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