Subject: Re: [RFC][patch 1/4] Network namespaces: cleanup of dev_base list use
Posted by ebiederm on Mon, 26 Jun 2006 15:13:52 GMT
View Forum Message <> Reply to Message

Andrey Savochkin <saw@swsoft.com> writes:

> Cleanup of dev_base list use, with the aim to make device list per-namespace.
> In almost every occasion, use of dev_base variable and dev->next pointer
> could be easily replaced by for_each_netdev loop.
> A few most complicated places were converted to using
> first_netdev()/next_netdev().

As a proof of concept patch this is ok.

As a real world patch this is much too big, which prevents review.
Plus it takes a few actions that are more than replace just
iterators through the device list.

In addition I suspect several if not all of these iterators
can be replaced with the an appropriate helper function.

The normal structure for a patch like this would be to
introduce the new helper function. for_each_netdev.
And then to replace all of the users while cc'ing the
maintainers of those drivers. With each different
driver being a different patch.

There is another topic for discussion in this patch as well.
How much of the context should be implicit and how much
should be explicit.

If the changes from netchannels had already been implemented, and all of
the network processing was happening in a process context then I would
trivially agree that implicit would be the way to go.

However short of always having code always execute in the proper
context I'm not comfortable with implicit parameters to functions.
Not that this the contents of this patch should address this but the
later patches should.

When I went through this, my patchset just added an explicit
continue if the devices was not in the appropriate namespace.
I actually prefer the multiple list implementation but at
the same time I think it is harder to get a clean implementation
out of it.

Eric

```
> --- ./drivers/block/aoe/aoecmd.c.vedevbase Wed Jun 21 18:50:28 2006
> +++ ./drivers/block/aoe/aoecmd.c Thu Jun 22 12:03:07 2006
> @@ -204,14 +204,17 @@ aoecmd_cfg_pkts(ushort aoemajor, unsigne
>   sl = sl_tail = NULL;
>
>   read_lock(&dev_base_lock);
> - for (ifp = dev_base; ifp; dev_put(ifp), ifp = ifp->next) {
> + for_each_netdev(dev) {
>    dev_hold(ifp);
> -  if (!is_aoe_netif(ifp))
> +  if (!is_aoe_netif(ifp)) {
> +   dev_put(ifp);
>     continue;
> +  }
>
>    skb = new_skb(ifp, sizeof *h + sizeof *ch);
>    if (skb == NULL) {
>     printk(KERN_INFO "aoe: aoecmd_cfg: skb alloc
> failure\n");
> +   dev_put(ifp);
>     continue;
>    }
>    if (sl_tail == NULL)
> @@ -229,6 +232,7 @@ aoecmd_cfg_pkts(ushort aoemajor, unsigne
>
>    skb->next = sl;
>    sl = skb;
> +  dev_put(ifp);
>   }
>   read_unlock(&dev_base_lock);
```

These hunks should use for_each_netdev(ifp);

```
> --- ./include/linux/netdevice.h.vedevbase Wed Jun 21 18:53:17 2006
> +++ ./include/linux/netdevice.h Thu Jun 22 18:57:50 2006
> @@ -289,8 +289,8 @@ struct net_device
>
>   unsigned long  state;
>
> - struct net_device *next;
> -
> + struct list_head dev_list;
> +
>   /* The device initialization function. Called only once. */
>   int   (*init)(struct net_device *dev);
>
> @@ -543,9 +543,27 @@ struct packet_type {
```

> #include <linux/interrupt.h>
> #include <linux/notifier.h>
>
> -extern struct net_device loopback_dev; /* The loopback */
> -extern struct net_device *dev_base; /* All devices */
> -extern rwlock_t dev_base_lock; /* Device list lock */
> +extern struct net_device loopback_dev; /* The loopback */
> +extern struct list_head dev_base_head; /* All devices */
> +extern rwlock_t dev_base_lock; /* Device list lock */
> +

No need to change the loopback_dev and dev_base_lock here.

What is the advantage of changing the type of dev_base?
I can guess but there should be an explanation of it.

> +#define for_each_netdev(p) list_for_each_entry(p, &dev_base_head, dev_list)
> +
> +/* DO NOT USE first_netdev/next_netdev, use loop defined above */
> +#define first_netdev() ({ \
> +   list_empty(&dev_base_head) ? NULL : \
> + list_entry(dev_base_head.next, \
> +   struct net_device, \
> +   dev_list); \
> +   })
> +#define next_netdev(dev) ({ \
> +   struct list_head *__next; \
> +   __next = (dev)->dev_list.next; \
> +   __next == &dev_base_head ? NULL : \
> +   list_entry(__next, \
> +   struct net_device, \
> +   dev_list); \
> +   })
>
> extern int netdev_boot_setup_check(struct net_device *dev);
> extern unsigned long netdev_boot_base(const char *prefix, int unit);

> @@ -1903,7 +1902,7 @@ static int dev_ifconf(char __user *arg)
>   */
>
>   total = 0;
> - for (dev = dev_base; dev; dev = dev->next) {
> + for_each_netdev(dev) {
>   for (i = 0; i < NPROTO; i++) {
>   if (gifconf_list[i]) {
>   int done;
> @@ -1935,26 +1934,25 @@ static int dev_ifconf(char __user *arg)

Hmm. The proc code here appears to be more than non-trivial
restructuring. I'm not certain it is but it looks that way
which make review harder.

```
>   * This is invoked by the /proc filesystem handler to display a device
>   * in detail.
>   */
> -static __inline__ struct net_device *dev_get_idx(loff_t pos)
> -{
> - struct net_device *dev;
> - loff_t i;
> -
> - for (i = 0, dev = dev_base; dev && i < pos; ++i, dev = dev->next);
> -
> - return i == pos ? dev : NULL;
> -}
> -
>  void *dev_seq_start(struct seq_file *seq, loff_t *pos)
>  {
> + struct net_device *dev;
> + loff_t off = 1;
>   read_lock(&dev_base_lock);
> - return *pos ? dev_get_idx(*pos - 1) : SEQ_START_TOKEN;
> + if (!*pos)
> +  return SEQ_START_TOKEN;
> + for_each_netdev(dev) {
> +  if (off++ == *pos)
> +   return dev;
> + }
> + return NULL;
>  }
>
>  void *dev_seq_next(struct seq_file *seq, void *v, loff_t *pos)
>  {
> + struct net_device *dev = v;
>   ++*pos;
> - return v == SEQ_START_TOKEN ? dev_base : ((struct net_device *)v)->next;
> + return v == SEQ_START_TOKEN ? first_netdev() : next_netdev(dev);
>  }
>
>  void dev_seq_stop(struct seq_file *seq, void *v)
```