Subject: [PATCH RFC 18/31] net: Implment network device movement between namespaces
Posted by ebiederm on Thu, 25 Jan 2007 19:00:20 GMT
View Forum Message <> Reply to Message

From: Eric W. Biederman <ebiederm@xmission.com> - unquoted

This patch introduces NETIF_F_NETNS_LOCAL a flag to indicate
a network device is local to a single network namespace and
should never be moved. Useful for pseudo devices that we
need an instance in each network namespace (like the loopback
device) and for any device we find that cannot handle multiple
network namespaces so we may trap them in the initial network
namespace.

This patch introduces the function dev_change_net_namespace
a function used to move a network device from one network
namespace to another. To the network device nothing
special appears to happen, to the components of the network
stack it appears as if the network device was unregistered
in the network namespace it is in, and a new device
was registered in the network namespace the device
was moved to.

This patch sets up a namespace device destructor that
upon the exit of a network namespace moves all of the
movable network devices to the initial network namespace
so they are not lost.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
drivers/net/loopback.c | 3 +-
include/linux/netdevice.h | 3 +
net/core/dev.c | 222 ++-----
3 files changed, 201 insertions(+), 27 deletions(-)

```
diff --git a/drivers/net/loopback.c b/drivers/net/loopback.c
index e9abf3f..7d15de0 100644
--- a/drivers/net/loopback.c
+++ b/drivers/net/loopback.c
@@ -225,7 +225,8 @@ DEFINE_PER_NET(struct net_device, loopback_dev) = {
       | NETIF_F_TSO
 #endif
       | NETIF_F_NO_CSUM | NETIF_F_HIGHDMA
-      | NETIF_F_LLTX,
+      | NETIF_F_LLTX
+      | NETIF_F_NETNS_LOCAL,
   .ethtool_ops  = &loopback_ethtool_ops,
```

```
 };

diff --git a/include/linux/netdevice.h b/include/linux/netdevice.h
index 0b4a4dc..3fcaf60 100644
--- a/include/linux/netdevice.h
+++ b/include/linux/netdevice.h
@@ -324,6 +324,7 @@ struct net_device
 #define NETIF_F_VLAN_CHALLENGED 1024 /* Device cannot handle VLAN packets */
 #define NETIF_F_GSO  2048 /* Enable software GSO. */
 #define NETIF_F_LLTX  4096 /* LockLess TX */
+#define NETIF_F_NETNS_LOCAL 8192 /* Does not change network namespaces */

  /* Segmentation offload features */
 #define NETIF_F_GSO_SHIFT 16
@@ -710,6 +711,8 @@ extern int  dev_ethtool(net_t net, struct ifreq *);
 extern unsigned  dev_get_flags(const struct net_device *);
 extern int  dev_change_flags(struct net_device *, unsigned);
 extern int  dev_change_name(struct net_device *, char *);
+extern int  dev_change_net_namespace(struct net_device *, net_t,
+       const char *);
 extern int  dev_set_mtu(struct net_device *, int);
 extern int  dev_set_mac_address(struct net_device *,
         struct sockaddr *);
diff --git a/net/core/dev.c b/net/core/dev.c
index fc0d2af..52994e4 100644
--- a/net/core/dev.c
+++ b/net/core/dev.c
@@ -198,6 +198,52 @@ static inline struct hlist_head *dev_index_hash(net_t net, int ifindex)
  return &per_net(dev_index_head, net)[ifindex & ((1<<NETDEV_HASHBITS)-1)];
 }

+/* Device list insertion */
+static int list_netdevice(struct net_device *dev)
+{
+ net_t net = dev->nd_net;
+
+ ASSERT_RTNL();
+
+ dev->next = NULL;
+ write_lock_bh(&per_net(dev_base_lock, net));
+ *per_net(dev_tail, net) = dev;
+ per_net(dev_tail, net) = &dev->next;
+ hlist_add_head(&dev->name_hlist, dev_name_hash(net, dev->name));
+ hlist_add_head(&dev->index_hlist, dev_index_hash(net, dev->ifindex));
+ write_unlock_bh(&per_net(dev_base_lock, net));
+ return 0;
+}
+
```

```
+/* Device list removal */
+static int unlist_netdevice(struct net_device *dev)
+{
+ struct net_device *d, **dp;
+ net_t net = dev->nd_net;
+
+ ASSERT_RTNL();
+
+ /* Unlink dev from the device chain */
+ for (dp = &per_net(dev_base, net); (d = *dp) != NULL; dp = &d->next) {
+  if (d == dev) {
+   write_lock_bh(&per_net(dev_base_lock, net));
+   hlist_del(&dev->name_hlist);
+   hlist_del(&dev->index_hlist);
+   if (per_net(dev_tail, net) == &dev->next)
+    per_net(dev_tail, net) = dp;
+   *dp = d->next;
+   write_unlock_bh(&per_net(dev_base_lock, net));
+   break;
+  }
+ }
+ if (!d) {
+  printk(KERN_ERR "unlist net_device: '%s' not found\n",
+         dev->name);
+  return -ENODEV;
+ }
+ return 0;
+}
+
 /*
  * Our notifier list
  */
@@ -3054,15 +3100,9 @@ int register_netdevice(struct net_device *dev)

  set_bit(__LINK_STATE_PRESENT, &dev->state);

- dev->next = NULL;
  dev_init_scheduler(dev);
- write_lock_bh(&per_net(dev_base_lock, net));
- *per_net(dev_tail, net) = dev;
- per_net(dev_tail, net) = &dev->next;
- hlist_add_head(&dev->name_hlist, head);
- hlist_add_head(&dev->index_hlist, dev_index_hash(net, dev->ifindex));
  dev_hold(dev);
- write_unlock_bh(&per_net(dev_base_lock, net));
+ list_netdevice(dev);

  /* Notify protocols, that a new device appeared. */
```

```
  raw_notifier_call_chain(&netdev_chain, NETDEV_REGISTER, dev);
@@ -3327,9 +3367,6 @@ void synchronize_net(void)

 int unregister_netdevice(struct net_device *dev)
 {
- struct net_device *d, **dp;
- net_t net = dev->nd_net;
-
  BUG_ON(dev_boot_phase);
  ASSERT_RTNL();

@@ -3347,23 +3384,8 @@ int unregister_netdevice(struct net_device *dev)
   dev_close(dev);

  /* And unlink it from device chain. */
- for (dp = &per_net(dev_base, net); (d = *dp) != NULL; dp = &d->next) {
-  if (d == dev) {
-   write_lock_bh(&per_net(dev_base_lock, net));
-   hlist_del(&dev->name_hlist);
-   hlist_del(&dev->index_hlist);
-   if (per_net(dev_tail, net) == &dev->next)
-    per_net(dev_tail, net) = dp;
-   *dp = d->next;
-   write_unlock_bh(&per_net(dev_base_lock, net));
-   break;
-  }
- }
- if (!d) {
-  printk(KERN_ERR "unregister net_device: '%s' not found\n",
-         dev->name);
+ if (unlist_netdevice(dev))
   return -ENODEV;
- }

  dev->reg_state = NETREG_UNREGISTERING;

@@ -3419,6 +3441,120 @@ void unregister_netdev(struct net_device *dev)

 EXPORT_SYMBOL(unregister_netdev);

+/**
+ * dev_change_net_namespace - move device to different nethost namespace
+ * @dev: device
+ * @net: network namespace
+ * @pat: If not NULL name pattern to try if the current device name
+ *       is already taken in the destination network namespace.
+ *
+ * This function shuts down a device interface and moves it
```

```
+ * to a new network namespace. On success 0 is returned, on
+ * a failure a netagive errno code is returned.
+ *
+ * Callers must hold the rtnl semaphore.
+ */
+
+int dev_change_net_namespace(struct net_device *dev, net_t net, const char *pat)
+{
+ char buf[IFNAMSIZ];
+ const char *destname;
+ int err;
+
+ ASSERT_RTNL();
+
+ /* Don't allow namespace local devices to be moved. */
+ err = -EINVAL;
+ if (dev->features & NETIF_F_NETNS_LOCAL)
+  goto out;
+
+ /* Ensure the device has been registrered */
+ err = -EINVAL;
+ if (dev->reg_state != NETREG_REGISTERED)
+  goto out;
+
+ /* Get out if there is nothing todo */
+ err = 0;
+ if (net_eq(dev->nd_net, net))
+  goto out;
+
+ /* Pick the destination device name, and ensure
+  * we can use it in the destination network namespace.
+  */
+ err = -EEXIST;
+ destname = dev->name;
+ if (__dev_get_by_name(net, destname) && pat) {
+  /* We get here if we can't use the current device name */
+  if (!dev_valid_name(pat))
+   goto out;
+  if (strchr(pat, '%')) {
+   if (__dev_alloc_name(net, pat, buf) < 0)
+    goto out;
+   destname = buf;
+  } else
+   destname = pat;
+  if (__dev_get_by_name(net, destname))
+   goto out;
+ }
+
```

```
+ /*
+  * And now a mini version of register_netdevice unregister_netdevice.
+  */
+
+ /* If device is running close it first. */
+ if (dev->flags & IFF_UP)
+  dev_close(dev);
+
+ /* And unlink it from device chain */
+ err = -ENODEV;
+ if (unlist_netdevice(dev))
+  goto out;
+
+ synchronize_net();
+
+ /* Shutdown queueing discipline. */
+ dev_shutdown(dev);
+
+ /* Notify protocols, that we are about to destroy
+    this device. They should clean all the things.
+ */
+ call_netdevice_notifiers(NETDEV_UNREGISTER, dev);
+
+ /*
+  * Flush the multicast chain
+  */
+ dev_mc_discard(dev);
+
+ /* Actually switch the network namespace */
+ dev->nd_net = net;
+
+ /* Assign the new device name */
+ if (destname != dev->name)
+  strcpy(dev->name, destname);
+
+ /* If there is an ifindex conflict assign a new one */
+ if (__dev_get_by_index(net, dev->ifindex)) {
+  int iflink = (dev->iflink == dev->ifindex);
+  dev->ifindex = dev_new_index(net);
+  if (iflink)
+   dev->iflink = dev->ifindex;
+ }
+
+ /* Fixup sysfs */
+ class_device_rename(&dev->class_dev, dev->name);
+
+ /* Add the device back in the hashes */
+ list_netdevice(dev);
```

```
+
+ /* Notify protocols, that a new device appeared. */
+ call_netdevice_notifiers(NETDEV_REGISTER, dev);
+
+ synchronize_net();
+ err = 0;
+out:
+ return err;
+}
+
 static int dev_cpu_callback(struct notifier_block *nfb,
        unsigned long action,
        void *ocpu)
@@ -3561,6 +3697,37 @@ static struct pernet_operations netdev_net_ops = {
  .init = netdev_init,
 };

+static void default_device_exit(net_t net)
+{
+ struct net_device *dev, *next;
+ /*
+  * Push all migratable of the network devices back to the
+  * initial network namespace
+  */
+ rtnl_lock();
+ for (dev = per_net(dev_base, net); dev; dev = next) {
+  int err;
+  next = dev->next;
+
+  /* Ignore unmoveable devices (i.e. loopback) */
+  if (dev->features & NETIF_F_NETNS_LOCAL)
+   continue;
+
+  /* Push remaing network devices to init_net */
+  err = dev_change_net_namespace(dev, init_net(), "dev%d");
+  if (err) {
+   printk(KERN_WARNING "%s: failed to move %s to init_net: %d\n",
+    __func__, dev->name, err);
+   unregister_netdevice(dev);
+  }
+ }
+ rtnl_unlock();
+}
+
+static struct pernet_operations default_device_ops = {
+ .exit = default_device_exit,
+};
+
```

```
 /*
  * Initialize the DEV module. At boot time this walks the device list and
  * unhooks any devices that fail to initialise (normally hardware not
@@ -3591,6 +3758,9 @@ static int __init net_dev_init(void)
  if (register_pernet_subsys(&netdev_net_ops))
   goto out;

+ if (register_pernet_device(&default_device_ops))
+  goto out;
+
  /*
   * Initialise the packet receive queues.
   */
--
1.4.4.1.g278f
```

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers