Subject: [PATCH net-2.6.25 3/19] [NETNS] Namespacing in the generic fib rules code.
Posted by den on Wed, 09 Jan 2008 18:01:15 GMT
View Forum Message <> Reply to Message

Move static rules_ops & rules_mod_lock to the struct net, register the
pernet subsys to init them and enjoy the fact that the core rules
infrastructure works in the namespace.

Real IPv4 fib rules virtualization requires fib tables support in the
namespace and will be done seriously later in the patchset.

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/net_namespace.h | 4 ++
 net/core/fib_rules.c | 91 ++++++++++++++++++++++++++++++++-----------
 2 files changed, 58 insertions(+), 37 deletions(-)

diff --git a/include/net/net_namespace.h b/include/net/net_namespace.h
index d04ddf2..a5055fa 100644
--- a/include/net/net_namespace.h
+++ b/include/net/net_namespace.h
@@ -39,6 +39,10 @@ struct net {
  struct hlist_head *dev_name_head;
  struct hlist_head *dev_index_head;

+ /* core fib_rules */
+ struct list_head rules_ops;
+ spinlock_t rules_mod_lock;
+
  struct sock   *rtnl;   /* rtnetlink socket */

  /* core sysctls */
diff --git a/net/core/fib_rules.c b/net/core/fib_rules.c
index e12e9f5..c5f78fe 100644
--- a/net/core/fib_rules.c
+++ b/net/core/fib_rules.c
@@ -15,9 +15,6 @@
 #include <net/sock.h>
 #include <net/fib_rules.h>

-static LIST_HEAD(rules_ops);
-static DEFINE_SPINLOCK(rules_mod_lock);
-
 int fib_default_rule_add(struct fib_rules_ops *ops,
     u32 pref, u32 table, u32 flags)

```
 {
@@ -40,16 +37,17 @@ int fib_default_rule_add(struct fib_rules_ops *ops,
 }
 EXPORT_SYMBOL(fib_default_rule_add);

-static void notify_rule_change(int event, struct fib_rule *rule,
+static void notify_rule_change(struct net *net, int event,
+          struct fib_rule *rule,
           struct fib_rules_ops *ops, struct nlmsghdr *nlh,
           u32 pid);

-static struct fib_rules_ops *lookup_rules_ops(int family)
+static struct fib_rules_ops *lookup_rules_ops(struct net *net, int family)
 {
  struct fib_rules_ops *ops;

  rcu_read_lock();
- list_for_each_entry_rcu(ops, &rules_ops, list) {
+ list_for_each_entry_rcu(ops, &net->rules_ops, list) {
   if (ops->family == family) {
    if (!try_module_get(ops->owner))
     ops = NULL;
@@ -87,15 +85,16 @@ int fib_rules_register(struct net *net, struct fib_rules_ops *ops)
      ops->action == NULL)
   return -EINVAL;

- spin_lock(&rules_mod_lock);
- list_for_each_entry(o, &rules_ops, list)
+ spin_lock(&net->rules_mod_lock);
+ list_for_each_entry(o, &net->rules_ops, list)
   if (ops->family == o->family)
    goto errout;

- list_add_tail_rcu(&ops->list, &rules_ops);
+ hold_net(net);
+ list_add_tail_rcu(&ops->list, &net->rules_ops);
  err = 0;
 errout:
- spin_unlock(&rules_mod_lock);
+ spin_unlock(&net->rules_mod_lock);

  return err;
 }
@@ -118,8 +117,8 @@ int fib_rules_unregister(struct net *net, struct fib_rules_ops *ops)
  int err = 0;
  struct fib_rules_ops *o;

- spin_lock(&rules_mod_lock);
```

```
- list_for_each_entry(o, &rules_ops, list) {
+ spin_lock(&net->rules_mod_lock);
+ list_for_each_entry(o, &net->rules_ops, list) {
   if (o == ops) {
    list_del_rcu(&o->list);
    fib_rules_cleanup_ops(ops);
@@ -129,9 +128,11 @@ int fib_rules_unregister(struct net *net, struct fib_rules_ops *ops)

  err = -ENOENT;
 out:
- spin_unlock(&rules_mod_lock);
+ spin_unlock(&net->rules_mod_lock);

  synchronize_rcu();
+ if (!err)
+  release_net(net);

  return err;
 }
@@ -229,13 +230,10 @@ static int fib_nl_newrule(struct sk_buff *skb, struct nlmsghdr* nlh, void
*arg)
  struct nlattr *tb[FRA_MAX+1];
  int err = -EINVAL, unresolved = 0;

- if (net != &init_net)
-  return -EINVAL;
-
  if (nlh->nlmsg_len < nlmsg_msg_size(sizeof(*frh)))
   goto errout;

- ops = lookup_rules_ops(frh->family);
+ ops = lookup_rules_ops(net, frh->family);
  if (ops == NULL) {
   err = EAFNOSUPPORT;
   goto errout;
@@ -348,7 +346,7 @@ static int fib_nl_newrule(struct sk_buff *skb, struct nlmsghdr* nlh, void
*arg)
  else
   list_add_rcu(&rule->list, &ops->rules_list);

- notify_rule_change(RTM_NEWRULE, rule, ops, nlh, NETLINK_CB(skb).pid);
+ notify_rule_change(net, RTM_NEWRULE, rule, ops, nlh, NETLINK_CB(skb).pid);
  flush_route_cache(ops);
  rules_ops_put(ops);
  return 0;
@@ -369,13 +367,10 @@ static int fib_nl_delrule(struct sk_buff *skb, struct nlmsghdr* nlh, void
*arg)
  struct nlattr *tb[FRA_MAX+1];
```

```
  int err = -EINVAL;

- if (net != &init_net)
-  return -EINVAL;
-
  if (nlh->nlmsg_len < nlmsg_msg_size(sizeof(*frh)))
   goto errout;

- ops = lookup_rules_ops(frh->family);
+ ops = lookup_rules_ops(net, frh->family);
  if (ops == NULL) {
   err = EAFNOSUPPORT;
   goto errout;
@@ -441,7 +436,7 @@ static int fib_nl_delrule(struct sk_buff *skb, struct nlmsghdr* nlh, void
*arg)
   }

   synchronize_rcu();
-  notify_rule_change(RTM_DELRULE, rule, ops, nlh,
+  notify_rule_change(net, RTM_DELRULE, rule, ops, nlh,
        NETLINK_CB(skb).pid);
   fib_rule_put(rule);
   flush_route_cache(ops);
@@ -551,13 +546,10 @@ static int fib_nl_dumprule(struct sk_buff *skb, struct netlink_callback
*cb)
  struct fib_rules_ops *ops;
  int idx = 0, family;

- if (net != &init_net)
-  return -EINVAL;
-
  family = rtnl_msg_family(cb->nlh);
  if (family != AF_UNSPEC) {
   /* Protocol specific dump request */
-  ops = lookup_rules_ops(family);
+  ops = lookup_rules_ops(net, family);
   if (ops == NULL)
    return -EAFNOSUPPORT;

@@ -565,7 +557,7 @@ static int fib_nl_dumprule(struct sk_buff *skb, struct netlink_callback *cb)
  }

  rcu_read_lock();
- list_for_each_entry_rcu(ops, &rules_ops, list) {
+ list_for_each_entry_rcu(ops, &net->rules_ops, list) {
   if (idx < cb->args[0] || !try_module_get(ops->owner))
    goto skip;
```

```
@@ -582,7 +574,7 @@ static int fib_nl_dumprule(struct sk_buff *skb, struct netlink_callback *cb)
  return skb->len;
 }

-static void notify_rule_change(int event, struct fib_rule *rule,
+static void notify_rule_change(struct net *net, int event, struct fib_rule *rule,
           struct fib_rules_ops *ops, struct nlmsghdr *nlh,
           u32 pid)
 {
@@ -600,10 +592,10 @@ static void notify_rule_change(int event, struct fib_rule *rule,
   kfree_skb(skb);
   goto errout;
  }
- err = rtnl_notify(skb, &init_net, pid, ops->nlgroup, nlh, GFP_KERNEL);
+ err = rtnl_notify(skb, net, pid, ops->nlgroup, nlh, GFP_KERNEL);
 errout:
  if (err < 0)
-  rtnl_set_sk_err(&init_net, ops->nlgroup, err);
+  rtnl_set_sk_err(net, ops->nlgroup, err);
 }

 static void attach_rules(struct list_head *rules, struct net_device *dev)
@@ -631,22 +623,20 @@ static int fib_rules_event(struct notifier_block *this, unsigned long
event,
        void *ptr)
 {
  struct net_device *dev = ptr;
+ struct net *net = dev->nd_net;
  struct fib_rules_ops *ops;

- if (dev->nd_net != &init_net)
-  return NOTIFY_DONE;
-
  ASSERT_RTNL();
  rcu_read_lock();

  switch (event) {
  case NETDEV_REGISTER:
-  list_for_each_entry(ops, &rules_ops, list)
+  list_for_each_entry(ops, &net->rules_ops, list)
    attach_rules(&ops->rules_list, dev);
   break;

  case NETDEV_UNREGISTER:
-  list_for_each_entry(ops, &rules_ops, list)
+  list_for_each_entry(ops, &net->rules_ops, list)
    detach_rules(&ops->rules_list, dev);
   break;
```

```
 }
@@ -660,13 +650,40 @@ static struct notifier_block fib_rules_notifier = {
  .notifier_call = fib_rules_event,
 };

+static int fib_rules_net_init(struct net *net)
+{
+ INIT_LIST_HEAD(&net->rules_ops);
+ spin_lock_init(&net->rules_mod_lock);
+ return 0;
+}
+
+static struct pernet_operations fib_rules_net_ops = {
+ .init = fib_rules_net_init,
+};
+
 static int __init fib_rules_init(void)
 {
+ int err;
  rtnl_register(PF_UNSPEC, RTM_NEWRULE, fib_nl_newrule, NULL);
  rtnl_register(PF_UNSPEC, RTM_DELRULE, fib_nl_delrule, NULL);
  rtnl_register(PF_UNSPEC, RTM_GETRULE, NULL, fib_nl_dumprule);

- return register_netdevice_notifier(&fib_rules_notifier);
+ err = register_netdevice_notifier(&fib_rules_notifier);
+ if (err < 0)
+  goto fail;
+
+ err = register_pernet_subsys(&fib_rules_net_ops);
+ if (err < 0)
+  goto fail_unregister;
+ return 0;
+
+fail_unregister:
+ unregister_netdevice_notifier(&fib_rules_notifier);
+fail:
+ rtnl_unregister(PF_UNSPEC, RTM_NEWRULE);
+ rtnl_unregister(PF_UNSPEC, RTM_DELRULE);
+ rtnl_unregister(PF_UNSPEC, RTM_GETRULE);
+ return err;
 }

 subsys_initcall(fib_rules_init);
--
1.5.3.rc5
```