Subject: [PATCH 3/12] L2 network namespace (v3): loopback device virtualization
Posted by Mishin Dmitry on Wed, 17 Jan 2007 16:00:32 GMT
View Forum Message <> Reply to Message

Added per-namespace network loopback device

Signed-off-by: Dmitry Mishin <dim@openvz.org>

```
---
 drivers/net/loopback.c       | 112 +++++++++++++++++++++++++++++-------------
 include/linux/net_namespace.h |   2
 include/linux/netdevice.h    |   6 +-
 net/core/dev.c               |   3 -
 net/core/net_namespace.c     |  16 ++++++
 net/ipv4/devinet.c           |   2
 net/ipv6/addrconf.c          |   2
 net/ipv6/route.c             |   6 +-
 8 files changed, 108 insertions(+), 41 deletions(-)

--- linux-2.6.20-rc4-mm1.net_ns.orig/drivers/net/loopback.c
+++ linux-2.6.20-rc4-mm1.net_ns/drivers/net/loopback.c
@@ -62,7 +62,12 @@ struct pcpu_lstats {
  unsigned long packets;
  unsigned long bytes;
 };
-static DEFINE_PER_CPU(struct pcpu_lstats, pcpu_lstats);
+#ifndef CONFIG_NET_NS
+struct pcpu_lstats  *pcpu_lstats_p;
+#define pcpu_lstats_ptr(ns)  pcpu_lstats_p
+#else
+#define pcpu_lstats_ptr(ns)  (ns->pcpu_lstats_p)
+#endif

 #define LOOPBACK_OVERHEAD (128 + MAX_HEADER + 16 + 16)

@@ -154,7 +159,8 @@ static int loopback_xmit(struct sk_buff
  dev->last_rx = jiffies;

  /* it's OK to use __get_cpu_var() because BHs are off */
- lb_stats = &__get_cpu_var(pcpu_lstats);
+ lb_stats = per_cpu_ptr(pcpu_lstats_ptr(dev->net_ns),
+       smp_processor_id());
  lb_stats->bytes += skb->len;
  lb_stats->packets++;

@@ -163,11 +169,9 @@ static int loopback_xmit(struct sk_buff
  return 0;
 }
```

```
-static struct net_device_stats loopback_stats;
-
 static struct net_device_stats *get_stats(struct net_device *dev)
 {
- struct net_device_stats *stats = &loopback_stats;
+ struct net_device_stats *stats = netdev_priv(dev);
  unsigned long bytes = 0;
  unsigned long packets = 0;
  int i;
@@ -175,7 +179,7 @@ static struct net_device_stats *get_stat
  for_each_possible_cpu(i) {
   const struct pcpu_lstats *lb_stats;

-  lb_stats = &per_cpu(pcpu_lstats, i);
+  lb_stats = per_cpu_ptr(pcpu_lstats_ptr(dev->net_ns), i);
   bytes   += lb_stats->bytes;
   packets += lb_stats->packets;
  }
@@ -200,40 +204,80 @@ static const struct ethtool_ops loopback
  .get_rx_csum  = always_on,
 };

-/*
- * 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,
- .priv   = &loopback_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
+struct net_device *loopback_dev_p;
+EXPORT_SYMBOL(loopback_dev_p);
+
+struct pcpu_lstats *pcpu_lstats_p;
+EXPORT_SYMBOL(pcpu_lstats_p);
```

```
+
+
+void loopback_dev_dtor(struct net_device *dev)
+{
+ free_percpu(pcpu_lstats_ptr(dev->net_ns));
+ free_netdev(dev);
+}
+
+void loopback_dev_ctor(struct net_device *dev)
+{
+ 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
 #endif
       | NETIF_F_NO_CSUM | NETIF_F_HIGHDMA
-      | NETIF_F_LLTX,
- .ethtool_ops  = &loopback_ethtool_ops,
-};
+      | NETIF_F_LLTX;
+ dev->ethtool_ops = &loopback_ethtool_ops;
+ dev->get_stats  = &get_stats;
+ dev->destructor  = loopback_dev_dtor;
+}

 /* Setup and register the loopback device. */
-static int __init loopback_init(void)
+int loopback_init(void)
 {
- return register_netdev(&loopback_dev);
-};
+ struct net_namespace *ns = current_net_ns;
+ int err;

-module_init(loopback_init);
+ err = -ENOMEM;
+ loopback_dev_ptr = alloc_netdev(sizeof(struct net_device_stats), "lo",
+       loopback_dev_ctor);
```

```
+ if (loopback_dev_ptr == NULL)
+  goto out;
+ pcpu_lstats_ptr(ns) = alloc_percpu(struct pcpu_lstats);
+ if (pcpu_lstats_ptr(ns) == NULL)
+  goto out_stats;
+ /*
+  * loopback device doesn't hold active reference: it doesn't prevent
+  * stopping of net_namespace. So, just put old namespace.
+  */
+#ifdef CONFIG_NET_NS
+ put_net_ns(loopback_dev_ptr->net_ns);
+ loopback_dev_ptr->net_ns = ns;
+#endif
+ err = register_netdev(loopback_dev_ptr);
+ if (err)
+  goto out_register;
+ return 0;

-EXPORT_SYMBOL(loopback_dev);
+out_register:
+ free_percpu(ns->pcpu_lstats_p);
+#ifdef CONFIG_NET_NS
+ /* in order to avoid put in free_netdev() */
+ loopback_dev_ptr->net_ns = NULL;
+#endif
+out_stats:
+ free_netdev(loopback_dev_ptr);
+out:
+ return err;
+}
+
+module_init(loopback_init);
--- linux-2.6.20-rc4-mm1.net_ns.orig/include/linux/net_namespace.h
+++ linux-2.6.20-rc4-mm1.net_ns/include/linux/net_namespace.h
@@ -9,6 +9,8 @@
 struct net_namespace {
   struct kref  kref;
   struct net_device *dev_base_p, **dev_tail_p;
+ struct net_device *loopback_dev_p;
+ struct pcpu_lstats *pcpu_lstats_p;
   unsigned int  hash;
 };

--- linux-2.6.20-rc4-mm1.net_ns.orig/include/linux/netdevice.h
+++ linux-2.6.20-rc4-mm1.net_ns/include/linux/netdevice.h
@@ -570,16 +570,20 @@ struct packet_type {
 #include <linux/notifier.h>
 #include <linux/net_namespace.h>
```

```
-extern struct net_device  loopback_dev;  /* The loopback */
+extern struct net_device  *loopback_dev_p;
 #ifndef CONFIG_NET_NS
+#define loopback_dev_ptr  loopback_dev_p
 extern struct net_device  *dev_base;  /* All devices */
 #define dev_base_ns(dev)  dev_base
 #else
+#define loopback_dev_ptr  (current_net_ns->loopback_dev_p)
 #define dev_base   (current_net_ns->dev_base_p)
 #define dev_base_ns(dev)  (dev->net_ns->dev_base_p)
 #endif
+#define loopback_dev   (*loopback_dev_ptr) /* The loopback */
 extern rwlock_t    dev_base_lock;  /* Device list lock */

+extern int   loopback_init(void);
 extern int    netdev_boot_setup_check(struct net_device *dev);
 extern unsigned long  netdev_boot_base(const char *prefix, int unit);
 extern struct net_device    *dev_getbyhwaddr(unsigned short type, char *hwaddr);
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/core/dev.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/core/dev.c
@@ -3253,7 +3253,8 @@ void free_netdev(struct net_device *dev)

  ns = dev->net_ns;
  if (ns != NULL) {
-  put_net_ns(ns);
+  if (dev != ns->loopback_dev_p)
+   put_net_ns(ns);
   dev->net_ns = NULL;
  }
 #endif
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/core/net_namespace.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/core/net_namespace.c
@@ -10,6 +10,7 @@
 #include <linux/nsproxy.h>
 #include <linux/net_namespace.h>
 #include <linux/net.h>
+#include <linux/netdevice.h>

 struct net_namespace init_net_ns = {
  .kref = {
@@ -17,6 +18,8 @@ struct net_namespace init_net_ns = {
  },
  .dev_base_p = NULL,
  .dev_tail_p = &init_net_ns.dev_base_p,
+ .loopback_dev_p = NULL,
+ .pcpu_lstats_p = NULL,
 };
```

```
 #ifdef CONFIG_NET_NS
@@ -41,7 +44,19 @@ static struct net_namespace *clone_net_n
  ns->dev_base_p = NULL;
  ns->dev_tail_p = &ns->dev_base_p;
  ns->hash = net_random();
+
+ if ((push_net_ns(ns)) != old_ns)
+  BUG();
+ if (loopback_init())
+  goto out_loopback;
+ pop_net_ns(old_ns);
  return ns;
+
+out_loopback:
+ pop_net_ns(old_ns);
+ BUG_ON(atomic_read(&ns->kref.refcount) != 1);
+ kfree(ns);
+ return NULL;
 }

 /*
@@ -79,6 +94,7 @@ void free_net_ns(struct kref *kref)
  struct net_namespace *ns;

  ns = container_of(kref, struct net_namespace, kref);
+ unregister_netdev(ns->loopback_dev_p);
  if (ns->dev_base_p != NULL) {
   printk("NET_NS: BUG: namespace %p has devices! ref %d\n",
     ns, atomic_read(&ns->kref.refcount));
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/ipv4/devinet.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/ipv4/devinet.c
@@ -198,7 +198,7 @@ static void inetdev_destroy(struct in_de
  ASSERT_RTNL();

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

  in_dev->dead = 1;
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/ipv6/addrconf.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/ipv6/addrconf.c
@@ -2360,7 +2360,7 @@ static int addrconf_ifdown(struct net_de

  ASSERT_RTNL();

- if (dev == &loopback_dev && how == 1)
```

```
+ if (dev == loopback_dev_p && how == 1)
   how = 0;

  rt6_ifdown(dev);
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/ipv6/route.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/ipv6/route.c
@@ -124,7 +124,6 @@ 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, },
@@ -150,7 +149,6 @@ 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, },
@@ -170,7 +168,6 @@ 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, },
@@ -2469,7 +2466,10 @@ void __init ip6_route_init(void)
 #endif
 #ifdef CONFIG_IPV6_MULTIPLE_TABLES
  fib6_rules_init();
+ ip6_prohibit_entry.u.dst.dev = &loopback_dev;
+ ip6_blk_hole_entry.u.dst.dev = &loopback_dev;
 #endif
+ ip6_null_entry.u.dst.dev = &loopback_dev;
 }

 void ip6_route_cleanup(void)
```

_____

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