Subject: [PATCH netns-2.6.25 0/19] routing virtualization
Posted by den on Wed, 19 Dec 2007 15:20:36 GMT
View Forum Message <> Reply to Message

Hi, David!

This set adds namespace support for routing tables & rules manipulation
in the different namespaces. So, one could create a namespace and
setup IPv4 routing there how he wants.

After this patch user will have the ability to configure and
observe its own isolated set of routing rules/tables, but they
all will be unused. I.e. routing decisions inside the network
stack are still made based on the init_net's rules. The reason for doing
so is to have something self-consistent and not too huge :)

The sequence is the following:
- viritualize generic FIB rules operations
- change IPv4 FIB initialization sequence
- virtualize FIB tables access

After this, the 'ip' utility and the /proc interface will start
working correctly inside a namespace, while the 'route' utility
will not, because IP sockets currently cannot be created in non-init
namespace.

Regards,
Den

Subject: [PATCH net-2.6.25 15/19] [NETNS] Provide correct namespace for fibnl netlink socket.
Posted by den on Wed, 19 Dec 2007 15:23:10 GMT
View Forum Message <> Reply to Message

Acked-by: Benjamin Thery <benjamin.thery@bull.net>
Signed-off-by: Denis V. Lunev <den@openvz.org>
---
include/net/netns/ipv4.h | 2 ++
net/ipv4/fib_frontend.c | 26 ++++++++++++++++++++--------
2 files changed, 20 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 {
  struct ctl_table_header *forw_hdr;
@@ -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 2e150ae..abe9f43 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))) {
@@ -858,26 +858,36 @@ static void nl_fib_input(struct sk_buff *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);
- return fibnl != NULL ? 0 : -EAFNOSUPPORT;
+ 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)
--
1.5.3.rc5
```

Subject: [PATCH net-2.6.25 17/19] [NETNS] Pass namespace through ip_rt_ioctl.
Posted by den on Wed, 19 Dec 2007 15:23:19 GMT

View Forum Message <> Reply to Message

... up to rtentry_to_fib_config

Acked-by: Benjamin Thery <benjamin.thery@bull.net>
Signed-off-by: Denis V. Lunev <den@openvz.org>

```
 include/net/route.h     |    2 +-
 net/ipv4/af_inet.c      |    2 +-
 net/ipv4/fib_frontend.c |    8 ++++----
 net/ipv4/ipconfig.c     |    2 +-
 4 files changed, 7 insertions(+), 7 deletions(-)

diff --git a/include/net/route.h b/include/net/route.h
index b777000..5847e6f 100644
--- a/include/net/route.h
+++ b/include/net/route.h
@@ -120,7 +120,7 @@ extern void  ip_rt_send_redirect(struct sk_buff *skb);
 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 int  ip_rt_ioctl(struct net *, unsigned int cmd, void __user *arg);
 extern void  ip_rt_get_source(u8 *src, struct rtable *rt);
 extern int  ip_rt_dump(struct sk_buff *skb,  struct netlink_callback *cb);

diff --git a/net/ipv4/af_inet.c b/net/ipv4/af_inet.c
index 6fac905..f0968a1 100644
--- a/net/ipv4/af_inet.c
+++ b/net/ipv4/af_inet.c
@@ -793,7 +793,7 @@ int inet_ioctl(struct socket *sock, unsigned int cmd, unsigned long arg)
   case SIOCADDRT:
   case SIOCDELRT:
   case SIOCRTMSG:
-   err = ip_rt_ioctl(cmd, (void __user *)arg);
+   err = ip_rt_ioctl(sk->sk_net, cmd, (void __user *)arg);
    break;
   case SIOCDARP:
   case SIOCGARP:
diff --git a/net/ipv4/fib_frontend.c b/net/ipv4/fib_frontend.c
index cecb660..a5c47fc 100644
--- a/net/ipv4/fib_frontend.c
+++ b/net/ipv4/fib_frontend.c
@@ -437,7 +437,7 @@ static int rtentry_to_fib_config(struct net *net, int cmd, struct rtentry *rt,
  * Handle IP routing ioctl calls. These are used to manipulate the routing tables
  */

-int ip_rt_ioctl(unsigned int cmd, void __user *arg)
+int ip_rt_ioctl(struct net *net, unsigned int cmd, void __user *arg)
 {
  struct fib_config cfg;
  struct rtentry rt;
@@ -453,18 +453,18 @@ int ip_rt_ioctl(unsigned int cmd, void __user *arg)
    return -EFAULT;
```

```
   rtnl_lock();
-  err = rtentry_to_fib_config(&init_net, cmd, &rt, &cfg);
+  err = rtentry_to_fib_config(net, cmd, &rt, &cfg);
   if (err == 0) {
    struct fib_table *tb;

    if (cmd == SIOCDELRT) {
-    tb = fib_get_table(&init_net, cfg.fc_table);
+    tb = fib_get_table(net, cfg.fc_table);
     if (tb)
      err = tb->tb_delete(tb, &cfg);
     else
      err = -ESRCH;
    } else {
-    tb = fib_new_table(&init_net, cfg.fc_table);
+    tb = fib_new_table(net, cfg.fc_table);
     if (tb)
      err = tb->tb_insert(tb, &cfg);
     else
diff --git a/net/ipv4/ipconfig.c b/net/ipv4/ipconfig.c
index 7288adb..7a89247 100644
--- a/net/ipv4/ipconfig.c
+++ b/net/ipv4/ipconfig.c
@@ -302,7 +302,7 @@ static int __init ic_route_ioctl(unsigned int cmd, struct rtentry *arg)

  mm_segment_t oldfs = get_fs();
  set_fs(get_ds());
- res = ip_rt_ioctl(cmd, (void __user *) arg);
+ res = ip_rt_ioctl(&init_net, cmd, (void __user *) arg);
  set_fs(oldfs);
  return res;
 }
--
1.5.3.rc5
```

Subject: [PATCH net-2.6.25 13/19] [NETNS] Namespacing IPv4 fib rules.
Posted by den on Wed, 19 Dec 2007 15:23:23 GMT
View Forum Message <> Reply to Message

The final trick for rules: place fib4_rules_ops into struct net and modify
initialization path for this.

Acked-by: Benjamin Thery <benjamin.thery@bull.net>
Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 include/net/netns/ipv4.h | 5 +++++
 net/ipv4/fib_rules.c | 44 ++----------

```
 2 files changed, 29 insertions(+), 20 deletions(-)

diff --git a/include/net/netns/ipv4.h b/include/net/netns/ipv4.h
index e06d7cf..299f8c6 100644
--- a/include/net/netns/ipv4.h
+++ b/include/net/netns/ipv4.h
@@ -4,12 +4,17 @@

 #ifndef __NETNS_IPV4_H__
 #define __NETNS_IPV4_H__
+
 struct ctl_table_header;
 struct ipv4_devconf;
+struct fib_rules_ops;

 struct netns_ipv4 {
  struct ctl_table_header *forw_hdr;
  struct ipv4_devconf *devconf_all;
  struct ipv4_devconf *devconf_dflt;
+#ifdef CONFIG_IP_MULTIPLE_TABLES
+ struct fib_rules_ops *rules_ops;
+#endif
 };
 #endif
diff --git a/net/ipv4/fib_rules.c b/net/ipv4/fib_rules.c
index 49819fe..72232ab 100644
--- a/net/ipv4/fib_rules.c
+++ b/net/ipv4/fib_rules.c
@@ -32,8 +32,6 @@
 #include <net/ip_fib.h>
 #include <net/fib_rules.h>

-static struct fib_rules_ops fib4_rules_ops;
-
 struct fib4_rule
 {
  struct fib_rule  common;
@@ -63,7 +61,7 @@ int fib_lookup(struct flowi *flp, struct fib_result *res)
  };
  int err;

- err = fib_rules_lookup(&fib4_rules_ops, flp, 0, &arg);
+ err = fib_rules_lookup(init_net.ipv4.rules_ops, flp, 0, &arg);
  res->r = arg.rule;

  return err;
@@ -149,6 +147,7 @@ static int fib4_rule_configure(struct fib_rule *rule, struct sk_buff *skb,
          struct nlmsghdr *nlh, struct fib_rule_hdr *frh,
```

```
          struct nlattr **tb)
 {
+ struct net *net = skb->sk->sk_net;
  int err = -EINVAL;
  struct fib4_rule *rule4 = (struct fib4_rule *) rule;

@@ -159,7 +158,7 @@ static int fib4_rule_configure(struct fib_rule *rule, struct sk_buff *skb,
   if (rule->action == FR_ACT_TO_TBL) {
    struct fib_table *table;

-   table = fib_empty_table(&init_net);
+   table = fib_empty_table(net);
    if (table == NULL) {
     err = -ENOBUFS;
     goto errout;
@@ -250,9 +249,9 @@ static u32 fib4_rule_default_pref(struct fib_rules_ops *ops)
  struct list_head *pos;
  struct fib_rule *rule;

- if (!list_empty(&fib4_rules_ops.rules_list)) {
-  pos = fib4_rules_ops.rules_list.next;
-  if (pos->next != &fib4_rules_ops.rules_list) {
+ if (!list_empty(&ops->rules_list)) {
+  pos = ops->rules_list.next;
+  if (pos->next != &ops->rules_list) {
    rule = list_entry(pos->next, struct fib_rule, list);
    if (rule->pref)
     return rule->pref - 1;
@@ -274,7 +273,7 @@ static void fib4_rule_flush_cache(void)
  rt_cache_flush(-1);
 }

-static struct fib_rules_ops fib4_rules_ops = {
+static struct fib_rules_ops fib4_rules_ops_template = {
  .family  = AF_INET,
  .rule_size = sizeof(struct fib4_rule),
  .addr_size = sizeof(u32),
@@ -288,24 +287,20 @@ static struct fib_rules_ops fib4_rules_ops = {
  .flush_cache = fib4_rule_flush_cache,
  .nlgroup = RTNLGRP_IPV4_RULE,
  .policy  = fib4_rule_policy,
- .rules_list = LIST_HEAD_INIT(fib4_rules_ops.rules_list),
  .owner  = THIS_MODULE,
 };

-static int __init fib_default_rules_init(void)
+static int fib_default_rules_init(struct fib_rules_ops *ops)
 {
```

```
  int err;

- err = fib_default_rule_add(&fib4_rules_ops, 0,
-       RT_TABLE_LOCAL, FIB_RULE_PERMANENT);
+ err = fib_default_rule_add(ops, 0, RT_TABLE_LOCAL, FIB_RULE_PERMANENT);
  if (err < 0)
   return err;
- err = fib_default_rule_add(&fib4_rules_ops, 0x7FFE,
-       RT_TABLE_MAIN, 0);
+ err = fib_default_rule_add(ops, 0x7FFE, RT_TABLE_MAIN, 0);
  if (err < 0)
   return err;
- err = fib_default_rule_add(&fib4_rules_ops, 0x7FFF,
-       RT_TABLE_DEFAULT, 0);
+ err = fib_default_rule_add(ops, 0x7FFF, RT_TABLE_DEFAULT, 0);
  if (err < 0)
   return err;
  return 0;
@@ -314,20 +309,29 @@ static int __init fib_default_rules_init(void)
 int __net_init fib4_rules_init(struct net *net)
 {
  int err;
+ struct fib_rules_ops *ops;
+
+ ops = kmemdup(&fib4_rules_ops_template, sizeof(*ops), GFP_KERNEL);
+ if (ops == NULL)
+  return -ENOMEM;
+ INIT_LIST_HEAD(&ops->rules_list);
+ fib_rules_register(net, ops);

- fib_rules_register(net, &fib4_rules_ops);
- err = fib_default_rules_init();
+ err = fib_default_rules_init(ops);
  if (err < 0)
   goto fail;
+ net->ipv4.rules_ops = ops;
  return 0;

 fail:
  /* also cleans all rules already added */
- fib_rules_unregister(net, &fib4_rules_ops);
+ fib_rules_unregister(net, ops);
+ kfree(ops);
  return err;
 }

 void __net_exit fib4_rules_exit(struct net *net)
 {
```

```
- fib_rules_unregister(net, &fib4_rules_ops);
+ fib_rules_unregister(net, net->ipv4.rules_ops);
+ kfree(net->ipv4.rules_ops);
 }
--
1.5.3.rc5
```

Subject: [PATCH net-2.6.25 7/19] [IPV4] Unify access to the routing tables.
Posted by den on Wed, 19 Dec 2007 15:24:31 GMT
View Forum Message <> Reply to Message

Replace the direct pointers to local and main tables with
calls to fib_get_table() with appropriate argument.

This doesn't introduce additional dereferences, but makes the access to fib
tables uniform in any (CONFIG_IP_MULTIPLE_TABLES) case.

Acked-by: Benjamin Thery <benjamin.thery@bull.net>
Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 include/net/ip_fib.h | 39 ++++++++++++++++++++++++++++++++++++++-----------
 net/ipv4/fib_frontend.c | 29 ++++++++++++++++++++++++++---------
 2 files changed, 40 insertions(+), 28 deletions(-)

```
diff --git a/include/net/ip_fib.h b/include/net/ip_fib.h
index 338d3ed..06e6a33 100644
--- a/include/net/ip_fib.h
+++ b/include/net/ip_fib.h
@@ -120,16 +120,22 @@ struct fib_result_nl {
  int             err;
 };

+extern struct hlist_head fib_table_hash[];
+
 #ifdef CONFIG_IP_ROUTE_MULTIPATH

 #define FIB_RES_NH(res)  ((res).fi->fib_nh[(res).nh_sel])
 #define FIB_RES_RESET(res) ((res).nh_sel = 0)

+#define FIB_TABLE_HASHSZ 2
+
 #else /* CONFIG_IP_ROUTE_MULTIPATH */

 #define FIB_RES_NH(res)  ((res).fi->fib_nh[0])
 #define FIB_RES_RESET(res)

+#define FIB_TABLE_HASHSZ 256
```

```
+
 #endif /* CONFIG_IP_ROUTE_MULTIPATH */

 #define FIB_RES_PREFSRC(res)  ((res).fi->fib_prefsrc ? : __fib_res_prefsrc(&res))
@@ -156,14 +162,17 @@ struct fib_table {

 #ifndef CONFIG_IP_MULTIPLE_TABLES

-extern struct fib_table *ip_fib_local_table;
-extern struct fib_table *ip_fib_main_table;
+#define TABLE_LOCAL_INDEX 0
+#define TABLE_MAIN_INDEX 1

 static inline struct fib_table *fib_get_table(u32 id)
 {
- if (id != RT_TABLE_LOCAL)
-  return ip_fib_main_table;
- return ip_fib_local_table;
+ struct hlist_head *ptr;
+
+ ptr = id == RT_TABLE_LOCAL ?
+  &fib_table_hash[TABLE_LOCAL_INDEX] :
+  &fib_table_hash[TABLE_MAIN_INDEX];
+ return hlist_entry(ptr->first, struct fib_table, tb_hlist);
 }

 static inline struct fib_table *fib_new_table(u32 id)
@@ -173,16 +182,24 @@ static inline struct fib_table *fib_new_table(u32 id)

 static inline int fib_lookup(const struct flowi *flp, struct fib_result *res)
 {
- if (ip_fib_local_table->tb_lookup(ip_fib_local_table, flp, res) &&
-     ip_fib_main_table->tb_lookup(ip_fib_main_table, flp, res))
-  return -ENETUNREACH;
- return 0;
+ struct fib_table *table;
+
+ table = fib_get_table(RT_TABLE_LOCAL);
+ if (!table->tb_lookup(table, flp, res))
+  return 0;
+
+ table = fib_get_table(RT_TABLE_MAIN);
+ if (!table->tb_lookup(table, flp, res))
+  return 0;
+ return -ENETUNREACH;
 }

-static inline void fib_select_default(const struct flowi *flp, struct fib_result *res)
```

```
+static inline void fib_select_default(const struct flowi *flp,
+          struct fib_result *res)
 {
+ struct fib_table *table = fib_get_table(RT_TABLE_MAIN);
  if (FIB_RES_GW(*res) && FIB_RES_NH(*res).nh_scope == RT_SCOPE_LINK)
-  ip_fib_main_table->tb_select_default(ip_fib_main_table, flp, res);
+  table->tb_select_default(table, flp, res);
 }

 #else /* CONFIG_IP_MULTIPLE_TABLES */
diff --git a/net/ipv4/fib_frontend.c b/net/ipv4/fib_frontend.c
index 714fff9..df9716f 100644
--- a/net/ipv4/fib_frontend.c
+++ b/net/ipv4/fib_frontend.c
@@ -50,39 +50,34 @@
 #define FFprint(a...) printk(KERN_DEBUG a)

 static struct sock *fibnl;
+struct hlist_head fib_table_hash[FIB_TABLE_HASHSZ];

 #ifndef CONFIG_IP_MULTIPLE_TABLES

-struct fib_table *ip_fib_local_table;
-struct fib_table *ip_fib_main_table;
-
-#define FIB_TABLE_HASHSZ 1
-static struct hlist_head fib_table_hash[FIB_TABLE_HASHSZ];
-
 static int __net_init fib4_rules_init(struct net *net)
 {
- ip_fib_local_table = fib_hash_init(RT_TABLE_LOCAL);
- if (ip_fib_local_table == NULL)
+ struct fib_table *local_table, *main_table;
+
+ local_table = fib_hash_init(RT_TABLE_LOCAL);
+ if (local_table == NULL)
   return -ENOMEM;

- ip_fib_main_table  = fib_hash_init(RT_TABLE_MAIN);
- if (ip_fib_main_table == NULL)
+ main_table  = fib_hash_init(RT_TABLE_MAIN);
+ if (main_table == NULL)
   goto fail;

- hlist_add_head_rcu(&ip_fib_local_table->tb_hlist, &fib_table_hash[0]);
- hlist_add_head_rcu(&ip_fib_main_table->tb_hlist, &fib_table_hash[0]);
+ hlist_add_head_rcu(&local_table->tb_hlist,
+    &fib_table_hash[TABLE_LOCAL_INDEX]);
```

```
+ hlist_add_head_rcu(&main_table->tb_hlist,
+     &fib_table_hash[TABLE_MAIN_INDEX]);
  return 0;

 fail:
- kfree(ip_fib_local_table);
- ip_fib_local_table = NULL;
+ kfree(local_table);
  return -ENOMEM;
 }
 #else

-#define FIB_TABLE_HASHSZ 256
-static struct hlist_head fib_table_hash[FIB_TABLE_HASHSZ];
-
 struct fib_table *fib_new_table(u32 id)
 {
  struct fib_table *tb;
--
1.5.3.rc5
```

Subject: [PATCH net-2.6.25 1/19] [NETNS] Add netns parameter to fib_rules_(un)register.

Posted by den on Wed, 19 Dec 2007 15:24:31 GMT

View Forum Message <> Reply to Message

This provides a ground for further fib rules operations virtualization.
Currently pass the init_net into them.

Acked-by: Benjamin Thery <benjamin.thery@bull.net>
Signed-off-by: Denis V. Lunev <den@openvz.org>

```
---
 include/net/fib_rules.h |   16 +++++++---------
 net/core/fib_rules.c    |    4 ++--
 net/decnet/dn_rules.c   |    4 ++--
 net/ipv4/fib_rules.c    |    2 +-
 net/ipv6/fib6_rules.c   |    4 ++--
 5 files changed, 14 insertions(+), 16 deletions(-)

diff --git a/include/net/fib_rules.h b/include/net/fib_rules.h
index 2364db1..af62345 100644
--- a/include/net/fib_rules.h
+++ b/include/net/fib_rules.h
@@ -101,14 +101,12 @@ static inline u32 frh_get_table(struct fib_rule_hdr *frh, struct nlattr
**nla)
  return frh->table;
 }
```

```
-extern int   fib_rules_register(struct fib_rules_ops *);
-extern int   fib_rules_unregister(struct fib_rules_ops *);
-extern void                  fib_rules_cleanup_ops(struct fib_rules_ops *);
+extern int fib_rules_register(struct net *, struct fib_rules_ops *);
+extern int fib_rules_unregister(struct net *, struct fib_rules_ops *);
+extern void fib_rules_cleanup_ops(struct fib_rules_ops *);

-extern int   fib_rules_lookup(struct fib_rules_ops *,
-       struct flowi *, int flags,
-       struct fib_lookup_arg *);
-extern int   fib_default_rule_add(struct fib_rules_ops *,
-           u32 pref, u32 table,
-           u32 flags);
+extern int fib_rules_lookup(struct fib_rules_ops *, struct flowi *, int flags,
+       struct fib_lookup_arg *);
+extern int fib_default_rule_add(struct fib_rules_ops *, u32 pref, u32 table,
+    u32 flags);
 #endif
diff --git a/net/core/fib_rules.c b/net/core/fib_rules.c
index fcbf41c..ada9c81 100644
--- a/net/core/fib_rules.c
+++ b/net/core/fib_rules.c
@@ -74,7 +74,7 @@ static void flush_route_cache(struct fib_rules_ops *ops)
   ops->flush_cache();
 }

-int fib_rules_register(struct fib_rules_ops *ops)
+int fib_rules_register(struct net *net, struct fib_rules_ops *ops)
 {
  int err = -EEXIST;
  struct fib_rules_ops *o;
@@ -113,7 +113,7 @@ void fib_rules_cleanup_ops(struct fib_rules_ops *ops)
 }
 EXPORT_SYMBOL_GPL(fib_rules_cleanup_ops);

-int fib_rules_unregister(struct fib_rules_ops *ops)
+int fib_rules_unregister(struct net *net, struct fib_rules_ops *ops)
 {
  int err = 0;
  struct fib_rules_ops *o;
diff --git a/net/decnet/dn_rules.c b/net/decnet/dn_rules.c
index ffebea0..0b5e2b9 100644
--- a/net/decnet/dn_rules.c
+++ b/net/decnet/dn_rules.c
@@ -255,12 +255,12 @@ void __init dn_fib_rules_init(void)
 {
  BUG_ON(fib_default_rule_add(&dn_fib_rules_ops, 0x7fff,
```

```
                RT_TABLE_MAIN, 0));
- fib_rules_register(&dn_fib_rules_ops);
+ fib_rules_register(&init_net, &dn_fib_rules_ops);
 }

 void __exit dn_fib_rules_cleanup(void)
 {
- fib_rules_unregister(&dn_fib_rules_ops);
+ fib_rules_unregister(&init_net, &dn_fib_rules_ops);
 }


diff --git a/net/ipv4/fib_rules.c b/net/ipv4/fib_rules.c
index a0ada3a..eac3f71 100644
--- a/net/ipv4/fib_rules.c
+++ b/net/ipv4/fib_rules.c
@@ -314,5 +314,5 @@ static int __init fib_default_rules_init(void)
 void __init fib4_rules_init(void)
 {
   BUG_ON(fib_default_rules_init());
- fib_rules_register(&fib4_rules_ops);
+ fib_rules_register(&init_net, &fib4_rules_ops);
 }
diff --git a/net/ipv6/fib6_rules.c b/net/ipv6/fib6_rules.c
index 9ce2e0a..e4d7e5a 100644
--- a/net/ipv6/fib6_rules.c
+++ b/net/ipv6/fib6_rules.c
@@ -273,7 +273,7 @@ int __init fib6_rules_init(void)
   if (ret)
     goto out;

- ret = fib_rules_register(&fib6_rules_ops);
+ ret = fib_rules_register(&init_net, &fib6_rules_ops);
   if (ret)
     goto out_default_rules_init;
 out:
@@ -286,5 +286,5 @@ out_default_rules_init:

 void fib6_rules_cleanup(void)
 {
- fib_rules_unregister(&fib6_rules_ops);
+ fib_rules_unregister(&init_net, &fib6_rules_ops);
 }
--
1.5.3.rc5
```

Subject: [PATCH net-2.6.25 8/19] [NETNS] Add netns parameter to fib_get_table/fib_new_table.
Posted by den on Wed, 19 Dec 2007 15:24:32 GMT
View Forum Message <> Reply to Message

This is a central entry point to the fib storage after the previous patch.

Acked-by: Benjamin Thery <benjamin.thery@bull.net>
Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 include/net/ip_fib.h | 16 ++++++++--------
 net/ipv4/fib_frontend.c | 24 ++++++++++++++----------
 net/ipv4/fib_hash.c | 4 ++--
 net/ipv4/fib_rules.c | 12 ++++++------
 net/ipv4/fib_trie.c | 8 ++++----
 5 files changed, 32 insertions(+), 32 deletions(-)

diff --git a/include/net/ip_fib.h b/include/net/ip_fib.h
index 06e6a33..1faad1f 100644
--- a/include/net/ip_fib.h
+++ b/include/net/ip_fib.h
@@ -165,7 +165,7 @@ struct fib_table {
 #define TABLE_LOCAL_INDEX 0
 #define TABLE_MAIN_INDEX 1

-static inline struct fib_table *fib_get_table(u32 id)
+static inline struct fib_table *fib_get_table(struct net *net, u32 id)
 {
  struct hlist_head *ptr;

@@ -175,20 +175,20 @@ static inline struct fib_table *fib_get_table(u32 id)
  return hlist_entry(ptr->first, struct fib_table, tb_hlist);
 }

-static inline struct fib_table *fib_new_table(u32 id)
+static inline struct fib_table *fib_new_table(struct net *net, u32 id)
 {
- return fib_get_table(id);
+ return fib_get_table(net, id);
 }

 static inline int fib_lookup(const struct flowi *flp, struct fib_result *res)
 {
  struct fib_table *table;

- table = fib_get_table(RT_TABLE_LOCAL);
+ table = fib_get_table(&init_net, RT_TABLE_LOCAL);
  if (!table->tb_lookup(table, flp, res))
   return 0;

```
- table = fib_get_table(RT_TABLE_MAIN);
+ table = fib_get_table(&init_net, RT_TABLE_MAIN);
  if (!table->tb_lookup(table, flp, res))
   return 0;
  return -ENETUNREACH;
@@ -197,7 +197,7 @@ static inline int fib_lookup(const struct flowi *flp, struct fib_result *res)
 static inline void fib_select_default(const struct flowi *flp,
          struct fib_result *res)
 {
- struct fib_table *table = fib_get_table(RT_TABLE_MAIN);
+ struct fib_table *table = fib_get_table(&init_net, RT_TABLE_MAIN);
  if (FIB_RES_GW(*res) && FIB_RES_NH(*res).nh_scope == RT_SCOPE_LINK)
   table->tb_select_default(table, flp, res);
 }
@@ -212,8 +212,8 @@ extern u32 fib_rules_tclass(struct fib_result *res);

 extern int fib_lookup(struct flowi *flp, struct fib_result *res);

-extern struct fib_table *fib_new_table(u32 id);
-extern struct fib_table *fib_get_table(u32 id);
+extern struct fib_table *fib_new_table(struct net *net, u32 id);
+extern struct fib_table *fib_get_table(struct net *net, u32 id);
 extern void fib_select_default(const struct flowi *flp, struct fib_result *res);

 #endif /* CONFIG_IP_MULTIPLE_TABLES */
diff --git a/net/ipv4/fib_frontend.c b/net/ipv4/fib_frontend.c
index df9716f..2c20908 100644
--- a/net/ipv4/fib_frontend.c
+++ b/net/ipv4/fib_frontend.c
@@ -78,14 +78,14 @@ fail:
 }
 #else

-struct fib_table *fib_new_table(u32 id)
+struct fib_table *fib_new_table(struct net *net, u32 id)
 {
  struct fib_table *tb;
  unsigned int h;

  if (id == 0)
   id = RT_TABLE_MAIN;
- tb = fib_get_table(id);
+ tb = fib_get_table(net, id);
  if (tb)
   return tb;
  tb = fib_hash_init(id);
@@ -96,7 +96,7 @@ struct fib_table *fib_new_table(u32 id)
```

```
  return tb;
 }

-struct fib_table *fib_get_table(u32 id)
+struct fib_table *fib_get_table(struct net *net, u32 id)
 {
  struct fib_table *tb;
  struct hlist_node *node;
@@ -148,7 +148,7 @@ struct net_device * ip_dev_find(__be32 addr)
  res.r = NULL;
 #endif

- local_table = fib_get_table(RT_TABLE_LOCAL);
+ local_table = fib_get_table(&init_net, RT_TABLE_LOCAL);
  if (!local_table || local_table->tb_lookup(local_table, &fl, &res))
   return NULL;
  if (res.type != RTN_LOCAL)
@@ -183,7 +183,7 @@ static inline unsigned __inet_dev_addr_type(const struct net_device
*dev,
  res.r = NULL;
 #endif

- local_table = fib_get_table(RT_TABLE_LOCAL);
+ local_table = fib_get_table(&init_net, RT_TABLE_LOCAL);
  if (local_table) {
   ret = RTN_UNICAST;
   if (!local_table->tb_lookup(local_table, &fl, &res)) {
@@ -453,13 +453,13 @@ int ip_rt_ioctl(unsigned int cmd, void __user *arg)
    struct fib_table *tb;

    if (cmd == SIOCDELRT) {
-    tb = fib_get_table(cfg.fc_table);
+    tb = fib_get_table(&init_net, cfg.fc_table);
     if (tb)
      err = tb->tb_delete(tb, &cfg);
     else
      err = -ESRCH;
    } else {
-    tb = fib_new_table(cfg.fc_table);
+    tb = fib_new_table(&init_net, cfg.fc_table);
     if (tb)
      err = tb->tb_insert(tb, &cfg);
     else
@@ -573,7 +573,7 @@ static int inet_rtm_delroute(struct sk_buff *skb, struct nlmsghdr* nlh, void
*ar
  if (err < 0)
   goto errout;
```

```
- tb = fib_get_table(cfg.fc_table);
+ tb = fib_get_table(net, cfg.fc_table);
  if (tb == NULL) {
   err = -ESRCH;
   goto errout;
@@ -598,7 +598,7 @@ static int inet_rtm_newroute(struct sk_buff *skb, struct nlmsghdr* nlh,
void *ar
  if (err < 0)
   goto errout;

- tb = fib_new_table(cfg.fc_table);
+ tb = fib_new_table(&init_net, cfg.fc_table);
  if (tb == NULL) {
   err = -ENOBUFS;
   goto errout;
@@ -671,9 +671,9 @@ static void fib_magic(int cmd, int type, __be32 dst, int dst_len, struct
in_ifad
  };

  if (type == RTN_UNICAST)
-  tb = fib_new_table(RT_TABLE_MAIN);
+  tb = fib_new_table(&init_net, RT_TABLE_MAIN);
  else
-  tb = fib_new_table(RT_TABLE_LOCAL);
+  tb = fib_new_table(&init_net, RT_TABLE_LOCAL);

  if (tb == NULL)
   return;
@@ -845,7 +845,7 @@ static void nl_fib_input(struct sk_buff *skb)
  }

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

  nl_fib_lookup(frn, tb);

diff --git a/net/ipv4/fib_hash.c b/net/ipv4/fib_hash.c
index 11277f6..75c1bde 100644
--- a/net/ipv4/fib_hash.c
+++ b/net/ipv4/fib_hash.c
@@ -795,7 +795,7 @@ struct fib_iter_state {
 static struct fib_alias *fib_get_first(struct seq_file *seq)
 {
  struct fib_iter_state *iter = seq->private;
- struct fib_table *main_table = fib_get_table(RT_TABLE_MAIN);
+ struct fib_table *main_table = fib_get_table(&init_net, RT_TABLE_MAIN);
  struct fn_hash *table = (struct fn_hash *)main_table->tb_data;
```

```
  iter->bucket    = 0;
@@ -935,7 +935,7 @@ static void *fib_seq_start(struct seq_file *seq, loff_t *pos)
  void *v = NULL;

  read_lock(&fib_hash_lock);
- if (fib_get_table(RT_TABLE_MAIN))
+ if (fib_get_table(&init_net, RT_TABLE_MAIN))
   v = *pos ? fib_get_idx(seq, *pos - 1) : SEQ_START_TOKEN;
  return v;
 }
diff --git a/net/ipv4/fib_rules.c b/net/ipv4/fib_rules.c
index 1aae61c..49819fe 100644
--- a/net/ipv4/fib_rules.c
+++ b/net/ipv4/fib_rules.c
@@ -93,7 +93,7 @@ static int fib4_rule_action(struct fib_rule *rule, struct flowi *flp,
   goto errout;
  }

- if ((tbl = fib_get_table(rule->table)) == NULL)
+ if ((tbl = fib_get_table(&init_net, rule->table)) == NULL)
   goto errout;

  err = tbl->tb_lookup(tbl, flp, (struct fib_result *) arg->result);
@@ -109,7 +109,7 @@ void fib_select_default(const struct flowi *flp, struct fib_result *res)
  if (res->r && res->r->action == FR_ACT_TO_TBL &&
      FIB_RES_GW(*res) && FIB_RES_NH(*res).nh_scope == RT_SCOPE_LINK) {
   struct fib_table *tb;
-  if ((tb = fib_get_table(res->r->table)) != NULL)
+  if ((tb = fib_get_table(&init_net, res->r->table)) != NULL)
    tb->tb_select_default(tb, flp, res);
  }
 }
@@ -130,13 +130,13 @@ static int fib4_rule_match(struct fib_rule *rule, struct flowi *fl, int flags)
  return 1;
 }

-static struct fib_table *fib_empty_table(void)
+static struct fib_table *fib_empty_table(struct net *net)
 {
  u32 id;

  for (id = 1; id <= RT_TABLE_MAX; id++)
-  if (fib_get_table(id) == NULL)
-   return fib_new_table(id);
+  if (fib_get_table(net, id) == NULL)
+   return fib_new_table(net, id);
  return NULL;
```

```
 }

@@ -159,7 +159,7 @@ static int fib4_rule_configure(struct fib_rule *rule, struct sk_buff *skb,
   if (rule->action == FR_ACT_TO_TBL) {
    struct fib_table *table;

-   table = fib_empty_table();
+   table = fib_empty_table(&init_net);
    if (table == NULL) {
     err = -ENOBUFS;
     goto errout;
diff --git a/net/ipv4/fib_trie.c b/net/ipv4/fib_trie.c
index 43b6e94..9526758 100644
--- a/net/ipv4/fib_trie.c
+++ b/net/ipv4/fib_trie.c
@@ -2164,12 +2164,12 @@ static int fib_triestat_seq_show(struct seq_file *seq, void *v)
  struct fib_table *tb;

  trie_local = NULL;
- tb = fib_get_table(RT_TABLE_LOCAL);
+ tb = fib_get_table(&init_net, RT_TABLE_LOCAL);
  if (tb)
   trie_local = (struct trie *) tb->tb_data;

  trie_main = NULL;
- tb = fib_get_table(RT_TABLE_MAIN);
+ tb = fib_get_table(&init_net, RT_TABLE_MAIN);
  if (tb)
   trie_main = (struct trie *) tb->tb_data;

@@ -2236,12 +2236,12 @@ static void *fib_trie_seq_start(struct seq_file *seq, loff_t *pos)
  struct fib_table *tb;

  if (!iter->trie_local) {
-  tb = fib_get_table(RT_TABLE_LOCAL);
+  tb = fib_get_table(&init_net, RT_TABLE_LOCAL);
   if (tb)
    iter->trie_local = (struct trie *) tb->tb_data;
  }
  if (!iter->trie_main) {
-  tb = fib_get_table(RT_TABLE_MAIN);
+  tb = fib_get_table(&init_net, RT_TABLE_MAIN);
   if (tb)
    iter->trie_main = (struct trie *) tb->tb_data;
  }
--
1.5.3.rc5
```

Subject: [PATCH net-2.6.25 2/19] [NETNS] Pass fib_rules_ops into default_pref method.
Posted by den on Wed, 19 Dec 2007 15:24:32 GMT
View Forum Message <> Reply to Message

fib_rules_ops contains operations and the list of configured rules. ops will
become per/namespace soon, so we need them to be known in the default_pref
callback.

Acked-by: Benjamin Thery <benjamin.thery@bull.net>
Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 include/net/fib_rules.h | 2 +-
 net/core/fib_rules.c | 2 +-
 net/decnet/dn_rules.c | 2 +-
 net/ipv4/fib_rules.c | 2 +-
 net/ipv6/fib6_rules.c | 2 +-
 5 files changed, 5 insertions(+), 5 deletions(-)

```
diff --git a/include/net/fib_rules.h b/include/net/fib_rules.h
index af62345..14b8a71 100644
--- a/include/net/fib_rules.h
+++ b/include/net/fib_rules.h
@@ -56,7 +56,7 @@ struct fib_rules_ops
   int   (*fill)(struct fib_rule *, struct sk_buff *,
       struct nlmsghdr *,
       struct fib_rule_hdr *);
- u32   (*default_pref)(void);
+ u32   (*default_pref)(struct fib_rules_ops *ops);
   size_t   (*nlmsg_payload)(struct fib_rule *);

   /* Called after modifications to the rules set, must flush
diff --git a/net/core/fib_rules.c b/net/core/fib_rules.c
index ada9c81..e12e9f5 100644
--- a/net/core/fib_rules.c
+++ b/net/core/fib_rules.c
@@ -285,7 +285,7 @@ static int fib_nl_newrule(struct sk_buff *skb, struct nlmsghdr* nlh, void
*arg)
   rule->table = frh_get_table(frh, tb);

   if (!rule->pref && ops->default_pref)
-  rule->pref = ops->default_pref();
+  rule->pref = ops->default_pref(ops);

   err = -EINVAL;
   if (tb[FRA_GOTO]) {
diff --git a/net/decnet/dn_rules.c b/net/decnet/dn_rules.c
index 0b5e2b9..c1fae23 100644
--- a/net/decnet/dn_rules.c
```

```
+++ b/net/decnet/dn_rules.c
@@ -212,7 +212,7 @@ nla_put_failure:
  return -ENOBUFS;
 }

-static u32 dn_fib_rule_default_pref(void)
+static u32 dn_fib_rule_default_pref(struct fib_rules_ops *ops)
 {
  struct list_head *pos;
  struct fib_rule *rule;
diff --git a/net/ipv4/fib_rules.c b/net/ipv4/fib_rules.c
index eac3f71..afe669d 100644
--- a/net/ipv4/fib_rules.c
+++ b/net/ipv4/fib_rules.c
@@ -245,7 +245,7 @@ nla_put_failure:
  return -ENOBUFS;
 }

-static u32 fib4_rule_default_pref(void)
+static u32 fib4_rule_default_pref(struct fib_rules_ops *ops)
 {
  struct list_head *pos;
  struct fib_rule *rule;
diff --git a/net/ipv6/fib6_rules.c b/net/ipv6/fib6_rules.c
index e4d7e5a..76437a1 100644
--- a/net/ipv6/fib6_rules.c
+++ b/net/ipv6/fib6_rules.c
@@ -223,7 +223,7 @@ nla_put_failure:
  return -ENOBUFS;
 }

-static u32 fib6_rule_default_pref(void)
+static u32 fib6_rule_default_pref(struct fib_rules_ops *ops)
 {
  return 0x3FFF;
 }
--
1.5.3.rc5
```

Subject: [PATCH net-2.6.25 3/19] [NETNS] Namespacing in the generic fib rules code.

Posted by den on Wed, 19 Dec 2007 15:24:33 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>
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 = {
 };

+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
```

Subject: [PATCH net-2.6.25 4/19] [NETNS] Add namespace to API for routing /proc entries creation.

Posted by den on Wed, 19 Dec 2007 15:24:34 GMT

View Forum Message <> Reply to Message

This adds netns parameter to fib_proc_init/exit and replaces __init specifier with __net_init. After this, we will not yet have these proc files show info from the specific namespace - this will be done when these tables become namespaced.

```
Acked-by: Benjamin Thery <benjamin.thery@bull.net>
Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 include/net/ip_fib.h |    4 ++--
 net/ipv4/af_inet.c   |    4 ++--
 net/ipv4/fib_hash.c  |    8 ++++----
 net/ipv4/fib_trie.c  |   21 ++++++++++++----------
 4 files changed, 19 insertions(+), 18 deletions(-)

diff --git a/include/net/ip_fib.h b/include/net/ip_fib.h
index d70b9b4..f74cbb2 100644
--- a/include/net/ip_fib.h
+++ b/include/net/ip_fib.h
@@ -253,8 +253,8 @@ static inline void fib_res_put(struct fib_result *res)
 }

 #ifdef CONFIG_PROC_FS
-extern int  fib_proc_init(void);
-extern void fib_proc_exit(void);
+extern int __net_init  fib_proc_init(struct net *net);
+extern void __net_exit fib_proc_exit(struct net *net);
 #endif

 #endif  /* _NET_FIB_H */
diff --git a/net/ipv4/af_inet.c b/net/ipv4/af_inet.c
index 03633b7..ea9fd3d 100644
--- a/net/ipv4/af_inet.c
+++ b/net/ipv4/af_inet.c
@@ -1470,14 +1470,14 @@ static int __init ipv4_proc_init(void)
   goto out_tcp;
  if (udp4_proc_init())
   goto out_udp;
- if (fib_proc_init())
+ if (fib_proc_init(&init_net))
   goto out_fib;
  if (ip_misc_proc_init())
   goto out_misc;
 out:
  return rc;
 out_misc:
- fib_proc_exit();
+ fib_proc_exit(&init_net);
 out_fib:
  udp4_proc_exit();
 out_udp:
diff --git a/net/ipv4/fib_hash.c b/net/ipv4/fib_hash.c
index 481de47..f5476e2 100644
--- a/net/ipv4/fib_hash.c
```

```
+++ b/net/ipv4/fib_hash.c
@@ -1038,15 +1038,15 @@ static const struct file_operations fib_seq_fops = {
  .release = seq_release_private,
 };

-int __init fib_proc_init(void)
+int __net_init fib_proc_init(struct net *net)
 {
- if (!proc_net_fops_create(&init_net, "route", S_IRUGO, &fib_seq_fops))
+ if (!proc_net_fops_create(net, "route", S_IRUGO, &fib_seq_fops))
   return -ENOMEM;
  return 0;
 }

-void __init fib_proc_exit(void)
+void __net_exit fib_proc_exit(struct net *net)
 {
- proc_net_remove(&init_net, "route");
+ proc_net_remove(net, "route");
 }
 #endif /* CONFIG_PROC_FS */
diff --git a/net/ipv4/fib_trie.c b/net/ipv4/fib_trie.c
index 46c3b01..3c7f668 100644
--- a/net/ipv4/fib_trie.c
+++ b/net/ipv4/fib_trie.c
@@ -2505,32 +2505,33 @@ static const struct file_operations fib_route_fops = {
  .release = seq_release_private,
 };

-int __init fib_proc_init(void)
+int __net_init fib_proc_init(struct net *net)
 {
- if (!proc_net_fops_create(&init_net, "fib_trie", S_IRUGO, &fib_trie_fops))
+ if (!proc_net_fops_create(net, "fib_trie", S_IRUGO, &fib_trie_fops))
   goto out1;

- if (!proc_net_fops_create(&init_net, "fib_triestat", S_IRUGO, &fib_triestat_fops))
+ if (!proc_net_fops_create(net, "fib_triestat", S_IRUGO,
+       &fib_triestat_fops))
   goto out2;

- if (!proc_net_fops_create(&init_net, "route", S_IRUGO, &fib_route_fops))
+ if (!proc_net_fops_create(net, "route", S_IRUGO, &fib_route_fops))
   goto out3;

  return 0;

 out3:
```

```
- proc_net_remove(&init_net, "fib_triestat");
+ proc_net_remove(net, "fib_triestat");
 out2:
- proc_net_remove(&init_net, "fib_trie");
+ proc_net_remove(net, "fib_trie");
 out1:
  return -ENOMEM;
 }

-void __init fib_proc_exit(void)
+void __net_exit fib_proc_exit(struct net *net)
 {
- proc_net_remove(&init_net, "fib_trie");
- proc_net_remove(&init_net, "fib_triestat");
- proc_net_remove(&init_net, "route");
+ proc_net_remove(net, "fib_trie");
+ proc_net_remove(net, "fib_triestat");
+ proc_net_remove(net, "route");
 }

 #endif /* CONFIG_PROC_FS */
--
1.5.3.rc5
```

Subject: [PATCH net-2.6.25 5/19] [IPV4] Check fib4_rules_init failure.
Posted by den on Wed, 19 Dec 2007 15:24:35 GMT
View Forum Message <> Reply to Message

This adds error paths into both versions of fib4_rules_init (with/without
CONFIG_IP_MULTIPLE_TABLES) and returns error code to the caller.

Acked-by: Benjamin Thery <benjamin.thery@bull.net>
Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 include/net/ip_fib.h | 2 +-
 net/ipv4/fib_frontend.c | 18 ++++++++++++++---
 net/ipv4/fib_rules.c | 14 ++++++++++--
 3 files changed, 28 insertions(+), 6 deletions(-)

```
diff --git a/include/net/ip_fib.h b/include/net/ip_fib.h
index f74cbb2..cbff18d 100644
--- a/include/net/ip_fib.h
+++ b/include/net/ip_fib.h
@@ -186,7 +186,7 @@ static inline void fib_select_default(const struct flowi *flp, struct fib_result
 }

 #else /* CONFIG_IP_MULTIPLE_TABLES */
```

```
-extern void __init fib4_rules_init(void);
+extern int __init fib4_rules_init(void);

 #ifdef CONFIG_NET_CLS_ROUTE
 extern u32 fib_rules_tclass(struct fib_result *res);
diff --git a/net/ipv4/fib_frontend.c b/net/ipv4/fib_frontend.c
index b03c4c6..82d0e00 100644
--- a/net/ipv4/fib_frontend.c
+++ b/net/ipv4/fib_frontend.c
@@ -59,12 +59,24 @@ struct fib_table *ip_fib_main_table;
 #define FIB_TABLE_HASHSZ 1
 static struct hlist_head fib_table_hash[FIB_TABLE_HASHSZ];

-static void __init fib4_rules_init(void)
+static int __init fib4_rules_init(void)
 {
  ip_fib_local_table = fib_hash_init(RT_TABLE_LOCAL);
- hlist_add_head_rcu(&ip_fib_local_table->tb_hlist, &fib_table_hash[0]);
+ if (ip_fib_local_table == NULL)
+  return -ENOMEM;
+
  ip_fib_main_table  = fib_hash_init(RT_TABLE_MAIN);
+ if (ip_fib_main_table == NULL)
+  goto fail;
+
+ hlist_add_head_rcu(&ip_fib_local_table->tb_hlist, &fib_table_hash[0]);
  hlist_add_head_rcu(&ip_fib_main_table->tb_hlist, &fib_table_hash[0]);
+ return 0;
+
+fail:
+ kfree(ip_fib_local_table);
+ ip_fib_local_table = NULL;
+ return -ENOMEM;
 }
 #else

@@ -941,7 +953,7 @@ void __init ip_fib_init(void)
  for (i = 0; i < FIB_TABLE_HASHSZ; i++)
   INIT_HLIST_HEAD(&fib_table_hash[i]);

- fib4_rules_init();
+ BUG_ON(fib4_rules_init());

  register_netdevice_notifier(&fib_netdev_notifier);
  register_inetaddr_notifier(&fib_inetaddr_notifier);
diff --git a/net/ipv4/fib_rules.c b/net/ipv4/fib_rules.c
index afe669d..0751734 100644
--- a/net/ipv4/fib_rules.c
```

```
+++ b/net/ipv4/fib_rules.c
@@ -311,8 +311,18 @@ static int __init fib_default_rules_init(void)
  return 0;
 }

-void __init fib4_rules_init(void)
+int __init fib4_rules_init()
 {
- BUG_ON(fib_default_rules_init());
+ int err;
+
  fib_rules_register(&init_net, &fib4_rules_ops);
+ err = fib_default_rules_init();
+ if (err < 0)
+  goto fail;
+ return 0;
+
+fail:
+ /* also cleans all rules already added */
+ fib_rules_unregister(&init_net, &fib4_rules_ops);
+ return err;
 }
--
1.5.3.rc5
```

Subject: [PATCH net-2.6.25 6/19] [NETNS] Refactor fib initialization so it can handle multiple namespaces.

Posted by den on Wed, 19 Dec 2007 15:24:36 GMT

View Forum Message <> Reply to Message

Currently the code is only called in the initial namespace, but
this is not so as soon as the code being called can handle it.

So collect the calls to all needed fib initialization functions
in one place and register it as a pernet subsys.

Acked-by: Benjamin Thery <benjamin.thery@bull.net>
Signed-off-by: Denis V. Lunev <den@openvz.org>

```
---
 include/net/ip_fib.h    |    3 +-
 net/ipv4/af_inet.c      |    4 --
 net/ipv4/fib_frontend.c |   86 ++++++++++++++++++++++++++++++++++++++++++++----
 net/ipv4/fib_hash.c     |    6 +---
 net/ipv4/fib_rules.c    |   11 ++++--
 net/ipv4/fib_trie.c     |    6 +---
 6 files changed, 90 insertions(+), 26 deletions(-)
```

```
diff --git a/include/net/ip_fib.h b/include/net/ip_fib.h
index cbff18d..338d3ed 100644
--- a/include/net/ip_fib.h
+++ b/include/net/ip_fib.h
@@ -186,7 +186,8 @@ static inline void fib_select_default(const struct flowi *flp, struct fib_result
 }

 #else /* CONFIG_IP_MULTIPLE_TABLES */
-extern int __init fib4_rules_init(void);
+extern int __net_init fib4_rules_init(struct net *net);
+extern void __net_exit fib4_rules_exit(struct net *net);

 #ifdef CONFIG_NET_CLS_ROUTE
 extern u32 fib_rules_tclass(struct fib_result *res);
diff --git a/net/ipv4/af_inet.c b/net/ipv4/af_inet.c
index ea9fd3d..19b557a 100644
--- a/net/ipv4/af_inet.c
+++ b/net/ipv4/af_inet.c
@@ -1470,15 +1470,11 @@ static int __init ipv4_proc_init(void)
   goto out_tcp;
  if (udp4_proc_init())
   goto out_udp;
- if (fib_proc_init(&init_net))
-  goto out_fib;
  if (ip_misc_proc_init())
   goto out_misc;
 out:
  return rc;
 out_misc:
- fib_proc_exit(&init_net);
-out_fib:
  udp4_proc_exit();
 out_udp:
  tcp4_proc_exit();
diff --git a/net/ipv4/fib_frontend.c b/net/ipv4/fib_frontend.c
index 82d0e00..714fff9 100644
--- a/net/ipv4/fib_frontend.c
+++ b/net/ipv4/fib_frontend.c
@@ -59,7 +59,7 @@ struct fib_table *ip_fib_main_table;
 #define FIB_TABLE_HASHSZ 1
 static struct hlist_head fib_table_hash[FIB_TABLE_HASHSZ];

-static int __init fib4_rules_init(void)
+static int __net_init fib4_rules_init(struct net *net)
 {
  ip_fib_local_table = fib_hash_init(RT_TABLE_LOCAL);
  if (ip_fib_local_table == NULL)
@@ -860,10 +860,16 @@ static void nl_fib_input(struct sk_buff *skb)
```

```
  netlink_unicast(fibnl, skb, pid, MSG_DONTWAIT);
 }

-static void nl_fib_lookup_init(void)
+static int nl_fib_lookup_init(struct net *net)
 {
- fibnl = netlink_kernel_create(&init_net, NETLINK_FIB_LOOKUP, 0,
+ fibnl = netlink_kernel_create(net, NETLINK_FIB_LOOKUP, 0,
           nl_fib_input, NULL, THIS_MODULE);
+ return fibnl != NULL ? 0 : -EAFNOSUPPORT;
+}
+
+static void nl_fib_lookup_exit(struct net *net)
+{
+ sock_put(fibnl);
 }

 static void fib_disable_ip(struct net_device *dev, int force)
@@ -946,22 +952,86 @@ static struct notifier_block fib_netdev_notifier = {
  .notifier_call =fib_netdev_event,
 };

-void __init ip_fib_init(void)
+static int __net_init ip_fib_net_init(struct net *net)
 {
  unsigned int i;

  for (i = 0; i < FIB_TABLE_HASHSZ; i++)
   INIT_HLIST_HEAD(&fib_table_hash[i]);

- BUG_ON(fib4_rules_init());
+ return fib4_rules_init(net);
+}
+
+static void __net_exit ip_fib_net_exit(struct net *net)
+{
+ unsigned int i;

- register_netdevice_notifier(&fib_netdev_notifier);
- register_inetaddr_notifier(&fib_inetaddr_notifier);
- nl_fib_lookup_init();
+#ifdef CONFIG_IP_MULTIPLE_TABLES
+ fib4_rules_exit(net);
+#endif
+
+ for (i = 0; i < FIB_TABLE_HASHSZ; i++) {
+  struct fib_table *tb;
+  struct hlist_head *head;
```

```
+  struct hlist_node *node, *tmp;
+
+  head = &fib_table_hash[i];
+  hlist_for_each_entry_safe(tb, node, tmp, head, tb_hlist) {
+   hlist_del(node);
+   tb->tb_flush(tb);
+   kfree(tb);
+  }
+ }
+}
+
+static int __net_init fib_net_init(struct net *net)
+{
+ int error;

+ error = 0;
+ if (net != &init_net)
+  goto out;
+
+ error = ip_fib_net_init(net);
+ if (error < 0)
+  goto out;
+ error = nl_fib_lookup_init(net);
+ if (error < 0)
+  goto out_nlfl;
+ error = fib_proc_init(net);
+ if (error < 0)
+  goto out_proc;
+out:
+ return error;
+
+out_proc:
+ nl_fib_lookup_exit(net);
+out_nlfl:
+ ip_fib_net_exit(net);
+ goto out;
+}
+
+static void __net_exit fib_net_exit(struct net *net)
+{
+ fib_proc_exit(net);
+ nl_fib_lookup_exit(net);
+ ip_fib_net_exit(net);
+}
+
+static struct pernet_operations fib_net_ops = {
+ .init = fib_net_init,
+ .exit = fib_net_exit,
```

```
+};
+
+void __init ip_fib_init(void)
+{
  rtnl_register(PF_INET, RTM_NEWROUTE, inet_rtm_newroute, NULL);
  rtnl_register(PF_INET, RTM_DELROUTE, inet_rtm_delroute, NULL);
  rtnl_register(PF_INET, RTM_GETROUTE, NULL, inet_dump_fib);
+
+ register_pernet_subsys(&fib_net_ops);
+ register_netdevice_notifier(&fib_netdev_notifier);
+ register_inetaddr_notifier(&fib_inetaddr_notifier);
 }

 EXPORT_SYMBOL(inet_addr_type);
diff --git a/net/ipv4/fib_hash.c b/net/ipv4/fib_hash.c
index f5476e2..11277f6 100644
--- a/net/ipv4/fib_hash.c
+++ b/net/ipv4/fib_hash.c
@@ -745,11 +745,7 @@ static int fn_hash_dump(struct fib_table *tb, struct sk_buff *skb, struct
netlin
  return skb->len;
 }

-#ifdef CONFIG_IP_MULTIPLE_TABLES
-struct fib_table * fib_hash_init(u32 id)
-#else
-struct fib_table * __init fib_hash_init(u32 id)
-#endif
+struct fib_table *fib_hash_init(u32 id)
 {
  struct fib_table *tb;

diff --git a/net/ipv4/fib_rules.c b/net/ipv4/fib_rules.c
index 0751734..1aae61c 100644
--- a/net/ipv4/fib_rules.c
+++ b/net/ipv4/fib_rules.c
@@ -311,11 +311,11 @@ static int __init fib_default_rules_init(void)
  return 0;
 }

-int __init fib4_rules_init()
+int __net_init fib4_rules_init(struct net *net)
 {
  int err;

- fib_rules_register(&init_net, &fib4_rules_ops);
+ fib_rules_register(net, &fib4_rules_ops);
  err = fib_default_rules_init();
```

```
  if (err < 0)
   goto fail;
@@ -323,6 +323,11 @@ int __init fib4_rules_init()

 fail:
  /* also cleans all rules already added */
- fib_rules_unregister(&init_net, &fib4_rules_ops);
+ fib_rules_unregister(net, &fib4_rules_ops);
  return err;
 }
+
+void __net_exit fib4_rules_exit(struct net *net)
+{
+ fib_rules_unregister(net, &fib4_rules_ops);
+}
diff --git a/net/ipv4/fib_trie.c b/net/ipv4/fib_trie.c
index 3c7f668..43b6e94 100644
--- a/net/ipv4/fib_trie.c
+++ b/net/ipv4/fib_trie.c
@@ -1953,11 +1953,7 @@ out:

 /* Fix more generic FIB names for init later */

-#ifdef CONFIG_IP_MULTIPLE_TABLES
-struct fib_table * fib_hash_init(u32 id)
-#else
-struct fib_table * __init fib_hash_init(u32 id)
-#endif
+struct fib_table *fib_hash_init(u32 id)
 {
  struct fib_table *tb;
  struct trie *t;
--
1.5.3.rc5
```

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 ff999ff..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 rtentry_to_fib_config(int cmd, struct rtentry *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_prefsrc) {
   if (cfg->fc_type != RTN_LOCAL || !cfg->fc_dst ||
       fi->fib_prefsrc != cfg->fc_dst)
-   if (inet_addr_type(fi->fib_prefsrc) != RTN_LOCAL)
+   if (inet_addr_type(&init_net, fi->fib_prefsrc) != 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 @@ __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 &&
--
1.5.3.rc5
```

Subject: [PATCH net-2.6.25 10/19] [NETNS] Add netns to nl_info structure.
Posted by den on Wed, 19 Dec 2007 15:24:40 GMT

View Forum Message <> Reply to Message

nl_info is used to track the end-user destination of routing change
notification. This is a natural object to hold a namespace on. Place it
there and utilize the context in the appropriate places.

Acked-by: Benjamin Thery <benjamin.thery@bull.net>
Signed-off-by: Denis V. Lunev <den@openvz.org>

```
---
 include/net/netlink.h    |    1 +
 net/ipv4/fib_frontend.c  |    5 +++++
 net/ipv4/fib_semantics.c |   16 ++++++++++------
 net/ipv6/ip6_fib.c       |    4 +++-
 net/ipv6/route.c         |    8 ++++++--
 5 files changed, 25 insertions(+), 9 deletions(-)

diff --git a/include/net/netlink.h b/include/net/netlink.h
index db4b935..87cf0bc 100644
--- a/include/net/netlink.h
+++ b/include/net/netlink.h
@@ -217,6 +217,7 @@ struct nla_policy {
  */
 struct nl_info {
  struct nlmsghdr  *nlh;
+ struct net  *nl_net;
  u32    pid;
 };

diff --git a/net/ipv4/fib_frontend.c b/net/ipv4/fib_frontend.c
index 312a728..08414eb 100644
--- a/net/ipv4/fib_frontend.c
+++ b/net/ipv4/fib_frontend.c
@@ -310,6 +310,7 @@ static int rtentry_to_fib_config(int cmd, struct rtentry *rt,
  int plen;

  memset(cfg, 0, sizeof(*cfg));
+ cfg->fc_nlinfo.nl_net = &init_net;

  if (rt->rt_dst.sa_family != AF_INET)
   return -EAFNOSUPPORT;
@@ -516,6 +517,7 @@ static int rtm_to_fib_config(struct sk_buff *skb, struct nlmsghdr *nlh,

  cfg->fc_nlinfo.pid = NETLINK_CB(skb).pid;
  cfg->fc_nlinfo.nlh = nlh;
+ cfg->fc_nlinfo.nl_net = &init_net;

  if (cfg->fc_type > RTN_MAX) {
   err = -EINVAL;
@@ -670,6 +672,9 @@ static void fib_magic(int cmd, int type, __be32 dst, int dst_len, struct
in_ifad
   .fc_prefsrc = ifa->ifa_local,
   .fc_oif = ifa->ifa_dev->dev->ifindex,
   .fc_nlflags = NLM_F_CREATE | NLM_F_APPEND,
+  .fc_nlinfo = {
+   .nl_net = &init_net,
+  },
```

```
 };

  if (type == RTN_UNICAST)
diff --git a/net/ipv4/fib_semantics.c b/net/ipv4/fib_semantics.c
index c1263e2..0de6102 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, &init_net, info->pid, RTNLGRP_IPV4_ROUTE,
+ err = rtnl_notify(skb, info->nl_net, info->pid, RTNLGRP_IPV4_ROUTE,
       info->nlh, GFP_KERNEL);
 errout:
  if (err < 0)
-  rtnl_set_sk_err(&init_net, RTNLGRP_IPV4_ROUTE, err);
+  rtnl_set_sk_err(info->nl_net, RTNLGRP_IPV4_ROUTE, err);
 }

 /* Return the first fib alias matching TOS with
@@ -531,9 +531,11 @@ 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(&init_net, nh->nh_gw) != RTN_UNICAST)
+   if (inet_addr_type(cfg->fc_nlinfo.nl_net,
+         nh->nh_gw) != RTN_UNICAST)
     return -EINVAL;
-   if ((dev = __dev_get_by_index(&init_net, nh->nh_oif)) == NULL)
+   if ((dev = __dev_get_by_index(cfg->fc_nlinfo.nl_net,
+           nh->nh_oif)) == NULL)
     return -ENODEV;
    if (!(dev->flags&IFF_UP))
     return -ENETDOWN;
@@ -795,7 +797,8 @@ struct fib_info *fib_create_info(struct fib_config *cfg)
   if (nhs != 1 || nh->nh_gw)
    goto err_inval;
   nh->nh_scope = RT_SCOPE_NOWHERE;
-  nh->nh_dev = dev_get_by_index(&init_net, fi->fib_nh->nh_oif);
+  nh->nh_dev = dev_get_by_index(cfg->fc_nlinfo.nl_net,
+           fi->fib_nh->nh_oif);
   err = -ENODEV;
   if (nh->nh_dev == NULL)
    goto failure;
@@ -809,7 +812,8 @@ struct fib_info *fib_create_info(struct fib_config *cfg)
  if (fi->fib_prefsrc) {
   if (cfg->fc_type != RTN_LOCAL || !cfg->fc_dst ||
```

```
       fi->fib_prefsrc != cfg->fc_dst)
-   if (inet_addr_type(&init_net, fi->fib_prefsrc) != RTN_LOCAL)
+   if (inet_addr_type(cfg->fc_nlinfo.nl_net,
+          fi->fib_prefsrc) != RTN_LOCAL)
      goto err_inval;
  }

diff --git a/net/ipv6/ip6_fib.c b/net/ipv6/ip6_fib.c
index df05c6f..242a8ca 100644
--- a/net/ipv6/ip6_fib.c
+++ b/net/ipv6/ip6_fib.c
@@ -1315,7 +1315,9 @@ static int fib6_walk(struct fib6_walker_t *w)

 static int fib6_clean_node(struct fib6_walker_t *w)
 {
- struct nl_info info = {};
+ struct nl_info info = {
+  .nl_net = &init_net,
+ };
  int res;
  struct rt6_info *rt;
  struct fib6_cleaner_t *c = container_of(w, struct fib6_cleaner_t, w);
diff --git a/net/ipv6/route.c b/net/ipv6/route.c
index 02354a7..7f03ae9 100644
--- a/net/ipv6/route.c
+++ b/net/ipv6/route.c
@@ -604,7 +604,9 @@ static int __ip6_ins_rt(struct rt6_info *rt, struct nl_info *info)

 int ip6_ins_rt(struct rt6_info *rt)
 {
- struct nl_info info = {};
+ struct nl_info info = {
+  .nl_net = &init_net,
+ };
  return __ip6_ins_rt(rt, &info);
 }

@@ -1262,7 +1264,9 @@ static int __ip6_del_rt(struct rt6_info *rt, struct nl_info *info)

 int ip6_del_rt(struct rt6_info *rt)
 {
- struct nl_info info = {};
+ struct nl_info info = {
+  .nl_net = &init_net,
+ };
  return __ip6_del_rt(rt, &info);
 }
```

--
1.5.3.rc5

Subject: [PATCH net-2.6.25 11/19] [NETNS] Show routing information from correct namespace (fib_hash.c)
Posted by den on Wed, 19 Dec 2007 15:24:41 GMT
View Forum Message <> Reply to Message

This is the second part (for the CONFIG_IP_FIB_HASH case) of the patch #4,
where we have created proc files in namespaces.

Now we can dump correct info in them.

Acked-by: Benjamin Thery <benjamin.thery@bull.net>
Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 net/ipv4/fib_hash.c | 17 ++++++++++------
 1 files changed, 11 insertions(+), 6 deletions(-)

```
diff --git a/net/ipv4/fib_hash.c b/net/ipv4/fib_hash.c
index 75c1bde..3bb3a0a 100644
--- a/net/ipv4/fib_hash.c
+++ b/net/ipv4/fib_hash.c
@@ -782,6 +782,7 @@ struct fib_table *fib_hash_init(u32 id)
 #ifdef CONFIG_PROC_FS

 struct fib_iter_state {
+ struct seq_net_private p;
  struct fn_zone *zone;
  int  bucket;
  struct hlist_head *hash_head;
@@ -795,8 +796,11 @@ struct fib_iter_state {
 static struct fib_alias *fib_get_first(struct seq_file *seq)
 {
  struct fib_iter_state *iter = seq->private;
- struct fib_table *main_table = fib_get_table(&init_net, RT_TABLE_MAIN);
- struct fn_hash *table = (struct fn_hash *)main_table->tb_data;
+ struct fib_table *main_table;
+ struct fn_hash *table;
+
+ main_table = fib_get_table(iter->p.net, RT_TABLE_MAIN);
+ table = (struct fn_hash *)main_table->tb_data;

  iter->bucket    = 0;
  iter->hash_head = NULL;
@@ -932,10 +936,11 @@ static struct fib_alias *fib_get_idx(struct seq_file *seq, loff_t pos)
```

```
 static void *fib_seq_start(struct seq_file *seq, loff_t *pos)
 {
+ struct fib_iter_state *iter = seq->private;
  void *v = NULL;

  read_lock(&fib_hash_lock);
- if (fib_get_table(&init_net, RT_TABLE_MAIN))
+ if (fib_get_table(iter->p.net, RT_TABLE_MAIN))
   v = *pos ? fib_get_idx(seq, *pos - 1) : SEQ_START_TOKEN;
  return v;
 }
@@ -1022,8 +1027,8 @@ static const struct seq_operations fib_seq_ops = {

 static int fib_seq_open(struct inode *inode, struct file *file)
 {
- return seq_open_private(file, &fib_seq_ops,
-   sizeof(struct fib_iter_state));
+ return seq_open_net(inode, file, &fib_seq_ops,
+       sizeof(struct fib_iter_state));
 }

 static const struct file_operations fib_seq_fops = {
@@ -1031,7 +1036,7 @@ static const struct file_operations fib_seq_fops = {
  .open           = fib_seq_open,
  .read           = seq_read,
  .llseek         = seq_lseek,
- .release = seq_release_private,
+ .release = seq_release_net,
 };

 int __net_init fib_proc_init(struct net *net)
--
1.5.3.rc5
```

Subject: [PATCH net-2.6.25 12/19] [NETNS] Show routing information from correct namespace (fib_trie.c)

Posted by den on Wed, 19 Dec 2007 15:24:42 GMT

View Forum Message <> Reply to Message

This is the second part (for the CONFIG_IP_FIB_TRIE case) of the patch #4,
where we have created proc files in namespaces.

Now we can dump correct info in them.

Acked-by: Benjamin Thery <benjamin.thery@bull.net>
Signed-off-by: Denis V. Lunev <den@openvz.org>

```
 net/ipv4/fib_trie.c |   44 ++++++++++++++++++++++++++++++++++++++------------
 1 files changed, 32 insertions(+), 12 deletions(-)

diff --git a/net/ipv4/fib_trie.c b/net/ipv4/fib_trie.c
index 9526758..708faa9 100644
--- a/net/ipv4/fib_trie.c
+++ b/net/ipv4/fib_trie.c
@@ -1992,6 +1992,7 @@ struct fib_table *fib_hash_init(u32 id)
 #ifdef CONFIG_PROC_FS
 /* Depth first Trie walk iterator */
 struct fib_trie_iter {
+ struct seq_net_private p;
  struct trie *trie_local, *trie_main;
  struct tnode *tnode;
  struct trie *trie;
@@ -2159,17 +2160,18 @@ static void trie_show_stats(struct seq_file *seq, struct trie_stat *stat)

 static int fib_triestat_seq_show(struct seq_file *seq, void *v)
 {
+ struct net *net = (struct net *)seq->private;
  struct trie *trie_local, *trie_main;
  struct trie_stat *stat;
  struct fib_table *tb;

  trie_local = NULL;
- tb = fib_get_table(&init_net, RT_TABLE_LOCAL);
+ tb = fib_get_table(net, RT_TABLE_LOCAL);
  if (tb)
   trie_local = (struct trie *) tb->tb_data;

  trie_main = NULL;
- tb = fib_get_table(&init_net, RT_TABLE_MAIN);
+ tb = fib_get_table(net, RT_TABLE_MAIN);
  if (tb)
   trie_main = (struct trie *) tb->tb_data;

@@ -2199,7 +2201,25 @@ static int fib_triestat_seq_show(struct seq_file *seq, void *v)

 static int fib_triestat_seq_open(struct inode *inode, struct file *file)
 {
- return single_open(file, fib_triestat_seq_show, NULL);
+ int err;
+ struct net *net;
+
+ net = get_proc_net(inode);
+ if (net == NULL)
+  return -ENXIO;
+ err = single_open(file, fib_triestat_seq_show, net);
```

```
+ if (err < 0) {
+  put_net(net);
+  return err;
+ }
+ return 0;
+}
+
+static int fib_triestat_seq_release(struct inode *ino, struct file *f)
+{
+ struct seq_file *seq = f->private_data;
+ put_net(seq->private);
+ return single_release(ino, f);
 }

 static const struct file_operations fib_triestat_fops = {
@@ -2207,7 +2227,7 @@ static const struct file_operations fib_triestat_fops = {
  .open = fib_triestat_seq_open,
  .read = seq_read,
  .llseek = seq_lseek,
- .release = single_release,
+ .release = fib_triestat_seq_release,
 };

 static struct node *fib_trie_get_idx(struct fib_trie_iter *iter,
@@ -2236,12 +2256,12 @@ static void *fib_trie_seq_start(struct seq_file *seq, loff_t *pos)
  struct fib_table *tb;

  if (!iter->trie_local) {
-  tb = fib_get_table(&init_net, RT_TABLE_LOCAL);
+  tb = fib_get_table(iter->p.net, RT_TABLE_LOCAL);
   if (tb)
    iter->trie_local = (struct trie *) tb->tb_data;
  }
  if (!iter->trie_main) {
-  tb = fib_get_table(&init_net, RT_TABLE_MAIN);
+  tb = fib_get_table(iter->p.net, RT_TABLE_MAIN);
   if (tb)
    iter->trie_main = (struct trie *) tb->tb_data;
  }
@@ -2385,8 +2405,8 @@ static const struct seq_operations fib_trie_seq_ops = {

 static int fib_trie_seq_open(struct inode *inode, struct file *file)
 {
- return seq_open_private(file, &fib_trie_seq_ops,
-  sizeof(struct fib_trie_iter));
+ return seq_open_net(inode, file, &fib_trie_seq_ops,
+       sizeof(struct fib_trie_iter));
 }
```

```
 static const struct file_operations fib_trie_fops = {
@@ -2394,7 +2414,7 @@ static const struct file_operations fib_trie_fops = {
  .open   = fib_trie_seq_open,
  .read   = seq_read,
  .llseek = seq_lseek,
- .release = seq_release_private,
+ .release = seq_release_net,
 };

 static unsigned fib_flag_trans(int type, __be32 mask, const struct fib_info *fi)
@@ -2489,8 +2509,8 @@ static const struct seq_operations fib_route_seq_ops = {

 static int fib_route_seq_open(struct inode *inode, struct file *file)
 {
- return seq_open_private(file, &fib_route_seq_ops,
-   sizeof(struct fib_trie_iter));
+ return seq_open_net(inode, file, &fib_route_seq_ops,
+       sizeof(struct fib_trie_iter));
 }

 static const struct file_operations fib_route_fops = {
@@ -2498,7 +2518,7 @@ static const struct file_operations fib_route_fops = {
  .open   = fib_route_seq_open,
  .read   = seq_read,
  .llseek = seq_lseek,
- .release = seq_release_private,
+ .release = seq_release_net,
 };

 int __net_init fib_proc_init(struct net *net)
--
1.5.3.rc5
```

Subject: [PATCH net-2.6.25 14/19] [NETNS] Place fib tables into netns.
Posted by den on Wed, 19 Dec 2007 15:24:44 GMT

View Forum Message <> Reply to Message

The preparatory work has been done. All we need is to substitute
fib_table_hash with net->ipv4.fib_table_hash. Netns context is available when
required.

Acked-by: Benjamin Thery <benjamin.thery@bull.net>
Signed-off-by: Denis V. Lunev <den@openvz.org>

```
 include/net/ip_fib.h     |    7 ++-----
 include/net/netns/ipv4.h |    2 ++
```

```
 net/ipv4/fib_frontend.c  |   36 ++++++++++++++++++++++++++----------
 3 files changed, 28 insertions(+), 17 deletions(-)

diff --git a/include/net/ip_fib.h b/include/net/ip_fib.h
index 1faad1f..ddb4c20 100644
--- a/include/net/ip_fib.h
+++ b/include/net/ip_fib.h
@@ -120,8 +120,6 @@ struct fib_result_nl {
  int             err;
 };

-extern struct hlist_head fib_table_hash[];
-
 #ifdef CONFIG_IP_ROUTE_MULTIPATH

 #define FIB_RES_NH(res)  ((res).fi->fib_nh[(res).nh_sel])
@@ -169,9 +167,8 @@ static inline struct fib_table *fib_get_table(struct net *net, u32 id)
 {
  struct hlist_head *ptr;

- ptr = id == RT_TABLE_LOCAL ?
-  &fib_table_hash[TABLE_LOCAL_INDEX] :
-  &fib_table_hash[TABLE_MAIN_INDEX];
+ ptr = net->ipv4.fib_table_hash;
+ ptr += id == RT_TABLE_LOCAL ? TABLE_LOCAL_INDEX : TABLE_MAIN_INDEX;
  return hlist_entry(ptr->first, struct fib_table, tb_hlist);
 }

diff --git a/include/net/netns/ipv4.h b/include/net/netns/ipv4.h
index 299f8c6..629ec6c 100644
--- a/include/net/netns/ipv4.h
+++ b/include/net/netns/ipv4.h
@@ -8,6 +8,7 @@
 struct ctl_table_header;
 struct ipv4_devconf;
 struct fib_rules_ops;
+struct hlist_head;

 struct netns_ipv4 {
  struct ctl_table_header *forw_hdr;
@@ -16,5 +17,6 @@ struct netns_ipv4 {
 #ifdef CONFIG_IP_MULTIPLE_TABLES
  struct fib_rules_ops *rules_ops;
 #endif
+ struct hlist_head *fib_table_hash;
 };
 #endif
diff --git a/net/ipv4/fib_frontend.c b/net/ipv4/fib_frontend.c
```

```
index 08414eb..2e150ae 100644
--- a/net/ipv4/fib_frontend.c
+++ b/net/ipv4/fib_frontend.c
@@ -50,7 +50,6 @@
 #define FFprint(a...) printk(KERN_DEBUG a)

 static struct sock *fibnl;
-struct hlist_head fib_table_hash[FIB_TABLE_HASHSZ];

 #ifndef CONFIG_IP_MULTIPLE_TABLES

@@ -67,9 +66,9 @@ static int __net_init fib4_rules_init(struct net *net)
   goto fail;

  hlist_add_head_rcu(&local_table->tb_hlist,
-    &fib_table_hash[TABLE_LOCAL_INDEX]);
+    &net->ipv4.fib_table_hash[TABLE_LOCAL_INDEX]);
  hlist_add_head_rcu(&main_table->tb_hlist,
-    &fib_table_hash[TABLE_MAIN_INDEX]);
+    &net->ipv4.fib_table_hash[TABLE_MAIN_INDEX]);
  return 0;

 fail:
@@ -92,7 +91,7 @@ struct fib_table *fib_new_table(struct net *net, u32 id)
  if (!tb)
   return NULL;
  h = id & (FIB_TABLE_HASHSZ - 1);
- hlist_add_head_rcu(&tb->tb_hlist, &fib_table_hash[h]);
+ hlist_add_head_rcu(&tb->tb_hlist, &net->ipv4.fib_table_hash[h]);
  return tb;
 }

@@ -100,13 +99,16 @@ struct fib_table *fib_get_table(struct net *net, u32 id)
 {
  struct fib_table *tb;
  struct hlist_node *node;
+ struct hlist_head *head;
  unsigned int h;

  if (id == 0)
   id = RT_TABLE_MAIN;
  h = id & (FIB_TABLE_HASHSZ - 1);
+
  rcu_read_lock();
- hlist_for_each_entry_rcu(tb, node, &fib_table_hash[h], tb_hlist) {
+ head = net->ipv4.fib_table_hash + h;
+ hlist_for_each_entry_rcu(tb, node, head, tb_hlist) {
   if (tb->tb_id == id) {
```

```
    rcu_read_unlock();
    return tb;
@@ -117,15 +119,17 @@ struct fib_table *fib_get_table(struct net *net, u32 id)
 }
 #endif /* CONFIG_IP_MULTIPLE_TABLES */

-static void fib_flush(void)
+static void fib_flush(struct net *net)
 {
  int flushed = 0;
  struct fib_table *tb;
  struct hlist_node *node;
+ struct hlist_head *head;
  unsigned int h;

  for (h = 0; h < FIB_TABLE_HASHSZ; h++) {
-  hlist_for_each_entry(tb, node, &fib_table_hash[h], tb_hlist)
+  head = net->ipv4.fib_table_hash + h;
+  hlist_for_each_entry(tb, node, head, tb_hlist)
     flushed += tb->tb_flush(tb);
  }

@@ -620,6 +624,7 @@ static int inet_dump_fib(struct sk_buff *skb, struct netlink_callback *cb)
  unsigned int e = 0, s_e;
  struct fib_table *tb;
  struct hlist_node *node;
+ struct hlist_head *head;
  int dumped = 0;

  if (net != &init_net)
@@ -634,7 +639,8 @@ static int inet_dump_fib(struct sk_buff *skb, struct netlink_callback *cb)

  for (h = s_h; h < FIB_TABLE_HASHSZ; h++, s_e = 0) {
   e = 0;
-  hlist_for_each_entry(tb, node, &fib_table_hash[h], tb_hlist) {
+  head = net->ipv4.fib_table_hash + h;
+  hlist_for_each_entry(tb, node, head, tb_hlist) {
     if (e < s_e)
      goto next;
     if (dumped)
@@ -797,7 +803,7 @@ static void fib_del_ifaddr(struct in_ifaddr *ifa)
       for stray nexthop entries, then ignite fib_flush.
     */
     if (fib_sync_down(ifa->ifa_local, NULL, 0))
-     fib_flush();
+     fib_flush(&init_net);
   }
  }
```

```
 #undef LOCAL_OK
@@ -877,7 +883,7 @@ static void nl_fib_lookup_exit(struct net *net)
 static void fib_disable_ip(struct net_device *dev, int force)
 {
  if (fib_sync_down(0, dev, force))
-  fib_flush();
+  fib_flush(&init_net);
  rt_cache_flush(0);
  arp_ifdown(dev);
 }
@@ -958,8 +964,13 @@ static int __net_init ip_fib_net_init(struct net *net)
 {
  unsigned int i;

+ net->ipv4.fib_table_hash = kzalloc(
+   sizeof(struct hlist_head)*FIB_TABLE_HASHSZ, GFP_KERNEL);
+ if (net->ipv4.fib_table_hash == NULL)
+  return -ENOMEM;
+
  for (i = 0; i < FIB_TABLE_HASHSZ; i++)
-  INIT_HLIST_HEAD(&fib_table_hash[i]);
+  INIT_HLIST_HEAD(&net->ipv4.fib_table_hash[i]);

  return fib4_rules_init(net);
 }
@@ -977,13 +988,14 @@ static void __net_exit ip_fib_net_exit(struct net *net)
   struct hlist_head *head;
   struct hlist_node *node, *tmp;

-  head = &fib_table_hash[i];
+  head = &net->ipv4.fib_table_hash[i];
   hlist_for_each_entry_safe(tb, node, tmp, head, tb_hlist) {
    hlist_del(node);
    tb->tb_flush(tb);
    kfree(tb);
   }
  }
+ kfree(net->ipv4.fib_table_hash);
 }

 static int __net_init fib_net_init(struct net *net)
--
1.5.3.rc5
```

Subject: [PATCH net-2.6.25 16/19] [NETNS] Correctly fill fib_config data.
Posted by den on Wed, 19 Dec 2007 15:24:46 GMT

```
Acked-by: Benjamin Thery <benjamin.thery@bull.net>
Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 net/ipv4/fib_frontend.c |   27 +++++++++++++++++----------
 1 files changed, 14 insertions(+), 13 deletions(-)

diff --git a/net/ipv4/fib_frontend.c b/net/ipv4/fib_frontend.c
index abe9f43..cecb660 100644
--- a/net/ipv4/fib_frontend.c
+++ b/net/ipv4/fib_frontend.c
@@ -305,14 +305,14 @@ static int put_rtax(struct nlattr *mx, int len, int type, u32 value)
  return len + nla_total_size(4);
 }

-static int rtentry_to_fib_config(int cmd, struct rtentry *rt,
+static int rtentry_to_fib_config(struct net *net, int cmd, struct rtentry *rt,
       struct fib_config *cfg)
 {
  __be32 addr;
  int plen;

  memset(cfg, 0, sizeof(*cfg));
- cfg->fc_nlinfo.nl_net = &init_net;
+ cfg->fc_nlinfo.nl_net = net;

  if (rt->rt_dst.sa_family != AF_INET)
   return -EAFNOSUPPORT;
@@ -373,7 +373,7 @@ static int rtentry_to_fib_config(int cmd, struct rtentry *rt,
   colon = strchr(devname, ':');
   if (colon)
    *colon = 0;
-  dev = __dev_get_by_name(&init_net, devname);
+  dev = __dev_get_by_name(net, devname);
   if (!dev)
    return -ENODEV;
   cfg->fc_oif = dev->ifindex;
@@ -396,7 +396,7 @@ static int rtentry_to_fib_config(int cmd, struct rtentry *rt,
   cfg->fc_gw = addr;
   if (rt->rt_flags & RTF_GATEWAY &&
-      inet_addr_type(&init_net, addr) == RTN_UNICAST)
+      inet_addr_type(net, addr) == RTN_UNICAST)
    cfg->fc_scope = RT_SCOPE_UNIVERSE;
  }

@@ -453,7 +453,7 @@ int ip_rt_ioctl(unsigned int cmd, void __user *arg)
    return -EFAULT;
```

```
   rtnl_lock();
-  err = rtentry_to_fib_config(cmd, &rt, &cfg);
+  err = rtentry_to_fib_config(&init_net, cmd, &rt, &cfg);
   if (err == 0) {
    struct fib_table *tb;

@@ -494,8 +494,8 @@ const struct nla_policy rtm_ipv4_policy[RTA_MAX+1] = {
  [RTA_FLOW]  = { .type = NLA_U32 },
 };

-static int rtm_to_fib_config(struct sk_buff *skb, struct nlmsghdr *nlh,
-       struct fib_config *cfg)
+static int rtm_to_fib_config(struct net *net, struct sk_buff *skb,
+       struct nlmsghdr *nlh, struct fib_config *cfg)
 {
  struct nlattr *attr;
  int err, remaining;
@@ -519,7 +519,7 @@ static int rtm_to_fib_config(struct sk_buff *skb, struct nlmsghdr *nlh,

  cfg->fc_nlinfo.pid = NETLINK_CB(skb).pid;
  cfg->fc_nlinfo.nlh = nlh;
- cfg->fc_nlinfo.nl_net = &init_net;
+ cfg->fc_nlinfo.nl_net = net;

  if (cfg->fc_type > RTN_MAX) {
   err = -EINVAL;
@@ -575,7 +575,7 @@ static int inet_rtm_delroute(struct sk_buff *skb, struct nlmsghdr* nlh, void
*ar
  if (net != &init_net)
   return -EINVAL;

- err = rtm_to_fib_config(skb, nlh, &cfg);
+ err = rtm_to_fib_config(net, skb, nlh, &cfg);
  if (err < 0)
   goto errout;

@@ -600,7 +600,7 @@ static int inet_rtm_newroute(struct sk_buff *skb, struct nlmsghdr* nlh,
void *ar
  if (net != &init_net)
   return -EINVAL;

- err = rtm_to_fib_config(skb, nlh, &cfg);
+ err = rtm_to_fib_config(net, skb, nlh, &cfg);
  if (err < 0)
   goto errout;

@@ -667,6 +667,7 @@ out:
```

```
 static void fib_magic(int cmd, int type, __be32 dst, int dst_len, struct in_ifaddr *ifa)
 {
+ struct net *net = ifa->ifa_dev->dev->nd_net;
  struct fib_table *tb;
  struct fib_config cfg = {
   .fc_protocol = RTPROT_KERNEL,
@@ -677,14 +678,14 @@ static void fib_magic(int cmd, int type, __be32 dst, int dst_len, struct
in_ifad
   .fc_oif = ifa->ifa_dev->dev->ifindex,
   .fc_nlflags = NLM_F_CREATE | NLM_F_APPEND,
   .fc_nlinfo = {
-   .nl_net = &init_net,
+   .nl_net = net,
   },
  };

  if (type == RTN_UNICAST)
-  tb = fib_new_table(&init_net, RT_TABLE_MAIN);
+  tb = fib_new_table(net, RT_TABLE_MAIN);
  else
-  tb = fib_new_table(&init_net, RT_TABLE_LOCAL);
+  tb = fib_new_table(net, RT_TABLE_LOCAL);

  if (tb == NULL)
   return;
--
1.5.3.rc5
```

Subject: [PATCH net-2.6.25 18/19] [NETNS] Replace init_net with the correct context in fib_frontend.c

Posted by den on Wed, 19 Dec 2007 15:24:48 GMT

View Forum Message <> Reply to Message

```
Acked-by: Benjamin Thery <benjamin.thery@bull.net>
Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 net/ipv4/fib_frontend.c |   24 ++++----------------------
 1 files changed, 4 insertions(+), 20 deletions(-)

diff --git a/net/ipv4/fib_frontend.c b/net/ipv4/fib_frontend.c
index a5c47fc..a5e2fb3 100644
--- a/net/ipv4/fib_frontend.c
+++ b/net/ipv4/fib_frontend.c
@@ -597,7 +594,7 @@ static int inet_rtm_newroute(struct sk_buff *skb, struct nlmsghdr* nlh,
void *ar
  if (err < 0)
```

```
   goto errout;

- tb = fib_new_table(&init_net, cfg.fc_table);
+ tb = fib_new_table(net, cfg.fc_table);
  if (tb == NULL) {
   err = -ENOBUFS;
   goto errout;
@@ -794,7 +785,7 @@ 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(&init_net, ifa->ifa_local) != RTN_LOCAL) {
+  if (inet_addr_type(dev->nd_net, ifa->ifa_local) != RTN_LOCAL) {
    /* And the last, but not the least thing.
       We must flush stray FIB entries.

@@ -802,7 +793,7 @@ static void fib_del_ifaddr(struct in_ifaddr *ifa)
       for stray nexthop entries, then ignite fib_flush.
    */
    if (fib_sync_down(ifa->ifa_local, NULL, 0))
-    fib_flush(&init_net);
+    fib_flush(dev->nd_net);
   }
  }
 #undef LOCAL_OK
@@ -894,7 +885,7 @@ static void nl_fib_lookup_exit(struct net *net)
 static void fib_disable_ip(struct net_device *dev, int force)
 {
  if (fib_sync_down(0, dev, force))
-  fib_flush(&init_net);
+  fib_flush(dev->nd_net);
  rt_cache_flush(0);
  arp_ifdown(dev);
 }
--
1.5.3.rc5
```

Subject: [PATCH net-2.6.25 19/19] [NETNS] Enable routing configuration in non-initial namespace.

Posted by den on Wed, 19 Dec 2007 15:24:49 GMT

View Forum Message <> Reply to Message

I.e. remove the net != &init_net checks from the places, that now can handle other-than-init net namespace.

Acked-by: Benjamin Thery <benjamin.thery@bull.net>
Signed-off-by: Denis V. Lunev <den@openvz.org>

```
---
 net/ipv4/fib_frontend.c |   24 ++++--------------------
 1 files changed, 4 insertions(+), 20 deletions(-)

diff --git a/net/ipv4/fib_frontend.c b/net/ipv4/fib_frontend.c
index a5c47fc..a5e2fb3 100644
--- a/net/ipv4/fib_frontend.c
+++ b/net/ipv4/fib_frontend.c
@@ -572,9 +572,6 @@ static int inet_rtm_delroute(struct sk_buff *skb, struct nlmsghdr* nlh, void
*ar
  struct fib_table *tb;
  int err;

- if (net != &init_net)
-  return -EINVAL;
-
  err = rtm_to_fib_config(net, skb, nlh, &cfg);
  if (err < 0)
   goto errout;
@@ -597,9 +594,6 @@ static int inet_rtm_newroute(struct sk_buff *skb, struct nlmsghdr* nlh,
void *ar
  struct fib_table *tb;
  int err;

- if (net != &init_net)
-  return -EINVAL;
-
  err = rtm_to_fib_config(net, skb, nlh, &cfg);
  if (err < 0)
   goto errout;
@@ -625,9 +619,6 @@ static int inet_dump_fib(struct sk_buff *skb, struct netlink_callback *cb)
  struct hlist_head *head;
  int dumped = 0;

- if (net != &init_net)
-  return 0;
-
  if (nlmsg_len(cb->nlh) >= sizeof(struct rtmsg) &&
      ((struct rtmsg *) nlmsg_data(cb->nlh))->rtm_flags & RTM_F_CLONED)
   return ip_rt_dump(skb, cb);
@@ -931,9 +922,6 @@ static int fib_netdev_event(struct notifier_block *this, unsigned long
event, vo
  struct net_device *dev = ptr;
  struct in_device *in_dev = __in_dev_get_rtnl(dev);

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

```
  if (event == NETDEV_UNREGISTER) {
   fib_disable_ip(dev, 2);
   return NOTIFY_DONE;
@@ -1013,10 +1001,6 @@ static int __net_init fib_net_init(struct net *net)
 {
  int error;

- error = 0;
- if (net != &init_net)
-  goto out;
-
  error = ip_fib_net_init(net);
  if (error < 0)
   goto out;
--
1.5.3.rc5
```

Subject: Re: [PATCH net-2.6.25 1/19] [NETNS] Add netns parameter to fib_rules_(un)register.
Posted by davem on Thu, 20 Dec 2007 23:46:36 GMT
View Forum Message <> Reply to Message

From: "Denis V. Lunev" <den@openvz.org>
Date: Wed, 19 Dec 2007 18:24:31 +0300

> @@ -101,14 +101,12 @@ static inline u32 frh_get_table(struct fib_rule_hdr *frh, struct nlattr
**nla)
>   return frh->table;
> }
>
> -extern int   fib_rules_register(struct fib_rules_ops *);
> -extern int   fib_rules_unregister(struct fib_rules_ops *);
> -extern void   fib_rules_cleanup_ops(struct fib_rules_ops *);
> +extern int fib_rules_register(struct net *, struct fib_rules_ops *);
> +extern int fib_rules_unregister(struct net *, struct fib_rules_ops *);
> +extern void fib_rules_cleanup_ops(struct fib_rules_ops *);
>
> -extern int   fib_rules_lookup(struct fib_rules_ops *,
> -   struct flowi *, int flags,
> -   struct fib_lookup_arg *);
> -extern int   fib_default_rule_add(struct fib_rules_ops *,
> -   u32 pref, u32 table,
> -   u32 flags);
> +extern int fib_rules_lookup(struct fib_rules_ops *, struct flowi *, int flags,
> +   struct fib_lookup_arg *);
> +extern int fib_default_rule_add(struct fib_rules_ops *, u32 pref, u32 table,
> +   u32 flags);

> #endif

Please do not make gratuitous coding style changes like this!

What bothers you so much that there is lots of whitespace there after the "extern int"? Does it bother you so much that you think the side effect of your patch being unreadable is worth it?!?!

Why is it unreadable? I'm glad you asked....

Just like me, someone will have to read this over carefully to see what you're actually doing.

Are you deleting all the existing declarations and adding new ones with different names?

Are you deleting some of them, but keeping others yet changing the arguments to them somehow?

Are you deleting some of them, but masterbating with the coding style of others?

NOBODY KNOWS!

Whereas if you just deleted the lines for the functions you are removing, it would be totally clear what is happening.

This patch, from a reviewability standpoint, sucks. It makes efficient patch review next to impossible.

I'm not looking at the rest of this patch set, clean this stuff up and resubmit it all, thank you.