
Subject: [net-2.6.24][patch 0/2] Dynamically allocate the loopback
Posted by [Daniel Lezcano](#) on Mon, 17 Sep 2007 13:45:09 GMT
[View Forum Message](#) <> [Reply to Message](#)

This patch allows to dynamically allocate the loopback like an usual network device.

This global static variable `loopback_dev` has been replaced by a `netdev` pointer and the init function does the usual allocation, initialization and registering of the loopback.

This patchset is splitted in two parts, the first one is a big but trivial patch which replace the usage of the static variable `loopback_dev` by the usage of a pointer. The second patch is the interesting part where the loopback is dynamically allocated.

--

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

Subject: [net-2.6.24][patch 1/2] Dynamically allocate the loopback device - mindless changes
Posted by [Daniel Lezcano](#) on Mon, 17 Sep 2007 13:45:10 GMT
[View Forum Message](#) <> [Reply to Message](#)

From: Daniel Lezcano <dlezcano@fr.ibm.com>

This patch replaces all occurrences to the static variable `loopback_dev` to a pointer `loopback_dev`. That provides the mindless, trivial, uninteresting change part for the dynamic allocation for the loopback.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>

Acked-By: Kirill Korotaev <dev@sw.ru>

Acked-by: Benjamin Thery <benjamin.thery@bull.net>

<code>drivers/net/loopback.c</code>		6 +++++--
<code>include/linux/netdevice.h</code>		2 +-
<code>net/core/dst.c</code>		8 +++++----
<code>net/decnet/dn_dev.c</code>		4 ++--
<code>net/decnet/dn_route.c</code>		14 ++++++-----
<code>net/ipv4/devinet.c</code>		6 +++---
<code>net/ipv4/ipconfig.c</code>		6 +++---
<code>net/ipv4/ipvs/ip_vs_core.c</code>		2 +-

```

net/ipv4/route.c      | 18 ++++++-----
net/ipv4/xfrm4_policy.c | 2 +-
net/ipv6/addrconf.c   | 15 ++++++-----
net/ipv6/ip6_input.c  | 2 +-
net/ipv6/netfilter/ip6t_REJECT.c | 2 +-
net/ipv6/route.c      | 15 ++++++-----
net/ipv6/xfrm6_policy.c | 2 +-
net/xfrm/xfrm_policy.c | 4 ++--
16 files changed, 55 insertions(+), 53 deletions(-)

```

Index: net-2.6.24/drivers/net/loopback.c

```

=====
--- net-2.6.24.orig/drivers/net/loopback.c
+++ net-2.6.24/drivers/net/loopback.c
@@ -202,7 +202,7 @@ static const struct ethtool_ops loopback
 * 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 = {
+struct net_device __loopback_dev = {
    .name    = "lo",
    .get_stats = &get_stats,
    .mtu     = (16 * 1024) + 20 + 20 + 12,
@@ -227,10 +227,12 @@ struct net_device loopback_dev = {
    .nd_net      = &init_net,
};

+struct net_device *loopback_dev = &__loopback_dev;
+
/* Setup and register the loopback device. */
static int __init loopback_init(void)
{
- int err = register_netdev(&loopback_dev);
+ int err = register_netdev(loopback_dev);

    if (err)
        panic("loopback: Failed to register netdevice: %d\n", err);

```

Index: net-2.6.24/include/linux/netdevice.h

```

=====
--- net-2.6.24.orig/include/linux/netdevice.h
+++ net-2.6.24/include/linux/netdevice.h
@@ -742,7 +742,7 @@ struct packet_type {
#include <linux/interrupt.h>
#include <linux/notifier.h>

-extern struct net_device loopback_dev; /* The loopback */
+extern struct net_device *loopback_dev; /* The loopback */
extern rwlock_t dev_base_lock; /* Device list lock */

```

Index: net-2.6.24/net/core/dst.c

```
=====
--- net-2.6.24.orig/net/core/dst.c
+++ net-2.6.24/net/core/dst.c
@@ -278,13 +278,13 @@ static inline void dst_ifdown(struct dst
     if (!unregister) {
         dst->input = dst->output = dst_discard;
     } else {
- dst->dev = &loopback_dev;
- dev_hold(&loopback_dev);
+ dst->dev = loopback_dev;
+ dev_hold(dst->dev);
     dev_put(dev);
     if (dst->neighbour && dst->neighbour->dev == dev) {
- dst->neighbour->dev = &loopback_dev;
+ dst->neighbour->dev = loopback_dev;
     dev_put(dev);
- dev_hold(&loopback_dev);
+ dev_hold(dst->neighbour->dev);
     }
 }
}
```

Index: net-2.6.24/net/decnet/dn_dev.c

```
=====
--- net-2.6.24.orig/net/decnet/dn_dev.c
+++ net-2.6.24/net/decnet/dn_dev.c
@@ -869,10 +869,10 @@ 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 == loopback_dev)
     return rv;
 }
- dev = &loopback_dev;
+ dev = loopback_dev;
 dev_hold(dev);
 goto last_chance;
}
```

Index: net-2.6.24/net/decnet/dn_route.c

```
=====
--- net-2.6.24.orig/net/decnet/dn_route.c
+++ net-2.6.24/net/decnet/dn_route.c
@@ -887,7 +887,7 @@ static int dn_route_output_slow(struct d
     .scope = RT_SCOPE_UNIVERSE,
     },
}
```

```

        .mark = oldflp->mark,
-       .iif = loopback_dev.ifindex,
+       .iif = loopback_dev->ifindex,
        .oif = oldflp->oif };
    struct dn_route *rt = NULL;
    struct net_device *dev_out = NULL, *dev;
@@ -904,7 +904,7 @@ static int dn_route_output_slow(struct d
    "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, loopback_dev->ifindex, oldflp->oif);

/* If we have an output interface, verify its a DECnet device */
if (oldflp->oif) {
@@ -957,7 +957,7 @@ source_ok:
    err = -EADDRNOTAVAIL;
    if (dev_out)
        dev_put(dev_out);
-    dev_out = &loopback_dev;
+    dev_out = loopback_dev;
    dev_hold(dev_out);
    if (!fl.fld_dst) {
        fl.fld_dst =
@@ -966,7 +966,7 @@ source_ok:
        if (!fl.fld_dst)
            goto out;
    }
-    fl.oif = loopback_dev.ifindex;
+    fl.oif = loopback_dev->ifindex;
    res.type = RTN_LOCAL;
    goto make_route;
}
@@ -1012,7 +1012,7 @@ source_ok:
    if (dev_out)
        dev_put(dev_out);
    if (dn_dev_islocal(neigh->dev, fl.fld_dst)) {
-        dev_out = &loopback_dev;
+        dev_out = loopback_dev;
        res.type = RTN_LOCAL;
    } else {
        dev_out = neigh->dev;
@@ -1033,7 +1033,7 @@ 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 = loopback_dev;

```

```

dev_hold(dev_out);
res.type = RTN_LOCAL;
goto select_source;
@@ -1069,7 +1069,7 @@ select_source:
    fl.fld_src = fl.fld_dst;
    if (dev_out)
        dev_put(dev_out);
-   dev_out = &loopback_dev;
+   dev_out = loopback_dev;
    dev_hold(dev_out);
    fl.oif = dev_out->ifindex;
    if (res.fi)

```

Index: net-2.6.24/net/ipv4/devinet.c

```

=====
--- net-2.6.24.orig/net/ipv4/devinet.c
+++ net-2.6.24/net/ipv4/devinet.c
@@ -203,7 +203,7 @@ static void inetdev_destroy(struct in_de
    ASSERT_RTNL();

    dev = in_dev->dev;
-   if (dev == &loopback_dev)
+   if (dev == loopback_dev)
        return;

```

```

    in_dev->dead = 1;
@@ -1061,7 +1061,7 @@ static int inetdev_event(struct notifier
    in_dev = inetdev_init(dev);
    if (!in_dev)
        return notifier_from_errno(-ENOMEM);
-   if (dev == &loopback_dev) {
+   if (dev == loopback_dev) {
        IN_DEV_CONF_SET(in_dev, NOXFRM, 1);
        IN_DEV_CONF_SET(in_dev, NOPOLICY, 1);
    }
@@ -1077,7 +1077,7 @@ static int inetdev_event(struct notifier
    case NETDEV_UP:
        if (dev->mtu < 68)
            break;
-   if (dev == &loopback_dev) {
+   if (dev == loopback_dev) {
        struct in_ifaddr *ifa;
        if ((ifa = inet_alloc_ifa()) != NULL) {
            ifa->ifa_local =

```

Index: net-2.6.24/net/ipv4/ipconfig.c

```

=====
--- net-2.6.24.orig/net/ipv4/ipconfig.c
+++ net-2.6.24/net/ipv4/ipconfig.c
@@ -190,11 +190,11 @@ static int __init ic_open_devs(void)

```

```

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);
+ 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);

for_each_netdev(&init_net, dev) {
- if (dev == &loopback_dev)
+ if (dev == loopback_dev)
    continue;
    if (user_dev_name[0] ? !strcmp(dev->name, user_dev_name) :
        (!(dev->flags & IFF_LOOPBACK) &&
Index: net-2.6.24/net/ipv4/ipvs/ip_vs_core.c
=====
--- net-2.6.24.orig/net/ipv4/ipvs/ip_vs_core.c
+++ net-2.6.24/net/ipv4/ipvs/ip_vs_core.c
@@ -961,7 +961,7 @@ ip_vs_in(unsigned int hooknum, struct sk
 * ... 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 == loopback_dev || skb->sk)) {
    IP_VS_DBG(12, "packet type=%d proto=%d daddr=%d.%d.%d.%d ignored\n",
        skb->pkt_type,
        ip_hdr(skb)->protocol,
Index: net-2.6.24/net/ipv4/route.c
=====
--- net-2.6.24.orig/net/ipv4/route.c
+++ net-2.6.24/net/ipv4/route.c
@@ -1402,8 +1402,8 @@ static void ipv4_dst_ifdown(struct dst_e
{
    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 != loopback_dev && idev && idev->dev == dev) {
+ struct in_device *loopback_idev = in_dev_get(loopback_dev);
    if (loopback_idev) {
        rt->idev = loopback_idev;
        in_dev_put(idev);
@@ -1555,7 +1555,7 @@ static int ip_route_input_mc(struct sk_b
#endif
    rth->rt_iif =
    rth->fl.iif = dev->ifindex;
- rth->u.dst.dev = &loopback_dev;
+ rth->u.dst.dev = loopback_dev;

```

```

dev_hold(rth->u.dst.dev);
rth->idev = in_dev_get(rth->u.dst.dev);
rth->fl.oif = 0;
@@ -1812,7 +1812,7 @@ static int ip_route_input_slow(struct sk
if (res.type == RTN_LOCAL) {
int result;
result = fib_validate_source(saddr, daddr, tos,
-    loopback_dev.ifindex,
+    loopback_dev->ifindex,
dev, &spec_dst, &itag);
if (result < 0)
goto martian_source;
@@ -1879,7 +1879,7 @@ local_input:
#endif
rth->rt_iif =
rth->fl.iif = dev->ifindex;
- rth->u.dst.dev = &loopback_dev;
+ rth->u.dst.dev = loopback_dev;
dev_hold(rth->u.dst.dev);
rth->idev = in_dev_get(rth->u.dst.dev);
rth->rt_gateway = daddr;
@@ -2149,7 +2149,7 @@ static int ip_route_output_slow(struct r
RT_SCOPE_UNIVERSE),
}},
.mark = oldflp->mark,
- .iif = loopback_dev.ifindex,
+ .iif = loopback_dev->ifindex,
.oif = oldflp->oif };
struct fib_result res;
unsigned flags = 0;
@@ -2243,9 +2243,9 @@ static int ip_route_output_slow(struct r
fl.fl4_dst = fl.fl4_src = htonl(INADDR_LOOPBACK);
if (dev_out)
dev_put(dev_out);
- dev_out = &loopback_dev;
+ dev_out = loopback_dev;
dev_hold(dev_out);
- fl.oif = loopback_dev.ifindex;
+ fl.oif = loopback_dev->ifindex;
res.type = RTN_LOCAL;
flags |= RTCF_LOCAL;
goto make_route;
@@ -2290,7 +2290,7 @@ static int ip_route_output_slow(struct r
fl.fl4_src = fl.fl4_dst;
if (dev_out)
dev_put(dev_out);
- dev_out = &loopback_dev;
+ dev_out = loopback_dev;

```

```

dev_hold(dev_out);
fl.oif = dev_out->ifindex;
if (res.fi)
Index: net-2.6.24/net/ipv4/xfrm4_policy.c
=====
--- net-2.6.24.orig/net/ipv4/xfrm4_policy.c
+++ net-2.6.24/net/ipv4/xfrm4_policy.c
@@ -306,7 +306,7 @@ static void xfrm4_dst_ifdown(struct dst_

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

```

```

do {
Index: net-2.6.24/net/ipv6/addrconf.c
=====
--- net-2.6.24.orig/net/ipv6/addrconf.c
+++ net-2.6.24/net/ipv6/addrconf.c
@@ -2410,7 +2410,7 @@ static int addrconf_ifdown(struct net_de

```

```

ASSERT_RTNL();

- if (dev == &loopback_dev && how == 1)
+ if (dev == loopback_dev && how == 1)
    how = 0;

rt6_ifdown(dev);
@@ -4212,16 +4212,19 @@ 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(loopback_dev))
    err = -ENOMEM;
    rtnl_unlock();
    if (err)
        return err;

- ip6_null_entry.rt6i_iddev = in6_dev_get(&loopback_dev);
+ ip6_null_entry.u.dst.dev = loopback_dev;
+ ip6_null_entry.rt6i_iddev = in6_dev_get(loopback_dev);
#ifdef CONFIG_IPV6_MULTIPLE_TABLES
- ip6_prohibit_entry.rt6i_iddev = in6_dev_get(&loopback_dev);
- ip6_blk_hole_entry.rt6i_iddev = in6_dev_get(&loopback_dev);
+ ip6_prohibit_entry.u.dst.dev = loopback_dev;
+ ip6_prohibit_entry.rt6i_iddev = in6_dev_get(loopback_dev);

```

```
+ ip6_blk_hole_entry.u.dst.dev = loopback_dev;
+ ip6_blk_hole_entry.rt6i_iddev = in6_dev_get(loopback_dev);
#endif
```

```
register_netdevice_notifier(&ipv6_dev_notf);
@@ -4276,7 +4279,7 @@ void __exit addrconf_cleanup(void)
    continue;
    addrconf_ifdown(dev, 1);
}
- addrconf_ifdown(&loopback_dev, 2);
+ addrconf_ifdown(loopback_dev, 2);
```

```
/*
 * Check hash table.
Index: net-2.6.24/net/ipv6/ip6_input.c
=====
--- net-2.6.24.orig/net/ipv6/ip6_input.c
+++ net-2.6.24/net/ipv6/ip6_input.c
@@ -91,7 +91,7 @@ int ipv6_rcv(struct sk_buff *skb, struct
 *
 * BTW, when we send a packet for our own local address on a
 * non-loopback interface (e.g. ethX), it is being delivered
- * via the loopback interface (lo) here; skb->dev = &loopback_dev.
+ * via the loopback interface (lo) here; skb->dev = loopback_dev.
 * It, however, should be considered as if it is being
 * arrived via the sending interface (ethX), because of the
 * nature of scoping architecture. --yoshfuji
```

```
Index: net-2.6.24/net/ipv6/netfilter/ip6t_REJECT.c
=====
--- net-2.6.24.orig/net/ipv6/netfilter/ip6t_REJECT.c
+++ net-2.6.24/net/ipv6/netfilter/ip6t_REJECT.c
@@ -167,7 +167,7 @@ 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 = loopback_dev;
```

```
    icmpv6_send(skb_in, ICMPV6_DEST_UNREACH, code, 0, NULL);
}
```

```
Index: net-2.6.24/net/ipv6/route.c
=====
--- net-2.6.24.orig/net/ipv6/route.c
+++ net-2.6.24/net/ipv6/route.c
@@ -138,7 +138,6 @@ struct rt6_info ip6_null_entry = {
    .dst = {
        .__refcnt = ATOMIC_INIT(1),
        .__use = 1,
```

```

- .dev = &loopback_dev,
  .obsolete = -1,
  .error = -ENETUNREACH,
  .metrics = { [RTAX_HOPLIMIT - 1] = 255, },
@@ -164,7 +163,6 @@ struct rt6_info ip6_prohibit_entry = {
  .dst = {
    .__refcnt = ATOMIC_INIT(1),
    .__use = 1,
- .dev = &loopback_dev,
  .obsolete = -1,
  .error = -EACCES,
  .metrics = { [RTAX_HOPLIMIT - 1] = 255, },
@@ -184,7 +182,6 @@ struct rt6_info ip6_blk_hole_entry = {
  .dst = {
    .__refcnt = ATOMIC_INIT(1),
    .__use = 1,
- .dev = &loopback_dev,
  .obsolete = -1,
  .error = -EINVAL,
  .metrics = { [RTAX_HOPLIMIT - 1] = 255, },
@@ -224,8 +221,8 @@ static void ip6_dst_ifdown(struct dst_en
  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 != loopback_dev && idev != NULL && idev->dev == dev) {
+ struct inet6_dev *loopback_idev = in6_dev_get(loopback_dev);
  if (loopback_idev != NULL) {
    rt->rt6i_idev = loopback_idev;
    in6_dev_put(idev);
@@ -1188,12 +1185,12 @@ int ip6_route_add(struct fib6_config *cf
  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 != loopback_dev) {
    if (dev) {
      dev_put(dev);
      in6_dev_put(idev);
    }
- dev = &loopback_dev;
+ dev = loopback_dev;
    dev_hold(dev);
    idev = in6_dev_get(dev);
    if (!idev) {
@@ -1897,13 +1894,13 @@ struct rt6_info *addrconf_dst_alloc(stru
  if (rt == NULL)

```

```

return ERR_PTR(-ENOMEM);

- dev_hold(&loopback_dev);
+ dev_hold(loopback_dev);
  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 = loopback_dev;
  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));
Index: net-2.6.24/net/ipv6/xfrm6_policy.c
=====
--- net-2.6.24.orig/net/ipv6/xfrm6_policy.c
+++ net-2.6.24/net/ipv6/xfrm6_policy.c
@@ -375,7 +375,7 @@ static void xfrm6_dst_ifdown(struct dst_

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

  do {
Index: net-2.6.24/net/xfrm/xfrm_policy.c
=====
--- net-2.6.24.orig/net/xfrm/xfrm_policy.c
+++ net-2.6.24/net/xfrm/xfrm_policy.c
@@ -1949,8 +1949,8 @@ static int stale_bundle(struct dst_entry
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 = loopback_dev;
+ dev_hold(dst->dev);
  dev_put(dev);
  }
}
--

```

Subject: [net-2.6.24][patch 2/2] Dynamically allocate the loopback device

Posted by [Daniel Lezcano](#) on Mon, 17 Sep 2007 13:45:11 GMT

[View Forum Message](#) <> [Reply to Message](#)

From: Daniel Lezcano <dlezcano@fr.ibm.com>

Doing this makes loopback.c a better example of how to do a simple network device, and it removes the special case single static allocation of a struct net_device, hopefully making maintenance easier.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>

Acked-By: Kirill Korotaev <dev@sw.ru>

Acked-by: Benjamin Thery <benjamin.thery@bull.net>

drivers/net/loopback.c | 69 ++++++-----
1 file changed, 43 insertions(+), 26 deletions(-)

Index: net-2.6.24/drivers/net/loopback.c

=====

--- net-2.6.24.orig/drivers/net/loopback.c

+++ net-2.6.24/drivers/net/loopback.c

@@ -202,44 +202,61 @@ static const struct ethtool_ops loopback

* 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 = {

- .name = "lo",

- .get_stats = &get_stats,

- .mtu = (16 * 1024) + 20 + 20 + 12,

- .hard_start_xmit = loopback_xmit,

- .hard_header = eth_header,

- .hard_header_cache = eth_header_cache,

- .header_cache_update = eth_header_cache_update,

- .hard_header_len = ETH_HLEN, /* 14 */

- .addr_len = ETH_ALEN, /* 6 */

- .tx_queue_len = 0,

- .type = ARPHRD_LOOPBACK, /* 0x0001*/

- .rebuild_header = eth_rebuild_header,

- .flags = IFF_LOOPBACK,

- .features = NETIF_F_SG | NETIF_F_FRAGLIST

+static void loopback_setup(struct net_device *dev)

+{

+ dev->get_stats = &get_stats;

+ dev->mtu = (16 * 1024) + 20 + 20 + 12;

+ dev->hard_start_xmit = loopback_xmit;

+ dev->hard_header = eth_header;

+ dev->hard_header_cache = eth_header_cache;

```

+ dev->header_cache_update = eth_header_cache_update;
+ dev->hard_header_len = ETH_HLEN; /* 14 */
+ dev->addr_len = ETH_ALEN; /* 6 */
+ dev->tx_queue_len = 0;
+ dev->type = ARPHRD_LOOPBACK; /* 0x0001*/
+ dev->rebuild_header = eth_rebuild_header;
+ dev->flags = IFF_LOOPBACK;
+ dev->features = NETIF_F_SG | NETIF_F_FRAGLIST
#ifdef LOOPBACK_TSO
-   | NETIF_F_TSO
+ | NETIF_F_TSO
#endif
-   | NETIF_F_NO_CSUM | NETIF_F_HIGHDMA
-   | NETIF_F_LLTX
-   | NETIF_F_NETNS_LOCAL,
- .ethtool_ops = &loopback_ethtool_ops,
- .nd_net         = &init_net,
-};
-
-struct net_device *loopback_dev = &__loopback_dev;
+ | NETIF_F_NO_CSUM
+ | NETIF_F_HIGHDMA
+ | NETIF_F_LLTX
+ | NETIF_F_NETNS_LOCAL,
+ dev->ethtool_ops = &loopback_ethtool_ops;
+}

/* Setup and register the loopback device. */
static int __init loopback_init(void)
{
- int err = register_netdev(loopback_dev);
+ struct net_device *dev;
+ int err;
+
+ err = -ENOMEM;
+ dev = alloc_netdev(0, "lo", loopback_setup);
+ if (!dev)
+   goto out;

+ err = register_netdev(dev);
+ if (err)
+   goto out_free_netdev;
+
+ err = 0;
+ loopback_dev = dev;
+
+out:
  if (err)

```

```
panic("loopback: Failed to register netdevice: %d\n", err);
+ return err;

+out_free_netdev:
+ free_netdev(dev);
+ goto out;
return err;
};

-module_init(loopback_init);
+fs_initcall(loopback_init);

+struct net_device *loopback_dev;
EXPORT_SYMBOL(loopback_dev);

--
```

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

Subject: Re: [net-2.6.24][patch 2/2] Dynamically allocate the loopback device
Posted by [Stephen Hemminger](#) on Mon, 17 Sep 2007 17:13:14 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Mon, 17 Sep 2007 15:45:11 +0200
dlezcano@fr.ibm.com wrote:

```
> From: Daniel Lezcano <dlezcano@fr.ibm.com>
>
> Doing this makes loopback.c a better example of how to do a
> simple network device, and it removes the special case
> single static allocation of a struct net_device, hopefully
> making maintenance easier.
>
```

What is before/after code and data size, does it make code smaller?

```
>
> -module_init(loopback_init);
> +fs_initcall(loopback_init);
>
> +struct net_device *loopback_dev;
> EXPORT_SYMBOL(loopback_dev);
```

--

Stephen Hemminger <shemminger@linux-foundation.org>

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>

Subject: Re: [net-2.6.24][patch 2/2] Dynamically allocate the loopback device

Posted by [Daniel Lezcano](#) on Mon, 17 Sep 2007 18:52:05 GMT

[View Forum Message](#) <> [Reply to Message](#)

Stephen Hemminger wrote:

> On Mon, 17 Sep 2007 15:45:11 +0200

> dlezcano@fr.ibm.com wrote:

>

>> From: Daniel Lezcano <dlezcano@fr.ibm.com>

>>

>> Doing this makes loopback.c a better example of how to do a

>> simple network device, and it removes the special case

>> single static allocation of a struct net_device, hopefully

>> making maintenance easier.

>>

>

> What is before/after code and data size, does it make code smaller?

Interesting question, here are the results based on the same config file.

Without the patchset:

vmlinux:

	text	data	bss	dec	hex	filename
	2446606	188243	163840	2798689	2ab461	vmlinux

loopback.o

	text	data	bss	dec	hex	filename
	417	1040	8	1465	5b9	drivers/net/loopback.o

With the patchset:

vmlinux:

	text	data	bss	dec	hex	filename
	2446853	187187	163840	2797880	2ab138	./vmlinux

loopback.o

text	data	bss	dec	hex	filename
609	4	12	625	271	drivers/net/loopback.o

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>

Subject: Re: [net-2.6.24][patch 2/2] Dynamically allocate the loopback device

Posted by [Peter Waskiewicz](#) on Mon, 17 Sep 2007 19:12:24 GMT

[View Forum Message](#) <> [Reply to Message](#)

This would be a good opportunity to remove the single-allocated queue struct in netdevice (at the bottom) that we had to put in to accomodate the static loopback. Now we can set it back to a zero element list, and have alloc_netdev_mq() just allocate the number of queues requested, not num_queues - 1.

I'll put a patch together based on this patchset.

Thanks,

-PJ Waskiewicz

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>

Subject: Re: [net-2.6.24][patch 2/2] Dynamically allocate the loopback device

Posted by [davem](#) on Tue, 18 Sep 2007 01:53:49 GMT

[View Forum Message](#) <> [Reply to Message](#)

From: "Peter Waskiewicz" <pjwaskiewicz@gmail.com>

Date: Mon, 17 Sep 2007 12:12:24 -0700

> This would be a good opportunity to remove the single-allocated queue struct
> in netdevice (at the bottom) that we had to put in to accomodate the static
> loopback. Now we can set it back to a zero element list, and have
> alloc_netdev_mq() just allocate the number of queues requested, not
> num_queues - 1.

>

> I'll put a patch together based on this patchset.

Thanks Peter.

I'll also let this sit so that Eric can provide any feedback he wants and also figure out how he will use this for the namespace stuff.

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

Subject: Re: [net-2.6.24][patch 2/2] Dynamically allocate the loopback device
Posted by [ebiederm](#) on Tue, 18 Sep 2007 02:44:14 GMT
[View Forum Message](#) <> [Reply to Message](#)

David Miller <davem@davemloft.net> writes:

> From: "Peter Waskiewicz" <pjwaskiewicz@gmail.com>
> Date: Mon, 17 Sep 2007 12:12:24 -0700
>
>> This would be a good opportunity to remove the single-allocated queue struct
>> in netdevice (at the bottom) that we had to put in to accomodate the static
>> loopback. Now we can set it back to a zero element list, and have
>> alloc_netdev_mq() just allocate the number of queues requested, not
>> num_queues - 1.
>>
>> I'll put a patch together based on this patchset.
>
> Thanks Peter.
>
> I'll also let this sit so that Eric can provide any feedback
> he wants and also figure out how he will use this for the
> namespace stuff.

Acked-by: "Eric W. Biederman" <ebiederm@xmission.com>
Not that it doesn't already have my signed off by.

I have an earlier version of this patch sitting in my tree,
along with some additional patches to make this per namespace.

I don't really care which version of this patch goes in and
I'm happy to give Daniel credit for doing the final work to get this
patch merged.

I think it is important for bisect reasons that we first dynamically
allocate the loopback device and then make it per network namespace.
So someone can determine which part of the work caused a problem if

there is one.

Eric

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

Subject: Re: [net-2.6.24][patch 2/2] Dynamically allocate the loopback device
Posted by [davem](#) on Wed, 26 Sep 2007 02:24:06 GMT

[View Forum Message](#) <> [Reply to Message](#)

From: ebiederm@xmission.com (Eric W. Biederman)
Date: Mon, 17 Sep 2007 20:44:14 -0600

> David Miller <davem@davemloft.net> writes:
>
> > From: "Peter Waskiewicz" <pjwaskiewicz@gmail.com>
> > Date: Mon, 17 Sep 2007 12:12:24 -0700
> >
> >> This would be a good opportunity to remove the single-allocated queue struct
> >> in netdevice (at the bottom) that we had to put in to accomodate the static
> >> loopback. Now we can set it back to a zero element list, and have
> >> alloc_netdev_mq() just allocate the number of queues requested, not
> >> num_queues - 1.
> >>
> >> I'll put a patch together based on this patchset.
> >
> > Thanks Peter.
> >
> > I'll also let this sit so that Eric can provide any feedback
> > he wants and also figure out how he will use this for the
> > namespace stuff.
>
> Acked-by: "Eric W. Biederman" <ebiederm@xmission.com>
> Not that it doesn't already have my signed off by.

I've put these patches into the just-rebased net-2.6.24 tree.

I made a minor modification to the second patch, the
out_free_netdev: code in loopback_init() ended like this:

```
out_free_netdev:  
    free_netdev(dev);  
    goto out;  
    return err;
```

```
};
```

I got rid of the spurious return statement and the trailing semi-colon after the function closing brace.

Thanks.

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

Subject: Re: [net-2.6.24][patch 2/2] Dynamically allocate the loopback device
Posted by [ebiederm](#) on Wed, 26 Sep 2007 21:40:25 GMT
[View Forum Message](#) <> [Reply to Message](#)

David Miller <davem@davemloft.net> writes:

```
> I've put these patches into the just-rebased net-2.6.24 tree.  
>  
> I made a minor modification to the second patch, the  
> out_free_netdev: code in loopback_init() ended like this:  
>  
> out_free_netdev:  
> free_netdev(dev);  
> goto out;  
> return err;  
> };  
>  
> I got rid of the spurious return statement and the trailing  
> semi-colon after the function closing brace.
```

Thanks. I feel silly for not doing a closer code review of this variant of the patch and missing this bug.

Eric

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