Subject: [PATCH] Virtual ethernet tunnel
Posted by Pavel Emelianov on Wed, 06 Jun 2007 15:11:38 GMT
View Forum Message <> Reply to Message

Veth stands for Virtual ETHernet. It is a simple tunnel driver
that works at the link layer and looks like a pair of ethernet
devices interconnected with each other.

Mainly it allows to communicate between network namespaces but
it can be used as is as well.

Eric recently sent a similar driver called etun. This
implementation uses another interface - the RTM_NRELINK
message introduced by Patric. The patch fits today netdev
tree with Patrick's patches.

The newlink callback is organized that way to make it easy
to create the peer device in the separate namespace when we
have them in kernel.

The patch for an ip utility is also provided.

Eric, since ethtool interface was from your patch, I add
your Signed-off-by line.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
Signed-off-by: Pavel Emelianov <xemul@openvz.org>

```
diff --git a/drivers/net/Kconfig b/drivers/net/Kconfig
index 7d57f4a..7e144be 100644
--- a/drivers/net/Kconfig
+++ b/drivers/net/Kconfig
@@ -119,6 +119,12 @@ config TUN

    If you don't know what to use this for, you don't need it.

+config VETH
+ tristate "Virtual ethernet device"
+ ---help---
+   The device is an ethernet tunnel. Devices are created in pairs. When
+   one end receives the packet it appears on its pair and vice versa.
+
 config NET_SB1000
  tristate "General Instruments Surfboard 1000"
  depends on PNP
diff --git a/drivers/net/Makefile b/drivers/net/Makefile
```

```
index a77affa..4764119 100644
--- a/drivers/net/Makefile
+++ b/drivers/net/Makefile
@@ -185,6 +185,7 @@ obj-$(CONFIG_MACSONIC) += macsonic.o
 obj-$(CONFIG_MACMACE) += macmace.o
 obj-$(CONFIG_MAC89x0) += mac89x0.o
 obj-$(CONFIG_TUN) += tun.o
+obj-$(CONFIG_VETH) += veth.o
 obj-$(CONFIG_NET_NETX) += netx-eth.o
 obj-$(CONFIG_DL2K) += dl2k.o
 obj-$(CONFIG_R8169) += r8169.o
diff --git a/drivers/net/veth.c b/drivers/net/veth.c
new file mode 100644
index 0000000..6746c91
--- /dev/null
+++ b/drivers/net/veth.c
@@ -0,0 +1,391 @@
+/*
+ *  drivers/net/veth.c
+ *
+ *  Copyright (C) 2007 OpenVZ http://openvz.org, SWsoft Inc
+ *
+ */
+
+#include <linux/list.h>
+#include <linux/netdevice.h>
+#include <linux/ethtool.h>
+#include <linux/etherdevice.h>
+
+#include <net/dst.h>
+#include <net/xfrm.h>
+#include <net/veth.h>
+
+#define DRV_NAME "veth"
+#define DRV_VERSION "1.0"
+
+struct veth_priv {
+ struct net_device *peer;
+ struct net_device *dev;
+ struct list_head list;
+ struct net_device_stats stats;
+ unsigned ip_summed;
+};
+
+static LIST_HEAD(veth_list);
+
+/*
+ * ethtool interface
```

```
+ */
+
+static struct {
+ const char string[ETH_GSTRING_LEN];
+} ethtool_stats_keys[] = {
+ { "peer_ifindex" },
+};
+
+static int veth_get_settings(struct net_device *dev, struct ethtool_cmd *cmd)
+{
+ cmd->supported  = 0;
+ cmd->advertising = 0;
+ cmd->speed  = SPEED_10000;
+ cmd->duplex  = DUPLEX_FULL;
+ cmd->port  = PORT_TP;
+ cmd->phy_address = 0;
+ cmd->transceiver = XCVR_INTERNAL;
+ cmd->autoneg  = AUTONEG_DISABLE;
+ cmd->maxtxpkt  = 0;
+ cmd->maxrxpkt  = 0;
+ return 0;
+}
+
+static void veth_get_drvinfo(struct net_device *dev, struct ethtool_drvinfo *info)
+{
+ strcpy(info->driver, DRV_NAME);
+ strcpy(info->version, DRV_VERSION);
+ strcpy(info->fw_version, "N/A");
+}
+
+static void veth_get_strings(struct net_device *dev, u32 stringset, u8 *buf)
+{
+ switch(stringset) {
+ case ETH_SS_STATS:
+  memcpy(buf, &ethtool_stats_keys, sizeof(ethtool_stats_keys));
+  break;
+ }
+}
+
+static int veth_get_stats_count(struct net_device *dev)
+{
+ return ARRAY_SIZE(ethtool_stats_keys);
+}
+
+static void veth_get_ethtool_stats(struct net_device *dev,
+  struct ethtool_stats *stats, u64 *data)
+{
+ struct veth_priv *priv;
```

```
+
+ priv = netdev_priv(dev);
+ data[0] = priv->peer->ifindex;
+}
+
+static u32 veth_get_rx_csum(struct net_device *dev)
+{
+ struct veth_priv *priv;
+
+ priv = netdev_priv(dev);
+ return priv->ip_summed == CHECKSUM_UNNECESSARY;
+}
+
+static int veth_set_rx_csum(struct net_device *dev, u32 data)
+{
+ struct veth_priv *priv;
+
+ priv = netdev_priv(dev);
+ priv->ip_summed = data ? CHECKSUM_UNNECESSARY : CHECKSUM_NONE;
+ return 0;
+}
+
+static u32 veth_get_tx_csum(struct net_device *dev)
+{
+ return (dev->features & NETIF_F_NO_CSUM) != 0;
+}
+
+static int veth_set_tx_csum(struct net_device *dev, u32 data)
+{
+ if (data)
+  dev->features |= NETIF_F_NO_CSUM;
+ else
+  dev->features &= ~NETIF_F_NO_CSUM;
+ return 0;
+}
+
+static struct ethtool_ops veth_ethtool_ops = {
+ .get_settings  = veth_get_settings,
+ .get_drvinfo  = veth_get_drvinfo,
+ .get_link  = ethtool_op_get_link,
+ .get_rx_csum  = veth_get_rx_csum,
+ .set_rx_csum  = veth_set_rx_csum,
+ .get_tx_csum  = veth_get_tx_csum,
+ .set_tx_csum  = veth_set_tx_csum,
+ .get_sg   = ethtool_op_get_sg,
+ .set_sg   = ethtool_op_set_sg,
+ .get_strings  = veth_get_strings,
+ .get_stats_count = veth_get_stats_count,
```

```
+ .get_ethtool_stats = veth_get_ethtool_stats,
+ .get_perm_addr  = ethtool_op_get_perm_addr,
+};
+
+/*
+ * xmit
+ */
+
+static int veth_xmit(struct sk_buff *skb, struct net_device *dev)
+{
+ struct net_device *rcv = NULL;
+ struct veth_priv *priv, *rcv_priv;
+ int length;
+
+ skb_orphan(skb);
+
+ priv = netdev_priv(dev);
+ rcv = priv->peer;
+ rcv_priv = netdev_priv(rcv);
+
+ if (!(rcv->flags & IFF_UP))
+  goto outf;
+
+ skb->dev = rcv;
+ skb->pkt_type = PACKET_HOST;
+ skb->protocol = eth_type_trans(skb, rcv);
+ if (dev->features & NETIF_F_NO_CSUM)
+  skb->ip_summed = rcv_priv->ip_summed;
+
+ dst_release(skb->dst);
+ skb->dst = NULL;
+
+ secpath_reset(skb);
+ nf_reset(skb);
+
+ length = skb->len;
+
+ priv->stats.tx_bytes += length;
+ priv->stats.tx_packets++;
+
+ rcv_priv->stats.rx_bytes += length;
+ rcv_priv->stats.rx_packets++;
+
+ netif_rx(skb);
+ return 0;
+
+outf:
+ kfree_skb(skb);
```

```
+ priv->stats.tx_dropped++;
+ return 0;
+}
+
+/*
+ * general routines
+ */
+
+static struct net_device_stats *veth_get_stats(struct net_device *dev)
+{
+ struct veth_priv *priv;
+
+ priv = netdev_priv(dev);
+ return &priv->stats;
+}
+
+static int veth_open(struct net_device *dev)
+{
+ struct veth_priv *priv;
+
+ priv = netdev_priv(dev);
+ if (priv->peer == NULL)
+  return -ENOTCONN;
+
+ if (priv->peer->flags & IFF_UP) {
+  netif_carrier_on(dev);
+  netif_carrier_on(priv->peer);
+ }
+ return 0;
+}
+
+static int veth_close(struct net_device *dev)
+{
+ struct veth_priv *priv;
+
+ if (netif_carrier_ok(dev)) {
+  priv = netdev_priv(dev);
+  netif_carrier_off(dev);
+  netif_carrier_off(priv->peer);
+ }
+ return 0;
+}
+
+/*
+ * netlink interface
+ */
+
+static void veth_setup(struct net_device *dev)
```

```
+{
+ ether_setup(dev);
+
+ dev->hard_start_xmit = veth_xmit;
+ dev->get_stats = veth_get_stats;
+ dev->open = veth_open;
+ dev->stop = veth_close;
+ dev->destructor = free_netdev;
+ dev->ethtool_ops = &veth_ethtool_ops;
+ dev->features |= NETIF_F_LLTX;
+ netif_carrier_off(dev);
+}
+
+static int veth_newlink(struct net_device *dev,
+    struct nlattr *tb[], struct nlattr *data[])
+{
+ int err;
+ struct net_device *peer;
+ struct veth_priv *priv;
+ char ifname[IFNAMSIZ];
+
+ /*
+  * setup the first device
+  */
+
+ if (data != NULL && data[VETH_INFO_MAC] != NULL)
+  memcpy(dev->dev_addr,
+    nla_data(data[VETH_INFO_MAC]), ETH_ALEN);
+ else
+  random_ether_addr(dev->dev_addr);
+
+ err = register_netdevice(dev);
+ if (err < 0)
+  goto err_register_dev;
+
+ /*
+  * alloc and setup the second one
+  *
+  * this should be done in another namespace, but we
+  * do not have them yet
+  */
+
+ if (data != NULL && data[VETH_INFO_PEER] != NULL)
+  nla_strlcpy(ifname, data[VETH_INFO_PEER], IFNAMSIZ);
+ else
+  snprintf(ifname, IFNAMSIZ, "veth%%d");
+
+ err = -ENOMEM;
```

```
+ peer = alloc_netdev(sizeof(struct veth_priv), ifname, veth_setup);
+ if (peer == NULL)
+  goto err_alloc;
+
+ if (strchr(peer->name, '%')) {
+  err = dev_alloc_name(peer, peer->name);
+  if (err < 0)
+   goto err_name;
+ }
+
+ if (data != NULL && data[VETH_INFO_PEER_MAC] != NULL)
+  memcpy(peer->dev_addr,
+    nla_data(data[VETH_INFO_PEER_MAC]), ETH_ALEN);
+ else
+  random_ether_addr(peer->dev_addr);
+
+ /* this should be in sync with rtnl_newlink */
+ peer->mtu = dev->mtu;
+ peer->tx_queue_len = dev->tx_queue_len;
+ peer->weight = dev->weight;
+ peer->link_mode = dev->link_mode;
+ peer->rtnl_link_ops = dev->rtnl_link_ops;
+
+ if (peer->operstate != dev->operstate) {
+  write_lock_bh(&dev_base_lock);
+  peer->operstate = dev->operstate;
+  write_unlock_bh(&dev_base_lock);
+  netdev_state_change(peer);
+ }
+
+ err = register_netdevice(peer);
+ if (err < 0)
+  goto err_register_peer;
+
+ /*
+  * tie the deviced together
+  */
+
+ priv = netdev_priv(dev);
+ priv->dev = dev;
+ priv->peer = peer;
+ list_add(&priv->list, &veth_list);
+
+ priv = netdev_priv(peer);
+ priv->dev = peer;
+ priv->peer = dev;
+ INIT_LIST_HEAD(&priv->list);
+ return 0;
```

```
+
+err_register_peer:
+ /* nothing special to do */
+err_name:
+ free_netdev(peer);
+err_alloc:
+ unregister_netdevice(dev);
+err_register_dev:
+ return err;
+}
+
+static void veth_dellink(struct net_device *dev)
+{
+ struct veth_priv *priv;
+ struct net_device *peer;
+
+ priv = netdev_priv(dev);
+ peer = priv->peer;
+
+ if (!list_empty(&priv->list))
+  list_del(&priv->list);
+
+ priv = netdev_priv(peer);
+ if (!list_empty(&priv->list))
+  list_del(&priv->list);
+
+ unregister_netdevice(dev);
+ unregister_netdevice(peer);
+}
+
+static const struct nla_policy veth_policy[VETH_INFO_MAX] = {
+ [VETH_INFO_MAC]  = { .type = NLA_BINARY, .len = ETH_ALEN },
+ [VETH_INFO_PEER] = { .type = NLA_STRING },
+ [VETH_INFO_PEER_MAC] = { .type = NLA_BINARY, .len = ETH_ALEN },
+};
+
+static struct rtnl_link_ops veth_link_ops = {
+ .name  = "veth",
+ .priv_size = sizeof(struct veth_priv),
+ .setup  = veth_setup,
+ .newlink = veth_newlink,
+ .dellink = veth_dellink,
+ .policy  = veth_policy,
+ .maxtype = VETH_INFO_MAX,
+};
+
+/*
+ * init/fini
```

```
+ */
+
+static __init int veth_init(void)
+{
+ return rtnl_link_register(&veth_link_ops);
+}
+
+static __exit void veth_exit(void)
+{
+ struct veth_priv *priv, *next;
+
+ rtnl_lock();
+ __rtnl_link_unregister(&veth_link_ops);
+
+ list_for_each_entry_safe(priv, next, &veth_list, list)
+  veth_dellink(priv->dev);
+ rtnl_unlock();
+}
+
+module_init(veth_init);
+module_exit(veth_exit);
+
+MODULE_DESCRIPTION("Virtual Ethernet Tunnel");
+MODULE_LICENSE("GPL v2");
diff --git a/include/net/veth.h b/include/net/veth.h
new file mode 100644
index 0000000..74c8e1e
--- /dev/null
+++ b/include/net/veth.h
@@ -0,0 +1,13 @@
+#ifndef __NET_VETH_H__
+#define __NET_VETH_H__
+
+enum {
+ VETH_INFO_UNSPEC,
+ VETH_INFO_MAC,
+ VETH_INFO_PEER,
+ VETH_INFO_PEER_MAC,
+
+ VETH_INFO_MAX
+};
+
+#endif
```

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: [PATCH] Module for ip utility to support veth device
Posted by Pavel Emelianov on Wed, 06 Jun 2007 15:17:19 GMT
View Forum Message <> Reply to Message

The usage is
ip link add [name] type veth [peer <name>] [mac <mac>] [peer_mac <mac>]

The Makefile is maybe not as beautiful as it could be. It
is to be discussed.

One thing I noticed during testing is the following. When launching
this with link_veth.so module and not specifying any module specific
parameters, the kernel refuses to accept the packet when parsing the
IFLA_LINKINFO. So the hunk for ip/iplink.c doesn't add an empty extra
header if no extra data expected.

Signed-off-by: Pavel Emelianov <xemul@openvz.org>

```
diff --git a/ip/Makefile b/ip/Makefile
index 9a5bfe3..b46bce3 100644
--- a/ip/Makefile
+++ b/ip/Makefile
@@ -8,8 +8,9 @@ RTMONOBJ=rtmon.o
 ALLOBJ=$(IPOBJ) $(RTMONOBJ)
 SCRIPTS=ifcfg rtpr routel routef
 TARGETS=ip rtmon
+LIBS=link_veth.so

-all: $(TARGETS) $(SCRIPTS)
+all: $(TARGETS) $(SCRIPTS) $(LIBS)

 ip: $(IPOBJ) $(LIBNETLINK) $(LIBUTIL)

@@ -24,3 +25,6 @@ clean:

 LDLIBS += -ldl
 LDFLAGS += -Wl,-export-dynamic
+
+%.so: %.c
+ $(CC) $(CFLAGS) -shared $< -o $@
diff --git a/ip/iplink.c b/ip/iplink.c
index 5170419..6975990 100644
--- a/ip/iplink.c
+++ b/ip/iplink.c
@@ -287,7 +287,7 @@ static int iplink_modify(int cmd, unsign
     strlen(type));
```

```
   lu = get_link_type(type);
-  if (lu) {
+  if (lu && argc) {
     struct rtattr * data = NLMSG_TAIL(&req.n);
     addattr_l(&req.n, sizeof(req), IFLA_INFO_DATA, NULL, 0);

diff --git a/ip/link_veth.c b/ip/link_veth.c
new file mode 100644
index 0000000..adfdef6
--- /dev/null
+++ b/ip/link_veth.c
@@ -0,0 +1,77 @@
+#include <string.h>
+
+#include "utils.h"
+#include "ip_common.h"
+#include "veth.h"
+
+#define ETH_ALEN 6
+
+static void usage(void)
+{
+ printf("Usage: ip link add ... "
+    "[peer <peer-name>] [mac <mac>] [peer_mac <mac>]\n");
+}
+
+static int veth_parse_opt(struct link_util *lu, int argc, char **argv,
+  struct nlmsghdr *hdr)
+{
+ __u8 mac[ETH_ALEN];
+
+ for (; argc != 0; argv++, argc--) {
+  if (strcmp(*argv, "peer") == 0) {
+   argv++;
+   argc--;
+   if (argc == 0) {
+    usage();
+    return -1;
+   }
+
+   addattr_l(hdr, 1024, VETH_INFO_PEER,
+     *argv, strlen(*argv));
+
+   continue;
+  }
+
+  if (strcmp(*argv, "mac") == 0) {
+   argv++;
```

```
+   argc--;
+   if (argc == 0) {
+    usage();
+    return -1;
+   }
+
+   if (hexstring_a2n(*argv, mac, sizeof(mac)) == NULL)
+    return -1;
+
+   addattr_l(hdr, 1024, VETH_INFO_MAC,
+     mac, ETH_ALEN);
+   continue;
+  }
+
+  if (strcmp(*argv, "peer_mac") == 0) {
+   argv++;
+   argc--;
+   if (argc == 0) {
+    usage();
+    return -1;
+   }
+
+   if (hexstring_a2n(*argv, mac, sizeof(mac)) == NULL)
+    return -1;
+
+   addattr_l(hdr, 1024, VETH_INFO_PEER_MAC,
+     mac, ETH_ALEN);
+   continue;
+  }
+
+  usage();
+  return -1;
+ }
+
+ return 0;
+}
+
+struct link_util veth_link_util = {
+ .id = "veth",
+ .parse_opt = veth_parse_opt,
+};
diff --git a/ip/veth.h b/ip/veth.h
new file mode 100644
index 0000000..74c8e1e
--- /dev/null
+++ b/ip/veth.h
@@ -0,0 +1,13 @@
+#ifndef __NET_VETH_H__
```

```
+#define __NET_VETH_H__
+
+enum {
+ VETH_INFO_UNSPEC,
+ VETH_INFO_MAC,
+ VETH_INFO_PEER,
+ VETH_INFO_PEER_MAC,
+
+ VETH_INFO_MAX
+};
+
+#endif
```

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH] Virtual ethernet tunnel
Posted by Daniel Lezcano on Thu, 07 Jun 2007 09:29:50 GMT
View Forum Message <> Reply to Message

Pavel Emelianov wrote:
> Patrick McHardy wrote:
>
>> Pavel Emelianov wrote:
>>
>>> Veth stands for Virtual ETHernet. It is a simple tunnel driver
>>> that works at the link layer and looks like a pair of ethernet
>>> devices interconnected with each other.
>>>
>>> Mainly it allows to communicate between network namespaces but
>>> it can be used as is as well.
>>>
>>> Eric recently sent a similar driver called etun. This
>>> implementation uses another interface - the RTM_NRELINK
>>> message introduced by Patric. The patch fits today netdev
>>> tree with Patrick's patches.
>>>
>>> The newlink callback is organized that way to make it easy
>>> to create the peer device in the separate namespace when we
>>> have them in kernel.
>>>
>>>
>>> +struct veth_priv {
>>> + struct net_device *peer;
>>> + struct net_device *dev;
>>> + struct list_head list;

>>> + struct net_device_stats stats;
>>>
>> You can use dev->stats instead.
>>
>
> OK. Actually I planned to use percpu stats to reduce cacheline
> trashing (Stephen has noticed it also). The reason I didn't do it
> here is that the patch would look more complicated, but I wanted to
> show and approve the netlink interface first.
>
>
>>> +static int veth_xmit(struct sk_buff *skb, struct net_device *dev)
>>> +{
>>> + struct net_device *rcv = NULL;
>>> + struct veth_priv *priv, *rcv_priv;
>>> + int length;
>>> +
>>> + skb_orphan(skb);
>>> +
>>> + priv = netdev_priv(dev);
>>> + rcv = priv->peer;
>>> + rcv_priv = netdev_priv(rcv);
>>> +
>>> + if (!(rcv->flags & IFF_UP))
>>> + goto outf;
>>> +
>>> + skb->dev = rcv;
>>>
>> eth_type_trans already sets skb->dev.
>>
>
> Ok. Thanks.
>
>
>>> + skb->pkt_type = PACKET_HOST;
>>> + skb->protocol = eth_type_trans(skb, rcv);
>>> + if (dev->features & NETIF_F_NO_CSUM)
>>> + skb->ip_summed = rcv_priv->ip_summed;
>>> +
>>> + dst_release(skb->dst);
>>> + skb->dst = NULL;
>>> +
>>> + secpath_reset(skb);
>>> + nf_reset(skb);
>>>
>> Is skb->mark supposed to survive communication between different
>> namespaces?
>>

>
> I guess it must not. Thanks.
>
>
>>> +static const struct nla_policy veth_policy[VETH_INFO_MAX] = {
>>> + [VETH_INFO_MAC] = { .type = NLA_BINARY, .len = ETH_ALEN },
>>> + [VETH_INFO_PEER] = { .type = NLA_STRING },
>>> + [VETH_INFO_PEER_MAC] = { .type = NLA_BINARY, .len = ETH_ALEN },
>>> +};
>>>
>> The rtnl_link codes looks fine. I don't like the VETH_INFO_MAC attribute
>> very much though, we already have a generic device attribute for MAC
>> addresses. Of course that only allows you to supply one MAC address, so
>> I'm wondering what you think of allocating only a single device per
>> newlink operation and binding them in a seperate enslave operation?
>>
>
> I did this at the very first version, but Alexey showed me that this
> would be wrong. Look. When we create the second device it must be in
> the other namespace as it is useless to have them in one namespace.
> But if we have the device in the other namespace the RTNL_NEWLINK
> message from kernel would come into this namespace thus confusing ip
> utility in the init namespace. Creating the device in the init ns and
> moving it into the new one is rather a complex task.
>
Pavel,

moving the netdevice to another namespace is not a complex task. Eric
Biederman did it in its patchset (cf. http://lxc.sf.net/network)

When the pair device is created, both extremeties are into the init
namespace and you can choose to which namespace to move one extremity.
When the network namespace dies, the netdev is moved back to the init
namespace.
That facilitate network device management.

Concerning netlink events, this is automatically generated when the
network device is moved through namespaces.

IMHO, we should have the network device movement between namespaces in
order to be able to move a physical network device too (eg. you have 4
NIC and you want to create 3 containers and assign 3 NIC to each of them)

> But with such approach the creation looks really logical. We send a
> packet to the kernel and have a single response about the new device
> appearance. At the same time we have a RTNL_NEWLINK message arrived at
> the destination namespace informing that a new device has appeared
> there as well.

>
>
>>> +enum {
>>> + VETH_INFO_UNSPEC,
>>> + VETH_INFO_MAC,
>>> + VETH_INFO_PEER,
>>> + VETH_INFO_PEER_MAC,
>>> +
>>> + VETH_INFO_MAX
>>> +};
>>>
>> Please follow the
>>
>> #define VETH_INFO_MAX (__VETH_INFO_MAX - 1)
>>
>> convention here.
>>
>
> Could you please clarify this point. I saw the lines
> enum {
> ...
> RTNL_NEWLINK
> #define RTNL_NEWLINK RTNL_NEWLINK
> ...
> }
> and had my brains exploded imagining what this would mean :(
>
>
>> -
>> To unsubscribe from this list: send the line "unsubscribe netdev" in
>> the body of a message to majordomo@vger.kernel.org
>> More majordomo info at http://vger.kernel.org/majordomo-info.html
>>
>>
>
> _____
> Containers mailing list
> Containers@lists.linux-foundation.org
> https://lists.linux-foundation.org/mailman/listinfo/containers
>
>

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: Re: [PATCH] Virtual ethernet tunnel
Posted by Pavel Emelianov on Thu, 07 Jun 2007 09:51:52 GMT
View Forum Message <> Reply to Message

>> I did this at the very first version, but Alexey showed me that this
>> would be wrong. Look. When we create the second device it must be in
>> the other namespace as it is useless to have them in one namespace.
>> But if we have the device in the other namespace the RTNL_NEWLINK
>> message from kernel would come into this namespace thus confusing ip
>> utility in the init namespace. Creating the device in the init ns and
>> moving it into the new one is rather a complex task.
>>
> Pavel,
>
> moving the netdevice to another namespace is not a complex task. Eric
> Biederman did it in its patchset (cf. http://lxc.sf.net/network)

By saying complex I didn't mean that this is difficult to implement,
but that it consists (must consist) of many stages. I.e. composite.
Making the device right in the namespace is liter.

> When the pair device is created, both extremeties are into the init
> namespace and you can choose to which namespace to move one extremity.

I do not mind that.

> When the network namespace dies, the netdev is moved back to the init
> namespace.
> That facilitate network device management.
>
> Concerning netlink events, this is automatically generated when the
> network device is moved through namespaces.
>
> IMHO, we should have the network device movement between namespaces in
> order to be able to move a physical network device too (eg. you have 4
> NIC and you want to create 3 containers and assign 3 NIC to each of them)

Agree. Moving the devices is a must-have functionality.

I do not mind making the pair in the init namespace and move the second
one into the desired namespace. But if we *always* will have two ends in
different namespaces what to complicate things for?

Thanks,
Pavel

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: Re: [PATCH] Virtual ethernet tunnel
Posted by Daniel Lezcano on Thu, 07 Jun 2007 14:05:51 GMT
View Forum Message <> Reply to Message

Pavel Emelianov wrote:
>>> I did this at the very first version, but Alexey showed me that this
>>> would be wrong. Look. When we create the second device it must be in
>>> the other namespace as it is useless to have them in one namespace.
>>> But if we have the device in the other namespace the RTNL_NEWLINK
>>> message from kernel would come into this namespace thus confusing ip
>>> utility in the init namespace. Creating the device in the init ns and
>>> moving it into the new one is rather a complex task.
>>>
>>>
>> Pavel,
>>
>> moving the netdevice to another namespace is not a complex task. Eric
>> Biederman did it in its patchset (cf. http://lxc.sf.net/network)
>>
>
> By saying complex I didn't mean that this is difficult to implement,
> but that it consists (must consist) of many stages. I.e. composite.
> Making the device right in the namespace is liter.
>
>
>> When the pair device is created, both extremeties are into the init
>> namespace and you can choose to which namespace to move one extremity.
>>
>
> I do not mind that.
>
>> When the network namespace dies, the netdev is moved back to the init
>> namespace.
>> That facilitate network device management.
>>
>> Concerning netlink events, this is automatically generated when the
>> network device is moved through namespaces.
>>
>> IMHO, we should have the network device movement between namespaces in
>> order to be able to move a physical network device too (eg. you have 4
>> NIC and you want to create 3 containers and assign 3 NIC to each of them)
>>
>
> Agree. Moving the devices is a must-have functionality.
>
> I do not mind making the pair in the init namespace and move the second
> one into the desired namespace. But if we *always* will have two ends in
> different namespaces what to complicate things for?
>

Just to provide a netdev sufficiently generic to be used by people who
don't want namespaces but just want to do some network testing, like Ben
Greear does. He mentioned in a previous email, he will be happy to stop
redirecting people to out of tree patch.

https://lists.linux-foundation.org/pipermail/containers/2007-April/004420.html

> Thanks,
> Pavel
>
>

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH] Virtual ethernet tunnel
Posted by dev on Thu, 07 Jun 2007 14:23:06 GMT
View Forum Message <> Reply to Message

Deniel,

Daniel Lezcano wrote:
> Pavel Emelianov wrote:
>
>>>>I did this at the very first version, but Alexey showed me that this
>>>>would be wrong. Look. When we create the second device it must be in
>>>>the other namespace as it is useless to have them in one namespace.
>>>>But if we have the device in the other namespace the RTNL_NEWLINK
>>>>message from kernel would come into this namespace thus confusing ip
>>>>utility in the init namespace. Creating the device in the init ns and
>>>>moving it into the new one is rather a complex task.
>>>>
>>>>
>>>
>>>Pavel,
>>>
>>>moving the netdevice to another namespace is not a complex task. Eric
>>>Biederman did it in its patchset (cf. http://lxc.sf.net/network)
>>>
>>
>>By saying complex I didn't mean that this is difficult to implement,
>>but that it consists (must consist) of many stages. I.e. composite.
>>Making the device right in the namespace is liter.
>>
>>

>>
>>>When the pair device is created, both extremeties are into the init
>>>namespace and you can choose to which namespace to move one extremity.
>>>
>>
>>I do not mind that.
>>
>>
>>>When the network namespace dies, the netdev is moved back to the init
>>>namespace.
>>>That facilitate network device management.
>>>
>>>Concerning netlink events, this is automatically generated when the
>>>network device is moved through namespaces.
>>>
>>>IMHO, we should have the network device movement between namespaces in
>>>order to be able to move a physical network device too (eg. you have 4
>>>NIC and you want to create 3 containers and assign 3 NIC to each of them)
>>>
>>
>>Agree. Moving the devices is a must-have functionality.
>>
>>I do not mind making the pair in the init namespace and move the second
>>one into the desired namespace. But if we *always* will have two ends in
>>different namespaces what to complicate things for?
>>
>
> Just to provide a netdev sufficiently generic to be used by people who
> don't want namespaces but just want to do some network testing, like Ben
> Greear does. He mentioned in a previous email, he will be happy to stop
> redirecting people to out of tree patch.
>
> https://lists.linux-foundation.org/pipermail/containers/2007-April/004420.html

no one is against generic code and ability to create 2 interfaces in *one* namespace.
(Like we currently allow to do so in OpenVZ)

However, believe me, moving an interface is a *hard* operation. Much harder then netdev
register from the scratch.

Because it requires to take into account many things like:
- packets in flight which requires synchronize and is slow on big machines
- asynchronous sysfs entries registration/deregistration from
  rtln_unlock -> netdev_run_todo
- name/ifindex collisions
- shutdown/cleanup of addresses/routes/qdisc and other similar stuff

Thanks,

Kirill

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH] Virtual ethernet tunnel
Posted by Daniel Lezcano on Thu, 07 Jun 2007 14:42:09 GMT
View Forum Message <> Reply to Message

Kirill Korotaev wrote:
> Deniel,
>
> Daniel Lezcano wrote:
>
>> Pavel Emelianov wrote:
>>
>>
>>>>>> I did this at the very first version, but Alexey showed me that this
>>>>>> would be wrong. Look. When we create the second device it must be in
>>>>>> the other namespace as it is useless to have them in one namespace.
>>>>>> But if we have the device in the other namespace the RTNL_NEWLINK
>>>>>> message from kernel would come into this namespace thus confusing ip
>>>>>> utility in the init namespace. Creating the device in the init ns and
>>>>>> moving it into the new one is rather a complex task.
>>>>>
>>>>>
>>>>>
>>>> Pavel,
>>>>
>>>> moving the netdevice to another namespace is not a complex task. Eric
>>>> Biederman did it in its patchset (cf. http://lxc.sf.net/network)
>>>>
>>>>
>>> By saying complex I didn't mean that this is difficult to implement,
>>> but that it consists (must consist) of many stages. I.e. composite.
>>> Making the device right in the namespace is liter.
>>>
>>>
>>>
>>>
>> When the pair device is created, both extremeties are into the init
>> namespace and you can choose to which namespace to move one extremity.
>>
>>
> I do not mind that.

>>>
>>>
>>>
>>>> When the network namespace dies, the netdev is moved back to the init
>>>> namespace.
>>>> That facilitate network device management.
>>>>
>>>> Concerning netlink events, this is automatically generated when the
>>>> network device is moved through namespaces.
>>>>
>>>> IMHO, we should have the network device movement between namespaces in
>>>> order to be able to move a physical network device too (eg. you have 4
>>>> NIC and you want to create 3 containers and assign 3 NIC to each of them)
>>>>
>>>>
>>> Agree. Moving the devices is a must-have functionality.
>>>
>>> I do not mind making the pair in the init namespace and move the second
>>> one into the desired namespace. But if we *always* will have two ends in
>>> different namespaces what to complicate things for?
>>>
>>>
>> Just to provide a netdev sufficiently generic to be used by people who
>> don't want namespaces but just want to do some network testing, like Ben
>> Greear does. He mentioned in a previous email, he will be happy to stop
>> redirecting people to out of tree patch.
>>
>> https://lists.linux-foundation.org/pipermail/containers/2007-April/004420.html
>>
>
> no one is against generic code and ability to create 2 interfaces in *one* namespace.
> (Like we currently allow to do so in OpenVZ)
>
> However, believe me, moving an interface is a *hard* operation. Much harder then netdev
> register from the scratch.
>
> Because it requires to take into account many things like:
> - packets in flight which requires synchronize and is slow on big machines
> - asynchronous sysfs entries registration/deregistration from
>   rtln_unlock -> netdev_run_todo
> - name/ifindex collisions
> - shutdown/cleanup of addresses/routes/qdisc and other similar stuff
>
>
All of what you are describing is already implemented in the Eric's
patchset.

You can have a look at :

http://lxc.sourceforge.net/patches/2.6.20/2.6.20-netns1/broken_out/

And more precisly:

for sysfs issues:
http://lxc.sourceforge.net/patches/2.6.20/2.6.20-netns1/broken_out/0065-sysfs-Shadow-directory-support.patch

for network device movement:
http://lxc.sourceforge.net/patches/2.6.20/2.6.20-netns1/broken_out/0096-net-Implment-network-device-movement-between-namesp.patch

Thanks,
Daniel

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH] Virtual ethernet tunnel
Posted by Pavel Emelianov on Thu, 07 Jun 2007 15:25:59 GMT
View Forum Message <> Reply to Message

Daniel Lezcano wrote:
> Pavel Emelianov wrote:
>>>> I did this at the very first version, but Alexey showed me that this
>>>> would be wrong. Look. When we create the second device it must be in
>>>> the other namespace as it is useless to have them in one namespace.
>>>> But if we have the device in the other namespace the RTNL_NEWLINK
>>>> message from kernel would come into this namespace thus confusing ip
>>>> utility in the init namespace. Creating the device in the init ns and
>>>> moving it into the new one is rather a complex task.
>>>>
>>> Pavel,
>>>
>>> moving the netdevice to another namespace is not a complex task. Eric
>>> Biederman did it in its patchset (cf. http://lxc.sf.net/network)
>>>
>>
>> By saying complex I didn't mean that this is difficult to implement,
>> but that it consists (must consist) of many stages. I.e. composite.
>> Making the device right in the namespace is liter.
>>
>>
>>> When the pair device is created, both extremeties are into the init
>>> namespace and you can choose to which namespace to move one extremity.

>>>
>>
>> I do not mind that.
>>
>>> When the network namespace dies, the netdev is moved back to the init
>>> namespace.
>>> That facilitate network device management.
>>>
>>> Concerning netlink events, this is automatically generated when the
>>> network device is moved through namespaces.
>>>
>>> IMHO, we should have the network device movement between namespaces in
>>> order to be able to move a physical network device too (eg. you have 4
>>> NIC and you want to create 3 containers and assign 3 NIC to each of
>>> them)
>>>
>>
>> Agree. Moving the devices is a must-have functionality.
>>
>> I do not mind making the pair in the init namespace and move the second
>> one into the desired namespace. But if we *always* will have two ends in
>> different namespaces what to complicate things for?
>>
> Just to provide a netdev sufficiently generic to be used by people who
> don't want namespaces but just want to do some network testing, like Ben
> Greear does. He mentioned in a previous email, he will be happy to stop
> redirecting people to out of tree patch.

This patch creates booth devices in the init namespace. That's what
you want, isn't it? When we have the namespaces we will be able to
create the pair with booth ends in the init namespace - just do not
specify the namespace id to create the 2nd end in and the driver will
leave it int the init one.

> https://lists.linux-foundation.org/pipermail/containers/2007-April/004420.html
>
>
>> Thanks,
>> Pavel
>>
>>
>
> -
> To unsubscribe from this list: send the line "unsubscribe netdev" in
> the body of a message to majordomo@vger.kernel.org
> More majordomo info at http://vger.kernel.org/majordomo-info.html
>

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: Re: [PATCH] Virtual ethernet tunnel
Posted by Pavel Emelianov on Thu, 07 Jun 2007 15:33:26 GMT
View Forum Message <> Reply to Message

>>
>> no one is against generic code and ability to create 2 interfaces in
>> *one* namespace.
>> (Like we currently allow to do so in OpenVZ)
>>
>> However, believe me, moving an interface is a *hard* operation. Much
>> harder then netdev
>> register from the scratch.
>>
>> Because it requires to take into account many things like:
>> - packets in flight which requires synchronize and is slow on big
>> machines
>> - asynchronous sysfs entries registration/deregistration from
>> rtln_unlock -> netdev_run_todo
>> - name/ifindex collisions
>> - shutdown/cleanup of addresses/routes/qdisc and other similar stuff
>>
>>
> All of what you are describing is already implemented in the Eric's
> patchset.

Daniel. We *agree* that this task *is implementable*. We just want
to say that creating the device in the namespace is less comp...
(oh, well, forget this word) faster than creating and then moving.

> You can have a look at :
>
> http://lxc.sourceforge.net/patches/2.6.20/2.6.20-netns1/broken_out/
>
> And more precisely:
>
> for sysfs issues:
>
http://lxc.sourceforge.net/patches/2.6.20/2.6.20-netns1/broken_out/0065-sysfs-Shadow-directory-support.patch
>
>
> for network device movement:

>
http://lxc.sourceforge.net/patches/2.6.20/2.6.20-netns1/broken_out/0096-net-Implment-network-device-movement-between-namesp.patch

Good job. Can you prove that this code is less buggy than the existing
register_netdevice() one? This requires testing, doesn't it? But on the
other hand we have the ability to create the device right in the namespace
using well known (and thus well debugged) code with minimal actions from
the kernel.

>
> Thanks,
> Daniel
> -
> To unsubscribe from this list: send the line "unsubscribe netdev" in
> the body of a message to majordomo@vger.kernel.org
> More majordomo info at http://vger.kernel.org/majordomo-info.html
>

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH] Virtual ethernet tunnel
Posted by Daniel Lezcano on Thu, 07 Jun 2007 15:44:37 GMT
View Forum Message <> Reply to Message

Pavel Emelianov wrote:
> Daniel Lezcano wrote:
>> Pavel Emelianov wrote:
>>>> I did this at the very first version, but Alexey showed me that this
>>>> would be wrong. Look. When we create the second device it must be in
>>>> the other namespace as it is useless to have them in one namespace.
>>>> But if we have the device in the other namespace the RTNL_NEWLINK
>>>> message from kernel would come into this namespace thus confusing ip
>>>> utility in the init namespace. Creating the device in the init ns and
>>>> moving it into the new one is rather a complex task.
>>>>
>>>> Pavel,
>>>>
>>>> moving the netdevice to another namespace is not a complex task. Eric
>>>> Biederman did it in its patchset (cf. http://lxc.sf.net/network)
>>>>
>>> By saying complex I didn't mean that this is difficult to implement,
>>> but that it consists (must consist) of many stages. I.e. composite.
>>> Making the device right in the namespace is liter.

>>>
>>>
>>>> When the pair device is created, both extremeties are into the init
>>>> namespace and you can choose to which namespace to move one extremity.
>>>>
>>> I do not mind that.
>>>
>>>> When the network namespace dies, the netdev is moved back to the init
>>>> namespace.
>>>> That facilitate network device management.
>>>>
>>>> Concerning netlink events, this is automatically generated when the
>>>> network device is moved through namespaces.
>>>>
>>>> IMHO, we should have the network device movement between namespaces in
>>>> order to be able to move a physical network device too (eg. you have 4
>>>> NIC and you want to create 3 containers and assign 3 NIC to each of
>>>> them)
>>>>
>>> Agree. Moving the devices is a must-have functionality.
>>>
>>> I do not mind making the pair in the init namespace and move the second
>>> one into the desired namespace. But if we *always* will have two ends in
>>> different namespaces what to complicate things for?
>>>
>> Just to provide a netdev sufficiently generic to be used by people who
>> don't want namespaces but just want to do some network testing, like Ben
>> Greear does. He mentioned in a previous email, he will be happy to stop
>> redirecting people to out of tree patch.
>
> This patch creates booth devices in the init namespace. That's what
> you want, isn't it? When we have the namespaces we will be able to
> create the pair with booth ends in the init namespace - just do not
> specify the namespace id to create the 2nd end in and the driver will
> leave it int the init one.

Ok, fine.

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH] Virtual ethernet tunnel
Posted by Patrick McHardy on Wed, 13 Jun 2007 11:12:19 GMT
View Forum Message <> Reply to Message

Pavel Emelianov wrote:
> Patrick McHardy wrote:
>
>>The question is how to proceed. I haven't read all mails yet, but it
>>seems there is some disagreement about whether to create all devices
>>in the same namespace and move them later or create them directly in
>
>
> The agreement was that we can make any of the above. We can create
> booth devices in the init namespace and then move one of them into the
> desired namespace, or we can explicitly specify which namespace to create
> the pair in.

I'm going to push my latest patches to Dave today, the easiest way is
probably is you just add whatever you need to the API afterwards.

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH] Virtual ethernet tunnel
Posted by Patrick McHardy on Wed, 13 Jun 2007 15:37:20 GMT
View Forum Message <> Reply to Message

Pavel Emelianov wrote:
> Patrick McHardy wrote:
>
>>I'm going to push my latest patches to Dave today, the easiest way is
>>probably is you just add whatever you need to the API afterwards.
>>
>
>
> OK. Dave didn't object against the driver. Hope he will accept it as well.
>
> I have also found a BUG in your API. Look, when you declare the alias with
> the MODULE_ALIAS_RTNL_LINK in drivers you use strings as an argument. But
> this macro stringifyes the argument itself which results in bad aliases.
>
> Signed-off-by: Pavel Emelianov <xemul@openvz.org>
>
> ---
>
> diff --git a/include/net/rtnetlink.h b/include/net/rtnetlink.h
> index d744198..f627e1f 100644
> --- a/include/net/rtnetlink.h

> +++ b/include/net/rtnetlink.h
> @@ -77,6 +77,6 @@ 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);
>
> -#define MODULE_ALIAS_RTNL_LINK(name) MODULE_ALIAS("rtnl-link-" #name)
> +#define MODULE_ALIAS_RTNL_LINK(name) MODULE_ALIAS("rtnl-link-" name)

Thanks, I already fixed this during final round of testing :)

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH] Virtual ethernet tunnel
Posted by Pavel Emelianov on Wed, 13 Jun 2007 16:02:22 GMT
View Forum Message <> Reply to Message

Patrick McHardy wrote:
> Pavel Emelianov wrote:
>> Patrick McHardy wrote:
>>
>>> The question is how to proceed. I haven't read all mails yet, but it
>>> seems there is some disagreement about whether to create all devices
>>> in the same namespace and move them later or create them directly in
>>
>> The agreement was that we can make any of the above. We can create
>> booth devices in the init namespace and then move one of them into the
>> desired namespace, or we can explicitly specify which namespace to create
>> the pair in.
>
>
> I'm going to push my latest patches to Dave today, the easiest way is
> probably is you just add whatever you need to the API afterwards.
>
>

OK. Dave didn't object against the driver. Hope he will accept it as well.

I have also found a BUG in your API. Look, when you declare the alias with
the MODULE_ALIAS_RTNL_LINK in drivers you use strings as an argument. But
this macro stringifyes the argument itself which results in bad aliases.

Signed-off-by: Pavel Emelianov <xemul@openvz.org>

```
diff --git a/include/net/rtnetlink.h b/include/net/rtnetlink.h
index d744198..f627e1f 100644
--- a/include/net/rtnetlink.h
+++ b/include/net/rtnetlink.h
@@ -77,6 +77,6 @@ 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);

-#define MODULE_ALIAS_RTNL_LINK(name) MODULE_ALIAS("rtnl-link-" #name)
+#define MODULE_ALIAS_RTNL_LINK(name) MODULE_ALIAS("rtnl-link-" name)

 #endif
```

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers