Subject: [PATCH RFC 2/31] net: Implement a place holder network namespace
Posted by ebiederm on Thu, 25 Jan 2007 19:00:04 GMT
View Forum Message <> Reply to Message

From: Eric W. Biederman <ebiederm@xmission.com> - unquoted

Many of the changes to the network stack will simply be adding a network namespace parameter to function calls or moving variables from globals to being per network namespace. When those variables have initializers that cannot statically compute the proper value, a function that runs at the creation and destruction of network namespaces will need to be registered, and the logic will need to be changed to accomidate that.

Adding unconditional support for these functions ensures that even when everything else is compiled out the modified network stack logic will continue to run correctly.

This patch adds struct pernet_operations that has an init (constructor) and an exit (destructor) method. When registered the init method is called for every existing namespace, and when unregistered the exit method is called for every existing namespace. When a new network namespace is created all of the init methods are called in the order in which they were registered, and when a network namespace is destroyed the exit methods are called in the reverse order in which they were registered.

There are two distinct types of pernet_operations recognized: subsys and device. At creation all subsys init functions are called before device init functions, and at destruction all device exit functions are called before subsys exit function. For other ordering the preservation of the order of registration combined with the various kinds of kernel initcalls should be sufficient.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

```
---
 include/net/net_namespace.h |   62 ++++++++++++++++++
 net/core/Makefile           |    2 +-
 net/core/net_namespace.c    |  149 ++++++++++++++++++++++++++++++++++++++++++++++
 3 files changed, 212 insertions(+), 1 deletions(-)

diff --git a/include/net/net_namespace.h b/include/net/net_namespace.h
new file mode 100644
index 0000000..06a9ba1
--- /dev/null
+++ b/include/net/net_namespace.h
@@ -0,0 +1,62 @@
+/*
```

```
+ * Operations on the network namespace
+ */
+#ifndef __NET_NET_NAMESPACE_H
+#define __NET_NET_NAMESPACE_H
+
+#include <asm/atomic.h>
+#include <linux/workqueue.h>
+#include <linux/nsproxy.h>
+#include <linux/net_namespace_type.h>
+
+/* How many bytes in each network namespace should we allocate
+ * for use by modules when they are loaded.
+ */
+#ifdef CONFIG_MODULES
+# define PER_NET_MODULE_RESERVE 2048
+#else
+# define PER_NET_MODULE_RESERVE 0
+#endif
+
+struct net_namespace_head {
+ atomic_t count;  /* To decided when the network namespace
+     * should go
+     */
+ atomic_t use_count; /* For references we destroy on demand */
+ struct list_head list;
+ struct work_struct work;
+};
+
+static inline net_t get_net(net_t net) { return net; }
+static inline void put_net(net_t net) {}
+static inline net_t hold_net(net_t net) { return net; }
+static inline void release_net(net_t net) {}
+
+#define __per_net_start ((char *)0)
+#define __per_net_end ((char *)0)
+
+static inline int copy_net(int flags, struct task_struct *tsk) { return 0; }
+
+/* Don't let the list of network namespaces change */
+static inline void net_lock(void) {}
+static inline void net_unlock(void) {}
+
+#define for_each_net(VAR) if (1)
+
+extern net_t net_template;
+
+#define NET_CREATE 0x0001 /* A network namespace has been created */
+#define NET_DESTROY 0x0002 /* A network namespace is being destroyed */
```

```
+
+struct pernet_operations {
+ struct list_head list;
+ int (*init)(net_t net);
+ void (*exit)(net_t net);
+};
+
+extern int register_pernet_subsys(struct pernet_operations *);
+extern void unregister_pernet_subsys(struct pernet_operations *);
+extern int register_pernet_device(struct pernet_operations *);
+extern void unregister_pernet_device(struct pernet_operations *);
+
+#endif /* __NET_NET_NAMESPACE_H */
diff --git a/net/core/Makefile b/net/core/Makefile
index 73272d5..554dbdc 100644
--- a/net/core/Makefile
+++ b/net/core/Makefile
@@ -3,7 +3,7 @@
 #

 obj-y := sock.o request_sock.o skbuff.o iovec.o datagram.o stream.o scm.o \
-  gen_stats.o gen_estimator.o
+  gen_stats.o gen_estimator.o net_namespace.o

 obj-$(CONFIG_SYSCTL) += sysctl_net_core.o

diff --git a/net/core/net_namespace.c b/net/core/net_namespace.c
new file mode 100644
index 0000000..4ae266d
--- /dev/null
+++ b/net/core/net_namespace.c
@@ -0,0 +1,149 @@
+#include <linux/rtnetlink.h>
+#include <net/net_namespace.h>
+
+/*
+ * Our network namespace constructor/destructor lists
+ */
+
+static LIST_HEAD(pernet_list);
+static struct list_head *first_device = &pernet_list;
+static DEFINE_MUTEX(net_mutex);
+net_t net_template;
+
+static int register_pernet_operations(struct list_head *list,
+          struct pernet_operations *ops)
+{
+ net_t net, undo_net;
```

```
+ int error;
+
+ error = 0;
+ list_add_tail(&ops->list, list);
+ for_each_net(net) {
+  if (ops->init) {
+   error = ops->init(net);
+   if (error)
+    goto out_undo;
+  }
+ }
+out:
+ return error;
+
+out_undo:
+ /* If I have an error cleanup all namespaces I initialized */
+ list_del(&ops->list);
+ for_each_net(undo_net) {
+  if (net_eq(undo_net, net))
+   goto undone;
+  if (ops->exit)
+   ops->exit(undo_net);
+ }
+undone:
+ goto out;
+}
+
+static void unregister_pernet_operations(struct pernet_operations *ops)
+{
+ net_t net;
+
+ list_del(&ops->list);
+ for_each_net(net)
+  if (ops->exit)
+   ops->exit(net);
+}
+
+/**
+ *      register_pernet_subsys - register a network namespace subsystem
+ * @ops:  pernet operations structure for the subsystem
+ *
+ * Register a subsystem which has init and exit functions
+ * that are called when network namespaces are created and
+ * destroyed respectively.
+ *
+ * When registered all network namespace init functions are
+ * called for every existing network namespace.  Allowing kernel
+ * modules to have a race free view of the set of network namespaces.
```

```
+ *
+ * When a new network namespace is created all of the init
+ * methods are called in the order in which they were registered.
+ *
+ * When a network namespace is destroyed all of the exit methods
+ * are called in the reverse of the order with which they were
+ * registered.
+ */
+int register_pernet_subsys(struct pernet_operations *ops)
+{
+ int error;
+ mutex_lock(&net_mutex);
+ error =  register_pernet_operations(first_device, ops);
+ mutex_unlock(&net_mutex);
+ return error;
+}
+EXPORT_SYMBOL_GPL(register_pernet_subsys);
+
+/**
+ *      unregister_pernet_subsys - unregister a network namespace subsystem
+ * @ops: pernet operations structure to manipulate
+ *
+ * Remove the pernet operations structure from the list to be
+ * used when network namespaces are created or destoryed.  In
+ * addition run the exit method for all existing network
+ * namespaces.
+ */
+void unregister_pernet_subsys(struct pernet_operations *module)
+{
+ mutex_lock(&net_mutex);
+ unregister_pernet_operations(module);
+ mutex_unlock(&net_mutex);
+}
+EXPORT_SYMBOL_GPL(unregister_pernet_subsys);
+
+/**
+ *      register_pernet_device - register a network namespace device
+ * @ops:  pernet operations structure for the subsystem
+ *
+ * Register a device which has init and exit functions
+ * that are called when network namespaces are created and
+ * destroyed respectively.
+ *
+ * When registered all network namespace init functions are
+ * called for every existing network namespace.  Allowing kernel
+ * modules to have a race free view of the set of network namespaces.
+ *
+ * When a new network namespace is created all of the init
```

```
+ * methods are called in the order in which they were registered.
+ *
+ * When a network namespace is destroyed all of the exit methods
+ * are called in the reverse of the order with which they were
+ * registered.
+ */
+int register_pernet_device(struct pernet_operations *ops)
+{
+ int error;
+ mutex_lock(&net_mutex);
+ error = register_pernet_operations(&pernet_list, ops);
+ if (!error && (first_device == &pernet_list))
+  first_device = &ops->list;
+ mutex_unlock(&net_mutex);
+ return error;
+}
+EXPORT_SYMBOL_GPL(register_pernet_device);
+
+/**
+ *      unregister_pernet_device - unregister a network namespace netdevice
+ * @ops: pernet operations structure to manipulate
+ *
+ * Remove the pernet operations structure from the list to be
+ * used when network namespaces are created or destoryed.  In
+ * addition run the exit method for all existing network
+ * namespaces.
+ */
+void unregister_pernet_device(struct pernet_operations *ops)
+{
+ mutex_lock(&net_mutex);
+ if (&ops->list == first_device)
+  first_device = first_device->next;
+ unregister_pernet_operations(ops);
+ mutex_unlock(&net_mutex);
+}
+EXPORT_SYMBOL_GPL(unregister_pernet_device);
+
--
1.4.4.1.g278f
```