Subject: Re: [PATCH] Virtual ethernet tunnel
Posted by Daniel Lezcano on Thu, 07 Jun 2007 09:29:50 GMT
View Forum Message <> Reply to Message

Pavel Emelianov wrote:
> 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.
>

Pavel,

moving the netdevice to another namespace is not a complex task. Eric Biederman did it in its patchset (cf. http://lxc.sf.net/network)

When the pair device is created, both extremeties are into the init namespace and you can choose to which namespace to move one extremity. When the network namespace dies, the netdev is moved back to the init namespace.
That facilitate network device management.

Concerning netlink events, this is automatically generated when the network device is moved through namespaces.

IMHO, we should have the network device movement between namespaces in order to be able to move a physical network device too (eg. you have 4 NIC and you want to create 3 containers and assign 3 NIC to each of them)

> 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 :(
>
>
>> -
>> To unsubscribe from this list: send the line "unsubscribe netdev" in
>> the body of a message to majordomo@vger.kernel.org
>> More majordomo info at http://vger.kernel.org/majordomo-info.html
>>
>>
>
> ____
> Containers mailing list
> Containers@lists.linux-foundation.org
> https://lists.linux-foundation.org/mailman/listinfo/containers
>
>

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