
Subject: [PATCH 10/12] L2 network namespace (v3): playing with pass-through device

Posted by [Mishin Dmitry](#) on Wed, 17 Jan 2007 16:15:15 GMT

[View Forum Message](#) <> [Reply to Message](#)

Temporary code to debug and play with pass-through device.

Create device pair by

modprobe veth

echo 'add veth1 0:1:2:3:4:1 eth0 0:1:2:3:4:2' >/proc/net/veth_ctl

and your shell will appear into a new namespace with `eth0' device.

Configure device in this namespace

ip l s eth0 up

ip a a 1.2.3.4/24 dev eth0

and in the root namespace

ip l s veth1 up

ip a a 1.2.3.1/24 dev veth1

to establish a communication channel between root namespace and the newly created one.

Code is done by Andrey Savochkin and ported by me over Cedric's patchset

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

```
drivers/net/veth.c      | 121 ++++++
fs/proc/array.c        |  8 +++
kernel/fork.c          |  1
kernel/nsproxy.c       |  1
net/core/net_namespace.c |  3 +
5 files changed, 134 insertions(+)
```

--- linux-2.6.20-rc4-mm1.net_ns.orig/drivers/net/veth.c

+++ linux-2.6.20-rc4-mm1.net_ns/drivers/net/veth.c

@@ -12,6 +12,7 @@

#include <linux/etherdevice.h>

#include <linux/proc_fs.h>

#include <linux/seq_file.h>

+#include <linux/syscalls.h>

#include <net/dst.h>

#include <net/xfrm.h>

@@ -245,6 +246,123 @@ void veth_entry_del_all(void)

/* ----- *

*

+ * Temporary interface to create veth devices

+ *

+ * ----- */

```

+
+ #ifdef CONFIG_PROC_FS
+
+ static int veth_debug_open(struct inode *inode, struct file *file)
+ {
+     return 0;
+ }
+
+ static char *parse_addr(char *s, char *addr)
+ {
+     int i, v;
+
+     for (i = 0; i < ETH_ALEN; i++) {
+         if (!isxdigit(*s))
+             return NULL;
+         *addr = 0;
+         v = isdigit(*s) ? *s - '0' : toupper(*s) - 'A' + 10;
+         s++;
+         if (isxdigit(*s)) {
+             *addr += v << 16;
+             v = isdigit(*s) ? *s - '0' : toupper(*s) - 'A' + 10;
+             s++;
+         }
+         *addr++ += v;
+         if (i < ETH_ALEN - 1 && ispunct(*s))
+             s++;
+     }
+     return s;
+ }
+
+ static ssize_t veth_debug_write(struct file *file, const char __user *user_buf,
+     size_t size, loff_t *ppos)
+ {
+     char buf[128], *s, *parent_name, *child_name;
+     char parent_addr[ETH_ALEN], child_addr[ETH_ALEN];
+     struct net_namespace *parent_ns, *child_ns;
+     int err;
+
+     s = buf;
+     err = -EINVAL;
+     if (size >= sizeof(buf))
+         goto out;
+     err = -EFAULT;
+     if (copy_from_user(buf, user_buf, size))
+         goto out;
+     buf[size] = 0;
+
+     err = -EBADRQC;

```

```

+ if (!strncmp(buf, "add ", 4)) {
+   parent_name = buf + 4;
+   if ((s = strchr(parent_name, ' ')) == NULL)
+     goto out;
+   *s = 0;
+   if ((s = parse_addr(s + 1, parent_addr)) == NULL)
+     goto out;
+   if (!*s)
+     goto out;
+   child_name = s + 1;
+   if ((s = strchr(child_name, ' ')) == NULL)
+     goto out;
+   *s = 0;
+   if ((s = parse_addr(s + 1, child_addr)) == NULL)
+     goto out;
+
+   get_net_ns(current_net_ns);
+   parent_ns = current_net_ns;
+   if (*s == ' ') {
+     unsigned int id;
+     id = simple_strtoul(s + 1, &s, 0);
+     err = sys_bind_ns(id, NS_ALL);
+   } else
+     err = sys_unshare(CLONE_NEWNET2);
+   if (err)
+     goto out;
+   /* after bind_ns() or unshare_ns() namespace is changed */
+   get_net_ns(current_net_ns);
+   child_ns = current_net_ns;
+   err = veth_entry_add(parent_name, parent_addr, parent_ns,
+     child_name, child_addr, child_ns);
+   if (err) {
+     put_net_ns(child_ns);
+     put_net_ns(parent_ns);
+   } else
+     err = size;
+ }
+out:
+ return err;
+}
+
+static struct file_operations veth_debug_ops = {
+ .open = &veth_debug_open,
+ .write = &veth_debug_write,
+};
+
+static int veth_debug_create(void)
+{

```



```

    put_group_info(group_info);

    buffer += sprintf(buffer, "\n");
+
+ #ifdef CONFIG_NET_NS
+ if (p == current)
+   buffer += sprintf(buffer, "NetContext: %p\n",
+   p->nsproxy->net_ns);
+ #endif
+
    return buffer;
}

--- linux-2.6.20-rc4-mm1.net_ns.orig/kernel/fork.c
+++ linux-2.6.20-rc4-mm1.net_ns/kernel/fork.c
@@ -1770,3 +1770,4 @@ bad_unshare_cleanup_thread:
bad_unshare_out:
    return err;
}
+EXPORT_SYMBOL_GPL(sys_unshare);
--- linux-2.6.20-rc4-mm1.net_ns.orig/kernel/nsproxy.c
+++ linux-2.6.20-rc4-mm1.net_ns/kernel/nsproxy.c
@@ -426,6 +426,7 @@ unlock:
    put_nsproxy(ns);
    return ret;
}
+EXPORT_SYMBOL(sys_bind_ns);

static int __init nshash_init(void)
{
--- 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
@@ -56,6 +56,8 @@ static struct net_namespace *clone_net_n
    if (loopback_init())
        goto out_loopback;
    pop_net_ns(old_ns);
+ printk(KERN_DEBUG "NET_NS: created new netcontext %p for %s "
+ "(pid=%d)\n", ns, current->comm, current->tgid);
    return ns;

out_loopback:
@@ -109,6 +111,7 @@ void free_net_ns(struct kref *kref)
    return;
}
ip_fib_struct_cleanup(ns);
+ printk(KERN_DEBUG "NET_NS: net namespace %p destroyed\n", ns);
    kfree(ns);
}

```

```
EXPORT_SYMBOL(free_net_ns);
```

Containers mailing list

Containers@lists.osdl.org

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