Subject: Re: [PATCH] Virtual ethernet tunnel
Posted by Patrick McHardy on Wed, 06 Jun 2007 15:28:22 GMT
View Forum Message <> Reply to Message

Pavel Emelianov wrote:
> Veth stands for Virtual ETHernet. It is a simple tunnel driver
> that works at the link layer and looks like a pair of ethernet
> devices interconnected with each other.
>
> Mainly it allows to communicate between network namespaces but
> it can be used as is as well.
>
> Eric recently sent a similar driver called etun. This
> implementation uses another interface - the RTM_NRELINK
> message introduced by Patric. The patch fits today netdev
> tree with Patrick's patches.
>
> The newlink callback is organized that way to make it easy
> to create the peer device in the separate namespace when we
> have them in kernel.
>

```
> +struct veth_priv {
> + struct net_device *peer;
> + struct net_device *dev;
> + struct list_head list;
> + struct net_device_stats stats;
```

You can use dev->stats instead.

```
> +static int veth_xmit(struct sk_buff *skb, struct net_device *dev)
> +{
> + struct net_device *rcv = NULL;
> + struct veth_priv *priv, *rcv_priv;
> + int length;
> +
> + skb_orphan(skb);
> +
> + priv = netdev_priv(dev);
> + rcv = priv->peer;
> + rcv_priv = netdev_priv(rcv);
> +
> + if (!(rcv->flags & IFF_UP))
> +  goto outf;
> +
> + skb->dev = rcv;
```

eth_type_trans already sets skb->dev.

```
> + skb->pkt_type = PACKET_HOST;
> + skb->protocol = eth_type_trans(skb, rcv);
> + if (dev->features & NETIF_F_NO_CSUM)
> +  skb->ip_summed = rcv_priv->ip_summed;
> +
> + dst_release(skb->dst);
> + skb->dst = NULL;
> +
> + secpath_reset(skb);
> + nf_reset(skb);
```

Is skb->mark supposed to survive communication between different namespaces?

```
> +static const struct nla_policy veth_policy[VETH_INFO_MAX] = {
> + [VETH_INFO_MAC]  = { .type = NLA_BINARY, .len = ETH_ALEN },
> + [VETH_INFO_PEER] = { .type = NLA_STRING },
> + [VETH_INFO_PEER_MAC] = { .type = NLA_BINARY, .len = ETH_ALEN },
> +};
```

The rtnl_link codes looks fine. I don't like the VETH_INFO_MAC attribute very much though, we already have a generic device attribute for MAC addresses. Of course that only allows you to supply one MAC address, so I'm wondering what you think of allocating only a single device per newlink operation and binding them in a seperate enslave operation?

```
> +enum {
> + VETH_INFO_UNSPEC,
> + VETH_INFO_MAC,
> + VETH_INFO_PEER,
> + VETH_INFO_PEER_MAC,
> +
> + VETH_INFO_MAX
> +};
```

Please follow the

```
#define VETH_INFO_MAX (__VETH_INFO_MAX - 1)
```

convention here.

____
Containers mailing list
Containers@lists.linux-foundation.org

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

Subject: Re: [PATCH] Virtual ethernet tunnel
Posted by Pavel Emelianov on Thu, 07 Jun 2007 08:09:01 GMT
View Forum Message <> Reply to Message

Patrick McHardy wrote:
> Pavel Emelianov wrote:
>> Veth stands for Virtual ETHernet. It is a simple tunnel driver
>> that works at the link layer and looks like a pair of ethernet
>> devices interconnected with each other.
>>
>> Mainly it allows to communicate between network namespaces but
>> it can be used as is as well.
>>
>> Eric recently sent a similar driver called etun. This
>> implementation uses another interface - the RTM_NRELINK
>> message introduced by Patric. The patch fits today netdev
>> tree with Patrick's patches.
>>
>> The newlink callback is organized that way to make it easy
>> to create the peer device in the separate namespace when we
>> have them in kernel.
>>
>
>> +struct veth_priv {
>> + struct net_device *peer;
>> + struct net_device *dev;
>> + struct list_head list;
>> + struct net_device_stats stats;
>
>
> You can use dev->stats instead.

OK. Actually I planned to use percpu stats to reduce cacheline
trashing (Stephen has noticed it also). The reason I didn't do it
here is that the patch would look more complicated, but I wanted to
show and approve the netlink interface first.

>> +static int veth_xmit(struct sk_buff *skb, struct net_device *dev)
>> +{
>> + struct net_device *rcv = NULL;
>> + struct veth_priv *priv, *rcv_priv;
>> + int length;
>> +
>> + skb_orphan(skb);
>> +

```
>> + priv = netdev_priv(dev);
>> + rcv = priv->peer;
>> + rcv_priv = netdev_priv(rcv);
>> +
>> + if (!(rcv->flags & IFF_UP))
>> +  goto outf;
>> +
>> + skb->dev = rcv;
```
>
> eth_type_trans already sets skb->dev.

Ok. Thanks.

```
>> + skb->pkt_type = PACKET_HOST;
>> + skb->protocol = eth_type_trans(skb, rcv);
>> + if (dev->features & NETIF_F_NO_CSUM)
>> +  skb->ip_summed = rcv_priv->ip_summed;
>> +
>> + dst_release(skb->dst);
>> + skb->dst = NULL;
>> +
>> + secpath_reset(skb);
>> + nf_reset(skb);
```
>
>
> Is skb->mark supposed to survive communication between different
> namespaces?

I guess it must not. Thanks.

```
>> +static const struct nla_policy veth_policy[VETH_INFO_MAX] = {
>> + [VETH_INFO_MAC]  = { .type = NLA_BINARY, .len = ETH_ALEN },
>> + [VETH_INFO_PEER] = { .type = NLA_STRING },
>> + [VETH_INFO_PEER_MAC] = { .type = NLA_BINARY, .len = ETH_ALEN },
>> +};
```
>
>
> The rtnl_link codes looks fine. I don't like the VETH_INFO_MAC attribute
> very much though, we already have a generic device attribute for MAC
> addresses. Of course that only allows you to supply one MAC address, so
> I'm wondering what you think of allocating only a single device per
> newlink operation and binding them in a seperate enslave operation?

I did this at the very first version, but Alexey showed me that this
would be wrong. Look. When we create the second device it must be in
the other namespace as it is useless to have them in one namespace.
But if we have the device in the other namespace the RTNL_NEWLINK
message from kernel would come into this namespace thus confusing ip

utility in the init namespace. Creating the device in the init ns and
moving it into the new one is rather a complex task.

But with such approach the creation looks really logical. We send a
packet to the kernel and have a single response about the new device
appearance. At the same time we have a RTNL_NEWLINK message arrived at
the destination namespace informing that a new device has appeared
there as well.

```
>> +enum {
>> + VETH_INFO_UNSPEC,
>> + VETH_INFO_MAC,
>> + VETH_INFO_PEER,
>> + VETH_INFO_PEER_MAC,
>> +
>> + VETH_INFO_MAX
>> +};
>
> Please follow the
>
> #define VETH_INFO_MAX (__VETH_INFO_MAX - 1)
>
> convention here.
```

Could you please clarify this point. I saw the lines

```
enum {
 ...
 RTNL_NEWLINK
#define RTNL_NEWLINK RTNL_NEWLINK
 ...
}
```

and had my brains exploded imagining what this would mean :(

Subject: Re: Re: [PATCH] Virtual ethernet tunnel
Posted by davem on Thu, 07 Jun 2007 09:07:33 GMT

View Forum Message <> Reply to Message

From: Kirill Korotaev <dev@sw.ru>
Date: Thu, 07 Jun 2007 12:14:29 +0400

> David Miller wrote:
> > From: Pavel Emelianov <xemul@openvz.org>
> > Date: Wed, 06 Jun 2007 19:11:38 +0400
> >
> >
> >>Veth stands for Virtual ETHernet. It is a simple tunnel driver
> >>that works at the link layer and looks like a pair of ethernet
> >>devices interconnected with each other.
> >
> >
> > I would suggest choosing a different name.
> >
> > 'veth' is also the name of the virtualized ethernet device
> > found on IBM machines, driven by driver/net/ibmveth.[ch]
>
> AFAICS, ibmveth.c registers ethX devices, while this driver registers
> vethX by default, so there is no much conflict IMHO.

If that's the case, veth is fine with me.

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

Subject: Re: Re: [PATCH] Virtual ethernet tunnel
Posted by Benjamin Thery on Thu, 07 Jun 2007 09:30:22 GMT

View Forum Message <> Reply to Message

David Miller wrote:
> From: Kirill Korotaev <dev@sw.ru>
> Date: Thu, 07 Jun 2007 12:14:29 +0400
>
>> David Miller wrote:
>>> From: Pavel Emelianov <xemul@openvz.org>
>>> Date: Wed, 06 Jun 2007 19:11:38 +0400
>>>
>>>
>>>> Veth stands for Virtual ETHernet. It is a simple tunnel driver
>>>> that works at the link layer and looks like a pair of ethernet
>>>> devices interconnected with each other.
>>>
>>> I would suggest choosing a different name.

>>>
>>> 'veth' is also the name of the virtualized ethernet device
>>> found on IBM machines, driven by driver/net/ibmveth.[ch]
>> AFAICS, ibmveth.c registers ethX devices, while this driver registers
>> vethX by default, so there is no much conflict IMHO.
>
> If that's the case, veth is fine with me.

I like Daniel's proposals with the tunnel or pipe thing in the name.
I think it is more explicit about what the device really is.

I'm currently using etun, Eric Biederman's implementation. It will be
nice to have this kind of device merged.

-- Benjamin

--
B e n j a m i n   T h e r y - BULL/DT/Open Software R&D

   http://www.bull.com
____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: Re: [PATCH] Virtual ethernet tunnel
Posted by Patrick McHardy on Mon, 11 Jun 2007 11:39:18 GMT
View Forum Message <> Reply to Message

Pavel Emelianov wrote:
> Patrick McHardy wrote:
>

```
>>>+ skb->pkt_type = PACKET_HOST;
>>>+ skb->protocol = eth_type_trans(skb, rcv);
>>>+ if (dev->features & NETIF_F_NO_CSUM)
>>>+  skb->ip_summed = rcv_priv->ip_summed;
>>>+
>>>+ dst_release(skb->dst);
>>>+ skb->dst = NULL;
>>>+
>>>+ secpath_reset(skb);
>>>+ nf_reset(skb);
```
>>
>>
>>Is skb->mark supposed to survive communication between different
>>namespaces?

>
>
> I guess it must not. Thanks.

I guess there are a few others that should be cleared as well,
like the tc related members, secmark, ipvs_property, ...

>>The rtnl_link codes looks fine. I don't like the VETH_INFO_MAC attribute
>>very much though, we already have a generic device attribute for MAC
>>addresses. Of course that only allows you to supply one MAC address, so
>>I'm wondering what you think of allocating only a single device per
>>newlink operation and binding them in a seperate enslave operation?
>
>
> I did this at the very first version, but Alexey showed me that this
> would be wrong. Look. When we create the second device it must be in
> the other namespace as it is useless to have them in one namespace.
> But if we have the device in the other namespace the RTNL_NEWLINK
> message from kernel would come into this namespace thus confusing ip
> utility in the init namespace. Creating the device in the init ns and
> moving it into the new one is rather a complex task.
>
> But with such approach the creation looks really logical. We send a
> packet to the kernel and have a single response about the new device
> appearance. At the same time we have a RTNL_NEWLINK message arrived at
> the destination namespace informing that a new device has appeared
> there as well.

The question is how to proceed. I haven't read all mails yet, but it
seems there is some disagreement about whether to create all devices
in the same namespace and move them later or create them directly in
their target namespace. For now I guess it doesn't matter much, so
can everyone agree to adding a IFLA_PARTNER attribute that includes
a complete ifinfomsg and the attributes and you later decide how to
handle namespaces?

```
>>>+enum {
>>>+ VETH_INFO_UNSPEC,
>>>+ VETH_INFO_MAC,
>>>+ VETH_INFO_PEER,
>>>+ VETH_INFO_PEER_MAC,
>>>+
>>>+ VETH_INFO_MAX
>>>+};
```
>>
>>Please follow the

```
>>
>>#define VETH_INFO_MAX (__VETH_INFO_MAX - 1)
>>
>>convention here.
>
>
> Could you please clarify this point. I saw the lines
> enum {
>  ...
>  RTNL_NEWLINK
> #define RTNL_NEWLINK RTNL_NEWLINK
>  ...
> }
> and had my brains exploded imagining what this would mean :(
```

Thats just to make the new attributes visible as preprocessor
symbols so userspace can use them for #ifdefs. We usually use
it when adding new attributes/message types, but its not necessary
for the initial set of attributes if you already have some other
preprocessor-visisble symbol (like VETH_INFO_MAX) userspace can
use.

What I was refering to is this convention:

```
enum {
...
	__IFLA_MAX
};

#define IFLA_MAX (__IFLA_MAX - 1)
```

Which is used to make sure that IFLA_MAX is really the max and
not max + 1 and additionally people won't forget to update it.

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

Subject: Re: [PATCH] Virtual ethernet tunnel
Posted by Pavel Emelianov on Wed, 13 Jun 2007 09:24:55 GMT
View Forum Message <> Reply to Message

Patrick McHardy wrote:
> Pavel Emelianov wrote:
>> Patrick McHardy wrote:
>>

>
>>>> + skb->pkt_type = PACKET_HOST;
>>>> + skb->protocol = eth_type_trans(skb, rcv);
>>>> + if (dev->features & NETIF_F_NO_CSUM)
>>>> + skb->ip_summed = rcv_priv->ip_summed;
>>>> +
>>>> + dst_release(skb->dst);
>>>> + skb->dst = NULL;
>>>> +
>>>> + secpath_reset(skb);
>>>> + nf_reset(skb);
>>>
>>> Is skb->mark supposed to survive communication between different
>>> namespaces?
>>
>> I guess it must not. Thanks.
>
>
> I guess there are a few others that should be cleared as well,
> like the tc related members, secmark, ipvs_property, ...

It seems like we are about to have some skb_reset_all() routine to
make the skb look like newborn.

>>> The rtnl_link codes looks fine. I don't like the VETH_INFO_MAC attribute
>>> very much though, we already have a generic device attribute for MAC
>>> addresses. Of course that only allows you to supply one MAC address, so
>>> I'm wondering what you think of allocating only a single device per
>>> newlink operation and binding them in a seperate enslave operation?
>>
>> I did this at the very first version, but Alexey showed me that this
>> would be wrong. Look. When we create the second device it must be in
>> the other namespace as it is useless to have them in one namespace.
>> But if we have the device in the other namespace the RTNL_NEWLINK
>> message from kernel would come into this namespace thus confusing ip
>> utility in the init namespace. Creating the device in the init ns and
>> moving it into the new one is rather a complex task.
>>
>> But with such approach the creation looks really logical. We send a
>> packet to the kernel and have a single response about the new device
>> appearance. At the same time we have a RTNL_NEWLINK message arrived at
>> the destination namespace informing that a new device has appeared
>> there as well.
>
>
> The question is how to proceed. I haven't read all mails yet, but it
> seems there is some disagreement about whether to create all devices
> in the same namespace and move them later or create them directly in

The agreement was that we can make any of the above. We can create
booth devices in the init namespace and then move one of them into the
desired namespace, or we can explicitly specify which namespace to create
the pair in.

> their target namespace. For now I guess it doesn't matter much, so
> can everyone agree to adding a IFLA_PARTNER attribute that includes
> a complete ifinfomsg and the attributes and you later decide how to
> handle namespaces?
>
>>>> +enum {
>>>> + VETH_INFO_UNSPEC,
>>>> + VETH_INFO_MAC,
>>>> + VETH_INFO_PEER,
>>>> + VETH_INFO_PEER_MAC,
>>>> +
>>>> + VETH_INFO_MAX
>>>> +};
>>> Please follow the
>>>
>>> #define VETH_INFO_MAX (__VETH_INFO_MAX - 1)
>>>
>>> convention here.
>>
>> Could you please clarify this point. I saw the lines
>> enum {
>> ...
>> RTNL_NEWLINK
>> #define RTNL_NEWLINK RTNL_NEWLINK
>> ...
>> }
>> and had my brains exploded imagining what this would mean :(
>
>
> Thats just to make the new attributes visible as preprocessor
> symbols so userspace can use them for #ifdefs. We usually use
> it when adding new attributes/message types, but its not necessary
> for the initial set of attributes if you already have some other
> preprocessor-visisble symbol (like VETH_INFO_MAX) userspace can
> use.
>
> What I was refering to is this convention:
>
> enum {
> ...
>   __IFLA_MAX
> };

>
> #define IFLA_MAX (__IFLA_MAX - 1)
>
> Which is used to make sure that IFLA_MAX is really the max and
> not max + 1 and additionally people won't forget to update it.

OK thanks. This is already done in the v2.

Thanks,
Pavel

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