
Subject: [PATCH 2/12] L2 network namespace: network devices virtualization
Posted by [Mishin Dmitry](#) on Wed, 06 Dec 2006 22:25:17 GMT
[View Forum Message](#) <> [Reply to Message](#)

Added ability to have per-namespace network devices.

Signed-off-by: Dmitry Mishin <dim@openvz.org>

```
include/linux/net_namespace.h | 6 ++-
include/linux/netdevice.h    | 10 +++++
net/core/dev.c               | 78 ++++++-----
net/core/net-sysfs.c         | 23 ++++++++
net/core/net_namespace.c     | 11 +++++
5 files changed, 114 insertions(+), 14 deletions(-)
```

--- linux-2.6.19-rc6-mm2.orig/include/linux/net_namespace.h

+++ linux-2.6.19-rc6-mm2/include/linux/net_namespace.h

@ @ -6,8 +6,10 @ @

```
#include <linux/errno.h>
```

```
struct net_namespace {
- struct kref kref;
- struct nsproxy *ns;
+ struct kref kref;
+ struct nsproxy *ns;
+ struct net_device *dev_base_p, **dev_tail_p;
+ unsigned int hash;
};
```

```
extern struct net_namespace init_net_ns;
```

--- linux-2.6.19-rc6-mm2.orig/include/linux/netdevice.h

+++ linux-2.6.19-rc6-mm2/include/linux/netdevice.h

```
@ @ -379,6 +379,9 @ @ struct net_device
int promiscuity;
int allmulti;
```

```
+#ifdef CONFIG_NET_NS
```

```
+ struct net_namespace *net_ns;
```

```
+#endif
```

```
/* Protocol specific pointers */
```

```
@ @ -557,9 +560,16 @ @ struct packet_type {
```

```
#include <linux/interrupt.h>
```

```
#include <linux/notifier.h>
```

```
+#include <linux/net_namespace.h>
```

```

extern struct net_device loopback_dev; /* The loopback */
#ifdef CONFIG_NET_NS
extern struct net_device *dev_base; /* All devices */
#define dev_base_ns(dev) dev_base
#else
#define dev_base (current_net_ns->dev_base_p)
#define dev_base_ns(dev) (dev->net_ns->dev_base_p)
#endif
extern rwlock_t dev_base_lock; /* Device list lock */

extern int netdev_boot_setup_check(struct net_device *dev);
--- linux-2.6.19-rc6-mm2.orig/net/core/dev.c
+++ linux-2.6.19-rc6-mm2/net/core/dev.c
@@ -90,6 +90,7 @@
#include <linux/if_ether.h>
#include <linux/netdevice.h>
#include <linux/etherdevice.h>
#include <linux/net_namespace.h>
#include <linux/notifier.h>
#include <linux/skbuff.h>
#include <net/sock.h>
@@ -174,20 +175,28 @@ static spinlock_t net_dma_event_lock;
 * unregister_netdevice(), which must be called with the rtnl
 * semaphore held.
 */
#ifdef CONFIG_NET_NS
struct net_device *dev_base;
static struct net_device **dev_tail = &dev_base;
-DEFINE_RWLOCK(dev_base_lock);
-
#define dev_tail_ns(dev) dev_tail
EXPORT_SYMBOL(dev_base);
#else
#define dev_tail (current_net_ns->dev_tail_p)
#define dev_tail_ns(dev) (dev->net_ns->dev_tail_p)
#endif
+
+DEFINE_RWLOCK(dev_base_lock);
EXPORT_SYMBOL(dev_base_lock);

#define NETDEV_HASHBITS 8
static struct hlist_head dev_name_head[1<<NETDEV_HASHBITS];
static struct hlist_head dev_index_head[1<<NETDEV_HASHBITS];

-static inline struct hlist_head *dev_name_hash(const char *name)
+static inline struct hlist_head *dev_name_hash(const char *name,
+ struct net_namespace *ns)

```

```

{
    unsigned hash = full_name_hash(name, strlen(name, IFNAMSIZ));
+ hash ^= net_ns_hash(ns);
    return &dev_name_head[hash & ((1<<NETDEV_HASHBITS)-1)];
}

@@ -212,10 +221,12 @@ DEFINE_PER_CPU(struct softnet_data, soft
extern int netdev_sysfs_init(void);
extern int netdev_register_sysfs(struct net_device *);
extern void netdev_unregister_sysfs(struct net_device *);
+extern int netdev_rename_sysfs(struct net_device *);
#else
#define netdev_sysfs_init() (0)
#define netdev_register_sysfs(dev) (0)
#define netdev_unregister_sysfs(dev) do { } while(0)
+#define netdev_rename_sysfs(dev) (0)
#endif

@@ -475,10 +486,13 @@ __setup("netdev=", netdev_boot_setup);
struct net_device *__dev_get_by_name(const char *name)
{
    struct hlist_node *p;
+ struct net_namespace *ns = current_net_ns;

- hlist_for_each(p, dev_name_hash(name)) {
+ hlist_for_each(p, dev_name_hash(name, ns)) {
    struct net_device *dev
        = hlist_entry(p, struct net_device, name_hlist);
+ if (!net_ns_match(dev->net_ns, ns))
+     continue;
    if (!strncmp(dev->name, name, IFNAMSIZ))
        return dev;
}

@@ -751,10 +765,11 @@ int dev_change_name(struct net_device *d
else
    strcpy(dev->name, newname, IFNAMSIZ);

- err = device_rename(&dev->dev, dev->name);
+ err = netdev_rename_sysfs(dev);
    if (!err) {
        hlist_del(&dev->name_hlist);
- hlist_add_head(&dev->name_hlist, dev_name_hash(dev->name));
+ hlist_add_head(&dev->name_hlist, dev_name_hash(dev->name,
+     current_net_ns));
        raw_notifier_call_chain(&netdev_chain,
            NETDEV_CHANGENAME, dev);
    }

```

```

@@ -1481,8 +1496,11 @@ gso:
    spin_lock(&dev->queue_lock);
    q = dev->qdisc;
    if (q->enqueue) {
+   struct net_namespace *orig_net_ns;
+   orig_net_ns = push_net_ns(dev->net_ns);
    rc = q->enqueue(skb, q);
    qdisc_run(dev);
+   pop_net_ns(orig_net_ns);
    spin_unlock(&dev->queue_lock);

    rc = rc == NET_XMIT_BYPASS ? NET_XMIT_SUCCESS : rc;
@@ -1678,7 +1696,10 @@ static void net_tx_action(struct softirq
    clear_bit(__LINK_STATE_SCHED, &dev->state);

    if (spin_trylock(&dev->queue_lock)) {
+   struct net_namespace *orig_net_ns;
+   orig_net_ns = push_net_ns(dev->net_ns);
    qdisc_run(dev);
+   pop_net_ns(orig_net_ns);
    spin_unlock(&dev->queue_lock);
    } else {
        netif_schedule(dev);
@@ -1765,6 +1786,7 @@ int netif_receive_skb(struct sk_buff *sk
{
    struct packet_type *ptype, *pt_prev;
    struct net_device *orig_dev;
+   struct net_namespace *orig_net_ns;
    int ret = NET_RX_DROP;
    __be16 type;

@@ -1783,6 +1805,8 @@ int netif_receive_skb(struct sk_buff *sk
    if (!orig_dev)
        return NET_RX_DROP;

+   orig_net_ns = push_net_ns(skb->dev->net_ns);
+
    __get_cpu_var(netdev_rx_stat).total++;

    skb->h.raw = skb->nh.raw = skb->data;
@@ -1851,6 +1875,7 @@ ncls:

out:
    rcu_read_unlock();
+   pop_net_ns(orig_net_ns);
    return ret;
}

```

```

@@ -2878,6 +2903,7 @@ int register_netdevice(struct net_device
{
    struct hlist_head *head;
    struct hlist_node *p;
+ struct net_namespace *ns;
    int ret;

    BUG_ON(dev_boot_phase);
@@ -2895,6 +2921,12 @@ int register_netdevice(struct net_device
    spin_lock_init(&dev->ingress_lock);
#endif

+ ns = NULL;
+ #ifdef CONFIG_NET_NS
+ BUG_ON(!dev->net_ns);
+ ns = dev->net_ns;
+ #endif
+
    dev->iflink = -1;

    /* Init, if this function is available */
@@ -2917,10 +2949,12 @@ int register_netdevice(struct net_device
    dev->iflink = dev->ifindex;

    /* Check for existence of name */
- head = dev_name_hash(dev->name);
+ head = dev_name_hash(dev->name, ns);
    hlist_for_each(p, head) {
        struct net_device *d
            = hlist_entry(p, struct net_device, name_hlist);
+ if (!net_ns_match(d->net_ns, ns))
+ continue;
        if (!strcmp(d->name, dev->name, IFNAMSIZ)) {
            ret = -EEXIST;
            goto out;
@@ -2980,8 +3014,8 @@ int register_netdevice(struct net_device
    dev->next = NULL;
    dev_init_scheduler(dev);
    write_lock_bh(&dev_base_lock);
- *dev_tail = dev;
- dev_tail = &dev->next;
+ *dev_tail_ns(dev) = dev;
+ dev_tail_ns(dev) = &dev->next;
    hlist_add_head(&dev->name_hlist, head);
    hlist_add_head(&dev->index_hlist, dev_index_hash(dev->ifindex));
    dev_hold(dev);
@@ -3195,6 +3229,10 @@ struct net_device *alloc_netdev(int size
    if (sizeof_priv)

```

```

dev->priv = netdev_priv(dev);

#ifdef CONFIG_NET_NS
+ get_net_ns(current_net_ns);
+ dev->net_ns = current_net_ns;
#endif
setup(dev);
strcpy(dev->name, name);
return dev;
@@ -3211,6 +3249,15 @@ EXPORT_SYMBOL(alloc_netdev);
*/
void free_netdev(struct net_device *dev)
{
#ifdef CONFIG_NET_NS
+ struct net_namespace *ns;
+
+ ns = dev->net_ns;
+ if (ns != NULL) {
+ put_net_ns(ns);
+ dev->net_ns = NULL;
+ }
#endif
#ifdef CONFIG_SYSFS
/* Compatibility with error handling in drivers */
if (dev->reg_state == NETREG_UNINITIALIZED) {
@@ -3221,6 +3268,13 @@ void free_netdev(struct net_device *dev)
BUG_ON(dev->reg_state != NETREG_UNREGISTERED);
dev->reg_state = NETREG_RELEASED;

#ifdef CONFIG_NET_NS
+ if (ns != NULL && ns != &init_net_ns) {
+ kfree((char *)dev - dev->padded);
+ return;
+ }
#endif
+
/* will free via device release */
put_device(&dev->dev);
#else
@@ -3268,13 +3322,13 @@ int unregister_netdevice(struct net_devi
dev_close(dev);

/* And unlink it from device chain. */
- for (dp = &dev_base; (d = *dp) != NULL; dp = &d->next) {
+ for (dp = &dev_base_ns(dev); (d = *dp) != NULL; dp = &d->next) {
if (d == dev) {
write_lock_bh(&dev_base_lock);
hlist_del(&dev->name_hlist);

```

```

    hlist_del(&dev->index_hlist);
-   if (dev_tail == &dev->next)
-   dev_tail = dp;
+   if (dev_tail_ns(dev) == &dev->next)
+   dev_tail_ns(dev) = dp;
    *dp = d->next;
    write_unlock_bh(&dev_base_lock);
    break;
--- linux-2.6.19-rc6-mm2.orig/net/core/net-sysfs.c
+++ linux-2.6.19-rc6-mm2/net/core/net-sysfs.c
@@ -453,6 +453,12 @@ static struct class net_class = {

void netdev_unregister_sysfs(struct net_device * net)
{
#ifdef CONFIG_NET_NS
+   if (net->net_ns != &init_net_ns)
+   /* not supported yet: sysfs virtualization is required */
+   return;
#endif
+
    device_del(&(net->dev));
}

@@ -462,6 +468,12 @@ int netdev_register_sysfs(struct net_dev
    struct device *dev = &(net->dev);
    struct attribute_group **groups = net->sysfs_groups;

#ifdef CONFIG_NET_NS
+   if (net->net_ns != &init_net_ns)
+   /* not supported yet: sysfs virtualization is required */
+   return 0;
#endif
+
    device_initialize(dev);
    dev->class = &net_class;
    dev->platform_data = net;
@@ -481,6 +493,17 @@ int netdev_register_sysfs(struct net_dev
    return device_add(dev);
}

+int netdev_rename_sysfs(struct net_device *net)
+{
#ifdef CONFIG_NET_NS
+   if (net->net_ns != &init_net_ns)
+   /* not supported yet: sysfs virtualization is required */
+   return 0;
#endif
+

```

```

+ return device_rename(&net->dev, net->name);
+}
+
int netdev_sysfs_init(void)
{
    return class_register(&net_class);
--- linux-2.6.19-rc6-mm2.orig/net/core/net_namespace.c
+++ linux-2.6.19-rc6-mm2/net/core/net_namespace.c
@@ -9,12 +9,15 @@
#include <linux/version.h>
#include <linux/nsproxy.h>
#include <linux/net_namespace.h>
+#include <linux/net.h>

struct net_namespace init_net_ns = {
    .kref = {
        .refcount = ATOMIC_INIT(2),
    },
    .ns = &init_nsproxy,
+ .dev_base_p = NULL,
+ .dev_tail_p = &init_net_ns.dev_base_p,
};

#ifdef CONFIG_NET_NS
@@ -34,6 +37,9 @@ static struct net_namespace *clone_net_n

    kref_init(&ns->kref);
    ns->ns = old_ns->ns;
+ ns->dev_base_p = NULL;
+ ns->dev_tail_p = &ns->dev_base_p;
+ ns->hash = net_random();
    return ns;
}

@@ -72,6 +78,11 @@ void free_net_ns(struct kref *kref)
    struct net_namespace *ns;

    ns = container_of(kref, struct net_namespace, kref);
+ if (ns->dev_base_p != NULL) {
+     printk("NET_NS: BUG: namespace %p has devices! ref %d\n",
+         ns, atomic_read(&ns->kref.refcount));
+     return;
+ }
    kfree(ns);
}

```

Containers mailing list

Subject: Re: [PATCH 2/12] L2 network namespace: network devices virtualization
Posted by [ebiederm](#) on Fri, 08 Dec 2006 20:14:38 GMT
[View Forum Message](#) <> [Reply to Message](#)

Dmitry Mishin <dim@openvz.org> writes:

```
> Added ability to have per-namespace network devices.
>
> Signed-off-by: Dmitry Mishin <dim@openvz.org>
>
> ---
> include/linux/net_namespace.h | 6 ++-
> include/linux/netdevice.h     | 10 +++++
> net/core/dev.c | 78 ++++++++++++++++++++++++++++++++++++++-----
> net/core/net-sysfs.c          | 23 ++++++++
> net/core/net_namespace.c      | 11 +++++
> 5 files changed, 114 insertions(+), 14 deletions(-)
>
> --- linux-2.6.19-rc6-mm2.orig/include/linux/net_namespace.h
> +++ linux-2.6.19-rc6-mm2/include/linux/net_namespace.h
> @@ -6,8 +6,10 @@
> #include <linux/errno.h>
>
> struct net_namespace {
> - struct kref kref;
> - struct nsproxy *ns;
> + struct kref kref;
> + struct nsproxy *ns;
> + struct net_device *dev_base_p, **dev_tail_p;
> + unsigned int hash;
> };
>
> extern struct net_namespace init_net_ns;
> --- linux-2.6.19-rc6-mm2.orig/include/linux/netdevice.h
> +++ linux-2.6.19-rc6-mm2/include/linux/netdevice.h
> @@ -379,6 +379,9 @@ struct net_device
> int promiscuity;
> int allmulti;
>
> +#ifdef CONFIG_NET_NS
> + struct net_namespace *net_ns;
> +#endif
>
> /* Protocol specific pointers */
```

```

>
> @@ -557,9 +560,16 @@ struct packet_type {
>
> #include <linux/interrupt.h>
> #include <linux/notifier.h>
> +#include <linux/net_namespace.h>
>
> extern struct net_device loopback_dev; /* The loopback */
> +#ifndef CONFIG_NET_NS
> extern struct net_device *dev_base; /* All devices */
> +#define dev_base_ns(dev) dev_base
> +#else
> +#define dev_base (current_net_ns->dev_base_p)
> +#define dev_base_ns(dev) (dev->net_ns->dev_base_p)
> +#endif
> extern rwlock_t dev_base_lock; /* Device list lock */
>
> extern int netdev_boot_setup_check(struct net_device *dev);
> --- linux-2.6.19-rc6-mm2.orig/net/core/dev.c
> +++ linux-2.6.19-rc6-mm2/net/core/dev.c
> @@ -90,6 +90,7 @@
> #include <linux/if_ether.h>
> #include <linux/netdevice.h>
> #include <linux/etherdevice.h>
> +#include <linux/net_namespace.h>
> #include <linux/notifier.h>
> #include <linux/skbuff.h>
> #include <net/sock.h>
> @@ -174,20 +175,28 @@ static spinlock_t net_dma_event_lock;
> * unregister_netdevice(), which must be called with the rtnl
> * semaphore held.
> */
> +#ifndef CONFIG_NET_NS
> struct net_device *dev_base;
> static struct net_device **dev_tail = &dev_base;
> -#DEFINE_RWLOCK(dev_base_lock);
> -
> +#define dev_tail_ns(dev) dev_tail
> EXPORT_SYMBOL(dev_base);
> +#else
> +#define dev_tail (current_net_ns->dev_tail_p)
> +#define dev_tail_ns(dev) (dev->net_ns->dev_tail_p)
> +#endif

```

No. Please do not hide a macro expansion (i.e. dev_tail).
That has the possibility of hiding costs, that shouldn't be hidden.

```

> --- linux-2.6.19-rc6-mm2.orig/net/core/net-sysfs.c

```

```
> +++ linux-2.6.19-rc6-mm2/net/core/net-sysfs.c
> @@ -453,6 +453,12 @@ static struct class net_class = {
>
> void netdev_unregister_sysfs(struct net_device * net)
> {
> +#ifdef CONFIG_NET_NS
> + if (net->net_ns != &init_net_ns)
> + /* not supported yet: sysfs virtualization is required */
> + return;
> +#endif
> +
> device_del(&(net->dev));
> }
```

This is something we need to seriously look at how to handle sysfs.

I believe separate directories per namespace and a magic symlink is going to be the sane approach.

Eric

Containers mailing list
Containers@lists.osdl.org
<https://lists.osdl.org/mailman/listinfo/containers>
