn_dev.c
+++ b/net/decnet/dn_dev.c
@@ -791,10 +791,10 @@ static void dn_ifaddr_notify(int event, struct dn_ifaddr *ifa)
    kfree_skb(skb);
    goto errout;
}
- err = rtnl_notify(skb, 0, RTNLGRP_DECnet_IFADDR, NULL, GFP_KERNEL);
+ err = rtnl_notify(skb, &init_net, 0, RTNLGRP_DECnet_IFADDR, NULL, GFP_KERNEL);
errout:
    if (err < 0)
-    rtnl_set_sk_err(RTNLGRP_DECnet_IFADDR, err);
+    rtnl_set_sk_err(&init_net, RTNLGRP_DECnet_IFADDR, err);
}

static int dn_nl_dump_ifaddr(struct sk_buff *skb, struct netlink_callback *cb)
diff --git a/net/decnet/dn_route.c b/net/decnet/dn_route.c
index 7cf97cf..eb07df9 100644
--- a/net/decnet/dn_route.c
+++ b/net/decnet/dn_route.c
@@ -1587,7 +1587,7 @@ static int dn_cache_getroute(struct sk_buff *in_skb, struct nlmsghdr
*nlh, void
    goto out_free;
}

- return rtnl_unicast(skb, NETLINK_CB(in_skb).pid);
+ return rtnl_unicast(skb, &init_net, NETLINK_CB(in_skb).pid);

out_free:
    kfree_skb(skb);
diff --git a/net/decnet/dn_table.c b/net/decnet/dn_table.c
index a3bdb8d..e09d915 100644
--- a/net/decnet/dn_table.c
+++ b/net/decnet/dn_table.c
@@ -375,10 +375,10 @@ static void dn_rtmsg_fib(int event, struct dn_fib_node *f, int z, u32
tb_id,
    kfree_skb(skb);
    goto errout;
}
- err = rtnl_notify(skb, pid, RTNLGRP_DECnet_ROUTE, nlh, GFP_KERNEL);
+ err = rtnl_notify(skb, &init_net, pid, RTNLGRP_DECnet_ROUTE, nlh, GFP_KERNEL);
errout:
    if (err < 0)
-    rtnl_set_sk_err(RTNLGRP_DECnet_ROUTE, err);

```

```

+ rtnl_set_sk_err(&init_net, RTNLGRP_DECnet_ROUTE, err);
}

static __inline__ int dn_hash_dump_bucket(struct sk_buff *skb,
diff --git a/net/ipv4/devinet.c b/net/ipv4/devinet.c
index 0ff2ef6..1ae7dcf 100644
--- a/net/ipv4/devinet.c
+++ b/net/ipv4/devinet.c
@@ -1241,10 +1241,10 @@ static void rtmsg_ifa(int event, struct in_ifaddr* ifa, struct nlmsghdr
*nlh,
    kfree_skb(skb);
    goto errout;
}
- err = rtnl_notify(skb, pid, RTNLGRP_IPV4_IFADDR, nlh, GFP_KERNEL);
+ err = rtnl_notify(skb, &init_net, pid, RTNLGRP_IPV4_IFADDR, nlh, GFP_KERNEL);
errout:
    if (err < 0)
- rtnl_set_sk_err(RTNLGRP_IPV4_IFADDR, err);
+ rtnl_set_sk_err(&init_net, RTNLGRP_IPV4_IFADDR, err);
}

#ifdef CONFIG_SYSCTL
diff --git a/net/ipv4/fib_semantics.c b/net/ipv4/fib_semantics.c
index 1351a26..33ec960 100644
--- a/net/ipv4/fib_semantics.c
+++ b/net/ipv4/fib_semantics.c
@@ -320,11 +320,11 @@ void rtmsg_fib(int event, __be32 key, struct fib_alias *fa,
    kfree_skb(skb);
    goto errout;
}
- err = rtnl_notify(skb, info->pid, RTNLGRP_IPV4_ROUTE,
+ err = rtnl_notify(skb, &init_net, info->pid, RTNLGRP_IPV4_ROUTE,
    info->nlh, GFP_KERNEL);
errout:
    if (err < 0)
- rtnl_set_sk_err(RTNLGRP_IPV4_ROUTE, err);
+ rtnl_set_sk_err(&init_net, RTNLGRP_IPV4_ROUTE, err);
}

/* Return the first fib alias matching TOS with
diff --git a/net/ipv4/ipmr.c b/net/ipv4/ipmr.c
index 8e5d47a..1187928 100644
--- a/net/ipv4/ipmr.c
+++ b/net/ipv4/ipmr.c
@@ -321,7 +321,7 @@ static void ipmr_destroy_unres(struct mfc_cache *c)
    e->error = -ETIMEDOUT;
    memset(&e->msg, 0, sizeof(e->msg));

```

```

- rtnl_unicast(skb, NETLINK_CB(skb).pid);
+ rtnl_unicast(skb, &init_net, NETLINK_CB(skb).pid);
  } else
    kfree_skb(skb);
  }
@@ -533,7 +533,7 @@ static void ipmr_cache_resolve(struct mfc_cache *uc, struct mfc_cache
*c)
    memset(&e->msg, 0, sizeof(e->msg));
  }

- rtnl_unicast(skb, NETLINK_CB(skb).pid);
+ rtnl_unicast(skb, &init_net, NETLINK_CB(skb).pid);
  } else
    ip_mr_forward(skb, c, 0);
  }
diff --git a/net/ipv4/route.c b/net/ipv4/route.c
index 4bb2db1..61efbfc 100644
--- a/net/ipv4/route.c
+++ b/net/ipv4/route.c
@@ -2607,7 +2607,7 @@ static int inet_rtm_getroute(struct sk_buff *in_skb, struct nlmsghdr*
nlh, void
    if (err <= 0)
        goto errout_free;

- err = rtnl_unicast(skb, NETLINK_CB(in_skb).pid);
+ err = rtnl_unicast(skb, &init_net, NETLINK_CB(in_skb).pid);
errout:
    return err;

diff --git a/net/ipv6/addrconf.c b/net/ipv6/addrconf.c
index 88ad440..24d340a 100644
--- a/net/ipv6/addrconf.c
+++ b/net/ipv6/addrconf.c
@@ -3388,7 +3388,7 @@ static int inet6_rtm_getaddr(struct sk_buff *in_skb, struct nlmsghdr*
nlh,
    kfree_skb(skb);
    goto errout_ifa;
  }
- err = rtnl_unicast(skb, NETLINK_CB(in_skb).pid);
+ err = rtnl_unicast(skb, &init_net, NETLINK_CB(in_skb).pid);
errout_ifa:
    in6_ifa_put(ifa);
errout:
@@ -3411,10 +3411,10 @@ static void inet6_ifa_notify(int event, struct inet6_ifaddr *ifa)
    kfree_skb(skb);
    goto errout;
  }
- err = rtnl_notify(skb, 0, RTNLGRP_IPV6_IFADDR, NULL, GFP_ATOMIC);

```

```

+ err = rtnl_notify(skb, &init_net, 0, RTNLGRP_IPV6_IFADDR, NULL, GFP_ATOMIC);
errout:
    if (err < 0)
-   rtnl_set_sk_err(RTNLGRP_IPV6_IFADDR, err);
+   rtnl_set_sk_err(&init_net, RTNLGRP_IPV6_IFADDR, err);
}

static inline void ipv6_store_devconf(struct ipv6_devconf *cnf,
@@ -3619,10 +3619,10 @@ void inet6_ifinfo_notify(int event, struct inet6_dev *idev)
    kfree_skb(skb);
    goto errout;
}
- err = rtnl_notify(skb, 0, RTNLGRP_IPV6_IFADDR, NULL, GFP_ATOMIC);
+ err = rtnl_notify(skb, &init_net, 0, RTNLGRP_IPV6_IFADDR, NULL, GFP_ATOMIC);
errout:
    if (err < 0)
-   rtnl_set_sk_err(RTNLGRP_IPV6_IFADDR, err);
+   rtnl_set_sk_err(&init_net, RTNLGRP_IPV6_IFADDR, err);
}

static inline size_t inet6_prefix_nlmsg_size(void)
@@ -3688,10 +3688,10 @@ static void inet6_prefix_notify(int event, struct inet6_dev *idev,
    kfree_skb(skb);
    goto errout;
}
- err = rtnl_notify(skb, 0, RTNLGRP_IPV6_PREFIX, NULL, GFP_ATOMIC);
+ err = rtnl_notify(skb, &init_net, 0, RTNLGRP_IPV6_PREFIX, NULL, GFP_ATOMIC);
errout:
    if (err < 0)
-   rtnl_set_sk_err(RTNLGRP_IPV6_PREFIX, err);
+   rtnl_set_sk_err(&init_net, RTNLGRP_IPV6_PREFIX, err);
}

static void __ipv6_ifa_notify(int event, struct inet6_ifaddr *ifp)
diff --git a/net/ipv6/addrlabel.c b/net/ipv6/addrlabel.c
index b9b5d57..6f1ca60 100644
--- a/net/ipv6/addrlabel.c
+++ b/net/ipv6/addrlabel.c
@@ -549,7 +549,7 @@ static int ip6addrlbl_get(struct sk_buff *in_skb, struct nlmsg_hdr* nlh,
    goto out;
}

- err = rtnl_unicast(skb, NETLINK_CB(in_skb).pid);
+ err = rtnl_unicast(skb, &init_net, NETLINK_CB(in_skb).pid);
out:
    return err;
}
diff --git a/net/ipv6/ndisc.c b/net/ipv6/ndisc.c

```

index 71b742a..39ff179 100644

--- a/net/ipv6/ndisc.c

+++ b/net/ipv6/ndisc.c

```
@@ -1049,7 +1049,8 @@ static void ndisc_ra_useropt(struct sk_buff *ra, struct nd_opt_hdr *opt)
    &ipv6_hdr(ra)->saddr);
    nlmsg_end(skb, nlh);
```

```
- err = rtnl_notify(skb, 0, RTNLGRP_ND_USEROPT, NULL, GFP_ATOMIC);
+ err = rtnl_notify(skb, &init_net, 0, RTNLGRP_ND_USEROPT, NULL,
+   GFP_ATOMIC);
    if (err < 0)
        goto errout;
```

```
@@ -1059,7 +1060,7 @@ nla_put_failure:
```

```
    nlmsg_free(skb);
    err = -EMSGSIZE;
errout:
- rtnl_set_sk_err(RTNLGRP_ND_USEROPT, err);
+ rtnl_set_sk_err(&init_net, RTNLGRP_ND_USEROPT, err);
}
```

```
static void ndisc_router_discovery(struct sk_buff *skb)
```

diff --git a/net/ipv6/route.c b/net/ipv6/route.c

index 6a2ca79..2597f3d 100644

--- a/net/ipv6/route.c

+++ b/net/ipv6/route.c

```
@@ -2225,7 +2225,7 @@ static int inet6_rtm_getroute(struct sk_buff *in_skb, struct nlmsghdr*
nlh, void
    goto errout;
}
```

```
- err = rtnl_unicast(skb, NETLINK_CB(in_skb).pid);
+ err = rtnl_unicast(skb, &init_net, NETLINK_CB(in_skb).pid);
errout:
    return err;
}
```

```
@@ -2255,10 +2255,10 @@ void inet6_rt_notify(int event, struct rt6_info *rt, struct nl_info *info)
    kfree_skb(skb);
    goto errout;
}
```

```
- err = rtnl_notify(skb, pid, RTNLGRP_IPV6_ROUTE, nlh, gfp_any());
+ err = rtnl_notify(skb, &init_net, pid, RTNLGRP_IPV6_ROUTE, nlh, gfp_any());
errout:
    if (err < 0)
-   rtnl_set_sk_err(RTNLGRP_IPV6_ROUTE, err);
+   rtnl_set_sk_err(&init_net, RTNLGRP_IPV6_ROUTE, err);
}
```

```

/*
diff --git a/net/sched/act_api.c b/net/sched/act_api.c
index 8528291..8150647 100644
--- a/net/sched/act_api.c
+++ b/net/sched/act_api.c
@@ -660,7 +660,7 @@ act_get_notify(u32 pid, struct nlmsg_hdr *n, struct tc_action *a, int event)
    return -EINVAL;
}

- return rtnl_unicast(skb, pid);
+ return rtnl_unicast(skb, &init_net, pid);
}

static struct tc_action *
@@ -781,7 +781,7 @@ static int tca_action_flush(struct rtattr *rta, struct nlmsg_hdr *n, u32 pid)
    nlh->nlmsg_flags |= NLM_F_ROOT;
    module_put(a->ops->owner);
    kfree(a);
- err = rtnetlink_send(skb, pid, RTNLGRP_TC, n->nlmsg_flags&NLM_F_ECHO);
+ err = rtnetlink_send(skb, &init_net, pid, RTNLGRP_TC, n->nlmsg_flags&NLM_F_ECHO);
    if (err > 0)
        return 0;

@@ -844,7 +844,7 @@ tca_action_gd(struct rtattr *rta, struct nlmsg_hdr *n, u32 pid, int event)

    /* now do the delete */
    tcf_action_destroy(head, 0);
- ret = rtnetlink_send(skb, pid, RTNLGRP_TC,
+ ret = rtnetlink_send(skb, &init_net, pid, RTNLGRP_TC,
        n->nlmsg_flags&NLM_F_ECHO);
    if (ret > 0)
        return 0;
@@ -888,7 +888,7 @@ static int tcf_add_notify(struct tc_action *a, u32 pid, u32 seq, int event,
    nlh->nlmsg_len = skb_tail_pointer(skb) - b;
    NETLINK_CB(skb).dst_group = RTNLGRP_TC;

- err = rtnetlink_send(skb, pid, RTNLGRP_TC, flags&NLM_F_ECHO);
+ err = rtnetlink_send(skb, &init_net, pid, RTNLGRP_TC, flags&NLM_F_ECHO);
    if (err > 0)
        err = 0;
    return err;
diff --git a/net/sched/cls_api.c b/net/sched/cls_api.c
index fdab6a5..80dccac 100644
--- a/net/sched/cls_api.c
+++ b/net/sched/cls_api.c
@@ -361,7 +361,7 @@ static int tfilter_notify(struct sk_buff *oskb, struct nlmsg_hdr *n,
    return -EINVAL;
}

```

```
- return rtnetlink_send(skb, pid, RTNLGRP_TC, n->nmsg_flags&NLM_F_ECHO);
+ return rtnetlink_send(skb, &init_net, pid, RTNLGRP_TC, n->nmsg_flags&NLM_F_ECHO);
}
```

```
struct tcf_dump_args
diff --git a/net/sched/sch_api.c b/net/sched/sch_api.c
index f30e3f7..273c628 100644
```

```
--- a/net/sched/sch_api.c
+++ b/net/sched/sch_api.c
@@ -872,7 +872,7 @@ static int qdisc_notify(struct sk_buff *oskb, struct nlmsghdr *n,
}
```

```
if (skb->len)
- return rtnetlink_send(skb, pid, RTNLGRP_TC, n->nmsg_flags&NLM_F_ECHO);
+ return rtnetlink_send(skb, &init_net, pid, RTNLGRP_TC, n->nmsg_flags&NLM_F_ECHO);
```

```
err_out:
kfree_skb(skb);
@@ -1103,7 +1103,7 @@ static int tclass_notify(struct sk_buff *oskb, struct nlmsghdr *n,
return -EINVAL;
}
```

```
- return rtnetlink_send(skb, pid, RTNLGRP_TC, n->nmsg_flags&NLM_F_ECHO);
+ return rtnetlink_send(skb, &init_net, pid, RTNLGRP_TC, n->nmsg_flags&NLM_F_ECHO);
}
```

```
struct qdisc_dump_args
diff --git a/net/wireless/wext.c b/net/wireless/wext.c
index 85e5f9d..85805f2 100644
```

```
--- a/net/wireless/wext.c
+++ b/net/wireless/wext.c
@@ -1137,7 +1137,7 @@ static void wireless_nlevent_process(unsigned long data)
struct sk_buff *skb;
```

```
while ((skb = skb_dequeue(&wireless_nlevent_queue)))
- rtnl_notify(skb, 0, RTNLGRP_LINK, NULL, GFP_ATOMIC);
+ rtnl_notify(skb, &init_net, 0, RTNLGRP_LINK, NULL, GFP_ATOMIC);
}
```

```
static DECLARE_TASKLET(wireless_nlevent_tasklet, wireless_nlevent_process, 0);
@@ -1189,6 +1189,9 @@ static void rtmsg_iwinfo(struct net_device *dev, char *event, int
event_len)
struct sk_buff *skb;
int err;
```

```
+ if (dev->nd_net != &init_net)
+ return;
```

```
+
skb = nlmsg_new(NLMSG_DEFAULT_SIZE, GFP_ATOMIC);
if (!skb)
return;
```

--

1.5.3.rc5

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>
