Subject: Re: [PATCH] net: Add etun driver
Posted by Stephen Hemminger on Fri, 06 Apr 2007 20:34:20 GMT
View Forum Message <> Reply to Message

Why not implement a true virtual network rather than simple
tunnel pairs?

Details:

> +static struct {
> + const char string[ETH_GSTRING_LEN];
> +} ethtool_stats_keys[ETUN_NUM_STATS] = {
> + { "partner_ifindex" },
> +};

You should use sysfs attributes for this rather than
ethtool.

> +struct etun_info {
> + struct net_device *rx_dev;
> + unsigned ip_summed;
> + struct net_device_stats stats;
> + struct list_head list;
> + struct net_device *dev;
> +};

> +static struct net_device_stats *etun_get_stats(struct net_device *dev)
> +{
> + struct etun_info *info = dev->priv;
> + return &info->stats;
> +}
> +
> +/* ethtool interface */
> +static int etun_get_settings(struct net_device *dev, struct ethtool_cmd *cmd)
> +{
> + cmd->supported = 0;
> + cmd->advertising = 0;
> + cmd->speed = SPEED_10000; /* Memory is fast! */
> + cmd->duplex = DUPLEX_FULL;
> + cmd->port = PORT_TP;
> + cmd->phy_address = 0;
> + cmd->transceiver = XCVR_INTERNAL;
> + cmd->autoneg = AUTONEG_DISABLE;
> + cmd->maxtxpkt = 0;
> + cmd->maxrxpkt = 0;
> + return 0;

```
> +}
> +
> +static void etun_get_drvinfo(struct net_device *dev, struct ethtool_drvinfo *info)
> +{
> + strcpy(info->driver, DRV_NAME);
> + strcpy(info->version, DRV_VERSION);
> + strcpy(info->fw_version, "N/A");
> +}
> +
> +static void etun_get_strings(struct net_device *dev, u32 stringset, u8 *buf)
> +{
> + switch(stringset) {
> + case ETH_SS_STATS:
> +  memcpy(buf, &ethtool_stats_keys, sizeof(ethtool_stats_keys));
> +  break;
> + case ETH_SS_TEST:
> + default:
> +  break;
> + }
> +}
> +
> +static int etun_get_stats_count(struct net_device *dev)
> +{
> + return ETUN_NUM_STATS;
> +}
> +
> +static void etun_get_ethtool_stats(struct net_device *dev,
> + struct ethtool_stats *stats, u64 *data)
> +{
> + struct etun_info *info = dev->priv;
> +
> + data[0] = info->rx_dev->ifindex;
> +}
> +
> +static u32 etun_get_rx_csum(struct net_device *dev)
> +{
> + struct etun_info *info = dev->priv;
> + return info->ip_summed == CHECKSUM_UNNECESSARY;
> +}
> +
> +static int etun_set_rx_csum(struct net_device *dev, u32 data)
> +{
> + struct etun_info *info = dev->priv;
> +
> + info->ip_summed = data ? CHECKSUM_UNNECESSARY : CHECKSUM_NONE;
> +
> + return 0;
> +}
```

```
> +
> +static u32 etun_get_tx_csum(struct net_device *dev)
> +{
> + return (dev->features & NETIF_F_NO_CSUM) != 0;
> +}
> +
> +static int etun_set_tx_csum(struct net_device *dev, u32 data)
> +{
> + dev->features &= ~NETIF_F_NO_CSUM;
> + if (data)
> +  dev->features |= NETIF_F_NO_CSUM;
> +
> + return 0;
> +}
> +
> +static struct ethtool_ops etun_ethtool_ops = {
> + .get_settings  = etun_get_settings,
> + .get_drvinfo  = etun_get_drvinfo,
> + .get_link  = ethtool_op_get_link,
> + .get_rx_csum  = etun_get_rx_csum,
> + .set_rx_csum  = etun_set_rx_csum,
> + .get_tx_csum  = etun_get_tx_csum,
> + .set_tx_csum  = etun_set_tx_csum,
> + .get_sg   = ethtool_op_get_sg,
> + .set_sg   = ethtool_op_set_sg,
> +#if 0 /* Does just setting the bit successfuly emulate tso? */
> + .get_tso  = ethtool_op_get_tso,
> + .set_tso  = ethtool_op_set_tso,
> +#endif
> + .get_strings  = etun_get_strings,
> + .get_stats_count = etun_get_stats_count,
> + .get_ethtool_stats = etun_get_ethtool_stats,
> + .get_perm_addr  = ethtool_op_get_perm_addr,
> +};
> +
> +static int etun_open(struct net_device *tx_dev)
> +{
> + struct etun_info *tx_info = tx_dev->priv;
> + struct net_device *rx_dev = tx_info->rx_dev;
> + /* If we attempt to bring up etun in the small window before
> +  * it is connected to it's partner error.
> +  */
> + if (!rx_dev)
> +  return -ENOTCONN;
> + if (rx_dev->flags & IFF_UP) {
> +  netif_carrier_on(tx_dev);
> +  netif_carrier_on(rx_dev);
> + }
```

```
> + netif_start_queue(tx_dev);
> + return 0;
> +}
> +
> +static int etun_stop(struct net_device *tx_dev)
> +{
> + struct etun_info *tx_info = tx_dev->priv;
> + struct net_device *rx_dev = tx_info->rx_dev;
> + netif_stop_queue(tx_dev);
> + if (netif_carrier_ok(tx_dev)) {
> +  netif_carrier_off(tx_dev);
> +  netif_carrier_off(rx_dev);
> + }
> + return 0;
> +}
> +
> +static int etun_change_mtu(struct net_device *dev, int new_mtu)
> +{
> + /* Don't allow ridiculously small mtus */
> + if (new_mtu < (ETH_ZLEN - ETH_HLEN))
> +  return -EINVAL;
> + dev->mtu = new_mtu;
> + return 0;
> +}
> +
> +static void etun_set_multicast_list(struct net_device *dev)
> +{
> + /* Nothing sane I can do here */
> + return;
> +}
> +
> +static int etun_ioctl(struct net_device *dev, struct ifreq *rq, int cmd)
> +{
> + return -EOPNOTSUPP;
> +}
```

If not supported, then no stub needed just leave null.

```
> +/* Only allow letters and numbers in an etun device name */
> +static int is_valid_name(const char *name)
> +{
> + const char *ptr;
> + for (ptr = name; *ptr; ptr++) {
> +  if (!isalnum(*ptr))
> +   return 0;
> + }
> + return 1;
> +}
```

```
> +
> +static struct net_device *etun_alloc(const char *name)
> +{
> + struct net_device *dev;
> + struct etun_info *info;
> + int err;
> +
> + if (!name || !is_valid_name(name))
> +  return ERR_PTR(-EINVAL);
> +
> + dev = alloc_netdev(sizeof(struct etun_info), name, ether_setup);
> + if (!dev)
> +  return ERR_PTR(-ENOMEM);
```

Use alloc_etherdev() instead.

```
> + info = dev->priv;
> + info->dev = dev;
> +
> + random_ether_addr(dev->dev_addr);
> + dev->tx_queue_len = 0; /* A queue is silly for a loopback device */
> + dev->hard_start_xmit = etun_xmit;
> + dev->get_stats  = etun_get_stats;
> + dev->open  = etun_open;
> + dev->stop  = etun_stop;
> + dev->set_multicast_list = etun_set_multicast_list;
> + dev->do_ioctl  = etun_ioctl;
> + dev->features  = NETIF_F_FRAGLIST
> +      | NETIF_F_HIGHDMA
> +      | NETIF_F_LLTX;
> + dev->flags  = IFF_BROADCAST | IFF_MULTICAST |IFF_PROMISC;
> + dev->ethtool_ops = &etun_ethtool_ops;
> + dev->destructor  = free_netdev;
> + dev->change_mtu  = etun_change_mtu;
> + err = register_netdev(dev);
> + if (err) {
> +  free_netdev(dev);
> +  dev = ERR_PTR(err);
> +  goto out;
> + }
> + netif_carrier_off(dev);
> +out:
> + return dev;
> +}
> +
> +static int etun_alloc_pair(const char *name0, const char *name1)
> +{
> + struct net_device *dev0, *dev1;
```

```
> + struct etun_info *info0, *info1;
> +
> + dev0 = etun_alloc(name0);
> + if (IS_ERR(dev0)) {
> +  return PTR_ERR(dev0);
> + }
> + info0 = dev0->priv;
> +
> + dev1 = etun_alloc(name1);
> + if (IS_ERR(dev1)) {
> +  unregister_netdev(dev0);
> +  return PTR_ERR(dev1);
> + }
> + info1 = dev1->priv;
> +
> + dev_hold(dev0);
> + dev_hold(dev1);
> + info0->rx_dev = dev1;
> + info1->rx_dev = dev0;
> +
> + /* Only place one member of the pair on the list
> +  * so I don't confuse list_for_each_entry_safe,
> +  * by deleting two list entries at once.
> +  */
> + rtnl_lock();
> + list_add(&info0->list, &etun_list);
> + INIT_LIST_HEAD(&info1->list);
> + rtnl_unlock();
> +
> + return 0;
> +}
> +
> +static int etun_unregister_pair(struct net_device *dev0)
> +{
> + struct etun_info *info0, *info1;
> + struct net_device *dev1;
> +
> + ASSERT_RTNL();
> +
> + if (!dev0)
> +  return -ENODEV;
> +
> + /* Ensure my network devices are not passing packets */
> + dev_close(dev0);
> + info0 = dev0->priv;
> + dev1  = info0->rx_dev;
> + info1 = dev1->priv;
> + dev_close(dev1);
```

```
> +
> + /* Drop the cross device references */
> + dev_put(dev0);
> + dev_put(dev1);
> +
> + /* Remove from the etun list */
> + if (!list_empty(&info0->list))
> +  list_del_init(&info0->list);
> + if (!list_empty(&info1->list))
> +  list_del_init(&info1->list);
> +
> + unregister_netdevice(dev0);
> + unregister_netdevice(dev1);
> + return 0;
> +}
> +
> +static int etun_noget(char *buffer, struct kernel_param *kp)
> +{
> + return 0;
> +}
> +
> +static int etun_newif(const char *val, struct kernel_param *kp)
> +{
> + char name0[IFNAMSIZ], name1[IFNAMSIZ];
> + const char *mid;
> + int len, len0, len1;
> + if (!capable(CAP_NET_ADMIN))
> +  return -EPERM;
> +
> + /* Avoid frustration by removing trailing whitespace */
> + len = strlen(val);
> + while (isspace(val[len - 1]))
> +  len--;
> +
> + /* Split the string into 2 names */
> + mid = memchr(val, ',', len);
> + if (!mid)
> +  return -EINVAL;
> +
> + /* Get the first device name */
> + len0 = mid - val;
> + if (len0 > sizeof(name0) - 1)
> +  len = sizeof(name0) - 1;
> + strncpy(name0, val, len0);
> + name0[len0] = '\0';
> +
> + /* And the second device name */
> + len1 = len - (len0 + 1);
```

```
> + if (len1 > sizeof(name1) - 1)
> +  len1 = sizeof(name1) - 1;
> + strncpy(name1, mid + 1, len1);
> + name1[len1] = '\0';
> +
> + return etun_alloc_pair(name0, name1);
> +}
> +
> +static int etun_delif(const char *val, struct kernel_param *kp)
> +{
> + char name[IFNAMSIZ];
> + int len;
> + struct net_device *dev;
> + int err;
> + if (!capable(CAP_NET_ADMIN))
> +  return -EPERM;
> +
> + /* Avoid frustration by removing trailing whitespace */
> + len = strlen(val);
> + while (isspace(val[len - 1]))
> +  len--;
> +
> + /* Get the device name */
> + if (len > sizeof(name) - 1)
> +  return -EINVAL;
> + strncpy(name, val, len);
> + name[len] = '\0';
> +
> + /* Double check I don't have strange characters in my device name */
> + if (!is_valid_name(name))
> +  return -EINVAL;
> +
> + rtnl_lock();
> + err = -ENODEV;
> + dev = __dev_get_by_name(name);
> + err = etun_unregister_pair(dev);
> + rtnl_unlock();
> + return err;
> +}
```

Doing create/delete via module parameter is wierd.
Why not either use an ioctl, or sysfs.

```
> +static int __init etun_init(void)
> +{
> + printk(KERN_INFO "etun: %s, %s\n", DRV_DESCRIPTION, DRV_VERSION);
> + printk(KERN_INFO "etun: %s\n", DRV_COPYRIGHT);
```

> +
> + return 0;

Why bother it is just advertising noise...

> +}
> +
> +static void etun_cleanup(void)
> +{
> + struct etun_info *info, *tmp;
> + rtnl_lock();
> + list_for_each_entry_safe(info, tmp, &etun_list, list) {
> + etun_unregister_pair(info->dev);
> + }
> + rtnl_unlock();
> +}
> +
> +module_param_call(newif, etun_newif, etun_noget, NULL, S_IWUSR);
> +module_param_call(delif, etun_delif, etun_noget, NULL, S_IWUSR);
> +module_init(etun_init);
> +module_exit(etun_cleanup);
> +MODULE_DESCRIPTION(DRV_DESCRIPTION);
> +MODULE_AUTHOR("Eric Biederman <ebiederm@xmission.com>");
> +MODULE_LICENSE("GPL");

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

---

Subject: Re: [PATCH] net: Add etun driver
Posted by Ben Greear on Fri, 06 Apr 2007 21:38:50 GMT
View Forum Message <> Reply to Message

Stephen Hemminger wrote:
> Why not implement a true virtual network rather than simple
> tunnel pairs?
>
What would a true virtual network do? You mean with routers and such?

I use my redirect device (basically same as etun) to join virtual
routers together,
but all of the virtual routing (and bridging, etc) logic is already in
the kernel
and just has to be configured properly....

Ben

--
Ben Greear <greearb@candelatech.com>
Candela Technologies Inc http://www.candelatech.com

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

---

Subject: Re: [PATCH] net: Add etun driver
Posted by ebiederm on Sat, 07 Apr 2007 02:51:58 GMT
View Forum Message <> Reply to Message

Stephen Hemminger <shemminger@linux-foundation.org> writes:

> Why not implement a true virtual network rather than simple
> tunnel pairs?

Mostly I don't even need etun. It is a cheap way to make up for
the lack of sufficient physical network adapters on the machine.
If you plug in enough network cards into the machine my network
namespace code works just fine without it.

etun is just a way to use the current bridging and routing code,
without a lot of complexity.

I think what I really want is something like your current ethernet
vlan code that worked at the mac address level. While handling
broadcasts and configuration similar to our current ethernet
bridging code.

However truly advanced low overhead things that require touching the
driver or are specific to a particular kind of networking are never
universally available so I need something that is.

With a little luck and a couple of tweaks as yet undetermined tweaks
to the etun driver and you won't be able to measure any overhead so it
will be enough. I doubt it but I can always hope.

Not doing anything extra and using a tool building approach puts
emphasis on improving our ethernet bridgine and our ip routing code.

> Details:
>
>

>> +static struct {
>> + const char string[ETH_GSTRING_LEN];
>> +} ethtool_stats_keys[ETUN_NUM_STATS] = {
>> + { "partner_ifindex" },
>> +};
>
>
> You should use sysfs attributes for this rather than
> ethtool.

I think it is six of one half a dozen of the other. It is easy
enough to change if need be.

>
>> +static int etun_ioctl(struct net_device *dev, struct ifreq *rq, int cmd)
>> +{
>> + return -EOPNOTSUPP;
>> +}
>
> If not supported, then no stub needed just leave null.

It's a place holder in case I need to support some ioctl. Having it
hear makes it clear that I don't support anything right now by design.

>> +/* Only allow letters and numbers in an etun device name */
>> +static int is_valid_name(const char *name)
>> +{
>> + const char *ptr;
>> + for (ptr = name; *ptr; ptr++) {
>> + if (!isalnum(*ptr))
>> +   return 0;
>> + }
>> + return 1;
>> +}
>> +
>> +static struct net_device *etun_alloc(const char *name)
>> +{
>> + struct net_device *dev;
>> + struct etun_info *info;
>> + int err;
>> +
>> + if (!name || !is_valid_name(name))
>> + return ERR_PTR(-EINVAL);
>> +
>> + dev = alloc_netdev(sizeof(struct etun_info), name, ether_setup);
>> + if (!dev)
>> + return ERR_PTR(-ENOMEM);
>

> Use alloc_etherdev() instead.

I could and it might be a little bit less code, but I would loose the
ability for users to request the name of their network device when
they allocate it.

Since I can't return any kind of a handle it seems very useful to
let the user pick the name a network device will initially be known
as.

>> +module_param_call(newif, etun_newif, etun_noget, NULL, S_IWUSR);
>> +module_param_call(delif, etun_delif, etun_noget, NULL, S_IWUSR);
>
>
> Doing create/delete via module parameter is wierd.
> Why not either use an ioctl, or sysfs.

I agree. However I could not find a create/delete idiom that was
at all common when I looked through the kernel virtual interfaces.
Which makes every way to create/delete an interface weird to
some degree.

In this case I am using sysfs.

The only sysfs attributes that were always available that I could find
were module parameters. A little odd because we can specify them on
the kernel command line, or when loading the module in addition to
being available at run time.

It gives me a general interface that is usable so long as the module
is loaded, and does not depend on the availability of any specific
network device. I will happily use any other interface that gives
me the same level of functionality for the roughly the same level
of effort.

>> +static int __init etun_init(void)
>> +{
>> + printk(KERN_INFO "etun: %s, %s\n", DRV_DESCRIPTION, DRV_VERSION);
>> + printk(KERN_INFO "etun: %s\n", DRV_COPYRIGHT);
>> +
>> + return 0;
>
> Why bother it is just advertising noise...

True, I really haven't given it a lot of though.

I doesn't really hurt to advertise and as well as social consequences
it technically allows detection of a kernel capability by reading boot

messages which can be very useful depending on the situation.

Eric

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

---

Subject: Re: [PATCH] net: Add etun driver
Posted by Johannes Berg on Mon, 09 Apr 2007 16:37:08 GMT
View Forum Message <> Reply to Message

On Fri, 2007-04-06 at 20:51 -0600, Eric W. Biederman wrote:

> The only sysfs attributes that were always available that I could find
> were module parameters. A little odd because we can specify them on
> the kernel command line, or when loading the module in addition to
> being available at run time.
>
> It gives me a general interface that is usable so long as the module
> is loaded, and does not depend on the availability of any specific
> network device. I will happily use any other interface that gives
> me the same level of functionality for the roughly the same level
> of effort.

Just for consideration, in wireless we currently create virtual network devices by having a "add_iface" and "remove_iface" sysfs files below the ieee80211 class.

johannes

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

---

Subject: Re: [PATCH] net: Add etun driver
Posted by Patrick McHardy on Mon, 09 Apr 2007 16:58:13 GMT
View Forum Message <> Reply to Message

Johannes Berg wrote:
> On Fri, 2007-04-06 at 20:51 -0600, Eric W. Biederman wrote:
>
>
>>The only sysfs attributes that were always available that I could find

>>were module parameters. A little odd because we can specify them on
>>the kernel command line, or when loading the module in addition to
>>being available at run time.
>>
>>It gives me a general interface that is usable so long as the module
>>is loaded, and does not depend on the availability of any specific
>>network device. I will happily use any other interface that gives
>>me the same level of functionality for the roughly the same level
>>of effort.
>
>
> Just for consideration, in wireless we currently create virtual network
> devices by having a "add_iface" and "remove_iface" sysfs files below the
> ieee80211 class.

It would be nice if someone would finally come up with a generic
interface based on netlink (RTM_NEWLINK) instead of adding yet
another couple of homegrown interfaces.

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