Subject: [PATCH RFC 15/31] net: Make the loopback device per network namespace
Posted by ebiederm on Thu, 25 Jan 2007 19:00:17 GMT
View Forum Message <> Reply to Message

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

This patch makes the loopback_dev per network namespace.
The loopback device registers itself as a pernet_device so
we can register the new loopback_dev instance when we add
a new network namespace and so we can unregister the
loopback device when we destory the network namespace.

Currently the loopback device statitics are kept accross
all loopback devices, a minor glitch that will not affect
correct operation but something we may want to fix.

This patch modifies all users the loopback_dev so they
access it as per_net(loopback_dev, init_net()), keeping all of the
code compiling and working. A later pass will be needed to
update the users to use something other than the initial network
namespace.

The only non-trivial modification was the ipv6 code in route.c as the
loopback_dev can no longer be used in static initializers, and
even that change was very simple.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
drivers/net/loopback.c | 24 ++++++++++++++++++++----
include/linux/netdevice.h | 2 +-
net/core/dst.c | 8 ++++----
net/decnet/dn_dev.c | 4 ++--
net/decnet/dn_route.c | 14 +++++++-------
net/ipv4/devinet.c | 4 ++--
net/ipv4/ipconfig.c | 8 ++++----
net/ipv4/ipvs/ip_vs_core.c | 2 +-
net/ipv4/route.c | 18 +++++++++---------
net/ipv4/xfrm4_policy.c | 2 +-
net/ipv6/addrconf.c | 8 ++++----
net/ipv6/netfilter/ip6t_REJECT.c | 2 +-
net/ipv6/route.c | 24 ++++++++++++++++++++----
net/ipv6/xfrm6_policy.c | 2 +-
net/xfrm/xfrm_policy.c | 4 ++--
15 files changed, 75 insertions(+), 51 deletions(-)

diff --git a/drivers/net/loopback.c b/drivers/net/loopback.c
index 22b672d..e9abf3f 100644

```
--- a/drivers/net/loopback.c
+++ b/drivers/net/loopback.c
@@ -57,6 +57,7 @@
 #include <linux/ip.h>
 #include <linux/tcp.h>
 #include <linux/percpu.h>
+#include <net/net_namespace.h>

 struct pcpu_lstats {
  unsigned long packets;
@@ -204,7 +205,7 @@ static const struct ethtool_ops loopback_ethtool_ops = {
  * The loopback device is special. There is only one instance and
  * it is statically allocated. Don't do this for other devices.
  */
-struct net_device loopback_dev = {
+DEFINE_PER_NET(struct net_device, loopback_dev) = {
  .name    = "lo",
  .get_stats  = &get_stats,
  .priv   = &loopback_stats,
@@ -228,13 +229,28 @@ struct net_device loopback_dev = {
  .ethtool_ops  = &loopback_ethtool_ops,
 };

+static int loopback_net_init(net_t net)
+{
+ per_net(loopback_dev, net).nd_net = net;
+ return register_netdev(&per_net(loopback_dev, net));
+}
+
+static void loopback_net_exit(net_t net)
+{
+ unregister_netdev(&per_net(loopback_dev, net));
+}
+
+static struct pernet_operations loopback_net_ops = {
+ .init = loopback_net_init,
+ .exit = loopback_net_exit,
+};
+
 /* Setup and register the loopback device. */
 static int __init loopback_init(void)
 {
- loopback_dev.nd_net = init_net();
- return register_netdev(&loopback_dev);
+ return register_pernet_device(&loopback_net_ops);
 };

 module_init(loopback_init);
```

```
-EXPORT_SYMBOL(loopback_dev);
+EXPORT_PER_NET_SYMBOL(loopback_dev);
diff --git a/include/linux/netdevice.h b/include/linux/netdevice.h
index 9e28671..73931a0 100644
--- a/include/linux/netdevice.h
+++ b/include/linux/netdevice.h
@@ -570,7 +570,7 @@ struct packet_type {
 #include <linux/interrupt.h>
 #include <linux/notifier.h>

-extern struct net_device  loopback_dev;  /* The loopback */
+DECLARE_PER_NET(struct net_device, loopback_dev);  /* The loopback */
 extern struct net_device  *dev_base;  /* All devices */
 extern rwlock_t    dev_base_lock;  /* Device list lock */

diff --git a/net/core/dst.c b/net/core/dst.c
index 8c4a272..3435771 100644
--- a/net/core/dst.c
+++ b/net/core/dst.c
@@ -241,13 +241,13 @@ static inline void dst_ifdown(struct dst_entry *dst, struct net_device
*dev,
   dst->input = dst_discard_in;
   dst->output = dst_discard_out;
  } else {
-  dst->dev = &loopback_dev;
-  dev_hold(&loopback_dev);
+  dst->dev = &per_net(loopback_dev, init_net());
+  dev_hold(dst->dev);
   dev_put(dev);
   if (dst->neighbour && dst->neighbour->dev == dev) {
-   dst->neighbour->dev = &loopback_dev;
+   dst->neighbour->dev = &per_net(loopback_dev, init_net());
    dev_put(dev);
-   dev_hold(&loopback_dev);
+   dev_hold(dst->neighbour->dev);
   }
  }
 }
diff --git a/net/decnet/dn_dev.c b/net/decnet/dn_dev.c
index 19b1469..dbaf001 100644
--- a/net/decnet/dn_dev.c
+++ b/net/decnet/dn_dev.c
@@ -866,10 +866,10 @@ last_chance:
   rv = dn_dev_get_first(dev, addr);
   read_unlock(&dev_base_lock);
   dev_put(dev);
-  if (rv == 0 || dev == &loopback_dev)
```

```
+  if (rv == 0 || dev == &per_net(loopback_dev, init_net()))
    return rv;
  }
- dev = &loopback_dev;
+ dev = &per_net(loopback_dev, init_net());
  dev_hold(dev);
  goto last_chance;
 }
diff --git a/net/decnet/dn_route.c b/net/decnet/dn_route.c
index 4263cd9..b553cd4 100644
--- a/net/decnet/dn_route.c
+++ b/net/decnet/dn_route.c
@@ -887,7 +887,7 @@ static int dn_route_output_slow(struct dst_entry **pprt, const struct flowi
*old
      .scope = RT_SCOPE_UNIVERSE,
         } },
        .mark = oldflp->mark,
-       .iif = loopback_dev.ifindex,
+       .iif = per_net(loopback_dev, init_net()).ifindex,
        .oif = oldflp->oif };
  struct dn_route *rt = NULL;
  struct net_device *dev_out = NULL;
@@ -904,7 +904,7 @@ static int dn_route_output_slow(struct dst_entry **pprt, const struct flowi
*old
          "dn_route_output_slow: dst=%04x src=%04x mark=%d"
          " iif=%d oif=%d\n", dn_ntohs(oldflp->fld_dst),
          dn_ntohs(oldflp->fld_src),
-                    oldflp->mark, loopback_dev.ifindex, oldflp->oif);
+                     oldflp->mark, per_net(loopback_dev, init_net()).ifindex, oldflp->oif);

  /* If we have an output interface, verify its a DECnet device */
  if (oldflp->oif) {
@@ -955,7 +955,7 @@ source_ok:
   err = -EADDRNOTAVAIL;
   if (dev_out)
    dev_put(dev_out);
-  dev_out = &loopback_dev;
+  dev_out = &per_net(loopback_dev, init_net());
   dev_hold(dev_out);
   if (!fl.fld_dst) {
    fl.fld_dst =
@@ -964,7 +964,7 @@ source_ok:
    if (!fl.fld_dst)
     goto out;
   }
-  fl.oif = loopback_dev.ifindex;
+  fl.oif = per_net(loopback_dev, init_net()).ifindex;
   res.type = RTN_LOCAL;
```

```
   goto make_route;
  }
@@ -1010,7 +1010,7 @@ source_ok:
       if (dev_out)
        dev_put(dev_out);
       if (dn_dev_islocal(neigh->dev, fl.fld_dst)) {
-       dev_out = &loopback_dev;
+       dev_out = &per_net(loopback_dev, init_net());
        res.type = RTN_LOCAL;
       } else {
        dev_out = neigh->dev;
@@ -1031,7 +1031,7 @@ source_ok:
    /* Possible improvement - check all devices for local addr */
    if (dn_dev_islocal(dev_out, fl.fld_dst)) {
     dev_put(dev_out);
-    dev_out = &loopback_dev;
+    dev_out = &per_net(loopback_dev, init_net());
     dev_hold(dev_out);
     res.type = RTN_LOCAL;
     goto select_source;
@@ -1067,7 +1067,7 @@ select_source:
     fl.fld_src = fl.fld_dst;
    if (dev_out)
     dev_put(dev_out);
-   dev_out = &loopback_dev;
+   dev_out = &per_net(loopback_dev, init_net());
    dev_hold(dev_out);
    fl.oif = dev_out->ifindex;
    if (res.fi)
diff --git a/net/ipv4/devinet.c b/net/ipv4/devinet.c
index a7d991d..201442c 100644
--- a/net/ipv4/devinet.c
+++ b/net/ipv4/devinet.c
@@ -1056,7 +1056,7 @@ static int inetdev_event(struct notifier_block *this, unsigned long event,
  ASSERT_RTNL();

  if (!in_dev) {
-  if (event == NETDEV_REGISTER && dev == &loopback_dev) {
+  if (event == NETDEV_REGISTER && dev == &per_net(loopback_dev, init_net())) {
     in_dev = inetdev_init(dev);
     if (!in_dev)
      panic("devinet: Failed to create loopback\n");
@@ -1074,7 +1074,7 @@ static int inetdev_event(struct notifier_block *this, unsigned long event,
   case NETDEV_UP:
    if (dev->mtu < 68)
     break;
-   if (dev == &loopback_dev) {
+   if (dev == &per_net(loopback_dev, init_net())) {
```

```
    struct in_ifaddr *ifa;
    if ((ifa = inet_alloc_ifa()) != NULL) {
     ifa->ifa_local =
diff --git a/net/ipv4/ipconfig.c b/net/ipv4/ipconfig.c
index 91b5729..ee77938 100644
--- a/net/ipv4/ipconfig.c
+++ b/net/ipv4/ipconfig.c
@@ -185,16 +185,18 @@ static int __init ic_open_devs(void)
  struct ic_device *d, **last;
  struct net_device *dev;
  unsigned short oflags;
+ struct net_device *lo;

  last = &ic_first_dev;
  rtnl_lock();

  /* bring loopback device up first */
- if (dev_change_flags(&loopback_dev, loopback_dev.flags | IFF_UP) < 0)
-  printk(KERN_ERR "IP-Config: Failed to open %s\n", loopback_dev.name);
+ lo = &per_net(loopback_dev, init_net());
+ if (dev_change_flags(lo, lo->flags | IFF_UP) < 0)
+  printk(KERN_ERR "IP-Config: Failed to open %s\n", lo->name);

  for (dev = dev_base; dev; dev = dev->next) {
-  if (dev == &loopback_dev)
+  if (dev == lo)
     continue;
    if (user_dev_name[0] ? !strcmp(dev->name, user_dev_name) :
        (!(dev->flags & IFF_LOOPBACK) &&
diff --git a/net/ipv4/ipvs/ip_vs_core.c b/net/ipv4/ipvs/ip_vs_core.c
index 3425752..2e1e41f 100644
--- a/net/ipv4/ipvs/ip_vs_core.c
+++ b/net/ipv4/ipvs/ip_vs_core.c
@@ -963,7 +963,7 @@ ip_vs_in(unsigned int hooknum, struct sk_buff **pskb,
   * ... don't know why 1st test DOES NOT include 2nd (?)
   */
  if (unlikely(skb->pkt_type != PACKET_HOST
-       || skb->dev == &loopback_dev || skb->sk)) {
+       || skb->dev == &per_net(loopback_dev, init_net()) || skb->sk)) {
   IP_VS_DBG(12, "packet type=%d proto=%d daddr=%d.%d.%d.%d ignored\n",
      skb->pkt_type,
      skb->nh.iph->protocol,
diff --git a/net/ipv4/route.c b/net/ipv4/route.c
index 8be7506..d23a0d7 100644
--- a/net/ipv4/route.c
+++ b/net/ipv4/route.c
@@ -1498,8 +1498,8 @@ static void ipv4_dst_ifdown(struct dst_entry *dst, struct net_device
*dev,
```

```
 {
  struct rtable *rt = (struct rtable *) dst;
  struct in_device *idev = rt->idev;
- if (dev != &loopback_dev && idev && idev->dev == dev) {
-  struct in_device *loopback_idev = in_dev_get(&loopback_dev);
+ if (dev != &per_net(loopback_dev, init_net()) && idev && idev->dev == dev) {
+  struct in_device *loopback_idev = in_dev_get(&per_net(loopback_dev, init_net()));
   if (loopback_idev) {
    rt->idev = loopback_idev;
    in_dev_put(idev);
@@ -1651,7 +1651,7 @@ static int ip_route_input_mc(struct sk_buff *skb, __be32 daddr,
__be32 saddr,
 #endif
  rth->rt_iif =
  rth->fl.iif = dev->ifindex;
- rth->u.dst.dev = &loopback_dev;
+ rth->u.dst.dev = &per_net(loopback_dev, init_net());
  dev_hold(rth->u.dst.dev);
  rth->idev = in_dev_get(rth->u.dst.dev);
  rth->fl.oif = 0;
@@ -1969,7 +1969,7 @@ static int ip_route_input_slow(struct sk_buff *skb, __be32 daddr,
__be32 saddr,
  if (res.type == RTN_LOCAL) {
   int result;
   result = fib_validate_source(saddr, daddr, tos,
-          loopback_dev.ifindex,
+          per_net(loopback_dev, init_net()).ifindex,
           dev, &spec_dst, &itag);
   if (result < 0)
    goto martian_source;
@@ -2036,7 +2036,7 @@ local_input:
 #endif
  rth->rt_iif =
  rth->fl.iif = dev->ifindex;
- rth->u.dst.dev = &loopback_dev;
+ rth->u.dst.dev = &per_net(loopback_dev, init_net());
  dev_hold(rth->u.dst.dev);
  rth->idev = in_dev_get(rth->u.dst.dev);
  rth->rt_gateway = daddr;
@@ -2375,7 +2375,7 @@ static int ip_route_output_slow(struct rtable **rp, const struct flowi
*oldflp)
         RT_SCOPE_UNIVERSE),
           } },
         .mark = oldflp->mark,
-        .iif = loopback_dev.ifindex,
+        .iif = per_net(loopback_dev, init_net()).ifindex,
         .oif = oldflp->oif };
  struct fib_result res;
```

```
  unsigned flags = 0;
@@ -2469,9 +2469,9 @@ static int ip_route_output_slow(struct rtable **rp, const struct flowi
*oldflp)
     fl.fl4_dst = fl.fl4_src = htonl(INADDR_LOOPBACK);
    if (dev_out)
     dev_put(dev_out);
-   dev_out = &loopback_dev;
+   dev_out = &per_net(loopback_dev, init_net());
    dev_hold(dev_out);
-   fl.oif = loopback_dev.ifindex;
+   fl.oif = per_net(loopback_dev, init_net()).ifindex;
    res.type = RTN_LOCAL;
    flags |= RTCF_LOCAL;
    goto make_route;
@@ -2516,7 +2516,7 @@ static int ip_route_output_slow(struct rtable **rp, const struct flowi
*oldflp)
     fl.fl4_src = fl.fl4_dst;
    if (dev_out)
     dev_put(dev_out);
-   dev_out = &loopback_dev;
+   dev_out = &per_net(loopback_dev, init_net());
    dev_hold(dev_out);
    fl.oif = dev_out->ifindex;
    if (res.fi)
diff --git a/net/ipv4/xfrm4_policy.c b/net/ipv4/xfrm4_policy.c
index fb9f69c..39a0ba2 100644
--- a/net/ipv4/xfrm4_policy.c
+++ b/net/ipv4/xfrm4_policy.c
@@ -289,7 +289,7 @@ static void xfrm4_dst_ifdown(struct dst_entry *dst, struct net_device
*dev,

   xdst = (struct xfrm_dst *)dst;
   if (xdst->u.rt.idev->dev == dev) {
-   struct in_device *loopback_idev = in_dev_get(&loopback_dev);
+   struct in_device *loopback_idev = in_dev_get(&per_net(loopback_dev, init_net()));
    BUG_ON(!loopback_idev);

    do {
diff --git a/net/ipv6/addrconf.c b/net/ipv6/addrconf.c
index 7be542f..c9fa27a 100644
--- a/net/ipv6/addrconf.c
+++ b/net/ipv6/addrconf.c
@@ -2365,7 +2365,7 @@ static int addrconf_ifdown(struct net_device *dev, int how)

   ASSERT_RTNL();

- if (dev == &loopback_dev && how == 1)
+ if (dev == &per_net(loopback_dev, init_net()) && how == 1)
```

```
   how = 0;

  rt6_ifdown(dev);
@@ -4074,13 +4074,13 @@ int __init addrconf_init(void)
   * device and it being up should be removed.
   */
  rtnl_lock();
- if (!ipv6_add_dev(&loopback_dev))
+ if (!ipv6_add_dev(&per_net(loopback_dev, init_net())))
   err = -ENOMEM;
  rtnl_unlock();
  if (err)
   return err;

- ip6_null_entry.rt6i_idev = in6_dev_get(&loopback_dev);
+ ip6_null_entry.rt6i_idev = in6_dev_get(&per_net(loopback_dev, init_net()));

  register_netdevice_notifier(&ipv6_dev_notf);

@@ -4121,7 +4121,7 @@ void __exit addrconf_cleanup(void)
    continue;
   addrconf_ifdown(dev, 1);
  }
- addrconf_ifdown(&loopback_dev, 2);
+ addrconf_ifdown(&per_net(loopback_dev, init_net()), 2);

  /*
   * Check hash table.
diff --git a/net/ipv6/netfilter/ip6t_REJECT.c b/net/ipv6/netfilter/ip6t_REJECT.c
index 311eae8..a80bbee 100644
--- a/net/ipv6/netfilter/ip6t_REJECT.c
+++ b/net/ipv6/netfilter/ip6t_REJECT.c
@@ -170,7 +170,7 @@ static inline void
 send_unreach(struct sk_buff *skb_in, unsigned char code, unsigned int hooknum)
 {
  if (hooknum == NF_IP6_LOCAL_OUT && skb_in->dev == NULL)
-  skb_in->dev = &loopback_dev;
+  skb_in->dev = &per_net(loopback_dev, init_net());

  icmpv6_send(skb_in, ICMPV6_DEST_UNREACH, code, 0, NULL);
 }
diff --git a/net/ipv6/route.c b/net/ipv6/route.c
index 8c9fef9..6805c39 100644
--- a/net/ipv6/route.c
+++ b/net/ipv6/route.c
@@ -125,7 +125,7 @@ struct rt6_info ip6_null_entry = {
   .dst = {
    .__refcnt = ATOMIC_INIT(1),
```

```
    .__use  = 1,
-   .dev  = &loopback_dev,
+   .dev  = NULL,
    .obsolete = -1,
    .error  = -ENETUNREACH,
    .metrics = { [RTAX_HOPLIMIT - 1] = 255, },
@@ -151,7 +151,7 @@ struct rt6_info ip6_prohibit_entry = {
   .dst = {
    .__refcnt = ATOMIC_INIT(1),
    .__use  = 1,
-   .dev  = &loopback_dev,
+   .dev  = NULL,
    .obsolete = -1,
    .error  = -EACCES,
    .metrics = { [RTAX_HOPLIMIT - 1] = 255, },
@@ -171,7 +171,7 @@ struct rt6_info ip6_blk_hole_entry = {
   .dst = {
    .__refcnt = ATOMIC_INIT(1),
    .__use  = 1,
-   .dev  = &loopback_dev,
+   .dev  = NULL,
    .obsolete = -1,
    .error  = -EINVAL,
    .metrics = { [RTAX_HOPLIMIT - 1] = 255, },
@@ -211,8 +211,8 @@ static void ip6_dst_ifdown(struct dst_entry *dst, struct net_device *dev,
  struct rt6_info *rt = (struct rt6_info *)dst;
  struct inet6_dev *idev = rt->rt6i_idev;

- if (dev != &loopback_dev && idev != NULL && idev->dev == dev) {
-  struct inet6_dev *loopback_idev = in6_dev_get(&loopback_dev);
+ if (dev != &per_net(loopback_dev, init_net()) && idev != NULL && idev->dev == dev) {
+  struct inet6_dev *loopback_idev = in6_dev_get(&per_net(loopback_dev, init_net()));
   if (loopback_idev != NULL) {
    rt->rt6i_idev = loopback_idev;
    in6_dev_put(idev);
@@ -1103,12 +1103,12 @@ int ip6_route_add(struct fib6_config *cfg)
  if ((cfg->fc_flags & RTF_REJECT) ||
      (dev && (dev->flags&IFF_LOOPBACK) && !(addr_type&IPV6_ADDR_LOOPBACK))) {
   /* hold loopback dev/idev if we haven't done so. */
-  if (dev != &loopback_dev) {
+  if (dev != &per_net(loopback_dev, init_net())) {
    if (dev) {
     dev_put(dev);
     in6_dev_put(idev);
    }
-   dev = &loopback_dev;
+   dev = &per_net(loopback_dev, init_net());
    dev_hold(dev);
```

```
    idev = in6_dev_get(dev);
    if (!idev) {
@@ -1803,13 +1803,13 @@ struct rt6_info *addrconf_dst_alloc(struct inet6_dev *idev,
  if (rt == NULL)
   return ERR_PTR(-ENOMEM);

- dev_hold(&loopback_dev);
+ dev_hold(&per_net(loopback_dev, init_net()));
  in6_dev_hold(idev);

  rt->u.dst.flags = DST_HOST;
  rt->u.dst.input = ip6_input;
  rt->u.dst.output = ip6_output;
- rt->rt6i_dev = &loopback_dev;
+ rt->rt6i_dev = &per_net(loopback_dev, init_net());
  rt->rt6i_idev = idev;
  rt->u.dst.metrics[RTAX_MTU-1] = ipv6_get_mtu(rt->rt6i_dev);
  rt->u.dst.metrics[RTAX_ADVMSS-1] = ipv6_advmss(dst_mtu(&rt->u.dst));
@@ -2457,6 +2457,12 @@ void __init ip6_route_init(void)
  ip6_dst_ops.kmem_cachep =
   kmem_cache_create("ip6_dst_cache", sizeof(struct rt6_info), 0,
       SLAB_HWCACHE_ALIGN|SLAB_PANIC, NULL, NULL);
+ /* Perform the initialization we can't perform at compile time */
+ ip6_null_entry.u.dst.dev = &per_net(loopback_dev, init_net());
+#ifdef CONFIG_IPV6_MULTIPLE_TABLES
+ ip6_prohibit_entry.u.dst.dev = &per_net(loopback_dev, init_net());
+ ip6_blk_hole_entry.u.dst.dev = &per_net(loopback_dev, init_net());
+#endif
  fib6_init();
 #ifdef  CONFIG_PROC_FS
  p = proc_net_create(init_net(), "ipv6_route", 0, rt6_proc_info);
diff --git a/net/ipv6/xfrm6_policy.c b/net/ipv6/xfrm6_policy.c
index 8dffd4d..2608c75 100644
--- a/net/ipv6/xfrm6_policy.c
+++ b/net/ipv6/xfrm6_policy.c
@@ -354,7 +354,7 @@ static void xfrm6_dst_ifdown(struct dst_entry *dst, struct net_device
*dev,

  xdst = (struct xfrm_dst *)dst;
  if (xdst->u.rt6.rt6i_idev->dev == dev) {
-  struct inet6_dev *loopback_idev = in6_dev_get(&loopback_dev);
+  struct inet6_dev *loopback_idev = in6_dev_get(&per_net(loopback_dev, init_net()));
   BUG_ON(!loopback_idev);

   do {
diff --git a/net/xfrm/xfrm_policy.c b/net/xfrm/xfrm_policy.c
index 0248343..51ab8ac 100644
--- a/net/xfrm/xfrm_policy.c
```

```
+++ b/net/xfrm/xfrm_policy.c
@@ -1799,8 +1799,8 @@ static int stale_bundle(struct dst_entry *dst)
 void xfrm_dst_ifdown(struct dst_entry *dst, struct net_device *dev)
 {
  while ((dst = dst->child) && dst->xfrm && dst->dev == dev) {
-  dst->dev = &loopback_dev;
-  dev_hold(&loopback_dev);
+  dst->dev = &per_net(loopback_dev, init_net());
+  dev_hold(dst->dev);
   dev_put(dev);
  }
 }
--
1.4.4.1.g278f
```