Subject: [PATCH 1/2] Introduce the generic rtnl_create_link()
Posted by Pavel Emelianov on Thu, 19 Jul 2007 09:26:52 GMT
View Forum Message <> Reply to Message

This routine gets the parsed rtnl attributes and creates a new
link with generic info (IFLA_LINKINFO policy). Its intention
is to help the drivers, that need to create several links at
once (like VETH).

This is nothing but a copy-paste-ed part of rtnl_newlink() function
that is responsible for creation of new device.

Signed-off-by: Pavel Emelianov <xemul@openvz.org>
Acked-by: Patrick McHardy <kaber@trash.net>

---

 include/net/rtnetlink.h |    4 ++
 net/core/rtnetlink.c    |   83 +++++++++++++++++++++++++++++++----------------
 2 files changed, 57 insertions(+), 30 deletions(-)

```
diff --git a/include/net/rtnetlink.h b/include/net/rtnetlink.h
index 3861c05..8218288 100644
--- a/include/net/rtnetlink.h
+++ b/include/net/rtnetlink.h
@@ -78,6 +78,10 @@ extern void __rtnl_link_unregister(struc
 extern int rtnl_link_register(struct rtnl_link_ops *ops);
 extern void rtnl_link_unregister(struct rtnl_link_ops *ops);

+extern struct net_device *rtnl_create_link(char *ifname,
+  const struct rtnl_link_ops *ops, struct nlattr *tb[]);
+extern const struct nla_policy ifla_policy[IFLA_MAX+1];
+
 #define MODULE_ALIAS_RTNL_LINK(kind) MODULE_ALIAS("rtnl-link-" kind)

 #endif
diff --git a/net/core/rtnetlink.c b/net/core/rtnetlink.c
index 864cbdf..02e00e8 100644
--- a/net/core/rtnetlink.c
+++ b/net/core/rtnetlink.c
@@ -714,7 +714,7 @@ cont:
  return skb->len;
 }

-static const struct nla_policy ifla_policy[IFLA_MAX+1] = {
+const struct nla_policy ifla_policy[IFLA_MAX+1] = {
  [IFLA_IFNAME]  = { .type = NLA_STRING, .len = IFNAMSIZ-1 },
  [IFLA_ADDRESS]  = { .type = NLA_BINARY, .len = MAX_ADDR_LEN },
```

```
  [IFLA_BROADCAST] = { .type = NLA_BINARY, .len = MAX_ADDR_LEN },
@@ -941,6 +941,50 @@ static int rtnl_dellink(struct sk_buff *
  return 0;
 }

+struct net_device *rtnl_create_link(char *ifname,
+  const struct rtnl_link_ops *ops, struct nlattr *tb[])
+{
+ int err;
+ struct net_device *dev;
+
+ err = -ENOMEM;
+ dev = alloc_netdev(ops->priv_size, ifname, ops->setup);
+ if (!dev)
+  goto err;
+
+ if (strchr(dev->name, '%')) {
+  err = dev_alloc_name(dev, dev->name);
+  if (err < 0)
+   goto err_free;
+ }
+
+ dev->rtnl_link_ops = ops;
+
+ if (tb[IFLA_MTU])
+  dev->mtu = nla_get_u32(tb[IFLA_MTU]);
+ if (tb[IFLA_ADDRESS])
+  memcpy(dev->dev_addr, nla_data(tb[IFLA_ADDRESS]),
+    nla_len(tb[IFLA_ADDRESS]));
+ if (tb[IFLA_BROADCAST])
+  memcpy(dev->broadcast, nla_data(tb[IFLA_BROADCAST]),
+    nla_len(tb[IFLA_BROADCAST]));
+ if (tb[IFLA_TXQLEN])
+  dev->tx_queue_len = nla_get_u32(tb[IFLA_TXQLEN]);
+ if (tb[IFLA_WEIGHT])
+  dev->weight = nla_get_u32(tb[IFLA_WEIGHT]);
+ if (tb[IFLA_OPERSTATE])
+  set_operstate(dev, nla_get_u8(tb[IFLA_OPERSTATE]));
+ if (tb[IFLA_LINKMODE])
+  dev->link_mode = nla_get_u8(tb[IFLA_LINKMODE]);
+
+ return dev;
+
+err_free:
+ free_netdev(dev);
+err:
+ return ERR_PTR(err);
+}
```

```
+
 static int rtnl_newlink(struct sk_buff *skb, struct nlmsghdr *nlh, void *arg)
 {
  const struct rtnl_link_ops *ops;
@@ -1051,40 +1095,17 @@ replay:

   if (!ifname[0])
    snprintf(ifname, IFNAMSIZ, "%s%%d", ops->kind);
-  dev = alloc_netdev(ops->priv_size, ifname, ops->setup);
-  if (!dev)
-   return -ENOMEM;
-
-  if (strchr(dev->name, '%')) {
-   err = dev_alloc_name(dev, dev->name);
-   if (err < 0)
-    goto err_free;
-  }
-  dev->rtnl_link_ops = ops;

-  if (tb[IFLA_MTU])
-   dev->mtu = nla_get_u32(tb[IFLA_MTU]);
-  if (tb[IFLA_ADDRESS])
-   memcpy(dev->dev_addr, nla_data(tb[IFLA_ADDRESS]),
-          nla_len(tb[IFLA_ADDRESS]));
-  if (tb[IFLA_BROADCAST])
-   memcpy(dev->broadcast, nla_data(tb[IFLA_BROADCAST]),
-          nla_len(tb[IFLA_BROADCAST]));
-  if (tb[IFLA_TXQLEN])
-   dev->tx_queue_len = nla_get_u32(tb[IFLA_TXQLEN]);
-  if (tb[IFLA_WEIGHT])
-   dev->weight = nla_get_u32(tb[IFLA_WEIGHT]);
-  if (tb[IFLA_OPERSTATE])
-   set_operstate(dev, nla_get_u8(tb[IFLA_OPERSTATE]));
-  if (tb[IFLA_LINKMODE])
-   dev->link_mode = nla_get_u8(tb[IFLA_LINKMODE]);
+  dev = rtnl_create_link(ifname, ops, tb);

-  if (ops->newlink)
+  if (IS_ERR(dev))
+   err = PTR_ERR(dev);
+  else if (ops->newlink)
    err = ops->newlink(dev, tb, data);
   else
    err = register_netdevice(dev);
-err_free:
-  if (err < 0)
+
+  if (err < 0 && !IS_ERR(dev))
```

```
    free_netdev(dev);
   return err;
  }
@@ -1333,3 +1354,5 @@ EXPORT_SYMBOL(rtnl_unlock);
 EXPORT_SYMBOL(rtnl_unicast);
 EXPORT_SYMBOL(rtnl_notify);
 EXPORT_SYMBOL(rtnl_set_sk_err);
+EXPORT_SYMBOL(rtnl_create_link);
+EXPORT_SYMBOL(ifla_policy);
```