Subject: [PATCH net-2.6.25 15/19] [NETNS] Provide correct namespace for fibnl netlink socket.
Posted by den on Wed, 09 Jan 2008 17:59:19 GMT
View Forum Message <> Reply to Message

This patch makes the netlink socket to be per namespace. That allows to have
each namespace its own socket for routing queries.

Acked-by: Benjamin Thery <benjamin.thery@bull.net>
Acked-by: Daniel Lezcano <dlezcano@fr.ibm.com>
Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 include/net/netns/ipv4.h | 2 ++
 net/ipv4/fib_frontend.c | 24 ++++++++++++++++++--------
 2 files changed, 18 insertions(+), 8 deletions(-)

diff --git a/include/net/netns/ipv4.h b/include/net/netns/ipv4.h
index 629ec6c..031d761 100644
--- a/include/net/netns/ipv4.h
+++ b/include/net/netns/ipv4.h
@@ -9,6 +9,7 @@ struct ctl_table_header;
 struct ipv4_devconf;
 struct fib_rules_ops;
 struct hlist_head;
+struct sock;

 struct netns_ipv4 {
 #ifdef CONFIG_SYSCTL
@@ -18,5 +19,6 @@ struct netns_ipv4 {
  struct fib_rules_ops *rules_ops;
 #endif
  struct hlist_head *fib_table_hash;
+ struct sock *fibnl;
 };
 #endif
diff --git a/net/ipv4/fib_frontend.c b/net/ipv4/fib_frontend.c
index 7fe54a3..a5e8167 100644
--- a/net/ipv4/fib_frontend.c
+++ b/net/ipv4/fib_frontend.c
@@ -49,8 +49,6 @@

 #define FFprint(a...) printk(KERN_DEBUG a)

-static struct sock *fibnl;
-
 #ifndef CONFIG_IP_MULTIPLE_TABLES

 static int __net_init fib4_rules_init(struct net *net)

```
@@ -845,11 +843,13 @@ static void nl_fib_lookup(struct fib_result_nl *frn, struct fib_table *tb )

 static void nl_fib_input(struct sk_buff *skb)
 {
+ struct net *net;
  struct fib_result_nl *frn;
  struct nlmsghdr *nlh;
  struct fib_table *tb;
  u32 pid;

+ net = skb->sk->sk_net;
  nlh = nlmsg_hdr(skb);
  if (skb->len < NLMSG_SPACE(0) || skb->len < nlh->nlmsg_len ||
      nlh->nlmsg_len < NLMSG_LENGTH(sizeof(*frn)))
@@ -861,28 +861,36 @@ static void nl_fib_input(struct sk_buff *skb)
  nlh = nlmsg_hdr(skb);

  frn = (struct fib_result_nl *) NLMSG_DATA(nlh);
- tb = fib_get_table(&init_net, frn->tb_id_in);
+ tb = fib_get_table(net, frn->tb_id_in);

  nl_fib_lookup(frn, tb);

  pid = NETLINK_CB(skb).pid;       /* pid of sending process */
  NETLINK_CB(skb).pid = 0;         /* from kernel */
  NETLINK_CB(skb).dst_group = 0;  /* unicast */
- netlink_unicast(fibnl, skb, pid, MSG_DONTWAIT);
+ netlink_unicast(net->ipv4.fibnl, skb, pid, MSG_DONTWAIT);
 }

 static int nl_fib_lookup_init(struct net *net)
 {
- fibnl = netlink_kernel_create(net, NETLINK_FIB_LOOKUP, 0,
-          nl_fib_input, NULL, THIS_MODULE);
- if (fibnl == NULL)
+ struct sock *sk;
+ sk = netlink_kernel_create(net, NETLINK_FIB_LOOKUP, 0,
+        nl_fib_input, NULL, THIS_MODULE);
+ if (sk == NULL)
    return -EAFNOSUPPORT;
+ /* Don't hold an extra reference on the namespace */
+ put_net(sk->sk_net);
+ net->ipv4.fibnl = sk;
  return 0;
 }

 static void nl_fib_lookup_exit(struct net *net)
 {
```

```
- sock_put(fibnl);
+ /* At the last minute lie and say this is a socket for the
+  * initial network namespace. So the socket will  be safe to free.
+  */
+ net->ipv4.fibnl->sk_net = get_net(&init_net);
+ sock_put(net->ipv4.fibnl);
 }

 static void fib_disable_ip(struct net_device *dev, int force)
```