
Subject: [PATCH 2/12] L2 network namespace (v3): network devices virtualization
Posted by [Mishin Dmitry](#) on Wed, 17 Jan 2007 15:59:17 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 | 8 +++-
include/linux/netdevice.h    | 8 ++++
net/core/dev.c                | 77 ++++++-----
net/core/net-sysfs.c          | 23 ++++++
net/core/net_namespace.c      | 11 ++++++
5 files changed, 114 insertions(+), 13 deletions(-)
```

--- linux-2.6.20-rc4-mm1.net_ns.orig/include/linux/net_namespace.h

+++ linux-2.6.20-rc4-mm1.net_ns/include/linux/net_namespace.h

@@ -7,7 +7,9 @@

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

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

```
extern struct net_namespace init_net_ns;
```

@@ -60,6 +62,8 @@ static inline void pop_net_ns(struct net

```
#define net_ns_match(target, ns) ((target) == (ns))
```

```
+#define net_ns_hash(ns) ((ns)->hash)
```

```
+
```

```
#else /* CONFIG_NET_NS */
```

```
#define INIT_NET_NS(net_ns)
```

@@ -103,6 +107,8 @@ static inline void pop_net_ns(struct net

```
#define net_ns_match(target, ns) ((void)(ns), 1)
```

```
+#define net_ns_hash(ns) (0)
```

```
+
```

```
#endif /* !CONFIG_NET_NS */
```

```
#endif /* _LINUX_NET_NAMESPACE_H */
```

--- linux-2.6.20-rc4-mm1.net_ns.orig/include/linux/netdevice.h

```

+++ linux-2.6.20-rc4-mm1.net_ns/include/linux/netdevice.h
@@ -389,6 +389,7 @@ struct net_device
    int  promiscuity;
    int  allmulti;

+ struct net_namespace *net_ns;

    /* Protocol specific pointers */

@@ -567,9 +568,16 @@ 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-2.6.20-rc4-mm1.net_ns.orig/net/core/dev.c
+++ linux-2.6.20-rc4-mm1.net_ns/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 +175,27 @@ 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_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 +220,12 @@ #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 +485,13 @@ __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 (!strcmp(dev->name, name, IFNAMSIZ))
        return dev;
}

@@ -751,10 +764,11 @@ 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 +1495,11 @@ 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 +1695,10 @@ 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 +1785,7 @@ 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 +1804,8 @@ 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 +1874,7 @@ ncls:

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

@@ -2878,6 +2902,7 @@ 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 +2920,12 @@ 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 +2948,12 @@ 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 +3013,8 @@ 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 +3228,10 @@ 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 +3248,15 @@ 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 +3267,13 @@ 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 +3321,13 @@ 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-2.6.20-rc4-mm1.net_ns.orig/net/core/net-sysfs.c
+++ linux-2.6.20-rc4-mm1.net_ns/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-2.6.20-rc4-mm1.net_ns.orig/net/core/net_namespace.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/core/net_namespace.c
@@ -9,11 +9,14 @@
#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),
    },
+ .dev_base_p = NULL,
+ .dev_tail_p = &init_net_ns.dev_base_p,
};

#ifdef CONFIG_NET_NS
@@ -35,6 +38,9 @@ static struct net_namespace *clone_net_n
    return NULL;

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

@@ -73,6 +79,11 @@ 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

Containers@lists.osdl.org

<https://lists.osdl.org/mailman/listinfo/containers>
