Subject: [patch 2/2][NETNS49][IPV4][IGMP] make igmp per namespace
Posted by Daniel Lezcano on Fri, 12 Oct 2007 17:10:15 GMT
View Forum Message <> Reply to Message

This patch allows to create multicast sockets per namespaces

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

---
 net/ipv4/igmp.c | 37 ++++++++++++++++++++++----------
 1 file changed, 20 insertions(+), 17 deletions(-)

Index: linux-2.6-netns/net/ipv4/igmp.c
===
--- linux-2.6-netns.orig/net/ipv4/igmp.c
+++ linux-2.6-netns/net/ipv4/igmp.c
@@ -291,13 +291,14 @@ static struct sk_buff *igmpv3_newpack(st
  struct rtable *rt;
  struct iphdr *pip;
  struct igmpv3_report *pig;
+ struct net *net = dev->nd_net;

  skb = alloc_skb(size + LL_RESERVED_SPACE(dev), GFP_ATOMIC);
  if (skb == NULL)
   return NULL;

  {
- struct flowi fl = { .fl_net = &init_net,
+ struct flowi fl = { .fl_net = net,
         .oif = dev->ifindex,
         .nl_u = { .ip4_u = {
         .daddr = IGMPV3_ALL_MCR } },
@@ -637,6 +638,7 @@ static int igmp_send_report(struct in_de
  struct igmphdr *ih;
  struct rtable *rt;
  struct net_device *dev = in_dev->dev;
+ struct net *net = dev->nd_net;
  __be32 group = pmc ? pmc->multiaddr : 0;
  __be32 dst;

@@ -648,7 +650,7 @@ static int igmp_send_report(struct in_de
   dst = group;

  {
- struct flowi fl = { .fl_net = &init_net,
+ struct flowi fl = { .fl_net = net,
         .oif = dev->ifindex,
         .nl_u = { .ip4_u = { .daddr = dst } },

```
        .proto = IPPROTO_IGMP };
@@ -932,11 +934,6 @@ int igmp_rcv(struct sk_buff *skb)
  struct in_device *in_dev = in_dev_get(skb->dev);
  int len = skb->len;

- if (skb->dev->nd_net != &init_net) {
-  kfree_skb(skb);
-  return 0;
- }
-
  if (in_dev==NULL) {
   kfree_skb(skb);
   return 0;
@@ -1399,10 +1396,10 @@ void ip_mc_destroy_dev(struct in_device
  write_unlock_bh(&in_dev->mc_list_lock);
 }

-static struct in_device * ip_mc_find_dev(struct ip_mreqn *imr)
+static struct in_device * ip_mc_find_dev(struct net *net, struct ip_mreqn *imr)
 {
  struct flowi fl = {
-  .fl_net = &init_net,
+  .fl_net = net,
   .nl_u = { .ip4_u = { .daddr = imr->imr_multiaddr.s_addr } }
  };
  struct rtable *rt;
@@ -1410,13 +1407,13 @@ static struct in_device * ip_mc_find_dev
  struct in_device *idev = NULL;

  if (imr->imr_ifindex) {
-  idev = inetdev_by_index(&init_net, imr->imr_ifindex);
+  idev = inetdev_by_index(net, imr->imr_ifindex);
   if (idev)
    __in_dev_put(idev);
   return idev;
  }
  if (imr->imr_address.s_addr) {
-  dev = ip_dev_find(&init_net, imr->imr_address.s_addr);
+  dev = ip_dev_find(net, imr->imr_address.s_addr);
   if (!dev)
    return NULL;
   dev_put(dev);
@@ -1760,6 +1757,7 @@ int ip_mc_join_group(struct sock *sk , s
  struct ip_mc_socklist *iml=NULL, *i;
  struct in_device *in_dev;
  struct inet_sock *inet = inet_sk(sk);
+ struct net *net = sk->sk_net;
  int ifindex;
```

```
  int count = 0;

@@ -1768,7 +1766,7 @@ int ip_mc_join_group(struct sock *sk , s

  rtnl_lock();

- in_dev = ip_mc_find_dev(imr);
+ in_dev = ip_mc_find_dev(net, imr);

  if (!in_dev) {
   iml = NULL;
@@ -1830,12 +1828,13 @@ int ip_mc_leave_group(struct sock *sk, s
  struct inet_sock *inet = inet_sk(sk);
  struct ip_mc_socklist *iml, **imlp;
  struct in_device *in_dev;
+ struct net *net = sk->sk_net;
  __be32 group = imr->imr_multiaddr.s_addr;
  u32 ifindex;
  int ret = -EADDRNOTAVAIL;

  rtnl_lock();
- in_dev = ip_mc_find_dev(imr);
+ in_dev = ip_mc_find_dev(net, imr);
  ifindex = imr->imr_ifindex;
  for (imlp = &inet->mc_list; (iml = *imlp) != NULL; imlp = &iml->next) {
   if (iml->multi.imr_multiaddr.s_addr != group)
@@ -1873,6 +1872,7 @@ int ip_mc_source(int add, int omode, str
  struct in_device *in_dev = NULL;
  struct inet_sock *inet = inet_sk(sk);
  struct ip_sf_socklist *psl;
+ struct net *net = sk->sk_net;
  int leavegroup = 0;
  int i, j, rv;

@@ -1884,7 +1884,7 @@ int ip_mc_source(int add, int omode, str
  imr.imr_multiaddr.s_addr = mreqs->imr_multiaddr;
  imr.imr_address.s_addr = mreqs->imr_interface;
  imr.imr_ifindex = ifindex;
- in_dev = ip_mc_find_dev(&imr);
+ in_dev = ip_mc_find_dev(net, &imr);

  if (!in_dev) {
   err = -ENODEV;
@@ -2004,6 +2004,7 @@ int ip_mc_msfilter(struct sock *sk, stru
  struct in_device *in_dev;
  struct inet_sock *inet = inet_sk(sk);
  struct ip_sf_socklist *newpsl, *psl;
+ struct net *net = sk->sk_net;
```

```
  int leavegroup = 0;

  if (!MULTICAST(addr))
@@ -2017,7 +2018,7 @@ int ip_mc_msfilter(struct sock *sk, stru
  imr.imr_multiaddr.s_addr = msf->imsf_multiaddr;
  imr.imr_address.s_addr = msf->imsf_interface;
  imr.imr_ifindex = ifindex;
- in_dev = ip_mc_find_dev(&imr);
+ in_dev = ip_mc_find_dev(net, &imr);

  if (!in_dev) {
   err = -ENODEV;
@@ -2088,6 +2089,7 @@ int ip_mc_msfget(struct sock *sk, struct
  struct in_device *in_dev;
  struct inet_sock *inet = inet_sk(sk);
  struct ip_sf_socklist *psl;
+ struct net *net = sk->sk_net;

  if (!MULTICAST(addr))
   return -EINVAL;
@@ -2097,7 +2099,7 @@ int ip_mc_msfget(struct sock *sk, struct
  imr.imr_multiaddr.s_addr = msf->imsf_multiaddr;
  imr.imr_address.s_addr = msf->imsf_interface;
  imr.imr_ifindex = 0;
- in_dev = ip_mc_find_dev(&imr);
+ in_dev = ip_mc_find_dev(net, &imr);

  if (!in_dev) {
   err = -ENODEV;
@@ -2235,6 +2237,7 @@ void ip_mc_drop_socket(struct sock *sk)
 {
  struct inet_sock *inet = inet_sk(sk);
  struct ip_mc_socklist *iml;
+ struct net *net = sk->sk_net;

  if (inet->mc_list == NULL)
   return;
@@ -2244,7 +2247,7 @@ void ip_mc_drop_socket(struct sock *sk)
   struct in_device *in_dev;
   inet->mc_list = iml->next;

-  in_dev = inetdev_by_index(&init_net, iml->multi.imr_ifindex);
+  in_dev = inetdev_by_index(net, iml->multi.imr_ifindex);
   (void) ip_mc_leave_src(sk, iml, in_dev);
   if (in_dev != NULL) {
    ip_mc_dec_group(in_dev, iml->multi.imr_multiaddr.s_addr);

--
```

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