Subject: [patch 0/2][NETNS49][IPV4][IGMP] activate multicast per namespace
Posted by Daniel Lezcano on Fri, 12 Oct 2007 17:10:13 GMT
View Forum Message <> Reply to Message

The following patches activate the multicast sockets for
the namespaces. The results is a traffic going through
differents namespaces. So if there are several applications
listenning to the same multicast group/port, running in
different namespaces, they will receive multicast packets.

The following program helps to test that.

Note that the TTL field is set to 2 to avoid to packets to
be dropped while going through the network namespace.

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <unistd.h>
#include <netinet/in.h>
#include <arpa/inet.h>

int run(int server, int daddr, int dport, int saddr, int nbmsg, int delay, int timeout)
{
	int fd,i;
	struct ip_mreq mreq;
	struct sockaddr_in addr;

	socklen_t len = sizeof(addr);
	int val;

	memset(&addr,0,len);
	memset(&mreq, 0, sizeof(mreq));

	mreq.imr_multiaddr.s_addr  = daddr;
	mreq.imr_interface.s_addr  = saddr;

	addr.sin_family      = AF_INET;
	addr.sin_port        = dport;
	addr.sin_addr.s_addr  = INADDR_ANY;

	if ((fd = socket(AF_INET,SOCK_DGRAM,0)) == -1) {
		perror("socket");
		return 1;
	}

	if (server)
```

```
        if (bind(fd, (struct sockaddr*)&addr, sizeof(addr))) {
                perror("bind");
                return 1;
        }

    if (setsockopt(fd, IPPROTO_IP, IP_ADD_MEMBERSHIP, &mreq, sizeof(mreq))) {
        perror("setsockopt ADD_MEMBER_SHIP");
        return 1;
    }

    val = 2;
    if (setsockopt(fd, IPPROTO_IP, IP_MULTICAST_TTL, &val, sizeof(val))) {
        perror("setsockopt MULTICAST_TTL");
        return 1;
    }

    /* val = 0; */
    val = 1;
    if (setsockopt(fd, IPPROTO_IP, IP_MULTICAST_LOOP, &val, sizeof(val))) {
        perror("setsockopt MULTICAST_LOOP");
        return 1;
    }

    addr.sin_addr.s_addr  = daddr;

    if (server) {
        for (i = 0; i < nbmsg; i++) {
                alarm(timeout);
                if (recv(fd,&i,sizeof(i),0) == -1) {
                        perror("recv");
                        return 1;
                }
                fprintf(stderr, ".");
        }
    } else {
        for (i = 0; i < nbmsg; i++) {
                if (sendto(fd, &i, sizeof(i), MSG_CONFIRM,
                          (const struct sockaddr*)&addr, len) == -1) {
                        perror("sendto");
                        return 1;
                }
                usleep(delay);
        }
    }

    fprintf(stderr," - done.\n");
    return 0;
}
```

```
int main(int argc, char* argv[])
{
	in_addr_t dest = inet_addr("234.5.6.7");
	int port = htons(10000);

	return run(argc > 1, dest, port, 0, 1000, 50000, 60);
}
```

--

Subject: [patch 1/2][NETNS49][IPV4][IGMP] make igmp proc per namespace
Posted by Daniel Lezcano on Fri, 12 Oct 2007 17:10:14 GMT
View Forum Message <> Reply to Message

This patch makes the proc files: "igmp" and "mcfilter" per namespace.

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

```
---
 net/ipv4/igmp.c |   68 ++++++++++++++++++++++++++++++++++++++++++++-------
 1 file changed, 60 insertions(+), 8 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
@@ -2291,6 +2291,7 @@ int ip_check_mc(struct in_device *in_dev
 struct igmp_mc_iter_state {
   struct net_device *dev;
   struct in_device *in_dev;
+ struct net *net;
 };

 #define igmp_mc_seq_private(seq) ((struct igmp_mc_iter_state *)(seq)->private)
@@ -2299,9 +2300,10 @@ static inline struct ip_mc_list *igmp_mc
 {
   struct ip_mc_list *im = NULL;
   struct igmp_mc_iter_state *state = igmp_mc_seq_private(seq);
+ struct net *net = state->net;

   state->in_dev = NULL;
- for_each_netdev(&init_net, state->dev) {
```

```
+ for_each_netdev(net, state->dev) {
   struct in_device *in_dev;
   in_dev = in_dev_get(state->dev);
   if (!in_dev)
@@ -2426,31 +2428,47 @@ static int igmp_mc_seq_open(struct inode

  if (!s)
   goto out;
+
+ s->net = get_proc_net(inode);
+ if (!s->net)
+  goto out_kfree;
+
  rc = seq_open(file, &igmp_mc_seq_ops);
  if (rc)
-  goto out_kfree;
+  goto out_put_net;

  seq = file->private_data;
  seq->private = s;
 out:
  return rc;
+out_put_net:
+ put_net(s->net);
 out_kfree:
  kfree(s);
  goto out;
 }

+static int igmp_mc_seq_release(struct inode *inode, struct file *file)
+{
+ struct seq_file *seq = file->private_data;
+ struct igmp_mc_iter_state *state = seq->private;
+ put_net(state->net);
+ return seq_release_private(inode, file);
+}
+
 static const struct file_operations igmp_mc_seq_fops = {
  .owner  = THIS_MODULE,
  .open  = igmp_mc_seq_open,
  .read  = seq_read,
  .llseek  = seq_lseek,
- .release = seq_release_private,
+ .release = igmp_mc_seq_release,
 };

 struct igmp_mcf_iter_state {
  struct net_device *dev;
```

```
  struct in_device *idev;
  struct ip_mc_list *im;
+ struct net *net;
 };

 #define igmp_mcf_seq_private(seq) ((struct igmp_mcf_iter_state *)(seq)->private)
@@ -2460,10 +2478,11 @@ static inline struct ip_sf_list *igmp_mc
  struct ip_sf_list *psf = NULL;
  struct ip_mc_list *im = NULL;
  struct igmp_mcf_iter_state *state = igmp_mcf_seq_private(seq);
+ struct net *net = state->net;

  state->idev = NULL;
  state->im = NULL;
- for_each_netdev(&init_net, state->dev) {
+ for_each_netdev(net, state->dev) {
   struct in_device *idev;
   idev = in_dev_get(state->dev);
   if (unlikely(idev == NULL))
@@ -2600,31 +2619,64 @@ static int igmp_mcf_seq_open(struct inod

  if (!s)
   goto out;
+
+ s->net = get_proc_net(inode);
+ if (!s->net)
+  goto out_kfree;
+
  rc = seq_open(file, &igmp_mcf_seq_ops);
  if (rc)
-  goto out_kfree;
+  goto out_put_net;

  seq = file->private_data;
  seq->private = s;
 out:
  return rc;
+out_put_net:
+ put_net(s->net);
 out_kfree:
  kfree(s);
  goto out;
 }

+static int igmp_mcf_seq_release(struct inode *inode, struct file *file)
+{
+ struct seq_file *seq = file->private_data;
+ struct igmp_mcf_iter_state *state = seq->private;
```

```
+ put_net(state->net);
+ return seq_release_private(inode, file);
+
+}
+
 static const struct file_operations igmp_mcf_seq_fops = {
  .owner  = THIS_MODULE,
  .open  = igmp_mcf_seq_open,
  .read  = seq_read,
  .llseek  = seq_lseek,
- .release = seq_release_private,
+ .release = igmp_mcf_seq_release,
+};
+
+static int igmp_mc_net_init(struct net *net)
+{
+ proc_net_fops_create(net, "igmp", S_IRUGO, &igmp_mc_seq_fops);
+ proc_net_fops_create(net, "mcfilter", S_IRUGO, &igmp_mcf_seq_fops);
+ return 0;
+}
+
+static void igmp_mc_net_exit(struct net *net)
+{
+ proc_net_remove(net, "igmp");
+ proc_net_remove(net, "mcfilter");
+}
+
+struct pernet_operations igmp_mc_net_ops = {
+ .init = igmp_mc_net_init,
+ .exit = igmp_mc_net_exit,
 };

 int __init igmp_mc_proc_init(void)
 {
- proc_net_fops_create(&init_net, "igmp", S_IRUGO, &igmp_mc_seq_fops);
- proc_net_fops_create(&init_net, "mcfilter", S_IRUGO, &igmp_mcf_seq_fops);
+ register_pernet_subsys(&igmp_mc_net_ops);
  return 0;
 }
 #endif

--
```

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);

--
```

Subject: Re: [patch 0/2][NETNS49][IPV4][IGMP] activate multicast per namespace
Posted by ebiederm on Fri, 12 Oct 2007 18:50:20 GMT
View Forum Message <> Reply to Message

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

> The following patches activate the multicast sockets for
> the namespaces. The results is a traffic going through
> differents namespaces. So if there are several applications
> listenning to the same multicast group/port, running in
> different namespaces, they will receive multicast packets.

At a first glance this feels wrong. I don't see any per
namespace filtering of multicast traffic. Unless the
multicast traffic is routed/bridged between namespaces
it should be possible to send multicast traffic in one
namespace and listen for that same traffic in another
namespace and not get it.

Eric

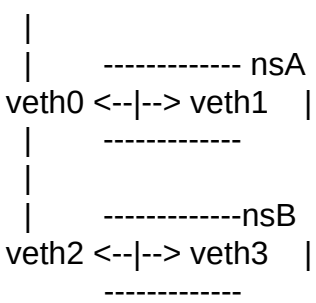
_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: Re: [patch 0/2][NETNS49][IPV4][IGMP] activate multicast per namespace
Posted by Daniel Lezcano on Fri, 12 Oct 2007 21:03:44 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> Daniel Lezcano <dlezcano@fr.ibm.com> writes:
>
>> The following patches activate the multicast sockets for
>> the namespaces. The results is a traffic going through
>> differents namespaces. So if there are several applications
>> listenning to the same multicast group/port, running in
>> different namespaces, they will receive multicast packets.
>
> At a first glance this feels wrong. I don't see any per
> namespace filtering of multicast traffic. Unless the

> multicast traffic is routed/bridged between namespaces
> it should be possible to send multicast traffic in one
> namespace and listen for that same traffic in another
> namespace and not get it.

The described behavior is the case were the namespaces are communicating
via veth like:



If an application is listening in nsA and nsB. And if in nsA, an
application sends multicast traffic, both will receive the packets
because they are routed by the pair device.
As you said this is the correct behavior, if we have two machines hostA
and hostB in the same network and both are listening on the multicast
address and if an application on hostA send multicast packets, both
should receive the multicast packets.
If the traffic is not routed, multicast will not pass through the
namespaces.

The description I gave in the patchset introduction was to describe such
behavior which is, IMHO, important for inter-container communication.
Perhaps, I should have not gave this description which seems to sow
confusion in mind, sorry for that.

Anyway, I hope the patchset is ok :)

Regards.

Daniel

Subject: Re: [patch 0/2][NETNS49][IPV4][IGMP] activate multicast per namespace
Posted by ebiederm on Fri, 12 Oct 2007 21:37:59 GMT
View Forum Message <> Reply to Message

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

> Eric W. Biederman wrote:
>> Daniel Lezcano <dlezcano@fr.ibm.com> writes:
>>
>>> The following patches activate the multicast sockets for
>>> the namespaces. The results is a traffic going through differents
>>> namespaces. So if there are several applications
>>> listenning to the same multicast group/port, running in
>>> different namespaces, they will receive multicast packets.
>>
>> At a first glance this feels wrong. I don't see any per
>> namespace filtering of multicast traffic. Unless the
>> multicast traffic is routed/bridged between namespaces
>> it should be possible to send multicast traffic in one
>> namespace and listen for that same traffic in another
>> namespace and not get it.
>
> The described behavior is the case were the namespaces are communicating via
> veth like:
>
> eth0
> |
> | ------------- nsA
> veth0 <--|--> veth1 |
> | -------------
> |
> | -------------nsB
> veth2 <--|--> veth3 |
>   -------------
>
>
> If an application is listening in nsA and nsB. And if in nsA, an application
> sends multicast traffic, both will receive the packets because they are routed
> by the pair device.
> As you said this is the correct behavior, if we have two machines hostA and
> hostB in the same network and both are listening on the multicast address and if
> an application on hostA send multicast packets, both should receive the
> multicast packets.
> If the traffic is not routed, multicast will not pass through the namespaces.
>
> The description I gave in the patchset introduction was to describe such
> behavior which is, IMHO, important for inter-container communication.
> Perhaps, I should have not gave this description which seems to sow confusion in
> mind, sorry for that.

>
> Anyway, I hope the patchset is ok :)

Sounds more reasonable. I didn't see the second patch when I replied
which was part of the reason I was worried. So at least at first glance
that patchset looks reasonable.

Thanks,
Eric

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

---

Subject: Re: Re: [patch 0/2][NETNS49][IPV4][IGMP] activate multicast per namespace
Posted by den on Mon, 15 Oct 2007 08:31:50 GMT
View Forum Message <> Reply to Message

Daniel Lezcano wrote:
> Eric W. Biederman wrote:
>> Daniel Lezcano <dlezcano@fr.ibm.com> writes:
>>
>>> The following patches activate the multicast sockets for
>>> the namespaces. The results is a traffic going through differents
>>> namespaces. So if there are several applications
>>> listenning to the same multicast group/port, running in
>>> different namespaces, they will receive multicast packets.
>>
>> At a first glance this feels wrong. I don't see any per
>> namespace filtering of multicast traffic. Unless the
>> multicast traffic is routed/bridged between namespaces
>> it should be possible to send multicast traffic in one
>> namespace and listen for that same traffic in another
>> namespace and not get it.
>
> The described behavior is the case were the namespaces are communicating
> via veth like:
>
> eth0
> |
> | ---------- nsA
> veth0 <--|--> veth1 |
> | ----------
> |
> | ----------nsB
> veth2 <--|--> veth3 |

> ----------
>
>
> If an application is listening in nsA and nsB. And if in nsA, an
> application sends multicast traffic, both will receive the packets
> because they are routed by the pair device.
> As you said this is the correct behavior, if we have two machines hostA
> and hostB in the same network and both are listening on the multicast
> address and if an application on hostA send multicast packets, both
> should receive the multicast packets.
> If the traffic is not routed, multicast will not pass through the
> namespaces.
>
> The description I gave in the patchset introduction was to describe such
> behavior which is, IMHO, important for inter-container communication.
> Perhaps, I should have not gave this description which seems to sow
> confusion in mind, sorry for that.
>
> Anyway, I hope the patchset is ok :)

hmm, by the way, will this work with macvlan?

also, I am dumb with multicasts :) who will clone the packet if there
are more than one namespace listen and there are some listeners behind
ethernet?

Regards,
Den

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

---

Subject: Re: Re: [patch 0/2][NETNS49][IPV4][IGMP] activate multicast per namespace
Posted by Daniel Lezcano on Mon, 15 Oct 2007 16:14:41 GMT
View Forum Message <> Reply to Message

Denis V. Lunev wrote:
> Daniel Lezcano wrote:
>> Eric W. Biederman wrote:
>>> Daniel Lezcano <dlezcano@fr.ibm.com> writes:
>>>
>>>> The following patches activate the multicast sockets for
>>>> the namespaces. The results is a traffic going through differents
>>>> namespaces. So if there are several applications
>>>> listenning to the same multicast group/port, running in

>>>> different namespaces, they will receive multicast packets.
>>>
>>> At a first glance this feels wrong. I don't see any per
>>> namespace filtering of multicast traffic. Unless the
>>> multicast traffic is routed/bridged between namespaces
>>> it should be possible to send multicast traffic in one
>>> namespace and listen for that same traffic in another
>>> namespace and not get it.
>>
>> The described behavior is the case were the namespaces are
>> communicating via veth like:
>>
>> eth0
>> |
>> | ------------- nsA
>> veth0 <--|--> veth1 |
>> | -------------
>> |
>> | -------------nsB
>> veth2 <--|--> veth3 |
>>   -------------
>>
>>
>> If an application is listening in nsA and nsB. And if in nsA, an
>> application sends multicast traffic, both will receive the packets
>> because they are routed by the pair device.
>> As you said this is the correct behavior, if we have two machines
>> hostA and hostB in the same network and both are listening on the
>> multicast address and if an application on hostA send multicast
>> packets, both should receive the multicast packets.
>> If the traffic is not routed, multicast will not pass through the
>> namespaces.
>>
>> The description I gave in the patchset introduction was to describe
>> such behavior which is, IMHO, important for inter-container
>> communication.
>> Perhaps, I should have not gave this description which seems to sow
>> confusion in mind, sorry for that.
>>
>> Anyway, I hope the patchset is ok :)
>
> hmm, by the way, will this work with macvlan?

I will check that. At the first glance, IMO it will not work if the
IP_MULTICAST_LOOP option is not set. Need to check ...

> also, I am dumb with multicasts :) who will clone the packet if there
> are more than one namespace listen and there are some listeners behind

> ethernet?

For local delivery, the function is:

__udp4_lib_mcast_deliver

For packet emission and loopbacking packets to ourself, it is:

ip_mc_output

For behind ethernet, the packet is multicasted to the network, so it is
up the peers to manage the packet.

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

---

Subject: Re: Re: [patch 0/2][NETNS49][IPV4][IGMP] activate multicast per namespace
Posted by ebiederm on Mon, 15 Oct 2007 18:03:22 GMT
View Forum Message <> Reply to Message

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

> Denis V. Lunev wrote:
>> Daniel Lezcano wrote:
>>> Eric W. Biederman wrote:
>>>> Daniel Lezcano <dlezcano@fr.ibm.com> writes:
>>>>
>>>>> The following patches activate the multicast sockets for
>>>>> the namespaces. The results is a traffic going through differents
>>>>> namespaces. So if there are several applications
>>>>> listenning to the same multicast group/port, running in
>>>>> different namespaces, they will receive multicast packets.
>>>>>
>>>> At a first glance this feels wrong. I don't see any per
>>>> namespace filtering of multicast traffic. Unless the
>>>> multicast traffic is routed/bridged between namespaces
>>>> it should be possible to send multicast traffic in one
>>>> namespace and listen for that same traffic in another
>>>> namespace and not get it.
>>>
>>> The described behavior is the case were the namespaces are communicating via
>>> veth like:
>>>
>>> eth0

>>> |
>>> | ------------- nsA
>>> veth0 <--|--> veth1 |
>>> | -------------
>>> |
>>> | -------------nsB
>>> veth2 <--|--> veth3 |
>>>   -------------
>>>
>>>
>>> If an application is listening in nsA and nsB. And if in nsA, an application
>>> sends multicast traffic, both will receive the packets because they are
>>> routed by the pair device.
>>> As you said this is the correct behavior, if we have two machines hostA and
>>> hostB in the same network and both are listening on the multicast address and
>>> if an application on hostA send multicast packets, both should receive the
>>> multicast packets.
>>> If the traffic is not routed, multicast will not pass through the namespaces.
>>>
>>> The description I gave in the patchset introduction was to describe such
>>> behavior which is, IMHO, important for inter-container communication.
>>> Perhaps, I should have not gave this description which seems to sow confusion
>>> in mind, sorry for that.
>>>
>>> Anyway, I hope the patchset is ok :)
>>
>> hmm, by the way, will this work with macvlan?
>
> I will check that. At the first glance, IMO it will not work if the
> IP_MULTICAST_LOOP option is not set. Need to check ...
>
>> also, I am dumb with multicasts :) who will clone the packet if there are more
>> than one namespace listen and there are some listeners behind ethernet?
>
> For local delivery, the function is:
>
> __udp4_lib_mcast_deliver
>
> For packet emission and loopbacking packets to ourself, it is:
>
> ip_mc_output
>
> For behind ethernet, the packet is multicasted to the network, so it is up the
> peers to manage the packet.

Right. So macvlan last I looked should handle the case of receiving an
external multicast transmission from ethernet fine as it replicates the
packet. However macvlan currently doesn't replicate the packet to other

macvlan devices on packet transmission, because with only a single namespace
the kernel will ensure local listeners get their multicast packets by
going through the loopback device. The same problem exists for broadcast
packets as well. So is likely we want to tweak the macvlan code just
a bit before we use it to heavily.

Eric

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