Subject: [patch 36/38][IPV6] addrconf - make addrconf per namespace
Posted by Daniel Lezcano on Mon, 03 Dec 2007 16:17:12 GMT
View Forum Message <> Reply to Message

All the infrastructure to propagate the network namespace
information is ready. Make use of it.

There is a special case here between the initial network namespace
and the other namespaces:

* When ipv6 is initialized at boot time (aka in the init_net), it
registers to the notifier callback. So addrconf_notify will be called
as many time as there are network devices setup on the system and the
function will add ipv6 addresses to the network devices. But the first
device which needs to have its ipv6 address setup is the loopback,
unfortunatly this is not the case. So the loopback address is setup
manually in the ipv6 init function.

* With the network namespace, this ordering problem does not appears
because notifier is already setup and active, so as soon as we register
the loopback the ipv6 address is setup and it will be the first device.

Signed-off-by: Benjamin Thery <benjamin.thery@bull.net>
Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>
---
 net/ipv6/addrconf.c | 175 ++++++++++++++++++++++++++----------------------
 1 file changed, 81 insertions(+), 94 deletions(-)

Index: linux-2.6-netns/net/ipv6/addrconf.c
===
--- linux-2.6-netns.orig/net/ipv6/addrconf.c
+++ linux-2.6-netns/net/ipv6/addrconf.c
@@ -726,9 +726,9 @@ static void ipv6_del_addr(struct inet6_i
  if ((ifp->flags & IFA_F_PERMANENT) && onlink < 1) {
   struct in6_addr prefix;
   struct rt6_info *rt;
-
+ struct net *net = ifp->idev->dev->nd_net;
   ipv6_addr_prefix(&prefix, &ifp->addr, ifp->prefix_len);
- rt = rt6_lookup(&init_net, &prefix, NULL, ifp->idev->dev->ifindex, 1);
+ rt = rt6_lookup(net, &prefix, NULL, ifp->idev->dev->ifindex, 1);

   if (rt && ((rt->rt6i_flags & (RTF_GATEWAY | RTF_DEFAULT)) == 0)) {
    if (onlink == 0) {
@@ -880,6 +880,7 @@ int ipv6_dev_get_saddr(struct net_device
 {
  struct ipv6_saddr_score hiscore;
  struct inet6_ifaddr *ifa_result = NULL;

```
+ struct net *net = daddr_dev->nd_net;
  int daddr_type = __ipv6_addr_type(daddr);
  int daddr_scope = __ipv6_addr_src_scope(daddr_type);
  int daddr_ifindex = daddr_dev ? daddr_dev->ifindex : 0;
@@ -891,7 +892,7 @@ int ipv6_dev_get_saddr(struct net_device
  read_lock(&dev_base_lock);
  rcu_read_lock();

- for_each_netdev(&init_net, dev) {
+ for_each_netdev(net, dev) {
   struct inet6_dev *idev;
   struct inet6_ifaddr *ifa;

@@ -1532,7 +1533,7 @@ addrconf_prefix_route(struct in6_addr *p
   .fc_flags = RTF_UP | flags,
   .fc_nlinfo.pid = 0,
   .fc_nlinfo.nlh = NULL,
-  .fc_nlinfo.net = &init_net,
+  .fc_nlinfo.net = dev->nd_net,
  };

  ipv6_addr_copy(&cfg.fc_dst, pfx);
@@ -1561,7 +1562,7 @@ static void addrconf_add_mroute(struct n
   .fc_flags = RTF_UP,
   .fc_nlinfo.pid = 0,
   .fc_nlinfo.nlh = NULL,
-  .fc_nlinfo.net = &init_net,
+  .fc_nlinfo.net = dev->nd_net,
  };

  ipv6_addr_set(&cfg.fc_dst, htonl(0xFF000000), 0, 0, 0);
@@ -1580,7 +1581,7 @@ static void sit_route_add(struct net_dev
   .fc_flags = RTF_UP | RTF_NONEXTHOP,
   .fc_nlinfo.pid = 0,
   .fc_nlinfo.nlh = NULL,
-  .fc_nlinfo.net = &init_net,
+  .fc_nlinfo.net = dev->nd_net,
  };

  /* prefix length - 96 bits "::d.d.d.d" */
@@ -1681,7 +1682,7 @@ void addrconf_prefix_rcv(struct net_devi

  if (pinfo->onlink) {
   struct rt6_info *rt;
-  rt = rt6_lookup(&init_net, &pinfo->prefix, NULL, dev->ifindex, 1);
+  rt = rt6_lookup(dev->nd_net, &pinfo->prefix, NULL, dev->ifindex, 1);

   if (rt && ((rt->rt6i_flags & (RTF_GATEWAY | RTF_DEFAULT)) == 0)) {
```

```
    if (rt->rt6i_flags&RTF_EXPIRES) {
@@ -2044,6 +2045,7 @@ static void sit_add_v4_addrs(struct inet
  struct inet6_ifaddr * ifp;
  struct in6_addr addr;
  struct net_device *dev;
+ struct net *net = idev->dev->nd_net;
  int scope;

  ASSERT_RTNL();
@@ -2070,7 +2072,7 @@ static void sit_add_v4_addrs(struct inet
   return;
  }

- for_each_netdev(&init_net, dev) {
+ for_each_netdev(net, dev) {
   struct in_device * in_dev = __in_dev_get_rtnl(dev);
   if (in_dev && (dev->flags & IFF_UP)) {
    struct in_ifaddr * ifa;
@@ -2223,15 +2225,16 @@ ipv6_inherit_linklocal(struct inet6_dev
 static void ip6_tnl_add_linklocal(struct inet6_dev *idev)
 {
  struct net_device *link_dev;
+ struct net *net = idev->dev->nd_net;

  /* first try to inherit the link-local address from the link device */
  if (idev->dev->iflink &&
-     (link_dev = __dev_get_by_index(&init_net, idev->dev->iflink))) {
+     (link_dev = __dev_get_by_index(net, idev->dev->iflink))) {
   if (!ipv6_inherit_linklocal(idev, link_dev))
    return;
  }
  /* then try to inherit it from any device */
- for_each_netdev(&init_net, link_dev) {
+ for_each_netdev(net, link_dev) {
   if (!ipv6_inherit_linklocal(idev, link_dev))
    return;
  }
@@ -2264,9 +2267,6 @@ static int addrconf_notify(struct notifi
  int run_pending = 0;
  int err;

- if (dev->nd_net != &init_net)
-  return NOTIFY_DONE;
-
  switch(event) {
  case NETDEV_REGISTER:
   if (!idev && dev->mtu >= IPV6_MIN_MTU) {
@@ -3023,9 +3023,6 @@ inet6_rtm_deladdr(struct sk_buff *skb, s
```

```
  struct in6_addr *pfx;
  int err;

- if (net != &init_net)
-  return -EINVAL;
-
  err = nlmsg_parse(nlh, sizeof(*ifm), tb, IFA_MAX, ifa_ipv6_policy);
  if (err < 0)
   return err;
@@ -3088,9 +3085,6 @@ inet6_rtm_newaddr(struct sk_buff *skb, s
  u8 ifa_flags;
  int err;

- if (net != &init_net)
-  return -EINVAL;
-
  err = nlmsg_parse(nlh, sizeof(*ifm), tb, IFA_MAX, ifa_ipv6_policy);
  if (err < 0)
   return err;
@@ -3291,12 +3285,13 @@ static int inet6_dump_addr(struct sk_buf
  struct inet6_ifaddr *ifa;
  struct ifmcaddr6 *ifmca;
  struct ifacaddr6 *ifaca;
+ struct net *net = skb->sk->sk_net;

  s_idx = cb->args[0];
  s_ip_idx = ip_idx = cb->args[1];

  idx = 0;
- for_each_netdev(&init_net, dev) {
+ for_each_netdev(net, dev) {
   if (idx < s_idx)
    goto cont;
   if (idx > s_idx)
@@ -3365,35 +3360,23 @@ done:

 static int inet6_dump_ifaddr(struct sk_buff *skb, struct netlink_callback *cb)
 {
- struct net *net = skb->sk->sk_net;
  enum addr_type_t type = UNICAST_ADDR;

- if (net != &init_net)
-  return 0;
-
  return inet6_dump_addr(skb, cb, type);
 }

 static int inet6_dump_ifmcaddr(struct sk_buff *skb, struct netlink_callback *cb)
```

```
 {
- struct net *net = skb->sk->sk_net;
  enum addr_type_t type = MULTICAST_ADDR;

- if (net != &init_net)
-  return 0;
-
  return inet6_dump_addr(skb, cb, type);
 }


 static int inet6_dump_ifacaddr(struct sk_buff *skb, struct netlink_callback *cb)
 {
- struct net *net = skb->sk->sk_net;
  enum addr_type_t type = ANYCAST_ADDR;

- if (net != &init_net)
-  return 0;
-
  return inet6_dump_addr(skb, cb, type);
 }

@@ -3409,9 +3392,6 @@ static int inet6_rtm_getaddr(struct sk_b
  struct sk_buff *skb;
  int err;

- if (net != &init_net)
-  return -EINVAL;
-
  err = nlmsg_parse(nlh, sizeof(*ifm), tb, IFA_MAX, ifa_ipv6_policy);
  if (err < 0)
   goto errout;
@@ -3424,7 +3404,7 @@ static int inet6_rtm_getaddr(struct sk_b

  ifm = nlmsg_data(nlh);
  if (ifm->ifa_index)
-  dev = __dev_get_by_index(&init_net, ifm->ifa_index);
+  dev = __dev_get_by_index(net, ifm->ifa_index);

  if ((ifa = ipv6_get_ifaddr(net, addr, dev, 1)) == NULL) {
   err = -EADDRNOTAVAIL;
@@ -3444,7 +3424,7 @@ static int inet6_rtm_getaddr(struct sk_b
   kfree_skb(skb);
   goto errout_ifa;
  }
- err = rtnl_unicast(skb, &init_net, NETLINK_CB(in_skb).pid);
+ err = rtnl_unicast(skb, net, NETLINK_CB(in_skb).pid);
 errout_ifa:
```

```
  in6_ifa_put(ifa);
 errout:
@@ -3454,6 +3434,7 @@ errout:
 static void inet6_ifa_notify(int event, struct inet6_ifaddr *ifa)
 {
  struct sk_buff *skb;
+ struct net *net = ifa->idev->dev->nd_net;
  int err = -ENOBUFS;

  skb = nlmsg_new(inet6_ifaddr_msgsize(), GFP_ATOMIC);
@@ -3467,10 +3448,10 @@ static void inet6_ifa_notify(int event,
   kfree_skb(skb);
   goto errout;
  }
- err = rtnl_notify(skb, &init_net, 0, RTNLGRP_IPV6_IFADDR, NULL, GFP_ATOMIC);
+ err = rtnl_notify(skb, net, 0, RTNLGRP_IPV6_IFADDR, NULL, GFP_ATOMIC);
 errout:
  if (err < 0)
-  rtnl_set_sk_err(&init_net, RTNLGRP_IPV6_IFADDR, err);
+  rtnl_set_sk_err(net, RTNLGRP_IPV6_IFADDR, err);
 }

 static inline void ipv6_store_devconf(struct ipv6_devconf *cnf,
@@ -3635,12 +3616,9 @@ static int inet6_dump_ifinfo(struct sk_b
  struct net_device *dev;
  struct inet6_dev *idev;

- if (net != &init_net)
-  return 0;
-
  read_lock(&dev_base_lock);
  idx = 0;
- for_each_netdev(&init_net, dev) {
+ for_each_netdev(net, dev) {
   if (idx < s_idx)
    goto cont;
   if ((idev = in6_dev_get(dev)) == NULL)
@@ -3662,6 +3640,7 @@ cont:
 void inet6_ifinfo_notify(int event, struct inet6_dev *idev)
 {
  struct sk_buff *skb;
+ struct net *net = idev->dev->nd_net;
  int err = -ENOBUFS;

  skb = nlmsg_new(inet6_if_nlmsg_size(), GFP_ATOMIC);
@@ -3675,10 +3654,10 @@ void inet6_ifinfo_notify(int event, stru
   kfree_skb(skb);
   goto errout;
```

```
  }
- err = rtnl_notify(skb, &init_net, 0, RTNLGRP_IPV6_IFADDR, NULL, GFP_ATOMIC);
+ err = rtnl_notify(skb, net, 0, RTNLGRP_IPV6_IFADDR, NULL, GFP_ATOMIC);
 errout:
  if (err < 0)
-  rtnl_set_sk_err(&init_net, RTNLGRP_IPV6_IFADDR, err);
+  rtnl_set_sk_err(net, RTNLGRP_IPV6_IFADDR, err);
 }

 static inline size_t inet6_prefix_nlmsg_size(void)
@@ -3731,6 +3710,7 @@ static void inet6_prefix_notify(int even
     struct prefix_info *pinfo)
 {
  struct sk_buff *skb;
+ struct net *net = idev->dev->nd_net;
  int err = -ENOBUFS;

  skb = nlmsg_new(inet6_prefix_nlmsg_size(), GFP_ATOMIC);
@@ -3744,10 +3724,10 @@ static void inet6_prefix_notify(int even
   kfree_skb(skb);
   goto errout;
  }
- err = rtnl_notify(skb, &init_net, 0, RTNLGRP_IPV6_PREFIX, NULL, GFP_ATOMIC);
+ err = rtnl_notify(skb, net, 0, RTNLGRP_IPV6_PREFIX, NULL, GFP_ATOMIC);
 errout:
  if (err < 0)
-  rtnl_set_sk_err(&init_net, RTNLGRP_IPV6_PREFIX, err);
+  rtnl_set_sk_err(net, RTNLGRP_IPV6_PREFIX, err);
 }

 static void __ipv6_ifa_notify(int event, struct inet6_ifaddr *ifp)
@@ -4234,12 +4214,41 @@ EXPORT_SYMBOL(unregister_inet6addr_notif

 static int addrconf_net_init(struct net *net)
 {
-       return 0;
+ int err = 0;
+
+  net->ip6_null_entry->u.dst.dev = net->loopback_dev;
+  net->ip6_null_entry->rt6i_idev = in6_dev_get(net->loopback_dev);
+#ifdef CONFIG_IPV6_MULTIPLE_TABLES
+  net->ip6_prohibit_entry->u.dst.dev = net->loopback_dev;
+  net->ip6_prohibit_entry->rt6i_idev = in6_dev_get(net->loopback_dev);
+  net->ip6_blk_hole_entry->u.dst.dev = net->loopback_dev;
+  net->ip6_blk_hole_entry->rt6i_idev = in6_dev_get(net->loopback_dev);
+#endif
+  return err;
 }
```

```
 static void addrconf_net_exit(struct net *net)
 {
- ;
+ struct net_device *dev;
+
+        /*
+  * Remove loopback references from default routing entries
+  */
+ in6_dev_put(net->ip6_null_entry->rt6i_idev);
+#ifdef CONFIG_IPV6_MULTIPLE_TABLES
+ in6_dev_put(net->ip6_prohibit_entry->rt6i_idev);
+ in6_dev_put(net->ip6_blk_hole_entry->rt6i_idev);
+#endif
+
+ rtnl_lock();
+ /* clean dev list */
+ for_each_netdev(net, dev) {
+  if (__in6_dev_get(dev) == NULL)
+   continue;
+  addrconf_ifdown(dev, 1);
+ }
+ addrconf_ifdown(net->loopback_dev, 2);
+ rtnl_unlock();
 }

 static struct pernet_operations addrconf_net_ops = {
@@ -4261,40 +4270,31 @@ int __init addrconf_init(void)
   return err;
  }

- /* The addrconf netdev notifier requires that loopback_dev
-  * has it's ipv6 private information allocated and setup
-  * before it can bring up and give link-local addresses
-  * to other devices which are up.
-  *
-  * Unfortunately, loopback_dev is not necessarily the first
-  * entry in the global dev_base list of net devices.  In fact,
-  * it is likely to be the very last entry on that list.
-  * So this causes the notifier registry below to try and
-  * give link-local addresses to all devices besides loopback_dev
-  * first, then loopback_dev, which cases all the non-loopback_dev
-  * devices to fail to get a link-local address.
-  *
-  * So, as a temporary fix, allocate the ipv6 structure for
-  * loopback_dev first by hand.
-  * Longer term, all of the dependencies ipv6 has upon the loopback
-  * device and it being up should be removed.
```

```
-  */
- rtnl_lock();
- if (!ipv6_add_dev(init_net.loopback_dev))
-  err = -ENOMEM;
- rtnl_unlock();
- if (err)
+  /* The addrconf netdev notifier requires that loopback_dev
+   * has it's ipv6 private information allocated and setup
+   * before it can bring up and give link-local addresses
+   * to other devices which are up.
+   *
+   * Unfortunately, loopback_dev is not necessarily the first
+   * entry in the global dev_base list of net devices.  In fact,
+   * it is likely to be the very last entry on that list.
+   * So this causes the notifier registry below to try and
+   * give link-local addresses to all devices besides loopback_dev
+   * first, then loopback_dev, which cases all the non-loopback_dev
+   * devices to fail to get a link-local address.
+   *
+   * So, as a temporary fix, allocate the ipv6 structure for
+   * loopback_dev first by hand.
+   * Longer term, all of the dependencies ipv6 has upon the loopback
+   * device and it being up should be removed.
+   */
+  rtnl_lock();
+  if (!ipv6_add_dev(init_net.loopback_dev))
+   err = -ENOMEM;
+  rtnl_unlock();
+  if (err)
    return err;

- init_net.ip6_null_entry->u.dst.dev = init_net.loopback_dev;
- init_net.ip6_null_entry->rt6i_idev = in6_dev_get(init_net.loopback_dev);
-#ifdef CONFIG_IPV6_MULTIPLE_TABLES
- init_net.ip6_prohibit_entry->u.dst.dev = init_net.loopback_dev;
- init_net.ip6_prohibit_entry->rt6i_idev = in6_dev_get(init_net.loopback_dev);
- init_net.ip6_blk_hole_entry->u.dst.dev = init_net.loopback_dev;
- init_net.ip6_blk_hole_entry->rt6i_idev = in6_dev_get(init_net.loopback_dev);
-#endif
-
  err = register_pernet_device(&addrconf_net_ops);
  if (err)
   return err;
@@ -4332,7 +4332,6 @@ errout:

 void __exit addrconf_cleanup(void)
 {
- struct net_device *dev;
```

```
  struct inet6_ifaddr *ifa;
  int i;

@@ -4346,20 +4345,8 @@ void __exit addrconf_cleanup(void)
  rtnl_lock();

  /*
-  * clean dev list.
-  */
-
- for_each_netdev(&init_net, dev) {
-  if (__in6_dev_get(dev) == NULL)
-   continue;
-  addrconf_ifdown(dev, 1);
- }
- addrconf_ifdown(init_net.loopback_dev, 2);
-
- /*
   * Check hash table.
   */
-
  write_lock_bh(&addrconf_hash_lock);
  for (i=0; i < IN6_ADDR_HSIZE; i++) {
   for (ifa=inet6_addr_lst[i]; ifa; ) {

--
```

_____

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