Subject: [PATCH 3/5] Hide the dead code in the net_namespace.c
Posted by Pavel Emelianov on Wed, 31 Oct 2007 18:27:00 GMT
View Forum Message <> Reply to Message

The namespace creation/destruction code is never called
if the CONFIG_NET_NS is n, so it's OK to move it under
appropriate ifdef.

The copy_net_ns() in the "n" case checks for flags and
returns -EINVAL when new net ns is requested. In a perfect
world this stub must be in net_namespace.h, but this
function need to know the CLONE_NEWNET value and thus
requires sched.h. On the other hand this header is to be
injected into almost every .c file in the networking code,
and making all this code depend on the sched.h is a
suicidal attempt.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

```
diff --git a/net/core/net_namespace.c b/net/core/net_namespace.c
index 4e52921..d5bf8b2 100644
--- a/net/core/net_namespace.c
+++ b/net/core/net_namespace.c
@@ -22,65 +22,6 @@ static struct kmem_cache *net_cachep;
 struct net init_net;
 EXPORT_SYMBOL_GPL(init_net);

-static struct net *net_alloc(void)
-{
- return kmem_cache_zalloc(net_cachep, GFP_KERNEL);
-}
-
-static void net_free(struct net *net)
-{
- if (!net)
-  return;
-
- if (unlikely(atomic_read(&net->use_count) != 0)) {
-  printk(KERN_EMERG "network namespace not free! Usage: %d\n",
-    atomic_read(&net->use_count));
-  return;
- }
-
- kmem_cache_free(net_cachep, net);
-}
-
```

```
-static void cleanup_net(struct work_struct *work)
-{
- struct pernet_operations *ops;
- struct net *net;
-
- net = container_of(work, struct net, work);
-
- mutex_lock(&net_mutex);
-
- /* Don't let anyone else find us. */
- rtnl_lock();
- list_del(&net->list);
- rtnl_unlock();
-
- /* Run all of the network namespace exit methods */
- list_for_each_entry_reverse(ops, &pernet_list, list) {
-  if (ops->exit)
-   ops->exit(net);
- }
-
- mutex_unlock(&net_mutex);
-
- /* Ensure there are no outstanding rcu callbacks using this
-  * network namespace.
-  */
- rcu_barrier();
-
- /* Finally it is safe to free my network namespace structure */
- net_free(net);
-}
-
-
-void __put_net(struct net *net)
-{
- /* Cleanup the network namespace in process context */
- INIT_WORK(&net->work, cleanup_net);
- schedule_work(&net->work);
-}
-EXPORT_SYMBOL_GPL(__put_net);
-
 /*
  * setup_net runs the initializers for the network namespace object.
  */
@@ -117,6 +58,12 @@ out_undo:
  goto out;
 }

+#ifdef CONFIG_NET_NS
```

```
+static struct net *net_alloc(void)
+{
+ return kmem_cache_zalloc(net_cachep, GFP_KERNEL);
+}
+
 struct net *copy_net_ns(unsigned long flags, struct net *old_net)
 {
  struct net *new_net = NULL;
@@ -127,10 +74,6 @@ struct net *copy_net_ns(unsigned long flags, struct net *old_net)
  if (!(flags & CLONE_NEWNET))
   return old_net;

-#ifndef CONFIG_NET_NS
- return ERR_PTR(-EINVAL);
-#endif
-
  err = -ENOMEM;
  new_net = net_alloc();
  if (!new_net)
@@ -157,6 +100,68 @@ out:
  return new_net;
 }

+static void net_free(struct net *net)
+{
+ if (!net)
+  return;
+
+ if (unlikely(atomic_read(&net->use_count) != 0)) {
+  printk(KERN_EMERG "network namespace not free! Usage: %d\n",
+    atomic_read(&net->use_count));
+  return;
+ }
+
+ kmem_cache_free(net_cachep, net);
+}
+
+static void cleanup_net(struct work_struct *work)
+{
+ struct pernet_operations *ops;
+ struct net *net;
+
+ net = container_of(work, struct net, work);
+
+ mutex_lock(&net_mutex);
+
+ /* Don't let anyone else find us. */
+ rtnl_lock();
```

```
+ list_del(&net->list);
+ rtnl_unlock();
+
+ /* Run all of the network namespace exit methods */
+ list_for_each_entry_reverse(ops, &pernet_list, list) {
+  if (ops->exit)
+   ops->exit(net);
+ }
+
+ mutex_unlock(&net_mutex);
+
+ /* Ensure there are no outstanding rcu callbacks using this
+  * network namespace.
+  */
+ rcu_barrier();
+
+ /* Finally it is safe to free my network namespace structure */
+ net_free(net);
+}
+
+void __put_net(struct net *net)
+{
+ /* Cleanup the network namespace in process context */
+ INIT_WORK(&net->work, cleanup_net);
+ schedule_work(&net->work);
+}
+EXPORT_SYMBOL_GPL(__put_net);
+
+#else
+struct net *copy_net_ns(unsigned long flags, struct net *old_net)
+{
+ if (flags & CLONE_NEWNET)
+  return ERR_PTR(-EINVAL);
+ return old_net;
+}
+#endif
+
 static int __init net_ns_init(void)
 {
  int err;
--
1.5.3.4
```