Subject: Re: [PATCH 12/12] network namespace: L2 and L3 intro
Posted by Mishin Dmitry on Thu, 07 Dec 2006 11:29:05 GMT
View Forum Message <> Reply to Message

On Thursday 07 December 2006 12:34, Cedric Le Goater wrote:
> I have issues with the hunks below. they don't apply :(
>
> Can I have an updated version ? and I'll release the full patchset ASAP.
Sorry, forgot to refresh.

 Inroduce two kind of network namespaces - level 2 and level 3. First one is
 namespace with full set of networking objects, while second one -
 socket-level with restricted set.

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

```
---
 drivers/net/veth.c           |    2 -
 include/linux/net_namespace.h |    5 +++-
 include/linux/nsproxy.h      |    5 ++--
 include/linux/sched.h        |    2 +
 kernel/fork.c                |    3 +-
 kernel/nsproxy.c             |    5 ++--
 net/core/net_namespace.c     |   43 +++++++++++++++++++++++++++++-------------
 7 files changed, 45 insertions(+), 20 deletions(-)

--- linux-2.6.19-rc6-mm2.orig/drivers/net/veth.c
+++ linux-2.6.19-rc6-mm2/drivers/net/veth.c
@@ -320,7 +320,7 @@ static ssize_t veth_debug_write(struct f
    id = simple_strtoul(s + 1, &s, 0);
    err = sys_bind_ns(id, NS_ALL);
   } else
-   err = sys_unshare_ns(NS_NET);
+   err = sys_unshare_ns(NS_NET2);
   if (err)
    goto out;
   /* after bind_ns() or unshare_ns() namespace is changed */
--- linux-2.6.19-rc6-mm2.orig/include/linux/net_namespace.h
+++ linux-2.6.19-rc6-mm2/include/linux/net_namespace.h
@@ -27,6 +27,9 @@ struct net_namespace {
  struct net_namespace *parent;
  struct list_head child_list, sibling_list;
  unsigned int  id;
+#define NET_NS_LEVEL2  1
+#define NET_NS_LEVEL3  2
+ unsigned int  level;
 };
```

```
 extern struct net_namespace init_net_ns;
@@ -85,7 +88,7 @@ static inline void get_net_ns(struct net
 static inline int unshare_net_ns(unsigned long unshare_flags,
      struct net_namespace **new_net)
 {
- if (unshare_flags & NS_NET)
+ if (!(flags & (NS_NET2 | NS_NET3)))
   return -EINVAL;

  return 0;
--- linux-2.6.19-rc6-mm2.orig/include/linux/nsproxy.h
+++ linux-2.6.19-rc6-mm2/include/linux/nsproxy.h
@@ -20,9 +20,10 @@ struct user_namespace;
 #define NS_UTS  0x00000002
 #define NS_IPC  0x00000004
 #define NS_PID  0x00000008
-#define NS_NET  0x00000010
+#define NS_NET2  0x00000010
 #define NS_USER  0x00000020
-#define NS_ALL  (NS_MNT|NS_UTS|NS_IPC|NS_PID|NS_NET|NS_USER)
+#define NS_NET3  0x00000040
+#define NS_ALL  (NS_MNT|NS_UTS|NS_IPC|NS_PID|NS_NET2|NS_USER)

 /*
  * A structure to contain pointers to all per-process
--- linux-2.6.19-rc6-mm2.orig/include/linux/sched.h
+++ linux-2.6.19-rc6-mm2/include/linux/sched.h
@@ -26,6 +26,8 @@
 #define CLONE_STOPPED  0x02000000 /* Start in stopped state */
 #define CLONE_NEWUTS  0x04000000 /* New utsname group? */
 #define CLONE_NEWIPC  0x08000000 /* New ipcs */
+#define CLONE_NEWNET2  0x10000000 /* New level 2 network namespace */
+#define CLONE_NEWNET3  0x20000000 /* New level 3 network namespace */

 /*
  * Scheduling policies
--- linux-2.6.19-rc6-mm2.orig/kernel/fork.c
+++ linux-2.6.19-rc6-mm2/kernel/fork.c
@@ -1627,7 +1627,8 @@ asmlinkage long sys_unshare(unsigned lon
  err = -EINVAL;
  if (unshare_flags & ~(CLONE_THREAD|CLONE_FS|CLONE_NEWNS|CLONE_SIGHAND|
     CLONE_VM|CLONE_FILES|CLONE_SYSVSEM|
-    CLONE_NEWUTS|CLONE_NEWIPC))
+    CLONE_NEWUTS|CLONE_NEWIPC|CLONE_NEWNET2|
+    CLONE_NEWNET3))
   goto bad_unshare_out;
```

```
  if ((err = unshare_thread(unshare_flags)))
--- linux-2.6.19-rc6-mm2.orig/kernel/nsproxy.c
+++ linux-2.6.19-rc6-mm2/kernel/nsproxy.c
@@ -120,7 +120,8 @@ int copy_namespaces(int flags, struct ta

  get_nsproxy(old_ns);

- if (!(flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC)))
+ if (!(flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC |
+     CLONE_NEWNET2 | CLONE_NEWNET3)))
   return 0;

  new_ns = clone_namespaces(old_ns);
@@ -329,7 +330,7 @@ static int switch_ns(int id, unsigned lo
    put_pid_ns(new_ns->pid_ns);
    new_ns->pid_ns = ns->pid_ns;
   }
-  if (flags & NS_NET) {
+  if (flags & (NS_NET2 | NS_NET3)) {
    get_net_ns(ns->net_ns);
    put_net_ns(new_ns->net_ns);
    new_ns->net_ns = ns->net_ns;
--- linux-2.6.19-rc6-mm2.orig/net/core/net_namespace.c
+++ linux-2.6.19-rc6-mm2/net/core/net_namespace.c
@@ -32,13 +32,19 @@ struct net_namespace init_net_ns = {

 /*
  * Clone a new ns copying an original net ns, setting refcount to 1
+ * @level: level of namespace to create
  * @old_ns: namespace to clone
- * Return NULL on error (failure to kmalloc), new ns otherwise
+ * Return ERR_PTR on error, new ns otherwise
  */
-static struct net_namespace *clone_net_ns(struct net_namespace *old_ns)
+static struct net_namespace *clone_net_ns(unsigned int level,
+      struct net_namespace *old_ns)
 {
  struct net_namespace *ns;

+ /* level 3 namespaces are incomplete in order to have childs */
+ if (current_net_ns->level == NET_NS_LEVEL3)
+  return ERR_PTR(-EPERM);
+
  ns = kmalloc(sizeof(struct net_namespace), GFP_KERNEL);
  if (!ns)
   return NULL;
@@ -55,19 +61,24 @@ static struct net_namespace *clone_net_n
  list_add_tail(&ns->sibling_list, &old_ns->child_list);
```

```
  spin_unlock_irq(&net_ns_list_lock);

+ if (level ==  NET_NS_LEVEL2) {
 #ifdef CONFIG_IP_MULTIPLE_TABLES
- INIT_LIST_HEAD(&ns->fib_rules_ops_list);
+  INIT_LIST_HEAD(&ns->fib_rules_ops_list);
 #endif
- if (ip_fib_struct_init(ns))
-  goto out_fib4;
+  if (ip_fib_struct_init(ns))
+   goto out_fib4;
+ }
+ ns->level = level;
  if (loopback_init(ns))
   goto out_loopback;
- printk(KERN_DEBUG "NET_NS: created new netcontext %p for %s "
-  "(pid=%d)\n", ns, current->comm, current->tgid);
+ printk(KERN_DEBUG "NET_NS: created new netcontext %p, level %u, "
+  "for %s (pid=%d)\n", ns, (ns->level == NET_NS_LEVEL2) ?
+     2 : 3, current->comm, current->tgid);
  return ns;

 out_loopback:
- ip_fib_struct_cleanup();
+ if (level ==  NET_NS_LEVEL2)
+  ip_fib_struct_cleanup();
 out_fib4:
  BUG_ON(atomic_read(&ns->kref.refcount) != 1);
  kfree(ns);
@@ -80,17 +91,22 @@ out_fib4:
 int unshare_net_ns(unsigned long unshare_flags,
      struct net_namespace **new_net)
 {
- if (unshare_flags & NS_NET) {
+ unsigned int level;
+
+ if (unshare_flags & (NS_NET2 | NS_NET3)) {
   if (!capable(CAP_SYS_ADMIN))
    return -EPERM;

-  *new_net = clone_net_ns(current->nsproxy->net_ns);
-  if (!*new_net)
-   return -ENOMEM;
+  level = (unshare_flags & NS_NET2) ? NET_NS_LEVEL2 :
+        NET_NS_LEVEL3;
+  *new_net = clone_net_ns(level, current->nsproxy->net_ns);
+  if (IS_ERR(*new_net))
+   return PTR_ERR(*new_net);
```

```
  }

  return 0;
 }
+EXPORT_SYMBOL(unshare_net_ns);

 int copy_net_ns(int flags, struct task_struct *tsk)
 {
@@ -127,7 +143,8 @@ void free_net_ns(struct kref *kref)
      ns, atomic_read(&ns->kref.refcount));
   return;
  }
- ip_fib_struct_cleanup();
+ if (ns->level == NET_NS_LEVEL2)
+  ip_fib_struct_cleanup();
  printk(KERN_DEBUG "NET_NS: net namespace %p (%u) destroyed\n",
    ns, ns->id);
  kfree(ns);
```

_____

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