Subject: [PATCH] small cleanup for fib rules
Posted by den on Fri, 14 Sep 2007 11:06:47 GMT
View Forum Message <> Reply to Message

From: Denis V. Lunev <den@openvz.org>

This patch slightly cleanups FIB rules framework. rules_list as a pointer
on struct fib_rules_ops is useless. It is always assigned with a static
per/subsystem list in IPv4, IPv6 and DecNet.

Signed-off-by: Denis V. Lunev <den@openvz.org>
Acked-by: Alexey Kuznetsov <kuznet@ms2.inr.ac.ru>

---
 include/net/fib_rules.h |    2 +-
 net/core/fib_rules.c    |   22 ++++++++++++----------
 net/decnet/dn_rules.c   |   13 ++++++-------
 net/ipv4/fib_rules.c    |   16 +++++++---------
 net/ipv6/fib6_rules.c   |    8 +++-----
 5 files changed, 28 insertions(+), 33 deletions(-)

diff --git a/include/net/fib_rules.h b/include/net/fib_rules.h
index 83e41dd..017aebd 100644
--- a/include/net/fib_rules.h
+++ b/include/net/fib_rules.h
@@ -65,7 +65,7 @@ struct fib_rules_ops

  int   nlgroup;
  const struct nla_policy *policy;
- struct list_head *rules_list;
+ struct list_head rules_list;
  struct module  *owner;
 };

diff --git a/net/core/fib_rules.c b/net/core/fib_rules.c
index 8c5474e..051cca1 100644
--- a/net/core/fib_rules.c
+++ b/net/core/fib_rules.c
@@ -82,7 +82,7 @@ static void cleanup_ops(struct fib_rules_ops *ops)
 {
  struct fib_rule *rule, *tmp;

- list_for_each_entry_safe(rule, tmp, ops->rules_list, list) {
+ list_for_each_entry_safe(rule, tmp, &ops->rules_list, list) {
   list_del_rcu(&rule->list);
   fib_rule_put(rule);
  }
@@ -137,7 +137,7 @@ int fib_rules_lookup(struct fib_rules_ops *ops, struct flowi *fl,

```
  rcu_read_lock();

- list_for_each_entry_rcu(rule, ops->rules_list, list) {
+ list_for_each_entry_rcu(rule, &ops->rules_list, list) {
 jumped:
   if (!fib_rule_match(rule, ops, fl, flags))
    continue;
@@ -268,7 +268,7 @@ static int fib_nl_newrule(struct sk_buff *skb, struct nlmsghdr* nlh, void
*arg)
   if (rule->target <= rule->pref)
    goto errout_free;

-  list_for_each_entry(r, ops->rules_list, list) {
+  list_for_each_entry(r, &ops->rules_list, list) {
    if (r->pref == rule->target) {
     rule->ctarget = r;
     break;
@@ -284,7 +284,7 @@ static int fib_nl_newrule(struct sk_buff *skb, struct nlmsghdr* nlh, void
*arg)
  if (err < 0)
   goto errout_free;

- list_for_each_entry(r, ops->rules_list, list) {
+ list_for_each_entry(r, &ops->rules_list, list) {
   if (r->pref > rule->pref)
    break;
   last = r;
@@ -297,7 +297,7 @@ static int fib_nl_newrule(struct sk_buff *skb, struct nlmsghdr* nlh, void
*arg)
    * There are unresolved goto rules in the list, check if
    * any of them are pointing to this new rule.
    */
-  list_for_each_entry(r, ops->rules_list, list) {
+  list_for_each_entry(r, &ops->rules_list, list) {
    if (r->action == FR_ACT_GOTO &&
        r->target == rule->pref) {
     BUG_ON(r->ctarget != NULL);
@@ -317,7 +317,7 @@ static int fib_nl_newrule(struct sk_buff *skb, struct nlmsghdr* nlh, void
*arg)
  if (last)
   list_add_rcu(&rule->list, &last->list);
  else
-  list_add_rcu(&rule->list, ops->rules_list);
+  list_add_rcu(&rule->list, &ops->rules_list);

  notify_rule_change(RTM_NEWRULE, rule, ops, nlh, NETLINK_CB(skb).pid);
  flush_route_cache(ops);
```

```
@@ -356,7 +356,7 @@ static int fib_nl_delrule(struct sk_buff *skb, struct nlmsghdr* nlh, void
*arg)
  if (err < 0)
   goto errout;

- list_for_each_entry(rule, ops->rules_list, list) {
+ list_for_each_entry(rule, &ops->rules_list, list) {
   if (frh->action && (frh->action != rule->action))
    continue;

@@ -399,7 +399,7 @@ static int fib_nl_delrule(struct sk_buff *skb, struct nlmsghdr* nlh, void
*arg)
    * actually been added.
    */
   if (ops->nr_goto_rules > 0) {
-   list_for_each_entry(tmp, ops->rules_list, list) {
+   list_for_each_entry(tmp, &ops->rules_list, list) {
     if (tmp->ctarget == rule) {
      rcu_assign_pointer(tmp->ctarget, NULL);
      ops->unresolved_rules++;
@@ -495,7 +495,7 @@ static int dump_rules(struct sk_buff *skb, struct netlink_callback *cb,
  int idx = 0;
  struct fib_rule *rule;

- list_for_each_entry(rule, ops->rules_list, list) {
+ list_for_each_entry(rule, &ops->rules_list, list) {
   if (idx < cb->args[1])
    goto skip;

@@ -602,12 +602,12 @@ static int fib_rules_event(struct notifier_block *this, unsigned long
event,
  switch (event) {
  case NETDEV_REGISTER:
   list_for_each_entry(ops, &rules_ops, list)
-   attach_rules(ops->rules_list, dev);
+   attach_rules(&ops->rules_list, dev);
   break;

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

diff --git a/net/decnet/dn_rules.c b/net/decnet/dn_rules.c
index 84ff3dd..4188d4a 100644
--- a/net/decnet/dn_rules.c
```

```
+++ b/net/decnet/dn_rules.c
@@ -57,8 +56,6 @@ static struct dn_fib_rule default_rule = {
  },
 };

-static LIST_HEAD(dn_fib_rules);
-

 int dn_fib_lookup(struct flowi *flp, struct dn_fib_res *res)
 {
@@ -228,9 +225,9 @@ static u32 dn_fib_rule_default_pref(void)
  struct list_head *pos;
  struct fib_rule *rule;

- if (!list_empty(&dn_fib_rules)) {
-  pos = dn_fib_rules.next;
-  if (pos->next != &dn_fib_rules) {
+ if (!list_empty(&dn_fib_rules_ops.rules_list)) {
+  pos = dn_fib_rules_ops.rules_list.next;
+  if (pos->next != &dn_fib_rules_ops.rules_list) {
     rule = list_entry(pos->next, struct fib_rule, list);
     if (rule->pref)
      return rule->pref - 1;
@@ -258,13 +255,14 @@ static struct fib_rules_ops dn_fib_rules_ops = {
  .flush_cache = dn_fib_rule_flush_cache,
  .nlgroup = RTNLGRP_DECnet_RULE,
  .policy  = dn_fib_rule_policy,
- .rules_list = &dn_fib_rules,
+ .rules_list = LIST_HEAD_INIT(dn_fib_rules_ops.rules_list),
  .owner  = THIS_MODULE,
 };

 void __init dn_fib_rules_init(void)
 {
- list_add_tail(&default_rule.common.list, &dn_fib_rules);
+ list_add_tail(&default_rule.common.list,
+    &dn_fib_rules_ops.rules_list);
  fib_rules_register(&dn_fib_rules_ops);
 }

diff --git a/net/ipv4/fib_rules.c b/net/ipv4/fib_rules.c
index 2a94784..f16839c 100644
--- a/net/ipv4/fib_rules.c
+++ b/net/ipv4/fib_rules.c
@@ -76,8 +76,6 @@ static struct fib4_rule local_rule = {
  },
 };
```

```
-static LIST_HEAD(fib4_rules);
-
 #ifdef CONFIG_NET_CLS_ROUTE
 u32 fib_rules_tclass(struct fib_result *res)
 {
@@ -279,9 +277,9 @@ static u32 fib4_rule_default_pref(void)
  struct list_head *pos;
  struct fib_rule *rule;

- if (!list_empty(&fib4_rules)) {
-  pos = fib4_rules.next;
-  if (pos->next != &fib4_rules) {
+ if (!list_empty(&fib4_rules_ops.rules_list)) {
+  pos = fib4_rules_ops.rules_list.next;
+  if (pos->next != &fib4_rules_ops.rules_list) {
    rule = list_entry(pos->next, struct fib_rule, list);
    if (rule->pref)
     return rule->pref - 1;
@@ -317,15 +315,15 @@ static struct fib_rules_ops fib4_rules_ops = {
  .flush_cache = fib4_rule_flush_cache,
  .nlgroup = RTNLGRP_IPV4_RULE,
  .policy  = fib4_rule_policy,
- .rules_list = &fib4_rules,
+ .rules_list = LIST_HEAD_INIT(fib4_rules_ops.rules_list),
  .owner  = THIS_MODULE,
 };

 void __init fib4_rules_init(void)
 {
- list_add_tail(&local_rule.common.list, &fib4_rules);
- list_add_tail(&main_rule.common.list, &fib4_rules);
- list_add_tail(&default_rule.common.list, &fib4_rules);
+ list_add_tail(&local_rule.common.list, &fib4_rules_ops.rules_list);
+ list_add_tail(&main_rule.common.list, &fib4_rules_ops.rules_list);
+ list_add_tail(&default_rule.common.list, &fib4_rules_ops.rules_list);

  fib_rules_register(&fib4_rules_ops);
 }
diff --git a/net/ipv6/fib6_rules.c b/net/ipv6/fib6_rules.c
index 53b3998..706622a 100644
--- a/net/ipv6/fib6_rules.c
+++ b/net/ipv6/fib6_rules.c
@@ -50,8 +50,6 @@ static struct fib6_rule local_rule = {
  },
 };

-static LIST_HEAD(fib6_rules);
-
```

```
 struct dst_entry *fib6_rule_lookup(struct flowi *fl, int flags,
        pol_lookup_t lookup)
 {
@@ -268,14 +266,14 @@ static struct fib_rules_ops fib6_rules_ops = {
   .nlmsg_payload  = fib6_rule_nlmsg_payload,
   .nlgroup  = RTNLGRP_IPV6_RULE,
   .policy   = fib6_rule_policy,
- .rules_list  = &fib6_rules,
+ .rules_list  = LIST_HEAD_INIT(fib6_rules_ops.rules_list),
   .owner   = THIS_MODULE,
 };

 void __init fib6_rules_init(void)
 {
- list_add_tail(&local_rule.common.list, &fib6_rules);
- list_add_tail(&main_rule.common.list, &fib6_rules);
+ list_add_tail(&local_rule.common.list, &fib6_rules_ops.rules_list);
+ list_add_tail(&main_rule.common.list, &fib6_rules_ops.rules_list);

   fib_rules_register(&fib6_rules_ops);
 }
```

---

Subject: Re: [PATCH] small cleanup for fib rules
Posted by davem on Sun, 16 Sep 2007 22:46:01 GMT
View Forum Message <> Reply to Message

From: "Denis V. Lunev" <den@openvz.org>
Date: Fri, 14 Sep 2007 15:06:47 +0400

> From: Denis V. Lunev <den@openvz.org>
>
> This patch slightly cleanups FIB rules framework. rules_list as a pointer
> on struct fib_rules_ops is useless. It is always assigned with a static
> per/subsystem list in IPv4, IPv6 and DecNet.
>
> Signed-off-by: Denis V. Lunev <den@openvz.org>
> Acked-by: Alexey Kuznetsov <kuznet@ms2.inr.ac.ru>

I am convinced you namespace guys will eventually eliminate
every single global variable in the kernel :-)

Patch applied to net-2.6.24, thanks!

---