Subject: [PATCH] net: Add etun driver
Posted by ebiederm on Fri, 06 Apr 2007 20:43:15 GMT
View Forum Message <> Reply to Message

etun is a simple two headed tunnel driver that at the link layer looks
like ethernet. It's target audience is communicating between network
namespaces but it is general enough it has other valid uses as well.

Ben Greear implemented a similar device called redir-dev, for network
emulation.

OpenVZ has a similar device that goes by the name veth.

I didn't want to mess with ioctls or weird non-general network
interfaces for creating devices, so I used sysfs as my control
mechanism.

To create a pair of devices called veth0 and veth1:
  echo -n 'veth0,veth1' > /sys/module/etun/parameters/newif

To destroy a pair of devices:
  echo -n 'veth0' > /sys/module/etun/parameters/delif

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 drivers/net/Kconfig | 14 ++
 drivers/net/Makefile | 1 +
 drivers/net/etun.c | 486 ++
 3 files changed, 501 insertions(+), 0 deletions(-)
 create mode 100644 drivers/net/etun.c

diff --git a/drivers/net/Kconfig b/drivers/net/Kconfig
index c3f9f59..03cf13f 100644
--- a/drivers/net/Kconfig
+++ b/drivers/net/Kconfig
@@ -119,6 +119,20 @@ config TUN

    If you don't know what to use this for, you don't need it.

+config ETUN
+ tristate "Ethernet tunnel device driver support"
+ depends on SYSFS
+ ---help---
+   ETUN provices a pair of network devices that can be used for
+   configuring interesting topolgies. What one devices transmits
+   the other receives and vice versa. The link level framing
+   is ethernet for wide compatibility with network stacks.
+

```
+   To compile this driver as a module, choose M here: the module
+   will be called etun.
+
+   If you don't know what to use this for, you don't need it.
+
 config NET_SB1000
  tristate "General Instruments Surfboard 1000"
  depends on PNP
diff --git a/drivers/net/Makefile b/drivers/net/Makefile
index 33af833..549ff3f 100644
--- a/drivers/net/Makefile
+++ b/drivers/net/Makefile
@@ -185,6 +185,7 @@ obj-$(CONFIG_MACSONIC) += macsonic.o
 obj-$(CONFIG_MACMACE) += macmace.o
 obj-$(CONFIG_MAC89x0) += mac89x0.o
 obj-$(CONFIG_TUN) += tun.o
+obj-$(CONFIG_ETUN) += etun.o
 obj-$(CONFIG_NET_NETX) += netx-eth.o
 obj-$(CONFIG_DL2K) += dl2k.o
 obj-$(CONFIG_R8169) += r8169.o
diff --git a/drivers/net/etun.c b/drivers/net/etun.c
new file mode 100644
index 0000000..e15893a
--- /dev/null
+++ b/drivers/net/etun.c
@@ -0,0 +1,486 @@
+/*
+ *  ETUN - Universal ETUN device driver.
+ *  Copyright (C) 2006 Linux Networx
+ *
+ */
+
+#define DRV_NAME "etun"
+#define DRV_VERSION "1.0"
+#define DRV_DESCRIPTION "Ethernet pseudo tunnel device driver"
+#define DRV_COPYRIGHT "(C) 2007 Linux Networx"
+
+#include <linux/module.h>
+#include <linux/kernel.h>
+#include <linux/list.h>
+#include <linux/spinlock.h>
+#include <linux/skbuff.h>
+#include <linux/netdevice.h>
+#include <linux/etherdevice.h>
+#include <linux/ethtool.h>
+#include <linux/rtnetlink.h>
+#include <linux/if.h>
+#include <linux/if_ether.h>
```

```
+#include <linux/ctype.h>
+#include <net/dst.h>
+
+
+/* Device cheksum strategy.
+ *
+ * etun is designed to a be a pair of virutal devices
+ * connecting two network stack instances.
+ *
+ * Typically it will either be used with ethernet bridging or
+ * it will be used to route packets between the two stacks.
+ *
+ * The only checksum offloading I can do is to completely
+ * skip the checksumming step all together.
+ *
+ * When used for ethernet bridging I don't believe any
+ * checksum off loading is safe.
+ * - If my source is an external interface the checksum may be
+ *   invalid so I don't want to report I have already checked it.
+ * - If my destination is an external interface I don't want to put
+ *   a packet on the wire with someone computing the checksum.
+ *
+ * When used for routing between two stacks checksums should
+ * be as unnecessary as they are on the loopback device.
+ *
+ * So by default I am safe and disable checksumming and
+ * other advanced features like SG and TSO.
+ *
+ * However because I think these features could be useful
+ * I provide the ethtool functions to and enable/disable
+ * them at runtime.
+ *
+ * If you think you can correctly enable these go ahead.
+ * For checksums both the transmitter and the receiver must
+ * agree before the are actually disabled.
+ */
+
+#define ETUN_NUM_STATS 1
+static struct {
+ const char string[ETH_GSTRING_LEN];
+} ethtool_stats_keys[ETUN_NUM_STATS] = {
+ { "partner_ifindex" },
+};
+
+struct etun_info {
+ struct net_device *rx_dev;
+ unsigned  ip_summed;
+ struct net_device_stats stats;
```

```
+ struct list_head list;
+ struct net_device *dev;
+};
+
+/*
+ * I have to hold the rtnl_lock during device delete.
+ * So I use the rtnl_lock to protect my list manipulations
+ * as well.  Crude but simple.
+ */
+static LIST_HEAD(etun_list);
+
+/*
+ * The higher levels take care of making this non-reentrant (it's
+ * called with bh's disabled).
+ */
+static int etun_xmit(struct sk_buff *skb, struct net_device *tx_dev)
+{
+ struct etun_info *tx_info = tx_dev->priv;
+ struct net_device *rx_dev = tx_info->rx_dev;
+ struct etun_info *rx_info = rx_dev->priv;
+
+ tx_info->stats.tx_packets++;
+ tx_info->stats.tx_bytes += skb->len;
+
+ /* Drop the skb state that was needed to get here */
+ skb_orphan(skb);
+ if (skb->dst)
+  skb->dst = dst_pop(skb->dst); /* Allow for smart routing */
+
+ /* Switch to the receiving device */
+ skb->pkt_type = PACKET_HOST;
+ skb->protocol = eth_type_trans(skb, rx_dev);
+ skb->dev = rx_dev;
+ skb->ip_summed = CHECKSUM_NONE;
+
+ /* If both halves agree no checksum is needed */
+ if (tx_dev->features & NETIF_F_NO_CSUM)
+  skb->ip_summed = rx_info->ip_summed;
+
+ rx_dev->last_rx = jiffies;
+ rx_info->stats.rx_packets++;
+ rx_info->stats.rx_bytes += skb->len;
+ netif_rx(skb);
+
+ return 0;
+}
+
+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;
+}
+
+/* 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);
+
+ 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;
+}
+
+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;
+}
+
+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");
--
1.5.0.g53756
```

Subject: Re: [PATCH] net: Add etun driver
Posted by Ben Greear on Mon, 09 Apr 2007 18:14:52 GMT
View Forum Message <> Reply to Message

Patrick McHardy wrote:

> 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.

My preference is for ioctls, procfs, or similar that does not
require extra libraries. Ethtool is an ioctl based approach,
so that could potentially be used, though I'm not sure if
that's the right place to put it...

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 davem on Mon, 09 Apr 2007 18:37:30 GMT
View Forum Message <> Reply to Message

From: Ben Greear <greearb@candelatech.com>
Date: Mon, 09 Apr 2007 11:14:52 -0700

> Patrick McHardy wrote:
>
> > 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.
>
> My preference is for ioctls, procfs, or similar that does not
> require extra libraries. Ethtool is an ioctl based approach,
> so that could potentially be used, though I'm not sure if
> that's the right place to put it...

I totally disagree.

Netlink doesn't require libraries, the libraries just make using it
easier. It's a simple protocol sent over a socket, if you include
the right headers you can do it all yourself.

We shouldn't fail to use our proper core network configuration
interface just because it's inconvenient for you.

_____

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 davem on Mon, 09 Apr 2007 18:44:02 GMT
View Forum Message <> Reply to Message

From: Patrick McHardy <kaber@trash.net>
Date: Mon, 09 Apr 2007 18:58:13 +0200

> 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.

I absolutely agree, using these ioctls and sysfs/procfs crap
is totally insane given that we have a core mechanism for
network configuration.

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 18:45:51 GMT
View Forum Message <> Reply to Message

Ben Greear wrote:
> Patrick McHardy wrote:
>
>> 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.
>
>
> My preference is for ioctls, procfs, or similar that does not
> require extra libraries. Ethtool is an ioctl based approach,
> so that could potentially be used, though I'm not sure if
> that's the right place to put it...

Extra libraries is one of the least important points in my opinion,
I also guess pretty much anyone using a software device already has
iproute installed, which could easily support all of them.

The more important things to consider are in my opinion extendability
and atomicity of changes and dumps:

- ioctls: atomicity, not easily extendable
- sysfs: no atomicity, easily extendable
- procfs: if atomicity not easily extendable, but can of course also
  be used similar to sysfs

Only netlink offers both in an easy to use fashion and is already used
for the main parts of network configuration anyway.

Subject: Re: [PATCH] net: Add etun driver
Posted by Ben Greear on Mon, 09 Apr 2007 18:48:07 GMT
View Forum Message <> Reply to Message

David Miller wrote:
> From: Ben Greear <greearb@candelatech.com>
> Date: Mon, 09 Apr 2007 11:14:52 -0700
>
>> Patrick McHardy wrote:
>>
>>> 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.
>> My preference is for ioctls, procfs, or similar that does not
>> require extra libraries. Ethtool is an ioctl based approach,
>> so that could potentially be used, though I'm not sure if
>> that's the right place to put it...
>
> I totally disagree.
>
> Netlink doesn't require libraries, the libraries just make using it
> easier. It's a simple protocol sent over a socket, if you include
> the right headers you can do it all yourself.
>
> We shouldn't fail to use our proper core network configuration
> interface just because it's inconvenient for you.

Well, I find it inconvenient, so my preference is for other
methods...that shouldn't be too supprising. I'm a lazy programmer
after all! :)

If it does get added, and something like 'ip' gains ability to

use it, then I can just system("ip whatever"), so maybe that's the
out for lazy programmers...

Thanks,
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 Mon, 09 Apr 2007 19:35:11 GMT
View Forum Message <> Reply to Message

David Miller <davem@davemloft.net> writes:

> From: Patrick McHardy <kaber@trash.net>
> Date: Mon, 09 Apr 2007 18:58:13 +0200
>
>> 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.
>
> I absolutely agree, using these ioctls and sysfs/procfs crap
> is totally insane given that we have a core mechanism for
> network configuration.

The core mechanism for network configuration does not support creating
virtual devices in a extensible reusable way.

In particular the tunnel types supported by iproute2 are hard coded
into the user space tool and into the kernel interface. The interface
seems to be not the least bit extensible for creating new types of
non hardware backed network devices.

So I don't see a readily usable mechanism for network configuration in
netlink. The fact that netlink it uses unreliable packets and an
asynchronous interface just adds to the difficulty in making use of
it.

Nor have I seen a rigorous adherence to all new network configuration
using netlink. The wireless code doesn't even seem to really try.

So I don't believe anyone has found a good maintainable, unix
compatible configuration mechanism for maintaining anything yet.

Regardless I really don't care what the interface is as long as people
agree and I don't have to rewrite significant portions of user space
to allow people to use it.

Since whatever user space interface we pick we will be stuck with
we might as well review it as closely as we can and figure out what
we can live with.

Just to be contrary I'm tempted to add a sysctl interface. No one has
even mentioned class of user space interfaces yet.

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 Jeff Garzik on Mon, 09 Apr 2007 19:48:01 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> Nor have I seen a rigorous adherence to all new network configuration
> using netlink. The wireless code doesn't even seem to really try.

Not true:

> commit 711e2c33ac9221a419a9e28d05dd78a6a9c5fd4d
> Author: Jean Tourrilhes <jt@hpl.hp.com>
> Date: Wed Feb 22 15:10:56 2006 -0800
>
>   [PATCH] WE-20 for kernel 2.6.16
>
>   This is version 20 of the Wireless Extensions. This is the
>   completion of the RtNetlink work I started early 2004, it enables the
>   full Wireless Extension API over RtNetlink.

Though this is/was pulled, and the upcoming mac80211 configuration
interface should have netlink support.

This does not discount your other claims about netlink annoyances when

used for interface configuration (unreliable, async, etc.), which I tend
to agree with.

Jeff

_____
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 20:03:54 GMT
View Forum Message <> Reply to Message

On Mon, 2007-04-09 at 13:35 -0600, Eric W. Biederman wrote:
>
> Nor have I seen a rigorous adherence to all new network configuration
> using netlink. The wireless code doesn't even seem to really try.

You're talking about the wext stuff that was invented ten years ago.
cfg80211/nl80211 which I was talking about actually does use netlink
almost exclusively (and you can add virtual interfaces using netlink)
but we also provided some easy sysfs attributes.

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 20:11:29 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> The core mechanism for network configuration does not support creating
> virtual devices in a extensible reusable way.
>
> In particular the tunnel types supported by iproute2 are hard coded
> into the user space tool and into the kernel interface. The interface
> seems to be not the least bit extensible for creating new types of
> non hardware backed network devices.

Yes, it sucks.

> So I don't see a readily usable mechanism for network configuration in
> netlink.

Thats why I suggested that we should create one, ideally before adding more sysfs/proc/ioctl/... based interfaces, which we'll have a hard time getting rid of again.

I could take care of this if you don't mind waiting until 2.6.23.

> The fact that netlink it uses unreliable packets and an
> asynchronous interface just adds to the difficulty in making use of
> it.

Its reliable on the userspace->kernel path as long as you don't use MSG_DONTWAIT.

Subject: Re: [PATCH] net: Add etun driver
Posted by Johannes Berg on Mon, 09 Apr 2007 20:29:17 GMT
View Forum Message <> Reply to Message

On Mon, 2007-04-09 at 22:11 +0200, Patrick McHardy wrote:

> Thats why I suggested that we should create one, ideally before adding
> more sysfs/proc/ioctl/... based interfaces, which we'll have a hard time
> getting rid of again.

But then how would we configure initial parameters along with creating the interface? That's something that's good to have, and I don't see how you can allow it generically.

Then again, you can of course always add an interface and then change its operating parameters etc. but that isn't atomic any more.

johannes

Subject: Re: [PATCH] net: Add etun driver
Posted by Patrick McHardy on Tue, 10 Apr 2007 00:06:47 GMT
View Forum Message <> Reply to Message

Johannes Berg wrote:
> On Mon, 2007-04-09 at 22:11 +0200, Patrick McHardy wrote:
>
>
>>Thats why I suggested that we should create one, ideally before adding
>>more sysfs/proc/ioctl/... based interfaces, which we'll have a hard time
>>getting rid of again.
>
>
> But then how would we configure initial parameters along with creating
> the interface? That's something that's good to have, and I don't see how
> you can allow it generically.
>
> Then again, you can of course always add an interface and then change
> its operating parameters etc. but that isn't atomic any more.

Same way as the current RTM_SETLINK message works, but with creating
a new link in advance. It works fine in other subsystems, so I don't
see why it would in this case as well. Some subsystems do it in an
atomic fashion (network schedulers for example), some first create
the object, then configure it (network classifiers in the non-compat
cases). In the network device case I suppose the later should work
fine since a device needs to be set UP in a second action before
it really does anything.

Subject: Re: [PATCH] net: Add etun driver
Posted by Johannes Berg on Tue, 10 Apr 2007 05:47:08 GMT
View Forum Message <> Reply to Message

On Tue, 2007-04-10 at 02:06 +0200, Patrick McHardy wrote:

> Same way as the current RTM_SETLINK message works, but with creating
> a new link in advance. It works fine in other subsystems, so I don't

> see why it would in this case as well. Some subsystems do it in an
> atomic fashion (network schedulers for example), some first create
> the object, then configure it (network classifiers in the non-compat
> cases). In the network device case I suppose the later should work
> fine since a device needs to be set UP in a second action before
> it really does anything.

Our virtual devices are always associated with a piece of hardware, and
we really want them to be associated with that at all times, even when
not UP. Everything else seems like a huge complication if only because
then we can't have whoever will be responsible for the device allocate
it's private space area.

johannes

Subject: Re: [PATCH] net: Add etun driver
Posted by ebiederm on Tue, 10 Apr 2007 06:08:48 GMT
View Forum Message <> Reply to Message

Johannes Berg <johannes@sipsolutions.net> writes:

> Our virtual devices are always associated with a piece of hardware, and
> we really want them to be associated with that at all times, even when
> not UP. Everything else seems like a huge complication if only because
> then we can't have whoever will be responsible for the device allocate
> it's private space area.

How are you specifying which piece of hardware today? By interface name?

What function do your virtual devices serve?

I'm trying to wrap my head around the model.

Eric

Subject: Re: [PATCH] net: Add etun driver

Posted by Johannes Berg on Tue, 10 Apr 2007 06:18:56 GMT
View Forum Message <> Reply to Message

On Tue, 2007-04-10 at 00:08 -0600, Eric W. Biederman wrote:

> How are you specifying which piece of hardware today? By interface name?

By wiphy index, which is just the hardware identifier we give each
802.11 device present on the system (if it uses cfg80211)

> What function do your virtual devices serve?

Basically you can have a monitor, access point and WDS link running on
the same 802.11 MAC at the same time and that's what we need to present,
each one as a netdev.

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 Tue, 10 Apr 2007 07:52:34 GMT
View Forum Message <> Reply to Message

Johannes Berg wrote:
> Our virtual devices are always associated with a piece of hardware, and
> we really want them to be associated with that at all times, even when
> not UP. Everything else seems like a huge complication if only because
> then we can't have whoever will be responsible for the device allocate
> it's private space area.

Shouldn't be a problem either. Creating the device atomically also
prevents that anything else is setting them UP before they're fully
configured.

_____
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 Tue, 10 Apr 2007 09:18:54 GMT

View Forum Message <> Reply to Message

On Tue, 2007-04-10 at 09:52 +0200, Patrick McHardy wrote:

> Shouldn't be a problem either. Creating the device atomically also
> prevents that anything else is setting them UP before they're fully
> configured.

How would you do it generically then? I'm absolutely not opposed to the
idea but for now haven't seen how to do it.

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 Tue, 10 Apr 2007 09:52:02 GMT
View Forum Message <> Reply to Message

Johannes Berg wrote:
> On Tue, 2007-04-10 at 09:52 +0200, Patrick McHardy wrote:
>
>
>>Shouldn't be a problem either. Creating the device atomically also
>>prevents that anything else is setting them UP before they're fully
>>configured.
>
>
> How would you do it generically then? I'm absolutely not opposed to the
> idea but for now haven't seen how to do it.

Without having thought much about it yet, roughly like this:

- driver receives RTM_NEWLINK message (under rtnl)
- driver allocates new device
- driver initializes device based on content of RTM_NEWLINK message
- driver returns

Device creation won't be generic of course since only the driver knows
how to do that.

_____
Containers mailing list

Subject: Re: [PATCH] net: Add etun driver
Posted by Johannes Berg on Tue, 10 Apr 2007 10:27:59 GMT
View Forum Message <> Reply to Message

On Tue, 2007-04-10 at 11:52 +0200, Patrick McHardy wrote:

> Without having thought much about it yet, roughly like this:
>
> - driver receives RTM_NEWLINK message (under rtnl)
> - driver allocates new device
> - driver initializes device based on content of RTM_NEWLINK message
> - driver returns

Sounds good to me, but where's the advantage over something that isn't generic if RTM_NEWLINK contains totally different things depending on the subsystem like wireless where it'd have to contain the hardware identifier?

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 Tue, 10 Apr 2007 10:46:58 GMT
View Forum Message <> Reply to Message

Johannes Berg wrote:
> On Tue, 2007-04-10 at 11:52 +0200, Patrick McHardy wrote:
>
>
>>Without having thought much about it yet, roughly like this:
>>
>>- driver receives RTM_NEWLINK message (under rtnl)
>>- driver allocates new device
>>- driver initializes device based on content of RTM_NEWLINK message
>>- driver returns
>
>
> Sounds good to me, but where's the advantage over something that isn't

> generic if RTM_NEWLINK contains totally different things depending on
> the subsystem like wireless where it'd have to contain the hardware
> identifier?

Not totally different, so far I think we should use the same attributes
as for RTM_SETLINK messages and include the device-specific stuff in
IFLA_PROTINFO, which is symetric to what the kernel sends in RTM_NETLINK
messages (see br_netlink.c for an example). The easiest case would be an
empty IFLA_PROTINFO attribute, which would simply create a device
without any configuration.

The main advantage that we don't get more weird sysfs/proc/ioctl based
interfaces and use the same interface that is used for all other network
configuration, which f.e. will allow to add support for all software
devices to iproute without much effort, so you don't need 30 different
tools for configuring the different software device types anymore.
Additionally we get atomic setup/dumps and extensibility.

Subject: Re: [PATCH] net: Add etun driver
Posted by Johannes Berg on Tue, 10 Apr 2007 11:02:10 GMT
View Forum Message <> Reply to Message

On Tue, 2007-04-10 at 12:46 +0200, Patrick McHardy wrote:

> Not totally different, so far I think we should use the same attributes
> as for RTM_SETLINK messages and include the device-specific stuff in
> IFLA_PROTINFO, which is symetric to what the kernel sends in RTM_NETLINK
> messages (see br_netlink.c for an example). The easiest case would be an
> empty IFLA_PROTINFO attribute, which would simply create a device
> without any configuration.

I'll have to look up these things.

> The main advantage that we don't get more weird sysfs/proc/ioctl based
> interfaces

Please don't put me into a corner I don't want to be in ;) The new
wireless stuff was completely designed using netlink. The sysfs
interface to these two specific things was a concession since it used to
exist before and we don't really have a fully functional userspace tool
yet.

> and use the same interface that is used for all other network
> configuration, which f.e. will allow to add support for all software
> devices to iproute without much effort, so you don't need 30 different
> tools for configuring the different software device types anymore.
> Additionally we get atomic setup/dumps and extensibility.

I don't think wireless can get away without a new tool. So much stuff
there. Look at
http://git.kernel.org/?p=linux/kernel/git/linville/wireless-dev.git;a=blob;f=include/linux/nl80211.h;hb=HEAD

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 Tue, 10 Apr 2007 11:09:20 GMT
View Forum Message <> Reply to Message

Johannes Berg wrote:
> On Tue, 2007-04-10 at 12:46 +0200, Patrick McHardy wrote:
>
>
>>The main advantage that we don't get more weird sysfs/proc/ioctl based
>>interfaces
>
>
> Please don't put me into a corner I don't want to be in ;) The new
> wireless stuff was completely designed using netlink. The sysfs
> interface to these two specific things was a concession since it used to
> exist before and we don't really have a fully functional userspace tool
> yet.

I know :) It was a few month ago when I noticed the new bonding
sysfs interface when I first thought that we really need this.

>> and use the same interface that is used for all other network
>>configuration, which f.e. will allow to add support for all software
>>devices to iproute without much effort, so you don't need 30 different
>>tools for configuring the different software device types anymore.
>>Additionally we get atomic setup/dumps and extensibility.
>

>
> I don't think wireless can get away without a new tool. So much stuff
> there. Look at
>
http://git.kernel.org/?p=linux/kernel/git/linville/wireless-dev.git;a=blob;f=include/linux/nl80211.h;hb=HEAD

Maybe not wireless, but bonding, briding, vlan, etun, possibly more.

_____
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 Jeff Garzik on Tue, 10 Apr 2007 11:16:24 GMT
View Forum Message <> Reply to Message

Patrick McHardy wrote:
> Maybe not wireless, but bonding, briding, vlan, etun, possibly more.

[if I understand you correctly] I agree. With ethtool, the idea is to have a single tool that supports multiple hardware platforms -- even to the point of introducing hardware-specific code into ethtool.

While I agree with wireless-dev that wireless /needs/ a separate tool, for things like bridging and vlan it would certainly be nice to mitigate the tool count explosion.

Jeff

_____
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 Tue, 10 Apr 2007 11:24:57 GMT
View Forum Message <> Reply to Message

On Tue, 2007-04-10 at 13:09 +0200, Patrick McHardy wrote:

> I know :) It was a few month ago when I noticed the new bonding
> sysfs interface when I first thought that we really need this.

:)

> > I don't think wireless can get away without a new tool. So much stuff
> > there. Look at
> >
http://git.kernel.org/?p=linux/kernel/git/linville/wireless-dev.git;a=blob;f=include/linux/nl80211.h;hb=HEAD
>
>
> Maybe not wireless, but bonding, briding, vlan, etun, possibly more.

Fair enough. Then the question however remains whether wireless should
try to do all things it needs in one or try leveraging multiple things
from other places. Thoughts?

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 Tue, 10 Apr 2007 12:05:46 GMT
View Forum Message <> Reply to Message

Johannes Berg wrote:
> Fair enough. Then the question however remains whether wireless should
> try to do all things it needs in one or try leveraging multiple things
> from other places. Thoughts?

I know too little about wireless to judge really this. My opinion is
that if it is possible to add and configure an interface (even if
only for simple cases) without knowledge about driver internals by
setting a few parameters, it would probably make sense to use
RTM_NEWLINK as well. If a userspace daemon or complex knowledge of
driver internals is needed, it probably should stay seperated.

So simply put: if I can implement support for
"ip wireless add dev wlan0 mode managed essid ... key ..."
in less than 100 lines and get a working connection afterwards, it
seems worth it.

_____
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 linville on Tue, 10 Apr 2007 13:44:11 GMT
View Forum Message <> Reply to Message

On Tue, Apr 10, 2007 at 02:05:46PM +0200, Patrick McHardy wrote:

> So simply put: if I can implement support for
> "ip wireless add dev wlan0 mode managed essid ... key ..."
> in less than 100 lines and get a working connection afterwards, it
> seems worth it.

ACK

--
John W. Linville
linville@tuxdriver.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 Johannes Berg on Tue, 10 Apr 2007 13:48:08 GMT
View Forum Message <> Reply to Message

On Tue, 2007-04-10 at 14:05 +0200, Patrick McHardy wrote:
>
> I know too little about wireless to judge really this. My opinion is
> that if it is possible to add and configure an interface (even if
> only for simple cases) without knowledge about driver internals by
> setting a few parameters, it would probably make sense to use
> RTM_NEWLINK as well. If a userspace daemon or complex knowledge of
> driver internals is needed, it probably should stay seperated.

It still is though we're moving towards the userspace daemon thing.

> So simply put: if I can implement support for
> "ip wireless add dev wlan0 mode managed essid ... key ..."
> in less than 100 lines and get a working connection afterwards, it
> seems worth it.

For completely open networks (no "key ...") or just WEP (static key) it

could probably be done, but I don't see how right now.

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 Johannes Berg on Tue, 10 Apr 2007 21:16:59 GMT
View Forum Message <> Reply to Message

On Tue, 2007-04-10 at 02:06 +0200, Patrick McHardy wrote:

> Same way as the current RTM_SETLINK message works, but with creating
> a new link in advance. It works fine in other subsystems, so I don't
> see why it would in this case as well. Some subsystems do it in an
> atomic fashion (network schedulers for example), some first create
> the object, then configure it (network classifiers in the non-compat
> cases). In the network device case I suppose the later should work
> fine since a device needs to be set UP in a second action before
> it really does anything.

Looking at br_netlink.c it seems that this sort of contradicts why
generic netlink was done, now all the sudden everything that wants to
create new links need its own netlink protocol number, no?

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 Wed, 11 Apr 2007 16:15:41 GMT
View Forum Message <> Reply to Message

Johannes Berg wrote:
> On Tue, 2007-04-10 at 02:06 +0200, Patrick McHardy wrote:
>
>
>>Same way as the current RTM_SETLINK message works, but with creating
>>a new link in advance. It works fine in other subsystems, so I don't

>>see why it would in this case as well. Some subsystems do it in an
>>atomic fashion (network schedulers for example), some first create
>>the object, then configure it (network classifiers in the non-compat
>>cases). In the network device case I suppose the later should work
>>fine since a device needs to be set UP in a second action before
>>it really does anything.
>
>
> Looking at br_netlink.c it seems that this sort of contradicts why
> generic netlink was done, now all the sudden everything that wants to
> create new links need its own netlink protocol number, no?

No, generic netlink avoids allocating netlink families. br_netlink
uses the same netlink family as the other network configuration stuff
(NETLINK_ROUTE), but a different rtgen_family (which matches the
address families). But you have a valid point, if we want to use
this for things like bonding or VLAN that aren't actually address
families, we should consider introducing "rtnetlink families" to
avoid adding AF_BONDING, AF_8021Q etc.

-
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

---

Subject: Re: [PATCH] net: Add etun driver
Posted by Stephen Hemminger on Wed, 11 Apr 2007 16:16:06 GMT
View Forum Message <> Reply to Message

On Tue, 10 Apr 2007 23:16:59 +0200
Johannes Berg <johannes@sipsolutions.net> wrote:

> On Tue, 2007-04-10 at 02:06 +0200, Patrick McHardy wrote:
>
> > Same way as the current RTM_SETLINK message works, but with creating
> > a new link in advance. It works fine in other subsystems, so I don't
> > see why it would in this case as well. Some subsystems do it in an
> > atomic fashion (network schedulers for example), some first create
> > the object, then configure it (network classifiers in the non-compat
> > cases). In the network device case I suppose the later should work

> > fine since a device needs to be set UP in a second action before
> > it really does anything.
>
> Looking at br_netlink.c it seems that this sort of contradicts why
> generic netlink was done, now all the sudden everything that wants to
> create new links need its own netlink protocol number, no?
>
> johannes

Bridging is different since there was already a bridge protocol number assigned,
there was no point in doing generic netlink.

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

---

Subject: Re: [PATCH] net: Add etun driver
Posted by Johannes Berg on Wed, 11 Apr 2007 16:43:13 GMT
View Forum Message <> Reply to Message

On Wed, 2007-04-11 at 18:15 +0200, Patrick McHardy wrote:

> No, generic netlink avoids allocating netlink families.

Well, yes, I thought that was pretty much the point. :)

> br_netlink
> uses the same netlink family as the other network configuration stuff
> (NETLINK_ROUTE), but a different rtgen_family (which matches the
> address families).

Ah ok. I got all the family types confused then.

> But you have a valid point, if we want to use
> this for things like bonding or VLAN that aren't actually address
> families, we should consider introducing "rtnetlink families" to
> avoid adding AF_BONDING, AF_8021Q etc.

True.

But this still doesn't help wireless which doesn't have either an
rtnetlink family nor an address family since it uses generic netlink
exclusively.

johannes

Subject: Re: [PATCH] net: Add etun driver
Posted by Patrick McHardy on Wed, 11 Apr 2007 16:52:33 GMT
View Forum Message <> Reply to Message

Johannes Berg wrote:
> On Wed, 2007-04-11 at 18:15 +0200, Patrick McHardy wrote:
>
>> But you have a valid point, if we want to use
>>this for things like bonding or VLAN that aren't actually address
>>families, we should consider introducing "rtnetlink families" to
>>avoid adding AF_BONDING, AF_8021Q etc.
>
>
> True.
>
> But this still doesn't help wireless which doesn't have either an
> rtnetlink family nor an address family since it uses generic netlink
> exclusively.

I'm not sure I'm following. I was under the impression that the
conclusion of yesterdays discussion was that its probably not
worth using rtnetlink for wireless so it will continue to use
generic netlink exclusively, but even if that is wrong, nothing
would prevent adding a "rtnetlink family" for wireless as well.
The idea of introducing "rtnetlink families" is exactly to avoid
adding real address families for things that don't have one to
avoid possible conflicts with IANA allocated numbers.

Subject: Re: [PATCH] net: Add etun driver
Posted by Johannes Berg on Wed, 11 Apr 2007 16:59:12 GMT
View Forum Message <> Reply to Message

On Wed, 2007-04-11 at 18:52 +0200, Patrick McHardy wrote:

> I'm not sure I'm following. I was under the impression that the
> conclusion of yesterdays discussion was that its probably not
> worth using rtnetlink for wireless so it will continue to use
> generic netlink exclusively, but even if that is wrong, nothing
> would prevent adding a "rtnetlink family" for wireless as well.

Oh, ok. Yeah nothing stops us from doing that, of course.

johannes