Subject: [PATCH 00/16] core network namespace support
Posted by ebiederm on Sat, 08 Sep 2007 21:07:37 GMT
View Forum Message <> Reply to Message

The following patchset was built against the latest net-2.6.24
tree, and should be safe to apply assume not issues are found during
the review. In the interest of keeping the patcheset to a reviewable
size, just the core of the network stack has been covered.

The 10,000 foot overview. We want to make it look to user space
like the kernel implements multiple network stacks.

To implement this some of the currently global variables in the
network stack need to have one instance per network namespace,
or the global data structure needs to have a network namespace
field.

Currently control enters the network stack in one of 4 major ways.
Through operations on a socket, through a packet coming in
from a network device, through miscellaneous syscalls from
a process, and through operations on a virtual filesystem.
So the current design calls for placing a pointer to
struct net (the network namespace structure) on network
devices, sockets, processes, and on filesystems so we
have a clear understanding of which network namespace
operations should be done in the context of.

Packets do not contain a pointer to a network device structure.
Instead their network device is derived from which network
device or which socket they are passing through.

On the input path we only need to look at the network namespace
to determine which routing tables to use, and which sockets the
packet can be destined for.

Similarly on the output path we only need to consult the network
namespace for the output routing tables which point to which
network devices we can use.

So while there are accesses to the network namespace as
we process each packet they are in well contained spots that occur
rarely.

Where the network namespace appears most is on the control,
setup, and clean up code paths, in the network stack that we
change rarely. There we currently don't have anything except
a global context so modifications are necessary, but since

the network parameter is not implicit it should not require
much thought to use.

The implementation strategy follows the classic global
lock reduction pattern. First all of the interfaces
at a given level in the network stack are made to filter
out traffic from anything except the initial network namespace,
and then those interfaces are allowed to see packets from
any network namespace. Then some subset of those interfaces
are taught to handle packets from all namespaces, after the
more specific protocol layers below them have been made to
filter those packets.

What this means is that we start out with large intrusive
stupid patches and end up with small patches that enable
small bits of functionality in the secondary network
namespaces.

Eric

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: [PATCH 01/16] appletalk: In notifier handlers convert the void pointer to a netdevice
Posted by ebiederm on Sat, 08 Sep 2007 21:09:36 GMT
View Forum Message <> Reply to Message

This slightly improves code safetly and clarity.

Later network namespace patches touch this code so this is a
preliminary cleanup.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 net/appletalk/aarp.c | 7 ++++---
 net/appletalk/ddp.c | 4 +++-
 2 files changed, 7 insertions(+), 4 deletions(-)

diff --git a/net/appletalk/aarp.c b/net/appletalk/aarp.c
index 3d1655f..80b5414 100644
--- a/net/appletalk/aarp.c
+++ b/net/appletalk/aarp.c
@@ -330,15 +330,16 @@ static void aarp_expire_timeout(unsigned long unused)
 static int aarp_device_event(struct notifier_block *this, unsigned long event,
 		void *ptr)

```
 {
+ struct net_device *dev = ptr;
  int ct;

  if (event == NETDEV_DOWN) {
   write_lock_bh(&aarp_lock);

   for (ct = 0; ct < AARP_HASH_SIZE; ct++) {
-   __aarp_expire_device(&resolved[ct], ptr);
-   __aarp_expire_device(&unresolved[ct], ptr);
-   __aarp_expire_device(&proxies[ct], ptr);
+   __aarp_expire_device(&resolved[ct], dev);
+   __aarp_expire_device(&unresolved[ct], dev);
+   __aarp_expire_device(&proxies[ct], dev);
   }

   write_unlock_bh(&aarp_lock);
diff --git a/net/appletalk/ddp.c b/net/appletalk/ddp.c
index fbdfb12..594b597 100644
--- a/net/appletalk/ddp.c
+++ b/net/appletalk/ddp.c
@@ -647,9 +647,11 @@ static inline void atalk_dev_down(struct net_device *dev)
 static int ddp_device_event(struct notifier_block *this, unsigned long event,
        void *ptr)
 {
+ struct net_device *dev = ptr;
+
  if (event == NETDEV_DOWN)
   /* Discard any use of this */
-  atalk_dev_down(ptr);
+  atalk_dev_down(dev);

  return NOTIFY_DONE;
 }
--
1.5.3.rc6.17.g1911
```

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: [PATCH 02/16] net: Don't implement dev_ifname32 inline
Posted by ebiederm on Sat, 08 Sep 2007 21:13:04 GMT
View Forum Message <> Reply to Message

The current implementation of dev_ifname makes maintenance difficult

because updates to the implementation of the ioctl have to made in two
places. So this patch updates dev_ifname32 to do a classic 32/64
structure conversion and call sys_ioctl like the rest of the
compat calls do.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 fs/compat_ioctl.c | 21 ++++++++++-----------
 1 files changed, 10 insertions(+), 11 deletions(-)

diff --git a/fs/compat_ioctl.c b/fs/compat_ioctl.c
index a6c9078..361b994 100644
--- a/fs/compat_ioctl.c
+++ b/fs/compat_ioctl.c
@@ -324,22 +324,21 @@ struct ifconf32 {

 static int dev_ifname32(unsigned int fd, unsigned int cmd, unsigned long arg)
 {
- struct net_device *dev;
- struct ifreq32 ifr32;
+ struct ifreq __user *uifr;
  int err;

- if (copy_from_user(&ifr32, compat_ptr(arg), sizeof(ifr32)))
+ uifr = compat_alloc_user_space(sizeof(struct ifreq));
+ if (copy_in_user(uifr, compat_ptr(arg), sizeof(struct ifreq32)));
   return -EFAULT;

- dev = dev_get_by_index(ifr32.ifr_ifindex);
- if (!dev)
- return -ENODEV;
+ err = sys_ioctl(fd, SIOCGIFNAME, (unsigned long)uifr);
+ if (err)
+ return err;

- strlcpy(ifr32.ifr_name, dev->name, sizeof(ifr32.ifr_name));
- dev_put(dev);
-
- err = copy_to_user(compat_ptr(arg), &ifr32, sizeof(ifr32));
- return (err ? -EFAULT : 0);
+ if (copy_in_user(compat_ptr(arg), uifr, sizeof(struct ifreq32)))
+ return -EFAULT;
+
+ return 0;
 }

 static int dev_ifconf(unsigned int fd, unsigned int cmd, unsigned long arg)
--

1.5.3.rc6.17.g1911

_____

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: [PATCH 03/16] net: Basic network namespace infrastructure.
Posted by ebiederm on Sat, 08 Sep 2007 21:15:34 GMT
View Forum Message <> Reply to Message

This is the basic infrastructure needed to support network
namespaces. This infrastructure is:
- Registration functions to support initializing per network
  namespace data when a network namespaces is created or destroyed.

- struct net. The network namespace data structure.
  This structure will grow as variables are made per network
  namespace but this is the minimal starting point.

- Functions to grab a reference to the network namespace.
  I provide both get/put functions that keep a network namespace
  from being freed. And hold/release functions serve as weak references
  and will warn if their count is not zero when the data structure
  is freed. Useful for dealing with more complicated data structures
  like the ipv4 route cache.

- A list of all of the network namespaces so we can iterate over them.

- A slab for the network namespace data structure allowing leaks
  to be spotted.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 include/net/net_namespace.h | 68 ++++++++++++
 net/core/Makefile | 2 +-
 net/core/net_namespace.c | 292 ++++++++++++

```
+/*
+ * 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/list.h>
+
+struct net {
+ atomic_t  count;  /* To decided when the network
+       *  namespace should be freed.
+       */
+ atomic_t  use_count; /* To track references we
+       * destroy on demand
+       */
+ struct list_head list;  /* list of network namespaces */
+ struct work_struct work;  /* work struct for freeing */
+};
+
+extern struct net init_net;
+extern struct list_head net_namespace_list;
+
+extern void __put_net(struct net *net);
+
+static inline struct net *get_net(struct net *net)
+{
+ atomic_inc(&net->count);
+ return net;
+}
+
+static inline void put_net(struct net *net)
+{
+ if (atomic_dec_and_test(&net->count))
+  __put_net(net);
+}
+
+static inline struct net *hold_net(struct net *net)
+{
+ atomic_inc(&net->use_count);
+ return net;
+}
+
+static inline void release_net(struct net *net)
+{
+ atomic_dec(&net->use_count);
+}
```

```
+
+extern void net_lock(void);
+extern void net_unlock(void);
+
+#define for_each_net(VAR)    \
+ list_for_each_entry(VAR, &net_namespace_list, list)
+
+
+struct pernet_operations {
+ struct list_head list;
+ int (*init)(struct net *net);
+ void (*exit)(struct net *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 4751613..ea9b3f3 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..f259a9b
--- /dev/null
+++ b/net/core/net_namespace.c
@@ -0,0 +1,292 @@
+#include <linux/workqueue.h>
+#include <linux/rtnetlink.h>
+#include <linux/cache.h>
+#include <linux/slab.h>
+#include <linux/list.h>
+#include <linux/delay.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);
+
+static DEFINE_MUTEX(net_list_mutex);
+LIST_HEAD(net_namespace_list);
+
+static struct kmem_cache *net_cachep;
+
+struct net init_net;
+EXPORT_SYMBOL_GPL(init_net);
+
+void net_lock(void)
+{
+ mutex_lock(&net_list_mutex);
+}
+
+void net_unlock(void)
+{
+ mutex_unlock(&net_list_mutex);
+}
+
+static struct net *net_alloc(void)
+{
+ return kmem_cache_alloc(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 list_head *ptr;
```

```
+ struct net *net;
+
+ net = container_of(work, struct net, work);
+
+ mutex_lock(&net_mutex);
+
+ /* Don't let anyone else find us. */
+ net_lock();
+ list_del(&net->list);
+ net_unlock();
+
+ /* Run all of the network namespace exit methods */
+ list_for_each_prev(ptr, &pernet_list) {
+  ops = list_entry(ptr, struct pernet_operations, 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.
+ */
+static int setup_net(struct net *net)
+{
+ /* Must be called with net_mutex held */
+ struct pernet_operations *ops;
+ struct list_head *ptr;
+ int error;
+
```

```
+ memset(net, 0, sizeof(struct net));
+ atomic_set(&net->count, 1);
+ atomic_set(&net->use_count, 0);
+
+ error = 0;
+ list_for_each(ptr, &pernet_list) {
+  ops = list_entry(ptr, struct pernet_operations, list);
+  if (ops->init) {
+   error = ops->init(net);
+   if (error < 0)
+    goto out_undo;
+  }
+ }
+out:
+ return error;
+out_undo:
+ /* Walk through the list backwards calling the exit functions
+  * for the pernet modules whose init functions did not fail.
+  */
+ for (ptr = ptr->prev; ptr != &pernet_list; ptr = ptr->prev) {
+  ops = list_entry(ptr, struct pernet_operations, list);
+  if (ops->exit)
+   ops->exit(net);
+ }
+ goto out;
+}
+
+static int __init net_ns_init(void)
+{
+ int err;
+
+ printk(KERN_INFO "net_namespace: %zd bytes\n", sizeof(struct net));
+ net_cachep = kmem_cache_create("net_namespace", sizeof(struct net),
+     SMP_CACHE_BYTES,
+     SLAB_PANIC, NULL);
+ mutex_lock(&net_mutex);
+ err = setup_net(&init_net);
+
+ net_lock();
+ list_add_tail(&init_net.list, &net_namespace_list);
+ net_unlock();
+
+ mutex_unlock(&net_mutex);
+ if (err)
+  panic("Could not setup the initial network namespace");
+
+ return 0;
+}
```

```
+
+pure_initcall(net_ns_init);
+
+static int register_pernet_operations(struct list_head *list,
+          struct pernet_operations *ops)
+{
+ struct net *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 (undo_net == net)
+   goto undone;
+  if (ops->exit)
+   ops->exit(undo_net);
+ }
+undone:
+ goto out;
+}
+
+static void unregister_pernet_operations(struct pernet_operations *ops)
+{
+ struct net *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);
--
1.5.3.rc6.17.g1911
```

_____

Containers mailing list

Subject: [PATCH 04/16] net: Add a network namespace parameter to tasks
Posted by ebiederm on Sat, 08 Sep 2007 21:17:03 GMT
View Forum Message <> Reply to Message

This is the network namespace from which all which all sockets
and anything else under user control ultimately get their network
namespace parameters.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 include/linux/init_task.h | 2 ++
 include/linux/nsproxy.h | 1 +
 2 files changed, 3 insertions(+), 0 deletions(-)

```
diff --git a/include/linux/init_task.h b/include/linux/init_task.h
index cab741c..685d631 100644
--- a/include/linux/init_task.h
+++ b/include/linux/init_task.h
@@ -9,6 +9,7 @@
 #include <linux/ipc.h>
 #include <linux/pid_namespace.h>
 #include <linux/user_namespace.h>
+#include <net/net_namespace.h>

 #define INIT_FDTABLE \
 {       \
@@ -78,6 +79,7 @@ extern struct nsproxy init_nsproxy;
  .nslock  = __SPIN_LOCK_UNLOCKED(nsproxy.nslock),  \
  .uts_ns  = &init_uts_ns,     \
  .mnt_ns  = NULL,      \
+ .net_ns  = &init_net,     \
  INIT_IPC_NS(ipc_ns)      \
  .user_ns = &init_user_ns,    \
 }
diff --git a/include/linux/nsproxy.h b/include/linux/nsproxy.h
index ce06188..bec4485 100644
--- a/include/linux/nsproxy.h
+++ b/include/linux/nsproxy.h
@@ -29,6 +29,7 @@ struct nsproxy {
  struct mnt_namespace *mnt_ns;
  struct pid_namespace *pid_ns;
  struct user_namespace *user_ns;
+ struct net       *net_ns;
 };
```

```
 extern struct nsproxy init_nsproxy;

--
1.5.3.rc6.17.g1911
```

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: [PATCH 05/16] net: Add a network namespace tag to struct net_device
Posted by ebiederm on Sat, 08 Sep 2007 21:18:12 GMT
View Forum Message <> Reply to Message

Please note that network devices do not increase the count
count on the network namespace. The are inside the network
namespace and so the network namespace tag is in the nature
of a back pointer and so getting and putting the network namespace
is unnecessary.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 include/linux/netdevice.h | 4 ++++
 1 files changed, 4 insertions(+), 0 deletions(-)

```
diff --git a/include/linux/netdevice.h b/include/linux/netdevice.h
index 9b26594..8eeced0 100644
--- a/include/linux/netdevice.h
+++ b/include/linux/netdevice.h
@@ -41,6 +41,7 @@
 #include <linux/dmaengine.h>
 #include <linux/workqueue.h>

+struct net;
 struct vlan_group;
 struct ethtool_ops;
 struct netpoll_info;
@@ -649,6 +650,9 @@ struct net_device
  void                  (*poll_controller)(struct net_device *dev);
 #endif

+ /* Network namespace this network device is inside */
+ struct net  *nd_net;
+
  /* bridge stuff */
  struct net_bridge_port *br_port;
  /* macvlan */
```

--
1.5.3.rc6.17.g1911

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: [PATCH 07/16] net: Make /proc/net per network namespace
Posted by ebiederm on Sat, 08 Sep 2007 21:20:36 GMT
View Forum Message <> Reply to Message

This patch makes /proc/net per network namespace. It modifies the global
variables proc_net and proc_net_stat to be per network namespace.
The proc_net file helpers are modified to take a network namespace argument,
and all of their callers are fixed to pass &init_net for that argument.
This ensures that all of the /proc/net files are only visible and
usable in the initial network namespace until the code behind them
has been updated to be handle multiple network namespaces.

Making /proc/net per namespace is necessary as at least some files
in /proc/net depend upon the set of network devices which is per
network namespace, and even more files in /proc/net have contents
that are relevant to a single network namespace.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
drivers/isdn/divert/divert_procfs.c | 7 +-
drivers/isdn/hardware/eicon/diva_didd.c | 5 +-
drivers/isdn/hysdn/hysdn_procconf.c | 5 +-
drivers/net/bonding/bond_main.c | 7 +-
drivers/net/hamradio/bpqether.c | 5 +-
drivers/net/hamradio/scc.c | 5 +-
drivers/net/hamradio/yam.c | 5 +-
drivers/net/ibmveth.c | 7 +-
drivers/net/pppoe.c | 5 +-
drivers/net/pppol2tp.c | 5 +-
drivers/net/tokenring/lanstreamer.c | 5 +-
drivers/net/tokenring/olympic.c | 9 +-
drivers/net/wireless/hostap/hostap_main.c | 7 +-
drivers/net/wireless/strip.c | 5 +-
fs/proc/Makefile | 1 +
fs/proc/internal.h | 5 +
fs/proc/proc_net.c | 192 ++++++++++++++++++++++
fs/proc/root.c | 8 +-
include/linux/proc_fs.h | 44 ++++---
include/net/net_namespace.h | 5 +

```
net/802/tr.c                                  |    3 +-
net/8021q/vlanproc.c                          |    5 +-
net/appletalk/atalk_proc.c                    |    7 +-
net/atm/proc.c                                |    5 +-
net/ax25/af_ax25.c                            |   13 +-
net/core/dev.c                                |   19 +-
net/core/dev_mcast.c                          |    3 +-
net/core/neighbour.c                          |    3 +-
net/core/pktgen.c                             |    9 +-
net/core/sock.c                               |    3 +-
net/dccp/probe.c                              |    7 +-
net/decnet/af_decnet.c                        |    5 +-
net/decnet/dn_dev.c                           |    5 +-
net/decnet/dn_neigh.c                         |    5 +-
net/decnet/dn_route.c                         |    5 +-
net/ieee80211/ieee80211_module.c              |    7 +-
net/ipv4/arp.c                                |    3 +-
net/ipv4/fib_hash.c                           |    5 +-
net/ipv4/fib_trie.c                           |   17 +-
net/ipv4/igmp.c                               |    5 +-
net/ipv4/ipconfig.c                           |    3 +-
net/ipv4/ipmr.c                               |    5 +-
net/ipv4/ipvs/ip_vs_app.c                     |    5 +-
net/ipv4/ipvs/ip_vs_conn.c                    |    5 +-
net/ipv4/ipvs/ip_vs_ctl.c                     |    9 +-
net/ipv4/ipvs/ip_vs_lblcr.c                   |    5 +-
net/ipv4/netfilter/ip_queue.c                 |    8 +-
net/ipv4/netfilter/ipt_CLUSTERIP.c            |    3 +-
net/ipv4/netfilter/ipt_recent.c               |    5 +-
.../netfilter/nf_conntrack_l3proto_ipv4_compat.c |   17 +-
net/ipv4/proc.c                               |   11 +-
net/ipv4/raw.c                                |    5 +-
net/ipv4/route.c                              |    7 +-
net/ipv4/tcp_ipv4.c                           |    5 +-
net/ipv4/tcp_probe.c                          |    7 +-
net/ipv4/udp.c                                |    5 +-
net/ipv6/addrconf.c                           |    7 +-
net/ipv6/anycast.c                            |    5 +-
net/ipv6/ip6_flowlabel.c                      |    5 +-
net/ipv6/mcast.c                              |    9 +-
net/ipv6/netfilter/ip6_queue.c                |    7 +-
net/ipv6/proc.c                               |   17 +-
net/ipv6/raw.c                                |    5 +-
net/ipv6/route.c                              |    9 +-
net/ipx/ipx_proc.c                            |    7 +-
net/irda/irproc.c                             |    5 +-
net/key/af_key.c                              |    5 +-
net/llc/llc_proc.c                            |    7 +-
```

```
 net/netfilter/core.c                          |    3 +-
 net/netfilter/nf_conntrack_expect.c           |    5 +-
 net/netfilter/nf_conntrack_standalone.c       |   13 +-
 net/netfilter/x_tables.c                      |   17 +-
 net/netfilter/xt_hashlimit.c                  |   11 +-
 net/netlink/af_netlink.c                      |    3 +-
 net/netrom/af_netrom.c                        |   13 +-
 net/packet/af_packet.c                        |    5 +-
 net/rose/af_rose.c                            |   17 +-
 net/rxrpc/af_rxrpc.c                          |    9 +-
 net/sched/sch_api.c                           |    3 +-
 net/sctp/protocol.c                           |    5 +-
 net/sunrpc/stats.c                            |    5 +-
 net/unix/af_unix.c                            |    5 +-
 net/wanrouter/wanproc.c                       |    7 +-
 net/wireless/wext.c                           |    3 +-
 net/x25/x25_proc.c                            |    7 +-
 85 files changed, 534 insertions(+), 261 deletions(-)
 create mode 100644 fs/proc/proc_net.c

diff --git a/drivers/isdn/divert/divert_procfs.c b/drivers/isdn/divert/divert_procfs.c
index 559a0d0..4fd4c46 100644
--- a/drivers/isdn/divert/divert_procfs.c
+++ b/drivers/isdn/divert/divert_procfs.c
@@ -17,6 +17,7 @@
 #include <linux/fs.h>
 #endif
 #include <linux/isdnif.h>
+#include <net/net_namespace.h>
 #include "isdn_divert.h"


@@ -284,12 +285,12 @@ divert_dev_init(void)
  init_waitqueue_head(&rd_queue);

 #ifdef CONFIG_PROC_FS
- isdn_proc_entry = proc_mkdir("net/isdn", NULL);
+ isdn_proc_entry = proc_mkdir("isdn", init_net.proc_net);
  if (!isdn_proc_entry)
   return (-1);
  isdn_divert_entry = create_proc_entry("divert", S_IFREG | S_IRUGO, isdn_proc_entry);
  if (!isdn_divert_entry) {
-  remove_proc_entry("net/isdn", NULL);
+  remove_proc_entry("isdn", init_net.proc_net);
   return (-1);
  }
  isdn_divert_entry->proc_fops = &isdn_fops;
@@ -309,7 +310,7 @@ divert_dev_deinit(void)
```

```
 #ifdef CONFIG_PROC_FS
  remove_proc_entry("divert", isdn_proc_entry);
- remove_proc_entry("net/isdn", NULL);
+ remove_proc_entry("isdn", init_net.proc_net);
 #endif /* CONFIG_PROC_FS */

  return (0);
diff --git a/drivers/isdn/hardware/eicon/diva_didd.c b/drivers/isdn/hardware/eicon/diva_didd.c
index d755d90..993b14c 100644
--- a/drivers/isdn/hardware/eicon/diva_didd.c
+++ b/drivers/isdn/hardware/eicon/diva_didd.c
@@ -15,6 +15,7 @@
 #include <linux/init.h>
 #include <linux/kernel.h>
 #include <linux/proc_fs.h>
+#include <net/net_namespace.h>

 #include "platform.h"
 #include "di_defs.h"
@@ -86,7 +87,7 @@ proc_read(char *page, char **start, off_t off, int count, int *eof,

 static int DIVA_INIT_FUNCTION create_proc(void)
 {
- proc_net_eicon = proc_mkdir("net/eicon", NULL);
+ proc_net_eicon = proc_mkdir("eicon", init_net.proc_net);

  if (proc_net_eicon) {
   if ((proc_didd =
@@ -102,7 +103,7 @@ static int DIVA_INIT_FUNCTION create_proc(void)
 static void remove_proc(void)
 {
  remove_proc_entry(DRIVERLNAME, proc_net_eicon);
- remove_proc_entry("net/eicon", NULL);
+ remove_proc_entry("eicon", init_net.proc_net);
 }

 static int DIVA_INIT_FUNCTION divadidd_init(void)
diff --git a/drivers/isdn/hysdn/hysdn_procconf.c b/drivers/isdn/hysdn/hysdn_procconf.c
index dc477e0..27d890b 100644
--- a/drivers/isdn/hysdn/hysdn_procconf.c
+++ b/drivers/isdn/hysdn/hysdn_procconf.c
@@ -16,6 +16,7 @@
 #include <linux/proc_fs.h>
 #include <linux/pci.h>
 #include <linux/smp_lock.h>
+#include <net/net_namespace.h>
```

```
 #include "hysdn_defs.h"

@@ -392,7 +393,7 @@ hysdn_procconf_init(void)
  hysdn_card *card;
  unsigned char conf_name[20];

- hysdn_proc_entry = proc_mkdir(PROC_SUBDIR_NAME, proc_net);
+ hysdn_proc_entry = proc_mkdir(PROC_SUBDIR_NAME, init_net.proc_net);
  if (!hysdn_proc_entry) {
   printk(KERN_ERR "HYSDN: unable to create hysdn subdir\n");
   return (-1);
@@ -437,5 +438,5 @@ hysdn_procconf_release(void)
   card = card->next; /* point to next card */
  }

- remove_proc_entry(PROC_SUBDIR_NAME, proc_net);
+ remove_proc_entry(PROC_SUBDIR_NAME, init_net.proc_net);
 }
diff --git a/drivers/net/bonding/bond_main.c b/drivers/net/bonding/bond_main.c
index 1afda32..5de648f 100644
--- a/drivers/net/bonding/bond_main.c
+++ b/drivers/net/bonding/bond_main.c
@@ -75,6 +75,7 @@
 #include <linux/if_vlan.h>
 #include <linux/if_bonding.h>
 #include <net/route.h>
+#include <net/net_namespace.h>
 #include "bonding.h"
 #include "bond_3ad.h"
 #include "bond_alb.h"
@@ -3144,7 +3145,7 @@ static void bond_create_proc_dir(void)
 {
  int len = strlen(DRV_NAME);

- for (bond_proc_dir = proc_net->subdir; bond_proc_dir;
+ for (bond_proc_dir = init_net.proc_net->subdir; bond_proc_dir;
       bond_proc_dir = bond_proc_dir->next) {
   if ((bond_proc_dir->namelen == len) &&
       !memcmp(bond_proc_dir->name, DRV_NAME, len)) {
@@ -3153,7 +3154,7 @@ static void bond_create_proc_dir(void)
  }

  if (!bond_proc_dir) {
-  bond_proc_dir = proc_mkdir(DRV_NAME, proc_net);
+  bond_proc_dir = proc_mkdir(DRV_NAME, init_net.proc_net);
   if (bond_proc_dir) {
    bond_proc_dir->owner = THIS_MODULE;
   } else {
```

```
@@ -3188,7 +3189,7 @@ static void bond_destroy_proc_dir(void)
    bond_proc_dir->owner = NULL;
   }
  } else {
-  remove_proc_entry(DRV_NAME, proc_net);
+  remove_proc_entry(DRV_NAME, init_net.proc_net);
   bond_proc_dir = NULL;
  }
 }
diff --git a/drivers/net/hamradio/bpqether.c b/drivers/net/hamradio/bpqether.c
index cc0ee93..1699d42 100644
--- a/drivers/net/hamradio/bpqether.c
+++ b/drivers/net/hamradio/bpqether.c
@@ -83,6 +83,7 @@

 #include <net/ip.h>
 #include <net/arp.h>
+#include <net/net_namespace.h>

 #include <linux/bpqether.h>

@@ -594,7 +595,7 @@ static int bpq_device_event(struct notifier_block *this,unsigned long
event, voi
 static int __init bpq_init_driver(void)
 {
 #ifdef CONFIG_PROC_FS
- if (!proc_net_fops_create("bpqether", S_IRUGO, &bpq_info_fops)) {
+ if (!proc_net_fops_create(&init_net, "bpqether", S_IRUGO, &bpq_info_fops)) {
   printk(KERN_ERR
    "bpq: cannot create /proc/net/bpqether entry.\n");
   return -ENOENT;
@@ -618,7 +619,7 @@ static void __exit bpq_cleanup_driver(void)

  unregister_netdevice_notifier(&bpq_dev_notifier);

- proc_net_remove("bpqether");
+ proc_net_remove(&init_net, "bpqether");

  rtnl_lock();
  while (!list_empty(&bpq_devices)) {
diff --git a/drivers/net/hamradio/scc.c b/drivers/net/hamradio/scc.c
index 6fdaad5..39b3b82 100644
--- a/drivers/net/hamradio/scc.c
+++ b/drivers/net/hamradio/scc.c
@@ -174,6 +174,7 @@
 #include <linux/seq_file.h>
 #include <linux/bitops.h>
```

```
+#include <net/net_namespace.h>
 #include <net/ax25.h>

 #include <asm/irq.h>
@@ -2114,7 +2115,7 @@ static int __init scc_init_driver (void)
  }
  rtnl_unlock();

- proc_net_fops_create("z8530drv", 0, &scc_net_seq_fops);
+ proc_net_fops_create(&init_net, "z8530drv", 0, &scc_net_seq_fops);

  return 0;
 }
@@ -2169,7 +2170,7 @@ static void __exit scc_cleanup_driver(void)
  if (Vector_Latch)
   release_region(Vector_Latch, 1);

- proc_net_remove("z8530drv");
+ proc_net_remove(&init_net, "z8530drv");
 }

 MODULE_AUTHOR("Joerg Reuter <jreuter@yaina.de>");
diff --git a/drivers/net/hamradio/yam.c b/drivers/net/hamradio/yam.c
index 467559d..401724d 100644
--- a/drivers/net/hamradio/yam.c
+++ b/drivers/net/hamradio/yam.c
@@ -65,6 +65,7 @@
 #include <linux/kernel.h>
 #include <linux/proc_fs.h>
 #include <linux/seq_file.h>
+#include <net/net_namespace.h>

 #include <asm/uaccess.h>
 #include <linux/init.h>
@@ -1142,7 +1143,7 @@ static int __init yam_init_driver(void)
  yam_timer.expires = jiffies + HZ / 100;
  add_timer(&yam_timer);

- proc_net_fops_create("yam", S_IRUGO, &yam_info_fops);
+ proc_net_fops_create(&init_net, "yam", S_IRUGO, &yam_info_fops);
  return 0;
  error:
  while (--i >= 0) {
@@ -1174,7 +1175,7 @@ static void __exit yam_cleanup_driver(void)
   kfree(p);
  }

- proc_net_remove("yam");
```

```
+ proc_net_remove(&init_net, "yam");
 }

 /* ------------------------------------------------------------------- */
diff --git a/drivers/net/ibmveth.c b/drivers/net/ibmveth.c
index 996aaca..74ca22d 100644
--- a/drivers/net/ibmveth.c
+++ b/drivers/net/ibmveth.c
@@ -47,6 +47,7 @@
 #include <linux/mm.h>
 #include <linux/ethtool.h>
 #include <linux/proc_fs.h>
+#include <net/net_namespace.h>
 #include <asm/semaphore.h>
 #include <asm/hvcall.h>
 #include <asm/atomic.h>
@@ -97,7 +98,7 @@ static void ibmveth_rxq_harvest_buffer(struct ibmveth_adapter *adapter);
 static struct kobj_type ktype_veth_pool;

 #ifdef CONFIG_PROC_FS
-#define IBMVETH_PROC_DIR "net/ibmveth"
+#define IBMVETH_PROC_DIR "ibmveth"
 static struct proc_dir_entry *ibmveth_proc_dir;
 #endif

@@ -1091,7 +1092,7 @@ static int __devexit ibmveth_remove(struct vio_dev *dev)
 #ifdef CONFIG_PROC_FS
 static void ibmveth_proc_register_driver(void)
 {
- ibmveth_proc_dir = proc_mkdir(IBMVETH_PROC_DIR, NULL);
+ ibmveth_proc_dir = proc_mkdir(IBMVETH_PROC_DIR, init_net.proc_net);
   if (ibmveth_proc_dir) {
    SET_MODULE_OWNER(ibmveth_proc_dir);
   }
@@ -1099,7 +1100,7 @@ static void ibmveth_proc_register_driver(void)

 static void ibmveth_proc_unregister_driver(void)
 {
- remove_proc_entry(IBMVETH_PROC_DIR, NULL);
+ remove_proc_entry(IBMVETH_PROC_DIR, init_net.proc_net);
 }

 static void *ibmveth_seq_start(struct seq_file *seq, loff_t *pos)
diff --git a/drivers/net/pppoe.c b/drivers/net/pppoe.c
index 68631a5..a9b6971 100644
--- a/drivers/net/pppoe.c
+++ b/drivers/net/pppoe.c
@@ -78,6 +78,7 @@
```

```
 #include <linux/proc_fs.h>
 #include <linux/seq_file.h>

+#include <net/net_namespace.h>
 #include <net/sock.h>

 #include <asm/uaccess.h>
@@ -1069,7 +1070,7 @@ static int __init pppoe_proc_init(void)
 {
  struct proc_dir_entry *p;

- p = create_proc_entry("net/pppoe", S_IRUGO, NULL);
+ p = create_proc_entry("pppoe", S_IRUGO, init_net.proc_net);
  if (!p)
   return -ENOMEM;

@@ -1140,7 +1141,7 @@ static void __exit pppoe_exit(void)
  dev_remove_pack(&pppoes_ptype);
  dev_remove_pack(&pppoed_ptype);
  unregister_netdevice_notifier(&pppoe_notifier);
- remove_proc_entry("net/pppoe", NULL);
+ remove_proc_entry("pppoe", init_net.proc_net);
  proto_unregister(&pppoe_sk_proto);
 }

diff --git a/drivers/net/pppol2tp.c b/drivers/net/pppol2tp.c
index 266e8b3..c12e0a8 100644
--- a/drivers/net/pppol2tp.c
+++ b/drivers/net/pppol2tp.c
@@ -91,6 +91,7 @@
 #include <linux/hash.h>
 #include <linux/sort.h>
 #include <linux/proc_fs.h>
+#include <net/net_namespace.h>
 #include <net/dst.h>
 #include <net/ip.h>
 #include <net/udp.h>
@@ -2456,7 +2457,7 @@ static int __init pppol2tp_init(void)
   goto out_unregister_pppol2tp_proto;

 #ifdef CONFIG_PROC_FS
- pppol2tp_proc = create_proc_entry("pppol2tp", 0, proc_net);
+ pppol2tp_proc = create_proc_entry("pppol2tp", 0, init_net.proc_net);
  if (!pppol2tp_proc) {
   err = -ENOMEM;
   goto out_unregister_pppox_proto;
@@ -2481,7 +2482,7 @@ static void __exit pppol2tp_exit(void)
  unregister_pppox_proto(PX_PROTO_OL2TP);
```

```
 #ifdef CONFIG_PROC_FS
- remove_proc_entry("pppol2tp", proc_net);
+ remove_proc_entry("pppol2tp", init_net.proc_net);
 #endif
  proto_unregister(&pppol2tp_sk_proto);
 }
diff --git a/drivers/net/tokenring/lanstreamer.c b/drivers/net/tokenring/lanstreamer.c
index 5d849c0..fc44955 100644
--- a/drivers/net/tokenring/lanstreamer.c
+++ b/drivers/net/tokenring/lanstreamer.c
@@ -123,6 +123,7 @@
 #include <linux/bitops.h>
 #include <linux/jiffies.h>

+#include <net/net_namespace.h>
 #include <net/checksum.h>

 #include <asm/io.h>
@@ -250,7 +251,7 @@ static int __devinit streamer_init_one(struct pci_dev *pdev,
 #if STREAMER_NETWORK_MONITOR
 #ifdef CONFIG_PROC_FS
  if (!dev_streamer)
-  create_proc_read_entry("net/streamer_tr", 0, 0,
+  create_proc_read_entry("streamer_tr", 0, init_net.proc_net,
       streamer_proc_info, NULL);
  streamer_priv->next = dev_streamer;
  dev_streamer = streamer_priv;
@@ -423,7 +424,7 @@ static void __devexit streamer_remove_one(struct pci_dev *pdev)
    }
   }
   if (!dev_streamer)
-   remove_proc_entry("net/streamer_tr", NULL);
+   remove_proc_entry("streamer_tr", init_net.proc_net);
  }
 #endif
 #endif
diff --git a/drivers/net/tokenring/olympic.c b/drivers/net/tokenring/olympic.c
index 09b3cfb..c323101 100644
--- a/drivers/net/tokenring/olympic.c
+++ b/drivers/net/tokenring/olympic.c
@@ -102,6 +102,7 @@
 #include <linux/jiffies.h>

 #include <net/checksum.h>
+#include <net/net_namespace.h>

 #include <asm/io.h>
```

```
 #include <asm/system.h>
@@ -268,9 +269,9 @@ static int __devinit olympic_probe(struct pci_dev *pdev, const struct
pci_device
  printk("Olympic: %s registered as: %s\n",olympic_priv->olympic_card_name,dev->name);
  if (olympic_priv->olympic_network_monitor) { /* Must go after register_netdev as we need the
device name */
   char proc_name[20] ;
-  strcpy(proc_name,"net/olympic_") ;
+  strcpy(proc_name,"olympic_") ;
   strcat(proc_name,dev->name) ;
-  create_proc_read_entry(proc_name,0,NULL,olympic_proc_info,(void *)dev) ;
+  create_proc_read_entry(proc_name,0,init_net.proc_net,olympic_proc_info,(void *)dev) ;
   printk("Olympic: Network Monitor information: /proc/%s\n",proc_name);
  }
  return  0 ;
@@ -1752,9 +1753,9 @@ static void __devexit olympic_remove_one(struct pci_dev *pdev)

  if (olympic_priv->olympic_network_monitor) {
   char proc_name[20] ;
-  strcpy(proc_name,"net/olympic_") ;
+  strcpy(proc_name,"olympic_") ;
   strcat(proc_name,dev->name) ;
-  remove_proc_entry(proc_name,NULL);
+  remove_proc_entry(proc_name,init_net.proc_net);
  }
  unregister_netdev(dev) ;
  iounmap(olympic_priv->olympic_mmio) ;
diff --git a/drivers/net/wireless/hostap/hostap_main.c b/drivers/net/wireless/hostap/hostap_main.c
index 446de51..9a470e8 100644
--- a/drivers/net/wireless/hostap/hostap_main.c
+++ b/drivers/net/wireless/hostap/hostap_main.c
@@ -24,6 +24,7 @@
 #include <linux/rtnetlink.h>
 #include <linux/wireless.h>
 #include <linux/etherdevice.h>
+#include <net/net_namespace.h>
 #include <net/iw_handler.h>
 #include <net/ieee80211.h>
 #include <net/ieee80211_crypt.h>
@@ -1093,8 +1094,8 @@ struct proc_dir_entry *hostap_proc;

 static int __init hostap_init(void)
 {
- if (proc_net != NULL) {
-  hostap_proc = proc_mkdir("hostap", proc_net);
+ if (init_net.proc_net != NULL) {
+  hostap_proc = proc_mkdir("hostap", init_net.proc_net);
   if (!hostap_proc)
```

```
    printk(KERN_WARNING "Failed to mkdir "
           "/proc/net/hostap\n");
@@ -1109,7 +1110,7 @@ static void __exit hostap_exit(void)
 {
  if (hostap_proc != NULL) {
   hostap_proc = NULL;
-  remove_proc_entry("hostap", proc_net);
+  remove_proc_entry("hostap", init_net.proc_net);
  }
 }

diff --git a/drivers/net/wireless/strip.c b/drivers/net/wireless/strip.c
index ef32a5c..edb214e 100644
--- a/drivers/net/wireless/strip.c
+++ b/drivers/net/wireless/strip.c
@@ -107,6 +107,7 @@ static const char StripVersion[] = "1.3A-STUART.CHESHIRE";
 #include <linux/serialP.h>
 #include <linux/rcupdate.h>
 #include <net/arp.h>
+#include <net/net_namespace.h>

 #include <linux/ip.h>
 #include <linux/tcp.h>
@@ -2787,7 +2788,7 @@ static int __init strip_init_driver(void)
  /*
   * Register the status file with /proc
   */
- proc_net_fops_create("strip", S_IFREG | S_IRUGO, &strip_seq_fops);
+ proc_net_fops_create(&init_net, "strip", S_IFREG | S_IRUGO, &strip_seq_fops);

  return status;
 }
@@ -2809,7 +2810,7 @@ static void __exit strip_exit_driver(void)
  }

  /* Unregister with the /proc/net file here. */
- proc_net_remove("strip");
+ proc_net_remove(&init_net, "strip");

  if ((i = tty_unregister_ldisc(N_STRIP)))
   printk(KERN_ERR "STRIP: can't unregister line discipline (err = %d)\n", i);
diff --git a/fs/proc/Makefile b/fs/proc/Makefile
index bce38e3..ebaba02 100644
--- a/fs/proc/Makefile
+++ b/fs/proc/Makefile
@@ -11,6 +11,7 @@ proc-y       += inode.o root.o base.o generic.o array.o \
   proc_tty.o proc_misc.o
```

```
 proc-$(CONFIG_PROC_SYSCTL) += proc_sysctl.o
+proc-$(CONFIG_NET)  += proc_net.o
 proc-$(CONFIG_PROC_KCORE) += kcore.o
 proc-$(CONFIG_PROC_VMCORE) += vmcore.o
 proc-$(CONFIG_PROC_DEVICETREE) += proc_devtree.o
diff --git a/fs/proc/internal.h b/fs/proc/internal.h
index b215c35..1820eb2 100644
--- a/fs/proc/internal.h
+++ b/fs/proc/internal.h
@@ -16,6 +16,11 @@ extern int proc_sys_init(void);
 #else
 static inline void proc_sys_init(void) { }
 #endif
+#ifdef CONFIG_NET
+extern int proc_net_init(void);
+#else
+static inline int proc_net_init(void) { return 0; }
+#endif

 struct vmalloc_info {
   unsigned long used;
diff --git a/fs/proc/proc_net.c b/fs/proc/proc_net.c
new file mode 100644
index 0000000..45dde2f
--- /dev/null
+++ b/fs/proc/proc_net.c
@@ -0,0 +1,192 @@
+/*
+ *  linux/fs/proc/net.c
+ *
+ *  Copyright (C) 2007
+ *
+ *  Author: Eric Biederman <ebiederm@xmission.com>
+ *
+ *  proc net directory handling functions
+ */
+
+#include <asm/uaccess.h>
+
+#include <linux/errno.h>
+#include <linux/time.h>
+#include <linux/proc_fs.h>
+#include <linux/stat.h>
+#include <linux/init.h>
+#include <linux/sched.h>
+#include <linux/module.h>
+#include <linux/bitops.h>
+#include <linux/smp_lock.h>
```

```
+#include <linux/mount.h>
+#include <linux/nsproxy.h>
+#include <net/net_namespace.h>
+
+#include "internal.h"
+
+
+struct proc_dir_entry *proc_net_create(struct net *net,
+ const char *name, mode_t mode, get_info_t *get_info)
+{
+ return create_proc_info_entry(name,mode, net->proc_net, get_info);
+}
+
+struct proc_dir_entry *proc_net_fops_create(struct net *net,
+ const char *name, mode_t mode, const struct file_operations *fops)
+{
+ struct proc_dir_entry *res;
+
+ res = create_proc_entry(name, mode, net->proc_net);
+ if (res)
+  res->proc_fops = fops;
+ return res;
+}
+
+void proc_net_remove(struct net *net, const char *name)
+{
+ remove_proc_entry(name, net->proc_net);
+}
+
+
+static struct proc_dir_entry *proc_net_shadow;
+
+static struct dentry *proc_net_shadow_dentry(struct dentry *parent,
+       struct proc_dir_entry *de)
+{
+ struct dentry *shadow = NULL;
+ struct inode *inode;
+ if (!de)
+  goto out;
+ de_get(de);
+ inode = proc_get_inode(parent->d_inode->i_sb, de->low_ino, de);
+ if (!inode)
+  goto out_de_put;
+ shadow = d_alloc_name(parent, de->name);
+ if (!shadow)
+  goto out_iput;
+ shadow->d_op = parent->d_op; /* proc_dentry_operations */
+ d_instantiate(shadow, inode);
```

```
+out:
+ return shadow;
+out_iput:
+ iput(inode);
+out_de_put:
+ de_put(de);
+ goto out;
+}
+
+static void *proc_net_follow_link(struct dentry *parent, struct nameidata *nd)
+{
+ struct net *net = current->nsproxy->net_ns;
+ struct dentry *shadow;
+ shadow = proc_net_shadow_dentry(parent, net->proc_net);
+ if (!shadow)
+  return ERR_PTR(-ENOENT);
+
+ dput(nd->dentry);
+ /* My dentry count is 1 and that should be enough as the
+  * shadow dentry is thrown away immediately.
+  */
+ nd->dentry = shadow;
+ return NULL;
+}
+
+static struct dentry *proc_net_lookup(struct inode *dir, struct dentry *dentry,
+          struct nameidata *nd)
+{
+ struct net *net = current->nsproxy->net_ns;
+ struct dentry *shadow;
+
+ shadow = proc_net_shadow_dentry(nd->dentry, net->proc_net);
+ if (!shadow)
+  return ERR_PTR(-ENOENT);
+
+ dput(nd->dentry);
+ nd->dentry = shadow;
+
+ return shadow->d_inode->i_op->lookup(shadow->d_inode, dentry, nd);
+}
+
+static int proc_net_setattr(struct dentry *dentry, struct iattr *iattr)
+{
+ struct net *net = current->nsproxy->net_ns;
+ struct dentry *shadow;
+ int ret;
+
+ shadow = proc_net_shadow_dentry(dentry->d_parent, net->proc_net);
```

```
+ if (!shadow)
+  return -ENOENT;
+ ret = shadow->d_inode->i_op->setattr(shadow, iattr);
+ dput(shadow);
+ return ret;
+}
+
+static const struct file_operations proc_net_dir_operations = {
+ .read   = generic_read_dir,
+};
+
+static struct inode_operations proc_net_dir_inode_operations = {
+ .follow_link = proc_net_follow_link,
+ .lookup  = proc_net_lookup,
+ .setattr = proc_net_setattr,
+};
+
+static int proc_net_ns_init(struct net *net)
+{
+ struct proc_dir_entry *root, *netd, *net_statd;
+ int err;
+
+ err = -ENOMEM;
+ root = kzalloc(sizeof(*root), GFP_KERNEL);
+ if (!root)
+  goto out;
+
+ err = -EEXIST;
+ netd = proc_mkdir("net", root);
+ if (!netd)
+  goto free_root;
+
+ err = -EEXIST;
+ net_statd = proc_mkdir("stat", netd);
+ if (!net_statd)
+  goto free_net;
+
+ root->data = net;
+ netd->data = net;
+ net_statd->data = net;
+
+ net->proc_net_root = root;
+ net->proc_net = netd;
+ net->proc_net_stat = net_statd;
+ err = 0;
+
+out:
+ return err;
```

```
+free_net:
+ remove_proc_entry("net", root);
+free_root:
+ kfree(root);
+ goto out;
+}
+
+static void proc_net_ns_exit(struct net *net)
+{
+ remove_proc_entry("stat", net->proc_net);
+ remove_proc_entry("net", net->proc_net_root);
+ kfree(net->proc_net_root);
+}
+
+struct pernet_operations proc_net_ns_ops = {
+ .init = proc_net_ns_init,
+ .exit = proc_net_ns_exit,
+};
+
+int proc_net_init(void)
+{
+ proc_net_shadow = proc_mkdir("net", NULL);
+ proc_net_shadow->proc_iops = &proc_net_dir_inode_operations;
+ proc_net_shadow->proc_fops = &proc_net_dir_operations;
+
+ return register_pernet_subsys(&proc_net_ns_ops);
+}
diff --git a/fs/proc/root.c b/fs/proc/root.c
index 41f1703..cf30466 100644
--- a/fs/proc/root.c
+++ b/fs/proc/root.c
@@ -21,7 +21,7 @@

 #include "internal.h"

-struct proc_dir_entry *proc_net, *proc_net_stat, *proc_bus, *proc_root_fs, *proc_root_driver;
+struct proc_dir_entry *proc_bus, *proc_root_fs, *proc_root_driver;

 static int proc_get_sb(struct file_system_type *fs_type,
  int flags, const char *dev_name, void *data, struct vfsmount *mnt)
@@ -61,8 +61,8 @@ void __init proc_root_init(void)
   return;
  }
  proc_misc_init();
- proc_net = proc_mkdir("net", NULL);
- proc_net_stat = proc_mkdir("net/stat", NULL);
+
+ proc_net_init();
```

```
 #ifdef CONFIG_SYSVIPC
  proc_mkdir("sysvipc", NULL);
@@ -159,7 +159,5 @@ EXPORT_SYMBOL(create_proc_entry);
 EXPORT_SYMBOL(remove_proc_entry);
 EXPORT_SYMBOL(proc_root);
 EXPORT_SYMBOL(proc_root_fs);
-EXPORT_SYMBOL(proc_net);
-EXPORT_SYMBOL(proc_net_stat);
 EXPORT_SYMBOL(proc_bus);
 EXPORT_SYMBOL(proc_root_driver);
diff --git a/include/linux/proc_fs.h b/include/linux/proc_fs.h
index cd13a78..5964670 100644
--- a/include/linux/proc_fs.h
+++ b/include/linux/proc_fs.h
@@ -7,6 +7,7 @@
 #include <linux/magic.h>
 #include <asm/atomic.h>

+struct net;
 struct completion;

 /*
@@ -97,8 +98,6 @@ struct vmcore {

 extern struct proc_dir_entry proc_root;
 extern struct proc_dir_entry *proc_root_fs;
-extern struct proc_dir_entry *proc_net;
-extern struct proc_dir_entry *proc_net_stat;
 extern struct proc_dir_entry *proc_bus;
 extern struct proc_dir_entry *proc_root_driver;
 extern struct proc_dir_entry *proc_root_kcore;
@@ -192,36 +191,21 @@ static inline struct proc_dir_entry *create_proc_info_entry(const char
*name,
  if (res) res->get_info=get_info;
  return res;
 }
-
-static inline struct proc_dir_entry *proc_net_create(const char *name,
- mode_t mode, get_info_t *get_info)
-{
- return create_proc_info_entry(name,mode,proc_net,get_info);
-}

-static inline struct proc_dir_entry *proc_net_fops_create(const char *name,
- mode_t mode, const struct file_operations *fops)
-{
- struct proc_dir_entry *res = create_proc_entry(name, mode, proc_net);
```

```
- if (res)
-  res->proc_fops = fops;
- return res;
-}
-
-static inline void proc_net_remove(const char *name)
-{
- remove_proc_entry(name,proc_net);
-}
+extern struct proc_dir_entry *proc_net_create(struct net *net,
+ const char *name, mode_t mode, get_info_t *get_info);
+extern struct proc_dir_entry *proc_net_fops_create(struct net *net,
+ const char *name, mode_t mode, const struct file_operations *fops);
+extern void proc_net_remove(struct net *net, const char *name);

 #else

 #define proc_root_driver NULL
-#define proc_net NULL
 #define proc_bus NULL

-#define proc_net_fops_create(name, mode, fops)  ({ (void)(mode), NULL; })
-#define proc_net_create(name, mode, info) ({ (void)(mode), NULL; })
-static inline void proc_net_remove(const char *name) {}
+#define proc_net_fops_create(net, name, mode, fops)  ({ (void)(mode), NULL; })
+#define proc_net_create(net, name, mode, info) ({ (void)(mode), NULL; })
+static inline void proc_net_remove(struct net *net, const char *name) {}

 static inline void proc_flush_task(struct task_struct *task) { }

@@ -281,6 +265,16 @@ static inline struct proc_dir_entry *PDE(const struct inode *inode)
  return PROC_I(inode)->pde;
 }

+static inline struct net *PDE_NET(struct proc_dir_entry *pde)
+{
+ return pde->parent->data;
+}
+
+static inline struct net *PROC_NET(const struct inode *inode)
+{
+ return PDE_NET(PDE(inode));
+}
+
 struct proc_maps_private {
  struct pid *pid;
  struct task_struct *task;
diff --git a/include/net/net_namespace.h b/include/net/net_namespace.h
```

```
index 6344b77..5472476 100644
--- a/include/net/net_namespace.h
+++ b/include/net/net_namespace.h
@@ -8,6 +8,7 @@
 #include <linux/workqueue.h>
 #include <linux/list.h>

+struct proc_dir_entry;
 struct net {
  atomic_t  count;  /* To decided when the network
       *  namespace should be freed.
@@ -17,6 +18,10 @@ struct net {
       */
  struct list_head list;  /* list of network namespaces */
  struct work_struct work;  /* work struct for freeing */
+
+ struct proc_dir_entry  *proc_net;
+ struct proc_dir_entry  *proc_net_stat;
+ struct proc_dir_entry  *proc_net_root;
 };

 extern struct net init_net;
diff --git a/net/802/tr.c b/net/802/tr.c
index e56e61a..032c31e 100644
--- a/net/802/tr.c
+++ b/net/802/tr.c
@@ -36,6 +36,7 @@
 #include <linux/seq_file.h>
 #include <linux/init.h>
 #include <net/arp.h>
+#include <net/net_namespace.h>

 static void tr_add_rif_info(struct trh_hdr *trh, struct net_device *dev);
 static void rif_check_expire(unsigned long dummy);
@@ -639,7 +640,7 @@ static int __init rif_init(void)
  rif_timer.function = rif_check_expire;
  add_timer(&rif_timer);

- proc_net_fops_create("tr_rif", S_IRUGO, &rif_seq_fops);
+ proc_net_fops_create(&init_net, "tr_rif", S_IRUGO, &rif_seq_fops);
  return 0;
 }

diff --git a/net/8021q/vlanproc.c b/net/8021q/vlanproc.c
index bd08aa0..ac80e6b 100644
--- a/net/8021q/vlanproc.c
+++ b/net/8021q/vlanproc.c
@@ -33,6 +33,7 @@
```

```
 #include <linux/fs.h>
 #include <linux/netdevice.h>
 #include <linux/if_vlan.h>
+#include <net/net_namespace.h>
 #include "vlanproc.h"
 #include "vlan.h"

@@ -143,7 +144,7 @@ void vlan_proc_cleanup(void)
   remove_proc_entry(name_conf, proc_vlan_dir);

  if (proc_vlan_dir)
-  proc_net_remove(name_root);
+  proc_net_remove(&init_net, name_root);

  /* Dynamically added entries should be cleaned up as their vlan_device
   * is removed, so we should not have to take care of it here...
@@ -156,7 +157,7 @@ void vlan_proc_cleanup(void)

 int __init vlan_proc_init(void)
 {
- proc_vlan_dir = proc_mkdir(name_root, proc_net);
+ proc_vlan_dir = proc_mkdir(name_root, init_net.proc_net);
  if (proc_vlan_dir) {
   proc_vlan_conf = create_proc_entry(name_conf,
         S_IFREG|S_IRUSR|S_IWUSR,
diff --git a/net/appletalk/atalk_proc.c b/net/appletalk/atalk_proc.c
index 87a582c..05d9652 100644
--- a/net/appletalk/atalk_proc.c
+++ b/net/appletalk/atalk_proc.c
@@ -11,6 +11,7 @@
 #include <linux/init.h>
 #include <linux/proc_fs.h>
 #include <linux/seq_file.h>
+#include <net/net_namespace.h>
 #include <net/sock.h>
 #include <linux/atalk.h>

@@ -271,7 +272,7 @@ int __init atalk_proc_init(void)
  struct proc_dir_entry *p;
  int rc = -ENOMEM;

- atalk_proc_dir = proc_mkdir("atalk", proc_net);
+ atalk_proc_dir = proc_mkdir("atalk", init_net.proc_net);
  if (!atalk_proc_dir)
   goto out;
  atalk_proc_dir->owner = THIS_MODULE;
@@ -306,7 +307,7 @@ out_socket:
 out_route:
```

```
  remove_proc_entry("interface", atalk_proc_dir);
 out_interface:
- remove_proc_entry("atalk", proc_net);
+ remove_proc_entry("atalk", init_net.proc_net);
  goto out;
 }

@@ -316,5 +317,5 @@ void __exit atalk_proc_exit(void)
  remove_proc_entry("route", atalk_proc_dir);
  remove_proc_entry("socket", atalk_proc_dir);
  remove_proc_entry("arp", atalk_proc_dir);
- remove_proc_entry("atalk", proc_net);
+ remove_proc_entry("atalk", init_net.proc_net);
 }
diff --git a/net/atm/proc.c b/net/atm/proc.c
index 99fc1fe..3a6be64 100644
--- a/net/atm/proc.c
+++ b/net/atm/proc.c
@@ -22,6 +22,7 @@
 #include <linux/netdevice.h>
 #include <linux/atmclip.h>
 #include <linux/init.h> /* for __init */
+#include <net/net_namespace.h>
 #include <net/atmclip.h>
 #include <asm/uaccess.h>
 #include <asm/atomic.h>
@@ -475,7 +476,7 @@ static void atm_proc_dirs_remove(void)
   if (e->dirent)
    remove_proc_entry(e->name, atm_proc_root);
  }
- remove_proc_entry("net/atm", NULL);
+ remove_proc_entry("atm", init_net.proc_net);
}

int __init atm_proc_init(void)
@@ -483,7 +484,7 @@ int __init atm_proc_init(void)
  static struct atm_proc_entry *e;
  int ret;

- atm_proc_root = proc_mkdir("net/atm",NULL);
+ atm_proc_root = proc_mkdir("atm", init_net.proc_net);
  if (!atm_proc_root)
   goto err_out;
  for (e = atm_proc_ents; e->name; e++) {
diff --git a/net/ax25/af_ax25.c b/net/ax25/af_ax25.c
index dae2a42..1d71f85 100644
--- a/net/ax25/af_ax25.c
+++ b/net/ax25/af_ax25.c
```

```
@@ -44,6 +44,7 @@
 #include <linux/sysctl.h>
 #include <linux/init.h>
 #include <linux/spinlock.h>
+#include <net/net_namespace.h>
 #include <net/tcp_states.h>
 #include <net/ip.h>
 #include <net/arp.h>
@@ -1998,9 +1999,9 @@ static int __init ax25_init(void)
  register_netdevice_notifier(&ax25_dev_notifier);
  ax25_register_sysctl();

- proc_net_fops_create("ax25_route", S_IRUGO, &ax25_route_fops);
- proc_net_fops_create("ax25", S_IRUGO, &ax25_info_fops);
- proc_net_fops_create("ax25_calls", S_IRUGO, &ax25_uid_fops);
+ proc_net_fops_create(&init_net, "ax25_route", S_IRUGO, &ax25_route_fops);
+ proc_net_fops_create(&init_net, "ax25", S_IRUGO, &ax25_info_fops);
+ proc_net_fops_create(&init_net, "ax25_calls", S_IRUGO, &ax25_uid_fops);
 out:
  return rc;
 }
@@ -2014,9 +2015,9 @@ MODULE_ALIAS_NETPROTO(PF_AX25);

 static void __exit ax25_exit(void)
 {
- proc_net_remove("ax25_route");
- proc_net_remove("ax25");
- proc_net_remove("ax25_calls");
+ proc_net_remove(&init_net, "ax25_route");
+ proc_net_remove(&init_net, "ax25");
+ proc_net_remove(&init_net, "ax25_calls");
  ax25_rt_free();
  ax25_uid_free();
  ax25_dev_free();
diff --git a/net/core/dev.c b/net/core/dev.c
index 7d19a48..2ade518 100644
--- a/net/core/dev.c
+++ b/net/core/dev.c
@@ -92,6 +92,7 @@
 #include <linux/etherdevice.h>
 #include <linux/notifier.h>
 #include <linux/skbuff.h>
+#include <net/net_namespace.h>
 #include <net/sock.h>
 #include <linux/rtnetlink.h>
 #include <linux/proc_fs.h>
@@ -2538,24 +2539,24 @@ static int __init dev_proc_init(void)
 {
```

```
  int rc = -ENOMEM;

- if (!proc_net_fops_create("dev", S_IRUGO, &dev_seq_fops))
+ if (!proc_net_fops_create(&init_net, "dev", S_IRUGO, &dev_seq_fops))
    goto out;
- if (!proc_net_fops_create("softnet_stat", S_IRUGO, &softnet_seq_fops))
+ if (!proc_net_fops_create(&init_net, "softnet_stat", S_IRUGO, &softnet_seq_fops))
    goto out_dev;
- if (!proc_net_fops_create("ptype", S_IRUGO, &ptype_seq_fops))
-  goto out_dev2;
+ if (!proc_net_fops_create(&init_net, "ptype", S_IRUGO, &ptype_seq_fops))
+  goto out_softnet;

  if (wext_proc_init())
-  goto out_softnet;
+  goto out_ptype;
  rc = 0;
 out:
  return rc;
+out_ptype:
+ proc_net_remove(&init_net, "ptype");
 out_softnet:
- proc_net_remove("ptype");
-out_dev2:
- proc_net_remove("softnet_stat");
+ proc_net_remove(&init_net, "softnet_stat");
 out_dev:
- proc_net_remove("dev");
+ proc_net_remove(&init_net, "dev");
  goto out;
 }
 #else
diff --git a/net/core/dev_mcast.c b/net/core/dev_mcast.c
index 20330c5..8e069fc 100644
--- a/net/core/dev_mcast.c
+++ b/net/core/dev_mcast.c
@@ -41,6 +41,7 @@
 #include <linux/proc_fs.h>
 #include <linux/seq_file.h>
 #include <linux/init.h>
+#include <net/net_namespace.h>
 #include <net/ip.h>
 #include <net/route.h>
 #include <linux/skbuff.h>
@@ -254,7 +255,7 @@ static const struct file_operations dev_mc_seq_fops = {

 void __init dev_mcast_init(void)
 {
```

```
- proc_net_fops_create("dev_mcast", 0, &dev_mc_seq_fops);
+ proc_net_fops_create(&init_net, "dev_mcast", 0, &dev_mc_seq_fops);
 }

 EXPORT_SYMBOL(dev_mc_add);
diff --git a/net/core/neighbour.c b/net/core/neighbour.c
index ecd43c4..5f25f4f 100644
--- a/net/core/neighbour.c
+++ b/net/core/neighbour.c
@@ -25,6 +25,7 @@
 #include <linux/sysctl.h>
 #endif
 #include <linux/times.h>
+#include <net/net_namespace.h>
 #include <net/neighbour.h>
 #include <net/dst.h>
 #include <net/sock.h>
@@ -1350,7 +1351,7 @@ void neigh_table_init_no_netlink(struct neigh_table *tbl)
   panic("cannot create neighbour cache statistics");

 #ifdef CONFIG_PROC_FS
- tbl->pde = create_proc_entry(tbl->id, 0, proc_net_stat);
+ tbl->pde = create_proc_entry(tbl->id, 0, init_net.proc_net_stat);
  if (!tbl->pde)
   panic("cannot create neighbour proc dir entry");
  tbl->pde->proc_fops = &neigh_stat_seq_fops;
diff --git a/net/core/pktgen.c b/net/core/pktgen.c
index 6fa1821..fa15cc7 100644
--- a/net/core/pktgen.c
+++ b/net/core/pktgen.c
@@ -149,6 +149,7 @@
 #include <linux/wait.h>
 #include <linux/etherdevice.h>
 #include <linux/kthread.h>
+#include <net/net_namespace.h>
 #include <net/checksum.h>
 #include <net/ipv6.h>
 #include <net/addrconf.h>
@@ -3801,7 +3802,7 @@ static int __init pg_init(void)

  printk(KERN_INFO "%s", version);

- pg_proc_dir = proc_mkdir(PG_PROC_DIR, proc_net);
+ pg_proc_dir = proc_mkdir(PG_PROC_DIR, init_net.proc_net);
  if (!pg_proc_dir)
   return -ENODEV;
  pg_proc_dir->owner = THIS_MODULE;
@@ -3810,7 +3811,7 @@ static int __init pg_init(void)
```

```
  if (pe == NULL) {
   printk(KERN_ERR "pktgen: ERROR: cannot create %s "
          "procfs entry.\n", PGCTRL);
-  proc_net_remove(PG_PROC_DIR);
+  proc_net_remove(&init_net, PG_PROC_DIR);
   return -EINVAL;
  }

@@ -3834,7 +3835,7 @@ static int __init pg_init(void)
          "all threads\n");
   unregister_netdevice_notifier(&pktgen_notifier_block);
   remove_proc_entry(PGCTRL, pg_proc_dir);
-  proc_net_remove(PG_PROC_DIR);
+  proc_net_remove(&init_net, PG_PROC_DIR);
   return -ENODEV;
  }

@@ -3861,7 +3862,7 @@ static void __exit pg_cleanup(void)

  /* Clean up proc file system */
  remove_proc_entry(PGCTRL, pg_proc_dir);
- proc_net_remove(PG_PROC_DIR);
+ proc_net_remove(&init_net, PG_PROC_DIR);
 }

 module_init(pg_init);
diff --git a/net/core/sock.c b/net/core/sock.c
index cfed7d4..3d16a4b 100644
--- a/net/core/sock.c
+++ b/net/core/sock.c
@@ -119,6 +119,7 @@
 #include <linux/netdevice.h>
 #include <net/protocol.h>
 #include <linux/skbuff.h>
+#include <net/net_namespace.h>
 #include <net/request_sock.h>
 #include <net/sock.h>
 #include <net/xfrm.h>
@@ -1963,7 +1964,7 @@ static const struct file_operations proto_seq_fops = {
 static int __init proto_init(void)
 {
  /* register /proc/net/protocols */
- return proc_net_fops_create("protocols", S_IRUGO, &proto_seq_fops) == NULL ? -ENOBUFS :
0;
+ return proc_net_fops_create(&init_net, "protocols", S_IRUGO, &proto_seq_fops) == NULL ?
-ENOBUFS : 0;
 }
```

```
 subsys_initcall(proto_init);
diff --git a/net/dccp/probe.c b/net/dccp/probe.c
index bae10b0..7053bb8 100644
--- a/net/dccp/probe.c
+++ b/net/dccp/probe.c
@@ -30,6 +30,7 @@
 #include <linux/module.h>
 #include <linux/kfifo.h>
 #include <linux/vmalloc.h>
+#include <net/net_namespace.h>

 #include "dccp.h"
 #include "ccid.h"
@@ -168,7 +169,7 @@ static __init int dccpprobe_init(void)
  if (IS_ERR(dccpw.fifo))
   return PTR_ERR(dccpw.fifo);

- if (!proc_net_fops_create(procname, S_IRUSR, &dccpprobe_fops))
+ if (!proc_net_fops_create(&init_net, procname, S_IRUSR, &dccpprobe_fops))
   goto err0;

  ret = register_jprobe(&dccp_send_probe);
@@ -178,7 +179,7 @@ static __init int dccpprobe_init(void)
  pr_info("DCCP watch registered (port=%d)\n", port);
  return 0;
 err1:
- proc_net_remove(procname);
+ proc_net_remove(&init_net, procname);
 err0:
  kfifo_free(dccpw.fifo);
  return ret;
@@ -188,7 +189,7 @@ module_init(dccpprobe_init);
 static __exit void dccpprobe_exit(void)
 {
  kfifo_free(dccpw.fifo);
- proc_net_remove(procname);
+ proc_net_remove(&init_net, procname);
  unregister_jprobe(&dccp_send_probe);

 }
diff --git a/net/decnet/af_decnet.c b/net/decnet/af_decnet.c
index ed76d4a..625d595 100644
--- a/net/decnet/af_decnet.c
+++ b/net/decnet/af_decnet.c
@@ -128,6 +128,7 @@ Version 0.0.6    2.1.110   07-aug-98   Eduardo Marcelo Serrat
 #include <linux/stat.h>
 #include <linux/init.h>
 #include <linux/poll.h>
```

```
+#include <net/net_namespace.h>
 #include <net/neighbour.h>
 #include <net/dst.h>
 #include <net/fib_rules.h>
@@ -2399,7 +2400,7 @@ static int __init decnet_init(void)
  dev_add_pack(&dn_dix_packet_type);
  register_netdevice_notifier(&dn_dev_notifier);

- proc_net_fops_create("decnet", S_IRUGO, &dn_socket_seq_fops);
+ proc_net_fops_create(&init_net, "decnet", S_IRUGO, &dn_socket_seq_fops);
  dn_register_sysctl();
 out:
  return rc;
@@ -2428,7 +2429,7 @@ static void __exit decnet_exit(void)
  dn_neigh_cleanup();
  dn_fib_cleanup();

- proc_net_remove("decnet");
+ proc_net_remove(&init_net, "decnet");

  proto_unregister(&dn_proto);
 }
diff --git a/net/decnet/dn_dev.c b/net/decnet/dn_dev.c
index fa6604f..04f12e2 100644
--- a/net/decnet/dn_dev.c
+++ b/net/decnet/dn_dev.c
@@ -42,6 +42,7 @@
 #include <linux/notifier.h>
 #include <asm/uaccess.h>
 #include <asm/system.h>
+#include <net/net_namespace.h>
 #include <net/neighbour.h>
 #include <net/dst.h>
 #include <net/flow.h>
@@ -1462,7 +1463,7 @@ void __init dn_dev_init(void)
  rtnl_register(PF_DECnet, RTM_DELADDR, dn_nl_deladdr, NULL);
  rtnl_register(PF_DECnet, RTM_GETADDR, NULL, dn_nl_dump_ifaddr);

- proc_net_fops_create("decnet_dev", S_IRUGO, &dn_dev_seq_fops);
+ proc_net_fops_create(&init_net, "decnet_dev", S_IRUGO, &dn_dev_seq_fops);

 #ifdef CONFIG_SYSCTL
  {
@@ -1483,7 +1484,7 @@ void __exit dn_dev_cleanup(void)
  }
 #endif /* CONFIG_SYSCTL */

- proc_net_remove("decnet_dev");
```

```
+ proc_net_remove(&init_net, "decnet_dev");

  dn_dev_devices_off();
 }
diff --git a/net/decnet/dn_neigh.c b/net/decnet/dn_neigh.c
index 174d8a7..a424a8d 100644
--- a/net/decnet/dn_neigh.c
+++ b/net/decnet/dn_neigh.c
@@ -38,6 +38,7 @@
 #include <linux/rcupdate.h>
 #include <linux/jhash.h>
 #include <asm/atomic.h>
+#include <net/net_namespace.h>
 #include <net/neighbour.h>
 #include <net/dst.h>
 #include <net/flow.h>
@@ -611,11 +612,11 @@ static const struct file_operations dn_neigh_seq_fops = {
 void __init dn_neigh_init(void)
 {
  neigh_table_init(&dn_neigh_table);
- proc_net_fops_create("decnet_neigh", S_IRUGO, &dn_neigh_seq_fops);
+ proc_net_fops_create(&init_net, "decnet_neigh", S_IRUGO, &dn_neigh_seq_fops);
 }

 void __exit dn_neigh_cleanup(void)
 {
- proc_net_remove("decnet_neigh");
+ proc_net_remove(&init_net, "decnet_neigh");
  neigh_table_clear(&dn_neigh_table);
 }
diff --git a/net/decnet/dn_route.c b/net/decnet/dn_route.c
index a4a6209..4cfea95 100644
--- a/net/decnet/dn_route.c
+++ b/net/decnet/dn_route.c
@@ -77,6 +77,7 @@
 #include <linux/rcupdate.h>
 #include <linux/times.h>
 #include <asm/errno.h>
+#include <net/net_namespace.h>
 #include <net/netlink.h>
 #include <net/neighbour.h>
 #include <net/dst.h>
@@ -1814,7 +1815,7 @@ void __init dn_route_init(void)

  dn_dst_ops.gc_thresh = (dn_rt_hash_mask + 1);

- proc_net_fops_create("decnet_cache", S_IRUGO, &dn_rt_cache_seq_fops);
+ proc_net_fops_create(&init_net, "decnet_cache", S_IRUGO, &dn_rt_cache_seq_fops);
```

```
 #ifdef CONFIG_DECNET_ROUTER
  rtnl_register(PF_DECnet, RTM_GETROUTE, dn_cache_getroute, dn_fib_dump);
@@ -1829,6 +1830,6 @@ void __exit dn_route_cleanup(void)
  del_timer(&dn_route_timer);
  dn_run_flush(0);

- proc_net_remove("decnet_cache");
+ proc_net_remove(&init_net, "decnet_cache");
 }

diff --git a/net/ieee80211/ieee80211_module.c b/net/ieee80211/ieee80211_module.c
index 17ad278..69cb6aa 100644
--- a/net/ieee80211/ieee80211_module.c
+++ b/net/ieee80211/ieee80211_module.c
@@ -47,6 +47,7 @@
 #include <linux/wireless.h>
 #include <linux/etherdevice.h>
 #include <asm/uaccess.h>
+#include <net/net_namespace.h>
 #include <net/arp.h>

 #include <net/ieee80211.h>
@@ -264,7 +265,7 @@ static int __init ieee80211_init(void)
  struct proc_dir_entry *e;

  ieee80211_debug_level = debug;
- ieee80211_proc = proc_mkdir(DRV_NAME, proc_net);
+ ieee80211_proc = proc_mkdir(DRV_NAME, init_net.proc_net);
  if (ieee80211_proc == NULL) {
   IEEE80211_ERROR("Unable to create " DRV_NAME
     " proc directory\n");
@@ -273,7 +274,7 @@ static int __init ieee80211_init(void)
  e = create_proc_entry("debug_level", S_IFREG | S_IRUGO | S_IWUSR,
         ieee80211_proc);
  if (!e) {
-  remove_proc_entry(DRV_NAME, proc_net);
+  remove_proc_entry(DRV_NAME, init_net.proc_net);
   ieee80211_proc = NULL;
   return -EIO;
  }
@@ -293,7 +294,7 @@ static void __exit ieee80211_exit(void)
 #ifdef CONFIG_IEEE80211_DEBUG
  if (ieee80211_proc) {
   remove_proc_entry("debug_level", ieee80211_proc);
-  remove_proc_entry(DRV_NAME, proc_net);
+  remove_proc_entry(DRV_NAME, init_net.proc_net);
   ieee80211_proc = NULL;
```

```
  }
 #endif    /* CONFIG_IEEE80211_DEBUG */
diff --git a/net/ipv4/arp.c b/net/ipv4/arp.c
index 9ab9d53..78dd344 100644
--- a/net/ipv4/arp.c
+++ b/net/ipv4/arp.c
@@ -103,6 +103,7 @@
 #include <linux/sysctl.h>
 #endif

+#include <net/net_namespace.h>
 #include <net/ip.h>
 #include <net/icmp.h>
 #include <net/route.h>
@@ -1400,7 +1401,7 @@ static const struct file_operations arp_seq_fops = {

 static int __init arp_proc_init(void)
 {
- if (!proc_net_fops_create("arp", S_IRUGO, &arp_seq_fops))
+ if (!proc_net_fops_create(&init_net, "arp", S_IRUGO, &arp_seq_fops))
   return -ENOMEM;
  return 0;
 }
diff --git a/net/ipv4/fib_hash.c b/net/ipv4/fib_hash.c
index 9ad1d9f..9fafbee 100644
--- a/net/ipv4/fib_hash.c
+++ b/net/ipv4/fib_hash.c
@@ -35,6 +35,7 @@
 #include <linux/netlink.h>
 #include <linux/init.h>

+#include <net/net_namespace.h>
 #include <net/ip.h>
 #include <net/protocol.h>
 #include <net/route.h>
@@ -1068,13 +1069,13 @@ static const struct file_operations fib_seq_fops = {

 int __init fib_proc_init(void)
 {
- if (!proc_net_fops_create("route", S_IRUGO, &fib_seq_fops))
+ if (!proc_net_fops_create(&init_net, "route", S_IRUGO, &fib_seq_fops))
   return -ENOMEM;
  return 0;
 }

 void __init fib_proc_exit(void)
 {
- proc_net_remove("route");
```

```
+ proc_net_remove(&init_net, "route");
 }
 #endif /* CONFIG_PROC_FS */
diff --git a/net/ipv4/fib_trie.c b/net/ipv4/fib_trie.c
index 52b2891..be34bd5 100644
--- a/net/ipv4/fib_trie.c
+++ b/net/ipv4/fib_trie.c
@@ -73,6 +73,7 @@
 #include <linux/netlink.h>
 #include <linux/init.h>
 #include <linux/list.h>
+#include <net/net_namespace.h>
 #include <net/ip.h>
 #include <net/protocol.h>
 #include <net/route.h>
@@ -2530,30 +2531,30 @@ static const struct file_operations fib_route_fops = {

 int __init fib_proc_init(void)
 {
- if (!proc_net_fops_create("fib_trie", S_IRUGO, &fib_trie_fops))
+ if (!proc_net_fops_create(&init_net, "fib_trie", S_IRUGO, &fib_trie_fops))
   goto out1;

- if (!proc_net_fops_create("fib_triestat", S_IRUGO, &fib_triestat_fops))
+ if (!proc_net_fops_create(&init_net, "fib_triestat", S_IRUGO, &fib_triestat_fops))
   goto out2;

- if (!proc_net_fops_create("route", S_IRUGO, &fib_route_fops))
+ if (!proc_net_fops_create(&init_net, "route", S_IRUGO, &fib_route_fops))
   goto out3;

  return 0;

 out3:
- proc_net_remove("fib_triestat");
+ proc_net_remove(&init_net, "fib_triestat");
 out2:
- proc_net_remove("fib_trie");
+ proc_net_remove(&init_net, "fib_trie");
 out1:
  return -ENOMEM;
 }

 void __init fib_proc_exit(void)
 {
- proc_net_remove("fib_trie");
- proc_net_remove("fib_triestat");
- proc_net_remove("route");
```

```
+ proc_net_remove(&init_net, "fib_trie");
+ proc_net_remove(&init_net, "fib_triestat");
+ proc_net_remove(&init_net, "route");
 }

 #endif /* CONFIG_PROC_FS */
diff --git a/net/ipv4/igmp.c b/net/ipv4/igmp.c
index a646409..d78599a 100644
--- a/net/ipv4/igmp.c
+++ b/net/ipv4/igmp.c
@@ -91,6 +91,7 @@
 #include <linux/rtnetlink.h>
 #include <linux/times.h>

+#include <net/net_namespace.h>
 #include <net/arp.h>
 #include <net/ip.h>
 #include <net/protocol.h>
@@ -2613,8 +2614,8 @@ static const struct file_operations igmp_mcf_seq_fops = {

 int __init igmp_mc_proc_init(void)
 {
- proc_net_fops_create("igmp", S_IRUGO, &igmp_mc_seq_fops);
- proc_net_fops_create("mcfilter", S_IRUGO, &igmp_mcf_seq_fops);
+ proc_net_fops_create(&init_net, "igmp", S_IRUGO, &igmp_mc_seq_fops);
+ proc_net_fops_create(&init_net, "mcfilter", S_IRUGO, &igmp_mcf_seq_fops);
  return 0;
 }
 #endif
diff --git a/net/ipv4/ipconfig.c b/net/ipv4/ipconfig.c
index c5b2470..5ae4849 100644
--- a/net/ipv4/ipconfig.c
+++ b/net/ipv4/ipconfig.c
@@ -55,6 +55,7 @@
 #include <linux/root_dev.h>
 #include <linux/delay.h>
 #include <linux/nfs_fs.h>
+#include <net/net_namespace.h>
 #include <net/arp.h>
 #include <net/ip.h>
 #include <net/ipconfig.h>
@@ -1253,7 +1254,7 @@ static int __init ip_auto_config(void)
  __be32 addr;

 #ifdef CONFIG_PROC_FS
- proc_net_fops_create("pnp", S_IRUGO, &pnp_seq_fops);
+ proc_net_fops_create(&init_net, "pnp", S_IRUGO, &pnp_seq_fops);
 #endif /* CONFIG_PROC_FS */
```

```
  if (!ic_enable)
diff --git a/net/ipv4/ipmr.c b/net/ipv4/ipmr.c
index 7003cc1..35683e1 100644
--- a/net/ipv4/ipmr.c
+++ b/net/ipv4/ipmr.c
@@ -49,6 +49,7 @@
 #include <linux/mroute.h>
 #include <linux/init.h>
 #include <linux/if_ether.h>
+#include <net/net_namespace.h>
 #include <net/ip.h>
 #include <net/protocol.h>
 #include <linux/skbuff.h>
@@ -1922,7 +1923,7 @@ void __init ip_mr_init(void)
  ipmr_expire_timer.function=ipmr_expire_process;
  register_netdevice_notifier(&ip_mr_notifier);
 #ifdef CONFIG_PROC_FS
- proc_net_fops_create("ip_mr_vif", 0, &ipmr_vif_fops);
- proc_net_fops_create("ip_mr_cache", 0, &ipmr_mfc_fops);
+ proc_net_fops_create(&init_net, "ip_mr_vif", 0, &ipmr_vif_fops);
+ proc_net_fops_create(&init_net, "ip_mr_cache", 0, &ipmr_mfc_fops);
 #endif
 }
diff --git a/net/ipv4/ipvs/ip_vs_app.c b/net/ipv4/ipvs/ip_vs_app.c
index 8d6901d..341474e 100644
--- a/net/ipv4/ipvs/ip_vs_app.c
+++ b/net/ipv4/ipvs/ip_vs_app.c
@@ -25,6 +25,7 @@
 #include <linux/skbuff.h>
 #include <linux/in.h>
 #include <linux/ip.h>
+#include <net/net_namespace.h>
 #include <net/protocol.h>
 #include <net/tcp.h>
 #include <asm/system.h>
@@ -616,12 +617,12 @@ int ip_vs_skb_replace(struct sk_buff *skb, gfp_t pri,
 int ip_vs_app_init(void)
 {
  /* we will replace it with proc_net_ipvs_create() soon */
- proc_net_fops_create("ip_vs_app", 0, &ip_vs_app_fops);
+ proc_net_fops_create(&init_net, "ip_vs_app", 0, &ip_vs_app_fops);
  return 0;
 }


 void ip_vs_app_cleanup(void)
 {
```

```
- proc_net_remove("ip_vs_app");
+ proc_net_remove(&init_net, "ip_vs_app");
 }
diff --git a/net/ipv4/ipvs/ip_vs_conn.c b/net/ipv4/ipvs/ip_vs_conn.c
index d612a6a..4b702f7 100644
--- a/net/ipv4/ipvs/ip_vs_conn.c
+++ b/net/ipv4/ipvs/ip_vs_conn.c
@@ -35,6 +35,7 @@
 #include <linux/jhash.h>
 #include <linux/random.h>

+#include <net/net_namespace.h>
 #include <net/ip_vs.h>


@@ -922,7 +923,7 @@ int ip_vs_conn_init(void)
   rwlock_init(&__ip_vs_conntbl_lock_array[idx].l);
  }

- proc_net_fops_create("ip_vs_conn", 0, &ip_vs_conn_fops);
+ proc_net_fops_create(&init_net, "ip_vs_conn", 0, &ip_vs_conn_fops);

  /* calculate the random value for connection hash */
  get_random_bytes(&ip_vs_conn_rnd, sizeof(ip_vs_conn_rnd));
@@ -938,6 +939,6 @@ void ip_vs_conn_cleanup(void)

  /* Release the empty cache */
  kmem_cache_destroy(ip_vs_conn_cachep);
- proc_net_remove("ip_vs_conn");
+ proc_net_remove(&init_net, "ip_vs_conn");
  vfree(ip_vs_conn_tab);
 }
diff --git a/net/ipv4/ipvs/ip_vs_ctl.c b/net/ipv4/ipvs/ip_vs_ctl.c
index 902fd57..a5f2703 100644
--- a/net/ipv4/ipvs/ip_vs_ctl.c
+++ b/net/ipv4/ipvs/ip_vs_ctl.c
@@ -35,6 +35,7 @@
 #include <linux/netfilter_ipv4.h>
 #include <linux/mutex.h>

+#include <net/net_namespace.h>
 #include <net/ip.h>
 #include <net/route.h>
 #include <net/sock.h>
@@ -2355,8 +2356,8 @@ int ip_vs_control_init(void)
   return ret;
  }
```

```
- proc_net_fops_create("ip_vs", 0, &ip_vs_info_fops);
- proc_net_fops_create("ip_vs_stats",0, &ip_vs_stats_fops);
+ proc_net_fops_create(&init_net, "ip_vs", 0, &ip_vs_info_fops);
+ proc_net_fops_create(&init_net, "ip_vs_stats",0, &ip_vs_stats_fops);

  sysctl_header = register_sysctl_table(vs_root_table);

@@ -2389,8 +2390,8 @@ void ip_vs_control_cleanup(void)
  cancel_work_sync(&defense_work.work);
  ip_vs_kill_estimator(&ip_vs_stats);
  unregister_sysctl_table(sysctl_header);
- proc_net_remove("ip_vs_stats");
- proc_net_remove("ip_vs");
+ proc_net_remove(&init_net, "ip_vs_stats");
+ proc_net_remove(&init_net, "ip_vs");
  nf_unregister_sockopt(&ip_vs_sockopts);
  LeaveFunction(2);
 }
diff --git a/net/ipv4/ipvs/ip_vs_lblcr.c b/net/ipv4/ipvs/ip_vs_lblcr.c
index 6225aca..6a1fec4 100644
--- a/net/ipv4/ipvs/ip_vs_lblcr.c
+++ b/net/ipv4/ipvs/ip_vs_lblcr.c
@@ -50,6 +50,7 @@
 #include <linux/sysctl.h>
 /* for proc_net_create/proc_net_remove */
 #include <linux/proc_fs.h>
+#include <net/net_namespace.h>

 #include <net/ip_vs.h>

@@ -843,7 +844,7 @@ static int __init ip_vs_lblcr_init(void)
  INIT_LIST_HEAD(&ip_vs_lblcr_scheduler.n_list);
  sysctl_header = register_sysctl_table(lblcr_root_table);
 #ifdef CONFIG_IP_VS_LBLCR_DEBUG
- proc_net_create("ip_vs_lblcr", 0, ip_vs_lblcr_getinfo);
+ proc_net_create(&init_net, "ip_vs_lblcr", 0, ip_vs_lblcr_getinfo);
 #endif
  return register_ip_vs_scheduler(&ip_vs_lblcr_scheduler);
 }
@@ -852,7 +853,7 @@ static int __init ip_vs_lblcr_init(void)
 static void __exit ip_vs_lblcr_cleanup(void)
 {
 #ifdef CONFIG_IP_VS_LBLCR_DEBUG
- proc_net_remove("ip_vs_lblcr");
+ proc_net_remove(&init_net, "ip_vs_lblcr");
 #endif
  unregister_sysctl_table(sysctl_header);
  unregister_ip_vs_scheduler(&ip_vs_lblcr_scheduler);
```

```
diff --git a/net/ipv4/netfilter/ip_queue.c b/net/ipv4/netfilter/ip_queue.c
index 702d94d..cb5e61a 100644
--- a/net/ipv4/netfilter/ip_queue.c
+++ b/net/ipv4/netfilter/ip_queue.c
@@ -24,6 +24,7 @@
 #include <linux/proc_fs.h>
 #include <linux/security.h>
 #include <linux/mutex.h>
+#include <net/net_namespace.h>
 #include <net/sock.h>
 #include <net/route.h>

@@ -674,7 +675,7 @@ static int __init ip_queue_init(void)
   goto cleanup_netlink_notifier;
  }

- proc = proc_net_create(IPQ_PROC_FS_NAME, 0, ipq_get_info);
+ proc = proc_net_create(&init_net, IPQ_PROC_FS_NAME, 0, ipq_get_info);
  if (proc)
   proc->owner = THIS_MODULE;
  else {
@@ -695,8 +696,7 @@ static int __init ip_queue_init(void)
 cleanup_sysctl:
  unregister_sysctl_table(ipq_sysctl_header);
  unregister_netdevice_notifier(&ipq_dev_notifier);
- proc_net_remove(IPQ_PROC_FS_NAME);
-
+ proc_net_remove(&init_net, IPQ_PROC_FS_NAME);
 cleanup_ipqnl:
  sock_release(ipqnl->sk_socket);
  mutex_lock(&ipqnl_mutex);
@@ -715,7 +715,7 @@ static void __exit ip_queue_fini(void)

  unregister_sysctl_table(ipq_sysctl_header);
  unregister_netdevice_notifier(&ipq_dev_notifier);
- proc_net_remove(IPQ_PROC_FS_NAME);
+ proc_net_remove(&init_net, IPQ_PROC_FS_NAME);

  sock_release(ipqnl->sk_socket);
  mutex_lock(&ipqnl_mutex);
diff --git a/net/ipv4/netfilter/ipt_CLUSTERIP.c b/net/ipv4/netfilter/ipt_CLUSTERIP.c
index 69bd362..50fc9e0 100644
--- a/net/ipv4/netfilter/ipt_CLUSTERIP.c
+++ b/net/ipv4/netfilter/ipt_CLUSTERIP.c
@@ -25,6 +25,7 @@
 #include <linux/netfilter_ipv4/ip_tables.h>
 #include <linux/netfilter_ipv4/ipt_CLUSTERIP.h>
 #include <net/netfilter/nf_conntrack.h>
```

```
+#include <net/net_namespace.h>
 #include <net/checksum.h>

 #define CLUSTERIP_VERSION "0.8"
@@ -726,7 +727,7 @@ static int __init ipt_clusterip_init(void)
   goto cleanup_target;

 #ifdef CONFIG_PROC_FS
- clusterip_procdir = proc_mkdir("ipt_CLUSTERIP", proc_net);
+ clusterip_procdir = proc_mkdir("ipt_CLUSTERIP", init_net.proc_net);
  if (!clusterip_procdir) {
   printk(KERN_ERR "CLUSTERIP: Unable to proc dir entry\n");
   ret = -ENOMEM;
diff --git a/net/ipv4/netfilter/ipt_recent.c b/net/ipv4/netfilter/ipt_recent.c
index 6d0c0f7..db2a798 100644
--- a/net/ipv4/netfilter/ipt_recent.c
+++ b/net/ipv4/netfilter/ipt_recent.c
@@ -24,6 +24,7 @@
 #include <linux/bitops.h>
 #include <linux/skbuff.h>
 #include <linux/inet.h>
+#include <net/net_namespace.h>

 #include <linux/netfilter/x_tables.h>
 #include <linux/netfilter_ipv4/ipt_recent.h>
@@ -487,7 +488,7 @@ static int __init ipt_recent_init(void)
 #ifdef CONFIG_PROC_FS
  if (err)
   return err;
- proc_dir = proc_mkdir("ipt_recent", proc_net);
+ proc_dir = proc_mkdir("ipt_recent", init_net.proc_net);
  if (proc_dir == NULL) {
   xt_unregister_match(&recent_match);
   err = -ENOMEM;
@@ -501,7 +502,7 @@ static void __exit ipt_recent_exit(void)
  BUG_ON(!list_empty(&tables));
  xt_unregister_match(&recent_match);
 #ifdef CONFIG_PROC_FS
- remove_proc_entry("ipt_recent", proc_net);
+ remove_proc_entry("ipt_recent", init_net.proc_net);
 #endif
 }

diff --git a/net/ipv4/netfilter/nf_conntrack_l3proto_ipv4_compat.c
b/net/ipv4/netfilter/nf_conntrack_l3proto_ipv4_compat.c
index b3dd5de..a5ae2ea 100644
--- a/net/ipv4/netfilter/nf_conntrack_l3proto_ipv4_compat.c
+++ b/net/ipv4/netfilter/nf_conntrack_l3proto_ipv4_compat.c
```

```
@@ -11,6 +11,7 @@
 #include <linux/proc_fs.h>
 #include <linux/seq_file.h>
 #include <linux/percpu.h>
+#include <net/net_namespace.h>

 #include <linux/netfilter.h>
 #include <net/netfilter/nf_conntrack_core.h>
@@ -408,16 +409,16 @@ int __init nf_conntrack_ipv4_compat_init(void)
 {
  struct proc_dir_entry *proc, *proc_exp, *proc_stat;

- proc = proc_net_fops_create("ip_conntrack", 0440, &ct_file_ops);
+ proc = proc_net_fops_create(&init_net, "ip_conntrack", 0440, &ct_file_ops);
  if (!proc)
   goto err1;

- proc_exp = proc_net_fops_create("ip_conntrack_expect", 0440,
+ proc_exp = proc_net_fops_create(&init_net, "ip_conntrack_expect", 0440,
      &ip_exp_file_ops);
  if (!proc_exp)
   goto err2;

- proc_stat = create_proc_entry("ip_conntrack", S_IRUGO, proc_net_stat);
+ proc_stat = create_proc_entry("ip_conntrack", S_IRUGO, init_net.proc_net_stat);
  if (!proc_stat)
   goto err3;

@@ -427,16 +428,16 @@ int __init nf_conntrack_ipv4_compat_init(void)
  return 0;

 err3:
- proc_net_remove("ip_conntrack_expect");
+ proc_net_remove(&init_net, "ip_conntrack_expect");
 err2:
- proc_net_remove("ip_conntrack");
+ proc_net_remove(&init_net, "ip_conntrack");
 err1:
  return -ENOMEM;
 }

 void __exit nf_conntrack_ipv4_compat_fini(void)
 {
- remove_proc_entry("ip_conntrack", proc_net_stat);
- proc_net_remove("ip_conntrack_expect");
- proc_net_remove("ip_conntrack");
+ remove_proc_entry("ip_conntrack", init_net.proc_net_stat);
+ proc_net_remove(&init_net, "ip_conntrack_expect");
```

```
+ proc_net_remove(&init_net, "ip_conntrack");
 }
diff --git a/net/ipv4/proc.c b/net/ipv4/proc.c
index 986d1c8..95a8f8f 100644
--- a/net/ipv4/proc.c
+++ b/net/ipv4/proc.c
@@ -34,6 +34,7 @@
  *  2 of the License, or (at your option) any later version.
  */
 #include <linux/types.h>
+#include <net/net_namespace.h>
 #include <net/icmp.h>
 #include <net/protocol.h>
 #include <net/tcp.h>
@@ -383,20 +384,20 @@ int __init ip_misc_proc_init(void)
 {
  int rc = 0;

- if (!proc_net_fops_create("netstat", S_IRUGO, &netstat_seq_fops))
+ if (!proc_net_fops_create(&init_net, "netstat", S_IRUGO, &netstat_seq_fops))
   goto out_netstat;

- if (!proc_net_fops_create("snmp", S_IRUGO, &snmp_seq_fops))
+ if (!proc_net_fops_create(&init_net, "snmp", S_IRUGO, &snmp_seq_fops))
   goto out_snmp;

- if (!proc_net_fops_create("sockstat", S_IRUGO, &sockstat_seq_fops))
+ if (!proc_net_fops_create(&init_net, "sockstat", S_IRUGO, &sockstat_seq_fops))
   goto out_sockstat;
 out:
  return rc;
 out_sockstat:
- proc_net_remove("snmp");
+ proc_net_remove(&init_net, "snmp");
 out_snmp:
- proc_net_remove("netstat");
+ proc_net_remove(&init_net, "netstat");
 out_netstat:
  rc = -ENOMEM;
  goto out;
diff --git a/net/ipv4/raw.c b/net/ipv4/raw.c
index c6d7152..216e01b 100644
--- a/net/ipv4/raw.c
+++ b/net/ipv4/raw.c
@@ -59,6 +59,7 @@
 #include <linux/in_route.h>
 #include <linux/route.h>
 #include <linux/skbuff.h>
```

```
+#include <net/net_namespace.h>
 #include <net/dst.h>
 #include <net/sock.h>
 #include <linux/gfp.h>
@@ -928,13 +929,13 @@ static const struct file_operations raw_seq_fops = {

 int __init raw_proc_init(void)
 {
- if (!proc_net_fops_create("raw", S_IRUGO, &raw_seq_fops))
+ if (!proc_net_fops_create(&init_net, "raw", S_IRUGO, &raw_seq_fops))
   return -ENOMEM;
  return 0;
 }

 void __init raw_proc_exit(void)
 {
- proc_net_remove("raw");
+ proc_net_remove(&init_net, "raw");
 }
 #endif /* CONFIG_PROC_FS */
diff --git a/net/ipv4/route.c b/net/ipv4/route.c
index c7ca94b..efd2a92 100644
--- a/net/ipv4/route.c
+++ b/net/ipv4/route.c
@@ -91,6 +91,7 @@
 #include <linux/jhash.h>
 #include <linux/rcupdate.h>
 #include <linux/times.h>
+#include <net/net_namespace.h>
 #include <net/protocol.h>
 #include <net/ip.h>
 #include <net/route.h>
@@ -3011,15 +3012,15 @@ int __init ip_rt_init(void)
 #ifdef CONFIG_PROC_FS
  {
  struct proc_dir_entry *rtstat_pde = NULL; /* keep gcc happy */
- if (!proc_net_fops_create("rt_cache", S_IRUGO, &rt_cache_seq_fops) ||
+ if (!proc_net_fops_create(&init_net, "rt_cache", S_IRUGO, &rt_cache_seq_fops) ||
      !(rtstat_pde = create_proc_entry("rt_cache", S_IRUGO,
-          proc_net_stat))) {
+          init_net.proc_net_stat))) {
   return -ENOMEM;
  }
  rtstat_pde->proc_fops = &rt_cpu_seq_fops;
  }
 #ifdef CONFIG_NET_CLS_ROUTE
- create_proc_read_entry("rt_acct", 0, proc_net, ip_rt_acct_read, NULL);
+ create_proc_read_entry("rt_acct", 0, init_net.proc_net, ip_rt_acct_read, NULL);
```

```
 #endif
 #endif
 #ifdef CONFIG_XFRM
diff --git a/net/ipv4/tcp_ipv4.c b/net/ipv4/tcp_ipv4.c
index 9c94627..7fed0a6 100644
--- a/net/ipv4/tcp_ipv4.c
+++ b/net/ipv4/tcp_ipv4.c
@@ -62,6 +62,7 @@
 #include <linux/init.h>
 #include <linux/times.h>

+#include <net/net_namespace.h>
 #include <net/icmp.h>
 #include <net/inet_hashtables.h>
 #include <net/tcp.h>
@@ -2250,7 +2251,7 @@ int tcp_proc_register(struct tcp_seq_afinfo *afinfo)
  afinfo->seq_fops->llseek = seq_lseek;
  afinfo->seq_fops->release = seq_release_private;

- p = proc_net_fops_create(afinfo->name, S_IRUGO, afinfo->seq_fops);
+ p = proc_net_fops_create(&init_net, afinfo->name, S_IRUGO, afinfo->seq_fops);
  if (p)
   p->data = afinfo;
  else
@@ -2262,7 +2263,7 @@ void tcp_proc_unregister(struct tcp_seq_afinfo *afinfo)
 {
  if (!afinfo)
   return;
- proc_net_remove(afinfo->name);
+ proc_net_remove(&init_net, afinfo->name);
  memset(afinfo->seq_fops, 0, sizeof(*afinfo->seq_fops));
 }

diff --git a/net/ipv4/tcp_probe.c b/net/ipv4/tcp_probe.c
index b76398d..87dd5bf 100644
--- a/net/ipv4/tcp_probe.c
+++ b/net/ipv4/tcp_probe.c
@@ -26,6 +26,7 @@
 #include <linux/module.h>
 #include <linux/ktime.h>
 #include <linux/time.h>
+#include <net/net_namespace.h>

 #include <net/tcp.h>

@@ -228,7 +229,7 @@ static __init int tcpprobe_init(void)
  if (!tcp_probe.log)
   goto err0;
```

```
- if (!proc_net_fops_create(procname, S_IRUSR, &tcpprobe_fops))
+ if (!proc_net_fops_create(&init_net, procname, S_IRUSR, &tcpprobe_fops))
   goto err0;

  ret = register_jprobe(&tcp_jprobe);
@@ -238,7 +239,7 @@ static __init int tcpprobe_init(void)
  pr_info("TCP probe registered (port=%d)\n", port);
  return 0;
  err1:
- proc_net_remove(procname);
+ proc_net_remove(&init_net, procname);
  err0:
  kfree(tcp_probe.log);
  return ret;
@@ -247,7 +248,7 @@ module_init(tcpprobe_init);

 static __exit void tcpprobe_exit(void)
 {
- proc_net_remove(procname);
+ proc_net_remove(&init_net, procname);
  unregister_jprobe(&tcp_jprobe);
  kfree(tcp_probe.log);
 }
diff --git a/net/ipv4/udp.c b/net/ipv4/udp.c
index 4beeea8..c0c695a 100644
--- a/net/ipv4/udp.c
+++ b/net/ipv4/udp.c
@@ -98,6 +98,7 @@
 #include <linux/skbuff.h>
 #include <linux/proc_fs.h>
 #include <linux/seq_file.h>
+#include <net/net_namespace.h>
 #include <net/icmp.h>
 #include <net/route.h>
 #include <net/checksum.h>
@@ -1566,7 +1567,7 @@ int udp_proc_register(struct udp_seq_afinfo *afinfo)
  afinfo->seq_fops->llseek = seq_lseek;
  afinfo->seq_fops->release = seq_release_private;

- p = proc_net_fops_create(afinfo->name, S_IRUGO, afinfo->seq_fops);
+ p = proc_net_fops_create(&init_net, afinfo->name, S_IRUGO, afinfo->seq_fops);
  if (p)
   p->data = afinfo;
  else
@@ -1578,7 +1579,7 @@ void udp_proc_unregister(struct udp_seq_afinfo *afinfo)
 {
  if (!afinfo)
```

```
   return;
- proc_net_remove(afinfo->name);
+ proc_net_remove(&init_net, afinfo->name);
  memset(afinfo->seq_fops, 0, sizeof(*afinfo->seq_fops));
 }

diff --git a/net/ipv6/addrconf.c b/net/ipv6/addrconf.c
index 91ef3be..5e62ee5 100644
--- a/net/ipv6/addrconf.c
+++ b/net/ipv6/addrconf.c
@@ -62,6 +62,7 @@
 #include <linux/notifier.h>
 #include <linux/string.h>

+#include <net/net_namespace.h>
 #include <net/sock.h>
 #include <net/snmp.h>

@@ -2827,14 +2828,14 @@ static const struct file_operations if6_fops = {

 int __init if6_proc_init(void)
 {
- if (!proc_net_fops_create("if_inet6", S_IRUGO, &if6_fops))
+ if (!proc_net_fops_create(&init_net, "if_inet6", S_IRUGO, &if6_fops))
   return -ENOMEM;
  return 0;
 }

 void if6_proc_exit(void)
 {
- proc_net_remove("if_inet6");
+ proc_net_remove(&init_net, "if_inet6");
 }
 #endif /* CONFIG_PROC_FS */

@@ -4293,6 +4294,6 @@ void __exit addrconf_cleanup(void)
  rtnl_unlock();

 #ifdef CONFIG_PROC_FS
- proc_net_remove("if_inet6");
+ proc_net_remove(&init_net, "if_inet6");
 #endif
 }
diff --git a/net/ipv6/anycast.c b/net/ipv6/anycast.c
index b8c533f..0bd6654 100644
--- a/net/ipv6/anycast.c
+++ b/net/ipv6/anycast.c
@@ -30,6 +30,7 @@
```

```
 #include <linux/proc_fs.h>
 #include <linux/seq_file.h>

+#include <net/net_namespace.h>
 #include <net/sock.h>
 #include <net/snmp.h>

@@ -578,7 +579,7 @@ static const struct file_operations ac6_seq_fops = {

 int __init ac6_proc_init(void)
 {
- if (!proc_net_fops_create("anycast6", S_IRUGO, &ac6_seq_fops))
+ if (!proc_net_fops_create(&init_net, "anycast6", S_IRUGO, &ac6_seq_fops))
    return -ENOMEM;

  return 0;
@@ -586,7 +587,7 @@ int __init ac6_proc_init(void)

 void ac6_proc_exit(void)
 {
- proc_net_remove("anycast6");
+ proc_net_remove(&init_net, "anycast6");
 }
 #endif

diff --git a/net/ipv6/ip6_flowlabel.c b/net/ipv6/ip6_flowlabel.c
index 413a4eb..1791399 100644
--- a/net/ipv6/ip6_flowlabel.c
+++ b/net/ipv6/ip6_flowlabel.c
@@ -21,6 +21,7 @@
 #include <linux/proc_fs.h>
 #include <linux/seq_file.h>

+#include <net/net_namespace.h>
 #include <net/sock.h>

 #include <net/ipv6.h>
@@ -690,7 +691,7 @@ static const struct file_operations ip6fl_seq_fops = {
 void ip6_flowlabel_init(void)
 {
 #ifdef CONFIG_PROC_FS
- proc_net_fops_create("ip6_flowlabel", S_IRUGO, &ip6fl_seq_fops);
+ proc_net_fops_create(&init_net, "ip6_flowlabel", S_IRUGO, &ip6fl_seq_fops);
 #endif
 }

@@ -698,6 +699,6 @@ void ip6_flowlabel_cleanup(void)
 {
```

```
  del_timer(&ip6_fl_gc_timer);
 #ifdef CONFIG_PROC_FS
- proc_net_remove("ip6_flowlabel");
+ proc_net_remove(&init_net, "ip6_flowlabel");
 #endif
 }
diff --git a/net/ipv6/mcast.c b/net/ipv6/mcast.c
index ae98818..a41d5a0 100644
--- a/net/ipv6/mcast.c
+++ b/net/ipv6/mcast.c
@@ -49,6 +49,7 @@
 #include <linux/netfilter.h>
 #include <linux/netfilter_ipv6.h>

+#include <net/net_namespace.h>
 #include <net/sock.h>
 #include <net/snmp.h>

@@ -2658,8 +2659,8 @@ int __init igmp6_init(struct net_proto_family *ops)
  np->hop_limit = 1;

 #ifdef CONFIG_PROC_FS
- proc_net_fops_create("igmp6", S_IRUGO, &igmp6_mc_seq_fops);
- proc_net_fops_create("mcfilter6", S_IRUGO, &igmp6_mcf_seq_fops);
+ proc_net_fops_create(&init_net, "igmp6", S_IRUGO, &igmp6_mc_seq_fops);
+ proc_net_fops_create(&init_net, "mcfilter6", S_IRUGO, &igmp6_mcf_seq_fops);
 #endif

  return 0;
@@ -2671,7 +2672,7 @@ void igmp6_cleanup(void)
  igmp6_socket = NULL; /* for safety */

 #ifdef CONFIG_PROC_FS
- proc_net_remove("mcfilter6");
- proc_net_remove("igmp6");
+ proc_net_remove(&init_net, "mcfilter6");
+ proc_net_remove(&init_net, "igmp6");
 #endif
 }
diff --git a/net/ipv6/netfilter/ip6_queue.c b/net/ipv6/netfilter/ip6_queue.c
index 0004db3..dfc58fb 100644
--- a/net/ipv6/netfilter/ip6_queue.c
+++ b/net/ipv6/netfilter/ip6_queue.c
@@ -24,6 +24,7 @@
 #include <linux/sysctl.h>
 #include <linux/proc_fs.h>
 #include <linux/mutex.h>
+#include <net/net_namespace.h>
```

```
 #include <net/sock.h>
 #include <net/ipv6.h>
 #include <net/ip6_route.h>
@@ -664,7 +665,7 @@ static int __init ip6_queue_init(void)
   goto cleanup_netlink_notifier;
  }

- proc = proc_net_create(IPQ_PROC_FS_NAME, 0, ipq_get_info);
+ proc = proc_net_create(&init_net, IPQ_PROC_FS_NAME, 0, ipq_get_info);
  if (proc)
   proc->owner = THIS_MODULE;
  else {
@@ -685,7 +686,7 @@ static int __init ip6_queue_init(void)
 cleanup_sysctl:
  unregister_sysctl_table(ipq_sysctl_header);
  unregister_netdevice_notifier(&ipq_dev_notifier);
- proc_net_remove(IPQ_PROC_FS_NAME);
+ proc_net_remove(&init_net, IPQ_PROC_FS_NAME);

 cleanup_ipqnl:
  sock_release(ipqnl->sk_socket);
@@ -705,7 +706,7 @@ static void __exit ip6_queue_fini(void)

  unregister_sysctl_table(ipq_sysctl_header);
  unregister_netdevice_notifier(&ipq_dev_notifier);
- proc_net_remove(IPQ_PROC_FS_NAME);
+ proc_net_remove(&init_net, IPQ_PROC_FS_NAME);

  sock_release(ipqnl->sk_socket);
  mutex_lock(&ipqnl_mutex);
diff --git a/net/ipv6/proc.c b/net/ipv6/proc.c
index 920dc9c..a712a22 100644
--- a/net/ipv6/proc.c
+++ b/net/ipv6/proc.c
@@ -23,6 +23,7 @@
 #include <linux/proc_fs.h>
 #include <linux/seq_file.h>
 #include <linux/stddef.h>
+#include <net/net_namespace.h>
 #include <net/ip.h>
 #include <net/sock.h>
 #include <net/tcp.h>
@@ -231,22 +232,22 @@ int __init ipv6_misc_proc_init(void)
 {
  int rc = 0;

- if (!proc_net_fops_create("snmp6", S_IRUGO, &snmp6_seq_fops))
+ if (!proc_net_fops_create(&init_net, "snmp6", S_IRUGO, &snmp6_seq_fops))
```

```
   goto proc_snmp6_fail;

- proc_net_devsnmp6 = proc_mkdir("dev_snmp6", proc_net);
+ proc_net_devsnmp6 = proc_mkdir("dev_snmp6", init_net.proc_net);
  if (!proc_net_devsnmp6)
   goto proc_dev_snmp6_fail;

- if (!proc_net_fops_create("sockstat6", S_IRUGO, &sockstat6_seq_fops))
+ if (!proc_net_fops_create(&init_net, "sockstat6", S_IRUGO, &sockstat6_seq_fops))
   goto proc_sockstat6_fail;
 out:
  return rc;

 proc_sockstat6_fail:
- proc_net_remove("dev_snmp6");
+ proc_net_remove(&init_net, "dev_snmp6");
 proc_dev_snmp6_fail:
- proc_net_remove("snmp6");
+ proc_net_remove(&init_net, "snmp6");
 proc_snmp6_fail:
  rc = -ENOMEM;
  goto out;
@@ -254,8 +255,8 @@ proc_snmp6_fail:

 void ipv6_misc_proc_exit(void)
 {
- proc_net_remove("sockstat6");
- proc_net_remove("dev_snmp6");
- proc_net_remove("snmp6");
+ proc_net_remove(&init_net, "sockstat6");
+ proc_net_remove(&init_net, "dev_snmp6");
+ proc_net_remove(&init_net, "snmp6");
 }

diff --git a/net/ipv6/raw.c b/net/ipv6/raw.c
index e27383d..20c1a76 100644
--- a/net/ipv6/raw.c
+++ b/net/ipv6/raw.c
@@ -35,6 +35,7 @@
 #include <asm/uaccess.h>
 #include <asm/ioctls.h>

+#include <net/net_namespace.h>
 #include <net/ip.h>
 #include <net/sock.h>
 #include <net/snmp.h>
@@ -1316,13 +1317,13 @@ static const struct file_operations raw6_seq_fops = {
```

```
 int __init raw6_proc_init(void)
 {
- if (!proc_net_fops_create("raw6", S_IRUGO, &raw6_seq_fops))
+ if (!proc_net_fops_create(&init_net, "raw6", S_IRUGO, &raw6_seq_fops))
    return -ENOMEM;
   return 0;
 }

 void raw6_proc_exit(void)
 {
- proc_net_remove("raw6");
+ proc_net_remove(&init_net, "raw6");
 }
 #endif /* CONFIG_PROC_FS */
diff --git a/net/ipv6/route.c b/net/ipv6/route.c
index 55ea80f..f4f0c34 100644
--- a/net/ipv6/route.c
+++ b/net/ipv6/route.c
@@ -44,6 +44,7 @@
 #include <linux/seq_file.h>
 #endif

+#include <net/net_namespace.h>
 #include <net/snmp.h>
 #include <net/ipv6.h>
 #include <net/ip6_fib.h>
@@ -2561,11 +2562,11 @@ void __init ip6_route_init(void)

   fib6_init();
 #ifdef  CONFIG_PROC_FS
- p = proc_net_create("ipv6_route", 0, rt6_proc_info);
+ p = proc_net_create(&init_net, "ipv6_route", 0, rt6_proc_info);
   if (p)
     p->owner = THIS_MODULE;

- proc_net_fops_create("rt6_stats", S_IRUGO, &rt6_stats_seq_fops);
+ proc_net_fops_create(&init_net, "rt6_stats", S_IRUGO, &rt6_stats_seq_fops);
 #endif
 #ifdef CONFIG_XFRM
   xfrm6_init();
@@ -2585,8 +2586,8 @@ void ip6_route_cleanup(void)
   fib6_rules_cleanup();
 #endif
 #ifdef CONFIG_PROC_FS
- proc_net_remove("ipv6_route");
- proc_net_remove("rt6_stats");
+ proc_net_remove(&init_net, "ipv6_route");
+ proc_net_remove(&init_net, "rt6_stats");
```

```
 #endif
 #ifdef CONFIG_XFRM
  xfrm6_fini();
diff --git a/net/ipx/ipx_proc.c b/net/ipx/ipx_proc.c
index 4226e71..d483a00 100644
--- a/net/ipx/ipx_proc.c
+++ b/net/ipx/ipx_proc.c
@@ -9,6 +9,7 @@
 #include <linux/proc_fs.h>
 #include <linux/spinlock.h>
 #include <linux/seq_file.h>
+#include <net/net_namespace.h>
 #include <net/tcp_states.h>
 #include <net/ipx.h>

@@ -353,7 +354,7 @@ int __init ipx_proc_init(void)
  struct proc_dir_entry *p;
  int rc = -ENOMEM;

- ipx_proc_dir = proc_mkdir("ipx", proc_net);
+ ipx_proc_dir = proc_mkdir("ipx", init_net.proc_net);

  if (!ipx_proc_dir)
   goto out;
@@ -381,7 +382,7 @@ out_socket:
 out_route:
  remove_proc_entry("interface", ipx_proc_dir);
 out_interface:
- remove_proc_entry("ipx", proc_net);
+ remove_proc_entry("ipx", init_net.proc_net);
  goto out;
 }

@@ -390,7 +391,7 @@ void __exit ipx_proc_exit(void)
  remove_proc_entry("interface", ipx_proc_dir);
  remove_proc_entry("route", ipx_proc_dir);
  remove_proc_entry("socket", ipx_proc_dir);
- remove_proc_entry("ipx", proc_net);
+ remove_proc_entry("ipx", init_net.proc_net);
 }

 #else /* CONFIG_PROC_FS */
diff --git a/net/irda/irproc.c b/net/irda/irproc.c
index 181cb51..cae24fb 100644
--- a/net/irda/irproc.c
+++ b/net/irda/irproc.c
@@ -28,6 +28,7 @@
 #include <linux/seq_file.h>
```

```
 #include <linux/module.h>
 #include <linux/init.h>
+#include <net/net_namespace.h>

 #include <net/irda/irda.h>
 #include <net/irda/irlap.h>
@@ -66,7 +67,7 @@ void __init irda_proc_register(void)
  int i;
  struct proc_dir_entry *d;

- proc_irda = proc_mkdir("irda", proc_net);
+ proc_irda = proc_mkdir("irda", init_net.proc_net);
  if (proc_irda == NULL)
   return;
  proc_irda->owner = THIS_MODULE;
@@ -92,7 +93,7 @@ void irda_proc_unregister(void)
   for (i=0; i<ARRAY_SIZE(irda_dirs); i++)
    remove_proc_entry(irda_dirs[i].name, proc_irda);

-  remove_proc_entry("irda", proc_net);
+  remove_proc_entry("irda", init_net.proc_net);
   proc_irda = NULL;
  }
 }
diff --git a/net/key/af_key.c b/net/key/af_key.c
index 17b2a69..c1a9583 100644
--- a/net/key/af_key.c
+++ b/net/key/af_key.c
@@ -26,6 +26,7 @@
 #include <linux/in6.h>
 #include <linux/proc_fs.h>
 #include <linux/init.h>
+#include <net/net_namespace.h>
 #include <net/xfrm.h>
 #include <linux/audit.h>

@@ -3777,7 +3778,7 @@ static struct xfrm_mgr pfkeyv2_mgr =
 static void __exit ipsec_pfkey_exit(void)
 {
  xfrm_unregister_km(&pfkeyv2_mgr);
- remove_proc_entry("net/pfkey", NULL);
+ remove_proc_entry("pfkey", init_net.proc_net);
  sock_unregister(PF_KEY);
  proto_unregister(&key_proto);
 }
@@ -3794,7 +3795,7 @@ static int __init ipsec_pfkey_init(void)
   goto out_unregister_key_proto;
 #ifdef CONFIG_PROC_FS
```

```
  err = -ENOMEM;
- if (create_proc_read_entry("net/pfkey", 0, NULL, pfkey_read_proc, NULL) == NULL)
+ if (create_proc_read_entry("pfkey", 0, init_net.proc_net, pfkey_read_proc, NULL) == NULL)
    goto out_sock_unregister;
 #endif
  err = xfrm_register_km(&pfkeyv2_mgr);
diff --git a/net/llc/llc_proc.c b/net/llc/llc_proc.c
index 49be6c9..4865d82 100644
--- a/net/llc/llc_proc.c
+++ b/net/llc/llc_proc.c
@@ -17,6 +17,7 @@
 #include <linux/proc_fs.h>
 #include <linux/errno.h>
 #include <linux/seq_file.h>
+#include <net/net_namespace.h>
 #include <net/sock.h>
 #include <net/llc.h>
 #include <net/llc_c_ac.h>
@@ -231,7 +232,7 @@ int __init llc_proc_init(void)
  int rc = -ENOMEM;
  struct proc_dir_entry *p;

- llc_proc_dir = proc_mkdir("llc", proc_net);
+ llc_proc_dir = proc_mkdir("llc", init_net.proc_net);
  if (!llc_proc_dir)
    goto out;
  llc_proc_dir->owner = THIS_MODULE;
@@ -254,7 +255,7 @@ out:
 out_core:
  remove_proc_entry("socket", llc_proc_dir);
 out_socket:
- remove_proc_entry("llc", proc_net);
+ remove_proc_entry("llc", init_net.proc_net);
  goto out;
 }

@@ -262,5 +263,5 @@ void llc_proc_exit(void)
 {
  remove_proc_entry("socket", llc_proc_dir);
  remove_proc_entry("core", llc_proc_dir);
- remove_proc_entry("llc", proc_net);
+ remove_proc_entry("llc", init_net.proc_net);
 }
diff --git a/net/netfilter/core.c b/net/netfilter/core.c
index 381a77c..a523fa4 100644
--- a/net/netfilter/core.c
+++ b/net/netfilter/core.c
@@ -19,6 +19,7 @@
```

```
 #include <linux/inetdevice.h>
 #include <linux/proc_fs.h>
 #include <linux/mutex.h>
+#include <net/net_namespace.h>
 #include <net/sock.h>

 #include "nf_internals.h"
@@ -293,7 +294,7 @@ void __init netfilter_init(void)
  }

 #ifdef CONFIG_PROC_FS
- proc_net_netfilter = proc_mkdir("netfilter", proc_net);
+ proc_net_netfilter = proc_mkdir("netfilter", init_net.proc_net);
  if (!proc_net_netfilter)
   panic("cannot create netfilter proc entry");
 #endif
diff --git a/net/netfilter/nf_conntrack_expect.c b/net/netfilter/nf_conntrack_expect.c
index 3ac64e2..8a3e3af 100644
--- a/net/netfilter/nf_conntrack_expect.c
+++ b/net/netfilter/nf_conntrack_expect.c
@@ -20,6 +20,7 @@
 #include <linux/percpu.h>
 #include <linux/kernel.h>
 #include <linux/jhash.h>
+#include <net/net_namespace.h>

 #include <net/netfilter/nf_conntrack.h>
 #include <net/netfilter/nf_conntrack_core.h>
@@ -505,7 +506,7 @@ static int __init exp_proc_init(void)
 #ifdef CONFIG_PROC_FS
  struct proc_dir_entry *proc;

- proc = proc_net_fops_create("nf_conntrack_expect", 0440, &exp_file_ops);
+ proc = proc_net_fops_create(&init_net, "nf_conntrack_expect", 0440, &exp_file_ops);
  if (!proc)
   return -ENOMEM;
 #endif /* CONFIG_PROC_FS */
@@ -515,7 +516,7 @@ static int __init exp_proc_init(void)
 static void exp_proc_remove(void)
 {
 #ifdef CONFIG_PROC_FS
- proc_net_remove("nf_conntrack_expect");
+ proc_net_remove(&init_net, "nf_conntrack_expect");
 #endif /* CONFIG_PROC_FS */
 }

diff --git a/net/netfilter/nf_conntrack_standalone.c b/net/netfilter/nf_conntrack_standalone.c
index a4ce5e8..2a19c5f 100644
```

```
--- a/net/netfilter/nf_conntrack_standalone.c
+++ b/net/netfilter/nf_conntrack_standalone.c
@@ -14,6 +14,7 @@
 #include <linux/seq_file.h>
 #include <linux/percpu.h>
 #include <linux/netdevice.h>
+#include <net/net_namespace.h>
 #ifdef CONFIG_SYSCTL
 #include <linux/sysctl.h>
 #endif
@@ -420,10 +421,10 @@ static int __init nf_conntrack_standalone_init(void)
   return ret;

 #ifdef CONFIG_PROC_FS
- proc = proc_net_fops_create("nf_conntrack", 0440, &ct_file_ops);
+ proc = proc_net_fops_create(&init_net, "nf_conntrack", 0440, &ct_file_ops);
  if (!proc) goto cleanup_init;

- proc_stat = create_proc_entry("nf_conntrack", S_IRUGO, proc_net_stat);
+ proc_stat = create_proc_entry("nf_conntrack", S_IRUGO, init_net.proc_net_stat);
  if (!proc_stat)
   goto cleanup_proc;

@@ -444,9 +445,9 @@ static int __init nf_conntrack_standalone_init(void)
  cleanup_proc_stat:
 #endif
 #ifdef CONFIG_PROC_FS
- remove_proc_entry("nf_conntrack", proc_net_stat);
+ remove_proc_entry("nf_conntrack", init_net. proc_net_stat);
  cleanup_proc:
- proc_net_remove("nf_conntrack");
+ proc_net_remove(&init_net, "nf_conntrack");
  cleanup_init:
 #endif /* CNFIG_PROC_FS */
  nf_conntrack_cleanup();
@@ -459,8 +460,8 @@ static void __exit nf_conntrack_standalone_fini(void)
  unregister_sysctl_table(nf_ct_sysctl_header);
 #endif
 #ifdef CONFIG_PROC_FS
- remove_proc_entry("nf_conntrack", proc_net_stat);
- proc_net_remove("nf_conntrack");
+ remove_proc_entry("nf_conntrack", init_net.proc_net_stat);
+ proc_net_remove(&init_net, "nf_conntrack");
 #endif /* CNFIG_PROC_FS */
  nf_conntrack_cleanup();
 }
diff --git a/net/netfilter/x_tables.c b/net/netfilter/x_tables.c
index cc2baa6..d9a3bde 100644
```

```
--- a/net/netfilter/x_tables.c
+++ b/net/netfilter/x_tables.c
@@ -22,6 +22,7 @@
 #include <linux/vmalloc.h>
 #include <linux/mutex.h>
 #include <linux/mm.h>
+#include <net/net_namespace.h>

 #include <linux/netfilter/x_tables.h>
 #include <linux/netfilter_arp.h>
@@ -795,7 +796,7 @@ int xt_proto_init(int af)
 #ifdef CONFIG_PROC_FS
  strlcpy(buf, xt_prefix[af], sizeof(buf));
  strlcat(buf, FORMAT_TABLES, sizeof(buf));
- proc = proc_net_fops_create(buf, 0440, &xt_file_ops);
+ proc = proc_net_fops_create(&init_net, buf, 0440, &xt_file_ops);
  if (!proc)
   goto out;
  proc->data = (void *) ((unsigned long) af | (TABLE << 16));
@@ -803,14 +804,14 @@ int xt_proto_init(int af)

  strlcpy(buf, xt_prefix[af], sizeof(buf));
  strlcat(buf, FORMAT_MATCHES, sizeof(buf));
- proc = proc_net_fops_create(buf, 0440, &xt_file_ops);
+ proc = proc_net_fops_create(&init_net, buf, 0440, &xt_file_ops);
  if (!proc)
   goto out_remove_tables;
  proc->data = (void *) ((unsigned long) af | (MATCH << 16));

  strlcpy(buf, xt_prefix[af], sizeof(buf));
  strlcat(buf, FORMAT_TARGETS, sizeof(buf));
- proc = proc_net_fops_create(buf, 0440, &xt_file_ops);
+ proc = proc_net_fops_create(&init_net, buf, 0440, &xt_file_ops);
  if (!proc)
   goto out_remove_matches;
  proc->data = (void *) ((unsigned long) af | (TARGET << 16));
@@ -822,12 +823,12 @@ int xt_proto_init(int af)
 out_remove_matches:
  strlcpy(buf, xt_prefix[af], sizeof(buf));
  strlcat(buf, FORMAT_MATCHES, sizeof(buf));
- proc_net_remove(buf);
+ proc_net_remove(&init_net, buf);

 out_remove_tables:
  strlcpy(buf, xt_prefix[af], sizeof(buf));
  strlcat(buf, FORMAT_TABLES, sizeof(buf));
- proc_net_remove(buf);
+ proc_net_remove(&init_net, buf);
```

```
 out:
  return -1;
 #endif
@@ -841,15 +842,15 @@ void xt_proto_fini(int af)

  strlcpy(buf, xt_prefix[af], sizeof(buf));
  strlcat(buf, FORMAT_TABLES, sizeof(buf));
- proc_net_remove(buf);
+ proc_net_remove(&init_net, buf);

  strlcpy(buf, xt_prefix[af], sizeof(buf));
  strlcat(buf, FORMAT_TARGETS, sizeof(buf));
- proc_net_remove(buf);
+ proc_net_remove(&init_net, buf);

  strlcpy(buf, xt_prefix[af], sizeof(buf));
  strlcat(buf, FORMAT_MATCHES, sizeof(buf));
- proc_net_remove(buf);
+ proc_net_remove(&init_net, buf);
 #endif /*CONFIG_PROC_FS*/
 }
 EXPORT_SYMBOL_GPL(xt_proto_fini);
diff --git a/net/netfilter/xt_hashlimit.c b/net/netfilter/xt_hashlimit.c
index bd45f9d..1910367 100644
--- a/net/netfilter/xt_hashlimit.c
+++ b/net/netfilter/xt_hashlimit.c
@@ -21,6 +21,7 @@
 #include <linux/in.h>
 #include <linux/ip.h>
 #include <linux/ipv6.h>
+#include <net/net_namespace.h>

 #include <linux/netfilter/x_tables.h>
 #include <linux/netfilter_ipv4/ip_tables.h>
@@ -743,13 +744,13 @@ static int __init xt_hashlimit_init(void)
   printk(KERN_ERR "xt_hashlimit: unable to create slab cache\n");
   goto err2;
  }
- hashlimit_procdir4 = proc_mkdir("ipt_hashlimit", proc_net);
+ hashlimit_procdir4 = proc_mkdir("ipt_hashlimit", init_net.proc_net);
  if (!hashlimit_procdir4) {
   printk(KERN_ERR "xt_hashlimit: unable to create proc dir "
     "entry\n");
   goto err3;
  }
- hashlimit_procdir6 = proc_mkdir("ip6t_hashlimit", proc_net);
+ hashlimit_procdir6 = proc_mkdir("ip6t_hashlimit", init_net.proc_net);
  if (!hashlimit_procdir6) {
```

```
   printk(KERN_ERR "xt_hashlimit: unable to create proc dir "
     "entry\n");
@@ -757,7 +758,7 @@ static int __init xt_hashlimit_init(void)
  }
  return 0;
 err4:
- remove_proc_entry("ipt_hashlimit", proc_net);
+ remove_proc_entry("ipt_hashlimit", init_net.proc_net);
 err3:
  kmem_cache_destroy(hashlimit_cachep);
 err2:
@@ -769,8 +770,8 @@ err1:

 static void __exit xt_hashlimit_fini(void)
 {
- remove_proc_entry("ipt_hashlimit", proc_net);
- remove_proc_entry("ip6t_hashlimit", proc_net);
+ remove_proc_entry("ipt_hashlimit", init_net.proc_net);
+ remove_proc_entry("ip6t_hashlimit", init_net.proc_net);
  kmem_cache_destroy(hashlimit_cachep);
  xt_unregister_matches(xt_hashlimit, ARRAY_SIZE(xt_hashlimit));
 }
diff --git a/net/netlink/af_netlink.c b/net/netlink/af_netlink.c
index a78d962..3982f13 100644
--- a/net/netlink/af_netlink.c
+++ b/net/netlink/af_netlink.c
@@ -57,6 +57,7 @@
 #include <linux/selinux.h>
 #include <linux/mutex.h>

+#include <net/net_namespace.h>
 #include <net/sock.h>
 #include <net/scm.h>
 #include <net/netlink.h>
@@ -1927,7 +1928,7 @@ static int __init netlink_proto_init(void)

  sock_register(&netlink_family_ops);
 #ifdef CONFIG_PROC_FS
- proc_net_fops_create("netlink", 0, &netlink_seq_fops);
+ proc_net_fops_create(&init_net, "netlink", 0, &netlink_seq_fops);
 #endif
  /* The netlink device handler may be needed early. */
  rtnetlink_init();
diff --git a/net/netrom/af_netrom.c b/net/netrom/af_netrom.c
index dc92732..15c8a92 100644
--- a/net/netrom/af_netrom.c
+++ b/net/netrom/af_netrom.c
@@ -27,6 +27,7 @@
```

```
 #include <linux/netdevice.h>
 #include <linux/if_arp.h>
 #include <linux/skbuff.h>
+#include <net/net_namespace.h>
 #include <net/sock.h>
 #include <asm/uaccess.h>
 #include <asm/system.h>
@@ -1447,9 +1448,9 @@ static int __init nr_proto_init(void)

  nr_loopback_init();

- proc_net_fops_create("nr", S_IRUGO, &nr_info_fops);
- proc_net_fops_create("nr_neigh", S_IRUGO, &nr_neigh_fops);
- proc_net_fops_create("nr_nodes", S_IRUGO, &nr_nodes_fops);
+ proc_net_fops_create(&init_net, "nr", S_IRUGO, &nr_info_fops);
+ proc_net_fops_create(&init_net, "nr_neigh", S_IRUGO, &nr_neigh_fops);
+ proc_net_fops_create(&init_net, "nr_nodes", S_IRUGO, &nr_nodes_fops);
 out:
  return rc;
 fail:
@@ -1477,9 +1478,9 @@ static void __exit nr_exit(void)
 {
  int i;

- proc_net_remove("nr");
- proc_net_remove("nr_neigh");
- proc_net_remove("nr_nodes");
+ proc_net_remove(&init_net, "nr");
+ proc_net_remove(&init_net, "nr_neigh");
+ proc_net_remove(&init_net, "nr_nodes");
  nr_loopback_clear();

  nr_rt_free();
diff --git a/net/packet/af_packet.c b/net/packet/af_packet.c
index 1322d62..1eecbd7 100644
--- a/net/packet/af_packet.c
+++ b/net/packet/af_packet.c
@@ -61,6 +61,7 @@
 #include <linux/wireless.h>
 #include <linux/kernel.h>
 #include <linux/kmod.h>
+#include <net/net_namespace.h>
 #include <net/ip.h>
 #include <net/protocol.h>
 #include <linux/skbuff.h>
@@ -1952,7 +1953,7 @@ static const struct file_operations packet_seq_fops = {

 static void __exit packet_exit(void)
```

```
 {
- proc_net_remove("packet");
+ proc_net_remove(&init_net, "packet");
  unregister_netdevice_notifier(&packet_netdev_notifier);
  sock_unregister(PF_PACKET);
  proto_unregister(&packet_proto);
@@ -1967,7 +1968,7 @@ static int __init packet_init(void)

  sock_register(&packet_family_ops);
  register_netdevice_notifier(&packet_netdev_notifier);
- proc_net_fops_create("packet", 0, &packet_seq_fops);
+ proc_net_fops_create(&init_net, "packet", 0, &packet_seq_fops);
 out:
  return rc;
 }
diff --git a/net/rose/af_rose.c b/net/rose/af_rose.c
index 976c3cc..48319f7 100644
--- a/net/rose/af_rose.c
+++ b/net/rose/af_rose.c
@@ -26,6 +26,7 @@
 #include <linux/sockios.h>
 #include <linux/net.h>
 #include <linux/stat.h>
+#include <net/net_namespace.h>
 #include <net/ax25.h>
 #include <linux/inet.h>
 #include <linux/netdevice.h>
@@ -1576,10 +1577,10 @@ static int __init rose_proto_init(void)

  rose_add_loopback_neigh();

- proc_net_fops_create("rose", S_IRUGO, &rose_info_fops);
- proc_net_fops_create("rose_neigh", S_IRUGO, &rose_neigh_fops);
- proc_net_fops_create("rose_nodes", S_IRUGO, &rose_nodes_fops);
- proc_net_fops_create("rose_routes", S_IRUGO, &rose_routes_fops);
+ proc_net_fops_create(&init_net, "rose", S_IRUGO, &rose_info_fops);
+ proc_net_fops_create(&init_net, "rose_neigh", S_IRUGO, &rose_neigh_fops);
+ proc_net_fops_create(&init_net, "rose_nodes", S_IRUGO, &rose_nodes_fops);
+ proc_net_fops_create(&init_net, "rose_routes", S_IRUGO, &rose_routes_fops);
 out:
  return rc;
 fail:
@@ -1606,10 +1607,10 @@ static void __exit rose_exit(void)
 {
  int i;

- proc_net_remove("rose");
- proc_net_remove("rose_neigh");
```

```
- proc_net_remove("rose_nodes");
- proc_net_remove("rose_routes");
+ proc_net_remove(&init_net, "rose");
+ proc_net_remove(&init_net, "rose_neigh");
+ proc_net_remove(&init_net, "rose_nodes");
+ proc_net_remove(&init_net, "rose_routes");
  rose_loopback_clear();

  rose_rt_free();
diff --git a/net/rxrpc/af_rxrpc.c b/net/rxrpc/af_rxrpc.c
index c58fa0d..122d55d 100644
--- a/net/rxrpc/af_rxrpc.c
+++ b/net/rxrpc/af_rxrpc.c
@@ -14,6 +14,7 @@
 #include <linux/skbuff.h>
 #include <linux/poll.h>
 #include <linux/proc_fs.h>
+#include <net/net_namespace.h>
 #include <net/sock.h>
 #include <net/af_rxrpc.h>
 #include "ar-internal.h"
@@ -829,8 +830,8 @@ static int __init af_rxrpc_init(void)
  }

 #ifdef CONFIG_PROC_FS
- proc_net_fops_create("rxrpc_calls", 0, &rxrpc_call_seq_fops);
- proc_net_fops_create("rxrpc_conns", 0, &rxrpc_connection_seq_fops);
+ proc_net_fops_create(&init_net, "rxrpc_calls", 0, &rxrpc_call_seq_fops);
+ proc_net_fops_create(&init_net, "rxrpc_conns", 0, &rxrpc_connection_seq_fops);
 #endif
  return 0;

@@ -868,8 +869,8 @@ static void __exit af_rxrpc_exit(void)

  _debug("flush scheduled work");
  flush_workqueue(rxrpc_workqueue);
- proc_net_remove("rxrpc_conns");
- proc_net_remove("rxrpc_calls");
+ proc_net_remove(&init_net, "rxrpc_conns");
+ proc_net_remove(&init_net, "rxrpc_calls");
  destroy_workqueue(rxrpc_workqueue);
  kmem_cache_destroy(rxrpc_call_jar);
  _leave("");
diff --git a/net/sched/sch_api.c b/net/sched/sch_api.c
index dee0d5f..efc383c 100644
--- a/net/sched/sch_api.c
+++ b/net/sched/sch_api.c
@@ -28,6 +28,7 @@
```

```
 #include <linux/list.h>
 #include <linux/hrtimer.h>

+#include <net/net_namespace.h>
 #include <net/netlink.h>
 #include <net/pkt_sched.h>

@@ -1251,7 +1252,7 @@ static int __init pktsched_init(void)
 {
  register_qdisc(&pfifo_qdisc_ops);
  register_qdisc(&bfifo_qdisc_ops);
- proc_net_fops_create("psched", 0, &psched_fops);
+ proc_net_fops_create(&init_net, "psched", 0, &psched_fops);

  rtnl_register(PF_UNSPEC, RTM_NEWQDISC, tc_modify_qdisc, NULL);
  rtnl_register(PF_UNSPEC, RTM_DELQDISC, tc_get_qdisc, NULL);
diff --git a/net/sctp/protocol.c b/net/sctp/protocol.c
index a746ebf..a015454 100644
--- a/net/sctp/protocol.c
+++ b/net/sctp/protocol.c
@@ -52,6 +52,7 @@
 #include <linux/inetdevice.h>
 #include <linux/seq_file.h>
 #include <linux/bootmem.h>
+#include <net/net_namespace.h>
 #include <net/protocol.h>
 #include <net/ip.h>
 #include <net/ipv6.h>
@@ -98,7 +99,7 @@ static __init int sctp_proc_init(void)
 {
  if (!proc_net_sctp) {
   struct proc_dir_entry *ent;
-  ent = proc_mkdir("net/sctp", NULL);
+  ent = proc_mkdir("sctp", init_net.proc_net);
   if (ent) {
    ent->owner = THIS_MODULE;
    proc_net_sctp = ent;
@@ -131,7 +132,7 @@ static void sctp_proc_exit(void)

  if (proc_net_sctp) {
   proc_net_sctp = NULL;
-  remove_proc_entry("net/sctp", NULL);
+  remove_proc_entry("sctp", init_net.proc_net);
  }
 }

diff --git a/net/sunrpc/stats.c b/net/sunrpc/stats.c
index 74ba7d4..4d4f373 100644
```

```
--- a/net/sunrpc/stats.c
+++ b/net/sunrpc/stats.c
@@ -21,6 +21,7 @@
 #include <linux/sunrpc/clnt.h>
 #include <linux/sunrpc/svcsock.h>
 #include <linux/sunrpc/metrics.h>
+#include <net/net_namespace.h>

 #define RPCDBG_FACILITY RPCDBG_MISC

@@ -265,7 +266,7 @@ rpc_proc_init(void)
  dprintk("RPC:       registering /proc/net/rpc\n");
  if (!proc_net_rpc) {
   struct proc_dir_entry *ent;
-  ent = proc_mkdir("rpc", proc_net);
+  ent = proc_mkdir("rpc", init_net.proc_net);
   if (ent) {
    ent->owner = THIS_MODULE;
    proc_net_rpc = ent;
@@ -279,7 +280,7 @@ rpc_proc_exit(void)
  dprintk("RPC:       unregistering /proc/net/rpc\n");
  if (proc_net_rpc) {
   proc_net_rpc = NULL;
-  remove_proc_entry("net/rpc", NULL);
+  remove_proc_entry("rpc", init_net.proc_net);
  }
 }

diff --git a/net/unix/af_unix.c b/net/unix/af_unix.c
index a05c342..2386090 100644
--- a/net/unix/af_unix.c
+++ b/net/unix/af_unix.c
@@ -103,6 +103,7 @@
 #include <asm/uaccess.h>
 #include <linux/skbuff.h>
 #include <linux/netdevice.h>
+#include <net/net_namespace.h>
 #include <net/sock.h>
 #include <net/tcp_states.h>
 #include <net/af_unix.h>
@@ -2135,7 +2136,7 @@ static int __init af_unix_init(void)

  sock_register(&unix_family_ops);
 #ifdef CONFIG_PROC_FS
- proc_net_fops_create("unix", 0, &unix_seq_fops);
+ proc_net_fops_create(&init_net, "unix", 0, &unix_seq_fops);
 #endif
  unix_sysctl_register();
```

```
 out:
@@ -2146,7 +2147,7 @@ static void __exit af_unix_exit(void)
 {
  sock_unregister(PF_UNIX);
  unix_sysctl_unregister();
- proc_net_remove("unix");
+ proc_net_remove(&init_net, "unix");
  proto_unregister(&unix_proto);
 }

diff --git a/net/wanrouter/wanproc.c b/net/wanrouter/wanproc.c
index 236e7ea..f2e54c3 100644
--- a/net/wanrouter/wanproc.c
+++ b/net/wanrouter/wanproc.c
@@ -29,6 +29,7 @@
 #include <linux/seq_file.h>
 #include <linux/smp_lock.h>

+#include <net/net_namespace.h>
 #include <asm/io.h>

 #define PROC_STATS_FORMAT "%30s: %12lu\n"
@@ -287,7 +288,7 @@ static const struct file_operations wandev_fops = {
 int __init wanrouter_proc_init(void)
 {
  struct proc_dir_entry *p;
- proc_router = proc_mkdir(ROUTER_NAME, proc_net);
+ proc_router = proc_mkdir(ROUTER_NAME, init_net.proc_net);
  if (!proc_router)
   goto fail;

@@ -303,7 +304,7 @@ int __init wanrouter_proc_init(void)
 fail_stat:
  remove_proc_entry("config", proc_router);
 fail_config:
- remove_proc_entry(ROUTER_NAME, proc_net);
+ remove_proc_entry(ROUTER_NAME, init_net.proc_net);
 fail:
  return -ENOMEM;
 }
@@ -316,7 +317,7 @@ void wanrouter_proc_cleanup(void)
 {
  remove_proc_entry("config", proc_router);
  remove_proc_entry("status", proc_router);
- remove_proc_entry(ROUTER_NAME, proc_net);
+ remove_proc_entry(ROUTER_NAME, init_net.proc_net);
 }
```

```
 /*
diff --git a/net/wireless/wext.c b/net/wireless/wext.c
index debf519..b8069af 100644
--- a/net/wireless/wext.c
+++ b/net/wireless/wext.c
@@ -93,6 +93,7 @@
 #include <linux/if_arp.h>  /* ARPHRD_ETHER */
 #include <linux/etherdevice.h>  /* compare_ether_addr */
 #include <linux/interrupt.h>
+#include <net/net_namespace.h>

 #include <linux/wireless.h>  /* Pretty obvious */
 #include <net/iw_handler.h>  /* New driver API */
@@ -686,7 +687,7 @@ static const struct file_operations wireless_seq_fops = {
 int __init wext_proc_init(void)
 {
  /* Create /proc/net/wireless entry */
- if (!proc_net_fops_create("wireless", S_IRUGO, &wireless_seq_fops))
+ if (!proc_net_fops_create(&init_net, "wireless", S_IRUGO, &wireless_seq_fops))
   return -ENOMEM;

  return 0;
diff --git a/net/x25/x25_proc.c b/net/x25/x25_proc.c
index 7405b9c..7d55e50 100644
--- a/net/x25/x25_proc.c
+++ b/net/x25/x25_proc.c
@@ -20,6 +20,7 @@
 #include <linux/init.h>
 #include <linux/proc_fs.h>
 #include <linux/seq_file.h>
+#include <net/net_namespace.h>
 #include <net/sock.h>
 #include <net/x25.h>

@@ -301,7 +302,7 @@ int __init x25_proc_init(void)
  struct proc_dir_entry *p;
  int rc = -ENOMEM;

- x25_proc_dir = proc_mkdir("x25", proc_net);
+ x25_proc_dir = proc_mkdir("x25", init_net.proc_net);
  if (!x25_proc_dir)
   goto out;

@@ -328,7 +329,7 @@ out_forward:
 out_socket:
  remove_proc_entry("route", x25_proc_dir);
 out_route:
- remove_proc_entry("x25", proc_net);
```

```
+ remove_proc_entry("x25", init_net.proc_net);
  goto out;
 }

@@ -337,7 +338,7 @@ void __exit x25_proc_exit(void)
  remove_proc_entry("forward", x25_proc_dir);
  remove_proc_entry("route", x25_proc_dir);
  remove_proc_entry("socket", x25_proc_dir);
- remove_proc_entry("x25", proc_net);
+ remove_proc_entry("x25", init_net.proc_net);
 }

 #else /* CONFIG_PROC_FS */
--
1.5.3.rc6.17.g1911
```

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: [PATCH 06/16] net: Add a network namespace parameter to struct sock
Posted by ebiederm on Sat, 08 Sep 2007 21:21:37 GMT
View Forum Message <> Reply to Message

Sockets need to get a reference to their network namespace,
or possibly a simple hold if someone registers on the network
namespace notifier and will free the sockets when the namespace
is going to be destroyed.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 include/net/inet_timewait_sock.h | 1 +
 include/net/sock.h | 3 +++
 2 files changed, 4 insertions(+), 0 deletions(-)

```
diff --git a/include/net/inet_timewait_sock.h b/include/net/inet_timewait_sock.h
index 47d52b2..abaff05 100644
--- a/include/net/inet_timewait_sock.h
+++ b/include/net/inet_timewait_sock.h
@@ -115,6 +115,7 @@ struct inet_timewait_sock {
 #define tw_refcnt  __tw_common.skc_refcnt
 #define tw_hash    __tw_common.skc_hash
 #define tw_prot    __tw_common.skc_prot
+#define tw_net     __tw_common.skc_net
  volatile unsigned char tw_substate;
  /* 3 bits hole, try to pack */
```

```
  unsigned char  tw_rcv_wscale;
diff --git a/include/net/sock.h b/include/net/sock.h
index 802c670..253df3f 100644
--- a/include/net/sock.h
+++ b/include/net/sock.h
@@ -106,6 +106,7 @@ struct proto;
  * @skc_refcnt: reference count
  * @skc_hash: hash value used with various protocol lookup tables
  * @skc_prot: protocol handlers inside a network family
+ * @skc_net: reference to the network namespace of this socket
  *
  * This is the minimal network layer representation of sockets, the header
  * for struct sock and struct inet_timewait_sock.
@@ -120,6 +121,7 @@ struct sock_common {
  atomic_t  skc_refcnt;
  unsigned int  skc_hash;
  struct proto  *skc_prot;
+ struct net   *skc_net;
 };

 /**
@@ -196,6 +198,7 @@ struct sock {
 #define sk_refcnt  __sk_common.skc_refcnt
 #define sk_hash   __sk_common.skc_hash
 #define sk_prot   __sk_common.skc_prot
+#define sk_net   __sk_common.skc_net
  unsigned char  sk_shutdown : 2,
     sk_no_check : 2,
     sk_userlocks : 4;
--
1.5.3.rc6.17.g1911
```

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: [PATCH 08/16] net: Make socket creation namespace safe.
Posted by ebiederm on Sat, 08 Sep 2007 21:23:01 GMT
View Forum Message <> Reply to Message

This patch passes in the namespace a new socket should be created in
and has the socket code do the appropriate reference counting. By
virtue of this all socket create methods are touched. In addition
the socket create methods are modified so that they will fail if
you attempt to create a socket in a non-default network namespace.

Failing if we attempt to create a socket outside of the default
network namespace ensures that as we incrementally make the network stack
network namespace aware we will not export functionality that someone
has not audited and made certain is network namespace safe.
Allowing us to partially enable network namespaces before all of the
exotic protocols are supported.

Any protocol layers I have missed will fail to compile because I now
pass an extra parameter into the socket creation code.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 drivers/net/pppoe.c | 4 ++--
 drivers/net/pppol2tp.c | 4 ++--
 drivers/net/pppox.c | 7 +++++--
 include/linux/if_pppox.h | 2 +-
 include/linux/net.h | 3 ++-
 include/net/llc_conn.h | 2 +-
 include/net/sock.h | 4 +++-
 net/appletalk/ddp.c | 7 +++++--
 net/atm/common.c | 4 ++--
 net/atm/common.h | 2 +-
 net/atm/pvc.c | 7 +++++--
 net/atm/svc.c | 11 +++++++----
 net/ax25/af_ax25.c | 9 ++++++---
 net/bluetooth/af_bluetooth.c | 7 +++++--
 net/bluetooth/bnep/sock.c | 4 ++--
 net/bluetooth/cmtp/sock.c | 4 ++--
 net/bluetooth/hci_sock.c | 4 ++--
 net/bluetooth/hidp/sock.c | 4 ++--
 net/bluetooth/l2cap.c | 10 +++++-----
 net/bluetooth/rfcomm/sock.c | 10 +++++-----
 net/bluetooth/sco.c | 10 +++++-----
 net/core/sock.c | 6 ++++--
 net/decnet/af_decnet.c | 13 +++++++++---
 net/econet/af_econet.c | 7 +++++--
 net/ipv4/af_inet.c | 7 +++++--
 net/ipv6/af_inet6.c | 7 +++++--
 net/ipx/af_ipx.c | 7 +++++--
 net/irda/af_irda.c | 11 +++++++----
 net/key/af_key.c | 7 +++++--
 net/llc/af_llc.c | 7 +++++--
 net/llc/llc_conn.c | 6 ++++--
 net/netlink/af_netlink.c | 15 ++++++++++---
 net/netrom/af_netrom.c | 9 ++++++---
 net/packet/af_packet.c | 7 +++++--
 net/rose/af_rose.c | 9 ++++++---
 net/rxrpc/af_rxrpc.c | 7 +++++--

```
 net/sctp/ipv6.c            |    2 +-
 net/sctp/protocol.c        |    2 +-
 net/socket.c               |    9 +++++----
 net/tipc/socket.c          |    9 ++++++---
 net/unix/af_unix.c         |   13 ++++++++-----
 net/x25/af_x25.c           |   13 ++++++++-----
 42 files changed, 182 insertions(+), 110 deletions(-)

diff --git a/drivers/net/pppoe.c b/drivers/net/pppoe.c
index a9b6971..f8bf5fc 100644
--- a/drivers/net/pppoe.c
+++ b/drivers/net/pppoe.c
@@ -477,12 +477,12 @@ static struct proto pppoe_sk_proto = {
  * Initialize a new struct sock.
  *
  **********************************************************************/
-static int pppoe_create(struct socket *sock)
+static int pppoe_create(struct net *net, struct socket *sock)
 {
  int error = -ENOMEM;
  struct sock *sk;

- sk = sk_alloc(PF_PPPOX, GFP_KERNEL, &pppoe_sk_proto, 1);
+ sk = sk_alloc(net, PF_PPPOX, GFP_KERNEL, &pppoe_sk_proto, 1);
  if (!sk)
   goto out;

diff --git a/drivers/net/pppol2tp.c b/drivers/net/pppol2tp.c
index c12e0a8..07d7f5b 100644
--- a/drivers/net/pppol2tp.c
+++ b/drivers/net/pppol2tp.c
@@ -1423,12 +1423,12 @@ static struct proto pppol2tp_sk_proto = {

 /* socket() handler. Initialize a new struct sock.
  */
-static int pppol2tp_create(struct socket *sock)
+static int pppol2tp_create(struct net *net, struct socket *sock)
 {
  int error = -ENOMEM;
  struct sock *sk;

- sk = sk_alloc(PF_PPPOX, GFP_KERNEL, &pppol2tp_sk_proto, 1);
+ sk = sk_alloc(net, PF_PPPOX, GFP_KERNEL, &pppol2tp_sk_proto, 1);
  if (!sk)
   goto out;

diff --git a/drivers/net/pppox.c b/drivers/net/pppox.c
index 25c52b5..c6898c1 100644
```

```
--- a/drivers/net/pppox.c
+++ b/drivers/net/pppox.c
@@ -104,10 +104,13 @@ int pppox_ioctl(struct socket *sock, unsigned int cmd, unsigned long
arg)

 EXPORT_SYMBOL(pppox_ioctl);

-static int pppox_create(struct socket *sock, int protocol)
+static int pppox_create(struct net *net, struct socket *sock, int protocol)
 {
  int rc = -EPROTOTYPE;

+ if (net != &init_net)
+  return -EAFNOSUPPORT;
+
  if (protocol < 0 || protocol > PX_MAX_PROTO)
   goto out;

@@ -123,7 +126,7 @@ static int pppox_create(struct socket *sock, int protocol)
      !try_module_get(pppox_protos[protocol]->owner))
   goto out;

- rc = pppox_protos[protocol]->create(sock);
+ rc = pppox_protos[protocol]->create(net, sock);

  module_put(pppox_protos[protocol]->owner);
 out:
diff --git a/include/linux/if_pppox.h b/include/linux/if_pppox.h
index 2565254..43cfc9f 100644
--- a/include/linux/if_pppox.h
+++ b/include/linux/if_pppox.h
@@ -172,7 +172,7 @@ static inline struct sock *sk_pppox(struct pppox_sock *po)
 struct module;

 struct pppox_proto {
- int  (*create)(struct socket *sock);
+ int  (*create)(struct net *net, struct socket *sock);
  int  (*ioctl)(struct socket *sock, unsigned int cmd,
      unsigned long arg);
  struct module *owner;
diff --git a/include/linux/net.h b/include/linux/net.h
index efc4517..c136abc 100644
--- a/include/linux/net.h
+++ b/include/linux/net.h
@@ -23,6 +23,7 @@

 struct poll_table_struct;
 struct inode;
```

```
+struct net;

 #define NPROTO  34  /* should be enough for now.. */

@@ -169,7 +170,7 @@ struct proto_ops {

 struct net_proto_family {
  int  family;
- int  (*create)(struct socket *sock, int protocol);
+ int  (*create)(struct net *net, struct socket *sock, int protocol);
  struct module *owner;
 };

diff --git a/include/net/llc_conn.h b/include/net/llc_conn.h
index 00730d2..e2374e3 100644
--- a/include/net/llc_conn.h
+++ b/include/net/llc_conn.h
@@ -93,7 +93,7 @@ static __inline__ char llc_backlog_type(struct sk_buff *skb)
  return skb->cb[sizeof(skb->cb) - 1];
 }

-extern struct sock *llc_sk_alloc(int family, gfp_t priority,
+extern struct sock *llc_sk_alloc(struct net *net, int family, gfp_t priority,
       struct proto *prot);
 extern void llc_sk_free(struct sock *sk);

diff --git a/include/net/sock.h b/include/net/sock.h
index 253df3f..898413f 100644
--- a/include/net/sock.h
+++ b/include/net/sock.h
@@ -56,6 +56,7 @@
 #include <asm/atomic.h>
 #include <net/dst.h>
 #include <net/checksum.h>
+#include <net/net_namespace.h>

 /*
  * This structure really needs to be cleaned up.
@@ -777,7 +778,7 @@ extern void FASTCALL(release_sock(struct sock *sk));
     SINGLE_DEPTH_NESTING)
 #define bh_unlock_sock(__sk) spin_unlock(&((__sk)->sk_lock.slock))

-extern struct sock  *sk_alloc(int family,
+extern struct sock  *sk_alloc(struct net *net, int family,
        gfp_t priority,
        struct proto *prot, int zero_it);
 extern void   sk_free(struct sock *sk);
@@ -1006,6 +1007,7 @@ static inline void sock_copy(struct sock *nsk, const struct sock *osk)
```

```
 #endif

  memcpy(nsk, osk, osk->sk_prot->obj_size);
+ get_net(nsk->sk_net);
 #ifdef CONFIG_SECURITY_NETWORK
  nsk->sk_security = sptr;
  security_sk_clone(osk, nsk);
diff --git a/net/appletalk/ddp.c b/net/appletalk/ddp.c
index 594b597..fd1d52f 100644
--- a/net/appletalk/ddp.c
+++ b/net/appletalk/ddp.c
@@ -1026,11 +1026,14 @@ static struct proto ddp_proto = {
  * Create a socket. Initialise the socket, blank the addresses
  * set the state.
  */
-static int atalk_create(struct socket *sock, int protocol)
+static int atalk_create(struct net *net, struct socket *sock, int protocol)
 {
  struct sock *sk;
  int rc = -ESOCKTNOSUPPORT;

+ if (net != &init_net)
+  return -EAFNOSUPPORT;
+
  /*
   * We permit SOCK_DGRAM and RAW is an extension. It is trivial to do
   * and gives you the full ELAP frame. Should be handy for CAP 8)
@@ -1038,7 +1041,7 @@ static int atalk_create(struct socket *sock, int protocol)
  if (sock->type != SOCK_RAW && sock->type != SOCK_DGRAM)
   goto out;
  rc = -ENOMEM;
- sk = sk_alloc(PF_APPLETALK, GFP_KERNEL, &ddp_proto, 1);
+ sk = sk_alloc(net, PF_APPLETALK, GFP_KERNEL, &ddp_proto, 1);
  if (!sk)
   goto out;
  rc = 0;
diff --git a/net/atm/common.c b/net/atm/common.c
index 299ec1e..e166d9e 100644
--- a/net/atm/common.c
+++ b/net/atm/common.c
@@ -125,7 +125,7 @@ static struct proto vcc_proto = {
  .obj_size = sizeof(struct atm_vcc),
 };

-int vcc_create(struct socket *sock, int protocol, int family)
+int vcc_create(struct net *net, struct socket *sock, int protocol, int family)
 {
  struct sock *sk;
```

```
  struct atm_vcc *vcc;
@@ -133,7 +133,7 @@ int vcc_create(struct socket *sock, int protocol, int family)
  sock->sk = NULL;
  if (sock->type == SOCK_STREAM)
   return -EINVAL;
- sk = sk_alloc(family, GFP_KERNEL, &vcc_proto, 1);
+ sk = sk_alloc(net, family, GFP_KERNEL, &vcc_proto, 1);
  if (!sk)
   return -ENOMEM;
  sock_init_data(sock, sk);
diff --git a/net/atm/common.h b/net/atm/common.h
index ad78c9e..16f32c1 100644
--- a/net/atm/common.h
+++ b/net/atm/common.h
@@ -10,7 +10,7 @@
 #include <linux/poll.h> /* for poll_table */


-int vcc_create(struct socket *sock, int protocol, int family);
+int vcc_create(struct net *net, struct socket *sock, int protocol, int family);
 int vcc_release(struct socket *sock);
 int vcc_connect(struct socket *sock, int itf, short vpi, int vci);
 int vcc_recvmsg(struct kiocb *iocb, struct socket *sock, struct msghdr *msg,
diff --git a/net/atm/pvc.c b/net/atm/pvc.c
index 848e6e1..43e8bf5 100644
--- a/net/atm/pvc.c
+++ b/net/atm/pvc.c
@@ -124,10 +124,13 @@ static const struct proto_ops pvc_proto_ops = {
 };


-static int pvc_create(struct socket *sock,int protocol)
+static int pvc_create(struct net *net, struct socket *sock,int protocol)
 {
+ if (net != &init_net)
+  return -EAFNOSUPPORT;
+
  sock->ops = &pvc_proto_ops;
- return vcc_create(sock, protocol, PF_ATMPVC);
+ return vcc_create(net, sock, protocol, PF_ATMPVC);
 }


diff --git a/net/atm/svc.c b/net/atm/svc.c
index 53d04c7..daf9a48 100644
--- a/net/atm/svc.c
+++ b/net/atm/svc.c
@@ -25,7 +25,7 @@
```

```
 #include "signaling.h"
 #include "addr.h"

-static int svc_create(struct socket *sock,int protocol);
+static int svc_create(struct net *net, struct socket *sock,int protocol);

 /*
  * Note: since all this is still nicely synchronized with the signaling demon,
@@ -326,7 +326,7 @@ static int svc_accept(struct socket *sock,struct socket *newsock,int flags)

  lock_sock(sk);

- error = svc_create(newsock,0);
+ error = svc_create(sk->sk_net, newsock,0);
  if (error)
   goto out;

@@ -627,12 +627,15 @@ static const struct proto_ops svc_proto_ops = {
 };


-static int svc_create(struct socket *sock,int protocol)
+static int svc_create(struct net *net, struct socket *sock,int protocol)
 {
  int error;

+ if (net != &init_net)
+  return -EAFNOSUPPORT;
+
  sock->ops = &svc_proto_ops;
- error = vcc_create(sock, protocol, AF_ATMSVC);
+ error = vcc_create(net, sock, protocol, AF_ATMSVC);
  if (error) return error;
  ATM_SD(sock)->local.sas_family = AF_ATMSVC;
  ATM_SD(sock)->remote.sas_family = AF_ATMSVC;
diff --git a/net/ax25/af_ax25.c b/net/ax25/af_ax25.c
index 1d71f85..def6c42 100644
--- a/net/ax25/af_ax25.c
+++ b/net/ax25/af_ax25.c
@@ -780,11 +780,14 @@ static struct proto ax25_proto = {
  .obj_size = sizeof(struct sock),
 };

-static int ax25_create(struct socket *sock, int protocol)
+static int ax25_create(struct net *net, struct socket *sock, int protocol)
 {
  struct sock *sk;
  ax25_cb *ax25;
```

```
+ if (net != &init_net)
+  return -EAFNOSUPPORT;
+
  switch (sock->type) {
  case SOCK_DGRAM:
   if (protocol == 0 || protocol == PF_AX25)
@@ -830,7 +833,7 @@ static int ax25_create(struct socket *sock, int protocol)
   return -ESOCKTNOSUPPORT;
  }

- if ((sk = sk_alloc(PF_AX25, GFP_ATOMIC, &ax25_proto, 1)) == NULL)
+ if ((sk = sk_alloc(net, PF_AX25, GFP_ATOMIC, &ax25_proto, 1)) == NULL)
   return -ENOMEM;

  ax25 = sk->sk_protinfo = ax25_create_cb();
@@ -855,7 +858,7 @@ struct sock *ax25_make_new(struct sock *osk, struct ax25_dev
*ax25_dev)
  struct sock *sk;
  ax25_cb *ax25, *oax25;

- if ((sk = sk_alloc(PF_AX25, GFP_ATOMIC, osk->sk_prot, 1)) == NULL)
+ if ((sk = sk_alloc(osk->sk_net, PF_AX25, GFP_ATOMIC, osk->sk_prot, 1)) == NULL)
   return NULL;

  if ((ax25 = ax25_create_cb()) == NULL) {
diff --git a/net/bluetooth/af_bluetooth.c b/net/bluetooth/af_bluetooth.c
index d942b94..1220d8a 100644
--- a/net/bluetooth/af_bluetooth.c
+++ b/net/bluetooth/af_bluetooth.c
@@ -95,10 +95,13 @@ int bt_sock_unregister(int proto)
 }
 EXPORT_SYMBOL(bt_sock_unregister);

-static int bt_sock_create(struct socket *sock, int proto)
+static int bt_sock_create(struct net *net, struct socket *sock, int proto)
 {
  int err;

+ if (net != &init_net)
+  return -EAFNOSUPPORT;
+
  if (proto < 0 || proto >= BT_MAX_PROTO)
   return -EINVAL;

@@ -113,7 +116,7 @@ static int bt_sock_create(struct socket *sock, int proto)
  read_lock(&bt_proto_lock);
```

```
  if (bt_proto[proto] && try_module_get(bt_proto[proto]->owner)) {
-  err = bt_proto[proto]->create(sock, proto);
+  err = bt_proto[proto]->create(net, sock, proto);
   module_put(bt_proto[proto]->owner);
  }

diff --git a/net/bluetooth/bnep/sock.c b/net/bluetooth/bnep/sock.c
index 10292e7..f718965 100644
--- a/net/bluetooth/bnep/sock.c
+++ b/net/bluetooth/bnep/sock.c
@@ -204,7 +204,7 @@ static struct proto bnep_proto = {
  .obj_size = sizeof(struct bt_sock)
 };

-static int bnep_sock_create(struct socket *sock, int protocol)
+static int bnep_sock_create(struct net *net, struct socket *sock, int protocol)
 {
  struct sock *sk;

@@ -213,7 +213,7 @@ static int bnep_sock_create(struct socket *sock, int protocol)
  if (sock->type != SOCK_RAW)
   return -ESOCKTNOSUPPORT;

- sk = sk_alloc(PF_BLUETOOTH, GFP_ATOMIC, &bnep_proto, 1);
+ sk = sk_alloc(net, PF_BLUETOOTH, GFP_ATOMIC, &bnep_proto, 1);
  if (!sk)
   return -ENOMEM;

diff --git a/net/bluetooth/cmtp/sock.c b/net/bluetooth/cmtp/sock.c
index 19be786..cf700c2 100644
--- a/net/bluetooth/cmtp/sock.c
+++ b/net/bluetooth/cmtp/sock.c
@@ -195,7 +195,7 @@ static struct proto cmtp_proto = {
  .obj_size = sizeof(struct bt_sock)
 };

-static int cmtp_sock_create(struct socket *sock, int protocol)
+static int cmtp_sock_create(struct net *net, struct socket *sock, int protocol)
 {
  struct sock *sk;

@@ -204,7 +204,7 @@ static int cmtp_sock_create(struct socket *sock, int protocol)
  if (sock->type != SOCK_RAW)
   return -ESOCKTNOSUPPORT;

- sk = sk_alloc(PF_BLUETOOTH, GFP_ATOMIC, &cmtp_proto, 1);
+ sk = sk_alloc(net, PF_BLUETOOTH, GFP_ATOMIC, &cmtp_proto, 1);
  if (!sk)
```

```
   return -ENOMEM;

diff --git a/net/bluetooth/hci_sock.c b/net/bluetooth/hci_sock.c
index 1dae3df..dc2ba7b 100644
--- a/net/bluetooth/hci_sock.c
+++ b/net/bluetooth/hci_sock.c
@@ -618,7 +618,7 @@ static struct proto hci_sk_proto = {
  .obj_size = sizeof(struct hci_pinfo)
 };

-static int hci_sock_create(struct socket *sock, int protocol)
+static int hci_sock_create(struct net *net, struct socket *sock, int protocol)
 {
  struct sock *sk;

@@ -629,7 +629,7 @@ static int hci_sock_create(struct socket *sock, int protocol)

  sock->ops = &hci_sock_ops;

- sk = sk_alloc(PF_BLUETOOTH, GFP_ATOMIC, &hci_sk_proto, 1);
+ sk = sk_alloc(net, PF_BLUETOOTH, GFP_ATOMIC, &hci_sk_proto, 1);
  if (!sk)
   return -ENOMEM;

diff --git a/net/bluetooth/hidp/sock.c b/net/bluetooth/hidp/sock.c
index 0c18525..1de2b6f 100644
--- a/net/bluetooth/hidp/sock.c
+++ b/net/bluetooth/hidp/sock.c
@@ -246,7 +246,7 @@ static struct proto hidp_proto = {
  .obj_size = sizeof(struct bt_sock)
 };

-static int hidp_sock_create(struct socket *sock, int protocol)
+static int hidp_sock_create(struct net *net, struct socket *sock, int protocol)
 {
  struct sock *sk;

@@ -255,7 +255,7 @@ static int hidp_sock_create(struct socket *sock, int protocol)
  if (sock->type != SOCK_RAW)
   return -ESOCKTNOSUPPORT;

- sk = sk_alloc(PF_BLUETOOTH, GFP_ATOMIC, &hidp_proto, 1);
+ sk = sk_alloc(net, PF_BLUETOOTH, GFP_ATOMIC, &hidp_proto, 1);
  if (!sk)
   return -ENOMEM;

diff --git a/net/bluetooth/l2cap.c b/net/bluetooth/l2cap.c
index c4e4ce4..36ef27b 100644
```

```
--- a/net/bluetooth/l2cap.c
+++ b/net/bluetooth/l2cap.c
@@ -518,11 +518,11 @@ static struct proto l2cap_proto = {
  .obj_size = sizeof(struct l2cap_pinfo)
 };

-static struct sock *l2cap_sock_alloc(struct socket *sock, int proto, gfp_t prio)
+static struct sock *l2cap_sock_alloc(struct net *net, struct socket *sock, int proto, gfp_t prio)
 {
  struct sock *sk;

- sk = sk_alloc(PF_BLUETOOTH, prio, &l2cap_proto, 1);
+ sk = sk_alloc(net, PF_BLUETOOTH, prio, &l2cap_proto, 1);
  if (!sk)
   return NULL;

@@ -543,7 +543,7 @@ static struct sock *l2cap_sock_alloc(struct socket *sock, int proto, gfp_t
prio)
  return sk;
 }

-static int l2cap_sock_create(struct socket *sock, int protocol)
+static int l2cap_sock_create(struct net *net, struct socket *sock, int protocol)
 {
  struct sock *sk;

@@ -560,7 +560,7 @@ static int l2cap_sock_create(struct socket *sock, int protocol)

  sock->ops = &l2cap_sock_ops;

- sk = l2cap_sock_alloc(sock, protocol, GFP_ATOMIC);
+ sk = l2cap_sock_alloc(net, sock, protocol, GFP_ATOMIC);
  if (!sk)
   return -ENOMEM;

@@ -1425,7 +1425,7 @@ static inline int l2cap_connect_req(struct l2cap_conn *conn, struct
l2cap_cmd_hd
   goto response;
  }

- sk = l2cap_sock_alloc(NULL, BTPROTO_L2CAP, GFP_ATOMIC);
+ sk = l2cap_sock_alloc(parent->sk_net, NULL, BTPROTO_L2CAP, GFP_ATOMIC);
  if (!sk)
   goto response;

diff --git a/net/bluetooth/rfcomm/sock.c b/net/bluetooth/rfcomm/sock.c
index 30586ab..266b697 100644
--- a/net/bluetooth/rfcomm/sock.c
```

```
+++ b/net/bluetooth/rfcomm/sock.c
@@ -282,12 +282,12 @@ static struct proto rfcomm_proto = {
  .obj_size = sizeof(struct rfcomm_pinfo)
 };

-static struct sock *rfcomm_sock_alloc(struct socket *sock, int proto, gfp_t prio)
+static struct sock *rfcomm_sock_alloc(struct net *net, struct socket *sock, int proto, gfp_t prio)
 {
  struct rfcomm_dlc *d;
  struct sock *sk;

- sk = sk_alloc(PF_BLUETOOTH, prio, &rfcomm_proto, 1);
+ sk = sk_alloc(net, PF_BLUETOOTH, prio, &rfcomm_proto, 1);
  if (!sk)
   return NULL;

@@ -323,7 +323,7 @@ static struct sock *rfcomm_sock_alloc(struct socket *sock, int proto, gfp_t
prio
  return sk;
 }

-static int rfcomm_sock_create(struct socket *sock, int protocol)
+static int rfcomm_sock_create(struct net *net, struct socket *sock, int protocol)
 {
  struct sock *sk;

@@ -336,7 +336,7 @@ static int rfcomm_sock_create(struct socket *sock, int protocol)

  sock->ops = &rfcomm_sock_ops;

- sk = rfcomm_sock_alloc(sock, protocol, GFP_ATOMIC);
+ sk = rfcomm_sock_alloc(net, sock, protocol, GFP_ATOMIC);
  if (!sk)
   return -ENOMEM;

@@ -868,7 +868,7 @@ int rfcomm_connect_ind(struct rfcomm_session *s, u8 channel, struct
rfcomm_dlc *
   goto done;
  }

- sk = rfcomm_sock_alloc(NULL, BTPROTO_RFCOMM, GFP_ATOMIC);
+ sk = rfcomm_sock_alloc(parent->sk_net, NULL, BTPROTO_RFCOMM, GFP_ATOMIC);
  if (!sk)
   goto done;

diff --git a/net/bluetooth/sco.c b/net/bluetooth/sco.c
index 3f5163e..65b6fb1 100644
--- a/net/bluetooth/sco.c
```

```
+++ b/net/bluetooth/sco.c
@@ -414,11 +414,11 @@ static struct proto sco_proto = {
  .obj_size = sizeof(struct sco_pinfo)
 };

-static struct sock *sco_sock_alloc(struct socket *sock, int proto, gfp_t prio)
+static struct sock *sco_sock_alloc(struct net *net, struct socket *sock, int proto, gfp_t prio)
 {
  struct sock *sk;

- sk = sk_alloc(PF_BLUETOOTH, prio, &sco_proto, 1);
+ sk = sk_alloc(net, PF_BLUETOOTH, prio, &sco_proto, 1);
  if (!sk)
   return NULL;

@@ -439,7 +439,7 @@ static struct sock *sco_sock_alloc(struct socket *sock, int proto, gfp_t
prio)
  return sk;
 }

-static int sco_sock_create(struct socket *sock, int protocol)
+static int sco_sock_create(struct net *net, struct socket *sock, int protocol)
 {
  struct sock *sk;

@@ -452,7 +452,7 @@ static int sco_sock_create(struct socket *sock, int protocol)

  sock->ops = &sco_sock_ops;

- sk = sco_sock_alloc(sock, protocol, GFP_ATOMIC);
+ sk = sco_sock_alloc(net, sock, protocol, GFP_ATOMIC);
  if (!sk)
   return -ENOMEM;

@@ -807,7 +807,7 @@ static void sco_conn_ready(struct sco_conn *conn)

   bh_lock_sock(parent);

-  sk = sco_sock_alloc(NULL, BTPROTO_SCO, GFP_ATOMIC);
+  sk = sco_sock_alloc(parent->sk_net, NULL, BTPROTO_SCO, GFP_ATOMIC);
   if (!sk) {
    bh_unlock_sock(parent);
    goto done;
diff --git a/net/core/sock.c b/net/core/sock.c
index 3d16a4b..9feec0f 100644
--- a/net/core/sock.c
+++ b/net/core/sock.c
@@ -863,7 +863,7 @@ static inline void sock_lock_init(struct sock *sk)
```

```
  * @prot: struct proto associated with this new sock instance
  * @zero_it: if we should zero the newly allocated sock
  */
-struct sock *sk_alloc(int family, gfp_t priority,
+struct sock *sk_alloc(struct net *net, int family, gfp_t priority,
         struct proto *prot, int zero_it)
 {
  struct sock *sk = NULL;
@@ -884,6 +884,7 @@ struct sock *sk_alloc(int family, gfp_t priority,
     */
    sk->sk_prot = sk->sk_prot_creator = prot;
    sock_lock_init(sk);
+   sk->sk_net = get_net(net);
   }

   if (security_sk_alloc(sk, family, priority))
@@ -923,6 +924,7 @@ void sk_free(struct sock *sk)
         __FUNCTION__, atomic_read(&sk->sk_omem_alloc));

  security_sk_free(sk);
+ put_net(sk->sk_net);
  if (sk->sk_prot_creator->slab != NULL)
   kmem_cache_free(sk->sk_prot_creator->slab, sk);
  else
@@ -932,7 +934,7 @@ void sk_free(struct sock *sk)

 struct sock *sk_clone(const struct sock *sk, const gfp_t priority)
 {
- struct sock *newsk = sk_alloc(sk->sk_family, priority, sk->sk_prot, 0);
+ struct sock *newsk = sk_alloc(sk->sk_net, sk->sk_family, priority, sk->sk_prot, 0);

  if (newsk != NULL) {
   struct sk_filter *filter;
diff --git a/net/decnet/af_decnet.c b/net/decnet/af_decnet.c
index 625d595..aca4c49 100644
--- a/net/decnet/af_decnet.c
+++ b/net/decnet/af_decnet.c
@@ -471,10 +471,10 @@ static struct proto dn_proto = {
  .obj_size  = sizeof(struct dn_sock),
 };

-static struct sock *dn_alloc_sock(struct socket *sock, gfp_t gfp)
+static struct sock *dn_alloc_sock(struct net *net, struct socket *sock, gfp_t gfp)
 {
  struct dn_scp *scp;
- struct sock *sk = sk_alloc(PF_DECnet, gfp, &dn_proto, 1);
+ struct sock *sk = sk_alloc(net, PF_DECnet, gfp, &dn_proto, 1);
```

```
  if  (!sk)
   goto out;
@@ -675,10 +675,13 @@ char *dn_addr2asc(__u16 addr, char *buf)



-static int dn_create(struct socket *sock, int protocol)
+static int dn_create(struct net *net, struct socket *sock, int protocol)
 {
  struct sock *sk;

+ if (net != &init_net)
+  return -EAFNOSUPPORT;
+
  switch(sock->type) {
   case SOCK_SEQPACKET:
    if (protocol != DNPROTO_NSP)
@@ -691,7 +694,7 @@ static int dn_create(struct socket *sock, int protocol)
  }


- if ((sk = dn_alloc_sock(sock, GFP_KERNEL)) == NULL)
+ if ((sk = dn_alloc_sock(net, sock, GFP_KERNEL)) == NULL)
   return -ENOBUFS;

  sk->sk_protocol = protocol;
@@ -1091,7 +1094,7 @@ static int dn_accept(struct socket *sock, struct socket *newsock, int
flags)

  cb = DN_SKB_CB(skb);
  sk->sk_ack_backlog--;
- newsk = dn_alloc_sock(newsock, sk->sk_allocation);
+ newsk = dn_alloc_sock(sk->sk_net, newsock, sk->sk_allocation);
  if (newsk == NULL) {
   release_sock(sk);
   kfree_skb(skb);
diff --git a/net/econet/af_econet.c b/net/econet/af_econet.c
index 35c96bc..a2429db 100644
--- a/net/econet/af_econet.c
+++ b/net/econet/af_econet.c
@@ -608,12 +608,15 @@ static struct proto econet_proto = {
  * Create an Econet socket
  */

-static int econet_create(struct socket *sock, int protocol)
+static int econet_create(struct net *net, struct socket *sock, int protocol)
 {
  struct sock *sk;
```

```
  struct econet_sock *eo;
  int err;

+ if (net != &init_net)
+  return -EAFNOSUPPORT;
+
  /* Econet only provides datagram services. */
  if (sock->type != SOCK_DGRAM)
   return -ESOCKTNOSUPPORT;
@@ -621,7 +624,7 @@ static int econet_create(struct socket *sock, int protocol)
  sock->state = SS_UNCONNECTED;

  err = -ENOBUFS;
- sk = sk_alloc(PF_ECONET, GFP_KERNEL, &econet_proto, 1);
+ sk = sk_alloc(net, PF_ECONET, GFP_KERNEL, &econet_proto, 1);
  if (sk == NULL)
   goto out;

diff --git a/net/ipv4/af_inet.c b/net/ipv4/af_inet.c
index e681034..110a19e 100644
--- a/net/ipv4/af_inet.c
+++ b/net/ipv4/af_inet.c
@@ -241,7 +241,7 @@ EXPORT_SYMBOL(build_ehash_secret);
  * Create an inet socket.
  */

-static int inet_create(struct socket *sock, int protocol)
+static int inet_create(struct net *net, struct socket *sock, int protocol)
 {
  struct sock *sk;
  struct list_head *p;
@@ -253,6 +253,9 @@ static int inet_create(struct socket *sock, int protocol)
  int try_loading_module = 0;
  int err;

+ if (net != &init_net)
+  return -EAFNOSUPPORT;
+
  if (sock->type != SOCK_RAW &&
      sock->type != SOCK_DGRAM &&
      !inet_ehash_secret)
@@ -320,7 +323,7 @@ lookup_protocol:
  BUG_TRAP(answer_prot->slab != NULL);

  err = -ENOBUFS;
- sk = sk_alloc(PF_INET, GFP_KERNEL, answer_prot, 1);
+ sk = sk_alloc(net, PF_INET, GFP_KERNEL, answer_prot, 1);
  if (sk == NULL)
```

```
   goto out;

diff --git a/net/ipv6/af_inet6.c b/net/ipv6/af_inet6.c
index b5f9637..21931c8 100644
--- a/net/ipv6/af_inet6.c
+++ b/net/ipv6/af_inet6.c
@@ -81,7 +81,7 @@ static __inline__ struct ipv6_pinfo *inet6_sk_generic(struct sock *sk)
  return (struct ipv6_pinfo *)(((u8 *)sk) + offset);
 }

-static int inet6_create(struct socket *sock, int protocol)
+static int inet6_create(struct net *net, struct socket *sock, int protocol)
 {
  struct inet_sock *inet;
  struct ipv6_pinfo *np;
@@ -94,6 +94,9 @@ static int inet6_create(struct socket *sock, int protocol)
  int try_loading_module = 0;
  int err;

+ if (net != &init_net)
+  return -EAFNOSUPPORT;
+
  if (sock->type != SOCK_RAW &&
      sock->type != SOCK_DGRAM &&
      !inet_ehash_secret)
@@ -159,7 +162,7 @@ lookup_protocol:
  BUG_TRAP(answer_prot->slab != NULL);

  err = -ENOBUFS;
- sk = sk_alloc(PF_INET6, GFP_KERNEL, answer_prot, 1);
+ sk = sk_alloc(net, PF_INET6, GFP_KERNEL, answer_prot, 1);
  if (sk == NULL)
   goto out;

diff --git a/net/ipx/af_ipx.c b/net/ipx/af_ipx.c
index 8400525..ee28bab 100644
--- a/net/ipx/af_ipx.c
+++ b/net/ipx/af_ipx.c
@@ -1360,11 +1360,14 @@ static struct proto ipx_proto = {
  .obj_size = sizeof(struct ipx_sock),
 };

-static int ipx_create(struct socket *sock, int protocol)
+static int ipx_create(struct net *net, struct socket *sock, int protocol)
 {
  int rc = -ESOCKTNOSUPPORT;
  struct sock *sk;
```

```
+ if (net != &init_net)
+  return -EAFNOSUPPORT;
+
  /*
   * SPX support is not anymore in the kernel sources. If you want to
   * ressurrect it, completing it and making it understand shared skbs,
@@ -1375,7 +1378,7 @@ static int ipx_create(struct socket *sock, int protocol)
   goto out;

  rc = -ENOMEM;
- sk = sk_alloc(PF_IPX, GFP_KERNEL, &ipx_proto, 1);
+ sk = sk_alloc(net, PF_IPX, GFP_KERNEL, &ipx_proto, 1);
  if (!sk)
   goto out;
 #ifdef IPX_REFCNT_DEBUG
diff --git a/net/irda/af_irda.c b/net/irda/af_irda.c
index c80949a..0328ae2 100644
--- a/net/irda/af_irda.c
+++ b/net/irda/af_irda.c
@@ -60,7 +60,7 @@

 #include <net/irda/af_irda.h>

-static int irda_create(struct socket *sock, int protocol);
+static int irda_create(struct net *net, struct socket *sock, int protocol);

 static const struct proto_ops irda_stream_ops;
 static const struct proto_ops irda_seqpacket_ops;
@@ -831,7 +831,7 @@ static int irda_accept(struct socket *sock, struct socket *newsock, int
flags)

  IRDA_DEBUG(2, "%s()\n", __FUNCTION__);

- err = irda_create(newsock, sk->sk_protocol);
+ err = irda_create(sk->sk_net, newsock, sk->sk_protocol);
  if (err)
   return err;

@@ -1057,13 +1057,16 @@ static struct proto irda_proto = {
  *    Create IrDA socket
  *
  */
-static int irda_create(struct socket *sock, int protocol)
+static int irda_create(struct net *net, struct socket *sock, int protocol)
 {
  struct sock *sk;
  struct irda_sock *self;
```

```
  IRDA_DEBUG(2, "%s()\n", __FUNCTION__);

+ if (net != &init_net)
+  return -EAFNOSUPPORT;
+
  /* Check for valid socket type */
  switch (sock->type) {
  case SOCK_STREAM:     /* For TTP connections with SAR disabled */
@@ -1075,7 +1078,7 @@ static int irda_create(struct socket *sock, int protocol)
  }

  /* Allocate networking socket */
- sk = sk_alloc(PF_IRDA, GFP_ATOMIC, &irda_proto, 1);
+ sk = sk_alloc(net, PF_IRDA, GFP_ATOMIC, &irda_proto, 1);
  if (sk == NULL)
   return -ENOMEM;

diff --git a/net/key/af_key.c b/net/key/af_key.c
index c1a9583..ad9d17f 100644
--- a/net/key/af_key.c
+++ b/net/key/af_key.c
@@ -137,11 +137,14 @@ static struct proto key_proto = {
  .obj_size = sizeof(struct pfkey_sock),
 };

-static int pfkey_create(struct socket *sock, int protocol)
+static int pfkey_create(struct net *net, struct socket *sock, int protocol)
 {
  struct sock *sk;
  int err;

+ if (net != &init_net)
+  return -EAFNOSUPPORT;
+
  if (!capable(CAP_NET_ADMIN))
   return -EPERM;
  if (sock->type != SOCK_RAW)
@@ -150,7 +153,7 @@ static int pfkey_create(struct socket *sock, int protocol)
   return -EPROTONOSUPPORT;

  err = -ENOMEM;
- sk = sk_alloc(PF_KEY, GFP_KERNEL, &key_proto, 1);
+ sk = sk_alloc(net, PF_KEY, GFP_KERNEL, &key_proto, 1);
  if (sk == NULL)
   goto out;

diff --git a/net/llc/af_llc.c b/net/llc/af_llc.c
index 6b8a103..b482441 100644
```

```
--- a/net/llc/af_llc.c
+++ b/net/llc/af_llc.c
@@ -150,14 +150,17 @@ static struct proto llc_proto = {
  * socket type we have available.
  * Returns 0 upon success, negative upon failure.
  */
-static int llc_ui_create(struct socket *sock, int protocol)
+static int llc_ui_create(struct net *net, struct socket *sock, int protocol)
 {
  struct sock *sk;
  int rc = -ESOCKTNOSUPPORT;

+ if (net != &init_net)
+  return -EAFNOSUPPORT;
+
  if (likely(sock->type == SOCK_DGRAM || sock->type == SOCK_STREAM)) {
   rc = -ENOMEM;
-  sk = llc_sk_alloc(PF_LLC, GFP_KERNEL, &llc_proto);
+  sk = llc_sk_alloc(net, PF_LLC, GFP_KERNEL, &llc_proto);
   if (sk) {
    rc = 0;
    llc_ui_sk_init(sock, sk);
diff --git a/net/llc/llc_conn.c b/net/llc/llc_conn.c
index 3b8cfbe..8ebc276 100644
--- a/net/llc/llc_conn.c
+++ b/net/llc/llc_conn.c
@@ -700,7 +700,7 @@ static struct sock *llc_create_incoming_sock(struct sock *sk,
           struct llc_addr *saddr,
           struct llc_addr *daddr)
 {
- struct sock *newsk = llc_sk_alloc(sk->sk_family, GFP_ATOMIC,
+ struct sock *newsk = llc_sk_alloc(sk->sk_net, sk->sk_family, GFP_ATOMIC,
        sk->sk_prot);
  struct llc_sock *newllc, *llc = llc_sk(sk);

@@ -867,9 +867,9 @@ static void llc_sk_init(struct sock* sk)
  * Allocates a LLC sock and initializes it. Returns the new LLC sock
  * or %NULL if there's no memory available for one
  */
-struct sock *llc_sk_alloc(int family, gfp_t priority, struct proto *prot)
+struct sock *llc_sk_alloc(struct net *net, int family, gfp_t priority, struct proto *prot)
 {
- struct sock *sk = sk_alloc(family, priority, prot, 1);
+ struct sock *sk = sk_alloc(net, family, priority, prot, 1);

  if (!sk)
   goto out;
diff --git a/net/netlink/af_netlink.c b/net/netlink/af_netlink.c
```

```
index 3982f13..406a493 100644
--- a/net/netlink/af_netlink.c
+++ b/net/netlink/af_netlink.c
@@ -384,15 +384,15 @@ static struct proto netlink_proto = {
  .obj_size = sizeof(struct netlink_sock),
 };

-static int __netlink_create(struct socket *sock, struct mutex *cb_mutex,
-       int protocol)
+static int __netlink_create(struct net *net, struct socket *sock,
+       struct mutex *cb_mutex, int protocol)
 {
  struct sock *sk;
  struct netlink_sock *nlk;

  sock->ops = &netlink_ops;

- sk = sk_alloc(PF_NETLINK, GFP_KERNEL, &netlink_proto, 1);
+ sk = sk_alloc(net, PF_NETLINK, GFP_KERNEL, &netlink_proto, 1);
  if (!sk)
   return -ENOMEM;

@@ -412,13 +412,16 @@ static int __netlink_create(struct socket *sock, struct mutex *cb_mutex,
  return 0;
 }

-static int netlink_create(struct socket *sock, int protocol)
+static int netlink_create(struct net *net, struct socket *sock, int protocol)
 {
  struct module *module = NULL;
  struct mutex *cb_mutex;
  struct netlink_sock *nlk;
  int err = 0;

+ if (net != &init_net)
+  return -EAFNOSUPPORT;
+
  sock->state = SS_UNCONNECTED;

  if (sock->type != SOCK_RAW && sock->type != SOCK_DGRAM)
@@ -441,7 +444,7 @@ static int netlink_create(struct socket *sock, int protocol)
  cb_mutex = nl_table[protocol].cb_mutex;
  netlink_unlock_table();

- if ((err = __netlink_create(sock, cb_mutex, protocol)) < 0)
+ if ((err = __netlink_create(net, sock, cb_mutex, protocol)) < 0)
   goto out_module;
```

```
  nlk = nlk_sk(sock->sk);
@@ -1318,7 +1321,7 @@ netlink_kernel_create(int unit, unsigned int groups,
  if (sock_create_lite(PF_NETLINK, SOCK_DGRAM, unit, &sock))
   return NULL;

- if (__netlink_create(sock, cb_mutex, unit) < 0)
+ if (__netlink_create(&init_net, sock, cb_mutex, unit) < 0)
   goto out_sock_release;

  if (groups < 32)
diff --git a/net/netrom/af_netrom.c b/net/netrom/af_netrom.c
index 15c8a92..e969d1b 100644
--- a/net/netrom/af_netrom.c
+++ b/net/netrom/af_netrom.c
@@ -409,15 +409,18 @@ static struct proto nr_proto = {
  .obj_size = sizeof(struct nr_sock),
 };

-static int nr_create(struct socket *sock, int protocol)
+static int nr_create(struct net *net, struct socket *sock, int protocol)
 {
  struct sock *sk;
  struct nr_sock *nr;

+ if (net != &init_net)
+  return -EAFNOSUPPORT;
+
  if (sock->type != SOCK_SEQPACKET || protocol != 0)
   return -ESOCKTNOSUPPORT;

- if ((sk = sk_alloc(PF_NETROM, GFP_ATOMIC, &nr_proto, 1)) == NULL)
+ if ((sk = sk_alloc(net, PF_NETROM, GFP_ATOMIC, &nr_proto, 1)) == NULL)
   return -ENOMEM;

  nr = nr_sk(sk);
@@ -459,7 +462,7 @@ static struct sock *nr_make_new(struct sock *osk)
  if (osk->sk_type != SOCK_SEQPACKET)
   return NULL;

- if ((sk = sk_alloc(PF_NETROM, GFP_ATOMIC, osk->sk_prot, 1)) == NULL)
+ if ((sk = sk_alloc(osk->sk_net, PF_NETROM, GFP_ATOMIC, osk->sk_prot, 1)) == NULL)
   return NULL;

  nr = nr_sk(sk);
diff --git a/net/packet/af_packet.c b/net/packet/af_packet.c
index 1eecbd7..72ff099 100644
--- a/net/packet/af_packet.c
+++ b/net/packet/af_packet.c
```

```
@@ -978,13 +978,16 @@ static struct proto packet_proto = {
  * Create a packet of type SOCK_PACKET.
  */

-static int packet_create(struct socket *sock, int protocol)
+static int packet_create(struct net *net, struct socket *sock, int protocol)
 {
  struct sock *sk;
  struct packet_sock *po;
  __be16 proto = (__force __be16)protocol; /* weird, but documented */
  int err;

+ if (net != &init_net)
+  return -EAFNOSUPPORT;
+
  if (!capable(CAP_NET_RAW))
   return -EPERM;
  if (sock->type != SOCK_DGRAM && sock->type != SOCK_RAW &&
@@ -994,7 +997,7 @@ static int packet_create(struct socket *sock, int protocol)
  sock->state = SS_UNCONNECTED;

  err = -ENOBUFS;
- sk = sk_alloc(PF_PACKET, GFP_KERNEL, &packet_proto, 1);
+ sk = sk_alloc(net, PF_PACKET, GFP_KERNEL, &packet_proto, 1);
  if (sk == NULL)
   goto out;

diff --git a/net/rose/af_rose.c b/net/rose/af_rose.c
index 48319f7..67e06ab 100644
--- a/net/rose/af_rose.c
+++ b/net/rose/af_rose.c
@@ -499,15 +499,18 @@ static struct proto rose_proto = {
  .obj_size = sizeof(struct rose_sock),
 };

-static int rose_create(struct socket *sock, int protocol)
+static int rose_create(struct net *net, struct socket *sock, int protocol)
 {
  struct sock *sk;
  struct rose_sock *rose;

+ if (net != &init_net)
+  return -EAFNOSUPPORT;
+
  if (sock->type != SOCK_SEQPACKET || protocol != 0)
   return -ESOCKTNOSUPPORT;

- if ((sk = sk_alloc(PF_ROSE, GFP_ATOMIC, &rose_proto, 1)) == NULL)
```

```
+ if ((sk = sk_alloc(net, PF_ROSE, GFP_ATOMIC, &rose_proto, 1)) == NULL)
   return -ENOMEM;

  rose = rose_sk(sk);
@@ -545,7 +548,7 @@ static struct sock *rose_make_new(struct sock *osk)
  if (osk->sk_type != SOCK_SEQPACKET)
   return NULL;

- if ((sk = sk_alloc(PF_ROSE, GFP_ATOMIC, &rose_proto, 1)) == NULL)
+ if ((sk = sk_alloc(osk->sk_net, PF_ROSE, GFP_ATOMIC, &rose_proto, 1)) == NULL)
   return NULL;

  rose = rose_sk(sk);
diff --git a/net/rxrpc/af_rxrpc.c b/net/rxrpc/af_rxrpc.c
index 122d55d..0803f30 100644
--- a/net/rxrpc/af_rxrpc.c
+++ b/net/rxrpc/af_rxrpc.c
@@ -606,13 +606,16 @@ static unsigned int rxrpc_poll(struct file *file, struct socket *sock,
 /*
  * create an RxRPC socket
  */
-static int rxrpc_create(struct socket *sock, int protocol)
+static int rxrpc_create(struct net *net, struct socket *sock, int protocol)
 {
  struct rxrpc_sock *rx;
  struct sock *sk;

  _enter("%p,%d", sock, protocol);

+ if (net != &init_net)
+  return -EAFNOSUPPORT;
+
  /* we support transport protocol UDP only */
  if (protocol != PF_INET)
   return -EPROTONOSUPPORT;
@@ -623,7 +626,7 @@ static int rxrpc_create(struct socket *sock, int protocol)
  sock->ops = &rxrpc_rpc_ops;
  sock->state = SS_UNCONNECTED;

- sk = sk_alloc(PF_RXRPC, GFP_KERNEL, &rxrpc_proto, 1);
+ sk = sk_alloc(net, PF_RXRPC, GFP_KERNEL, &rxrpc_proto, 1);
  if (!sk)
   return -ENOMEM;

diff --git a/net/sctp/ipv6.c b/net/sctp/ipv6.c
index cd57a51..5953260 100644
--- a/net/sctp/ipv6.c
+++ b/net/sctp/ipv6.c
```

```
@@ -619,7 +619,7 @@ static struct sock *sctp_v6_create_accept_sk(struct sock *sk,
  struct ipv6_pinfo *newnp, *np = inet6_sk(sk);
  struct sctp6_sock *newsctp6sk;

- newsk = sk_alloc(PF_INET6, GFP_KERNEL, sk->sk_prot, 1);
+ newsk = sk_alloc(sk->sk_net, PF_INET6, GFP_KERNEL, sk->sk_prot, 1);
  if (!newsk)
   goto out;

diff --git a/net/sctp/protocol.c b/net/sctp/protocol.c
index a015454..0052ff4 100644
--- a/net/sctp/protocol.c
+++ b/net/sctp/protocol.c
@@ -545,7 +545,7 @@ static struct sock *sctp_v4_create_accept_sk(struct sock *sk,
 {
  struct inet_sock *inet = inet_sk(sk);
  struct inet_sock *newinet;
- struct sock *newsk = sk_alloc(PF_INET, GFP_KERNEL, sk->sk_prot, 1);
+ struct sock *newsk = sk_alloc(sk->sk_net, PF_INET, GFP_KERNEL, sk->sk_prot, 1);

  if (!newsk)
   goto out;
diff --git a/net/socket.c b/net/socket.c
index 7d44453..bf5aba7 100644
--- a/net/socket.c
+++ b/net/socket.c
@@ -84,6 +84,7 @@
 #include <linux/kmod.h>
 #include <linux/audit.h>
 #include <linux/wireless.h>
+#include <linux/nsproxy.h>

 #include <asm/uaccess.h>
 #include <asm/unistd.h>
@@ -1074,7 +1075,7 @@ call_kill:
  return 0;
 }

-static int __sock_create(int family, int type, int protocol,
+static int __sock_create(struct net *net, int family, int type, int protocol,
     struct socket **res, int kern)
 {
  int err;
@@ -1150,7 +1151,7 @@ static int __sock_create(int family, int type, int protocol,
  /* Now protected by module ref count */
  rcu_read_unlock();

- err = pf->create(sock, protocol);
```

```
+ err = pf->create(net, sock, protocol);
  if (err < 0)
   goto out_module_put;

@@ -1189,12 +1190,12 @@ out_release:

 int sock_create(int family, int type, int protocol, struct socket **res)
 {
- return __sock_create(family, type, protocol, res, 0);
+ return __sock_create(current->nsproxy->net_ns, family, type, protocol, res, 0);
 }

 int sock_create_kern(int family, int type, int protocol, struct socket **res)
 {
- return __sock_create(family, type, protocol, res, 1);
+ return __sock_create(&init_net, family, type, protocol, res, 1);
 }

 asmlinkage long sys_socket(int family, int type, int protocol)
diff --git a/net/tipc/socket.c b/net/tipc/socket.c
index 8411017..e36b4b5 100644
--- a/net/tipc/socket.c
+++ b/net/tipc/socket.c
@@ -162,13 +162,16 @@ static void advance_queue(struct tipc_sock *tsock)
  *
  * Returns 0 on success, errno otherwise
  */
-static int tipc_create(struct socket *sock, int protocol)
+static int tipc_create(struct net *net, struct socket *sock, int protocol)
 {
  struct tipc_sock *tsock;
  struct tipc_port *port;
  struct sock *sk;
  u32 ref;

+ if (net != &init_net)
+  return -EAFNOSUPPORT;
+
  if (unlikely(protocol != 0))
   return -EPROTONOSUPPORT;

@@ -198,7 +201,7 @@ static int tipc_create(struct socket *sock, int protocol)
   return -EPROTOTYPE;
  }

- sk = sk_alloc(AF_TIPC, GFP_KERNEL, &tipc_proto, 1);
+ sk = sk_alloc(net, AF_TIPC, GFP_KERNEL, &tipc_proto, 1);
  if (!sk) {
```

```
   tipc_deleteport(ref);
   return -ENOMEM;
@@ -1372,7 +1375,7 @@ static int accept(struct socket *sock, struct socket *newsock, int flags)
  }
  buf = skb_peek(&sock->sk->sk_receive_queue);

- res = tipc_create(newsock, 0);
+ res = tipc_create(sock->sk->sk_net, newsock, 0);
  if (!res) {
   struct tipc_sock *new_tsock = tipc_sk(newsock->sk);
   struct tipc_portid id;
diff --git a/net/unix/af_unix.c b/net/unix/af_unix.c
index 2386090..10e7312 100644
--- a/net/unix/af_unix.c
+++ b/net/unix/af_unix.c
@@ -594,7 +594,7 @@ static struct proto unix_proto = {
  */
 static struct lock_class_key af_unix_sk_receive_queue_lock_key;

-static struct sock * unix_create1(struct socket *sock)
+static struct sock * unix_create1(struct net *net, struct socket *sock)
 {
  struct sock *sk = NULL;
  struct unix_sock *u;
@@ -602,7 +602,7 @@ static struct sock * unix_create1(struct socket *sock)
  if (atomic_read(&unix_nr_socks) >= 2*get_max_files())
   goto out;

- sk = sk_alloc(PF_UNIX, GFP_KERNEL, &unix_proto, 1);
+ sk = sk_alloc(net, PF_UNIX, GFP_KERNEL, &unix_proto, 1);
  if (!sk)
   goto out;

@@ -628,8 +628,11 @@ out:
  return sk;
 }

-static int unix_create(struct socket *sock, int protocol)
+static int unix_create(struct net *net, struct socket *sock, int protocol)
 {
+ if (net != &init_net)
+  return -EAFNOSUPPORT;
+
  if (protocol && protocol != PF_UNIX)
   return -EPROTONOSUPPORT;

@@ -655,7 +658,7 @@ static int unix_create(struct socket *sock, int protocol)
   return -ESOCKTNOSUPPORT;
```

```
  }

- return unix_create1(sock) ? 0 : -ENOMEM;
+ return unix_create1(net, sock) ? 0 : -ENOMEM;
 }

 static int unix_release(struct socket *sock)
@@ -1039,7 +1042,7 @@ static int unix_stream_connect(struct socket *sock, struct sockaddr
*uaddr,
  err = -ENOMEM;

  /* create new sock for complete connection */
- newsk = unix_create1(NULL);
+ newsk = unix_create1(sk->sk_net, NULL);
  if (newsk == NULL)
   goto out;

diff --git a/net/x25/af_x25.c b/net/x25/af_x25.c
index 479927c..2e99315 100644
--- a/net/x25/af_x25.c
+++ b/net/x25/af_x25.c
@@ -466,10 +466,10 @@ static struct proto x25_proto = {
  .obj_size = sizeof(struct x25_sock),
 };

-static struct sock *x25_alloc_socket(void)
+static struct sock *x25_alloc_socket(struct net *net)
 {
  struct x25_sock *x25;
- struct sock *sk = sk_alloc(AF_X25, GFP_ATOMIC, &x25_proto, 1);
+ struct sock *sk = sk_alloc(net, AF_X25, GFP_ATOMIC, &x25_proto, 1);

  if (!sk)
   goto out;
@@ -485,17 +485,20 @@ out:
  return sk;
 }

-static int x25_create(struct socket *sock, int protocol)
+static int x25_create(struct net *net, struct socket *sock, int protocol)
 {
  struct sock *sk;
  struct x25_sock *x25;
  int rc = -ESOCKTNOSUPPORT;

+ if (net != &init_net)
+  return -EAFNOSUPPORT;
+
```

```
  if (sock->type != SOCK_SEQPACKET || protocol)
   goto out;

  rc = -ENOMEM;
- if ((sk = x25_alloc_socket()) == NULL)
+ if ((sk = x25_alloc_socket(net)) == NULL)
   goto out;

  x25 = x25_sk(sk);
@@ -543,7 +546,7 @@ static struct sock *x25_make_new(struct sock *osk)
  if (osk->sk_type != SOCK_SEQPACKET)
   goto out;

- if ((sk = x25_alloc_socket()) == NULL)
+ if ((sk = x25_alloc_socket(osk->sk_net)) == NULL)
   goto out;

  x25 = x25_sk(sk);
--
1.5.3.rc6.17.g1911
```

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: [PATCH 09/16] net: Initialize the network namespace of network devices.
Posted by ebiederm on Sat, 08 Sep 2007 21:24:21 GMT
View Forum Message <> Reply to Message

Except for carefully selected pseudo devices all network
interfaces should start out in the initial network namespace.
Ultimately it will be register_netdev that examines what
dev->nd_net is set to and places a device in a network namespace.

This patch modifies alloc_netdev to initialize the network
namespace a device is in with the initial network namespace.
This gets it right for the vast majority of devices so their
drivers need not be modified and for those few pseudo devices
that need something different they can change this parameter
before calling register_netdevice.

The network namespace parameter on a network device is not
reference counted as the devices are inside of a network namespace
and cannot remain in that namespace past the lifetime of the
network namespace.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 net/core/dev.c | 1 +
 1 files changed, 1 insertions(+), 0 deletions(-)

diff --git a/net/core/dev.c b/net/core/dev.c
index 2ade518..316043b 100644
--- a/net/core/dev.c
+++ b/net/core/dev.c
@@ -3707,6 +3707,7 @@ struct net_device *alloc_netdev_mq(int sizeof_priv, const char *name,
  dev = (struct net_device *)
   (((long)p + NETDEV_ALIGN_CONST) & ~NETDEV_ALIGN_CONST);
  dev->padded = (char *)dev - (char *)p;
+ dev->nd_net = &init_net;

  if (sizeof_priv) {
   dev->priv = ((char *)dev +
--
1.5.3.rc6.17.g1911

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: [PATCH 10/16] net: Make packet reception network namespace safe
Posted by ebiederm on Sat, 08 Sep 2007 21:25:43 GMT
View Forum Message <> Reply to Message

This patch modifies every packet receive function
registered with dev_add_pack() to drop packets if they
are not from the initial network namespace.

This should ensure that the various network stacks do
not receive packets in a anything but the initial network
namespace until the code has been converted and is ready
for them.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 drivers/block/aoe/aoenet.c | 4 ++++
 drivers/net/bonding/bond_3ad.c | 4 ++++
 drivers/net/bonding/bond_alb.c | 3 +++
 drivers/net/bonding/bond_main.c | 3 +++
 drivers/net/hamradio/bpqether.c | 3 +++
 drivers/net/pppoe.c | 6 ++++++
 drivers/net/wan/hdlc.c | 7 +++++++

```
 drivers/net/wan/lapbether.c     |    3 +++
 drivers/net/wan/syncppp.c       |    6 ++++++
 net/8021q/vlan_dev.c            |    5 +++++
 net/appletalk/aarp.c            |    3 +++
 net/appletalk/ddp.c             |    6 ++++++
 net/ax25/ax25_in.c              |    5 +++++
 net/bridge/br_stp_bpdu.c        |    4 ++++
 net/decnet/dn_route.c           |    3 +++
 net/econet/af_econet.c          |    3 +++
 net/ipv4/arp.c                  |    3 +++
 net/ipv4/ip_input.c             |    3 +++
 net/ipv4/ipconfig.c             |    6 ++++++
 net/ipv6/ip6_input.c            |    5 +++++
 net/ipx/af_ipx.c                |    3 +++
 net/irda/irlap_frame.c          |    3 +++
 net/llc/llc_input.c             |    4 ++++
 net/packet/af_packet.c          |    9 +++++++++
 net/tipc/eth_media.c            |    6 ++++++
 net/x25/x25_dev.c               |    3 +++
 26 files changed, 113 insertions(+), 0 deletions(-)

diff --git a/drivers/block/aoe/aoenet.c b/drivers/block/aoe/aoenet.c
index f9ddfda..4dc0fb7 100644
--- a/drivers/block/aoe/aoenet.c
+++ b/drivers/block/aoe/aoenet.c
@@ -8,6 +8,7 @@
 #include <linux/blkdev.h>
 #include <linux/netdevice.h>
 #include <linux/moduleparam.h>
+#include <net/net_namespace.h>
 #include <asm/unaligned.h>
 #include "aoe.h"

@@ -114,6 +115,9 @@ aoenet_rcv(struct sk_buff *skb, struct net_device *ifp, struct packet_type
*pt,
  struct aoe_hdr *h;
  u32 n;

+ if (ifp->nd_net != &init_net)
+  goto exit;
+
  skb = skb_share_check(skb, GFP_ATOMIC);
  if (skb == NULL)
   return 0;
diff --git a/drivers/net/bonding/bond_3ad.c b/drivers/net/bonding/bond_3ad.c
index f829e4a..94bd739 100644
--- a/drivers/net/bonding/bond_3ad.c
+++ b/drivers/net/bonding/bond_3ad.c
```

```
@@ -29,6 +29,7 @@
 #include <linux/ethtool.h>
 #include <linux/if_bonding.h>
 #include <linux/pkt_sched.h>
+#include <net/net_namespace.h>
 #include "bonding.h"
 #include "bond_3ad.h"

@@ -2448,6 +2449,9 @@ int bond_3ad_lacpdu_recv(struct sk_buff *skb, struct net_device *dev,
struct pac
  struct slave *slave = NULL;
  int ret = NET_RX_DROP;

+ if (dev->nd_net != &init_net)
+  goto out;
+
  if (!(dev->flags & IFF_MASTER))
   goto out;

diff --git a/drivers/net/bonding/bond_alb.c b/drivers/net/bonding/bond_alb.c
index 92c3b6f..419a9f8 100644
--- a/drivers/net/bonding/bond_alb.c
+++ b/drivers/net/bonding/bond_alb.c
@@ -345,6 +345,9 @@ static int rlb_arp_recv(struct sk_buff *skb, struct net_device *bond_dev,
struct
  struct arp_pkt *arp = (struct arp_pkt *)skb->data;
  int res = NET_RX_DROP;

+ if (bond_dev->nd_net != &init_net)
+  goto out;
+
  if (!(bond_dev->flags & IFF_MASTER))
   goto out;

diff --git a/drivers/net/bonding/bond_main.c b/drivers/net/bonding/bond_main.c
index 5de648f..e4e5fdc 100644
--- a/drivers/net/bonding/bond_main.c
+++ b/drivers/net/bonding/bond_main.c
@@ -2458,6 +2458,9 @@ static int bond_arp_rcv(struct sk_buff *skb, struct net_device *dev,
struct pack
  unsigned char *arp_ptr;
  u32 sip, tip;

+ if (dev->nd_net != &init_net)
+  goto out;
+
  if (!(dev->priv_flags & IFF_BONDING) || !(dev->flags & IFF_MASTER))
   goto out;
```

```
diff --git a/drivers/net/hamradio/bpqether.c b/drivers/net/hamradio/bpqether.c
index 1699d42..85fb8e7 100644
--- a/drivers/net/hamradio/bpqether.c
+++ b/drivers/net/hamradio/bpqether.c
@@ -173,6 +173,9 @@ static int bpq_rcv(struct sk_buff *skb, struct net_device *dev, struct
packet_ty
  struct ethhdr *eth;
  struct bpqdev *bpq;

+ if (dev->nd_net != &init_net)
+  goto drop;
+
  if ((skb = skb_share_check(skb, GFP_ATOMIC)) == NULL)
   return NET_RX_DROP;

diff --git a/drivers/net/pppoe.c b/drivers/net/pppoe.c
index f8bf5fc..a29ea22 100644
--- a/drivers/net/pppoe.c
+++ b/drivers/net/pppoe.c
@@ -386,6 +386,9 @@ static int pppoe_rcv(struct sk_buff *skb,
  struct pppoe_hdr *ph;
  struct pppox_sock *po;

+ if (dev->nd_net != &init_net)
+  goto drop;
+
  if (!pskb_may_pull(skb, sizeof(struct pppoe_hdr)))
   goto drop;

@@ -418,6 +421,9 @@ static int pppoe_disc_rcv(struct sk_buff *skb,
  struct pppoe_hdr *ph;
  struct pppox_sock *po;

+ if (dev->nd_net != &init_net)
+  goto abort;
+
  if (!pskb_may_pull(skb, sizeof(struct pppoe_hdr)))
   goto abort;

diff --git a/drivers/net/wan/hdlc.c b/drivers/net/wan/hdlc.c
index 65ad2e2..3b57350 100644
--- a/drivers/net/wan/hdlc.c
+++ b/drivers/net/wan/hdlc.c
@@ -36,6 +36,7 @@
 #include <linux/rtnetlink.h>
 #include <linux/notifier.h>
 #include <linux/hdlc.h>
```

```
+#include <net/net_namespace.h>


 static const char* version = "HDLC support module revision 1.21";
@@ -66,6 +67,12 @@ static int hdlc_rcv(struct sk_buff *skb, struct net_device *dev,
       struct packet_type *p, struct net_device *orig_dev)
 {
  struct hdlc_device_desc *desc = dev_to_desc(dev);
+
+ if (dev->nd_net != &init_net) {
+  kfree_skb(skb);
+  return 0;
+ }
+
  if (desc->netif_rx)
   return desc->netif_rx(skb);

diff --git a/drivers/net/wan/lapbether.c b/drivers/net/wan/lapbether.c
index 6c302e9..ca8b3c3 100644
--- a/drivers/net/wan/lapbether.c
+++ b/drivers/net/wan/lapbether.c
@@ -91,6 +91,9 @@ static int lapbeth_rcv(struct sk_buff *skb, struct net_device *dev, struct
packe
  int len, err;
  struct lapbethdev *lapbeth;

+ if (dev->nd_net != &init_net)
+  goto drop;
+
  if ((skb = skb_share_check(skb, GFP_ATOMIC)) == NULL)
   return NET_RX_DROP;

diff --git a/drivers/net/wan/syncppp.c b/drivers/net/wan/syncppp.c
index 67fc67c..5c71af6 100644
--- a/drivers/net/wan/syncppp.c
+++ b/drivers/net/wan/syncppp.c
@@ -51,6 +51,7 @@
 #include <linux/spinlock.h>
 #include <linux/rcupdate.h>

+#include <net/net_namespace.h>
 #include <net/syncppp.h>

 #include <asm/byteorder.h>
@@ -1445,6 +1446,11 @@ static void sppp_print_bytes (u_char *p, u16 len)

 static int sppp_rcv(struct sk_buff *skb, struct net_device *dev, struct packet_type *p, struct
net_device *orig_dev)
```

```
 {
+ if (dev->nd_net != &init_net) {
+  kfree_skb(skb);
+  return 0;
+ }
+
  if ((skb = skb_share_check(skb, GFP_ATOMIC)) == NULL)
   return NET_RX_DROP;
  sppp_input(dev,skb);
diff --git a/net/8021q/vlan_dev.c b/net/8021q/vlan_dev.c
index 328759c..6644e8f 100644
--- a/net/8021q/vlan_dev.c
+++ b/net/8021q/vlan_dev.c
@@ -122,6 +122,11 @@ int vlan_skb_recv(struct sk_buff *skb, struct net_device *dev,
  unsigned short vlan_TCI;
  __be16 proto;

+ if (dev->nd_net != &init_net) {
+  kfree_skb(skb);
+  return -1;
+ }
+
  if ((skb = skb_share_check(skb, GFP_ATOMIC)) == NULL)
   return -1;

diff --git a/net/appletalk/aarp.c b/net/appletalk/aarp.c
index 80b5414..9267f48 100644
--- a/net/appletalk/aarp.c
+++ b/net/appletalk/aarp.c
@@ -713,6 +713,9 @@ static int aarp_rcv(struct sk_buff *skb, struct net_device *dev,
  struct atalk_addr sa, *ma, da;
  struct atalk_iface *ifa;

+ if (dev->nd_net != &init_net)
+  goto out0;
+
  /* We only do Ethernet SNAP AARP. */
  if (dev->type != ARPHRD_ETHER)
   goto out0;
diff --git a/net/appletalk/ddp.c b/net/appletalk/ddp.c
index fd1d52f..c1f1367 100644
--- a/net/appletalk/ddp.c
+++ b/net/appletalk/ddp.c
@@ -1403,6 +1403,9 @@ static int atalk_rcv(struct sk_buff *skb, struct net_device *dev,
  int origlen;
  __u16 len_hops;

+ if (dev->nd_net != &init_net)
```

```
+  goto freeit;
+
   /* Don't mangle buffer if shared */
   if (!(skb = skb_share_check(skb, GFP_ATOMIC)))
    goto out;
@@ -1488,6 +1491,9 @@ freeit:
 static int ltalk_rcv(struct sk_buff *skb, struct net_device *dev,
        struct packet_type *pt, struct net_device *orig_dev)
 {
+ if (dev->nd_net != &init_net)
+  goto freeit;
+
   /* Expand any short form frames */
   if (skb_mac_header(skb)[2] == 1) {
    struct ddpehdr *ddp;
diff --git a/net/ax25/ax25_in.c b/net/ax25/ax25_in.c
index 0ddaff0..3b7d172 100644
--- a/net/ax25/ax25_in.c
+++ b/net/ax25/ax25_in.c
@@ -451,6 +451,11 @@ int ax25_kiss_rcv(struct sk_buff *skb, struct net_device *dev,
   skb->sk = NULL;  /* Initially we don't know who it's for */
   skb->destructor = NULL; /* Who initializes this, dammit?! */

+ if (dev->nd_net != &init_net) {
+  kfree_skb(skb);
+  return 0;
+ }
+
   if ((*skb->data & 0x0F) != 0) {
    kfree_skb(skb); /* Not a KISS data frame */
    return 0;
diff --git a/net/bridge/br_stp_bpdu.c b/net/bridge/br_stp_bpdu.c
index 14f0c88..0edbd2a 100644
--- a/net/bridge/br_stp_bpdu.c
+++ b/net/bridge/br_stp_bpdu.c
@@ -17,6 +17,7 @@
 #include <linux/netfilter_bridge.h>
 #include <linux/etherdevice.h>
 #include <linux/llc.h>
+#include <net/net_namespace.h>
 #include <net/llc.h>
 #include <net/llc_pdu.h>
 #include <asm/unaligned.h>
@@ -141,6 +142,9 @@ int br_stp_rcv(struct sk_buff *skb, struct net_device *dev,
   struct net_bridge *br;
   const unsigned char *buf;

+ if (dev->nd_net != &init_net)
```

```
+  goto err;
+
  if (!p)
   goto err;

diff --git a/net/decnet/dn_route.c b/net/decnet/dn_route.c
index 4cfea95..580e786 100644
--- a/net/decnet/dn_route.c
+++ b/net/decnet/dn_route.c
@@ -584,6 +584,9 @@ int dn_route_rcv(struct sk_buff *skb, struct net_device *dev, struct
packet_type
  struct dn_dev *dn = (struct dn_dev *)dev->dn_ptr;
  unsigned char padlen = 0;

+ if (dev->nd_net != &init_net)
+  goto dump_it;
+
  if (dn == NULL)
   goto dump_it;

diff --git a/net/econet/af_econet.c b/net/econet/af_econet.c
index a2429db..7de3006 100644
--- a/net/econet/af_econet.c
+++ b/net/econet/af_econet.c
@@ -1065,6 +1065,9 @@ static int econet_rcv(struct sk_buff *skb, struct net_device *dev, struct
packet
  struct sock *sk;
  struct ec_device *edev = dev->ec_ptr;

+ if (dev->nd_net != &init_net)
+  goto drop;
+
  if (skb->pkt_type == PACKET_OTHERHOST)
   goto drop;

diff --git a/net/ipv4/arp.c b/net/ipv4/arp.c
index 78dd344..bde1297 100644
--- a/net/ipv4/arp.c
+++ b/net/ipv4/arp.c
@@ -932,6 +932,9 @@ static int arp_rcv(struct sk_buff *skb, struct net_device *dev,
 {
  struct arphdr *arp;

+ if (dev->nd_net != &init_net)
+  goto freeskb;
+
  /* ARP header, plus 2 device addresses, plus 2 IP addresses.  */
  if (!pskb_may_pull(skb, (sizeof(struct arphdr) +
```

```
      (2 * dev->addr_len) +
diff --git a/net/ipv4/ip_input.c b/net/ipv4/ip_input.c
index 9706939..41d8964 100644
--- a/net/ipv4/ip_input.c
+++ b/net/ipv4/ip_input.c
@@ -382,6 +382,9 @@ int ip_rcv(struct sk_buff *skb, struct net_device *dev, struct packet_type
*pt,
  struct iphdr *iph;
  u32 len;

+ if (dev->nd_net != &init_net)
+  goto drop;
+
  /* When the interface is in promisc. mode, drop all the crap
   * that it receives, do not try to analyse it.
   */
diff --git a/net/ipv4/ipconfig.c b/net/ipv4/ipconfig.c
index 5ae4849..08ff623 100644
--- a/net/ipv4/ipconfig.c
+++ b/net/ipv4/ipconfig.c
@@ -426,6 +426,9 @@ ic_rarp_recv(struct sk_buff *skb, struct net_device *dev, struct
packet_type *pt
  unsigned char *sha, *tha;  /* s for "source", t for "target" */
  struct ic_device *d;

+ if (dev->nd_net != &init_net)
+  goto drop;
+
  if ((skb = skb_share_check(skb, GFP_ATOMIC)) == NULL)
   return NET_RX_DROP;

@@ -835,6 +838,9 @@ static int __init ic_bootp_recv(struct sk_buff *skb, struct net_device *dev,
str
  struct ic_device *d;
  int len, ext_len;

+ if (dev->nd_net != &init_net)
+  goto drop;
+
  /* Perform verifications before taking the lock.  */
  if (skb->pkt_type == PACKET_OTHERHOST)
   goto drop;
diff --git a/net/ipv6/ip6_input.c b/net/ipv6/ip6_input.c
index 30a5cb1..7d18cac 100644
--- a/net/ipv6/ip6_input.c
+++ b/net/ipv6/ip6_input.c
@@ -61,6 +61,11 @@ int ipv6_rcv(struct sk_buff *skb, struct net_device *dev, struct packet_type
*pt
```

```
  u32   pkt_len;
  struct inet6_dev *idev;

+ if (dev->nd_net != &init_net) {
+  kfree_skb(skb);
+  return 0;
+ }
+
  if (skb->pkt_type == PACKET_OTHERHOST) {
   kfree_skb(skb);
   return 0;
diff --git a/net/ipx/af_ipx.c b/net/ipx/af_ipx.c
index ee28bab..f7b4d38 100644
--- a/net/ipx/af_ipx.c
+++ b/net/ipx/af_ipx.c
@@ -1647,6 +1647,9 @@ static int ipx_rcv(struct sk_buff *skb, struct net_device *dev, struct
packet_ty
  u16 ipx_pktsize;
  int rc = 0;

+ if (dev->nd_net != &init_net)
+  goto drop;
+
  /* Not ours */
  if (skb->pkt_type == PACKET_OTHERHOST)
   goto drop;
diff --git a/net/irda/irlap_frame.c b/net/irda/irlap_frame.c
index 25a3444..77ac27e 100644
--- a/net/irda/irlap_frame.c
+++ b/net/irda/irlap_frame.c
@@ -1326,6 +1326,9 @@ int irlap_driver_rcv(struct sk_buff *skb, struct net_device *dev,
  int command;
  __u8 control;

+ if (dev->nd_net != &init_net)
+  goto out;
+
  /* FIXME: should we get our own field? */
  self = (struct irlap_cb *) dev->atalk_ptr;

diff --git a/net/llc/llc_input.c b/net/llc/llc_input.c
index 099ed8f..c40c9b2 100644
--- a/net/llc/llc_input.c
+++ b/net/llc/llc_input.c
@@ -12,6 +12,7 @@
  * See the GNU General Public License for more details.
  */
 #include <linux/netdevice.h>
```

```
+#include <net/net_namespace.h>
 #include <net/llc.h>
 #include <net/llc_pdu.h>
 #include <net/llc_sap.h>
@@ -145,6 +146,9 @@ int llc_rcv(struct sk_buff *skb, struct net_device *dev,
  int (*rcv)(struct sk_buff *, struct net_device *,
      struct packet_type *, struct net_device *);

+ if (dev->nd_net != &init_net)
+  goto drop;
+
  /*
   * When the interface is in promisc. mode, drop all the crap that it
   * receives, do not try to analyse it.
diff --git a/net/packet/af_packet.c b/net/packet/af_packet.c
index 72ff099..51b0d5c 100644
--- a/net/packet/af_packet.c
+++ b/net/packet/af_packet.c
@@ -252,6 +252,9 @@ static int packet_rcv_spkt(struct sk_buff *skb, struct net_device *dev,
struct
  struct sock *sk;
  struct sockaddr_pkt *spkt;

+ if (dev->nd_net != &init_net)
+  goto out;
+
  /*
   * When we registered the protocol we saved the socket in the data
   * field for just this event.
@@ -452,6 +455,9 @@ static int packet_rcv(struct sk_buff *skb, struct net_device *dev, struct
packet
  int skb_len = skb->len;
  unsigned int snaplen, res;

+ if (dev->nd_net != &init_net)
+  goto drop;
+
  if (skb->pkt_type == PACKET_LOOPBACK)
   goto drop;

@@ -568,6 +574,9 @@ static int tpacket_rcv(struct sk_buff *skb, struct net_device *dev, struct
packe
  struct sk_buff *copy_skb = NULL;
  struct timeval tv;

+ if (dev->nd_net != &init_net)
+  goto drop;
+
```

```
  if (skb->pkt_type == PACKET_LOOPBACK)
   goto drop;

diff --git a/net/tipc/eth_media.c b/net/tipc/eth_media.c
index 711ca4b..d2ed237 100644
--- a/net/tipc/eth_media.c
+++ b/net/tipc/eth_media.c
@@ -38,6 +38,7 @@
 #include <net/tipc/tipc_bearer.h>
 #include <net/tipc/tipc_msg.h>
 #include <linux/netdevice.h>
+#include <net/net_namespace.h>

 #define MAX_ETH_BEARERS  2
 #define ETH_LINK_PRIORITY TIPC_DEF_LINK_PRI
@@ -100,6 +101,11 @@ static int recv_msg(struct sk_buff *buf, struct net_device *dev,
  struct eth_bearer *eb_ptr = (struct eth_bearer *)pt->af_packet_priv;
  u32 size;

+ if (dev->nd_net != &init_net) {
+  kfree_skb(buf);
+  return 0;
+ }
+
  if (likely(eb_ptr->bearer)) {
   if (likely(buf->pkt_type <= PACKET_BROADCAST)) {
    size = msg_size((struct tipc_msg *)buf->data);
diff --git a/net/x25/x25_dev.c b/net/x25/x25_dev.c
index 848a6b6..f0679d2 100644
--- a/net/x25/x25_dev.c
+++ b/net/x25/x25_dev.c
@@ -95,6 +95,9 @@ int x25_lapb_receive_frame(struct sk_buff *skb, struct net_device *dev,
  struct sk_buff *nskb;
  struct x25_neigh *nb;

+ if (dev->nd_net != &init_net)
+  goto drop;
+
  nskb = skb_copy(skb, GFP_ATOMIC);
  if (!nskb)
   goto drop;
--
1.5.3.rc6.17.g1911
```

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: [PATCH 11/16] net: Make device event notification network namespace safe
Posted by ebiederm on Sat, 08 Sep 2007 21:27:11 GMT
View Forum Message <> Reply to Message

Every user of the network device notifiers is either a protocol
stack or a pseudo device. If a protocol stack that does not have
support for multiple network namespaces receives an event for a
device that is not in the initial network namespace it quite possibly
can get confused and do the wrong thing.

To avoid problems until all of the protocol stacks are converted
this patch modifies all netdev event handlers to ignore events on
devices that are not in the initial network namespace.

As the rest of the code is made network namespace aware these
checks can be removed.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 arch/ia64/hp/sim/simeth.c | 3 +++
 drivers/net/bonding/bond_main.c | 3 +++
 drivers/net/hamradio/bpqether.c | 3 +++
 drivers/net/pppoe.c | 3 +++
 drivers/net/wan/dlci.c | 3 +++
 drivers/net/wan/hdlc.c | 3 +++
 drivers/net/wan/lapbether.c | 3 +++
 net/8021q/vlan.c | 4 ++++
 net/appletalk/aarp.c | 3 +++
 net/appletalk/ddp.c | 3 +++
 net/atm/clip.c | 3 +++
 net/atm/mpc.c | 4 ++++
 net/ax25/af_ax25.c | 3 +++
 net/bridge/br_notify.c | 4 ++++
 net/core/dst.c | 4 ++++
 net/core/fib_rules.c | 4 ++++
 net/core/pktgen.c | 3 +++
 net/core/rtnetlink.c | 4 ++++
 net/decnet/af_decnet.c | 3 +++
 net/econet/af_econet.c | 3 +++
 net/ipv4/arp.c | 3 +++
 net/ipv4/devinet.c | 3 +++
 net/ipv4/fib_frontend.c | 3 +++
 net/ipv4/ipmr.c | 7 ++++++-
 net/ipv4/netfilter/ip_queue.c | 3 +++
 net/ipv4/netfilter/ipt_MASQUERADE.c | 3 +++
 net/ipv6/addrconf.c | 3 +++
 net/ipv6/ndisc.c | 3 +++
 net/ipv6/netfilter/ip6_queue.c | 3 +++

```
 net/ipx/af_ipx.c                 |    3 +++
 net/netfilter/nfnetlink_queue.c  |    3 +++
 net/netrom/af_netrom.c           |    3 +++
 net/packet/af_packet.c           |    3 +++
 net/rose/af_rose.c               |    3 +++
 net/tipc/eth_media.c             |    3 +++
 net/x25/af_x25.c                 |    3 +++
 net/xfrm/xfrm_policy.c           |    5 +++++
 security/selinux/netif.c         |    4 ++++
 38 files changed, 126 insertions(+), 1 deletions(-)

diff --git a/arch/ia64/hp/sim/simeth.c b/arch/ia64/hp/sim/simeth.c
index f26077a..93d6004 100644
--- a/arch/ia64/hp/sim/simeth.c
+++ b/arch/ia64/hp/sim/simeth.c
@@ -300,6 +300,9 @@ simeth_device_event(struct notifier_block *this,unsigned long event, void
*ptr)
   return NOTIFY_DONE;
  }

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  if ( event != NETDEV_UP && event != NETDEV_DOWN ) return NOTIFY_DONE;

  /*
diff --git a/drivers/net/bonding/bond_main.c b/drivers/net/bonding/bond_main.c
index e4e5fdc..cf97d8a 100644
--- a/drivers/net/bonding/bond_main.c
+++ b/drivers/net/bonding/bond_main.c
@@ -3299,6 +3299,9 @@ static int bond_netdev_event(struct notifier_block *this, unsigned long
event, v
 {
  struct net_device *event_dev = (struct net_device *)ptr;

+ if (event_dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  dprintk("event_dev: %s, event: %lx\n",
   (event_dev ? event_dev->name : "None"),
   event);
diff --git a/drivers/net/hamradio/bpqether.c b/drivers/net/hamradio/bpqether.c
index 85fb8e7..df09210 100644
--- a/drivers/net/hamradio/bpqether.c
+++ b/drivers/net/hamradio/bpqether.c
@@ -563,6 +563,9 @@ static int bpq_device_event(struct notifier_block *this,unsigned long
event, voi
 {
```

```
  struct net_device *dev = (struct net_device *)ptr;

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  if (!dev_is_ethdev(dev))
   return NOTIFY_DONE;

diff --git a/drivers/net/pppoe.c b/drivers/net/pppoe.c
index a29ea22..b4ba584 100644
--- a/drivers/net/pppoe.c
+++ b/drivers/net/pppoe.c
@@ -301,6 +301,9 @@ static int pppoe_device_event(struct notifier_block *this,
 {
  struct net_device *dev = (struct net_device *) ptr;

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  /* Only look at sockets that are using this specific device. */
  switch (event) {
  case NETDEV_CHANGEMTU:
diff --git a/drivers/net/wan/dlci.c b/drivers/net/wan/dlci.c
index 66be20c..61041d5 100644
--- a/drivers/net/wan/dlci.c
+++ b/drivers/net/wan/dlci.c
@@ -513,6 +513,9 @@ static int dlci_dev_event(struct notifier_block *unused,
 {
  struct net_device *dev = (struct net_device *) ptr;

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  if (event == NETDEV_UNREGISTER) {
   struct dlci_local *dlp;

diff --git a/drivers/net/wan/hdlc.c b/drivers/net/wan/hdlc.c
index 3b57350..ee23b91 100644
--- a/drivers/net/wan/hdlc.c
+++ b/drivers/net/wan/hdlc.c
@@ -109,6 +109,9 @@ static int hdlc_device_event(struct notifier_block *this, unsigned long
event,
  unsigned long flags;
  int on;

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
```

```
  if (dev->get_stats != hdlc_get_stats)
   return NOTIFY_DONE; /* not an HDLC device */

diff --git a/drivers/net/wan/lapbether.c b/drivers/net/wan/lapbether.c
index ca8b3c3..699b934 100644
--- a/drivers/net/wan/lapbether.c
+++ b/drivers/net/wan/lapbether.c
@@ -394,6 +394,9 @@ static int lapbeth_device_event(struct notifier_block *this,
  struct lapbethdev *lapbeth;
  struct net_device *dev = ptr;

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  if (!dev_is_ethdev(dev))
   return NOTIFY_DONE;

diff --git a/net/8021q/vlan.c b/net/8021q/vlan.c
index 1583c5e..55f2dbb 100644
--- a/net/8021q/vlan.c
+++ b/net/8021q/vlan.c
@@ -31,6 +31,7 @@
 #include <net/arp.h>
 #include <linux/rtnetlink.h>
 #include <linux/notifier.h>
+#include <net/net_namespace.h>

 #include <linux/if_vlan.h>
 #include "vlan.h"
@@ -605,6 +606,9 @@ static int vlan_device_event(struct notifier_block *unused, unsigned long
event,
  int i, flgs;
  struct net_device *vlandev;

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  if (!grp)
   goto out;

diff --git a/net/appletalk/aarp.c b/net/appletalk/aarp.c
index 9267f48..e9a51a6 100644
--- a/net/appletalk/aarp.c
+++ b/net/appletalk/aarp.c
@@ -333,6 +333,9 @@ static int aarp_device_event(struct notifier_block *this, unsigned long
event,
  struct net_device *dev = ptr;
  int ct;
```

```
+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  if (event == NETDEV_DOWN) {
   write_lock_bh(&aarp_lock);

diff --git a/net/appletalk/ddp.c b/net/appletalk/ddp.c
index c1f1367..36fcdbf 100644
--- a/net/appletalk/ddp.c
+++ b/net/appletalk/ddp.c
@@ -649,6 +649,9 @@ static int ddp_device_event(struct notifier_block *this, unsigned long
event,
 {
  struct net_device *dev = ptr;

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  if (event == NETDEV_DOWN)
   /* Discard any use of this */
   atalk_dev_down(dev);
diff --git a/net/atm/clip.c b/net/atm/clip.c
index 806ea98..741742f 100644
--- a/net/atm/clip.c
+++ b/net/atm/clip.c
@@ -612,6 +612,9 @@ static int clip_device_event(struct notifier_block *this, unsigned long
event,
 {
  struct net_device *dev = arg;

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  if (event == NETDEV_UNREGISTER) {
   neigh_ifdown(&clip_tbl, dev);
   return NOTIFY_DONE;
diff --git a/net/atm/mpc.c b/net/atm/mpc.c
index 7c85aa5..0968430 100644
--- a/net/atm/mpc.c
+++ b/net/atm/mpc.c
@@ -956,6 +956,10 @@ static int mpoa_event_listener(struct notifier_block *mpoa_notifier,
unsigned lo
  struct lec_priv *priv;

  dev = (struct net_device *)dev_ptr;
+
+ if (dev->nd_net != &init_net)
```

```
+  return NOTIFY_DONE;
+
  if (dev->name == NULL || strncmp(dev->name, "lec", 3))
   return NOTIFY_DONE; /* we are only interested in lec:s */

diff --git a/net/ax25/af_ax25.c b/net/ax25/af_ax25.c
index def6c42..8d13a8b 100644
--- a/net/ax25/af_ax25.c
+++ b/net/ax25/af_ax25.c
@@ -104,6 +104,9 @@ static int ax25_device_event(struct notifier_block *this, unsigned long
event,
 {
  struct net_device *dev = (struct net_device *)ptr;

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  /* Reject non AX.25 devices */
  if (dev->type != ARPHRD_AX25)
   return NOTIFY_DONE;
diff --git a/net/bridge/br_notify.c b/net/bridge/br_notify.c
index c8451d3..07ac3ae 100644
--- a/net/bridge/br_notify.c
+++ b/net/bridge/br_notify.c
@@ -15,6 +15,7 @@

 #include <linux/kernel.h>
 #include <linux/rtnetlink.h>
+#include <net/net_namespace.h>

 #include "br_private.h"

@@ -36,6 +37,9 @@ static int br_device_event(struct notifier_block *unused, unsigned long
event, v
  struct net_bridge_port *p = dev->br_port;
  struct net_bridge *br;

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  /* not a port of a bridge */
  if (p == NULL)
   return NOTIFY_DONE;
diff --git a/net/core/dst.c b/net/core/dst.c
index c6a0587..32267a1 100644
--- a/net/core/dst.c
+++ b/net/core/dst.c
@@ -15,6 +15,7 @@
```

```
 #include <linux/skbuff.h>
 #include <linux/string.h>
 #include <linux/types.h>
+#include <net/net_namespace.h>

 #include <net/dst.h>

@@ -252,6 +253,9 @@ static int dst_dev_event(struct notifier_block *this, unsigned long event,
void
  struct net_device *dev = ptr;
  struct dst_entry *dst;

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  switch (event) {
  case NETDEV_UNREGISTER:
  case NETDEV_DOWN:
diff --git a/net/core/fib_rules.c b/net/core/fib_rules.c
index 8c5474e..9eabe1a 100644
--- a/net/core/fib_rules.c
+++ b/net/core/fib_rules.c
@@ -11,6 +11,7 @@
 #include <linux/types.h>
 #include <linux/kernel.h>
 #include <linux/list.h>
+#include <net/net_namespace.h>
 #include <net/fib_rules.h>

 static LIST_HEAD(rules_ops);
@@ -596,6 +597,9 @@ static int fib_rules_event(struct notifier_block *this, unsigned long event,
  struct net_device *dev = ptr;
  struct fib_rules_ops *ops;

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  ASSERT_RTNL();
  rcu_read_lock();

diff --git a/net/core/pktgen.c b/net/core/pktgen.c
index fa15cc7..21ae955 100644
--- a/net/core/pktgen.c
+++ b/net/core/pktgen.c
@@ -1966,6 +1966,9 @@ static int pktgen_device_event(struct notifier_block *unused,
 {
  struct net_device *dev = ptr;
```

```
+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  /* It is OK that we do not hold the group lock right now,
   * as we run under the RTNL lock.
   */
diff --git a/net/core/rtnetlink.c b/net/core/rtnetlink.c
index dca9e80..4185950 100644
--- a/net/core/rtnetlink.c
+++ b/net/core/rtnetlink.c
@@ -1286,6 +1286,10 @@ static void rtnetlink_rcv(struct sock *sk, int len)
 static int rtnetlink_event(struct notifier_block *this, unsigned long event, void *ptr)
 {
  struct net_device *dev = ptr;
+
+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  switch (event) {
  case NETDEV_UNREGISTER:
   rtmsg_ifinfo(RTM_DELLINK, dev, ~0U);
diff --git a/net/decnet/af_decnet.c b/net/decnet/af_decnet.c
index aca4c49..83398da 100644
--- a/net/decnet/af_decnet.c
+++ b/net/decnet/af_decnet.c
@@ -2089,6 +2089,9 @@ static int dn_device_event(struct notifier_block *this, unsigned long
event,
 {
  struct net_device *dev = (struct net_device *)ptr;

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  switch(event) {
   case NETDEV_UP:
    dn_dev_up(dev);
diff --git a/net/econet/af_econet.c b/net/econet/af_econet.c
index 7de3006..f877f3b 100644
--- a/net/econet/af_econet.c
+++ b/net/econet/af_econet.c
@@ -1122,6 +1122,9 @@ static int econet_notifier(struct notifier_block *this, unsigned long msg,
void
  struct net_device *dev = (struct net_device *)data;
  struct ec_device *edev;

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
```

```
  switch (msg) {
  case NETDEV_UNREGISTER:
   /* A device has gone down - kill any data we hold for it. */
diff --git a/net/ipv4/arp.c b/net/ipv4/arp.c
index bde1297..a11e7a5 100644
--- a/net/ipv4/arp.c
+++ b/net/ipv4/arp.c
@@ -1205,6 +1205,9 @@ static int arp_netdev_event(struct notifier_block *this, unsigned long
event, vo
 {
  struct net_device *dev = ptr;

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  switch (event) {
  case NETDEV_CHANGEADDR:
   neigh_changeaddr(&arp_tbl, dev);
diff --git a/net/ipv4/devinet.c b/net/ipv4/devinet.c
index 5b77bda..9996b57 100644
--- a/net/ipv4/devinet.c
+++ b/net/ipv4/devinet.c
@@ -1051,6 +1051,9 @@ static int inetdev_event(struct notifier_block *this, unsigned long event,
  struct net_device *dev = ptr;
  struct in_device *in_dev = __in_dev_get_rtnl(dev);

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  ASSERT_RTNL();

  if (!in_dev) {
diff --git a/net/ipv4/fib_frontend.c b/net/ipv4/fib_frontend.c
index eff6bce..cefb55e 100644
--- a/net/ipv4/fib_frontend.c
+++ b/net/ipv4/fib_frontend.c
@@ -860,6 +860,9 @@ static int fib_netdev_event(struct notifier_block *this, unsigned long
event, vo
  struct net_device *dev = ptr;
  struct in_device *in_dev = __in_dev_get_rtnl(dev);

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  if (event == NETDEV_UNREGISTER) {
   fib_disable_ip(dev, 2);
   return NOTIFY_DONE;
diff --git a/net/ipv4/ipmr.c b/net/ipv4/ipmr.c
```

```
index 35683e1..0365988 100644
--- a/net/ipv4/ipmr.c
+++ b/net/ipv4/ipmr.c
@@ -1083,13 +1083,18 @@ int ipmr_ioctl(struct sock *sk, int cmd, void __user *arg)

 static int ipmr_device_event(struct notifier_block *this, unsigned long event, void *ptr)
 {
+ struct net_device *dev = ptr;
  struct vif_device *v;
  int ct;
+
+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  if (event != NETDEV_UNREGISTER)
   return NOTIFY_DONE;
  v=&vif_table[0];
  for (ct=0;ct<maxvif;ct++,v++) {
-  if (v->dev==ptr)
+  if (v->dev==dev)
    vif_delete(ct);
  }
  return NOTIFY_DONE;
diff --git a/net/ipv4/netfilter/ip_queue.c b/net/ipv4/netfilter/ip_queue.c
index cb5e61a..d918560 100644
--- a/net/ipv4/netfilter/ip_queue.c
+++ b/net/ipv4/netfilter/ip_queue.c
@@ -557,6 +557,9 @@ ipq_rcv_dev_event(struct notifier_block *this,
 {
  struct net_device *dev = ptr;

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  /* Drop any packets associated with the downed device */
  if (event == NETDEV_DOWN)
   ipq_dev_drop(dev->ifindex);
diff --git a/net/ipv4/netfilter/ipt_MASQUERADE.c b/net/ipv4/netfilter/ipt_MASQUERADE.c
index 7c4e4be..3e0b562 100644
--- a/net/ipv4/netfilter/ipt_MASQUERADE.c
+++ b/net/ipv4/netfilter/ipt_MASQUERADE.c
@@ -125,6 +125,9 @@ static int masq_device_event(struct notifier_block *this,
 {
  const struct net_device *dev = ptr;

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
```

```
  if (event == NETDEV_DOWN) {
   /* Device was downed.  Search entire table for
      conntracks which were associated with that device,
diff --git a/net/ipv6/addrconf.c b/net/ipv6/addrconf.c
index 5e62ee5..5b191a3 100644
--- a/net/ipv6/addrconf.c
+++ b/net/ipv6/addrconf.c
@@ -2259,6 +2259,9 @@ static int addrconf_notify(struct notifier_block *this, unsigned long
event,
  int run_pending = 0;
  int err;

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  switch(event) {
  case NETDEV_REGISTER:
   if (!idev && dev->mtu >= IPV6_MIN_MTU) {
diff --git a/net/ipv6/ndisc.c b/net/ipv6/ndisc.c
index 0358e60..4d03d35 100644
--- a/net/ipv6/ndisc.c
+++ b/net/ipv6/ndisc.c
@@ -1524,6 +1524,9 @@ static int ndisc_netdev_event(struct notifier_block *this, unsigned long
event,
 {
  struct net_device *dev = ptr;

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  switch (event) {
  case NETDEV_CHANGEADDR:
   neigh_changeaddr(&nd_tbl, dev);
diff --git a/net/ipv6/netfilter/ip6_queue.c b/net/ipv6/netfilter/ip6_queue.c
index dfc58fb..64536a3 100644
--- a/net/ipv6/netfilter/ip6_queue.c
+++ b/net/ipv6/netfilter/ip6_queue.c
@@ -547,6 +547,9 @@ ipq_rcv_dev_event(struct notifier_block *this,
 {
  struct net_device *dev = ptr;

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  /* Drop any packets associated with the downed device */
  if (event == NETDEV_DOWN)
   ipq_dev_drop(dev->ifindex);
diff --git a/net/ipx/af_ipx.c b/net/ipx/af_ipx.c
```

```
index f7b4d38..24921f1 100644
--- a/net/ipx/af_ipx.c
+++ b/net/ipx/af_ipx.c
@@ -347,6 +347,9 @@ static int ipxitf_device_event(struct notifier_block *notifier,
  struct net_device *dev = ptr;
  struct ipx_interface *i, *tmp;

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  if (event != NETDEV_DOWN && event != NETDEV_UP)
   goto out;

diff --git a/net/netfilter/nfnetlink_queue.c b/net/netfilter/nfnetlink_queue.c
index bb65a38..5a8e8ff 100644
--- a/net/netfilter/nfnetlink_queue.c
+++ b/net/netfilter/nfnetlink_queue.c
@@ -734,6 +734,9 @@ nfqnl_rcv_dev_event(struct notifier_block *this,
 {
  struct net_device *dev = ptr;

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  /* Drop any packets associated with the downed device */
  if (event == NETDEV_DOWN)
   nfqnl_dev_drop(dev->ifindex);
diff --git a/net/netrom/af_netrom.c b/net/netrom/af_netrom.c
index e969d1b..3a4d479 100644
--- a/net/netrom/af_netrom.c
+++ b/net/netrom/af_netrom.c
@@ -106,6 +106,9 @@ static int nr_device_event(struct notifier_block *this, unsigned long event,
voi
 {
  struct net_device *dev = (struct net_device *)ptr;

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  if (event != NETDEV_DOWN)
   return NOTIFY_DONE;

diff --git a/net/packet/af_packet.c b/net/packet/af_packet.c
index 51b0d5c..30f2143 100644
--- a/net/packet/af_packet.c
+++ b/net/packet/af_packet.c
@@ -1478,6 +1478,9 @@ static int packet_notifier(struct notifier_block *this, unsigned long msg,
void
```

```
  struct hlist_node *node;
  struct net_device *dev = data;

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  read_lock(&packet_sklist_lock);
  sk_for_each(sk, node, &packet_sklist) {
   struct packet_sock *po = pkt_sk(sk);
diff --git a/net/rose/af_rose.c b/net/rose/af_rose.c
index 67e06ab..509defe 100644
--- a/net/rose/af_rose.c
+++ b/net/rose/af_rose.c
@@ -197,6 +197,9 @@ static int rose_device_event(struct notifier_block *this, unsigned long
event,
 {
  struct net_device *dev = (struct net_device *)ptr;

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  if (event != NETDEV_DOWN)
   return NOTIFY_DONE;

diff --git a/net/tipc/eth_media.c b/net/tipc/eth_media.c
index d2ed237..406f0d2 100644
--- a/net/tipc/eth_media.c
+++ b/net/tipc/eth_media.c
@@ -198,6 +198,9 @@ static int recv_notification(struct notifier_block *nb, unsigned long evt,
  struct eth_bearer *eb_ptr = &eth_bearers[0];
  struct eth_bearer *stop = &eth_bearers[MAX_ETH_BEARERS];

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
  while ((eb_ptr->dev != dev)) {
   if (++eb_ptr == stop)
    return NOTIFY_DONE; /* couldn't find device */
diff --git a/net/x25/af_x25.c b/net/x25/af_x25.c
index 2e99315..fc416f9 100644
--- a/net/x25/af_x25.c
+++ b/net/x25/af_x25.c
@@ -191,6 +191,9 @@ static int x25_device_event(struct notifier_block *this, unsigned long
event,
  struct net_device *dev = ptr;
  struct x25_neigh *nb;

+ if (dev->nd_net != &init_net)
```

```
+  return NOTIFY_DONE;
+
   if (dev->type == ARPHRD_X25
  #if defined(CONFIG_LLC) || defined(CONFIG_LLC_MODULE)
    || dev->type == ARPHRD_ETHER
diff --git a/net/xfrm/xfrm_policy.c b/net/xfrm/xfrm_policy.c
index 6ab81b1..bc316bc 100644
--- a/net/xfrm/xfrm_policy.c
+++ b/net/xfrm/xfrm_policy.c
@@ -2358,6 +2358,11 @@ static void xfrm_policy_unlock_afinfo(struct xfrm_policy_afinfo
*afinfo)

 static int xfrm_dev_event(struct notifier_block *this, unsigned long event, void *ptr)
 {
+ struct net_device *dev = ptr;
+
+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
   switch (event) {
   case NETDEV_DOWN:
    xfrm_flush_bundles();
diff --git a/security/selinux/netif.c b/security/selinux/netif.c
index b10c34e..e87ab94 100644
--- a/security/selinux/netif.c
+++ b/security/selinux/netif.c
@@ -20,6 +20,7 @@
 #include <linux/notifier.h>
 #include <linux/netdevice.h>
 #include <linux/rcupdate.h>
+#include <net/net_namespace.h>

 #include "security.h"
 #include "objsec.h"
@@ -234,6 +235,9 @@ static int sel_netif_netdev_notifier_handler(struct notifier_block *this,
 {
   struct net_device *dev = ptr;

+ if (dev->nd_net != &init_net)
+  return NOTIFY_DONE;
+
   if (event == NETDEV_DOWN)
    sel_netif_kill(dev);

--
1.5.3.rc6.17.g1911
```

Subject: [PATCH 12/16] net: Support multiple network namespaces with netlink
Posted by ebiederm on Sat, 08 Sep 2007 21:28:27 GMT
View Forum Message <> Reply to Message

Each netlink socket will live in exactly one network namespace,
this includes the controlling kernel sockets.

This patch updates all of the existing netlink protocols
to only support the initial network namespace. Request
by clients in other namespaces will get -ECONREFUSED.
As they would if the kernel did not have the support for
that netlink protocol compiled in.

As each netlink protocol is updated to be multiple network
namespace safe it can register multiple kernel sockets
to acquire a presence in the rest of the network namespaces.

The implementation in af_netlink is a simple filter implementation
at hash table insertion and hash table look up time.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
drivers/connector/connector.c | 2 +-
drivers/scsi/scsi_netlink.c | 2 +-
drivers/scsi/scsi_transport_iscsi.c | 2 +-
fs/ecryptfs/netlink.c | 2 +-
include/linux/netlink.h | 6 ++-
kernel/audit.c | 4 +-
lib/kobject_uevent.c | 5 +-
net/bridge/netfilter/ebt_ulog.c | 5 +-
net/core/rtnetlink.c | 4 +-
net/decnet/netfilter/dn_rtmsg.c | 3 +-
net/ipv4/fib_frontend.c | 4 +-
net/ipv4/inet_diag.c | 4 +-
net/ipv4/netfilter/ip_queue.c | 6 +-
net/ipv4/netfilter/ipt_ULOG.c | 3 +-
net/ipv6/netfilter/ip6_queue.c | 6 +-
net/netfilter/nfnetlink.c | 2 +-
net/netfilter/nfnetlink_log.c | 3 +-
net/netfilter/nfnetlink_queue.c | 3 +-
net/netlink/af_netlink.c | 106 ++++++++++++++++++++++++++++++++---------
net/netlink/genetlink.c | 4 +-
net/xfrm/xfrm_user.c | 2 +-

```
 security/selinux/netlink.c         |    5 +-
 22 files changed, 122 insertions(+), 61 deletions(-)

diff --git a/drivers/connector/connector.c b/drivers/connector/connector.c
index a7b9e9b..5690709 100644
--- a/drivers/connector/connector.c
+++ b/drivers/connector/connector.c
@@ -446,7 +446,7 @@ static int __devinit cn_init(void)
  dev->id.idx = cn_idx;
  dev->id.val = cn_val;

- dev->nls = netlink_kernel_create(NETLINK_CONNECTOR,
+ dev->nls = netlink_kernel_create(&init_net, NETLINK_CONNECTOR,
       CN_NETLINK_USERS + 0xf,
       dev->input, NULL, THIS_MODULE);
  if (!dev->nls)
diff --git a/drivers/scsi/scsi_netlink.c b/drivers/scsi/scsi_netlink.c
index 4bf9aa5..163acf6 100644
--- a/drivers/scsi/scsi_netlink.c
+++ b/drivers/scsi/scsi_netlink.c
@@ -167,7 +167,7 @@ scsi_netlink_init(void)
   return;
  }

- scsi_nl_sock = netlink_kernel_create(NETLINK_SCSITRANSPORT,
+ scsi_nl_sock = netlink_kernel_create(&init_net, NETLINK_SCSITRANSPORT,
     SCSI_NL_GRP_CNT, scsi_nl_rcv, NULL,
     THIS_MODULE);
  if (!scsi_nl_sock) {
diff --git a/drivers/scsi/scsi_transport_iscsi.c b/drivers/scsi/scsi_transport_iscsi.c
index 34c1860..4916f01 100644
--- a/drivers/scsi/scsi_transport_iscsi.c
+++ b/drivers/scsi/scsi_transport_iscsi.c
@@ -1523,7 +1523,7 @@ static __init int iscsi_transport_init(void)
  if (err)
   goto unregister_conn_class;

- nls = netlink_kernel_create(NETLINK_ISCSI, 1, iscsi_if_rx, NULL,
+ nls = netlink_kernel_create(&init_net, NETLINK_ISCSI, 1, iscsi_if_rx, NULL,
    THIS_MODULE);
  if (!nls) {
   err = -ENOBUFS;
diff --git a/fs/ecryptfs/netlink.c b/fs/ecryptfs/netlink.c
index fe91863..056519c 100644
--- a/fs/ecryptfs/netlink.c
+++ b/fs/ecryptfs/netlink.c
@@ -227,7 +227,7 @@ int ecryptfs_init_netlink(void)
 {
```

```
  int rc;

- ecryptfs_nl_sock = netlink_kernel_create(NETLINK_ECRYPTFS, 0,
+ ecryptfs_nl_sock = netlink_kernel_create(&init_net, NETLINK_ECRYPTFS, 0,
        ecryptfs_receive_nl_message,
        NULL, THIS_MODULE);
  if (!ecryptfs_nl_sock) {
diff --git a/include/linux/netlink.h b/include/linux/netlink.h
index 83d8239..d2843ae 100644
--- a/include/linux/netlink.h
+++ b/include/linux/netlink.h
@@ -27,6 +27,8 @@

 #define MAX_LINKS 32

+struct net;
+
 struct sockaddr_nl
 {
  sa_family_t nl_family; /* AF_NETLINK */
@@ -157,7 +159,8 @@ struct netlink_skb_parms
 #define NETLINK_CREDS(skb) (&NETLINK_CB((skb)).creds)


-extern struct sock *netlink_kernel_create(int unit, unsigned int groups,
+extern struct sock *netlink_kernel_create(struct net *net,
+       int unit,unsigned int groups,
        void (*input)(struct sock *sk, int len),
        struct mutex *cb_mutex,
        struct module *module);
@@ -206,6 +209,7 @@ struct netlink_callback

 struct netlink_notify
 {
+ struct net *net;
  int pid;
  int protocol;
 };
diff --git a/kernel/audit.c b/kernel/audit.c
index eb0f916..f3c390f 100644
--- a/kernel/audit.c
+++ b/kernel/audit.c
@@ -876,8 +876,8 @@ static int __init audit_init(void)

  printk(KERN_INFO "audit: initializing netlink socket (%s)\n",
         audit_default ? "enabled" : "disabled");
- audit_sock = netlink_kernel_create(NETLINK_AUDIT, 0, audit_receive,
-        NULL, THIS_MODULE);
```

```
+ audit_sock = netlink_kernel_create(&init_net, NETLINK_AUDIT, 0,
+        audit_receive, NULL, THIS_MODULE);
  if (!audit_sock)
   audit_panic("cannot initialize netlink socket");
  else
diff --git a/lib/kobject_uevent.c b/lib/kobject_uevent.c
index df02814..e06a8dc 100644
--- a/lib/kobject_uevent.c
+++ b/lib/kobject_uevent.c
@@ -280,9 +280,8 @@ EXPORT_SYMBOL_GPL(add_uevent_var);
 #if defined(CONFIG_NET)
 static int __init kobject_uevent_init(void)
 {
- uevent_sock = netlink_kernel_create(NETLINK_KOBJECT_UEVENT, 1, NULL,
-         NULL, THIS_MODULE);
-
+ uevent_sock = netlink_kernel_create(&init_net, NETLINK_KOBJECT_UEVENT,
+         1, NULL, NULL, THIS_MODULE);
  if (!uevent_sock) {
   printk(KERN_ERR
          "kobject_uevent: unable to create netlink socket!\n");
diff --git a/net/bridge/netfilter/ebt_ulog.c b/net/bridge/netfilter/ebt_ulog.c
index 204c968..e7cfd30 100644
--- a/net/bridge/netfilter/ebt_ulog.c
+++ b/net/bridge/netfilter/ebt_ulog.c
@@ -300,8 +300,9 @@ static int __init ebt_ulog_init(void)
   spin_lock_init(&ulog_buffers[i].lock);
  }

- ebtulognl = netlink_kernel_create(NETLINK_NFLOG, EBT_ULOG_MAXNLGROUPS,
-       NULL, NULL, THIS_MODULE);
+ ebtulognl = netlink_kernel_create(&init_net, NETLINK_NFLOG,
+       EBT_ULOG_MAXNLGROUPS, NULL, NULL,
+       THIS_MODULE);
  if (!ebtulognl)
   ret = -ENOMEM;
  else if ((ret = ebt_register_watcher(&ulog)))
diff --git a/net/core/rtnetlink.c b/net/core/rtnetlink.c
index 4185950..416768d 100644
--- a/net/core/rtnetlink.c
+++ b/net/core/rtnetlink.c
@@ -1327,8 +1327,8 @@ void __init rtnetlink_init(void)
  if (!rta_buf)
   panic("rtnetlink_init: cannot allocate rta_buf\n");

- rtnl = netlink_kernel_create(NETLINK_ROUTE, RTNLGRP_MAX, rtnetlink_rcv,
-         &rtnl_mutex, THIS_MODULE);
+ rtnl = netlink_kernel_create(&init_net, NETLINK_ROUTE, RTNLGRP_MAX,
```

```
+         rtnetlink_rcv, &rtnl_mutex, THIS_MODULE);
  if (rtnl == NULL)
   panic("rtnetlink_init: cannot initialize rtnetlink\n");
  netlink_set_nonroot(NETLINK_ROUTE, NL_NONROOT_RECV);
diff --git a/net/decnet/netfilter/dn_rtmsg.c b/net/decnet/netfilter/dn_rtmsg.c
index 6962346..ebb38fe 100644
--- a/net/decnet/netfilter/dn_rtmsg.c
+++ b/net/decnet/netfilter/dn_rtmsg.c
@@ -137,7 +137,8 @@ static int __init dn_rtmsg_init(void)
 {
  int rv = 0;

- dnrmg = netlink_kernel_create(NETLINK_DNRTMSG, DNRNG_NLGRP_MAX,
+ dnrmg = netlink_kernel_create(&init_net,
+          NETLINK_DNRTMSG, DNRNG_NLGRP_MAX,
           dnrmg_receive_user_sk, NULL, THIS_MODULE);
  if (dnrmg == NULL) {
   printk(KERN_ERR "dn_rtmsg: Cannot create netlink socket");
diff --git a/net/ipv4/fib_frontend.c b/net/ipv4/fib_frontend.c
index cefb55e..140bf7a 100644
--- a/net/ipv4/fib_frontend.c
+++ b/net/ipv4/fib_frontend.c
@@ -816,8 +816,8 @@ static void nl_fib_input(struct sock *sk, int len)

 static void nl_fib_lookup_init(void)
 {
-      netlink_kernel_create(NETLINK_FIB_LOOKUP, 0, nl_fib_input, NULL,
-       THIS_MODULE);
+ netlink_kernel_create(&init_net, NETLINK_FIB_LOOKUP, 0, nl_fib_input,
+    NULL, THIS_MODULE);
 }

 static void fib_disable_ip(struct net_device *dev, int force)
diff --git a/net/ipv4/inet_diag.c b/net/ipv4/inet_diag.c
index 06922da..169026d 100644
--- a/net/ipv4/inet_diag.c
+++ b/net/ipv4/inet_diag.c
@@ -893,8 +893,8 @@ static int __init inet_diag_init(void)
  if (!inet_diag_table)
   goto out;

- idiagnl = netlink_kernel_create(NETLINK_INET_DIAG, 0, inet_diag_rcv,
-     NULL, THIS_MODULE);
+ idiagnl = netlink_kernel_create(&init_net, NETLINK_INET_DIAG, 0,
+     inet_diag_rcv, NULL, THIS_MODULE);
  if (idiagnl == NULL)
   goto out_free_table;
  err = 0;
```

```
diff --git a/net/ipv4/netfilter/ip_queue.c b/net/ipv4/netfilter/ip_queue.c
index d918560..82fda92 100644
--- a/net/ipv4/netfilter/ip_queue.c
+++ b/net/ipv4/netfilter/ip_queue.c
@@ -579,7 +579,7 @@ ipq_rcv_nl_event(struct notifier_block *this,
  if (event == NETLINK_URELEASE &&
      n->protocol == NETLINK_FIREWALL && n->pid) {
   write_lock_bh(&queue_lock);
-  if (n->pid == peer_pid)
+  if ((n->net == &init_net) && (n->pid == peer_pid))
    __ipq_reset();
   write_unlock_bh(&queue_lock);
  }
@@ -671,8 +671,8 @@ static int __init ip_queue_init(void)
  struct proc_dir_entry *proc;

  netlink_register_notifier(&ipq_nl_notifier);
- ipqnl = netlink_kernel_create(NETLINK_FIREWALL, 0, ipq_rcv_sk,
-          NULL, THIS_MODULE);
+ ipqnl = netlink_kernel_create(&init_net, NETLINK_FIREWALL, 0,
+          ipq_rcv_sk, NULL, THIS_MODULE);
  if (ipqnl == NULL) {
   printk(KERN_ERR "ip_queue: failed to create netlink socket\n");
   goto cleanup_netlink_notifier;
diff --git a/net/ipv4/netfilter/ipt_ULOG.c b/net/ipv4/netfilter/ipt_ULOG.c
index 6ca43e4..c636d6d 100644
--- a/net/ipv4/netfilter/ipt_ULOG.c
+++ b/net/ipv4/netfilter/ipt_ULOG.c
@@ -409,7 +409,8 @@ static int __init ipt_ulog_init(void)
  for (i = 0; i < ULOG_MAXNLGROUPS; i++)
   setup_timer(&ulog_buffers[i].timer, ulog_timer, i);

- nflognl = netlink_kernel_create(NETLINK_NFLOG, ULOG_MAXNLGROUPS, NULL,
+ nflognl = netlink_kernel_create(&init_net,
+     NETLINK_NFLOG, ULOG_MAXNLGROUPS, NULL,
      NULL, THIS_MODULE);
  if (!nflognl)
   return -ENOMEM;
diff --git a/net/ipv6/netfilter/ip6_queue.c b/net/ipv6/netfilter/ip6_queue.c
index 64536a3..2f5a524 100644
--- a/net/ipv6/netfilter/ip6_queue.c
+++ b/net/ipv6/netfilter/ip6_queue.c
@@ -569,7 +569,7 @@ ipq_rcv_nl_event(struct notifier_block *this,
  if (event == NETLINK_URELEASE &&
      n->protocol == NETLINK_IP6_FW && n->pid) {
   write_lock_bh(&queue_lock);
-  if (n->pid == peer_pid)
+  if ((n->net == &init_net) && (n->pid == peer_pid))
```

```
    __ipq_reset();
   write_unlock_bh(&queue_lock);
  }
@@ -661,8 +661,8 @@ static int __init ip6_queue_init(void)
  struct proc_dir_entry *proc;

  netlink_register_notifier(&ipq_nl_notifier);
- ipqnl = netlink_kernel_create(NETLINK_IP6_FW, 0, ipq_rcv_sk, NULL,
-          THIS_MODULE);
+ ipqnl = netlink_kernel_create(&init_net, NETLINK_IP6_FW, 0, ipq_rcv_sk,
+      NULL, THIS_MODULE);
  if (ipqnl == NULL) {
   printk(KERN_ERR "ip6_queue: failed to create netlink socket\n");
   goto cleanup_netlink_notifier;
diff --git a/net/netfilter/nfnetlink.c b/net/netfilter/nfnetlink.c
index 8797e69..fa974e8 100644
--- a/net/netfilter/nfnetlink.c
+++ b/net/netfilter/nfnetlink.c
@@ -264,7 +264,7 @@ static int __init nfnetlink_init(void)
 {
  printk("Netfilter messages via NETLINK v%s.\n", nfversion);

- nfnl = netlink_kernel_create(NETLINK_NETFILTER, NFNLGRP_MAX,
+ nfnl = netlink_kernel_create(&init_net, NETLINK_NETFILTER, NFNLGRP_MAX,
          nfnetlink_rcv, NULL, THIS_MODULE);
  if (!nfnl) {
   printk(KERN_ERR "cannot initialize nfnetlink!\n");
diff --git a/net/netfilter/nfnetlink_log.c b/net/netfilter/nfnetlink_log.c
index e185a5b..994fff0 100644
--- a/net/netfilter/nfnetlink_log.c
+++ b/net/netfilter/nfnetlink_log.c
@@ -705,7 +705,8 @@ nfulnl_rcv_nl_event(struct notifier_block *this,

    hlist_for_each_entry_safe(inst, tmp, t2, head, hlist) {
     UDEBUG("node = %p\n", inst);
-    if (n->pid == inst->peer_pid)
+    if ((n->net == &init_net) &&
+        (n->pid == inst->peer_pid))
      __instance_destroy(inst);
    }
   }
diff --git a/net/netfilter/nfnetlink_queue.c b/net/netfilter/nfnetlink_queue.c
index 5a8e8ff..c97369f 100644
--- a/net/netfilter/nfnetlink_queue.c
+++ b/net/netfilter/nfnetlink_queue.c
@@ -765,7 +765,8 @@ nfqnl_rcv_nl_event(struct notifier_block *this,
    struct hlist_head *head = &instance_table[i];
```

```
    hlist_for_each_entry_safe(inst, tmp, t2, head, hlist) {
-    if (n->pid == inst->peer_pid)
+    if ((n->net == &init_net) &&
+        (n->pid == inst->peer_pid))
      __instance_destroy(inst);
    }
   }
diff --git a/net/netlink/af_netlink.c b/net/netlink/af_netlink.c
index 406a493..6726957 100644
--- a/net/netlink/af_netlink.c
+++ b/net/netlink/af_netlink.c
@@ -211,7 +211,7 @@ netlink_unlock_table(void)
   wake_up(&nl_table_wait);
 }

-static __inline__ struct sock *netlink_lookup(int protocol, u32 pid)
+static __inline__ struct sock *netlink_lookup(struct net *net, int protocol, u32 pid)
 {
  struct nl_pid_hash *hash = &nl_table[protocol].hash;
  struct hlist_head *head;
@@ -221,7 +221,7 @@ static __inline__ struct sock *netlink_lookup(int protocol, u32 pid)
  read_lock(&nl_table_lock);
  head = nl_pid_hashfn(hash, pid);
  sk_for_each(sk, node, head) {
-  if (nlk_sk(sk)->pid == pid) {
+  if ((sk->sk_net == net) && (nlk_sk(sk)->pid == pid)) {
    sock_hold(sk);
    goto found;
   }
@@ -328,7 +328,7 @@ netlink_update_listeners(struct sock *sk)
   * makes sure updates are visible before bind or setsockopt return. */
 }

-static int netlink_insert(struct sock *sk, u32 pid)
+static int netlink_insert(struct sock *sk, struct net *net, u32 pid)
 {
  struct nl_pid_hash *hash = &nl_table[sk->sk_protocol].hash;
  struct hlist_head *head;
@@ -341,7 +341,7 @@ static int netlink_insert(struct sock *sk, u32 pid)
  head = nl_pid_hashfn(hash, pid);
  len = 0;
  sk_for_each(osk, node, head) {
-  if (nlk_sk(osk)->pid == pid)
+  if ((osk->sk_net == net) && (nlk_sk(osk)->pid == pid))
    break;
   len++;
  }
@@ -419,9 +419,6 @@ static int netlink_create(struct net *net, struct socket *sock, int protocol)
```

```
  struct netlink_sock *nlk;
  int err = 0;

- if (net != &init_net)
-  return -EAFNOSUPPORT;
-
  sock->state = SS_UNCONNECTED;

  if (sock->type != SOCK_RAW && sock->type != SOCK_DGRAM)
@@ -481,6 +478,7 @@ static int netlink_release(struct socket *sock)

  if (nlk->pid && !nlk->subscriptions) {
   struct netlink_notify n = {
+      .net = sk->sk_net,
       .protocol = sk->sk_protocol,
       .pid = nlk->pid,
       };
@@ -509,6 +507,7 @@ static int netlink_release(struct socket *sock)
 static int netlink_autobind(struct socket *sock)
 {
  struct sock *sk = sock->sk;
+ struct net *net = sk->sk_net;
  struct nl_pid_hash *hash = &nl_table[sk->sk_protocol].hash;
  struct hlist_head *head;
  struct sock *osk;
@@ -522,6 +521,8 @@ retry:
  netlink_table_grab();
  head = nl_pid_hashfn(hash, pid);
  sk_for_each(osk, node, head) {
+  if ((osk->sk_net != net))
+   continue;
   if (nlk_sk(osk)->pid == pid) {
    /* Bind collision, search negative pid values. */
    pid = rover--;
@@ -533,7 +534,7 @@ retry:
  }
  netlink_table_ungrab();

- err = netlink_insert(sk, pid);
+ err = netlink_insert(sk, net, pid);
  if (err == -EADDRINUSE)
   goto retry;

@@ -598,6 +599,7 @@ static int netlink_realloc_groups(struct sock *sk)
 static int netlink_bind(struct socket *sock, struct sockaddr *addr, int addr_len)
 {
  struct sock *sk = sock->sk;
+ struct net *net = sk->sk_net;
```

```
  struct netlink_sock *nlk = nlk_sk(sk);
  struct sockaddr_nl *nladdr = (struct sockaddr_nl *)addr;
  int err;
@@ -619,7 +621,7 @@ static int netlink_bind(struct socket *sock, struct sockaddr *addr, int
addr_len
    return -EINVAL;
  } else {
   err = nladdr->nl_pid ?
-   netlink_insert(sk, nladdr->nl_pid) :
+   netlink_insert(sk, net, nladdr->nl_pid) :
    netlink_autobind(sock);
   if (err)
    return err;
@@ -703,10 +705,12 @@ static void netlink_overrun(struct sock *sk)
 static struct sock *netlink_getsockbypid(struct sock *ssk, u32 pid)
 {
  int protocol = ssk->sk_protocol;
+ struct net *net;
  struct sock *sock;
  struct netlink_sock *nlk;

- sock = netlink_lookup(protocol, pid);
+ net = ssk->sk_net;
+ sock = netlink_lookup(net, protocol, pid);
  if (!sock)
   return ERR_PTR(-ECONNREFUSED);

@@ -887,6 +891,7 @@ static __inline__ int netlink_broadcast_deliver(struct sock *sk, struct
sk_buff

 struct netlink_broadcast_data {
  struct sock *exclude_sk;
+ struct net *net;
  u32 pid;
  u32 group;
  int failure;
@@ -909,6 +914,9 @@ static inline int do_one_broadcast(struct sock *sk,
      !test_bit(p->group - 1, nlk->groups))
   goto out;

+ if ((sk->sk_net != p->net))
+  goto out;
+
  if (p->failure) {
   netlink_overrun(sk);
   goto out;
@@ -947,6 +955,7 @@ out:
 int netlink_broadcast(struct sock *ssk, struct sk_buff *skb, u32 pid,
```

```
        u32 group, gfp_t allocation)
 {
+ struct net *net = ssk->sk_net;
  struct netlink_broadcast_data info;
  struct hlist_node *node;
  struct sock *sk;
@@ -954,6 +963,7 @@ int netlink_broadcast(struct sock *ssk, struct sk_buff *skb, u32 pid,
  skb = netlink_trim(skb, allocation);

  info.exclude_sk = ssk;
+ info.net = net;
  info.pid = pid;
  info.group = group;
  info.failure = 0;
@@ -1002,6 +1012,9 @@ static inline int do_one_set_err(struct sock *sk,
  if (sk == p->exclude_sk)
   goto out;

+ if (sk->sk_net != p->exclude_sk->sk_net)
+  goto out;
+
  if (nlk->pid == p->pid || p->group - 1 >= nlk->ngroups ||
      !test_bit(p->group - 1, nlk->groups))
   goto out;
@@ -1304,7 +1317,7 @@ static void netlink_data_ready(struct sock *sk, int len)
  */

 struct sock *
-netlink_kernel_create(int unit, unsigned int groups,
+netlink_kernel_create(struct net *net, int unit, unsigned int groups,
        void (*input)(struct sock *sk, int len),
        struct mutex *cb_mutex, struct module *module)
 {
@@ -1321,7 +1334,7 @@ netlink_kernel_create(int unit, unsigned int groups,
  if (sock_create_lite(PF_NETLINK, SOCK_DGRAM, unit, &sock))
   return NULL;

- if (__netlink_create(&init_net, sock, cb_mutex, unit) < 0)
+ if (__netlink_create(net, sock, cb_mutex, unit) < 0)
   goto out_sock_release;

  if (groups < 32)
@@ -1336,18 +1349,20 @@ netlink_kernel_create(int unit, unsigned int groups,
  if (input)
   nlk_sk(sk)->data_ready = input;

- if (netlink_insert(sk, 0))
+ if (netlink_insert(sk, net, 0))
```

```
   goto out_sock_release;

  nlk = nlk_sk(sk);
  nlk->flags |= NETLINK_KERNEL_SOCKET;

  netlink_table_grab();
- nl_table[unit].groups = groups;
- nl_table[unit].listeners = listeners;
- nl_table[unit].cb_mutex = cb_mutex;
- nl_table[unit].module = module;
- nl_table[unit].registered = 1;
+ if (!nl_table[unit].registered) {
+  nl_table[unit].groups = groups;
+  nl_table[unit].listeners = listeners;
+  nl_table[unit].cb_mutex = cb_mutex;
+  nl_table[unit].module = module;
+  nl_table[unit].registered = 1;
+ }
  netlink_table_ungrab();

  return sk;
@@ -1513,7 +1528,7 @@ int netlink_dump_start(struct sock *ssk, struct sk_buff *skb,
  atomic_inc(&skb->users);
  cb->skb = skb;

- sk = netlink_lookup(ssk->sk_protocol, NETLINK_CB(skb).pid);
+ sk = netlink_lookup(ssk->sk_net, ssk->sk_protocol, NETLINK_CB(skb).pid);
  if (sk == NULL) {
   netlink_destroy_callback(cb);
   return -ECONNREFUSED;
@@ -1555,7 +1570,8 @@ void netlink_ack(struct sk_buff *in_skb, struct nlmsghdr *nlh, int err)
  if (!skb) {
   struct sock *sk;

-  sk = netlink_lookup(in_skb->sk->sk_protocol,
+  sk = netlink_lookup(in_skb->sk->sk_net,
+        in_skb->sk->sk_protocol,
         NETLINK_CB(in_skb).pid);
   if (sk) {
    sk->sk_err = ENOBUFS;
@@ -1706,6 +1722,7 @@ int nlmsg_notify(struct sock *sk, struct sk_buff *skb, u32 pid,

 #ifdef CONFIG_PROC_FS
 struct nl_seq_iter {
+ struct net *net;
  int link;
  int hash_idx;
 };
```

```
@@ -1723,6 +1740,8 @@ static struct sock *netlink_seq_socket_idx(struct seq_file *seq, loff_t
pos)

   for (j = 0; j <= hash->mask; j++) {
    sk_for_each(s, node, &hash->table[j]) {
+    if (iter->net != s->sk_net)
+     continue;
     if (off == pos) {
      iter->link = i;
      iter->hash_idx = j;
@@ -1752,11 +1771,14 @@ static void *netlink_seq_next(struct seq_file *seq, void *v, loff_t
*pos)
  if (v == SEQ_START_TOKEN)
   return netlink_seq_socket_idx(seq, 0);

- s = sk_next(v);
+ iter = seq->private;
+ s = v;
+ do {
+  s = sk_next(s);
+ } while (s && (iter->net != s->sk_net));
  if (s)
   return s;

- iter = seq->private;
  i = iter->link;
  j = iter->hash_idx + 1;

@@ -1765,6 +1787,8 @@ static void *netlink_seq_next(struct seq_file *seq, void *v, loff_t *pos)

   for (; j <= hash->mask; j++) {
    s = sk_head(&hash->table[j]);
+   while (s && (iter->net != s->sk_net))
+    s = sk_next(s);
    if (s) {
     iter->link = i;
     iter->hash_idx = j;
@@ -1835,15 +1859,24 @@ static int netlink_seq_open(struct inode *inode, struct file *file)

  seq = file->private_data;
  seq->private = iter;
+ iter->net = get_net(PROC_NET(inode));
  return 0;
 }

+static int netlink_seq_release(struct inode *inode, struct file *file)
+{
+ struct seq_file *seq = file->private_data;
```

```
+ struct nl_seq_iter *iter = seq->private;
+ put_net(iter->net);
+ return seq_release_private(inode, file);
+}
+
 static const struct file_operations netlink_seq_fops = {
  .owner  = THIS_MODULE,
  .open  = netlink_seq_open,
  .read  = seq_read,
  .llseek  = seq_lseek,
- .release = seq_release_private,
+ .release = netlink_seq_release,
 };

 #endif
@@ -1885,6 +1918,27 @@ static struct net_proto_family netlink_family_ops = {
  .owner = THIS_MODULE, /* for consistency 8) */
 };

+static int netlink_net_init(struct net *net)
+{
+#ifdef CONFIG_PROC_FS
+ if (!proc_net_fops_create(net, "netlink", 0, &netlink_seq_fops))
+  return -ENOMEM;
+#endif
+ return 0;
+}
+
+static void netlink_net_exit(struct net *net)
+{
+#ifdef CONFIG_PROC_FS
+ proc_net_remove(net, "netlink");
+#endif
+}
+
+static struct pernet_operations netlink_net_ops = {
+ .init = netlink_net_init,
+ .exit = netlink_net_exit,
+};
+
 static int __init netlink_proto_init(void)
 {
  struct sk_buff *dummy_skb;
@@ -1930,10 +1984,8 @@ static int __init netlink_proto_init(void)
  }

  sock_register(&netlink_family_ops);
-#ifdef CONFIG_PROC_FS
```

```
- proc_net_fops_create(&init_net, "netlink", 0, &netlink_seq_fops);
-#endif
- /* The netlink device handler may be needed early. */
+ register_pernet_subsys(&netlink_net_ops);
+ /* The netlink device handler may be needed early. */
  rtnetlink_init();
 out:
  return err;
diff --git a/net/netlink/genetlink.c b/net/netlink/genetlink.c
index 8c11ca4..af8fe26 100644
--- a/net/netlink/genetlink.c
+++ b/net/netlink/genetlink.c
@@ -782,8 +782,8 @@ static int __init genl_init(void)
  netlink_set_nonroot(NETLINK_GENERIC, NL_NONROOT_RECV);

  /* we'll bump the group number right afterwards */
- genl_sock = netlink_kernel_create(NETLINK_GENERIC, 0, genl_rcv,
-       NULL, THIS_MODULE);
+ genl_sock = netlink_kernel_create(&init_net, NETLINK_GENERIC, 0,
+       genl_rcv, NULL, THIS_MODULE);
  if (genl_sock == NULL)
   panic("GENL: Cannot initialize generic netlink\n");

diff --git a/net/xfrm/xfrm_user.c b/net/xfrm/xfrm_user.c
index 46076f5..2035d62 100644
--- a/net/xfrm/xfrm_user.c
+++ b/net/xfrm/xfrm_user.c
@@ -2391,7 +2391,7 @@ static int __init xfrm_user_init(void)

  printk(KERN_INFO "Initializing XFRM netlink socket\n");

- nlsk = netlink_kernel_create(NETLINK_XFRM, XFRMNLGRP_MAX,
+ nlsk = netlink_kernel_create(&init_net, NETLINK_XFRM, XFRMNLGRP_MAX,
         xfrm_netlink_rcv, NULL, THIS_MODULE);
  if (nlsk == NULL)
   return -ENOMEM;
diff --git a/security/selinux/netlink.c b/security/selinux/netlink.c
index f49046d..b59871d 100644
--- a/security/selinux/netlink.c
+++ b/security/selinux/netlink.c
@@ -17,6 +17,7 @@
 #include <linux/skbuff.h>
 #include <linux/netlink.h>
 #include <linux/selinux_netlink.h>
+#include <net/net_namespace.h>

 static struct sock *selnl;
```

```
@@ -104,8 +105,8 @@ void selnl_notify_policyload(u32 seqno)

 static int __init selnl_init(void)
 {
- selnl = netlink_kernel_create(NETLINK_SELINUX, SELNLGRP_MAX, NULL, NULL,
-                               THIS_MODULE);
+ selnl = netlink_kernel_create(&init_net, NETLINK_SELINUX,
+         SELNLGRP_MAX, NULL, NULL, THIS_MODULE);
  if (selnl == NULL)
   panic("SELinux:  Cannot create netlink socket.");
  netlink_set_nonroot(NETLINK_SELINUX, NL_NONROOT_RECV);
--
1.5.3.rc6.17.g1911
```

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: [PATCH 13/16] net: Make the device list and device lookups per namespace.
Posted by ebiederm on Sat, 08 Sep 2007 21:35:46 GMT
View Forum Message <> Reply to Message

This patch makes most of the generic device layer network
namespace safe. This patch makes dev_base_head a
network namespace variable, and then it picks up
a few associated variables. The functions:
dev_getbyhwaddr
dev_getfirsthwbytype
dev_get_by_flags
dev_get_by_name
__dev_get_by_name
dev_get_by_index
__dev_get_by_index
dev_ioctl
dev_ethtool
dev_load
wireless_process_ioctl

were modified to take a network namespace argument, and
deal with it.

vlan_ioctl_set and brioctl_set were modified so their
hooks will receive a network namespace argument.

So basically anthing in the core of the network stack that was

affected to by the change of dev_base was modified to handle
multiple network namespaces. The rest of the network stack was
simply modified to explicitly use &init_net the initial network
namespace. This can be fixed when those components of the network
stack are modified to handle multiple network namespaces.

For now the ifindex generator is left global.

Fundametally ifindex numbers are per namespace, or else
we will have corner case problems with migration when
we get that far.

At the same time there are assumptions in the network stack
that the ifindex of a network device won't change. Making
the ifindex number global seems a good compromise until
the network stack can cope with ifindex changes when
you change namespaces, and the like.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 arch/s390/appldata/appldata_net_sum.c | 3 +-
 arch/sparc64/solaris/ioctl.c | 3 +-
 drivers/atm/idt77252.c | 2 +-
 drivers/block/aoe/aoecmd.c | 3 +-
 drivers/infiniband/hw/cxgb3/cxio_hal.c | 3 +-
 drivers/net/bonding/bond_main.c | 2 +-
 drivers/net/bonding/bond_sysfs.c | 3 +-
 drivers/net/eql.c | 9 +-
 drivers/net/ifb.c | 3 +-
 drivers/net/macvlan.c | 2 +-
 drivers/net/pppoe.c | 4 +-
 drivers/net/shaper.c | 3 +-
 drivers/net/tun.c | 3 +-
 drivers/net/veth.c | 2 +-
 drivers/net/wan/dlci.c | 4 +-
 drivers/net/wan/sbni.c | 3 +-
 drivers/net/wireless/strip.c | 2 +-
 drivers/parisc/led.c | 2 +-
 fs/afs/netdevices.c | 5 +-
 include/linux/if_bridge.h | 2 +-
 include/linux/if_vlan.h | 2 +-
 include/linux/netdevice.h | 66 ++++++----
 include/net/net_namespace.h | 4 +
 include/net/pkt_cls.h | 3 +-
 include/net/rtnetlink.h | 2 +-
 include/net/wext.h | 15 ++-
 net/802/tr.c | 2 +-
 net/8021q/vlan.c | 6 +-

```
net/8021q/vlan_netlink.c            |   3 +-
net/8021q/vlanproc.c                |   6 +-
net/appletalk/ddp.c                 |   6 +-
net/atm/mpc.c                       |   2 +-
net/ax25/af_ax25.c                  |   2 +-
net/bridge/br_if.c                  |   4 +-
net/bridge/br_ioctl.c               |   7 +-
net/bridge/br_netlink.c             |   5 +-
net/bridge/br_private.h             |   2 +-
net/core/dev.c                      | 271 ++++++++++++++++++++++-----------
net/core/dev_mcast.c                |  41 +++++-
net/core/ethtool.c                  |   4 +-
net/core/fib_rules.c                |   4 +-
net/core/neighbour.c                |   6 +-
net/core/netpoll.c                  |   2 +-
net/core/pktgen.c                   |   2 +-
net/core/rtnetlink.c                |  35 +++--
net/core/sock.c                     |   3 +-
net/decnet/af_decnet.c              |   2 +-
net/decnet/dn_dev.c                 |  20 ++--
net/decnet/dn_fib.c                 |   8 +-
net/decnet/dn_route.c               |   6 +-
net/decnet/sysctl_net_decnet.c      |   4 +-
net/econet/af_econet.c              |   2 +-
net/ipv4/arp.c                      |   4 +-
net/ipv4/devinet.c                  |  18 +-
net/ipv4/fib_frontend.c             |   2 +-
net/ipv4/fib_semantics.c            |   4 +-
net/ipv4/icmp.c                     |   2 +-
net/ipv4/igmp.c                     |   4 +-
net/ipv4/ip_fragment.c              |   2 +-
net/ipv4/ip_gre.c                   |   4 +-
net/ipv4/ip_sockglue.c              |   2 +-
net/ipv4/ipconfig.c                 |   2 +-
net/ipv4/ipip.c                     |   4 +-
net/ipv4/ipmr.c                     |   4 +-
net/ipv4/ipvs/ip_vs_sync.c          |  10 +-
net/ipv4/netfilter/ipt_CLUSTERIP.c  |   2 +-
net/ipv4/route.c                    |   4 +-
net/ipv6/addrconf.c                 |  28 ++--
net/ipv6/af_inet6.c                 |   2 +-
net/ipv6/anycast.c                  |  12 +-
net/ipv6/datagram.c                 |   2 +-
net/ipv6/ip6_tunnel.c               |   6 +-
net/ipv6/ipv6_sockglue.c            |   2 +-
net/ipv6/mcast.c                    |  12 +-
net/ipv6/raw.c                      |   2 +-
net/ipv6/reassembly.c               |   2 +-
```

```
 net/ipv6/route.c                   |    4 +-
 net/ipv6/sit.c                     |    4 +-
 net/ipx/af_ipx.c                   |    6 +-
 net/irda/irnetlink.c               |    9 +-
 net/llc/af_llc.c                   |    4 +-
 net/llc/llc_core.c                 |    3 +-
 net/mac80211/ieee80211.c           |    1 +
 net/mac80211/ieee80211_cfg.c       |    3 +-
 net/mac80211/tx.c                  |    9 +-
 net/mac80211/util.c                |    7 +-
 net/netrom/nr_route.c              |    6 +-
 net/packet/af_packet.c             |   18 +-
 net/rose/rose_route.c              |    8 +-
 net/sched/act_mirred.c             |    3 +-
 net/sched/cls_api.c                |    4 +-
 net/sched/em_meta.c                |    2 +-
 net/sched/sch_api.c                |   10 +-
 net/sctp/ipv6.c                    |    4 +-
 net/sctp/protocol.c                |    2 +-
 net/socket.c                       |   22 ++-
 net/tipc/eth_media.c               |    2 +-
 net/wireless/wext.c                |   38 ++++-
 net/x25/x25_route.c                |    2 +-
 99 files changed, 555 insertions(+), 362 deletions(-)

diff --git a/arch/s390/appldata/appldata_net_sum.c b/arch/s390/appldata/appldata_net_sum.c
index 2180ac1..6c1815a 100644
--- a/arch/s390/appldata/appldata_net_sum.c
+++ b/arch/s390/appldata/appldata_net_sum.c
@@ -16,6 +16,7 @@
 #include <linux/errno.h>
 #include <linux/kernel_stat.h>
 #include <linux/netdevice.h>
+#include <net/net_namespace.h>

 #include "appldata.h"

@@ -107,7 +108,7 @@ static void appldata_get_net_sum_data(void *data)
  tx_dropped = 0;
  collisions = 0;
  read_lock(&dev_base_lock);
- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
   stats = dev->get_stats(dev);
   rx_packets += stats->rx_packets;
   tx_packets += stats->tx_packets;
diff --git a/arch/sparc64/solaris/ioctl.c b/arch/sparc64/solaris/ioctl.c
index 18352a4..8ad10a6 100644
```

```
--- a/arch/sparc64/solaris/ioctl.c
+++ b/arch/sparc64/solaris/ioctl.c
@@ -28,6 +28,7 @@
 #include <linux/compat.h>

 #include <net/sock.h>
+#include <net/net_namespace.h>

 #include <asm/uaccess.h>
 #include <asm/termios.h>
@@ -686,7 +687,7 @@ static inline int solaris_i(unsigned int fd, unsigned int cmd, u32 arg)
    int i = 0;

    read_lock_bh(&dev_base_lock);
-   for_each_netdev(d)
+   for_each_netdev(&init_net, d)
     i++;
    read_unlock_bh(&dev_base_lock);

diff --git a/drivers/atm/idt77252.c b/drivers/atm/idt77252.c
index f8b1700..eee54c0 100644
--- a/drivers/atm/idt77252.c
+++ b/drivers/atm/idt77252.c
@@ -3576,7 +3576,7 @@ init_card(struct atm_dev *dev)
   * XXX: <hack>
   */
  sprintf(tname, "eth%d", card->index);
- tmp = dev_get_by_name(tname); /* jhs: was "tmp = dev_get(tname);" */
+ tmp = dev_get_by_name(&init_net, tname); /* jhs: was "tmp = dev_get(tname);" */
  if (tmp) {
   memcpy(card->atmdev->esi, tmp->dev_addr, 6);

diff --git a/drivers/block/aoe/aoecmd.c b/drivers/block/aoe/aoecmd.c
index 01fbdd3..30394f7 100644
--- a/drivers/block/aoe/aoecmd.c
+++ b/drivers/block/aoe/aoecmd.c
@@ -9,6 +9,7 @@
 #include <linux/skbuff.h>
 #include <linux/netdevice.h>
 #include <linux/genhd.h>
+#include <net/net_namespace.h>
 #include <asm/unaligned.h>
 #include "aoe.h"

@@ -194,7 +195,7 @@ aoecmd_cfg_pkts(ushort aoemajor, unsigned char aoeminor, struct
sk_buff **tail)
  sl = sl_tail = NULL;
```

```
  read_lock(&dev_base_lock);
- for_each_netdev(ifp) {
+ for_each_netdev(&init_net, ifp) {
   dev_hold(ifp);
   if (!is_aoe_netif(ifp))
    goto cont;
diff --git a/drivers/infiniband/hw/cxgb3/cxio_hal.c b/drivers/infiniband/hw/cxgb3/cxio_hal.c
index 1518b41..7b92b7d 100644
--- a/drivers/infiniband/hw/cxgb3/cxio_hal.c
+++ b/drivers/infiniband/hw/cxgb3/cxio_hal.c
@@ -37,6 +37,7 @@
 #include <linux/spinlock.h>
 #include <linux/pci.h>
 #include <linux/dma-mapping.h>
+#include <net/net_namespace.h>

 #include "cxio_resource.h"
 #include "cxio_hal.h"
@@ -894,7 +895,7 @@ int cxio_rdev_open(struct cxio_rdev *rdev_p)
   if (cxio_hal_find_rdev_by_name(rdev_p->dev_name)) {
    return -EBUSY;
   }
-  netdev_p = dev_get_by_name(rdev_p->dev_name);
+  netdev_p = dev_get_by_name(&init_net, rdev_p->dev_name);
   if (!netdev_p) {
    return -EINVAL;
   }
diff --git a/drivers/net/bonding/bond_main.c b/drivers/net/bonding/bond_main.c
index cf97d8a..559fe94 100644
--- a/drivers/net/bonding/bond_main.c
+++ b/drivers/net/bonding/bond_main.c
@@ -3719,7 +3719,7 @@ static int bond_do_ioctl(struct net_device *bond_dev, struct ifreq *ifr,
int cmd
  }

  down_write(&(bonding_rwsem));
- slave_dev = dev_get_by_name(ifr->ifr_slave);
+ slave_dev = dev_get_by_name(&init_net, ifr->ifr_slave);

  dprintk("slave_dev=%p: \n", slave_dev);

diff --git a/drivers/net/bonding/bond_sysfs.c b/drivers/net/bonding/bond_sysfs.c
index 60cccf2..8289e27 100644
--- a/drivers/net/bonding/bond_sysfs.c
+++ b/drivers/net/bonding/bond_sysfs.c
@@ -35,6 +35,7 @@
 #include <linux/ctype.h>
 #include <linux/inet.h>
```

```
 #include <linux/rtnetlink.h>
+#include <net/net_namespace.h>

 /* #define BONDING_DEBUG 1 */
 #include "bonding.h"
@@ -299,7 +300,7 @@ static ssize_t bonding_store_slaves(struct device *d,
   read_unlock_bh(&bond->lock);
   printk(KERN_INFO DRV_NAME ": %s: Adding slave %s.\n",
          bond->dev->name, ifname);
-  dev = dev_get_by_name(ifname);
+  dev = dev_get_by_name(&init_net, ifname);
   if (!dev) {
    printk(KERN_INFO DRV_NAME
           ": %s: Interface %s does not exist!\n",
diff --git a/drivers/net/eql.c b/drivers/net/eql.c
index 102218c..f1cc66d 100644
--- a/drivers/net/eql.c
+++ b/drivers/net/eql.c
@@ -116,6 +116,7 @@
 #include <linux/init.h>
 #include <linux/timer.h>
 #include <linux/netdevice.h>
+#include <net/net_namespace.h>

 #include <linux/if.h>
 #include <linux/if_arp.h>
@@ -412,7 +413,7 @@ static int eql_enslave(struct net_device *master_dev, slaving_request_t
__user *
  if (copy_from_user(&srq, srqp, sizeof (slaving_request_t)))
   return -EFAULT;

- slave_dev  = dev_get_by_name(srq.slave_name);
+ slave_dev  = dev_get_by_name(&init_net, srq.slave_name);
  if (slave_dev) {
   if ((master_dev->flags & IFF_UP) == IFF_UP) {
    /* slave is not a master & not already a slave: */
@@ -460,7 +461,7 @@ static int eql_emancipate(struct net_device *master_dev,
slaving_request_t __use
  if (copy_from_user(&srq, srqp, sizeof (slaving_request_t)))
   return -EFAULT;

- slave_dev = dev_get_by_name(srq.slave_name);
+ slave_dev = dev_get_by_name(&init_net, srq.slave_name);
  ret = -EINVAL;
  if (slave_dev) {
   spin_lock_bh(&eql->queue.lock);
@@ -493,7 +494,7 @@ static int eql_g_slave_cfg(struct net_device *dev, slave_config_t __user
*scp)
```

```
  if (copy_from_user(&sc, scp, sizeof (slave_config_t)))
   return -EFAULT;

- slave_dev = dev_get_by_name(sc.slave_name);
+ slave_dev = dev_get_by_name(&init_net, sc.slave_name);
  if (!slave_dev)
   return -ENODEV;

@@ -528,7 +529,7 @@ static int eql_s_slave_cfg(struct net_device *dev, slave_config_t __user
*scp)
  if (copy_from_user(&sc, scp, sizeof (slave_config_t)))
   return -EFAULT;

- slave_dev = dev_get_by_name(sc.slave_name);
+ slave_dev = dev_get_by_name(&init_net, sc.slave_name);
  if (!slave_dev)
   return -ENODEV;

diff --git a/drivers/net/ifb.c b/drivers/net/ifb.c
index f5c3598..b06c6db 100644
--- a/drivers/net/ifb.c
+++ b/drivers/net/ifb.c
@@ -34,6 +34,7 @@
 #include <linux/init.h>
 #include <linux/moduleparam.h>
 #include <net/pkt_sched.h>
+#include <net/net_namespace.h>

 #define TX_TIMEOUT  (2*HZ)

@@ -97,7 +98,7 @@ static void ri_tasklet(unsigned long dev)
   stats->tx_packets++;
   stats->tx_bytes +=skb->len;

-  skb->dev = __dev_get_by_index(skb->iif);
+  skb->dev = __dev_get_by_index(&init_net, skb->iif);
   if (!skb->dev) {
    dev_kfree_skb(skb);
    stats->tx_dropped++;
diff --git a/drivers/net/macvlan.c b/drivers/net/macvlan.c
index dc74d00..2de073d 100644
--- a/drivers/net/macvlan.c
+++ b/drivers/net/macvlan.c
@@ -376,7 +376,7 @@ static int macvlan_newlink(struct net_device *dev,
  if (!tb[IFLA_LINK])
   return -EINVAL;

- lowerdev = __dev_get_by_index(nla_get_u32(tb[IFLA_LINK]));
```

```
+ lowerdev = __dev_get_by_index(dev->nd_net, nla_get_u32(tb[IFLA_LINK]));
  if (lowerdev == NULL)
   return -ENODEV;

diff --git a/drivers/net/pppoe.c b/drivers/net/pppoe.c
index b4ba584..5f5f124 100644
--- a/drivers/net/pppoe.c
+++ b/drivers/net/pppoe.c
@@ -216,7 +216,7 @@ static inline struct pppox_sock *get_item_by_addr(struct sockaddr_pppox
*sp)
  struct net_device *dev;
  int ifindex;

- dev = dev_get_by_name(sp->sa_addr.pppoe.dev);
+ dev = dev_get_by_name(&init_net, sp->sa_addr.pppoe.dev);
  if(!dev)
   return NULL;
  ifindex = dev->ifindex;
@@ -603,7 +603,7 @@ static int pppoe_connect(struct socket *sock, struct sockaddr *uservaddr,

  /* Don't re-bind if sid==0 */
  if (sp->sa_addr.pppoe.sid != 0) {
-  dev = dev_get_by_name(sp->sa_addr.pppoe.dev);
+  dev = dev_get_by_name(&init_net, sp->sa_addr.pppoe.dev);

   error = -ENODEV;
   if (!dev)
diff --git a/drivers/net/shaper.c b/drivers/net/shaper.c
index 4c3d98f..3773b38 100644
--- a/drivers/net/shaper.c
+++ b/drivers/net/shaper.c
@@ -86,6 +86,7 @@

 #include <net/dst.h>
 #include <net/arp.h>
+#include <net/net_namespace.h>

 struct shaper_cb {
  unsigned long shapeclock;  /* Time it should go out */
@@ -488,7 +489,7 @@ static int shaper_ioctl(struct net_device *dev,  struct ifreq *ifr, int cmd)
  {
   case SHAPER_SET_DEV:
   {
-   struct net_device *them=__dev_get_by_name(ss->ss_name);
+   struct net_device *them=__dev_get_by_name(&init_net, ss->ss_name);
    if(them==NULL)
     return -ENODEV;
    if(sh->dev)
```

```
diff --git a/drivers/net/tun.c b/drivers/net/tun.c
index 62b2b30..691d264 100644
--- a/drivers/net/tun.c
+++ b/drivers/net/tun.c
@@ -62,6 +62,7 @@
 #include <linux/if_ether.h>
 #include <linux/if_tun.h>
 #include <linux/crc32.h>
+#include <net/net_namespace.h>

 #include <asm/system.h>
 #include <asm/uaccess.h>
@@ -475,7 +476,7 @@ static int tun_set_iff(struct file *file, struct ifreq *ifr)
        !capable(CAP_NET_ADMIN))
    return -EPERM;
  }
- else if (__dev_get_by_name(ifr->ifr_name))
+ else if (__dev_get_by_name(&init_net, ifr->ifr_name))
   return -EINVAL;
  else {
   char *name;
diff --git a/drivers/net/veth.c b/drivers/net/veth.c
index 743fd62..9e6a746 100644
--- a/drivers/net/veth.c
+++ b/drivers/net/veth.c
@@ -346,7 +346,7 @@ static int veth_newlink(struct net_device *dev,
  else
   snprintf(ifname, IFNAMSIZ, DRV_NAME "%%d");

- peer = rtnl_create_link(ifname, &veth_link_ops, tbp);
+ peer = rtnl_create_link(dev->nd_net, ifname, &veth_link_ops, tbp);
  if (IS_ERR(peer))
   return PTR_ERR(peer);

diff --git a/drivers/net/wan/dlci.c b/drivers/net/wan/dlci.c
index 61041d5..bc12810 100644
--- a/drivers/net/wan/dlci.c
+++ b/drivers/net/wan/dlci.c
@@ -361,7 +361,7 @@ static int dlci_add(struct dlci_add *dlci)


  /* validate slave device */
- slave = dev_get_by_name(dlci->devname);
+ slave = dev_get_by_name(&init_net, dlci->devname);
  if (!slave)
   return -ENODEV;

@@ -427,7 +427,7 @@ static int dlci_del(struct dlci_add *dlci)
```

```
  int   err;

  /* validate slave device */
- master = __dev_get_by_name(dlci->devname);
+ master = __dev_get_by_name(&init_net, dlci->devname);
  if (!master)
   return(-ENODEV);

diff --git a/drivers/net/wan/sbni.c b/drivers/net/wan/sbni.c
index 1cc18e7..8d7e01e 100644
--- a/drivers/net/wan/sbni.c
+++ b/drivers/net/wan/sbni.c
@@ -54,6 +54,7 @@
 #include <linux/init.h>
 #include <linux/delay.h>

+#include <net/net_namespace.h>
 #include <net/arp.h>

 #include <asm/io.h>
@@ -1361,7 +1362,7 @@ sbni_ioctl( struct net_device  *dev,  struct ifreq  *ifr,  int  cmd )

   if (copy_from_user( slave_name, ifr->ifr_data, sizeof slave_name ))
    return -EFAULT;
-  slave_dev = dev_get_by_name( slave_name );
+  slave_dev = dev_get_by_name(&init_net, slave_name );
   if( !slave_dev  ||  !(slave_dev->flags & IFF_UP) ) {
    printk( KERN_ERR "%s: trying to enslave non-active "
     "device %s\n", dev->name, slave_name );
diff --git a/drivers/net/wireless/strip.c b/drivers/net/wireless/strip.c
index edb214e..904e548 100644
--- a/drivers/net/wireless/strip.c
+++ b/drivers/net/wireless/strip.c
@@ -1972,7 +1972,7 @@ static struct net_device *get_strip_dev(struct strip *strip_info)
         sizeof(zero_address))) {
   struct net_device *dev;
   read_lock_bh(&dev_base_lock);
-  for_each_netdev(dev) {
+  for_each_netdev(&init_net, dev) {
    if (dev->type == strip_info->dev->type &&
        !memcmp(dev->dev_addr,
         &strip_info->true_dev_addr,
diff --git a/drivers/parisc/led.c b/drivers/parisc/led.c
index e5d7ed9..a6d6b24 100644
--- a/drivers/parisc/led.c
+++ b/drivers/parisc/led.c
@@ -359,7 +359,7 @@ static __inline__ int led_get_net_activity(void)
   * for reading should be OK */
```

```
  read_lock(&dev_base_lock);
  rcu_read_lock();
- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
      struct net_device_stats *stats;
      struct in_device *in_dev = __in_dev_get_rcu(dev);
      if (!in_dev || !in_dev->ifa_list)
diff --git a/fs/afs/netdevices.c b/fs/afs/netdevices.c
index fc27d4b..49f1894 100644
--- a/fs/afs/netdevices.c
+++ b/fs/afs/netdevices.c
@@ -8,6 +8,7 @@
 #include <linux/inetdevice.h>
 #include <linux/netdevice.h>
 #include <linux/if_arp.h>
+#include <net/net_namespace.h>
 #include "internal.h"

 /*
@@ -23,7 +24,7 @@ int afs_get_MAC_address(u8 *mac, size_t maclen)
   BUG();

  rtnl_lock();
- dev = __dev_getfirstbyhwtype(ARPHRD_ETHER);
+ dev = __dev_getfirstbyhwtype(&init_net, ARPHRD_ETHER);
  if (dev) {
   memcpy(mac, dev->dev_addr, maclen);
   ret = 0;
@@ -47,7 +48,7 @@ int afs_get_ipv4_interfaces(struct afs_interface *bufs, size_t maxbufs,
  ASSERT(maxbufs > 0);

  rtnl_lock();
- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
   if (dev->type == ARPHRD_LOOPBACK && !wantloopback)
    continue;
   idev = __in_dev_get_rtnl(dev);
diff --git a/include/linux/if_bridge.h b/include/linux/if_bridge.h
index 4ff211d..99e3a1a 100644
--- a/include/linux/if_bridge.h
+++ b/include/linux/if_bridge.h
@@ -104,7 +104,7 @@ struct __fdb_entry

 #include <linux/netdevice.h>

-extern void brioctl_set(int (*ioctl_hook)(unsigned int, void __user *));
+extern void brioctl_set(int (*ioctl_hook)(struct net *, unsigned int, void __user *));
 extern struct sk_buff *(*br_handle_frame_hook)(struct net_bridge_port *p,
```

```
             struct sk_buff *skb);
 extern int (*br_should_route_hook)(struct sk_buff **pskb);
diff --git a/include/linux/if_vlan.h b/include/linux/if_vlan.h
index f8443fd..976d4b1 100644
--- a/include/linux/if_vlan.h
+++ b/include/linux/if_vlan.h
@@ -62,7 +62,7 @@ struct vlan_hdr {
 #define VLAN_VID_MASK 0xfff

 /* found in socket.c */
-extern void vlan_ioctl_set(int (*hook)(void __user *));
+extern void vlan_ioctl_set(int (*hook)(struct net *, void __user *));

 #define VLAN_NAME "vlan"

diff --git a/include/linux/netdevice.h b/include/linux/netdevice.h
index 8eeced0..ec90d1a 100644
--- a/include/linux/netdevice.h
+++ b/include/linux/netdevice.h
@@ -727,44 +727,48 @@ struct packet_type {
 #include <linux/notifier.h>

 extern struct net_device  loopback_dev;  /* The loopback */
-extern struct list_head   dev_base_head;  /* All devices */
 extern rwlock_t     dev_base_lock;  /* Device list lock */

-#define for_each_netdev(d)  \
-  list_for_each_entry(d, &dev_base_head, dev_list)
-#define for_each_netdev_safe(d, n) \
-  list_for_each_entry_safe(d, n, &dev_base_head, dev_list)
-#define for_each_netdev_continue(d)  \
-  list_for_each_entry_continue(d, &dev_base_head, dev_list)
-#define net_device_entry(lh) list_entry(lh, struct net_device, dev_list)
-
-static inline struct net_device *next_net_device(struct net_device *dev)
-{
- struct list_head *lh;

- lh = dev->dev_list.next;
- return lh == &dev_base_head ? NULL : net_device_entry(lh);
-}
+#define for_each_netdev(net, d)  \
+  list_for_each_entry(d, &(net)->dev_base_head, dev_list)
+#define for_each_netdev_safe(net, d, n) \
+  list_for_each_entry_safe(d, n, &(net)->dev_base_head, dev_list)
+#define for_each_netdev_continue(net, d)  \
+  list_for_each_entry_continue(d, &(net)->dev_base_head, dev_list)
+#define net_device_entry(lh) list_entry(lh, struct net_device, dev_list)
```

```
-static inline struct net_device *first_net_device(void)
-{
- return list_empty(&dev_base_head) ? NULL :
-  net_device_entry(dev_base_head.next);
-}
+#define next_net_device(d)       \
+({         \
+ struct net_device *dev = d;     \
+ struct list_head *lh;      \
+ struct net *net;      \
+         \
+ net = dev->nd_net;      \
+ lh = dev->dev_list.next;     \
+ lh == &net->dev_base_head ? NULL : net_device_entry(lh); \
+})
+
+#define first_net_device(N)     \
+({        \
+ struct net *NET = (N);     \
+ list_empty(&NET->dev_base_head) ? NULL :  \
+  net_device_entry(NET->dev_base_head.next); \
+})

 extern int    netdev_boot_setup_check(struct net_device *dev);
 extern unsigned long  netdev_boot_base(const char *prefix, int unit);
-extern struct net_device    *dev_getbyhwaddr(unsigned short type, char *hwaddr);
-extern struct net_device *dev_getfirstbyhwtype(unsigned short type);
-extern struct net_device *__dev_getfirstbyhwtype(unsigned short type);
+extern struct net_device    *dev_getbyhwaddr(struct net *net, unsigned short type, char *hwaddr);
+extern struct net_device *dev_getfirstbyhwtype(struct net *net, unsigned short type);
+extern struct net_device *__dev_getfirstbyhwtype(struct net *net, unsigned short type);
 extern void  dev_add_pack(struct packet_type *pt);
 extern void  dev_remove_pack(struct packet_type *pt);
 extern void  __dev_remove_pack(struct packet_type *pt);

-extern struct net_device *dev_get_by_flags(unsigned short flags,
+extern struct net_device *dev_get_by_flags(struct net *net, unsigned short flags,
        unsigned short mask);
-extern struct net_device *dev_get_by_name(const char *name);
-extern struct net_device *__dev_get_by_name(const char *name);
+extern struct net_device *dev_get_by_name(struct net *net, const char *name);
+extern struct net_device *__dev_get_by_name(struct net *net, const char *name);
 extern int  dev_alloc_name(struct net_device *dev, const char *name);
 extern int  dev_open(struct net_device *dev);
 extern int  dev_close(struct net_device *dev);
@@ -776,8 +780,8 @@ extern void  synchronize_net(void);
 extern int   register_netdevice_notifier(struct notifier_block *nb);
```

```
 extern int  unregister_netdevice_notifier(struct notifier_block *nb);
 extern int  call_netdevice_notifiers(unsigned long val, void *v);
-extern struct net_device *dev_get_by_index(int ifindex);
-extern struct net_device *__dev_get_by_index(int ifindex);
+extern struct net_device *dev_get_by_index(struct net *net, int ifindex);
+extern struct net_device *__dev_get_by_index(struct net *net, int ifindex);
 extern int  dev_restart(struct net_device *dev);
 #ifdef CONFIG_NETPOLL_TRAP
 extern int  netpoll_trap(void);
@@ -993,8 +997,8 @@ extern int  netif_rx_ni(struct sk_buff *skb);
 #define HAVE_NETIF_RECEIVE_SKB 1
 extern int  netif_receive_skb(struct sk_buff *skb);
 extern int  dev_valid_name(const char *name);
-extern int  dev_ioctl(unsigned int cmd, void __user *);
-extern int  dev_ethtool(struct ifreq *);
+extern int  dev_ioctl(struct net *net, unsigned int cmd, void __user *);
+extern int  dev_ethtool(struct net *net, struct ifreq *);
 extern unsigned  dev_get_flags(const struct net_device *);
 extern int  dev_change_flags(struct net_device *, unsigned);
 extern int  dev_change_name(struct net_device *, char *);
@@ -1320,7 +1324,7 @@ extern void  dev_set_allmulti(struct net_device *dev, int inc);
 extern void  netdev_state_change(struct net_device *dev);
 extern void  netdev_features_change(struct net_device *dev);
 /* Load a device via the kmod */
-extern void  dev_load(const char *name);
+extern void  dev_load(struct net *net, const char *name);
 extern void  dev_mcast_init(void);
 extern int  netdev_max_backlog;
 extern int  weight_p;
diff --git a/include/net/net_namespace.h b/include/net/net_namespace.h
index 5472476..fac42db 100644
--- a/include/net/net_namespace.h
+++ b/include/net/net_namespace.h
@@ -22,6 +22,10 @@ struct net {
  struct proc_dir_entry  *proc_net;
  struct proc_dir_entry  *proc_net_stat;
  struct proc_dir_entry  *proc_net_root;
+
+ struct list_head  dev_base_head;
+ struct hlist_head  *dev_name_head;
+ struct hlist_head *dev_index_head;
 };

 extern struct net init_net;
diff --git a/include/net/pkt_cls.h b/include/net/pkt_cls.h
index 7968b1d..f285de6 100644
--- a/include/net/pkt_cls.h
+++ b/include/net/pkt_cls.h
```

```
@@ -2,6 +2,7 @@
 #define __NET_PKT_CLS_H

 #include <linux/pkt_cls.h>
+#include <net/net_namespace.h>
 #include <net/sch_generic.h>
 #include <net/act_api.h>

@@ -351,7 +352,7 @@ tcf_match_indev(struct sk_buff *skb, char *indev)
  if (indev[0]) {
   if  (!skb->iif)
    return 0;
-  dev = __dev_get_by_index(skb->iif);
+  dev = __dev_get_by_index(&init_net, skb->iif);
   if (!dev || strcmp(indev, dev->name))
    return 0;
  }
diff --git a/include/net/rtnetlink.h b/include/net/rtnetlink.h
index 8218288..793863e 100644
--- a/include/net/rtnetlink.h
+++ b/include/net/rtnetlink.h
@@ -78,7 +78,7 @@ extern void __rtnl_link_unregister(struct rtnl_link_ops *ops);
 extern int rtnl_link_register(struct rtnl_link_ops *ops);
 extern void rtnl_link_unregister(struct rtnl_link_ops *ops);

-extern struct net_device *rtnl_create_link(char *ifname,
+extern struct net_device *rtnl_create_link(struct net *net, char *ifname,
   const struct rtnl_link_ops *ops, struct nlattr *tb[]);
 extern const struct nla_policy ifla_policy[IFLA_MAX+1];

diff --git a/include/net/wext.h b/include/net/wext.h
index c02b8de..80b31d8 100644
--- a/include/net/wext.h
+++ b/include/net/wext.h
@@ -5,16 +5,23 @@
  * wireless extensions interface to the core code
  */

+struct net;
+
 #ifdef CONFIG_WIRELESS_EXT
-extern int wext_proc_init(void);
-extern int wext_handle_ioctl(struct ifreq *ifr, unsigned int cmd,
+extern int wext_proc_init(struct net *net);
+extern void wext_proc_exit(struct net *net);
+extern int wext_handle_ioctl(struct net *net, struct ifreq *ifr, unsigned int cmd,
        void __user *arg);
 #else
```

```
-static inline int wext_proc_init(void)
+static inline int wext_proc_init(struct net *net)
 {
  return 0;
 }
-static inline int wext_handle_ioctl(struct ifreq *ifr, unsigned int cmd,
+static inline void wext_proc_exit(struct net *net)
+{
+ return;
+}
+static inline int wext_handle_ioctl(struct net *net, struct ifreq *ifr, unsigned int cmd,
         void __user *arg)
 {
  return -EINVAL;
diff --git a/net/802/tr.c b/net/802/tr.c
index 032c31e..55c76d7 100644
--- a/net/802/tr.c
+++ b/net/802/tr.c
@@ -533,7 +533,7 @@ static int rif_seq_show(struct seq_file *seq, void *v)
   seq_puts(seq,
        "if     TR address       TTL   rcf   routing segments\n");
  else {
-  struct net_device *dev = dev_get_by_index(entry->iface);
+  struct net_device *dev = dev_get_by_index(&init_net, entry->iface);
   long ttl = (long) (entry->last_used + sysctl_tr_rif_timeout)
     - (long) jiffies;

diff --git a/net/8021q/vlan.c b/net/8021q/vlan.c
index 55f2dbb..546f6db 100644
--- a/net/8021q/vlan.c
+++ b/net/8021q/vlan.c
@@ -51,7 +51,7 @@ static char vlan_copyright[] = "Ben Greear <greearb@candelatech.com>";
 static char vlan_buggyright[] = "David S. Miller <davem@redhat.com>";

 static int vlan_device_event(struct notifier_block *, unsigned long, void *);
-static int vlan_ioctl_handler(void __user *);
+static int vlan_ioctl_handler(struct net *net, void __user *);
 static int unregister_vlan_dev(struct net_device *, unsigned short );

 static struct notifier_block vlan_notifier_block = {
@@ -699,7 +699,7 @@ out:
  * o execute requested action or pass command to the device driver
  *   arg is really a struct vlan_ioctl_args __user *.
  */
-static int vlan_ioctl_handler(void __user *arg)
+static int vlan_ioctl_handler(struct net *net, void __user *arg)
 {
  int err;
```

```
  unsigned short vid = 0;
@@ -728,7 +728,7 @@ static int vlan_ioctl_handler(void __user *arg)
  case GET_VLAN_REALDEV_NAME_CMD:
  case GET_VLAN_VID_CMD:
   err = -ENODEV;
-  dev = __dev_get_by_name(args.device1);
+  dev = __dev_get_by_name(&init_net, args.device1);
   if (!dev)
    goto out;

diff --git a/net/8021q/vlan_netlink.c b/net/8021q/vlan_netlink.c
index 6cdd1e0..0996185 100644
--- a/net/8021q/vlan_netlink.c
+++ b/net/8021q/vlan_netlink.c
@@ -11,6 +11,7 @@
 #include <linux/kernel.h>
 #include <linux/netdevice.h>
 #include <linux/if_vlan.h>
+#include <net/net_namespace.h>
 #include <net/netlink.h>
 #include <net/rtnetlink.h>
 #include "vlan.h"
@@ -112,7 +113,7 @@ static int vlan_newlink(struct net_device *dev,

  if (!tb[IFLA_LINK])
   return -EINVAL;
- real_dev = __dev_get_by_index(nla_get_u32(tb[IFLA_LINK]));
+ real_dev = __dev_get_by_index(&init_net, nla_get_u32(tb[IFLA_LINK]));
  if (!real_dev)
   return -ENODEV;

diff --git a/net/8021q/vlanproc.c b/net/8021q/vlanproc.c
index ac80e6b..6cefdf8 100644
--- a/net/8021q/vlanproc.c
+++ b/net/8021q/vlanproc.c
@@ -254,7 +254,7 @@ static void *vlan_seq_start(struct seq_file *seq, loff_t *pos)
  if (*pos == 0)
   return SEQ_START_TOKEN;

- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
   if (!is_vlan_dev(dev))
    continue;

@@ -273,9 +273,9 @@ static void *vlan_seq_next(struct seq_file *seq, void *v, loff_t *pos)

  dev = (struct net_device *)v;
  if (v == SEQ_START_TOKEN)
```

```
-  dev = net_device_entry(&dev_base_head);
+  dev = net_device_entry(&init_net.dev_base_head);

- for_each_netdev_continue(dev) {
+ for_each_netdev_continue(&init_net, dev) {
   if (!is_vlan_dev(dev))
    continue;

diff --git a/net/appletalk/ddp.c b/net/appletalk/ddp.c
index 36fcdbf..7c0b515 100644
--- a/net/appletalk/ddp.c
+++ b/net/appletalk/ddp.c
@@ -677,7 +677,7 @@ static int atif_ioctl(int cmd, void __user *arg)
  if (copy_from_user(&atreq, arg, sizeof(atreq)))
   return -EFAULT;

- dev = __dev_get_by_name(atreq.ifr_name);
+ dev = __dev_get_by_name(&init_net, atreq.ifr_name);
  if (!dev)
   return -ENODEV;

@@ -901,7 +901,7 @@ static int atrtr_ioctl(unsigned int cmd, void __user *arg)
     if (copy_from_user(name, rt.rt_dev, IFNAMSIZ-1))
      return -EFAULT;
     name[IFNAMSIZ-1] = '\0';
-    dev = __dev_get_by_name(name);
+    dev = __dev_get_by_name(&init_net, name);
     if (!dev)
      return -ENODEV;
    }
@@ -1273,7 +1273,7 @@ static __inline__ int is_ip_over_ddp(struct sk_buff *skb)

 static int handle_ip_over_ddp(struct sk_buff *skb)
 {
- struct net_device *dev = __dev_get_by_name("ipddp0");
+ struct net_device *dev = __dev_get_by_name(&init_net, "ipddp0");
  struct net_device_stats *stats;

  /* This needs to be able to handle ipddp"N" devices */
diff --git a/net/atm/mpc.c b/net/atm/mpc.c
index 0968430..2086396 100644
--- a/net/atm/mpc.c
+++ b/net/atm/mpc.c
@@ -244,7 +244,7 @@ static struct net_device *find_lec_by_itfnum(int itf)
  char name[IFNAMSIZ];

  sprintf(name, "lec%d", itf);
- dev = dev_get_by_name(name);
```

```
+ dev = dev_get_by_name(&init_net, name);

  return dev;
 }
diff --git a/net/ax25/af_ax25.c b/net/ax25/af_ax25.c
index 8d13a8b..993e5c7 100644
--- a/net/ax25/af_ax25.c
+++ b/net/ax25/af_ax25.c
@@ -631,7 +631,7 @@ static int ax25_setsockopt(struct socket *sock, int level, int optname,
    break;
   }

-  dev = dev_get_by_name(devname);
+  dev = dev_get_by_name(&init_net, devname);
   if (dev == NULL) {
    res = -ENODEV;
    break;
diff --git a/net/bridge/br_if.c b/net/bridge/br_if.c
index 749f0e8..f4847ab 100644
--- a/net/bridge/br_if.c
+++ b/net/bridge/br_if.c
@@ -303,7 +303,7 @@ int br_del_bridge(const char *name)
  int ret = 0;

  rtnl_lock();
- dev = __dev_get_by_name(name);
+ dev = __dev_get_by_name(&init_net, name);
  if (dev == NULL)
   ret =  -ENXIO;  /* Could not find device */

@@ -444,7 +444,7 @@ void __exit br_cleanup_bridges(void)
  struct net_device *dev, *nxt;

  rtnl_lock();
- for_each_netdev_safe(dev, nxt)
+ for_each_netdev_safe(&init_net, dev, nxt)
   if (dev->priv_flags & IFF_EBRIDGE)
    del_br(dev->priv);
  rtnl_unlock();
diff --git a/net/bridge/br_ioctl.c b/net/bridge/br_ioctl.c
index bb15e9e..0655a5f 100644
--- a/net/bridge/br_ioctl.c
+++ b/net/bridge/br_ioctl.c
@@ -18,6 +18,7 @@
 #include <linux/if_bridge.h>
 #include <linux/netdevice.h>
 #include <linux/times.h>
+#include <net/net_namespace.h>
```

```
 #include <asm/uaccess.h>
 #include "br_private.h"

@@ -27,7 +28,7 @@ static int get_bridge_ifindices(int *indices, int num)
  struct net_device *dev;
  int i = 0;

- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
   if (i >= num)
    break;
   if (dev->priv_flags & IFF_EBRIDGE)
@@ -90,7 +91,7 @@ static int add_del_if(struct net_bridge *br, int ifindex, int isadd)
  if (!capable(CAP_NET_ADMIN))
   return -EPERM;

- dev = dev_get_by_index(ifindex);
+ dev = dev_get_by_index(&init_net, ifindex);
  if (dev == NULL)
   return -EINVAL;

@@ -364,7 +365,7 @@ static int old_deviceless(void __user *uarg)
  return -EOPNOTSUPP;
 }

-int br_ioctl_deviceless_stub(unsigned int cmd, void __user *uarg)
+int br_ioctl_deviceless_stub(struct net *net, unsigned int cmd, void __user *uarg)
 {
  switch (cmd) {
  case SIOCGIFBR:
diff --git a/net/bridge/br_netlink.c b/net/bridge/br_netlink.c
index 0fcf6f0..53ab8e0 100644
--- a/net/bridge/br_netlink.c
+++ b/net/bridge/br_netlink.c
@@ -12,6 +12,7 @@

 #include <linux/kernel.h>
 #include <net/rtnetlink.h>
+#include <net/net_namespace.h>
 #include "br_private.h"

 static inline size_t br_nlmsg_size(void)
@@ -110,7 +111,7 @@ static int br_dump_ifinfo(struct sk_buff *skb, struct netlink_callback *cb)
  int idx;

  idx = 0;
- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
```

```
   /* not a bridge port */
   if (dev->br_port == NULL || idx < cb->args[0])
    goto skip;
@@ -155,7 +156,7 @@ static int br_rtm_setlink(struct sk_buff *skb,  struct nlmsghdr *nlh, void
*arg)
  if (new_state > BR_STATE_BLOCKING)
   return -EINVAL;

- dev = __dev_get_by_index(ifm->ifi_index);
+ dev = __dev_get_by_index(&init_net, ifm->ifi_index);
  if (!dev)
   return -ENODEV;

diff --git a/net/bridge/br_private.h b/net/bridge/br_private.h
index 21bf3a9..5ca6bd7 100644
--- a/net/bridge/br_private.h
+++ b/net/bridge/br_private.h
@@ -196,7 +196,7 @@ extern struct sk_buff *br_handle_frame(struct net_bridge_port *p,

 /* br_ioctl.c */
 extern int br_dev_ioctl(struct net_device *dev, struct ifreq *rq, int cmd);
-extern int br_ioctl_deviceless_stub(unsigned int cmd, void __user *arg);
+extern int br_ioctl_deviceless_stub(struct net *net, unsigned int cmd, void __user *arg);

 /* br_netfilter.c */
 #ifdef CONFIG_BRIDGE_NETFILTER
diff --git a/net/core/dev.c b/net/core/dev.c
index 316043b..c51cf40 100644
--- a/net/core/dev.c
+++ b/net/core/dev.c
@@ -190,25 +190,22 @@ static struct net_dma net_dma = {
  * unregister_netdevice(), which must be called with the rtnl
  * semaphore held.
  */
-LIST_HEAD(dev_base_head);
 DEFINE_RWLOCK(dev_base_lock);

-EXPORT_SYMBOL(dev_base_head);
 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];
+#define NETDEV_HASHENTRIES (1 << NETDEV_HASHBITS)

-static inline struct hlist_head *dev_name_hash(const char *name)
+static inline struct hlist_head *dev_name_hash(struct net *net, const char *name)
 {
```

```
  unsigned hash = full_name_hash(name, strnlen(name, IFNAMSIZ));
- return &dev_name_head[hash & ((1<<NETDEV_HASHBITS)-1)];
+ return &net->dev_name_head[hash & ((1 << NETDEV_HASHBITS) - 1)];
 }

-static inline struct hlist_head *dev_index_hash(int ifindex)
+static inline struct hlist_head *dev_index_hash(struct net *net, int ifindex)
 {
- return &dev_index_head[ifindex & ((1<<NETDEV_HASHBITS)-1)];
+ return &net->dev_index_head[ifindex & ((1 << NETDEV_HASHBITS) - 1)];
 }

 /*
@@ -492,7 +489,7 @@ unsigned long netdev_boot_base(const char *prefix, int unit)
   * If device already registered then return base of 1
   * to indicate not to probe for this interface
   */
- if (__dev_get_by_name(name))
+ if (__dev_get_by_name(&init_net, name))
   return 1;

  for (i = 0; i < NETDEV_BOOT_SETUP_MAX; i++)
@@ -547,11 +544,11 @@ __setup("netdev=", netdev_boot_setup);
  * careful with locks.
  */

-struct net_device *__dev_get_by_name(const char *name)
+struct net_device *__dev_get_by_name(struct net *net, const char *name)
 {
  struct hlist_node *p;

- hlist_for_each(p, dev_name_hash(name)) {
+ hlist_for_each(p, dev_name_hash(net, name)) {
   struct net_device *dev
    = hlist_entry(p, struct net_device, name_hlist);
   if (!strncmp(dev->name, name, IFNAMSIZ))
@@ -571,12 +568,12 @@ struct net_device *__dev_get_by_name(const char *name)
  * matching device is found.
  */

-struct net_device *dev_get_by_name(const char *name)
+struct net_device *dev_get_by_name(struct net *net, const char *name)
 {
  struct net_device *dev;

  read_lock(&dev_base_lock);
- dev = __dev_get_by_name(name);
+ dev = __dev_get_by_name(net, name);
```

```
  if (dev)
   dev_hold(dev);
  read_unlock(&dev_base_lock);
@@ -594,11 +591,11 @@ struct net_device *dev_get_by_name(const char *name)
  * or @dev_base_lock.
  */

-struct net_device *__dev_get_by_index(int ifindex)
+struct net_device *__dev_get_by_index(struct net *net, int ifindex)
 {
  struct hlist_node *p;

- hlist_for_each(p, dev_index_hash(ifindex)) {
+ hlist_for_each(p, dev_index_hash(net, ifindex)) {
   struct net_device *dev
    = hlist_entry(p, struct net_device, index_hlist);
   if (dev->ifindex == ifindex)
@@ -618,12 +615,12 @@ struct net_device *__dev_get_by_index(int ifindex)
  * dev_put to indicate they have finished with it.
  */

-struct net_device *dev_get_by_index(int ifindex)
+struct net_device *dev_get_by_index(struct net *net, int ifindex)
 {
  struct net_device *dev;

  read_lock(&dev_base_lock);
- dev = __dev_get_by_index(ifindex);
+ dev = __dev_get_by_index(net, ifindex);
  if (dev)
   dev_hold(dev);
  read_unlock(&dev_base_lock);
@@ -644,13 +641,13 @@ struct net_device *dev_get_by_index(int ifindex)
  * If the API was consistent this would be __dev_get_by_hwaddr
  */

-struct net_device *dev_getbyhwaddr(unsigned short type, char *ha)
+struct net_device *dev_getbyhwaddr(struct net *net, unsigned short type, char *ha)
 {
  struct net_device *dev;

  ASSERT_RTNL();

- for_each_netdev(dev)
+ for_each_netdev(&init_net, dev)
   if (dev->type == type &&
       !memcmp(dev->dev_addr, ha, dev->addr_len))
    return dev;
```

```
@@ -660,12 +657,12 @@ struct net_device *dev_getbyhwaddr(unsigned short type, char *ha)

 EXPORT_SYMBOL(dev_getbyhwaddr);

-struct net_device *__dev_getfirstbyhwtype(unsigned short type)
+struct net_device *__dev_getfirstbyhwtype(struct net *net, unsigned short type)
 {
  struct net_device *dev;

  ASSERT_RTNL();
- for_each_netdev(dev)
+ for_each_netdev(net, dev)
   if (dev->type == type)
    return dev;

@@ -674,12 +671,12 @@ struct net_device *__dev_getfirstbyhwtype(unsigned short type)

 EXPORT_SYMBOL(__dev_getfirstbyhwtype);

-struct net_device *dev_getfirstbyhwtype(unsigned short type)
+struct net_device *dev_getfirstbyhwtype(struct net *net, unsigned short type)
 {
  struct net_device *dev;

  rtnl_lock();
- dev = __dev_getfirstbyhwtype(type);
+ dev = __dev_getfirstbyhwtype(net, type);
  if (dev)
   dev_hold(dev);
  rtnl_unlock();
@@ -699,13 +696,13 @@ EXPORT_SYMBOL(dev_getfirstbyhwtype);
  * dev_put to indicate they have finished with it.
  */

-struct net_device * dev_get_by_flags(unsigned short if_flags, unsigned short mask)
+struct net_device * dev_get_by_flags(struct net *net, unsigned short if_flags, unsigned short
mask)
 {
  struct net_device *dev, *ret;

  ret = NULL;
  read_lock(&dev_base_lock);
- for_each_netdev(dev) {
+ for_each_netdev(net, dev) {
   if (((dev->flags ^ if_flags) & mask) == 0) {
    dev_hold(dev);
    ret = dev;
@@ -763,6 +760,10 @@ int dev_alloc_name(struct net_device *dev, const char *name)
```

```
  const int max_netdevices = 8*PAGE_SIZE;
  long *inuse;
  struct net_device *d;
+ struct net *net;
+
+ BUG_ON(!dev->nd_net);
+ net = dev->nd_net;

  p = strnchr(name, IFNAMSIZ-1, '%');
  if (p) {
@@ -779,7 +780,7 @@ int dev_alloc_name(struct net_device *dev, const char *name)
   if (!inuse)
    return -ENOMEM;

-  for_each_netdev(d) {
+  for_each_netdev(net, d) {
    if (!sscanf(d->name, name, &i))
     continue;
    if (i < 0 || i >= max_netdevices)
@@ -796,7 +797,7 @@ int dev_alloc_name(struct net_device *dev, const char *name)
  }

  snprintf(buf, sizeof(buf), name, i);
- if (!__dev_get_by_name(buf)) {
+ if (!__dev_get_by_name(net, buf)) {
   strlcpy(dev->name, buf, IFNAMSIZ);
   return i;
  }
@@ -822,9 +823,12 @@ int dev_change_name(struct net_device *dev, char *newname)
  char oldname[IFNAMSIZ];
  int err = 0;
  int ret;
+ struct net *net;

  ASSERT_RTNL();
+ BUG_ON(!dev->nd_net);

+ net = dev->nd_net;
  if (dev->flags & IFF_UP)
   return -EBUSY;

@@ -839,7 +843,7 @@ int dev_change_name(struct net_device *dev, char *newname)
    return err;
   strcpy(newname, dev->name);
  }
- else if (__dev_get_by_name(newname))
+ else if (__dev_get_by_name(net, newname))
   return -EEXIST;
```

```
  else
   strlcpy(dev->name, newname, IFNAMSIZ);
@@ -849,7 +853,7 @@ rollback:

  write_lock_bh(&dev_base_lock);
  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(net, dev->name));
  write_unlock_bh(&dev_base_lock);

  ret = raw_notifier_call_chain(&netdev_chain, NETDEV_CHANGENAME, dev);
@@ -908,12 +912,12 @@ void netdev_state_change(struct net_device *dev)
  * available in this kernel then it becomes a nop.
  */

-void dev_load(const char *name)
+void dev_load(struct net *net, const char *name)
 {
  struct net_device *dev;

  read_lock(&dev_base_lock);
- dev = __dev_get_by_name(name);
+ dev = __dev_get_by_name(net, name);
  read_unlock(&dev_base_lock);

  if (!dev && capable(CAP_SYS_MODULE))
@@ -1052,6 +1056,8 @@ int dev_close(struct net_device *dev)
 }


+static int dev_boot_phase = 1;
+
 /*
  * Device change register/unregister. These are not inline or static
  * as we export them to the world.
@@ -1075,23 +1081,27 @@ int register_netdevice_notifier(struct notifier_block *nb)
 {
  struct net_device *dev;
  struct net_device *last;
+ struct net *net;
  int err;

  rtnl_lock();
  err = raw_notifier_chain_register(&netdev_chain, nb);
  if (err)
   goto unlock;
+ if (dev_boot_phase)
+  goto unlock;
```

```
+ for_each_net(net) {
+  for_each_netdev(net, dev) {
+   err = nb->notifier_call(nb, NETDEV_REGISTER, dev);
+   err = notifier_to_errno(err);
+   if (err)
+    goto rollback;
+
+   if (!(dev->flags & IFF_UP))
+    continue;

- for_each_netdev(dev) {
-  err = nb->notifier_call(nb, NETDEV_REGISTER, dev);
-  err = notifier_to_errno(err);
-  if (err)
-   goto rollback;
-
-  if (!(dev->flags & IFF_UP))
-   continue;
-
-  nb->notifier_call(nb, NETDEV_UP, dev);
+   nb->notifier_call(nb, NETDEV_UP, dev);
+  }
  }

 unlock:
@@ -1100,15 +1110,17 @@ unlock:

 rollback:
  last = dev;
- for_each_netdev(dev) {
-  if (dev == last)
-   break;
+ for_each_net(net) {
+  for_each_netdev(net, dev) {
+   if (dev == last)
+    break;

-  if (dev->flags & IFF_UP) {
-   nb->notifier_call(nb, NETDEV_GOING_DOWN, dev);
-   nb->notifier_call(nb, NETDEV_DOWN, dev);
+   if (dev->flags & IFF_UP) {
+    nb->notifier_call(nb, NETDEV_GOING_DOWN, dev);
+    nb->notifier_call(nb, NETDEV_DOWN, dev);
+   }
+   nb->notifier_call(nb, NETDEV_UNREGISTER, dev);
   }
-  nb->notifier_call(nb, NETDEV_UNREGISTER, dev);
  }
```

```
  goto unlock;
 }
@@ -2169,7 +2181,7 @@ int register_gifconf(unsigned int family, gifconf_func_t * gifconf)
  * match.  --pb
  */

-static int dev_ifname(struct ifreq __user *arg)
+static int dev_ifname(struct net *net, struct ifreq __user *arg)
 {
  struct net_device *dev;
  struct ifreq ifr;
@@ -2182,7 +2194,7 @@ static int dev_ifname(struct ifreq __user *arg)
   return -EFAULT;

  read_lock(&dev_base_lock);
- dev = __dev_get_by_index(ifr.ifr_ifindex);
+ dev = __dev_get_by_index(net, ifr.ifr_ifindex);
  if (!dev) {
   read_unlock(&dev_base_lock);
   return -ENODEV;
@@ -2202,7 +2214,7 @@ static int dev_ifname(struct ifreq __user *arg)
  * Thus we will need a 'compatibility mode'.
  */

-static int dev_ifconf(char __user *arg)
+static int dev_ifconf(struct net *net, char __user *arg)
 {
  struct ifconf ifc;
  struct net_device *dev;
@@ -2226,7 +2238,7 @@ static int dev_ifconf(char __user *arg)
   */

  total = 0;
- for_each_netdev(dev) {
+ for_each_netdev(net, dev) {
   for (i = 0; i < NPROTO; i++) {
    if (gifconf_list[i]) {
     int done;
@@ -2260,6 +2272,7 @@ static int dev_ifconf(char __user *arg)
  */
 void *dev_seq_start(struct seq_file *seq, loff_t *pos)
 {
+ struct net *net = seq->private;
  loff_t off;
  struct net_device *dev;

@@ -2268,7 +2281,7 @@ void *dev_seq_start(struct seq_file *seq, loff_t *pos)
   return SEQ_START_TOKEN;
```

```
  off = 1;
- for_each_netdev(dev)
+ for_each_netdev(net, dev)
    if (off++ == *pos)
     return dev;

@@ -2277,9 +2290,10 @@ void *dev_seq_start(struct seq_file *seq, loff_t *pos)

 void *dev_seq_next(struct seq_file *seq, void *v, loff_t *pos)
 {
+ struct net *net = seq->private;
   ++*pos;
   return v == SEQ_START_TOKEN ?
-  first_net_device() : next_net_device((struct net_device *)v);
+  first_net_device(net) : next_net_device((struct net_device *)v);
 }

 void dev_seq_stop(struct seq_file *seq, void *v)
@@ -2375,7 +2389,22 @@ static const struct seq_operations dev_seq_ops = {

 static int dev_seq_open(struct inode *inode, struct file *file)
 {
- return seq_open(file, &dev_seq_ops);
+ struct seq_file *seq;
+ int res;
+ res =  seq_open(file, &dev_seq_ops);
+ if (!res) {
+  seq = file->private_data;
+  seq->private = get_net(PROC_NET(inode));
+ }
+ return res;
+}
+
+static int dev_seq_release(struct inode *inode, struct file *file)
+{
+ struct seq_file *seq = file->private_data;
+ struct net *net = seq->private;
+ put_net(net);
+ return seq_release(inode, file);
 }

 static const struct file_operations dev_seq_fops = {
@@ -2383,7 +2412,7 @@ static const struct file_operations dev_seq_fops = {
   .open    = dev_seq_open,
   .read    = seq_read,
   .llseek  = seq_lseek,
- .release = seq_release,
```

```
+ .release = dev_seq_release,
 };

 static const struct seq_operations softnet_seq_ops = {
@@ -2535,30 +2564,49 @@ static const struct file_operations ptype_seq_fops = {
 };


-static int __init dev_proc_init(void)
+static int dev_proc_net_init(struct net *net)
 {
  int rc = -ENOMEM;

- if (!proc_net_fops_create(&init_net, "dev", S_IRUGO, &dev_seq_fops))
+ if (!proc_net_fops_create(net, "dev", S_IRUGO, &dev_seq_fops))
    goto out;
- if (!proc_net_fops_create(&init_net, "softnet_stat", S_IRUGO, &softnet_seq_fops))
+ if (!proc_net_fops_create(net, "softnet_stat", S_IRUGO, &softnet_seq_fops))
    goto out_dev;
- if (!proc_net_fops_create(&init_net, "ptype", S_IRUGO, &ptype_seq_fops))
+ if (!proc_net_fops_create(net, "ptype", S_IRUGO, &ptype_seq_fops))
    goto out_softnet;

- if (wext_proc_init())
+ if (wext_proc_init(net))
    goto out_ptype;
  rc = 0;
 out:
  return rc;
 out_ptype:
- proc_net_remove(&init_net, "ptype");
+ proc_net_remove(net, "ptype");
 out_softnet:
- proc_net_remove(&init_net, "softnet_stat");
+ proc_net_remove(net, "softnet_stat");
 out_dev:
- proc_net_remove(&init_net, "dev");
+ proc_net_remove(net, "dev");
  goto out;
 }
+
+static void dev_proc_net_exit(struct net *net)
+{
+ wext_proc_exit(net);
+
+ proc_net_remove(net, "ptype");
+ proc_net_remove(net, "softnet_stat");
+ proc_net_remove(net, "dev");
```

```
+}
+
+static struct pernet_operations dev_proc_ops = {
+ .init = dev_proc_net_init,
+ .exit = dev_proc_net_exit,
+};
+
+static int __init dev_proc_init(void)
+{
+ return register_pernet_subsys(&dev_proc_ops);
+}
 #else
 #define dev_proc_init() 0
 #endif /* CONFIG_PROC_FS */
@@ -2993,10 +3041,10 @@ int dev_set_mac_address(struct net_device *dev, struct sockaddr
*sa)
 /*
  * Perform the SIOCxIFxxx calls.
  */
-static int dev_ifsioc(struct ifreq *ifr, unsigned int cmd)
+static int dev_ifsioc(struct net *net, struct ifreq *ifr, unsigned int cmd)
 {
  int err;
- struct net_device *dev = __dev_get_by_name(ifr->ifr_name);
+ struct net_device *dev = __dev_get_by_name(net, ifr->ifr_name);

  if (!dev)
   return -ENODEV;
@@ -3149,7 +3197,7 @@ static int dev_ifsioc(struct ifreq *ifr, unsigned int cmd)
  * positive or a negative errno code on error.
  */

-int dev_ioctl(unsigned int cmd, void __user *arg)
+int dev_ioctl(struct net *net, unsigned int cmd, void __user *arg)
 {
  struct ifreq ifr;
  int ret;
@@ -3162,12 +3210,12 @@ int dev_ioctl(unsigned int cmd, void __user *arg)

  if (cmd == SIOCGIFCONF) {
   rtnl_lock();
-  ret = dev_ifconf((char __user *) arg);
+  ret = dev_ifconf(net, (char __user *) arg);
   rtnl_unlock();
   return ret;
  }
  if (cmd == SIOCGIFNAME)
-  return dev_ifname((struct ifreq __user *)arg);
```

```
+  return dev_ifname(net, (struct ifreq __user *)arg);

  if (copy_from_user(&ifr, arg, sizeof(struct ifreq)))
   return -EFAULT;
@@ -3197,9 +3245,9 @@ int dev_ioctl(unsigned int cmd, void __user *arg)
   case SIOCGIFMAP:
   case SIOCGIFINDEX:
   case SIOCGIFTXQLEN:
-   dev_load(ifr.ifr_name);
+   dev_load(net, ifr.ifr_name);
    read_lock(&dev_base_lock);
-   ret = dev_ifsioc(&ifr, cmd);
+   ret = dev_ifsioc(net, &ifr, cmd);
    read_unlock(&dev_base_lock);
    if (!ret) {
     if (colon)
@@ -3211,9 +3259,9 @@ int dev_ioctl(unsigned int cmd, void __user *arg)
    return ret;

   case SIOCETHTOOL:
-   dev_load(ifr.ifr_name);
+   dev_load(net, ifr.ifr_name);
    rtnl_lock();
-   ret = dev_ethtool(&ifr);
+   ret = dev_ethtool(net, &ifr);
    rtnl_unlock();
    if (!ret) {
     if (colon)
@@ -3235,9 +3283,9 @@ int dev_ioctl(unsigned int cmd, void __user *arg)
    if (!capable(CAP_NET_ADMIN))
     return -EPERM;
-   dev_load(ifr.ifr_name);
+   dev_load(net, ifr.ifr_name);
    rtnl_lock();
-   ret = dev_ifsioc(&ifr, cmd);
+   ret = dev_ifsioc(net, &ifr, cmd);
    rtnl_unlock();
    if (!ret) {
     if (colon)
@@ -3276,9 +3324,9 @@ int dev_ioctl(unsigned int cmd, void __user *arg)
    /* fall through */
   case SIOCBONDSLAVEINFOQUERY:
   case SIOCBONDINFOQUERY:
-   dev_load(ifr.ifr_name);
+   dev_load(net, ifr.ifr_name);
    rtnl_lock();
-   ret = dev_ifsioc(&ifr, cmd);
```

```
+   ret = dev_ifsioc(net, &ifr, cmd);
    rtnl_unlock();
    return ret;

@@ -3298,9 +3346,9 @@ int dev_ioctl(unsigned int cmd, void __user *arg)
    if (cmd == SIOCWANDEV ||
        (cmd >= SIOCDEVPRIVATE &&
         cmd <= SIOCDEVPRIVATE + 15)) {
-    dev_load(ifr.ifr_name);
+    dev_load(net, ifr.ifr_name);
     rtnl_lock();
-    ret = dev_ifsioc(&ifr, cmd);
+    ret = dev_ifsioc(net, &ifr, cmd);
     rtnl_unlock();
     if (!ret && copy_to_user(arg, &ifr,
         sizeof(struct ifreq)))
@@ -3309,7 +3357,7 @@ int dev_ioctl(unsigned int cmd, void __user *arg)
    }
    /* Take care of Wireless Extensions */
    if (cmd >= SIOCIWFIRST && cmd <= SIOCIWLAST)
-    return wext_handle_ioctl(&ifr, cmd, arg);
+    return wext_handle_ioctl(net, &ifr, cmd, arg);
    return -EINVAL;
  }
 }
@@ -3322,19 +3370,17 @@ int dev_ioctl(unsigned int cmd, void __user *arg)
  * number.  The caller must hold the rtnl semaphore or the
  * dev_base_lock to be sure it remains unique.
  */
-static int dev_new_index(void)
+static int dev_new_index(struct net *net)
 {
  static int ifindex;
  for (;;) {
   if (++ifindex <= 0)
    ifindex = 1;
-  if (!__dev_get_by_index(ifindex))
+  if (!__dev_get_by_index(net, ifindex))
    return ifindex;
  }
 }

-static int dev_boot_phase = 1;
-
 /* Delayed registration/unregisteration */
 static DEFINE_SPINLOCK(net_todo_list_lock);
 static struct list_head net_todo_list = LIST_HEAD_INIT(net_todo_list);
@@ -3368,6 +3414,7 @@ int register_netdevice(struct net_device *dev)
```

```
  struct hlist_head *head;
  struct hlist_node *p;
  int ret;
+ struct net *net;

  BUG_ON(dev_boot_phase);
  ASSERT_RTNL();
@@ -3376,6 +3423,8 @@ int register_netdevice(struct net_device *dev)

  /* When net_device's are persistent, this will be fatal. */
  BUG_ON(dev->reg_state != NETREG_UNINITIALIZED);
+ BUG_ON(!dev->nd_net);
+ net = dev->nd_net;

  spin_lock_init(&dev->queue_lock);
  spin_lock_init(&dev->_xmit_lock);
@@ -3400,12 +3449,12 @@ int register_netdevice(struct net_device *dev)
   goto err_uninit;
  }

- dev->ifindex = dev_new_index();
+ dev->ifindex = dev_new_index(net);
  if (dev->iflink == -1)
   dev->iflink = dev->ifindex;

  /* Check for existence of name */
- head = dev_name_hash(dev->name);
+ head = dev_name_hash(net, dev->name);
  hlist_for_each(p, head) {
   struct net_device *d
    = hlist_entry(p, struct net_device, name_hlist);
@@ -3483,9 +3532,9 @@ int register_netdevice(struct net_device *dev)

  dev_init_scheduler(dev);
  write_lock_bh(&dev_base_lock);
- list_add_tail(&dev->dev_list, &dev_base_head);
+ list_add_tail(&dev->dev_list, &net->dev_base_head);
  hlist_add_head(&dev->name_hlist, head);
- hlist_add_head(&dev->index_hlist, dev_index_hash(dev->ifindex));
+ hlist_add_head(&dev->index_hlist, dev_index_hash(net, dev->ifindex));
  dev_hold(dev);
  write_unlock_bh(&dev_base_lock);

@@ -4049,6 +4098,45 @@ int netdev_compute_features(unsigned long all, unsigned long one)
 }
 EXPORT_SYMBOL(netdev_compute_features);

+/* Initialize per network namespace state */
```

```
+static int netdev_init(struct net *net)
+{
+ int i;
+ INIT_LIST_HEAD(&net->dev_base_head);
+ rwlock_init(&dev_base_lock);
+
+ net->dev_name_head = kmalloc(
+  sizeof(*net->dev_name_head)*NETDEV_HASHENTRIES, GFP_KERNEL);
+ if (!net->dev_name_head)
+  return -ENOMEM;
+
+ net->dev_index_head = kmalloc(
+  sizeof(*net->dev_index_head)*NETDEV_HASHENTRIES, GFP_KERNEL);
+ if (!net->dev_index_head) {
+  kfree(net->dev_name_head);
+  return -ENOMEM;
+ }
+
+ for (i = 0; i < NETDEV_HASHENTRIES; i++)
+  INIT_HLIST_HEAD(&net->dev_name_head[i]);
+
+ for (i = 0; i < NETDEV_HASHENTRIES; i++)
+  INIT_HLIST_HEAD(&net->dev_index_head[i]);
+
+ return 0;
+}
+
+static void netdev_exit(struct net *net)
+{
+ kfree(net->dev_name_head);
+ kfree(net->dev_index_head);
+}
+
+static struct pernet_operations netdev_net_ops = {
+ .init = netdev_init,
+ .exit = netdev_exit,
+};
+
 /*
  * Initialize the DEV module. At boot time this walks the device list and
  * unhooks any devices that fail to initialise (normally hardware not
@@ -4076,11 +4164,8 @@ static int __init net_dev_init(void)
  for (i = 0; i < 16; i++)
   INIT_LIST_HEAD(&ptype_base[i]);

- for (i = 0; i < ARRAY_SIZE(dev_name_head); i++)
-  INIT_HLIST_HEAD(&dev_name_head[i]);
-
```

```
- for (i = 0; i < ARRAY_SIZE(dev_index_head); i++)
-  INIT_HLIST_HEAD(&dev_index_head[i]);
+ if (register_pernet_subsys(&netdev_net_ops))
+  goto out;

  /*
   * Initialise the packet receive queues.
diff --git a/net/core/dev_mcast.c b/net/core/dev_mcast.c
index 8e069fc..1c4f619 100644
--- a/net/core/dev_mcast.c
+++ b/net/core/dev_mcast.c
@@ -187,11 +187,12 @@ EXPORT_SYMBOL(dev_mc_unsync);
 #ifdef CONFIG_PROC_FS
 static void *dev_mc_seq_start(struct seq_file *seq, loff_t *pos)
 {
+ struct net *net = seq->private;
  struct net_device *dev;
  loff_t off = 0;

  read_lock(&dev_base_lock);
- for_each_netdev(dev) {
+ for_each_netdev(net, dev) {
   if (off++ == *pos)
    return dev;
  }
@@ -240,7 +241,22 @@ static const struct seq_operations dev_mc_seq_ops = {

 static int dev_mc_seq_open(struct inode *inode, struct file *file)
 {
- return seq_open(file, &dev_mc_seq_ops);
+ struct seq_file *seq;
+ int res;
+ res = seq_open(file, &dev_mc_seq_ops);
+ if (!res) {
+  seq = file->private_data;
+  seq->private = get_net(PROC_NET(inode));
+ }
+ return res;
+}
+
+static int dev_mc_seq_release(struct inode *inode, struct file *file)
+{
+ struct seq_file *seq = file->private_data;
+ struct net *net = seq->private;
+ put_net(net);
+ return seq_release(inode, file);
 }
```

```
 static const struct file_operations dev_mc_seq_fops = {
@@ -248,14 +264,31 @@ static const struct file_operations dev_mc_seq_fops = {
  .open    = dev_mc_seq_open,
  .read    = seq_read,
  .llseek  = seq_lseek,
- .release = seq_release,
+ .release = dev_mc_seq_release,
 };

 #endif

+static int dev_mc_net_init(struct net *net)
+{
+ if (!proc_net_fops_create(net, "dev_mcast", 0, &dev_mc_seq_fops))
+  return -ENOMEM;
+ return 0;
+}
+
+static void dev_mc_net_exit(struct net *net)
+{
+ proc_net_remove(net, "dev_mcast");
+}
+
+static struct pernet_operations dev_mc_net_ops = {
+ .init = dev_mc_net_init,
+ .exit = dev_mc_net_exit,
+};
+
 void __init dev_mcast_init(void)
 {
- proc_net_fops_create(&init_net, "dev_mcast", 0, &dev_mc_seq_fops);
+ register_pernet_subsys(&dev_mc_net_ops);
 }

 EXPORT_SYMBOL(dev_mc_add);
diff --git a/net/core/ethtool.c b/net/core/ethtool.c
index 7c43f03..0d0b13c 100644
--- a/net/core/ethtool.c
+++ b/net/core/ethtool.c
@@ -779,9 +779,9 @@ static int ethtool_set_value(struct net_device *dev, char __user
*useraddr,

 /* The main entry point in this file.  Called from net/core/dev.c */

-int dev_ethtool(struct ifreq *ifr)
+int dev_ethtool(struct net *net, struct ifreq *ifr)
 {
- struct net_device *dev = __dev_get_by_name(ifr->ifr_name);
```

```
+ struct net_device *dev = __dev_get_by_name(net, ifr->ifr_name);
  void __user *useraddr = ifr->ifr_data;
  u32 ethcmd;
  int rc;
diff --git a/net/core/fib_rules.c b/net/core/fib_rules.c
index 9eabe1a..1ba71ba 100644
--- a/net/core/fib_rules.c
+++ b/net/core/fib_rules.c
@@ -12,6 +12,7 @@
 #include <linux/kernel.h>
 #include <linux/list.h>
 #include <net/net_namespace.h>
+#include <net/sock.h>
 #include <net/fib_rules.h>

 static LIST_HEAD(rules_ops);
@@ -198,6 +199,7 @@ errout:

 static int fib_nl_newrule(struct sk_buff *skb, struct nlmsghdr* nlh, void *arg)
 {
+ struct net *net = skb->sk->sk_net;
  struct fib_rule_hdr *frh = nlmsg_data(nlh);
  struct fib_rules_ops *ops = NULL;
  struct fib_rule *rule, *r, *last = NULL;
@@ -235,7 +237,7 @@ static int fib_nl_newrule(struct sk_buff *skb, struct nlmsghdr* nlh, void
*arg)

   rule->ifindex = -1;
   nla_strlcpy(rule->ifname, tb[FRA_IFNAME], IFNAMSIZ);
-  dev = __dev_get_by_name(rule->ifname);
+  dev = __dev_get_by_name(net, rule->ifname);
   if (dev)
    rule->ifindex = dev->ifindex;
  }
diff --git a/net/core/neighbour.c b/net/core/neighbour.c
index 5f25f4f..2c6577c 100644
--- a/net/core/neighbour.c
+++ b/net/core/neighbour.c
@@ -1441,6 +1441,7 @@ int neigh_table_clear(struct neigh_table *tbl)

 static int neigh_delete(struct sk_buff *skb, struct nlmsghdr *nlh, void *arg)
 {
+ struct net *net = skb->sk->sk_net;
  struct ndmsg *ndm;
  struct nlattr *dst_attr;
  struct neigh_table *tbl;
@@ -1456,7 +1457,7 @@ static int neigh_delete(struct sk_buff *skb, struct nlmsghdr *nlh, void
*arg)
```

```
  ndm = nlmsg_data(nlh);
  if (ndm->ndm_ifindex) {
-  dev = dev_get_by_index(ndm->ndm_ifindex);
+  dev = dev_get_by_index(net, ndm->ndm_ifindex);
    if (dev == NULL) {
     err = -ENODEV;
     goto out;
@@ -1506,6 +1507,7 @@ out:

 static int neigh_add(struct sk_buff *skb, struct nlmsghdr *nlh, void *arg)
 {
+ struct net *net = skb->sk->sk_net;
  struct ndmsg *ndm;
  struct nlattr *tb[NDA_MAX+1];
  struct neigh_table *tbl;
@@ -1522,7 +1524,7 @@ static int neigh_add(struct sk_buff *skb, struct nlmsghdr *nlh, void
*arg)

  ndm = nlmsg_data(nlh);
  if (ndm->ndm_ifindex) {
-  dev = dev_get_by_index(ndm->ndm_ifindex);
+  dev = dev_get_by_index(net, ndm->ndm_ifindex);
    if (dev == NULL) {
     err = -ENODEV;
     goto out;
diff --git a/net/core/netpoll.c b/net/core/netpoll.c
index 0952f93..bb7523a 100644
--- a/net/core/netpoll.c
+++ b/net/core/netpoll.c
@@ -653,7 +653,7 @@ int netpoll_setup(struct netpoll *np)
  int err;

  if (np->dev_name)
-  ndev = dev_get_by_name(np->dev_name);
+  ndev = dev_get_by_name(&init_net, np->dev_name);
  if (!ndev) {
   printk(KERN_ERR "%s: %s doesn't exist, aborting.\n",
          np->name, np->dev_name);
diff --git a/net/core/pktgen.c b/net/core/pktgen.c
index 21ae955..ba5338a 100644
--- a/net/core/pktgen.c
+++ b/net/core/pktgen.c
@@ -1999,7 +1999,7 @@ static int pktgen_setup_dev(struct pktgen_dev *pkt_dev, const char
*ifname)
   pkt_dev->odev = NULL;
  }
```

```
- odev = dev_get_by_name(ifname);
+ odev = dev_get_by_name(&init_net, ifname);
  if (!odev) {
   printk(KERN_ERR "pktgen: no such netdevice: \"%s\"\n", ifname);
   return -ENODEV;
diff --git a/net/core/rtnetlink.c b/net/core/rtnetlink.c
index 416768d..44f91bb 100644
--- a/net/core/rtnetlink.c
+++ b/net/core/rtnetlink.c
@@ -306,10 +306,13 @@ EXPORT_SYMBOL_GPL(rtnl_link_register);
 void __rtnl_link_unregister(struct rtnl_link_ops *ops)
 {
  struct net_device *dev, *n;
+ struct net *net;

- for_each_netdev_safe(dev, n) {
-  if (dev->rtnl_link_ops == ops)
-   ops->dellink(dev);
+ for_each_net(net) {
+  for_each_netdev_safe(net, dev, n) {
+   if (dev->rtnl_link_ops == ops)
+    ops->dellink(dev);
+  }
  }
  list_del(&ops->list);
 }
@@ -693,12 +696,13 @@ nla_put_failure:

 static int rtnl_dump_ifinfo(struct sk_buff *skb, struct netlink_callback *cb)
 {
+ struct net *net = skb->sk->sk_net;
  int idx;
  int s_idx = cb->args[0];
  struct net_device *dev;

  idx = 0;
- for_each_netdev(dev) {
+ for_each_netdev(net, dev) {
   if (idx < s_idx)
    goto cont;
   if (rtnl_fill_ifinfo(skb, dev, RTM_NEWLINK,
@@ -858,6 +862,7 @@ errout:

 static int rtnl_setlink(struct sk_buff *skb, struct nlmsghdr *nlh, void *arg)
 {
+ struct net *net = skb->sk->sk_net;
  struct ifinfomsg *ifm;
  struct net_device *dev;
```

```
  int err;
@@ -876,9 +881,9 @@ static int rtnl_setlink(struct sk_buff *skb, struct nlmsghdr *nlh, void *arg)
  err = -EINVAL;
  ifm = nlmsg_data(nlh);
  if (ifm->ifi_index > 0)
-  dev = dev_get_by_index(ifm->ifi_index);
+  dev = dev_get_by_index(net, ifm->ifi_index);
  else if (tb[IFLA_IFNAME])
-  dev = dev_get_by_name(ifname);
+  dev = dev_get_by_name(net, ifname);
  else
   goto errout;

@@ -904,6 +909,7 @@ errout:

 static int rtnl_dellink(struct sk_buff *skb, struct nlmsghdr *nlh, void *arg)
 {
+ struct net *net = skb->sk->sk_net;
  const struct rtnl_link_ops *ops;
  struct net_device *dev;
  struct ifinfomsg *ifm;
@@ -920,9 +926,9 @@ static int rtnl_dellink(struct sk_buff *skb, struct nlmsghdr *nlh, void *arg)

  ifm = nlmsg_data(nlh);
  if (ifm->ifi_index > 0)
-  dev = __dev_get_by_index(ifm->ifi_index);
+  dev = __dev_get_by_index(net, ifm->ifi_index);
  else if (tb[IFLA_IFNAME])
-  dev = __dev_get_by_name(ifname);
+  dev = __dev_get_by_name(net, ifname);
  else
   return -EINVAL;

@@ -937,7 +943,7 @@ static int rtnl_dellink(struct sk_buff *skb, struct nlmsghdr *nlh, void *arg)
  return 0;
 }

-struct net_device *rtnl_create_link(char *ifname,
+struct net_device *rtnl_create_link(struct net *net, char *ifname,
   const struct rtnl_link_ops *ops, struct nlattr *tb[])
 {
  int err;
@@ -954,6 +960,7 @@ struct net_device *rtnl_create_link(char *ifname,
    goto err_free;
  }

+ dev->nd_net = net;
  dev->rtnl_link_ops = ops;
```

```
  if (tb[IFLA_MTU])
@@ -981,6 +988,7 @@ err:

 static int rtnl_newlink(struct sk_buff *skb, struct nlmsghdr *nlh, void *arg)
 {
+ struct net *net = skb->sk->sk_net;
  const struct rtnl_link_ops *ops;
  struct net_device *dev;
  struct ifinfomsg *ifm;
@@ -1004,9 +1012,9 @@ replay:

  ifm = nlmsg_data(nlh);
  if (ifm->ifi_index > 0)
-  dev = __dev_get_by_index(ifm->ifi_index);
+  dev = __dev_get_by_index(net, ifm->ifi_index);
  else if (ifname[0])
-  dev = __dev_get_by_name(ifname);
+  dev = __dev_get_by_name(net, ifname);
  else
   dev = NULL;

@@ -1092,7 +1100,7 @@ replay:
   if (!ifname[0])
    snprintf(ifname, IFNAMSIZ, "%s%%d", ops->kind);

-  dev = rtnl_create_link(ifname, ops, tb);
+  dev = rtnl_create_link(net, ifname, ops, tb);

   if (IS_ERR(dev))
    err = PTR_ERR(dev);
@@ -1109,6 +1117,7 @@ replay:

 static int rtnl_getlink(struct sk_buff *skb, struct nlmsghdr* nlh, void *arg)
 {
+ struct net *net = skb->sk->sk_net;
  struct ifinfomsg *ifm;
  struct nlattr *tb[IFLA_MAX+1];
  struct net_device *dev = NULL;
@@ -1121,7 +1130,7 @@ static int rtnl_getlink(struct sk_buff *skb, struct nlmsghdr* nlh, void
*arg)

  ifm = nlmsg_data(nlh);
  if (ifm->ifi_index > 0) {
-  dev = dev_get_by_index(ifm->ifi_index);
+  dev = dev_get_by_index(net, ifm->ifi_index);
   if (dev == NULL)
    return -ENODEV;
```

```
  } else
diff --git a/net/core/sock.c b/net/core/sock.c
index 9feec0f..ff52c83 100644
--- a/net/core/sock.c
+++ b/net/core/sock.c
@@ -372,6 +372,7 @@ int sock_setsockopt(struct socket *sock, int level, int optname,
       char __user *optval, int optlen)
 {
  struct sock *sk=sock->sk;
+ struct net *net = sk->sk_net;
  struct sk_filter *filter;
  int val;
  int valbool;
@@ -613,7 +614,7 @@ set_rcvbuf:
    if (devname[0] == '\0') {
     sk->sk_bound_dev_if = 0;
    } else {
-    struct net_device *dev = dev_get_by_name(devname);
+    struct net_device *dev = dev_get_by_name(net, devname);
     if (!dev) {
      ret = -ENODEV;
      break;
diff --git a/net/decnet/af_decnet.c b/net/decnet/af_decnet.c
index 83398da..aabe98d 100644
--- a/net/decnet/af_decnet.c
+++ b/net/decnet/af_decnet.c
@@ -751,7 +751,7 @@ static int dn_bind(struct socket *sock, struct sockaddr *uaddr, int
addr_len)
   if (dn_ntohs(saddr->sdn_nodeaddrl)) {
    read_lock(&dev_base_lock);
    ldev = NULL;
-   for_each_netdev(dev) {
+   for_each_netdev(&init_net, dev) {
     if (!dev->dn_ptr)
      continue;
     if (dn_dev_islocal(dev, dn_saddr2dn(saddr))) {
diff --git a/net/decnet/dn_dev.c b/net/decnet/dn_dev.c
index 04f12e2..db25a97 100644
--- a/net/decnet/dn_dev.c
+++ b/net/decnet/dn_dev.c
@@ -513,7 +513,7 @@ int dn_dev_ioctl(unsigned int cmd, void __user *arg)
  ifr->ifr_name[IFNAMSIZ-1] = 0;

 #ifdef CONFIG_KMOD
- dev_load(ifr->ifr_name);
+ dev_load(&init_net, ifr->ifr_name);
 #endif
```

```
  switch(cmd) {
@@ -531,7 +531,7 @@ int dn_dev_ioctl(unsigned int cmd, void __user *arg)

  rtnl_lock();

- if ((dev = __dev_get_by_name(ifr->ifr_name)) == NULL) {
+ if ((dev = __dev_get_by_name(&init_net, ifr->ifr_name)) == NULL) {
    ret = -ENODEV;
    goto done;
  }
@@ -629,7 +629,7 @@ static struct dn_dev *dn_dev_by_index(int ifindex)
 {
  struct net_device *dev;
  struct dn_dev *dn_dev = NULL;
- dev = dev_get_by_index(ifindex);
+ dev = dev_get_by_index(&init_net, ifindex);
  if (dev) {
    dn_dev = dev->dn_ptr;
    dev_put(dev);
@@ -694,7 +694,7 @@ static int dn_nl_newaddr(struct sk_buff *skb, struct nlmsghdr *nlh, void
*arg)
    return -EINVAL;

  ifm = nlmsg_data(nlh);
- if ((dev = __dev_get_by_index(ifm->ifa_index)) == NULL)
+ if ((dev = __dev_get_by_index(&init_net, ifm->ifa_index)) == NULL)
    return -ENODEV;

  if ((dn_db = dev->dn_ptr) == NULL) {
@@ -800,7 +800,7 @@ static int dn_nl_dump_ifaddr(struct sk_buff *skb, struct netlink_callback
*cb)
  skip_naddr = cb->args[1];

  idx = 0;
- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
    if (idx < skip_ndevs)
      goto cont;
    else if (idx > skip_ndevs) {
@@ -1297,7 +1297,7 @@ void dn_dev_devices_off(void)
  struct net_device *dev;

  rtnl_lock();
- for_each_netdev(dev)
+ for_each_netdev(&init_net, dev)
    dn_dev_down(dev);
  rtnl_unlock();
```

```
@@ -1308,7 +1308,7 @@ void dn_dev_devices_on(void)
  struct net_device *dev;

  rtnl_lock();
- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
   if (dev->flags & IFF_UP)
    dn_dev_up(dev);
  }
@@ -1342,7 +1342,7 @@ static void *dn_dev_seq_start(struct seq_file *seq, loff_t *pos)
   return SEQ_START_TOKEN;

  i = 1;
- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
   if (!is_dn_dev(dev))
    continue;

@@ -1361,9 +1361,9 @@ static void *dn_dev_seq_next(struct seq_file *seq, void *v, loff_t *pos)

  dev = (struct net_device *)v;
  if (v == SEQ_START_TOKEN)
-  dev = net_device_entry(&dev_base_head);
+  dev = net_device_entry(&init_net.dev_base_head);

- for_each_netdev_continue(dev) {
+ for_each_netdev_continue(&init_net, dev) {
   if (!is_dn_dev(dev))
    continue;

diff --git a/net/decnet/dn_fib.c b/net/decnet/dn_fib.c
index d2bc19d..3760a20 100644
--- a/net/decnet/dn_fib.c
+++ b/net/decnet/dn_fib.c
@@ -212,7 +212,7 @@ static int dn_fib_check_nh(const struct rtmsg *r, struct dn_fib_info *fi,
struct
     return -EINVAL;
    if (dnet_addr_type(nh->nh_gw) != RTN_UNICAST)
     return -EINVAL;
-   if ((dev = __dev_get_by_index(nh->nh_oif)) == NULL)
+   if ((dev = __dev_get_by_index(&init_net, nh->nh_oif)) == NULL)
     return -ENODEV;
    if (!(dev->flags&IFF_UP))
     return -ENETDOWN;
@@ -255,7 +255,7 @@ out:
   if (nh->nh_flags&(RTNH_F_PERVASIVE|RTNH_F_ONLINK))
    return -EINVAL;
```

```
-  dev = __dev_get_by_index(nh->nh_oif);
+  dev = __dev_get_by_index(&init_net, nh->nh_oif);
   if (dev == NULL || dev->dn_ptr == NULL)
    return -ENODEV;
   if (!(dev->flags&IFF_UP))
@@ -355,7 +355,7 @@ struct dn_fib_info *dn_fib_create_info(const struct rtmsg *r, struct
dn_kern_rta
   if (nhs != 1 || nh->nh_gw)
    goto err_inval;
   nh->nh_scope = RT_SCOPE_NOWHERE;
-  nh->nh_dev = dev_get_by_index(fi->fib_nh->nh_oif);
+  nh->nh_dev = dev_get_by_index(&init_net, fi->fib_nh->nh_oif);
   err = -ENODEV;
   if (nh->nh_dev == NULL)
    goto failure;
@@ -602,7 +602,7 @static void dn_fib_del_ifaddr(struct dn_ifaddr *ifa)

  /* Scan device list */
  read_lock(&dev_base_lock);
- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
   dn_db = dev->dn_ptr;
   if (dn_db == NULL)
    continue;
diff --git a/net/decnet/dn_route.c b/net/decnet/dn_route.c
index 580e786..70b1c3f 100644
--- a/net/decnet/dn_route.c
+++ b/net/decnet/dn_route.c
@@ -908,7 +908,7 @@ static int dn_route_output_slow(struct dst_entry **pprt, const struct flowi
*old

  /* If we have an output interface, verify its a DECnet device */
  if (oldflp->oif) {
-  dev_out = dev_get_by_index(oldflp->oif);
+  dev_out = dev_get_by_index(&init_net, oldflp->oif);
   err = -ENODEV;
   if (dev_out && dev_out->dn_ptr == NULL) {
    dev_put(dev_out);
@@ -929,7 +929,7 @@ static int dn_route_output_slow(struct dst_entry **pprt, const struct flowi
*old
    goto out;
   }
   read_lock(&dev_base_lock);
-  for_each_netdev(dev) {
+  for_each_netdev(&init_net, dev) {
    if (!dev->dn_ptr)
     continue;
    if (!dn_dev_islocal(dev, oldflp->fld_src))
```

```
@@ -1556,7 +1556,7 @@ static int dn_cache_getroute(struct sk_buff *in_skb, struct nlmsghdr
*nlh, void

  if (fl.iif) {
   struct net_device *dev;
-  if ((dev = dev_get_by_index(fl.iif)) == NULL) {
+  if ((dev = dev_get_by_index(&init_net, fl.iif)) == NULL) {
    kfree_skb(skb);
    return -ENODEV;
   }
diff --git a/net/decnet/sysctl_net_decnet.c b/net/decnet/sysctl_net_decnet.c
index 52e40d7..ae354a4 100644
--- a/net/decnet/sysctl_net_decnet.c
+++ b/net/decnet/sysctl_net_decnet.c
@@ -259,7 +259,7 @@ static int dn_def_dev_strategy(ctl_table *table, int __user *name, int
nlen,

   devname[newlen] = 0;

-  dev = dev_get_by_name(devname);
+  dev = dev_get_by_name(&init_net, devname);
   if (dev == NULL)
    return -ENODEV;

@@ -299,7 +299,7 @@ static int dn_def_dev_handler(ctl_table *table, int write,
   devname[*lenp] = 0;
   strip_it(devname);

-  dev = dev_get_by_name(devname);
+  dev = dev_get_by_name(&init_net, devname);
   if (dev == NULL)
    return -ENODEV;

diff --git a/net/econet/af_econet.c b/net/econet/af_econet.c
index f877f3b..9938e76 100644
--- a/net/econet/af_econet.c
+++ b/net/econet/af_econet.c
@@ -662,7 +662,7 @@ static int ec_dev_ioctl(struct socket *sock, unsigned int cmd, void __user
*arg)
  if (copy_from_user(&ifr, arg, sizeof(struct ifreq)))
   return -EFAULT;

- if ((dev = dev_get_by_name(ifr.ifr_name)) == NULL)
+ if ((dev = dev_get_by_name(&init_net, ifr.ifr_name)) == NULL)
   return -ENODEV;

  sec = (struct sockaddr_ec *)&ifr.ifr_addr;
diff --git a/net/ipv4/arp.c b/net/ipv4/arp.c
```

```
index a11e7a5..3a68300 100644
--- a/net/ipv4/arp.c
+++ b/net/ipv4/arp.c
@@ -981,7 +981,7 @@ static int arp_req_set(struct arpreq *r, struct net_device * dev)
   if (mask && mask != htonl(0xFFFFFFFF))
    return -EINVAL;
   if (!dev && (r->arp_flags & ATF_COM)) {
-   dev = dev_getbyhwaddr(r->arp_ha.sa_family, r->arp_ha.sa_data);
+   dev = dev_getbyhwaddr(&init_net, r->arp_ha.sa_family, r->arp_ha.sa_data);
    if (!dev)
     return -ENODEV;
   }
@@ -1169,7 +1169,7 @@ int arp_ioctl(unsigned int cmd, void __user *arg)
  rtnl_lock();
  if (r.arp_dev[0]) {
   err = -ENODEV;
-  if ((dev = __dev_get_by_name(r.arp_dev)) == NULL)
+  if ((dev = __dev_get_by_name(&init_net, r.arp_dev)) == NULL)
    goto out;

   /* Mmmm... It is wrong... ARPHRD_NETROM==0 */
diff --git a/net/ipv4/devinet.c b/net/ipv4/devinet.c
index 9996b57..3db853b 100644
--- a/net/ipv4/devinet.c
+++ b/net/ipv4/devinet.c
@@ -420,7 +420,7 @@ struct in_device *inetdev_by_index(int ifindex)
  struct net_device *dev;
  struct in_device *in_dev = NULL;
  read_lock(&dev_base_lock);
- dev = __dev_get_by_index(ifindex);
+ dev = __dev_get_by_index(&init_net, ifindex);
  if (dev)
   in_dev = in_dev_get(dev);
  read_unlock(&dev_base_lock);
@@ -506,7 +506,7 @@ static struct in_ifaddr *rtm_to_ifaddr(struct nlmsghdr *nlh)
   goto errout;
  }

- dev = __dev_get_by_index(ifm->ifa_index);
+ dev = __dev_get_by_index(&init_net, ifm->ifa_index);
  if (dev == NULL) {
   err = -ENODEV;
   goto errout;
@@ -628,7 +628,7 @@ int devinet_ioctl(unsigned int cmd, void __user *arg)
   *colon = 0;

 #ifdef CONFIG_KMOD
- dev_load(ifr.ifr_name);
```

```
+ dev_load(&init_net, ifr.ifr_name);
 #endif

  switch (cmd) {
@@ -669,7 +669,7 @@ int devinet_ioctl(unsigned int cmd, void __user *arg)
  rtnl_lock();

  ret = -ENODEV;
- if ((dev = __dev_get_by_name(ifr.ifr_name)) == NULL)
+ if ((dev = __dev_get_by_name(&init_net, ifr.ifr_name)) == NULL)
   goto done;

  if (colon)
@@ -909,7 +909,7 @@ no_in_dev:
   */
  read_lock(&dev_base_lock);
  rcu_read_lock();
- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
   if ((in_dev = __in_dev_get_rcu(dev)) == NULL)
    continue;

@@ -988,7 +988,7 @@ __be32 inet_confirm_addr(const struct net_device *dev, __be32 dst,
__be32 local,

  read_lock(&dev_base_lock);
  rcu_read_lock();
- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
   if ((in_dev = __in_dev_get_rcu(dev))) {
    addr = confirm_addr_indev(in_dev, dst, local, scope);
    if (addr)
@@ -1185,7 +1185,7 @@ static int inet_dump_ifaddr(struct sk_buff *skb, struct netlink_callback
*cb)

  s_ip_idx = ip_idx = cb->args[1];
  idx = 0;
- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
   if (idx < s_idx)
    goto cont;
   if (idx > s_idx)
@@ -1244,7 +1244,7 @@ static void devinet_copy_dflt_conf(int i)
  struct net_device *dev;

  read_lock(&dev_base_lock);
- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
```

```
   struct in_device *in_dev;
   rcu_read_lock();
   in_dev = __in_dev_get_rcu(dev);
@@ -1333,7 +1333,7 @@ void inet_forward_change(void)
  IPV4_DEVCONF_DFLT(FORWARDING) = on;

  read_lock(&dev_base_lock);
- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
   struct in_device *in_dev;
   rcu_read_lock();
   in_dev = __in_dev_get_rcu(dev);
diff --git a/net/ipv4/fib_frontend.c b/net/ipv4/fib_frontend.c
index 140bf7a..df17aab 100644
--- a/net/ipv4/fib_frontend.c
+++ b/net/ipv4/fib_frontend.c
@@ -334,7 +334,7 @@ static int rtentry_to_fib_config(int cmd, struct rtentry *rt,
   colon = strchr(devname, ':');
   if (colon)
    *colon = 0;
-  dev = __dev_get_by_name(devname);
+  dev = __dev_get_by_name(&init_net, devname);
   if (!dev)
    return -ENODEV;
   cfg->fc_oif = dev->ifindex;
diff --git a/net/ipv4/fib_semantics.c b/net/ipv4/fib_semantics.c
index c434119..d30fb68 100644
--- a/net/ipv4/fib_semantics.c
+++ b/net/ipv4/fib_semantics.c
@@ -533,7 +533,7 @@ static int fib_check_nh(struct fib_config *cfg, struct fib_info *fi,
     return -EINVAL;
    if (inet_addr_type(nh->nh_gw) != RTN_UNICAST)
     return -EINVAL;
-   if ((dev = __dev_get_by_index(nh->nh_oif)) == NULL)
+   if ((dev = __dev_get_by_index(&init_net, nh->nh_oif)) == NULL)
     return -ENODEV;
    if (!(dev->flags&IFF_UP))
     return -ENETDOWN;
@@ -799,7 +799,7 @@ struct fib_info *fib_create_info(struct fib_config *cfg)
   if (nhs != 1 || nh->nh_gw)
    goto err_inval;
   nh->nh_scope = RT_SCOPE_NOWHERE;
-  nh->nh_dev = dev_get_by_index(fi->fib_nh->nh_oif);
+  nh->nh_dev = dev_get_by_index(&init_net, fi->fib_nh->nh_oif);
   err = -ENODEV;
   if (nh->nh_dev == NULL)
    goto failure;
diff --git a/net/ipv4/icmp.c b/net/ipv4/icmp.c
```

```
index 02a899b..68a2267 100644
--- a/net/ipv4/icmp.c
+++ b/net/ipv4/icmp.c
@@ -517,7 +517,7 @@ void icmp_send(struct sk_buff *skb_in, int type, int code, __be32 info)
   struct net_device *dev = NULL;

   if (rt->fl.iif && sysctl_icmp_errors_use_inbound_ifaddr)
-   dev = dev_get_by_index(rt->fl.iif);
+   dev = dev_get_by_index(&init_net, rt->fl.iif);

   if (dev) {
    saddr = inet_select_addr(dev, 0, RT_SCOPE_LINK);
diff --git a/net/ipv4/igmp.c b/net/ipv4/igmp.c
index d78599a..ad500a4 100644
--- a/net/ipv4/igmp.c
+++ b/net/ipv4/igmp.c
@@ -2292,7 +2292,7 @@ static inline struct ip_mc_list *igmp_mc_get_first(struct seq_file *seq)
  struct igmp_mc_iter_state *state = igmp_mc_seq_private(seq);

  state->in_dev = NULL;
- for_each_netdev(state->dev) {
+ for_each_netdev(&init_net, state->dev) {
   struct in_device *in_dev;
   in_dev = in_dev_get(state->dev);
   if (!in_dev)
@@ -2454,7 +2454,7 @@ static inline struct ip_sf_list *igmp_mcf_get_first(struct seq_file *seq)

  state->idev = NULL;
  state->im = NULL;
- for_each_netdev(state->dev) {
+ for_each_netdev(&init_net, state->dev) {
   struct in_device *idev;
   idev = in_dev_get(state->dev);
   if (unlikely(idev == NULL))
diff --git a/net/ipv4/ip_fragment.c b/net/ipv4/ip_fragment.c
index 0231bdc..fabb86d 100644
--- a/net/ipv4/ip_fragment.c
+++ b/net/ipv4/ip_fragment.c
@@ -292,7 +292,7 @@ static void ip_expire(unsigned long arg)
  if ((qp->last_in&FIRST_IN) && qp->fragments != NULL) {
   struct sk_buff *head = qp->fragments;
   /* Send an ICMP "Fragment Reassembly Timeout" message. */
-  if ((head->dev = dev_get_by_index(qp->iif)) != NULL) {
+  if ((head->dev = dev_get_by_index(&init_net, qp->iif)) != NULL) {
    icmp_send(head, ICMP_TIME_EXCEEDED, ICMP_EXC_FRAGTIME, 0);
    dev_put(head->dev);
   }
diff --git a/net/ipv4/ip_gre.c b/net/ipv4/ip_gre.c
```

```
index 5c14ed6..3106225 100644
--- a/net/ipv4/ip_gre.c
+++ b/net/ipv4/ip_gre.c
@@ -262,7 +262,7 @@ static struct ip_tunnel * ipgre_tunnel_locate(struct ip_tunnel_parm
*parms, int
   int i;
   for (i=1; i<100; i++) {
    sprintf(name, "gre%d", i);
-   if (__dev_get_by_name(name) == NULL)
+   if (__dev_get_by_name(&init_net, name) == NULL)
     break;
   }
   if (i==100)
@@ -1196,7 +1196,7 @@ static int ipgre_tunnel_init(struct net_device *dev)
  }

  if (!tdev && tunnel->parms.link)
-  tdev = __dev_get_by_index(tunnel->parms.link);
+  tdev = __dev_get_by_index(&init_net, tunnel->parms.link);

  if (tdev) {
   hlen = tdev->hard_header_len;
diff --git a/net/ipv4/ip_sockglue.c b/net/ipv4/ip_sockglue.c
index 6b420ae..b2b3053 100644
--- a/net/ipv4/ip_sockglue.c
+++ b/net/ipv4/ip_sockglue.c
@@ -602,7 +602,7 @@ static int do_ip_setsockopt(struct sock *sk, int level,
     dev_put(dev);
    }
   } else
-   dev = __dev_get_by_index(mreq.imr_ifindex);
+   dev = __dev_get_by_index(&init_net, mreq.imr_ifindex);


   err = -EADDRNOTAVAIL;
diff --git a/net/ipv4/ipconfig.c b/net/ipv4/ipconfig.c
index 08ff623..4303851 100644
--- a/net/ipv4/ipconfig.c
+++ b/net/ipv4/ipconfig.c
@@ -193,7 +193,7 @@ static int __init ic_open_devs(void)
  if (dev_change_flags(&loopback_dev, loopback_dev.flags | IFF_UP) < 0)
   printk(KERN_ERR "IP-Config: Failed to open %s\n", loopback_dev.name);

- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
   if (dev == &loopback_dev)
    continue;
   if (user_dev_name[0] ? !strcmp(dev->name, user_dev_name) :
```

```
diff --git a/net/ipv4/ipip.c b/net/ipv4/ipip.c
index 3964372..652bd86 100644
--- a/net/ipv4/ipip.c
+++ b/net/ipv4/ipip.c
@@ -225,7 +225,7 @@ static struct ip_tunnel * ipip_tunnel_locate(struct ip_tunnel_parm *parms,
int c
   int i;
   for (i=1; i<100; i++) {
    sprintf(name, "tunl%d", i);
-   if (__dev_get_by_name(name) == NULL)
+   if (__dev_get_by_name(&init_net, name) == NULL)
     break;
   }
   if (i==100)
@@ -822,7 +822,7 @@ static int ipip_tunnel_init(struct net_device *dev)
  }

  if (!tdev && tunnel->parms.link)
-  tdev = __dev_get_by_index(tunnel->parms.link);
+  tdev = __dev_get_by_index(&init_net, tunnel->parms.link);

  if (tdev) {
   dev->hard_header_len = tdev->hard_header_len + sizeof(struct iphdr);
diff --git a/net/ipv4/ipmr.c b/net/ipv4/ipmr.c
index 0365988..b8b4b49 100644
--- a/net/ipv4/ipmr.c
+++ b/net/ipv4/ipmr.c
@@ -125,7 +125,7 @@ struct net_device *ipmr_new_tunnel(struct vifctl *v)
 {
  struct net_device  *dev;

- dev = __dev_get_by_name("tunl0");
+ dev = __dev_get_by_name(&init_net, "tunl0");

  if (dev) {
   int err;
@@ -149,7 +149,7 @@ struct net_device *ipmr_new_tunnel(struct vifctl *v)

   dev = NULL;

-  if (err == 0 && (dev = __dev_get_by_name(p.name)) != NULL) {
+  if (err == 0 && (dev = __dev_get_by_name(&init_net, p.name)) != NULL) {
    dev->flags |= IFF_MULTICAST;

    in_dev = __in_dev_get_rtnl(dev);
diff --git a/net/ipv4/ipvs/ip_vs_sync.c b/net/ipv4/ipvs/ip_vs_sync.c
index 356f067..1960747 100644
--- a/net/ipv4/ipvs/ip_vs_sync.c
```

```
+++ b/net/ipv4/ipvs/ip_vs_sync.c
@@ -387,7 +387,7 @@ static int set_mcast_if(struct sock *sk, char *ifname)
  struct net_device *dev;
  struct inet_sock *inet = inet_sk(sk);

- if ((dev = __dev_get_by_name(ifname)) == NULL)
+ if ((dev = __dev_get_by_name(&init_net, ifname)) == NULL)
   return -ENODEV;

  if (sk->sk_bound_dev_if && dev->ifindex != sk->sk_bound_dev_if)
@@ -412,7 +412,7 @@ static int set_sync_mesg_maxlen(int sync_state)
  int num;

  if (sync_state == IP_VS_STATE_MASTER) {
-  if ((dev = __dev_get_by_name(ip_vs_master_mcast_ifn)) == NULL)
+  if ((dev = __dev_get_by_name(&init_net, ip_vs_master_mcast_ifn)) == NULL)
    return -ENODEV;

   num = (dev->mtu - sizeof(struct iphdr) -
@@ -423,7 +423,7 @@ static int set_sync_mesg_maxlen(int sync_state)
   IP_VS_DBG(7, "setting the maximum length of sync sending "
     "message %d.\n", sync_send_mesg_maxlen);
  } else if (sync_state == IP_VS_STATE_BACKUP) {
-  if ((dev = __dev_get_by_name(ip_vs_backup_mcast_ifn)) == NULL)
+  if ((dev = __dev_get_by_name(&init_net, ip_vs_backup_mcast_ifn)) == NULL)
    return -ENODEV;

   sync_recv_mesg_maxlen = dev->mtu -
@@ -451,7 +451,7 @@ join_mcast_group(struct sock *sk, struct in_addr *addr, char *ifname)
  memset(&mreq, 0, sizeof(mreq));
  memcpy(&mreq.imr_multiaddr, addr, sizeof(struct in_addr));

- if ((dev = __dev_get_by_name(ifname)) == NULL)
+ if ((dev = __dev_get_by_name(&init_net, ifname)) == NULL)
   return -ENODEV;
  if (sk->sk_bound_dev_if && dev->ifindex != sk->sk_bound_dev_if)
   return -EINVAL;
@@ -472,7 +472,7 @@ static int bind_mcastif_addr(struct socket *sock, char *ifname)
  __be32 addr;
  struct sockaddr_in sin;

- if ((dev = __dev_get_by_name(ifname)) == NULL)
+ if ((dev = __dev_get_by_name(&init_net, ifname)) == NULL)
   return -ENODEV;

  addr = inet_select_addr(dev, 0, RT_SCOPE_UNIVERSE);
diff --git a/net/ipv4/netfilter/ipt_CLUSTERIP.c b/net/ipv4/netfilter/ipt_CLUSTERIP.c
index 50fc9e0..27f14e1 100644
```

```
--- a/net/ipv4/netfilter/ipt_CLUSTERIP.c
+++ b/net/ipv4/netfilter/ipt_CLUSTERIP.c
@@ -401,7 +401,7 @@ checkentry(const char *tablename,
     return false;
    }

-   dev = dev_get_by_name(e->ip.iniface);
+   dev = dev_get_by_name(&init_net, e->ip.iniface);
    if (!dev) {
     printk(KERN_WARNING "CLUSTERIP: no such interface %s\n", e->ip.iniface);
     return false;
diff --git a/net/ipv4/route.c b/net/ipv4/route.c
index efd2a92..396c631 100644
--- a/net/ipv4/route.c
+++ b/net/ipv4/route.c
@@ -2213,7 +2213,7 @@ static int ip_route_output_slow(struct rtable **rp, const struct flowi
*oldflp)


  if (oldflp->oif) {
-  dev_out = dev_get_by_index(oldflp->oif);
+  dev_out = dev_get_by_index(&init_net, oldflp->oif);
   err = -ENODEV;
   if (dev_out == NULL)
    goto out;
@@ -2592,7 +2592,7 @@ static int inet_rtm_getroute(struct sk_buff *in_skb, struct nlmsghdr*
nlh, void
  if (iif) {
   struct net_device *dev;

-  dev = __dev_get_by_index(iif);
+  dev = __dev_get_by_index(&init_net, iif);
   if (dev == NULL) {
    err = -ENODEV;
    goto errout_free;
diff --git a/net/ipv6/addrconf.c b/net/ipv6/addrconf.c
index 5b191a3..6de13c1 100644
--- a/net/ipv6/addrconf.c
+++ b/net/ipv6/addrconf.c
@@ -450,7 +450,7 @@ static void addrconf_forward_change(void)
  struct inet6_dev *idev;

  read_lock(&dev_base_lock);
- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
   rcu_read_lock();
   idev = __in6_dev_get(dev);
   if (idev) {
```

```
@@ -912,7 +912,7 @@ int ipv6_dev_get_saddr(struct net_device *daddr_dev,
  read_lock(&dev_base_lock);
  rcu_read_lock();

- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
   struct inet6_dev *idev;
   struct inet6_ifaddr *ifa;

@@ -1858,7 +1858,7 @@ int addrconf_set_dstaddr(void __user *arg)
  if (copy_from_user(&ireq, arg, sizeof(struct in6_ifreq)))
   goto err_exit;

- dev = __dev_get_by_index(ireq.ifr6_ifindex);
+ dev = __dev_get_by_index(&init_net, ireq.ifr6_ifindex);

  err = -ENODEV;
  if (dev == NULL)
@@ -1889,7 +1889,7 @@ int addrconf_set_dstaddr(void __user *arg)

   if (err == 0) {
    err = -ENOBUFS;
-   if ((dev = __dev_get_by_name(p.name)) == NULL)
+   if ((dev = __dev_get_by_name(&init_net, p.name)) == NULL)
     goto err_exit;
    err = dev_open(dev);
   }
@@ -1919,7 +1919,7 @@ static int inet6_addr_add(int ifindex, struct in6_addr *pfx, int plen,
  if (!valid_lft || prefered_lft > valid_lft)
   return -EINVAL;

- if ((dev = __dev_get_by_index(ifindex)) == NULL)
+ if ((dev = __dev_get_by_index(&init_net, ifindex)) == NULL)
   return -ENODEV;

  if ((idev = addrconf_add_dev(dev)) == NULL)
@@ -1970,7 +1970,7 @@ static int inet6_addr_del(int ifindex, struct in6_addr *pfx, int plen)
  struct inet6_dev *idev;
  struct net_device *dev;

- if ((dev = __dev_get_by_index(ifindex)) == NULL)
+ if ((dev = __dev_get_by_index(&init_net, ifindex)) == NULL)
   return -ENODEV;

  if ((idev = __in6_dev_get(dev)) == NULL)
@@ -2065,7 +2065,7 @@ static void sit_add_v4_addrs(struct inet6_dev *idev)
   return;
  }
```

```
- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
   struct in_device * in_dev = __in_dev_get_rtnl(dev);
   if (in_dev && (dev->flags & IFF_UP)) {
    struct in_ifaddr * ifa;
@@ -2221,12 +2221,12 @@ static void ip6_tnl_add_linklocal(struct inet6_dev *idev)

  /* first try to inherit the link-local address from the link device */
  if (idev->dev->iflink &&
-     (link_dev = __dev_get_by_index(idev->dev->iflink))) {
+     (link_dev = __dev_get_by_index(&init_net, idev->dev->iflink))) {
   if (!ipv6_inherit_linklocal(idev, link_dev))
    return;
  }
  /* then try to inherit it from any device */
- for_each_netdev(link_dev) {
+ for_each_netdev(&init_net, link_dev) {
   if (!ipv6_inherit_linklocal(idev, link_dev))
    return;
  }
@@ -3084,7 +3084,7 @@ inet6_rtm_newaddr(struct sk_buff *skb, struct nlmsghdr *nlh, void
*arg)
   valid_lft = INFINITY_LIFE_TIME;
  }

- dev =  __dev_get_by_index(ifm->ifa_index);
+ dev =  __dev_get_by_index(&init_net, ifm->ifa_index);
  if (dev == NULL)
   return -ENODEV;

@@ -3268,7 +3268,7 @@ static int inet6_dump_addr(struct sk_buff *skb, struct netlink_callback
*cb,
  s_ip_idx = ip_idx = cb->args[1];

  idx = 0;
- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
   if (idx < s_idx)
    goto cont;
   if (idx > s_idx)
@@ -3377,7 +3377,7 @@ static int inet6_rtm_getaddr(struct sk_buff *in_skb, struct nlmsghdr*
nlh,

  ifm = nlmsg_data(nlh);
  if (ifm->ifa_index)
-  dev = __dev_get_by_index(ifm->ifa_index);
+  dev = __dev_get_by_index(&init_net, ifm->ifa_index);
```

```
  if ((ifa = ipv6_get_ifaddr(addr, dev, 1)) == NULL) {
   err = -EADDRNOTAVAIL;
@@ -3589,7 +3589,7 @@ static int inet6_dump_ifinfo(struct sk_buff *skb, struct netlink_callback
*cb)

  read_lock(&dev_base_lock);
  idx = 0;
- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
   if (idx < s_idx)
    goto cont;
   if ((idev = in6_dev_get(dev)) == NULL)
@@ -4266,7 +4266,7 @@ void __exit addrconf_cleanup(void)
   * clean dev list.
   */

- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
   if (__in6_dev_get(dev) == NULL)
    continue;
   addrconf_ifdown(dev, 1);
diff --git a/net/ipv6/af_inet6.c b/net/ipv6/af_inet6.c
index 21931c8..e5c5aad 100644
--- a/net/ipv6/af_inet6.c
+++ b/net/ipv6/af_inet6.c
@@ -302,7 +302,7 @@ int inet6_bind(struct socket *sock, struct sockaddr *uaddr, int addr_len)
      err = -EINVAL;
      goto out;
     }
-    dev = dev_get_by_index(sk->sk_bound_dev_if);
+    dev = dev_get_by_index(&init_net, sk->sk_bound_dev_if);
     if (!dev) {
      err = -ENODEV;
      goto out;
diff --git a/net/ipv6/anycast.c b/net/ipv6/anycast.c
index 0bd6654..d407992 100644
--- a/net/ipv6/anycast.c
+++ b/net/ipv6/anycast.c
@@ -112,10 +112,10 @@ int ipv6_sock_ac_join(struct sock *sk, int ifindex, struct in6_addr *addr)
   } else {
    /* router, no matching interface: just pick one */

-   dev = dev_get_by_flags(IFF_UP, IFF_UP|IFF_LOOPBACK);
+   dev = dev_get_by_flags(&init_net, IFF_UP, IFF_UP|IFF_LOOPBACK);
   }
  } else
-  dev = dev_get_by_index(ifindex);
```

```
+  dev = dev_get_by_index(&init_net, ifindex);

  if (dev == NULL) {
   err = -ENODEV;
@@ -196,7 +196,7 @@ int ipv6_sock_ac_drop(struct sock *sk, int ifindex, struct in6_addr *addr)

  write_unlock_bh(&ipv6_sk_ac_lock);

- dev = dev_get_by_index(pac->acl_ifindex);
+ dev = dev_get_by_index(&init_net, pac->acl_ifindex);
  if (dev) {
   ipv6_dev_ac_dec(dev, &pac->acl_addr);
   dev_put(dev);
@@ -224,7 +224,7 @@ void ipv6_sock_ac_close(struct sock *sk)
   if (pac->acl_ifindex != prev_index) {
    if (dev)
     dev_put(dev);
-   dev = dev_get_by_index(pac->acl_ifindex);
+   dev = dev_get_by_index(&init_net, pac->acl_ifindex);
    prev_index = pac->acl_ifindex;
   }
   if (dev)
@@ -429,7 +429,7 @@ int ipv6_chk_acast_addr(struct net_device *dev, struct in6_addr *addr)
  if (dev)
   return ipv6_chk_acast_dev(dev, addr);
  read_lock(&dev_base_lock);
- for_each_netdev(dev)
+ for_each_netdev(&init_net, dev)
   if (ipv6_chk_acast_dev(dev, addr)) {
    found = 1;
    break;
@@ -453,7 +453,7 @@ static inline struct ifacaddr6 *ac6_get_first(struct seq_file *seq)
  struct ac6_iter_state *state = ac6_seq_private(seq);

  state->idev = NULL;
- for_each_netdev(state->dev) {
+ for_each_netdev(&init_net, state->dev) {
   struct inet6_dev *idev;
   idev = in6_dev_get(state->dev);
   if (!idev)
diff --git a/net/ipv6/datagram.c b/net/ipv6/datagram.c
index fe0f490..2ed689a 100644
--- a/net/ipv6/datagram.c
+++ b/net/ipv6/datagram.c
@@ -544,7 +544,7 @@ int datagram_send_ctl(struct msghdr *msg, struct flowi *fl,
     if (!src_info->ipi6_ifindex)
      return -EINVAL;
     else {
```

```
-     dev = dev_get_by_index(src_info->ipi6_ifindex);
+     dev = dev_get_by_index(&init_net, src_info->ipi6_ifindex);
      if (!dev)
       return -ENODEV;
     }
diff --git a/net/ipv6/ip6_tunnel.c b/net/ipv6/ip6_tunnel.c
index ca774d8..937625e 100644
--- a/net/ipv6/ip6_tunnel.c
+++ b/net/ipv6/ip6_tunnel.c
@@ -235,7 +235,7 @@ static struct ip6_tnl *ip6_tnl_create(struct ip6_tnl_parm *p)
   int i;
   for (i = 1; i < IP6_TNL_MAX; i++) {
    sprintf(name, "ip6tnl%d", i);
-   if (__dev_get_by_name(name) == NULL)
+   if (__dev_get_by_name(&init_net, name) == NULL)
     break;
   }
   if (i == IP6_TNL_MAX)
@@ -650,7 +650,7 @@ static inline int ip6_tnl_rcv_ctl(struct ip6_tnl *t)
   struct net_device *ldev = NULL;

   if (p->link)
-   ldev = dev_get_by_index(p->link);
+   ldev = dev_get_by_index(&init_net, p->link);

   if ((ipv6_addr_is_multicast(&p->laddr) ||
        likely(ipv6_chk_addr(&p->laddr, ldev, 0))) &&
@@ -786,7 +786,7 @@ static inline int ip6_tnl_xmit_ctl(struct ip6_tnl *t)
   struct net_device *ldev = NULL;

   if (p->link)
-   ldev = dev_get_by_index(p->link);
+   ldev = dev_get_by_index(&init_net, p->link);

   if (unlikely(!ipv6_chk_addr(&p->laddr, ldev, 0)))
    printk(KERN_WARNING
diff --git a/net/ipv6/ipv6_sockglue.c b/net/ipv6/ipv6_sockglue.c
index 74254fc..eb330a4 100644
--- a/net/ipv6/ipv6_sockglue.c
+++ b/net/ipv6/ipv6_sockglue.c
@@ -542,7 +542,7 @@ done:
   if (sk->sk_bound_dev_if && sk->sk_bound_dev_if != val)
    goto e_inval;

-  if (__dev_get_by_index(val) == NULL) {
+  if (__dev_get_by_index(&init_net, val) == NULL) {
    retv = -ENODEV;
    break;
```

```
   }
diff --git a/net/ipv6/mcast.c b/net/ipv6/mcast.c
index a41d5a0..e2ab43c 100644
--- a/net/ipv6/mcast.c
+++ b/net/ipv6/mcast.c
@@ -215,7 +215,7 @@ int ipv6_sock_mc_join(struct sock *sk, int ifindex, struct in6_addr *addr)
    dst_release(&rt->u.dst);
   }
  } else
-  dev = dev_get_by_index(ifindex);
+  dev = dev_get_by_index(&init_net, ifindex);

  if (dev == NULL) {
   sock_kfree_s(sk, mc_lst, sizeof(*mc_lst));
@@ -266,7 +266,7 @@ int ipv6_sock_mc_drop(struct sock *sk, int ifindex, struct in6_addr *addr)
    *lnk = mc_lst->next;
    write_unlock_bh(&ipv6_sk_mc_lock);

-   if ((dev = dev_get_by_index(mc_lst->ifindex)) != NULL) {
+   if ((dev = dev_get_by_index(&init_net, mc_lst->ifindex)) != NULL) {
     struct inet6_dev *idev = in6_dev_get(dev);

     (void) ip6_mc_leave_src(sk, mc_lst, idev);
@@ -301,7 +301,7 @@ static struct inet6_dev *ip6_mc_find_dev(struct in6_addr *group, int
ifindex)
    dst_release(&rt->u.dst);
   }
  } else
-  dev = dev_get_by_index(ifindex);
+  dev = dev_get_by_index(&init_net, ifindex);

  if (!dev)
   return NULL;
@@ -332,7 +332,7 @@ void ipv6_sock_mc_close(struct sock *sk)
   np->ipv6_mc_list = mc_lst->next;
   write_unlock_bh(&ipv6_sk_mc_lock);

-  dev = dev_get_by_index(mc_lst->ifindex);
+  dev = dev_get_by_index(&init_net, mc_lst->ifindex);
   if (dev) {
    struct inet6_dev *idev = in6_dev_get(dev);

@@ -2333,7 +2333,7 @@ static inline struct ifmcaddr6 *igmp6_mc_get_first(struct seq_file *seq)
  struct igmp6_mc_iter_state *state = igmp6_mc_seq_private(seq);

  state->idev = NULL;
- for_each_netdev(state->dev) {
+ for_each_netdev(&init_net, state->dev) {
```

```
   struct inet6_dev *idev;
   idev = in6_dev_get(state->dev);
   if (!idev)
@@ -2477,7 +2477,7 @@ static inline struct ip6_sf_list *igmp6_mcf_get_first(struct seq_file *seq)

  state->idev = NULL;
  state->im = NULL;
- for_each_netdev(state->dev) {
+ for_each_netdev(&init_net, state->dev) {
   struct inet6_dev *idev;
   idev = in6_dev_get(state->dev);
   if (unlikely(idev == NULL))
diff --git a/net/ipv6/raw.c b/net/ipv6/raw.c
index 20c1a76..71dcd48 100644
--- a/net/ipv6/raw.c
+++ b/net/ipv6/raw.c
@@ -283,7 +283,7 @@ static int rawv6_bind(struct sock *sk, struct sockaddr *uaddr, int
addr_len)
    if (!sk->sk_bound_dev_if)
     goto out;

-   dev = dev_get_by_index(sk->sk_bound_dev_if);
+   dev = dev_get_by_index(&init_net, sk->sk_bound_dev_if);
    if (!dev) {
     err = -ENODEV;
     goto out;
diff --git a/net/ipv6/reassembly.c b/net/ipv6/reassembly.c
index de795c0..31601c9 100644
--- a/net/ipv6/reassembly.c
+++ b/net/ipv6/reassembly.c
@@ -301,7 +301,7 @@ static void ip6_frag_expire(unsigned long data)

  fq_kill(fq);

- dev = dev_get_by_index(fq->iif);
+ dev = dev_get_by_index(&init_net, fq->iif);
  if (!dev)
   goto out;

diff --git a/net/ipv6/route.c b/net/ipv6/route.c
index f4f0c34..5bdd9d4 100644
--- a/net/ipv6/route.c
+++ b/net/ipv6/route.c
@@ -1130,7 +1130,7 @@ int ip6_route_add(struct fib6_config *cfg)
 #endif
  if (cfg->fc_ifindex) {
   err = -ENODEV;
-  dev = dev_get_by_index(cfg->fc_ifindex);
```

```
+  dev = dev_get_by_index(&init_net, cfg->fc_ifindex);
   if (!dev)
    goto out;
   idev = in6_dev_get(dev);
@@ -2265,7 +2265,7 @@ static int inet6_rtm_getroute(struct sk_buff *in_skb, struct nlmsghdr*
nlh, void

  if (iif) {
   struct net_device *dev;
-  dev = __dev_get_by_index(iif);
+  dev = __dev_get_by_index(&init_net, iif);
   if (!dev) {
    err = -ENODEV;
    goto errout;
diff --git a/net/ipv6/sit.c b/net/ipv6/sit.c
index eb20bb6..e79f419 100644
--- a/net/ipv6/sit.c
+++ b/net/ipv6/sit.c
@@ -167,7 +167,7 @@ static struct ip_tunnel * ipip6_tunnel_locate(struct ip_tunnel_parm
*parms, int
   int i;
   for (i=1; i<100; i++) {
    sprintf(name, "sit%d", i);
-   if (__dev_get_by_name(name) == NULL)
+   if (__dev_get_by_name(&init_net, name) == NULL)
     break;
   }
   if (i==100)
@@ -761,7 +761,7 @@ static int ipip6_tunnel_init(struct net_device *dev)
  }

  if (!tdev && tunnel->parms.link)
-  tdev = __dev_get_by_index(tunnel->parms.link);
+  tdev = __dev_get_by_index(&init_net, tunnel->parms.link);

  if (tdev) {
   dev->hard_header_len = tdev->hard_header_len + sizeof(struct iphdr);
diff --git a/net/ipx/af_ipx.c b/net/ipx/af_ipx.c
index 24921f1..29b063d 100644
--- a/net/ipx/af_ipx.c
+++ b/net/ipx/af_ipx.c
@@ -989,7 +989,7 @@ static int ipxitf_create(struct ipx_interface_definition *idef)
  if (intrfc)
   ipxitf_put(intrfc);

- dev = dev_get_by_name(idef->ipx_device);
+ dev = dev_get_by_name(&init_net, idef->ipx_device);
  rc = -ENODEV;
```

```
  if (!dev)
   goto out;
@@ -1097,7 +1097,7 @@ static int ipxitf_delete(struct ipx_interface_definition *idef)
  if (!dlink_type)
   goto out;

- dev = __dev_get_by_name(idef->ipx_device);
+ dev = __dev_get_by_name(&init_net, idef->ipx_device);
  rc = -ENODEV;
  if (!dev)
   goto out;
@@ -1192,7 +1192,7 @@ static int ipxitf_ioctl(unsigned int cmd, void __user *arg)
   if (copy_from_user(&ifr, arg, sizeof(ifr)))
    break;
   sipx = (struct sockaddr_ipx *)&ifr.ifr_addr;
-  dev  = __dev_get_by_name(ifr.ifr_name);
+  dev  = __dev_get_by_name(&init_net, ifr.ifr_name);
   rc   = -ENODEV;
   if (!dev)
    break;
diff --git a/net/irda/irnetlink.c b/net/irda/irnetlink.c
index 1e429c9..cd9ff17 100644
--- a/net/irda/irnetlink.c
+++ b/net/irda/irnetlink.c
@@ -15,6 +15,7 @@

 #include <linux/socket.h>
 #include <linux/irda.h>
+#include <net/net_namespace.h>
 #include <net/sock.h>
 #include <net/irda/irda.h>
 #include <net/irda/irlap.h>
@@ -30,7 +31,7 @@ static struct genl_family irda_nl_family = {
  .maxattr = IRDA_NL_CMD_MAX,
 };

-static struct net_device * ifname_to_netdev(struct genl_info *info)
+static struct net_device * ifname_to_netdev(struct net *net, struct genl_info *info)
 {
  char * ifname;

@@ -41,7 +42,7 @@ static struct net_device * ifname_to_netdev(struct genl_info *info)

  IRDA_DEBUG(5, "%s(): Looking for %s\n", __FUNCTION__, ifname);

- return dev_get_by_name(ifname);
+ return dev_get_by_name(net, ifname);
 }
```

```
 static int irda_nl_set_mode(struct sk_buff *skb, struct genl_info *info)
@@ -57,7 +58,7 @@ static int irda_nl_set_mode(struct sk_buff *skb, struct genl_info *info)

  IRDA_DEBUG(5, "%s(): Switching to mode: %d\n", __FUNCTION__, mode);

- dev = ifname_to_netdev(info);
+ dev = ifname_to_netdev(&init_net, info);
  if (!dev)
   return -ENODEV;

@@ -82,7 +83,7 @@ static int irda_nl_get_mode(struct sk_buff *skb, struct genl_info *info)
  void *hdr;
  int ret = -ENOBUFS;

- dev = ifname_to_netdev(info);
+ dev = ifname_to_netdev(&init_net, info);
  if (!dev)
   return -ENODEV;

diff --git a/net/llc/af_llc.c b/net/llc/af_llc.c
index b482441..49eacba 100644
--- a/net/llc/af_llc.c
+++ b/net/llc/af_llc.c
@@ -252,7 +252,7 @@ static int llc_ui_autobind(struct socket *sock, struct sockaddr_llc *addr)
  if (!sock_flag(sk, SOCK_ZAPPED))
   goto out;
  rc = -ENODEV;
- llc->dev = dev_getfirstbyhwtype(addr->sllc_arphrd);
+ llc->dev = dev_getfirstbyhwtype(&init_net, addr->sllc_arphrd);
  if (!llc->dev)
   goto out;
  rc = -EUSERS;
@@ -303,7 +303,7 @@ static int llc_ui_bind(struct socket *sock, struct sockaddr *uaddr, int
addrlen)
   goto out;
  rc = -ENODEV;
  rtnl_lock();
- llc->dev = dev_getbyhwaddr(addr->sllc_arphrd, addr->sllc_mac);
+ llc->dev = dev_getbyhwaddr(&init_net, addr->sllc_arphrd, addr->sllc_mac);
  rtnl_unlock();
  if (!llc->dev)
   goto out;
diff --git a/net/llc/llc_core.c b/net/llc/llc_core.c
index d4b13a0..248b590 100644
--- a/net/llc/llc_core.c
+++ b/net/llc/llc_core.c
@@ -19,6 +19,7 @@
```

```
 #include <linux/slab.h>
 #include <linux/string.h>
 #include <linux/init.h>
+#include <net/net_namespace.h>
 #include <net/llc.h>

 LIST_HEAD(llc_sap_list);
@@ -162,7 +163,7 @@ static int __init llc_init(void)
 {
  struct net_device *dev;

- dev = first_net_device();
+ dev = first_net_device(&init_net);
  if (dev != NULL)
   dev = next_net_device(dev);

diff --git a/net/mac80211/ieee80211.c b/net/mac80211/ieee80211.c
index 56786ff..5ea86f5 100644
--- a/net/mac80211/ieee80211.c
+++ b/net/mac80211/ieee80211.c
@@ -21,6 +21,7 @@
 #include <linux/wireless.h>
 #include <linux/rtnetlink.h>
 #include <linux/bitmap.h>
+#include <net/net_namespace.h>
 #include <net/cfg80211.h>

 #include "ieee80211_common.h"
diff --git a/net/mac80211/ieee80211_cfg.c b/net/mac80211/ieee80211_cfg.c
index 509096e..b1c13bc 100644
--- a/net/mac80211/ieee80211_cfg.c
+++ b/net/mac80211/ieee80211_cfg.c
@@ -8,6 +8,7 @@

 #include <linux/nl80211.h>
 #include <linux/rtnetlink.h>
+#include <net/net_namespace.h>
 #include <net/cfg80211.h>
 #include "ieee80211_i.h"
 #include "ieee80211_cfg.h"
@@ -50,7 +51,7 @@ static int ieee80211_del_iface(struct wiphy *wiphy, int ifindex)
  if (unlikely(local->reg_state != IEEE80211_DEV_REGISTERED))
   return -ENODEV;

- dev = dev_get_by_index(ifindex);
+ dev = dev_get_by_index(&init_net, ifindex);
  if (!dev)
   return 0;
```

```
diff --git a/net/mac80211/tx.c b/net/mac80211/tx.c
index b65ff65..9e952e3 100644
--- a/net/mac80211/tx.c
+++ b/net/mac80211/tx.c
@@ -17,6 +17,7 @@
 #include <linux/skbuff.h>
 #include <linux/etherdevice.h>
 #include <linux/bitmap.h>
+#include <net/net_namespace.h>
 #include <net/ieee80211_radiotap.h>
 #include <net/cfg80211.h>
 #include <net/mac80211.h>
@@ -1018,7 +1019,7 @@ static int inline ieee80211_tx_prepare(struct ieee80211_txrx_data *tx,
  struct net_device *dev;

  pkt_data = (struct ieee80211_tx_packet_data *)skb->cb;
- dev = dev_get_by_index(pkt_data->ifindex);
+ dev = dev_get_by_index(&init_net, pkt_data->ifindex);
  if (unlikely(dev && !is_ieee80211_device(dev, mdev))) {
   dev_put(dev);
   dev = NULL;
@@ -1226,7 +1227,7 @@ int ieee80211_master_start_xmit(struct sk_buff *skb,
  memset(&control, 0, sizeof(struct ieee80211_tx_control));

  if (pkt_data->ifindex)
-  odev = dev_get_by_index(pkt_data->ifindex);
+  odev = dev_get_by_index(&init_net, pkt_data->ifindex);
  if (unlikely(odev && !is_ieee80211_device(odev, dev))) {
   dev_put(odev);
   odev = NULL;
@@ -1722,7 +1723,7 @@ struct sk_buff *ieee80211_beacon_get(struct ieee80211_hw *hw, int
if_id,
  u8 *b_head, *b_tail;
  int bh_len, bt_len;

- bdev = dev_get_by_index(if_id);
+ bdev = dev_get_by_index(&init_net, if_id);
  if (bdev) {
   sdata = IEEE80211_DEV_TO_SUB_IF(bdev);
   ap = &sdata->u.ap;
@@ -1836,7 +1837,7 @@ ieee80211_get_buffered_bc(struct ieee80211_hw *hw, int if_id,
  struct ieee80211_sub_if_data *sdata;
  struct ieee80211_if_ap *bss = NULL;

- bdev = dev_get_by_index(if_id);
+ bdev = dev_get_by_index(&init_net, if_id);
  if (bdev) {
```

```
   sdata = IEEE80211_DEV_TO_SUB_IF(bdev);
   bss = &sdata->u.ap;
diff --git a/net/mac80211/util.c b/net/mac80211/util.c
index 07686bd..c970996 100644
--- a/net/mac80211/util.c
+++ b/net/mac80211/util.c
@@ -20,6 +20,7 @@
 #include <linux/if_arp.h>
 #include <linux/wireless.h>
 #include <linux/bitmap.h>
+#include <net/net_namespace.h>
 #include <net/cfg80211.h>

 #include "ieee80211_i.h"
@@ -318,7 +319,7 @@ __le16 ieee80211_generic_frame_duration(struct ieee80211_hw *hw, int
if_id,
      size_t frame_len, int rate)
 {
  struct ieee80211_local *local = hw_to_local(hw);
- struct net_device *bdev = dev_get_by_index(if_id);
+ struct net_device *bdev = dev_get_by_index(&init_net, if_id);
  struct ieee80211_sub_if_data *sdata;
  u16 dur;
  int erp;
@@ -342,7 +343,7 @@ __le16 ieee80211_rts_duration(struct ieee80211_hw *hw, int if_id,
 {
  struct ieee80211_local *local = hw_to_local(hw);
  struct ieee80211_rate *rate;
- struct net_device *bdev = dev_get_by_index(if_id);
+ struct net_device *bdev = dev_get_by_index(&init_net, if_id);
  struct ieee80211_sub_if_data *sdata;
  int short_preamble;
  int erp;
@@ -378,7 +379,7 @@ __le16 ieee80211_ctstoself_duration(struct ieee80211_hw *hw, int if_id,
 {
  struct ieee80211_local *local = hw_to_local(hw);
  struct ieee80211_rate *rate;
- struct net_device *bdev = dev_get_by_index(if_id);
+ struct net_device *bdev = dev_get_by_index(&init_net, if_id);
  struct ieee80211_sub_if_data *sdata;
  int short_preamble;
  int erp;
diff --git a/net/netrom/nr_route.c b/net/netrom/nr_route.c
index 24fe4a6..e943c16 100644
--- a/net/netrom/nr_route.c
+++ b/net/netrom/nr_route.c
@@ -580,7 +580,7 @@ static struct net_device *nr_ax25_dev_get(char *devname)
 {
```

```
  struct net_device *dev;

- if ((dev = dev_get_by_name(devname)) == NULL)
+ if ((dev = dev_get_by_name(&init_net, devname)) == NULL)
    return NULL;

  if ((dev->flags & IFF_UP) && dev->type == ARPHRD_AX25)
@@ -598,7 +598,7 @@ struct net_device *nr_dev_first(void)
  struct net_device *dev, *first = NULL;

  read_lock(&dev_base_lock);
- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
    if ((dev->flags & IFF_UP) && dev->type == ARPHRD_NETROM)
     if (first == NULL || strncmp(dev->name, first->name, 3) < 0)
      first = dev;
@@ -618,7 +618,7 @@ struct net_device *nr_dev_get(ax25_address *addr)
  struct net_device *dev;

  read_lock(&dev_base_lock);
- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
    if ((dev->flags & IFF_UP) && dev->type == ARPHRD_NETROM && ax25cmp(addr,
(ax25_address *)dev->dev_addr) == 0) {
     dev_hold(dev);
     goto out;
diff --git a/net/packet/af_packet.c b/net/packet/af_packet.c
index 30f2143..f14d66b 100644
--- a/net/packet/af_packet.c
+++ b/net/packet/af_packet.c
@@ -347,7 +347,7 @@ static int packet_sendmsg_spkt(struct kiocb *iocb, struct socket *sock,
    */

  saddr->spkt_device[13] = 0;
- dev = dev_get_by_name(saddr->spkt_device);
+ dev = dev_get_by_name(&init_net, saddr->spkt_device);
  err = -ENODEV;
  if (dev == NULL)
    goto out_unlock;
@@ -743,7 +743,7 @@ static int packet_sendmsg(struct kiocb *iocb, struct socket *sock,
  }


- dev = dev_get_by_index(ifindex);
+ dev = dev_get_by_index(&init_net, ifindex);
  err = -ENXIO;
  if (dev == NULL)
    goto out_unlock;
```

```
@@ -938,7 +938,7 @@ static int packet_bind_spkt(struct socket *sock, struct sockaddr *uaddr,
int add
   return -EINVAL;
  strlcpy(name,uaddr->sa_data,sizeof(name));

- dev = dev_get_by_name(name);
+ dev = dev_get_by_name(&init_net, name);
  if (dev) {
   err = packet_do_bind(sk, dev, pkt_sk(sk)->num);
   dev_put(dev);
@@ -965,7 +965,7 @@ static int packet_bind(struct socket *sock, struct sockaddr *uaddr, int
addr_len

  if (sll->sll_ifindex) {
   err = -ENODEV;
-  dev = dev_get_by_index(sll->sll_ifindex);
+  dev = dev_get_by_index(&init_net, sll->sll_ifindex);
   if (dev == NULL)
    goto out;
  }
@@ -1162,7 +1162,7 @@ static int packet_getname_spkt(struct socket *sock, struct sockaddr
*uaddr,
   return -EOPNOTSUPP;

  uaddr->sa_family = AF_PACKET;
- dev = dev_get_by_index(pkt_sk(sk)->ifindex);
+ dev = dev_get_by_index(&init_net, pkt_sk(sk)->ifindex);
  if (dev) {
   strlcpy(uaddr->sa_data, dev->name, 15);
   dev_put(dev);
@@ -1187,7 +1187,7 @@ static int packet_getname(struct socket *sock, struct sockaddr *uaddr,
  sll->sll_family = AF_PACKET;
  sll->sll_ifindex = po->ifindex;
  sll->sll_protocol = po->num;
- dev = dev_get_by_index(po->ifindex);
+ dev = dev_get_by_index(&init_net, po->ifindex);
  if (dev) {
   sll->sll_hatype = dev->type;
   sll->sll_halen = dev->addr_len;
@@ -1239,7 +1239,7 @@ static int packet_mc_add(struct sock *sk, struct packet_mreq_max
*mreq)
  rtnl_lock();

  err = -ENODEV;
- dev = __dev_get_by_index(mreq->mr_ifindex);
+ dev = __dev_get_by_index(&init_net, mreq->mr_ifindex);
  if (!dev)
   goto done;
```

```
@@ -1293,7 +1293,7 @@ static int packet_mc_drop(struct sock *sk, struct packet_mreq_max
*mreq)
    if (--ml->count == 0) {
     struct net_device *dev;
     *mlp = ml->next;
-    dev = dev_get_by_index(ml->ifindex);
+    dev = dev_get_by_index(&init_net, ml->ifindex);
     if (dev) {
      packet_dev_mc(dev, ml, -1);
      dev_put(dev);
@@ -1321,7 +1321,7 @@ static void packet_flush_mclist(struct sock *sk)
   struct net_device *dev;

   po->mclist = ml->next;
-  if ((dev = dev_get_by_index(ml->ifindex)) != NULL) {
+  if ((dev = dev_get_by_index(&init_net, ml->ifindex)) != NULL) {
    packet_dev_mc(dev, ml, -1);
    dev_put(dev);
   }
diff --git a/net/rose/rose_route.c b/net/rose/rose_route.c
index bbcbad1..a6e29fc 100644
--- a/net/rose/rose_route.c
+++ b/net/rose/rose_route.c
@@ -578,7 +578,7 @@ static struct net_device *rose_ax25_dev_get(char *devname)
 {
  struct net_device *dev;

- if ((dev = dev_get_by_name(devname)) == NULL)
+ if ((dev = dev_get_by_name(&init_net, devname)) == NULL)
   return NULL;

  if ((dev->flags & IFF_UP) && dev->type == ARPHRD_AX25)
@@ -596,7 +596,7 @@ struct net_device *rose_dev_first(void)
  struct net_device *dev, *first = NULL;

  read_lock(&dev_base_lock);
- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
   if ((dev->flags & IFF_UP) && dev->type == ARPHRD_ROSE)
    if (first == NULL || strncmp(dev->name, first->name, 3) < 0)
     first = dev;
@@ -614,7 +614,7 @@ struct net_device *rose_dev_get(rose_address *addr)
  struct net_device *dev;

  read_lock(&dev_base_lock);
- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
```

```
   if ((dev->flags & IFF_UP) && dev->type == ARPHRD_ROSE && rosecmp(addr, (rose_address
*)dev->dev_addr) == 0) {
    dev_hold(dev);
    goto out;
@@ -631,7 +631,7 @@ static int rose_dev_exists(rose_address *addr)
  struct net_device *dev;

  read_lock(&dev_base_lock);
- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
   if ((dev->flags & IFF_UP) && dev->type == ARPHRD_ROSE && rosecmp(addr, (rose_address
*)dev->dev_addr) == 0)
    goto out;
  }
diff --git a/net/sched/act_mirred.c b/net/sched/act_mirred.c
index 5795789..fd7bca4 100644
--- a/net/sched/act_mirred.c
+++ b/net/sched/act_mirred.c
@@ -20,6 +20,7 @@
 #include <linux/rtnetlink.h>
 #include <linux/module.h>
 #include <linux/init.h>
+#include <net/net_namespace.h>
 #include <net/netlink.h>
 #include <net/pkt_sched.h>
 #include <linux/tc_act/tc_mirred.h>
@@ -73,7 +74,7 @@ static int tcf_mirred_init(struct rtattr *rta, struct rtattr *est,
  parm = RTA_DATA(tb[TCA_MIRRED_PARMS-1]);

  if (parm->ifindex) {
-  dev = __dev_get_by_index(parm->ifindex);
+  dev = __dev_get_by_index(&init_net, parm->ifindex);
   if (dev == NULL)
    return -ENODEV;
   switch (dev->type) {
diff --git a/net/sched/cls_api.c b/net/sched/cls_api.c
index 5f0fbca..0365797 100644
--- a/net/sched/cls_api.c
+++ b/net/sched/cls_api.c
@@ -154,7 +154,7 @@ replay:
  /* Find head of filter chain. */

  /* Find link */
- if ((dev = __dev_get_by_index(t->tcm_ifindex)) == NULL)
+ if ((dev = __dev_get_by_index(&init_net, t->tcm_ifindex)) == NULL)
   return -ENODEV;

  /* Find qdisc */
```

```
@@ -387,7 +387,7 @@ static int tc_dump_tfilter(struct sk_buff *skb, struct netlink_callback *cb)

  if (cb->nlh->nlmsg_len < NLMSG_LENGTH(sizeof(*tcm)))
   return skb->len;
- if ((dev = dev_get_by_index(tcm->tcm_ifindex)) == NULL)
+ if ((dev = dev_get_by_index(&init_net, tcm->tcm_ifindex)) == NULL)
   return skb->len;

  if (!tcm->tcm_parent)
diff --git a/net/sched/em_meta.c b/net/sched/em_meta.c
index 650f09c..e998961 100644
--- a/net/sched/em_meta.c
+++ b/net/sched/em_meta.c
@@ -291,7 +291,7 @@ META_COLLECTOR(var_sk_bound_if)
   } else  {
   struct net_device *dev;

-  dev = dev_get_by_index(skb->sk->sk_bound_dev_if);
+  dev = dev_get_by_index(&init_net, skb->sk->sk_bound_dev_if);
   *err = var_dev(dev, dst);
   if (dev)
    dev_put(dev);
diff --git a/net/sched/sch_api.c b/net/sched/sch_api.c
index efc383c..39d3278 100644
--- a/net/sched/sch_api.c
+++ b/net/sched/sch_api.c
@@ -607,7 +607,7 @@ static int tc_get_qdisc(struct sk_buff *skb, struct nlmsghdr *n, void *arg)
  struct Qdisc *p = NULL;
  int err;

- if ((dev = __dev_get_by_index(tcm->tcm_ifindex)) == NULL)
+ if ((dev = __dev_get_by_index(&init_net, tcm->tcm_ifindex)) == NULL)
   return -ENODEV;

  if (clid) {
@@ -674,7 +674,7 @@ replay:
  clid = tcm->tcm_parent;
  q = p = NULL;

- if ((dev = __dev_get_by_index(tcm->tcm_ifindex)) == NULL)
+ if ((dev = __dev_get_by_index(&init_net, tcm->tcm_ifindex)) == NULL)
   return -ENODEV;

  if (clid) {
@@ -881,7 +881,7 @@ static int tc_dump_qdisc(struct sk_buff *skb, struct netlink_callback *cb)
  s_q_idx = q_idx = cb->args[1];
  read_lock(&dev_base_lock);
  idx = 0;
```

```
- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
   if (idx < s_idx)
    goto cont;
   if (idx > s_idx)
@@ -932,7 +932,7 @@ static int tc_ctl_tclass(struct sk_buff *skb, struct nlmsghdr *n, void *arg)
  u32 qid = TC_H_MAJ(clid);
  int err;

- if ((dev = __dev_get_by_index(tcm->tcm_ifindex)) == NULL)
+ if ((dev = __dev_get_by_index(&init_net, tcm->tcm_ifindex)) == NULL)
   return -ENODEV;

  /*
@@ -1115,7 +1115,7 @@ static int tc_dump_tclass(struct sk_buff *skb, struct netlink_callback
*cb)

  if (cb->nlh->nlmsg_len < NLMSG_LENGTH(sizeof(*tcm)))
   return 0;
- if ((dev = dev_get_by_index(tcm->tcm_ifindex)) == NULL)
+ if ((dev = dev_get_by_index(&init_net, tcm->tcm_ifindex)) == NULL)
   return 0;

  s_t = cb->args[0];
diff --git a/net/sctp/ipv6.c b/net/sctp/ipv6.c
index 5953260..00a093a 100644
--- a/net/sctp/ipv6.c
+++ b/net/sctp/ipv6.c
@@ -843,7 +843,7 @@ static int sctp_inet6_bind_verify(struct sctp_sock *opt, union sctp_addr
*addr)
   if (type & IPV6_ADDR_LINKLOCAL) {
    if (!addr->v6.sin6_scope_id)
     return 0;
-   dev = dev_get_by_index(addr->v6.sin6_scope_id);
+   dev = dev_get_by_index(&init_net, addr->v6.sin6_scope_id);
    if (!dev)
     return 0;
    if (!ipv6_chk_addr(&addr->v6.sin6_addr, dev, 0)) {
@@ -874,7 +874,7 @@ static int sctp_inet6_send_verify(struct sctp_sock *opt, union sctp_addr
*addr)
   if (type & IPV6_ADDR_LINKLOCAL) {
    if (!addr->v6.sin6_scope_id)
     return 0;
-   dev = dev_get_by_index(addr->v6.sin6_scope_id);
+   dev = dev_get_by_index(&init_net, addr->v6.sin6_scope_id);
    if (!dev)
     return 0;
    dev_put(dev);
```

```
diff --git a/net/sctp/protocol.c b/net/sctp/protocol.c
index 0052ff4..193835d 100644
--- a/net/sctp/protocol.c
+++ b/net/sctp/protocol.c
@@ -176,7 +176,7 @@ static void sctp_get_local_addr_list(void)
  struct sctp_af *af;

  read_lock(&dev_base_lock);
- for_each_netdev(dev) {
+ for_each_netdev(&init_net, dev) {
   __list_for_each(pos, &sctp_address_families) {
    af = list_entry(pos, struct sctp_af, list);
    af->copy_addrlist(&sctp_local_addr_list, dev);
diff --git a/net/socket.c b/net/socket.c
index bf5aba7..f123343 100644
--- a/net/socket.c
+++ b/net/socket.c
@@ -794,9 +794,9 @@ static ssize_t sock_aio_write(struct kiocb *iocb, const struct iovec *iov,
  */

 static DEFINE_MUTEX(br_ioctl_mutex);
-static int (*br_ioctl_hook) (unsigned int cmd, void __user *arg) = NULL;
+static int (*br_ioctl_hook) (struct net *, unsigned int cmd, void __user *arg) = NULL;

-void brioctl_set(int (*hook) (unsigned int, void __user *))
+void brioctl_set(int (*hook) (struct net *, unsigned int, void __user *))
 {
  mutex_lock(&br_ioctl_mutex);
  br_ioctl_hook = hook;
@@ -806,9 +806,9 @@ void brioctl_set(int (*hook) (unsigned int, void __user *))
 EXPORT_SYMBOL(brioctl_set);

 static DEFINE_MUTEX(vlan_ioctl_mutex);
-static int (*vlan_ioctl_hook) (void __user *arg);
+static int (*vlan_ioctl_hook) (struct net *, void __user *arg);

-void vlan_ioctl_set(int (*hook) (void __user *))
+void vlan_ioctl_set(int (*hook) (struct net *, void __user *))
 {
  mutex_lock(&vlan_ioctl_mutex);
  vlan_ioctl_hook = hook;
@@ -837,16 +837,20 @@ EXPORT_SYMBOL(dlci_ioctl_set);
 static long sock_ioctl(struct file *file, unsigned cmd, unsigned long arg)
 {
  struct socket *sock;
+ struct sock *sk;
  void __user *argp = (void __user *)arg;
  int pid, err;
```

```
+ struct net *net;

  sock = file->private_data;
+ sk = sock->sk;
+ net = sk->sk_net;
  if (cmd >= SIOCDEVPRIVATE && cmd <= (SIOCDEVPRIVATE + 15)) {
-  err = dev_ioctl(cmd, argp);
+  err = dev_ioctl(net, cmd, argp);
  } else
 #ifdef CONFIG_WIRELESS_EXT
  if (cmd >= SIOCIWFIRST && cmd <= SIOCIWLAST) {
-  err = dev_ioctl(cmd, argp);
+  err = dev_ioctl(net, cmd, argp);
  } else
 #endif    /* CONFIG_WIRELESS_EXT */
   switch (cmd) {
@@ -872,7 +876,7 @@ static long sock_ioctl(struct file *file, unsigned cmd, unsigned long arg)

    mutex_lock(&br_ioctl_mutex);
    if (br_ioctl_hook)
-    err = br_ioctl_hook(cmd, argp);
+    err = br_ioctl_hook(net, cmd, argp);
    mutex_unlock(&br_ioctl_mutex);
    break;
   case SIOCGIFVLAN:
@@ -883,7 +887,7 @@ static long sock_ioctl(struct file *file, unsigned cmd, unsigned long arg)

    mutex_lock(&vlan_ioctl_mutex);
    if (vlan_ioctl_hook)
-    err = vlan_ioctl_hook(argp);
+    err = vlan_ioctl_hook(net, argp);
    mutex_unlock(&vlan_ioctl_mutex);
    break;
   case SIOCADDDLCI:
@@ -906,7 +910,7 @@ static long sock_ioctl(struct file *file, unsigned cmd, unsigned long arg)
     * to the NIC driver.
     */
    if (err == -ENOIOCTLCMD)
-    err = dev_ioctl(cmd, argp);
+    err = dev_ioctl(net, cmd, argp);
    break;
   }
  return err;
diff --git a/net/tipc/eth_media.c b/net/tipc/eth_media.c
index 406f0d2..d6fc057 100644
--- a/net/tipc/eth_media.c
+++ b/net/tipc/eth_media.c
@@ -135,7 +135,7 @@ static int enable_bearer(struct tipc_bearer *tb_ptr)
```

```
  /* Find device with specified name */

- for_each_netdev(pdev){
+ for_each_netdev(&init_net, pdev){
   if (!strncmp(pdev->name, driver_name, IFNAMSIZ)) {
    dev = pdev;
    break;
diff --git a/net/wireless/wext.c b/net/wireless/wext.c
index b8069af..e8b3409 100644
--- a/net/wireless/wext.c
+++ b/net/wireless/wext.c
@@ -673,7 +673,22 @@ static const struct seq_operations wireless_seq_ops = {

 static int wireless_seq_open(struct inode *inode, struct file *file)
 {
- return seq_open(file, &wireless_seq_ops);
+ struct seq_file *seq;
+ int res;
+ res = seq_open(file, &wireless_seq_ops);
+ if (!res) {
+  seq = file->private_data;
+  seq->private = get_net(PROC_NET(inode));
+ }
+ return res;
+}
+
+static int wireless_seq_release(struct inode *inode, struct file *file)
+{
+ struct seq_file *seq = file->private_data;
+ struct net *net = seq->private;
+ put_net(net);
+ return seq_release(inode, file);
 }

 static const struct file_operations wireless_seq_fops = {
@@ -681,17 +696,22 @@ static const struct file_operations wireless_seq_fops = {
  .open    = wireless_seq_open,
  .read    = seq_read,
  .llseek  = seq_lseek,
- .release = seq_release,
+ .release = wireless_seq_release,
 };

-int __init wext_proc_init(void)
+int wext_proc_init(struct net *net)
 {
  /* Create /proc/net/wireless entry */
```

```
- if (!proc_net_fops_create(&init_net, "wireless", S_IRUGO, &wireless_seq_fops))
+ if (!proc_net_fops_create(net, "wireless", S_IRUGO, &wireless_seq_fops))
   return -ENOMEM;

  return 0;
 }
+
+void wext_proc_exit(struct net *net)
+{
+ proc_net_remove(net, "wireless");
+}
 #endif /* CONFIG_PROC_FS */

 /************************ IOCTL SUPPORT ************************/
@@ -1011,7 +1031,7 @@ static int ioctl_private_call(struct net_device *dev, struct ifreq *ifr,
  * Main IOCTl dispatcher.
  * Check the type of IOCTL and call the appropriate wrapper...
  */
-static int wireless_process_ioctl(struct ifreq *ifr, unsigned int cmd)
+static int wireless_process_ioctl(struct net *net, struct ifreq *ifr, unsigned int cmd)
 {
  struct net_device *dev;
  iw_handler handler;
@@ -1020,7 +1040,7 @@ static int wireless_process_ioctl(struct ifreq *ifr, unsigned int cmd)
   * The copy_to/from_user() of ifr is also dealt with in there */

  /* Make sure the device exist */
- if ((dev = __dev_get_by_name(ifr->ifr_name)) == NULL)
+ if ((dev = __dev_get_by_name(net, ifr->ifr_name)) == NULL)
   return -ENODEV;

  /* A bunch of special cases, then the generic case...
@@ -1054,7 +1074,7 @@ static int wireless_process_ioctl(struct ifreq *ifr, unsigned int cmd)
 }

 /* entry point from dev ioctl */
-int wext_handle_ioctl(struct ifreq *ifr, unsigned int cmd,
+int wext_handle_ioctl(struct net *net, struct ifreq *ifr, unsigned int cmd,
         void __user *arg)
 {
  int ret;
@@ -1066,9 +1086,9 @@ int wext_handle_ioctl(struct ifreq *ifr, unsigned int cmd,
      && !capable(CAP_NET_ADMIN))
   return -EPERM;

- dev_load(ifr->ifr_name);
+ dev_load(net, ifr->ifr_name);
  rtnl_lock();
```

```
- ret = wireless_process_ioctl(ifr, cmd);
+ ret = wireless_process_ioctl(net, ifr, cmd);
  rtnl_unlock();
  if (IW_IS_GET(cmd) && copy_to_user(arg, ifr, sizeof(struct ifreq)))
   return -EFAULT;
diff --git a/net/x25/x25_route.c b/net/x25/x25_route.c
index 060fcfa..86b5b4d 100644
--- a/net/x25/x25_route.c
+++ b/net/x25/x25_route.c
@@ -129,7 +129,7 @@ void x25_route_device_down(struct net_device *dev)
  */
 struct net_device *x25_dev_get(char *devname)
 {
- struct net_device *dev = dev_get_by_name(devname);
+ struct net_device *dev = dev_get_by_name(&init_net, devname);

  if (dev &&
      (!(dev->flags & IFF_UP) || (dev->type != ARPHRD_X25
--
1.5.3.rc6.17.g1911
```


_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: [PATCH 14/16] net: Factor out __dev_alloc_name from dev_alloc_name
Posted by ebiederm on Sat, 08 Sep 2007 21:36:56 GMT
View Forum Message <> Reply to Message

When forcibly changing the network namespace of a device
I need something that can generate a name for the device
in the new namespace without overwriting the old name.

__dev_alloc_name provides me that functionality.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 net/core/dev.c | 48 ++++++++++++++++++++++++++++++++++++++----------
 1 files changed, 35 insertions(+), 13 deletions(-)

```
diff --git a/net/core/dev.c b/net/core/dev.c
index c51cf40..53cdb64 100644
--- a/net/core/dev.c
+++ b/net/core/dev.c
@@ -739,9 +739,10 @@ int dev_valid_name(const char *name)
 }
```

```
 /**
- * dev_alloc_name - allocate a name for a device
- * @dev: device
+ * __dev_alloc_name - allocate a name for a device
+ * @net: network namespace to allocate the device name in
  * @name: name format string
+ * @buf:  scratch buffer and result name string
  *
  * Passed a format string - eg "lt%d" it will try and find a suitable
  * id. It scans list of devices to build up a free map, then chooses
@@ -752,18 +753,13 @@ int dev_valid_name(const char *name)
  * Returns the number of the unit assigned or a negative errno code.
  */

-int dev_alloc_name(struct net_device *dev, const char *name)
+static int __dev_alloc_name(struct net *net, const char *name, char *buf)
 {
  int i = 0;
- char buf[IFNAMSIZ];
  const char *p;
  const int max_netdevices = 8*PAGE_SIZE;
  long *inuse;
  struct net_device *d;
- struct net *net;
-
- BUG_ON(!dev->nd_net);
- net = dev->nd_net;

  p = strnchr(name, IFNAMSIZ-1, '%');
  if (p) {
@@ -787,7 +783,7 @@ int dev_alloc_name(struct net_device *dev, const char *name)
     continue;

    /*  avoid cases where sscanf is not exact inverse of printf */
-   snprintf(buf, sizeof(buf), name, i);
+   snprintf(buf, IFNAMSIZ, name, i);
    if (!strncmp(buf, d->name, IFNAMSIZ))
     set_bit(i, inuse);
   }
@@ -796,11 +792,9 @@ int dev_alloc_name(struct net_device *dev, const char *name)
   free_page((unsigned long) inuse);
  }

- snprintf(buf, sizeof(buf), name, i);
- if (!__dev_get_by_name(net, buf)) {
-  strlcpy(dev->name, buf, IFNAMSIZ);
+ snprintf(buf, IFNAMSIZ, name, i);
```

```
+ if (!__dev_get_by_name(net, buf))
   return i;
- }

  /* It is possible to run out of possible slots
   * when the name is long and there isn't enough space left
@@ -809,6 +803,34 @@ int dev_alloc_name(struct net_device *dev, const char *name)
  return -ENFILE;
 }

+/**
+ * dev_alloc_name - allocate a name for a device
+ * @dev: device
+ * @name: name format string
+ *
+ * Passed a format string - eg "lt%d" it will try and find a suitable
+ * id. It scans list of devices to build up a free map, then chooses
+ * the first empty slot. The caller must hold the dev_base or rtnl lock
+ * while allocating the name and adding the device in order to avoid
+ * duplicates.
+ * Limited to bits_per_byte * page size devices (ie 32K on most platforms).
+ * Returns the number of the unit assigned or a negative errno code.
+ */
+
+int dev_alloc_name(struct net_device *dev, const char *name)
+{
+ char buf[IFNAMSIZ];
+ struct net *net;
+ int ret;
+
+ BUG_ON(!dev->nd_net);
+ net = dev->nd_net;
+ ret = __dev_alloc_name(net, name, buf);
+ if (ret >= 0)
+  strlcpy(dev->name, buf, IFNAMSIZ);
+ return ret;
+}
+

 /**
  * dev_change_name - change name of a device
--
1.5.3.rc6.17.g1911
```

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: [PATCH 15/16] net: Implement network device movement between namespaces
Posted by ebiederm on Sat, 08 Sep 2007 21:38:46 GMT
View Forum Message <> Reply to Message

This patch introduces NETIF_F_NETNS_LOCAL a flag to indicate
a network device is local to a single network namespace and
should never be moved. Useful for pseudo devices that we
need an instance in each network namespace (like the loopback
device) and for any device we find that cannot handle multiple
network namespaces so we may trap them in the initial network
namespace.

This patch introduces the function dev_change_net_namespace
a function used to move a network device from one network
namespace to another. To the network device nothing
special appears to happen, to the components of the network
stack it appears as if the network device was unregistered
in the network namespace it is in, and a new device
was registered in the network namespace the device
was moved to.

This patch sets up a namespace device destructor that
upon the exit of a network namespace moves all of the
movable network devices to the initial network namespace
so they are not lost.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
drivers/net/loopback.c | 3 +-
include/linux/netdevice.h | 3 +
net/core/dev.c | 189 ++---
3 files changed, 184 insertions(+), 11 deletions(-)

```
diff --git a/drivers/net/loopback.c b/drivers/net/loopback.c
index 5106c23..e399f7b 100644
--- a/drivers/net/loopback.c
+++ b/drivers/net/loopback.c
@@ -222,7 +222,8 @@ struct net_device loopback_dev = {
       | NETIF_F_TSO
 #endif
       | NETIF_F_NO_CSUM | NETIF_F_HIGHDMA
-      | NETIF_F_LLTX,
+      | NETIF_F_LLTX
+      | NETIF_F_NETNS_LOCAL,
   .ethtool_ops  = &loopback_ethtool_ops,
 };

diff --git a/include/linux/netdevice.h b/include/linux/netdevice.h
```

```
index ec90d1a..d33d897 100644
--- a/include/linux/netdevice.h
+++ b/include/linux/netdevice.h
@@ -435,6 +435,7 @@ struct net_device
 #define NETIF_F_VLAN_CHALLENGED 1024 /* Device cannot handle VLAN packets */
 #define NETIF_F_GSO  2048 /* Enable software GSO. */
 #define NETIF_F_LLTX  4096 /* LockLess TX */
+#define NETIF_F_NETNS_LOCAL 8192 /* Does not change network namespaces */
 #define NETIF_F_MULTI_QUEUE 16384 /* Has multiple TX/RX queues */
 #define NETIF_F_LRO  32768 /* large receive offload */

@@ -1002,6 +1003,8 @@ extern int  dev_ethtool(struct net *net, struct ifreq *);
 extern unsigned  dev_get_flags(const struct net_device *);
 extern int  dev_change_flags(struct net_device *, unsigned);
 extern int  dev_change_name(struct net_device *, char *);
+extern int  dev_change_net_namespace(struct net_device *,
+       struct net *, const char *);
 extern int  dev_set_mtu(struct net_device *, int);
 extern int  dev_set_mac_address(struct net_device *,
         struct sockaddr *);
diff --git a/net/core/dev.c b/net/core/dev.c
index 53cdb64..d82ec5a 100644
--- a/net/core/dev.c
+++ b/net/core/dev.c
@@ -208,6 +208,34 @@ static inline struct hlist_head *dev_index_hash(struct net *net, int
ifindex)
  return &net->dev_index_head[ifindex & ((1 << NETDEV_HASHBITS) - 1)];
 }

+/* Device list insertion */
+static int list_netdevice(struct net_device *dev)
+{
+ struct net *net = dev->nd_net;
+
+ ASSERT_RTNL();
+
+ write_lock_bh(&dev_base_lock);
+ list_add_tail(&dev->dev_list, &net->dev_base_head);
+ hlist_add_head(&dev->name_hlist, dev_name_hash(net, dev->name));
+ hlist_add_head(&dev->index_hlist, dev_index_hash(net, dev->ifindex));
+ write_unlock_bh(&dev_base_lock);
+ return 0;
+}
+
+/* Device list removal */
+static void unlist_netdevice(struct net_device *dev)
+{
+ ASSERT_RTNL();
```

```
+
+ /* Unlink dev from the device chain */
+ write_lock_bh(&dev_base_lock);
+ list_del(&dev->dev_list);
+ hlist_del(&dev->name_hlist);
+ hlist_del(&dev->index_hlist);
+ write_unlock_bh(&dev_base_lock);
+}
+
 /*
  * Our notifier list
  */
@@ -3553,12 +3581,8 @@ int register_netdevice(struct net_device *dev)
  set_bit(__LINK_STATE_PRESENT, &dev->state);

  dev_init_scheduler(dev);
- write_lock_bh(&dev_base_lock);
- list_add_tail(&dev->dev_list, &net->dev_base_head);
- hlist_add_head(&dev->name_hlist, head);
- hlist_add_head(&dev->index_hlist, dev_index_hash(net, dev->ifindex));
  dev_hold(dev);
- write_unlock_bh(&dev_base_lock);
+ list_netdevice(dev);

  /* Notify protocols, that a new device appeared. */
  ret = raw_notifier_call_chain(&netdev_chain, NETDEV_REGISTER, dev);
@@ -3865,11 +3889,7 @@ void unregister_netdevice(struct net_device *dev)
   dev_close(dev);

  /* And unlink it from device chain. */
- write_lock_bh(&dev_base_lock);
- list_del(&dev->dev_list);
- hlist_del(&dev->name_hlist);
- hlist_del(&dev->index_hlist);
- write_unlock_bh(&dev_base_lock);
+ unlist_netdevice(dev);

  dev->reg_state = NETREG_UNREGISTERING;

@@ -3927,6 +3947,122 @@ void unregister_netdev(struct net_device *dev)

 EXPORT_SYMBOL(unregister_netdev);

+/**
+ * dev_change_net_namespace - move device to different nethost namespace
+ * @dev: device
+ * @net: network namespace
+ * @pat: If not NULL name pattern to try if the current device name
```

```
+ *       is already taken in the destination network namespace.
+ *
+ * This function shuts down a device interface and moves it
+ * to a new network namespace. On success 0 is returned, on
+ * a failure a netagive errno code is returned.
+ *
+ * Callers must hold the rtnl semaphore.
+ */
+
+int dev_change_net_namespace(struct net_device *dev, struct net *net, const char *pat)
+{
+ char buf[IFNAMSIZ];
+ const char *destname;
+ int err;
+
+ ASSERT_RTNL();
+
+ /* Don't allow namespace local devices to be moved. */
+ err = -EINVAL;
+ if (dev->features & NETIF_F_NETNS_LOCAL)
+  goto out;
+
+ /* Ensure the device has been registrered */
+ err = -EINVAL;
+ if (dev->reg_state != NETREG_REGISTERED)
+  goto out;
+
+ /* Get out if there is nothing todo */
+ err = 0;
+ if (dev->nd_net == net)
+  goto out;
+
+ /* Pick the destination device name, and ensure
+  * we can use it in the destination network namespace.
+  */
+ err = -EEXIST;
+ destname = dev->name;
+ if (__dev_get_by_name(net, destname)) {
+  /* We get here if we can't use the current device name */
+  if (!pat)
+   goto out;
+  if (!dev_valid_name(pat))
+   goto out;
+  if (strchr(pat, '%')) {
+   if (__dev_alloc_name(net, pat, buf) < 0)
+    goto out;
+   destname = buf;
+  } else
```

```
+   destname = pat;
+  if (__dev_get_by_name(net, destname))
+   goto out;
+ }
+
+ /*
+  * And now a mini version of register_netdevice unregister_netdevice.
+  */
+
+ /* If device is running close it first. */
+ if (dev->flags & IFF_UP)
+  dev_close(dev);
+
+ /* And unlink it from device chain */
+ err = -ENODEV;
+ unlist_netdevice(dev);
+
+ synchronize_net();
+
+ /* Shutdown queueing discipline. */
+ dev_shutdown(dev);
+
+ /* Notify protocols, that we are about to destroy
+    this device. They should clean all the things.
+ */
+ call_netdevice_notifiers(NETDEV_UNREGISTER, dev);
+
+ /*
+  * Flush the unicast and multicast chains
+  */
+ dev_addr_discard(dev);
+
+ /* Actually switch the network namespace */
+ dev->nd_net = net;
+
+ /* Assign the new device name */
+ if (destname != dev->name)
+  strcpy(dev->name, destname);
+
+ /* If there is an ifindex conflict assign a new one */
+ if (__dev_get_by_index(net, dev->ifindex)) {
+  int iflink = (dev->iflink == dev->ifindex);
+  dev->ifindex = dev_new_index(net);
+  if (iflink)
+   dev->iflink = dev->ifindex;
+ }
+
+ /* Fixup sysfs */
```

```
+ err = device_rename(&dev->dev, dev->name);
+ BUG_ON(err);
+
+ /* Add the device back in the hashes */
+ list_netdevice(dev);
+
+ /* Notify protocols, that a new device appeared. */
+ call_netdevice_notifiers(NETDEV_REGISTER, dev);
+
+ synchronize_net();
+ err = 0;
+out:
+ return err;
+}
+
 static int dev_cpu_callback(struct notifier_block *nfb,
        unsigned long action,
        void *ocpu)
@@ -4159,6 +4295,36 @@ static struct pernet_operations netdev_net_ops = {
  .exit = netdev_exit,
 };

+static void default_device_exit(struct net *net)
+{
+ struct net_device *dev, *next;
+ /*
+  * Push all migratable of the network devices back to the
+  * initial network namespace
+  */
+ rtnl_lock();
+ for_each_netdev_safe(net, dev, next) {
+  int err;
+
+  /* Ignore unmoveable devices (i.e. loopback) */
+  if (dev->features & NETIF_F_NETNS_LOCAL)
+   continue;
+
+  /* Push remaing network devices to init_net */
+  err = dev_change_net_namespace(dev, &init_net, "dev%d");
+  if (err) {
+   printk(KERN_WARNING "%s: failed to move %s to init_net: %d\n",
+    __func__, dev->name, err);
+   unregister_netdevice(dev);
+  }
+ }
+ rtnl_unlock();
+}
+
```

```
+static struct pernet_operations default_device_ops = {
+ .exit = default_device_exit,
+};
+
 /*
  * Initialize the DEV module. At boot time this walks the device list and
  * unhooks any devices that fail to initialise (normally hardware not
@@ -4189,6 +4355,9 @@ static int __init net_dev_init(void)
  if (register_pernet_subsys(&netdev_net_ops))
   goto out;

+ if (register_pernet_device(&default_device_ops))
+  goto out;
+
  /*
   * Initialise the packet receive queues.
   */
--
1.5.3.rc6.17.g1911
```

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: [PATCH 16/16] net: netlink support for moving devices between network namespaces.

Posted by ebiederm on Sat, 08 Sep 2007 21:43:44 GMT

View Forum Message <> Reply to Message

The simplest thing to implement is moving network devices between
namespaces. However with the same attribute IFLA_NET_NS_PID we can
easily implement creating devices in the destination network
namespace as well. However that is a little bit trickier so this
patch sticks to what is simple and easy.

A pid is used to identify a process that happens to be a member
of the network namespace we want to move the network device to.

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

```
---
 include/linux/if_link.h |    1 +
 net/core/rtnetlink.c    |   35 +++++++++++++++++++++++++++++++++++
 2 files changed, 36 insertions(+), 0 deletions(-)

diff --git a/include/linux/if_link.h b/include/linux/if_link.h
index 422084d..84c3492 100644
```

```
--- a/include/linux/if_link.h
+++ b/include/linux/if_link.h
@@ -78,6 +78,7 @@ enum
  IFLA_LINKMODE,
  IFLA_LINKINFO,
 #define IFLA_LINKINFO IFLA_LINKINFO
+ IFLA_NET_NS_PID,
  __IFLA_MAX
 };

diff --git a/net/core/rtnetlink.c b/net/core/rtnetlink.c
index 44f91bb..1b9c32d 100644
--- a/net/core/rtnetlink.c
+++ b/net/core/rtnetlink.c
@@ -35,6 +35,7 @@
 #include <linux/security.h>
 #include <linux/mutex.h>
 #include <linux/if_addr.h>
+#include <linux/nsproxy.h>

 #include <asm/uaccess.h>
 #include <asm/system.h>
@@ -727,6 +728,7 @@ const struct nla_policy ifla_policy[IFLA_MAX+1] = {
  [IFLA_WEIGHT]  = { .type = NLA_U32 },
  [IFLA_OPERSTATE] = { .type = NLA_U8 },
  [IFLA_LINKMODE]  = { .type = NLA_U8 },
+ [IFLA_NET_NS_PID] = { .type = NLA_U32 },
 };

 static const struct nla_policy ifla_info_policy[IFLA_INFO_MAX+1] = {
@@ -734,12 +736,45 @@ static const struct nla_policy ifla_info_policy[IFLA_INFO_MAX+1] = {
  [IFLA_INFO_DATA] = { .type = NLA_NESTED },
 };

+static struct net *get_net_ns_by_pid(pid_t pid)
+{
+ struct task_struct *tsk;
+ struct net *net;
+
+ /* Lookup the network namespace */
+ net = ERR_PTR(-ESRCH);
+ rcu_read_lock();
+ tsk = find_task_by_pid(pid);
+ if (tsk) {
+  task_lock(tsk);
+  if (tsk->nsproxy)
+   net = get_net(tsk->nsproxy->net_ns);
+  task_unlock(tsk);
```

```
+ }
+ rcu_read_unlock();
+ return net;
+}
+
 static int do_setlink(struct net_device *dev, struct ifinfomsg *ifm,
        struct nlattr **tb, char *ifname, int modified)
 {
  int send_addr_notify = 0;
  int err;

+ if (tb[IFLA_NET_NS_PID]) {
+  struct net *net;
+  net = get_net_ns_by_pid(nla_get_u32(tb[IFLA_NET_NS_PID]));
+  if (IS_ERR(net)) {
+   err = PTR_ERR(net);
+   goto errout;
+  }
+  err = dev_change_net_namespace(dev, net, ifname);
+  put_net(net);
+  if (err)
+   goto errout;
+  modified = 1;
+ }
+
  if (tb[IFLA_MAP]) {
   struct rtnl_link_ifmap *u_map;
   struct ifmap k_map;
--
1.5.3.rc6.17.g1911
```

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: [PATCH 17/16] net: Disable netfilter sockopts when not in the initial network namespace

Posted by ebiederm on Sat, 08 Sep 2007 21:47:12 GMT

View Forum Message <> Reply to Message

Until we support multiple network namespaces with netfilter only allow
netfilter configuration in the initial network namespace.

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

Ooops I overlooked this one on my first path through when gathering up this
patchset.

```
 net/netfilter/nf_sockopt.c |    7 +++++++
 1 files changed, 7 insertions(+), 0 deletions(-)

diff --git a/net/netfilter/nf_sockopt.c b/net/netfilter/nf_sockopt.c
index 8b8ece7..c12ea9b 100644
--- a/net/netfilter/nf_sockopt.c
+++ b/net/netfilter/nf_sockopt.c
@@ -80,6 +80,9 @@ static int nf_sockopt(struct sock *sk, int pf, int val,
  struct nf_sockopt_ops *ops;
  int ret;

+ if (sk->sk_net != &init_net)
+  return -ENOPROTOOPT;
+
  if (mutex_lock_interruptible(&nf_sockopt_mutex) != 0)
   return -EINTR;

@@ -138,6 +141,10 @@ static int compat_nf_sockopt(struct sock *sk, int pf, int val,
  struct nf_sockopt_ops *ops;
  int ret;

+ if (sk->sk_net != &init_net)
+  return -ENOPROTOOPT;
+
+
  if (mutex_lock_interruptible(&nf_sockopt_mutex) != 0)
   return -EINTR;

--
1.5.3.rc6.17.g1911
```

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 03/16] net: Basic network namespace infrastructure.
Posted by paulmck on Sun, 09 Sep 2007 00:33:08 GMT
View Forum Message <> Reply to Message

On Sat, Sep 08, 2007 at 03:15:34PM -0600, Eric W. Biederman wrote:
>
> This is the basic infrastructure needed to support network
> namespaces. This infrastructure is:

> - Registration functions to support initializing per network
>   namespace data when a network namespaces is created or destroyed.
>
> - struct net. The network namespace data structure.
>   This structure will grow as variables are made per network
>   namespace but this is the minimal starting point.
>
> - Functions to grab a reference to the network namespace.
>   I provide both get/put functions that keep a network namespace
>   from being freed. And hold/release functions serve as weak references
>   and will warn if their count is not zero when the data structure
>   is freed. Useful for dealing with more complicated data structures
>   like the ipv4 route cache.
>
> - A list of all of the network namespaces so we can iterate over them.
>
> - A slab for the network namespace data structure allowing leaks
>   to be spotted.

If I understand this correctly, the only way to get to a namespace is
via get_net_ns_by_pid(), which contains the rcu_read_lock() that matches
the rcu_barrier() below.

So, is the get_net() in sock_copy() in this patch adding a reference to
an element that is guaranteed to already have at least one reference?
If not, how are we preventing sock_copy() from running concurrently with
cleanup_net()? Ah, I see -- in sock_copy() we are getting a reference
to the new struct sock that no one else can get a reference to, so OK.
Ditto for the get_net() in sk_alloc().

But I still don't understand what is protecting the get_net() in
dev_seq_open(). Is there an existing reference? If so, how do we know
that it won't be removed just as we are trying to add our reference
(while at the same time cleanup_net() is running)? Ditto for the other
_open() operations in the same patch. And for netlink_seq_open().

Enlightenment?

    Thanx, Paul

> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
> ---
> include/net/net_namespace.h | 68 ++++++++++
> net/core/Makefile | 2 +-
> net/core/net_namespace.c | 292 ++++++++++

>
> diff --git a/include/net/net_namespace.h b/include/net/net_namespace.h
> new file mode 100644
> index 0000000..6344b77
> --- /dev/null
> +++ b/include/net/net_namespace.h
> @@ -0,0 +1,68 @@
> +/*
> + * 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/list.h>
> +
> +struct net {
> + atomic_t count; /* To decided when the network
> +   * namespace should be freed.
> +   */
> + atomic_t use_count; /* To track references we
> +   * destroy on demand
> +   */
> + struct list_head list; /* list of network namespaces */
> + struct work_struct work; /* work struct for freeing */
> +};
> +
> +extern struct net init_net;
> +extern struct list_head net_namespace_list;
> +
> +extern void __put_net(struct net *net);
> +
> +static inline struct net *get_net(struct net *net)
> +{
> + atomic_inc(&net->count);
> + return net;
> +}
> +
> +static inline void put_net(struct net *net)
> +{
> + if (atomic_dec_and_test(&net->count))
> + __put_net(net);
> +}
> +
> +static inline struct net *hold_net(struct net *net)
> +{
> + atomic_inc(&net->use_count);

> + return net;
> +}
> +
> +static inline void release_net(struct net *net)
> +{
> + atomic_dec(&net->use_count);
> +}
> +
> +extern void net_lock(void);
> +extern void net_unlock(void);
> +
> +#define for_each_net(VAR)   \
> + list_for_each_entry(VAR, &net_namespace_list, list)
> +
> +
> +struct pernet_operations {
> + struct list_head list;
> + int (*init)(struct net *net);
> + void (*exit)(struct net *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 4751613..ea9b3f3 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..f259a9b
> --- /dev/null
> +++ b/net/core/net_namespace.c
> @@ -0,0 +1,292 @@
> +#include <linux/workqueue.h>
> +#include <linux/rtnetlink.h>

```
> +#include <linux/cache.h>
> +#include <linux/slab.h>
> +#include <linux/list.h>
> +#include <linux/delay.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);
> +
> +static DEFINE_MUTEX(net_list_mutex);
> +LIST_HEAD(net_namespace_list);
> +
> +static struct kmem_cache *net_cachep;
> +
> +struct net init_net;
> +EXPORT_SYMBOL_GPL(init_net);
> +
> +void net_lock(void)
> +{
> + mutex_lock(&net_list_mutex);
> +}
> +
> +void net_unlock(void)
> +{
> + mutex_unlock(&net_list_mutex);
> +}
> +
> +static struct net *net_alloc(void)
> +{
> + return kmem_cache_alloc(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 list_head *ptr;
> + struct net *net;
> +
> + net = container_of(work, struct net, work);
> +
> + mutex_lock(&net_mutex);
> +
> + /* Don't let anyone else find us. */
> + net_lock();
> + list_del(&net->list);
> + net_unlock();
> +
> + /* Run all of the network namespace exit methods */
> + list_for_each_prev(ptr, &pernet_list) {
> +  ops = list_entry(ptr, struct pernet_operations, 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.
> + */
```

```
> +static int setup_net(struct net *net)
> +{
> + /* Must be called with net_mutex held */
> + struct pernet_operations *ops;
> + struct list_head *ptr;
> + int error;
> +
> + memset(net, 0, sizeof(struct net));
> + atomic_set(&net->count, 1);
> + atomic_set(&net->use_count, 0);
> +
> + error = 0;
> + list_for_each(ptr, &pernet_list) {
> +  ops = list_entry(ptr, struct pernet_operations, list);
> +  if (ops->init) {
> +   error = ops->init(net);
> +   if (error < 0)
> +    goto out_undo;
> +  }
> + }
> +out:
> + return error;
> +out_undo:
> + /* Walk through the list backwards calling the exit functions
> +  * for the pernet modules whose init functions did not fail.
> +  */
> + for (ptr = ptr->prev; ptr != &pernet_list; ptr = ptr->prev) {
> +  ops = list_entry(ptr, struct pernet_operations, list);
> +  if (ops->exit)
> +   ops->exit(net);
> + }
> + goto out;
> +}
> +
> +static int __init net_ns_init(void)
> +{
> + int err;
> +
> + printk(KERN_INFO "net_namespace: %zd bytes\n", sizeof(struct net));
> + net_cachep = kmem_cache_create("net_namespace", sizeof(struct net),
> +     SMP_CACHE_BYTES,
> +     SLAB_PANIC, NULL);
> + mutex_lock(&net_mutex);
> + err = setup_net(&init_net);
> +
> + net_lock();
> + list_add_tail(&init_net.list, &net_namespace_list);
> + net_unlock();
```

```
> +
> + mutex_unlock(&net_mutex);
> + if (err)
> +  panic("Could not setup the initial network namespace");
> +
> + return 0;
> +}
> +
> +pure_initcall(net_ns_init);
> +
> +static int register_pernet_operations(struct list_head *list,
> +          struct pernet_operations *ops)
> +{
> + struct net *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 (undo_net == net)
> +   goto undone;
> +  if (ops->exit)
> +   ops->exit(undo_net);
> + }
> +undone:
> + goto out;
> +}
> +
> +static void unregister_pernet_operations(struct pernet_operations *ops)
> +{
> + struct net *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);
> --
> 1.5.3.rc6.17.g1911
>
> -
> To unsubscribe from this list: send the line "unsubscribe netdev" in
> the body of a message to majordomo@vger.kernel.org
> More majordomo info at http://vger.kernel.org/majordomo-info.html

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 03/16] net: Basic network namespace infrastructure.
Posted by Eric Dumazet on Sun, 09 Sep 2007 08:44:49 GMT
View Forum Message <> Reply to Message

> This is the basic infrastructure needed to support network
> namespaces. This infrastructure is:
> - Registration functions to support initializing per network
>   namespace data when a network namespaces is created or destroyed.
>
> - struct net. The network namespace data structure.
>   This structure will grow as variables are made per network
>   namespace but this is the minimal starting point.
>
> - Functions to grab a reference to the network namespace.
>   I provide both get/put functions that keep a network namespace
>   from being freed. And hold/release functions serve as weak references
>   and will warn if their count is not zero when the data structure
>   is freed. Useful for dealing with more complicated data structures
>   like the ipv4 route cache.
>
> - A list of all of the network namespaces so we can iterate over them.
>
> - A slab for the network namespace data structure allowing leaks
>   to be spotted.
>
> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

Nice work Eric !

"struct net" is not a very descriptive name imho, why dont stick "ns" or
"namespace" somewhere ?

Do we really need yet another "struct kmem_cache *net_cachep;" ?
The object is so small that the standard caches should be OK (kzalloc())

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 03/16] net: Basic network namespace infrastructure.
Posted by ebiederm on Sun, 09 Sep 2007 10:04:45 GMT
View Forum Message <> Reply to Message

"Paul E. McKenney" <paulmck@linux.vnet.ibm.com> writes:

> On Sat, Sep 08, 2007 at 03:15:34PM -0600, Eric W. Biederman wrote:
>>
>> This is the basic infrastructure needed to support network
>> namespaces. This infrastructure is:
>> - Registration functions to support initializing per network
>>   namespace data when a network namespaces is created or destroyed.
>>
>> - struct net. The network namespace data structure.
>>   This structure will grow as variables are made per network
>>   namespace but this is the minimal starting point.
>>
>> - Functions to grab a reference to the network namespace.
>>   I provide both get/put functions that keep a network namespace
>>   from being freed. And hold/release functions serve as weak references
>>   and will warn if their count is not zero when the data structure
>>   is freed. Useful for dealing with more complicated data structures
>>   like the ipv4 route cache.
>>
>> - A list of all of the network namespaces so we can iterate over them.
>>
>> - A slab for the network namespace data structure allowing leaks
>>   to be spotted.
>
> If I understand this correctly, the only way to get to a namespace is
> via get_net_ns_by_pid(), which contains the rcu_read_lock() that matches
> the rcu_barrier() below.

Not quite. That is the convoluted case for getting a namespace someone
else is using. current->nsproxy->net_ns works and should require no
locking to read (only the current process may modify it) and does hold
a reference to the network namespace. Similarly for sock->sk_net.

> So, is the get_net() in sock_copy() in this patch adding a reference to
> an element that is guaranteed to already have at least one reference?

Yes.

> If not, how are we preventing sock_copy() from running concurrently with
> cleanup_net()? Ah, I see -- in sock_copy() we are getting a reference
> to the new struct sock that no one else can get a reference to, so OK.
> Ditto for the get_net() in sk_alloc().

> But I still don't understand what is protecting the get_net() in
> dev_seq_open(). Is there an existing reference?

Sort of. The directories under /proc/net are created when create
a network namespace and they are destroyed when the network namespace
is removed. And those directories remember which network namespace
they are for and that is what dev_seq_open is referencing.

So the tricky case what happens if we open a directory under /proc/net
as we are cleaning up a network namespace.

> If so, how do we know
> that it won't be removed just as we are trying to add our reference
> (while at the same time cleanup_net() is running)? Ditto for the other
> _open() operations in the same patch. And for netlink_seq_open().
>
> Enlightenment?

Good spotting. It looks like you have found a legitimate race. Grr.
I thought I had a reference to the network namespace there. I need to
step back and think about this a bit, and see if I can come up with a
legitimate idiom.

I know the network namespace exists and I have not finished
cleanup_net because I can still get to the /proc entries.

I know I cannot use get_net for the reference in in /proc because
otherwise I could not release the network namespace unless I was to
unmount the filesystem, which is not a desirable property.

I think I can change the idiom to:

```
struct net *maybe_get_net(struct net *net)
{
        if (!atomic_inc_not_zero(&net->count))
         net = NULL;
 return net;
```

}

Which would make dev_seq_open be:

```
static int dev_seq_open(struct inode *inode, struct file *file)
{
 struct seq_file *seq;
 int res;
 res =  seq_open(file, &dev_seq_ops);
 if (!res) {
  seq = file->private_data;
  seq->private = maybe_get_net(PROC_NET(inode));
  if (!seq->private) {
   res = -ENOENT;
                        seq_release(inode, file);
  }
 }
 return res;
}
```

I'm still asking myself if I need any kind of locking to ensure
struct net does not go away in the mean time, if so rcu_read_lock()
should be sufficient.

I will read through the generic proc code very carefully after
I have slept and see if there is what I the code above is sufficient,
and if so update the patchset.

Eric

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 03/16] net: Basic network namespace infrastructure.
Posted by ebiederm on Sun, 09 Sep 2007 10:18:22 GMT
View Forum Message <> Reply to Message

Eric Dumazet <dada1@cosmosbay.com> writes:
>
> Nice work Eric !

Thanks.

> "struct net" is not a very descriptive name imho, why dont stick "ns" or
> "namespace" somewhere ?

My fingers rebelled, and struct net seems to be sufficiently descriptive.
However that is a cosmetic detail and if there is a general consensus
that renaming it to be struct netns or whatever would be a more
readable/maintainable name I can change it.

> Do we really need yet another "struct kmem_cache *net_cachep;" ?
> The object is so small that the standard caches should be OK (kzalloc())

The practical issue at this point in the cycle is visibility. With a
kmem cache it is easy to spot ref counting leaks or other problems
if they happen. Without it debugging is much more difficult. While I
am touched with your faith in my ability to write perfect patches I
think it makes a lot of sense to keep the cache at least until
sometime after the network namespace code is merged and people
generally have confidence in the implementation.

Eric

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 03/16] net: Basic network namespace infrastructure.
Posted by paulmck on Sun, 09 Sep 2007 16:45:47 GMT
View Forum Message <> Reply to Message

On Sun, Sep 09, 2007 at 04:04:45AM -0600, Eric W. Biederman wrote:
> "Paul E. McKenney" <paulmck@linux.vnet.ibm.com> writes:
>
> > On Sat, Sep 08, 2007 at 03:15:34PM -0600, Eric W. Biederman wrote:
> >>
> >> This is the basic infrastructure needed to support network
> >> namespaces. This infrastructure is:
> >> - Registration functions to support initializing per network
> >> namespace data when a network namespaces is created or destroyed.
> >>
> >> - struct net. The network namespace data structure.
> >> This structure will grow as variables are made per network
> >> namespace but this is the minimal starting point.
> >>
> >> - Functions to grab a reference to the network namespace.
> >> I provide both get/put functions that keep a network namespace
> >> from being freed. And hold/release functions serve as weak references
> >> and will warn if their count is not zero when the data structure
> >> is freed. Useful for dealing with more complicated data structures
> >> like the ipv4 route cache.
> >>

> >> - A list of all of the network namespaces so we can iterate over them.
> >>
> >> - A slab for the network namespace data structure allowing leaks
> >>   to be spotted.
> >
> > If I understand this correctly, the only way to get to a namespace is
> > via get_net_ns_by_pid(), which contains the rcu_read_lock() that matches
> > the rcu_barrier() below.
>
> Not quite. That is the convoluted case for getting a namespace someone
> else is using. current->nsproxy->net_ns works and should require no
> locking to read (only the current process may modify it) and does hold
> a reference to the network namespace. Similarly for sock->sk_net.

Ah! Got it, thank you for the explanation.

> > So, is the get_net() in sock_copy() in this patch adding a reference to
> > an element that is guaranteed to already have at least one reference?
>
> Yes.
>
> > If not, how are we preventing sock_copy() from running concurrently with
> > cleanup_net()? Ah, I see -- in sock_copy() we are getting a reference
> > to the new struct sock that no one else can get a reference to, so OK.
> > Ditto for the get_net() in sk_alloc().
>
> > But I still don't understand what is protecting the get_net() in
> > dev_seq_open(). Is there an existing reference?
>
> Sort of. The directories under /proc/net are created when create
> a network namespace and they are destroyed when the network namespace
> is removed. And those directories remember which network namespace
> they are for and that is what dev_seq_open is referencing.
>
> So the tricky case what happens if we open a directory under /proc/net
> as we are cleaning up a network namespace.

Yep! ;-)

> > If so, how do we know
> > that it won't be removed just as we are trying to add our reference
> > (while at the same time cleanup_net() is running)? Ditto for the other
> > _open() operations in the same patch. And for netlink_seq_open().
> >
> > Enlightenment?
>
> Good spotting. It looks like you have found a legitimate race. Grr.
> I thought I had a reference to the network namespace there. I need to

> step back and think about this a bit, and see if I can come up with a
> legitimate idiom.
>
> I know the network namespace exists and I have not finished
> cleanup_net because I can still get to the /proc entries.

OK. Hmmm... I need to go review locking for /proc...

> I know I cannot use get_net for the reference in in /proc because
> otherwise I could not release the network namespace unless I was to
> unmount the filesystem, which is not a desirable property.
>
> I think I can change the idiom to:
>
> struct net *maybe_get_net(struct net *net)
> {
>   if (!atomic_inc_not_zero(&net->count))
>   net = NULL;
> return net;
> }
>
> Which would make dev_seq_open be:
>
> static int dev_seq_open(struct inode *inode, struct file *file)
> {
> struct seq_file *seq;
> int res;
> res = seq_open(file, &dev_seq_ops);
> if (!res) {
>   seq = file->private_data;
>   seq->private = maybe_get_net(PROC_NET(inode));
>   if (!seq->private) {
>   res = -ENOENT;
>   seq_release(inode, file);
>   }
> }
> return res;
> }
>
> I'm still asking myself if I need any kind of locking to ensure
> struct net does not go away in the mean time, if so rcu_read_lock()
> should be sufficient.

Agreed -- and it might be possible to leverage the existing locking
in the /proc code.

      Thanx, Paul

> I will read through the generic proc code very carefully after
> I have slept and see if there is what I the code above is sufficient,
> and if so update the patchset.
>
> Eric

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 03/16] net: Basic network namespace infrastructure.
Posted by Krishna Kumar2 on Mon, 10 Sep 2007 05:46:44 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote on 09/09/2007 02:45:34 AM:

Hi Eric,

```
> +static int register_pernet_operations(struct list_head *list,
> +                struct pernet_operations *ops)
> +{
> <snip>
> +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 (undo_net == net)
> +         goto undone;
> +      if (ops->exit)
> +         ops->exit(undo_net);
> +   }
> +undone:
> +   goto out;
> +}
```

You could remove "undone" label (and associated) goto with a "break".

```
> +static void unregister_pernet_operations(struct pernet_operations *ops)
> +{
> +   struct net *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.
> + */
<snip>
> +/**
> + *	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);
> +}

Don't you require something like for_each_net_backwards to 'exit' in
reverse order? Same comment for unregister_subnet_subsys(). Should this
be done for failure in register_pernet_operations too?

thanks,

- KK

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: Re: [PATCH 03/16] net: Basic network namespace infrastructure.
Posted by ebiederm on Mon, 10 Sep 2007 06:32:43 GMT
View Forum Message <> Reply to Message

"Paul E. McKenney" <paulmck@linux.vnet.ibm.com> writes:

>> I know I cannot use get_net for the reference in in /proc because
>> otherwise I could not release the network namespace unless I was to
>> unmount the filesystem, which is not a desirable property.
>>
>> I think I can change the idiom to:
>>
>> struct net *maybe_get_net(struct net *net)
>> {
>>   if (!atomic_inc_not_zero(&net->count))
>>   net = NULL;
>> return net;
>> }
>>
>> Which would make dev_seq_open be:
>>
>> static int dev_seq_open(struct inode *inode, struct file *file)
>> {
>> struct seq_file *seq;
>> int res;
>> res = seq_open(file, &dev_seq_ops);
>> if (!res) {
>>   seq = file->private_data;
>>   seq->private = maybe_get_net(PROC_NET(inode));
>>   if (!seq->private) {
>>   res = -ENOENT;
>>   seq_release(inode, file);
>>   }
>> }
>> return res;
>> }
>>
>> I'm still asking myself if I need any kind of locking to ensure
>> struct net does not go away in the mean time, if so rcu_read_lock()
>> should be sufficient.
>
> Agreed -- and it might be possible to leverage the existing locking

> in the /proc code.

Yes. The generic /proc code takes care of this. It appears
to ensure that any ongoing operations will be waited for and
no more operations will be started once remove_proc_entry
is called. So I just need the maybe_get_net thing to have
safe ref counting.

That is what I thought but I figured I would review that part while
I was looking at everything.

Eric

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 03/16] net: Basic network namespace infrastructure.
Posted by ebiederm on Mon, 10 Sep 2007 06:40:15 GMT
View Forum Message <> Reply to Message

Krishna Kumar2 <krkumar2@in.ibm.com> writes:

> Eric W. Biederman wrote on 09/09/2007 02:45:34 AM:
>
> Hi Eric,
>
>> +static int register_pernet_operations(struct list_head *list,
>> +   struct pernet_operations *ops)
>> +{
>> <snip>
>> +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 (undo_net == net)
>> +   goto undone;
>> +   if (ops->exit)
>> +   ops->exit(undo_net);
>> +   }
>> +undone:
>> +   goto out;
>> +}
>

> You could remove "undone" label (and associated) goto with a "break".

I could but there is there is no guarantee that the for_each_net macro is
implemented with standard C looping construct such that break will work,
and in one of the earlier versions that wasn't the case. So my paranoia
says an explicit label safer and just as clear.

>> +static void unregister_pernet_operations(struct pernet_operations *ops)
>> +{
>> +   struct net *net;
>> +
>> +   list_del(&ops->list);
>> +   for_each_net(net)
>> +   if (ops->exit)
>> +   ops->exit(net);
>> +}
>
> Don't you require something like for_each_net_backwards to 'exit' in
> reverse order? Same comment for unregister_subnet_subsys(). Should this
> be done for failure in register_pernet_operations too?

There are no ordering guarantees between the initialization and
cleanup of different network namespaces.   The only real guarantee
is that the initial network namespace is always present. Which
means any order will work so always doing it in a forwards order
in the list should be fine.

Eric

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 03/16] net: Basic network namespace infrastructure.
Posted by Pavel Emelianov on Mon, 10 Sep 2007 13:16:16 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:

[snip]

> --- /dev/null
> +++ b/include/net/net_namespace.h
> @@ -0,0 +1,68 @@
> +/*
> + * 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/list.h>
> +
> +struct net {

Isn't this name is too generic? Why not net_namespace?

> + atomic_t count; /* To decided when the network
> +   * namespace should be freed.
> +   */
> + atomic_t use_count; /* To track references we
> +   * destroy on demand
> +   */
> + struct list_head list; /* list of network namespaces */
> + struct work_struct work; /* work struct for freeing */
> +};

[snip]

> --- /dev/null
> +++ b/net/core/net_namespace.c

[snip]

> +static int setup_net(struct net *net)
> +{
> + /* Must be called with net_mutex held */
> + struct pernet_operations *ops;
> + struct list_head *ptr;
> + int error;
> +
> + memset(net, 0, sizeof(struct net));
> + atomic_set(&net->count, 1);
> + atomic_set(&net->use_count, 0);
> +
> + error = 0;
> + list_for_each(ptr, &pernet_list) {
> + ops = list_entry(ptr, struct pernet_operations, list);
> + if (ops->init) {
> +   error = ops->init(net);
> +   if (error < 0)
> +   goto out_undo;
> + }
> + }

```
> +out:
> + return error;
> +out_undo:
> + /* Walk through the list backwards calling the exit functions
> +  * for the pernet modules whose init functions did not fail.
> +  */
> + for (ptr = ptr->prev; ptr != &pernet_list; ptr = ptr->prev) {
```

Good reason for adding list_for_each_continue_reverse :)

```
> +  ops = list_entry(ptr, struct pernet_operations, list);
> +  if (ops->exit)
> +   ops->exit(net);
> + }
> + goto out;
> +}
> +
> +static int __init net_ns_init(void)
> +{
> + int err;
> +
> + printk(KERN_INFO "net_namespace: %zd bytes\n", sizeof(struct net));
> + net_cachep = kmem_cache_create("net_namespace", sizeof(struct net),
> +     SMP_CACHE_BYTES,
> +     SLAB_PANIC, NULL);
> + mutex_lock(&net_mutex);
> + err = setup_net(&init_net);
> +
> + net_lock();
> + list_add_tail(&init_net.list, &net_namespace_list);
> + net_unlock();
> +
> + mutex_unlock(&net_mutex);
> + if (err)
> +  panic("Could not setup the initial network namespace");
> +
> + return 0;
> +}
> +
> +pure_initcall(net_ns_init);
> +
> +static int register_pernet_operations(struct list_head *list,
> +          struct pernet_operations *ops)
> +{
> + struct net *net, *undo_net;
> + int error;
> +
> + error = 0;
```

> + list_add_tail(&ops->list, list);
> + for_each_net(net) {
> + if (ops->init) {

Maybe it's better to do it vice-versa?
if (ops->init)
  for_each_net(net)
    ops->init(net);
...

> +   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 (undo_net == net)
> +   goto undone;
> + if (ops->exit)
> +   ops->exit(undo_net);
> + }
> +undone:
> + goto out;
> +}
> +
> +static void unregister_pernet_operations(struct pernet_operations *ops)
> +{
> + struct net *net;
> +
> + list_del(&ops->list);
> + for_each_net(net)
> + if (ops->exit)

The same here.

> +   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))

Very minor: why do you give the name "device" to some pernet_operations?

Thanks,
Pavel

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 12/16] net: Support multiple network namespaces with netlink
Posted by Pavel Emelianov on Mon, 10 Sep 2007 13:46:48 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> Each netlink socket will live in exactly one network namespace,
> this includes the controlling kernel sockets.
>
> This patch updates all of the existing netlink protocols
> to only support the initial network namespace. Request
> by clients in other namespaces will get -ECONREFUSED.
> As they would if the kernel did not have the support for
> that netlink protocol compiled in.
>
> As each netlink protocol is updated to be multiple network
> namespace safe it can register multiple kernel sockets
> to acquire a presence in the rest of the network namespaces.
>

> The implementation in af_netlink is a simple filter implementation
> at hash table insertion and hash table look up time.
>
> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
> ---
> drivers/connector/connector.c | 2 +-
> drivers/scsi/scsi_netlink.c | 2 +-
> drivers/scsi/scsi_transport_iscsi.c | 2 +-
> fs/ecryptfs/netlink.c | 2 +-
> include/linux/netlink.h | 6 ++-
> kernel/audit.c | 4 +-
> lib/kobject_uevent.c | 5 +-
> net/bridge/netfilter/ebt_ulog.c | 5 +-
> net/core/rtnetlink.c | 4 +-
> net/decnet/netfilter/dn_rtmsg.c | 3 +-
> net/ipv4/fib_frontend.c | 4 +-
> net/ipv4/inet_diag.c | 4 +-
> net/ipv4/netfilter/ip_queue.c | 6 +-
> net/ipv4/netfilter/ipt_ULOG.c | 3 +-
> net/ipv6/netfilter/ip6_queue.c | 6 +-
> net/netfilter/nfnetlink.c | 2 +-
> net/netfilter/nfnetlink_log.c | 3 +-
> net/netfilter/nfnetlink_queue.c | 3 +-
> net/netlink/af_netlink.c | 106 ++++++++++++++++++++++++++++---------
> net/netlink/genetlink.c | 4 +-
> net/xfrm/xfrm_user.c | 2 +-
> security/selinux/netlink.c | 5 +-
> 22 files changed, 122 insertions(+), 61 deletions(-)

Rrrrrr. This is the 5th or even the 6th patch that changes tens of files
but (!) most of these changes are just propagating some core thing into
protocols, drivers, etc. E.g. you add an argument to some function and
then make all the rest use it, but the chunk adding the argument itself
is buried in these changes.

Why not make a reviewers' lifes easier and make (with hands) the core
hunks go first and the "propagation" ones at the end? For RFC purpose
I would even break the git-bisect safeness and splitted these patches
into 2 parts: those with the core and those with the propagation.

Thanks,
Pavel

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: Re: [PATCH 17/16] net: Disable netfilter sockopts when not in the initial network namespace
Posted by Pavel Emelianov on Mon, 10 Sep 2007 13:50:14 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> Until we support multiple network namespaces with netfilter only allow
> netfilter configuration in the initial network namespace.

PATCH 17/16? :)

Sorry,
Pavel

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: Re: [PATCH 12/16] net: Support multiple network namespaces with netlink
Posted by ebiederm on Mon, 10 Sep 2007 15:24:23 GMT
View Forum Message <> Reply to Message

Pavel Emelyanov <xemul@openvz.org> writes:
>
> Rrrrrr. This is the 5th or even the 6th patch that changes tens of files
> but (!) most of these changes are just propagating some core thing into
> protocols, drivers, etc. E.g. you add an argument to some function and
> then make all the rest use it, but the chunk adding the argument itself
> is buried in these changes.
>
> Why not make a reviewers' lifes easier and make (with hands) the core
> hunks go first and the "propagation" ones at the end? For RFC purpose
> I would even break the git-bisect safeness and splitted these patches
> into 2 parts: those with the core and those with the propagation.

Agreed, this is an issue for easy review. My apologies.

I guess it was a failure in my imagination on how to prepare these
patches for review, in a way that was both reviewer and preparer friendly.

There is at least one /proc idiom change that needs to be made so I
will keep this in mind for the resend.

Eric

_____
Containers mailing list
Containers@lists.linux-foundation.org

Subject: Re: [PATCH 17/16] net: Disable netfilter sockopts when not in the initial network namespace
Posted by ebiederm on Mon, 10 Sep 2007 15:27:54 GMT
View Forum Message <> Reply to Message

Pavel Emelyanov <xemul@openvz.org> writes:

> Eric W. Biederman wrote:
>> Until we support multiple network namespaces with netfilter only allow
>> netfilter configuration in the initial network namespace.
>
> PATCH 17/16? :)

Exactly!

If my target was the core of the networking stack I figured I better include the change that keeps netfilter commands isolated to the initial network namespace, and in my review of completeness I had missed that in my first pass through my patches.

Eric

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: Re: [PATCH 03/16] net: Basic network namespace infrastructure.
Posted by ebiederm on Mon, 10 Sep 2007 15:53:55 GMT
View Forum Message <> Reply to Message

Pavel Emelyanov <xemul@openvz.org> writes:

> Eric W. Biederman wrote:
>
> [snip]
>
>> --- /dev/null
>> +++ b/include/net/net_namespace.h
>> @@ -0,0 +1,68 @@
>> +/*
>> + * 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/list.h>
>> +
>> +struct net {
>
> Isn't this name is too generic? Why not net_namespace?

My fingers went on strike. struct netns probably wouldn't be bad.
The very long and spelled out version seemed painful. I don't really
care except that not changing it is easier for me. I'm happy to do
whatever is considered most maintainable.

>> + /* Walk through the list backwards calling the exit functions
>> + * for the pernet modules whose init functions did not fail.
>> + */
>> + for (ptr = ptr->prev; ptr != &pernet_list; ptr = ptr->prev) {
>
> Good reason for adding list_for_each_continue_reverse :)

Sounds reasonable.

>> +static int register_pernet_operations(struct list_head *list,
>> +   struct pernet_operations *ops)
>> +{
>> + struct net *net, *undo_net;
>> + int error;
>> +
>> + error = 0;
>> + list_add_tail(&ops->list, list);
>> + for_each_net(net) {
>> + if (ops->init) {
>
> Maybe it's better to do it vice-versa?
> if (ops->init)
>   for_each_net(net)
>   ops->init(net);
> ...

My gut feel says it is more readable with the test on the inside.
Although something like
if (ops->init)
  goto out;

might be more readable.

>> +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))
>
> Very minor: why do you give the name "device" to some pernet_operations?

Subsystems need to be initialized before and cleaned up after network
devices. We don't have much in the way that needs this distinction,
but we have just enough that it is useful to make this distinction.

Looking at my complete patchset all I have in this category is the
loopback device, and it is important on the teardown side of things
that the loopback device be unregistered before I clean up the
protocols or else I get weird leaks.

Reflecting on it I'm not quite certain about the setup side of things.
I'm on the fence if I need to completely dynamically allocate the
loopback device or if I need to embed it struct net.

There also may be a call for some other special network devices
to have one off instances in each network namespace. With the
new netlink creation API it isn't certain we need that idiom anymore
but before that point I was certain we would have other network
devices besides the loopback that would care.

Eric

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 16/16] net: netlink support for moving devices between network namespaces.
Posted by serue on Mon, 10 Sep 2007 19:07:08 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
>
> The simplest thing to implement is moving network devices between
> namespaces. However with the same attribute IFLA_NET_NS_PID we can
> easily implement creating devices in the destination network
> namespace as well. However that is a little bit trickier so this
> patch sticks to what is simple and easy.

>
> A pid is used to identify a process that happens to be a member
> of the network namespace we want to move the network device to.
>
> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
> ---
> include/linux/if_link.h | 1 +
> net/core/rtnetlink.c | 35 +++++++++++++++++++++++++++++++++++
> 2 files changed, 36 insertions(+), 0 deletions(-)
>
> diff --git a/include/linux/if_link.h b/include/linux/if_link.h
> index 422084d..84c3492 100644
> --- a/include/linux/if_link.h
> +++ b/include/linux/if_link.h
> @@ -78,6 +78,7 @@ enum
>   IFLA_LINKMODE,
>   IFLA_LINKINFO,
> #define IFLA_LINKINFO IFLA_LINKINFO
> + IFLA_NET_NS_PID,
>   __IFLA_MAX
> };
>
> diff --git a/net/core/rtnetlink.c b/net/core/rtnetlink.c
> index 44f91bb..1b9c32d 100644
> --- a/net/core/rtnetlink.c
> +++ b/net/core/rtnetlink.c
> @@ -35,6 +35,7 @@
> #include <linux/security.h>
> #include <linux/mutex.h>
> #include <linux/if_addr.h>
> +#include <linux/nsproxy.h>
>
> #include <asm/uaccess.h>
> #include <asm/system.h>
> @@ -727,6 +728,7 @@ const struct nla_policy ifla_policy[IFLA_MAX+1] = {
>   [IFLA_WEIGHT] = { .type = NLA_U32 },
>   [IFLA_OPERSTATE] = { .type = NLA_U8 },
>   [IFLA_LINKMODE] = { .type = NLA_U8 },
> + [IFLA_NET_NS_PID] = { .type = NLA_U32 },
> };
>
> static const struct nla_policy ifla_info_policy[IFLA_INFO_MAX+1] = {
> @@ -734,12 +736,45 @@ static const struct nla_policy ifla_info_policy[IFLA_INFO_MAX+1] = {
>   [IFLA_INFO_DATA] = { .type = NLA_NESTED },
> };
>
> +static struct net *get_net_ns_by_pid(pid_t pid)
> +{

> + struct task_struct *tsk;
> + struct net *net;
> +
> + /* Lookup the network namespace */
> + net = ERR_PTR(-ESRCH);
> + rcu_read_lock();
> + tsk = find_task_by_pid(pid);
> + if (tsk) {
> + task_lock(tsk);
> + if (tsk->nsproxy)
> +   net = get_net(tsk->nsproxy->net_ns);
> + task_unlock(tsk);

Thinking... Ok, I'm not sure this is 100% safe in the target tree, but the long-term correct way probably isn't yet implemented in the net-tree. Eventually you will want to:

```
net_ns = NULL;
rcu_read_lock();
tsk = find_task_by_pid();  /* or _pidns equiv? */
nsproxy = task_nsproxy(tsk);
if (nsproxy)
 net_ns = get_net(nsproxy->net_ns);
rcu_read_unlock;
```

What you have here is probably unsafe if tsk is the last task pointing to it's nsproxy and it does an unshare, bc unshare isn't protected by task_lock, and you're not rcu_dereferencing tsk->nsproxy (which task_nsproxy does). At one point we floated a patch to reuse the same nsproxy in that case which would prevent you having to worry about it, but that isn't being done in -mm now so i doubt it's in -net.

> + }
> + rcu_read_unlock();
> + return net;
> +}
> +
> static int do_setlink(struct net_device *dev, struct ifinfomsg *ifm,
>   struct nlattr **tb, char *ifname, int modified)
> {
>   int send_addr_notify = 0;
>   int err;
>
> + if (tb[IFLA_NET_NS_PID]) {
> + struct net *net;
> + net = get_net_ns_by_pid(nla_get_u32(tb[IFLA_NET_NS_PID]));
> + if (IS_ERR(net)) {
> +   err = PTR_ERR(net);

> +   goto errout;
> + }
> + err = dev_change_net_namespace(dev, net, ifname);
> + put_net(net);
> + if (err)
> +   goto errout;
> + modified = 1;
> + }
> +
>   if (tb[IFLA_MAP]) {
>   struct rtnl_link_ifmap *u_map;
>   struct ifmap k_map;
> --
> 1.5.3.rc6.17.g1911
>
> _____
> Containers mailing list
> Containers@lists.linux-foundation.org
> https://lists.linux-foundation.org/mailman/listinfo/containers

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 16/16] net: netlink support for moving devices between network namespaces.
Posted by ebiederm on Mon, 10 Sep 2007 19:30:46 GMT
View Forum Message <> Reply to Message

"Serge E. Hallyn" <serue@us.ibm.com> writes:
>>
>> +static struct net *get_net_ns_by_pid(pid_t pid)
>> +{
>> + struct task_struct *tsk;
>> + struct net *net;
>> +
>> + /* Lookup the network namespace */
>> + net = ERR_PTR(-ESRCH);
>> + rcu_read_lock();
>> + tsk = find_task_by_pid(pid);
>> + if (tsk) {
>> + task_lock(tsk);
>> + if (tsk->nsproxy)
>> +   net = get_net(tsk->nsproxy->net_ns);
>> + task_unlock(tsk);
>
> Thinking... Ok, I'm not sure this is 100% safe in the target tree, but

> the long-term correct way probably isn't yet implemented in the net-
> tree. Eventually you will want to:
>
> net_ns = NULL;
> rcu_read_lock();
> tsk = find_task_by_pid(); /* or _pidns equiv? */
> nsproxy = task_nsproxy(tsk);
> if (nsproxy)
>   net_ns = get_net(nsproxy->net_ns);
> rcu_read_unlock;
>
> What you have here is probably unsafe if tsk is the last task pointing
> to it's nsproxy and it does an unshare, bc unshare isn't protected by
> task_lock, and you're not rcu_dereferencing tsk->nsproxy (which
> task_nsproxy does). At one point we floated a patch to reuse the same
> nsproxy in that case which would prevent you having to worry about it,
> but that isn't being done in -mm now so i doubt it's in -net.

That change isn't merged upstream yet, so it isn't in David's
net-2.6.24 tree. Currently task->nsproxy is protected but
task_lock(current). So the code is fine.

I am aware that removing the task_lock(current) for the setting
of current->nsproxy is currently in the works, and I have planned
to revisit this later when all of these pieces come together.

For now the code is fine.

If need be we can drop this patch to remove the potential merge
conflict. But I figured it was useful for this part of the user space
interface to be available for review.

Eric

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 16/16] net: netlink support for moving devices between network namespaces.
Posted by serue on Tue, 11 Sep 2007 00:54:04 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> "Serge E. Hallyn" <serue@us.ibm.com> writes:
> >>

> >> +static struct net *get_net_ns_by_pid(pid_t pid)
> >> +{
> >> + struct task_struct *tsk;
> >> + struct net *net;
> >> +
> >> + /* Lookup the network namespace */
> >> + net = ERR_PTR(-ESRCH);
> >> + rcu_read_lock();
> >> + tsk = find_task_by_pid(pid);
> >> + if (tsk) {
> >> + task_lock(tsk);
> >> + if (tsk->nsproxy)
> >> +   net = get_net(tsk->nsproxy->net_ns);
> >> + task_unlock(tsk);
> >
> > Thinking... Ok, I'm not sure this is 100% safe in the target tree, but
> > the long-term correct way probably isn't yet implemented in the net-
> > tree. Eventually you will want to:
> >
> > net_ns = NULL;
> > rcu_read_lock();
> > tsk = find_task_by_pid(); /* or _pidns equiv? */
> > nsproxy = task_nsproxy(tsk);
> > if (nsproxy)
> >   net_ns = get_net(nsproxy->net_ns);
> > rcu_read_unlock;
> >
> > What you have here is probably unsafe if tsk is the last task pointing
> > to it's nsproxy and it does an unshare, bc unshare isn't protected by
> > task_lock, and you're not rcu_dereferencing tsk->nsproxy (which
> > task_nsproxy does). At one point we floated a patch to reuse the same
> > nsproxy in that case which would prevent you having to worry about it,
> > but that isn't being done in -mm now so i doubt it's in -net.
>
>
> That change isn't merged upstream yet, so it isn't in David's
> net-2.6.24 tree. Currently task->nsproxy is protected but
> task_lock(current). So the code is fine.
>
> I am aware that removing the task_lock(current) for the setting
> of current->nsproxy is currently in the works, and I have planned
> to revisit this later when all of these pieces come together.
>
> For now the code is fine.
>
> If need be we can drop this patch to remove the potential merge
> conflict.

No, no. Like you say it's correct at the moment. Just something we
need to watch out for when it does get merged with the newer changes.

> But I figured it was useful

Absolutely.

> for this part of the user space
> interface to be available for review.

Agreed. And the rest of the patchset looks good to me.

Thanks.

-serge

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 01/16] appletalk: In notifier handlers convert the void pointer to a netdevice
Posted by davem on Wed, 12 Sep 2007 09:27:56 GMT
View Forum Message <> Reply to Message

From: ebiederm@xmission.com (Eric W. Biederman)
Date: Sat, 08 Sep 2007 15:09:36 -0600

>
> This slightly improves code safetly and clarity.
>
> Later network namespace patches touch this code so this is a
> preliminary cleanup.
>
> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

Applied to net-2.6.24

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 02/16] net: Don't implement dev_ifname32 inline
Posted by davem on Wed, 12 Sep 2007 09:39:40 GMT
View Forum Message <> Reply to Message

From: ebiederm@xmission.com (Eric W. Biederman)
Date: Sat, 08 Sep 2007 15:13:04 -0600

>
> The current implementation of dev_ifname makes maintenance difficult
> because updates to the implementation of the ioctl have to made in two
> places. So this patch updates dev_ifname32 to do a classic 32/64
> structure conversion and call sys_ioctl like the rest of the
> compat calls do.
>
> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

Applied to net-2.6.24, thanks.

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 03/16] net: Basic network namespace infrastructure.
Posted by davem on Wed, 12 Sep 2007 09:52:44 GMT

View Forum Message <> Reply to Message

From: ebiederm@xmission.com (Eric W. Biederman)
Date: Sat, 08 Sep 2007 15:15:34 -0600

>
> This is the basic infrastructure needed to support network
> namespaces. This infrastructure is:
> - Registration functions to support initializing per network
>   namespace data when a network namespaces is created or destroyed.
>
> - struct net. The network namespace data structure.
>   This structure will grow as variables are made per network
>   namespace but this is the minimal starting point.
>
> - Functions to grab a reference to the network namespace.
>   I provide both get/put functions that keep a network namespace
>   from being freed. And hold/release functions serve as weak references
>   and will warn if their count is not zero when the data structure
>   is freed. Useful for dealing with more complicated data structures
>   like the ipv4 route cache.
>
> - A list of all of the network namespaces so we can iterate over them.
>
> - A slab for the network namespace data structure allowing leaks
>   to be spotted.
>

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

I realize there are some discussions about naming and fixing
some races, but I applied this anyways so we can make some
forward progress.

We can make name changes and fixes on top of this initial work.

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 04/16] net: Add a network namespace parameter to tasks
Posted by davem on Wed, 12 Sep 2007 09:55:54 GMT
View Forum Message <> Reply to Message

From: ebiederm@xmission.com (Eric W. Biederman)
Date: Sat, 08 Sep 2007 15:17:03 -0600

>
> This is the network namespace from which all which all sockets
> and anything else under user control ultimately get their network
> namespace parameters.
>
> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

Applied to net-2.6.24

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 05/16] net: Add a network namespace tag to struct net_device
Posted by davem on Wed, 12 Sep 2007 09:57:09 GMT
View Forum Message <> Reply to Message

From: ebiederm@xmission.com (Eric W. Biederman)
Date: Sat, 08 Sep 2007 15:18:12 -0600

>
> Please note that network devices do not increase the count
> count on the network namespace. The are inside the network
> namespace and so the network namespace tag is in the nature
> of a back pointer and so getting and putting the network namespace
> is unnecessary.

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

Applied to net-2.6.24, thanks.

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 06/16] net: Add a network namespace parameter to struct sock
Posted by davem on Wed, 12 Sep 2007 09:58:50 GMT
View Forum Message <> Reply to Message

From: ebiederm@xmission.com (Eric W. Biederman)
Date: Sat, 08 Sep 2007 15:21:37 -0600

>
> Sockets need to get a reference to their network namespace,
> or possibly a simple hold if someone registers on the network
> namespace notifier and will free the sockets when the namespace
> is going to be destroyed.
>
> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

Applied, thanks.

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 07/16] net: Make /proc/net per network namespace
Posted by davem on Wed, 12 Sep 2007 10:02:11 GMT
View Forum Message <> Reply to Message

From: ebiederm@xmission.com (Eric W. Biederman)
Date: Sat, 08 Sep 2007 15:20:36 -0600

>
> This patch makes /proc/net per network namespace. It modifies the global
> variables proc_net and proc_net_stat to be per network namespace.
> The proc_net file helpers are modified to take a network namespace argument,
> and all of their callers are fixed to pass &init_net for that argument.
> This ensures that all of the /proc/net files are only visible and
> usable in the initial network namespace until the code behind them

> has been updated to be handle multiple network namespaces.
>
> Making /proc/net per namespace is necessary as at least some files
> in /proc/net depend upon the set of network devices which is per
> network namespace, and even more files in /proc/net have contents
> that are relevant to a single network namespace.
>
> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

Patch applied, thanks.

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 08/16] net: Make socket creation namespace safe.
Posted by davem on Wed, 12 Sep 2007 10:04:27 GMT
View Forum Message <> Reply to Message

From: ebiederm@xmission.com (Eric W. Biederman)
Date: Sat, 08 Sep 2007 15:23:01 -0600

>
> This patch passes in the namespace a new socket should be created in
> and has the socket code do the appropriate reference counting. By
> virtue of this all socket create methods are touched. In addition
> the socket create methods are modified so that they will fail if
> you attempt to create a socket in a non-default network namespace.
>
> Failing if we attempt to create a socket outside of the default
> network namespace ensures that as we incrementally make the network stack
> network namespace aware we will not export functionality that someone
> has not audited and made certain is network namespace safe.
> Allowing us to partially enable network namespaces before all of the
> exotic protocols are supported.
>
> Any protocol layers I have missed will fail to compile because I now
> pass an extra parameter into the socket creation code.
>
> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

Patch applied, thanks.

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: Re: [PATCH 09/16] net: Initialize the network namespace of network devices.
Posted by davem on Wed, 12 Sep 2007 10:58:19 GMT
View Forum Message <> Reply to Message

From: ebiederm@xmission.com (Eric W. Biederman)
Date: Sat, 08 Sep 2007 15:24:21 -0600

>
> Except for carefully selected pseudo devices all network
> interfaces should start out in the initial network namespace.
> Ultimately it will be register_netdev that examines what
> dev->nd_net is set to and places a device in a network namespace.
>
> This patch modifies alloc_netdev to initialize the network
> namespace a device is in with the initial network namespace.
> This gets it right for the vast majority of devices so their
> drivers need not be modified and for those few pseudo devices
> that need something different they can change this parameter
> before calling register_netdevice.
>
> The network namespace parameter on a network device is not
> reference counted as the devices are inside of a network namespace
> and cannot remain in that namespace past the lifetime of the
> network namespace.
>
> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

Applied to net-2.6.24, thanks.

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: Re: [PATCH 10/16] net: Make packet reception network namespace safe
Posted by davem on Wed, 12 Sep 2007 11:00:47 GMT
View Forum Message <> Reply to Message

From: ebiederm@xmission.com (Eric W. Biederman)
Date: Sat, 08 Sep 2007 15:25:43 -0600

>
> This patch modifies every packet receive function
> registered with dev_add_pack() to drop packets if they
> are not from the initial network namespace.
>
> This should ensure that the various network stacks do

> not receive packets in a anything but the initial network
> namespace until the code has been converted and is ready
> for them.
>
> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

Applied to net-2.6.24, thanks.

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 11/16] net: Make device event notification network namespace safe
Posted by davem on Wed, 12 Sep 2007 11:02:55 GMT
View Forum Message <> Reply to Message

From: ebiederm@xmission.com (Eric W. Biederman)
Date: Sat, 08 Sep 2007 15:27:11 -0600

>
> Every user of the network device notifiers is either a protocol
> stack or a pseudo device. If a protocol stack that does not have
> support for multiple network namespaces receives an event for a
> device that is not in the initial network namespace it quite possibly
> can get confused and do the wrong thing.
>
> To avoid problems until all of the protocol stacks are converted
> this patch modifies all netdev event handlers to ignore events on
> devices that are not in the initial network namespace.
>
> As the rest of the code is made network namespace aware these
> checks can be removed.
>
> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

Applied, thanks.

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 12/16] net: Support multiple network namespaces with netlink
Posted by davem on Wed, 12 Sep 2007 11:06:16 GMT
View Forum Message <> Reply to Message

From: ebiederm@xmission.com (Eric W. Biederman)
Date: Sat, 08 Sep 2007 15:28:27 -0600

>
> Each netlink socket will live in exactly one network namespace,
> this includes the controlling kernel sockets.
>
> This patch updates all of the existing netlink protocols
> to only support the initial network namespace. Request
> by clients in other namespaces will get -ECONREFUSED.
> As they would if the kernel did not have the support for
> that netlink protocol compiled in.
>
> As each netlink protocol is updated to be multiple network
> namespace safe it can register multiple kernel sockets
> to acquire a presence in the rest of the network namespaces.
>
> The implementation in af_netlink is a simple filter implementation
> at hash table insertion and hash table look up time.
>
> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

Applied to net-2.6.24, thanks.

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 13/16] net: Make the device list and device lookups per namespace.
Posted by davem on Wed, 12 Sep 2007 11:39:47 GMT
View Forum Message <> Reply to Message

From: ebiederm@xmission.com (Eric W. Biederman)
Date: Sat, 08 Sep 2007 15:35:46 -0600

>
> This patch makes most of the generic device layer network
> namespace safe. This patch makes dev_base_head a
> network namespace variable, and then it picks up
> a few associated variables. The functions:
> dev_getbyhwaddr
> dev_getfirsthwbytype
> dev_get_by_flags
> dev_get_by_name
> __dev_get_by_name
> dev_get_by_index

> __dev_get_by_index
> dev_ioctl
> dev_ethtool
> dev_load
> wireless_process_ioctl
>
> were modified to take a network namespace argument, and
> deal with it.
>
> vlan_ioctl_set and brioctl_set were modified so their
> hooks will receive a network namespace argument.
>
> So basically anthing in the core of the network stack that was
> affected to by the change of dev_base was modified to handle
> multiple network namespaces. The rest of the network stack was
> simply modified to explicitly use &init_net the initial network
> namespace. This can be fixed when those components of the network
> stack are modified to handle multiple network namespaces.
>
> For now the ifindex generator is left global.
>
> Fundametally ifindex numbers are per namespace, or else
> we will have corner case problems with migration when
> we get that far.
>
> At the same time there are assumptions in the network stack
> that the ifindex of a network device won't change. Making
> the ifindex number global seems a good compromise until
> the network stack can cope with ifindex changes when
> you change namespaces, and the like.
>
> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

Applied to net-2.6.24, thanks.

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 14/16] net: Factor out __dev_alloc_name from dev_alloc_name
Posted by davem on Wed, 12 Sep 2007 11:49:19 GMT
View Forum Message <> Reply to Message

From: ebiederm@xmission.com (Eric W. Biederman)
Date: Sat, 08 Sep 2007 15:36:56 -0600

>
> When forcibly changing the network namespace of a device
> I need something that can generate a name for the device
> in the new namespace without overwriting the old name.
>
> __dev_alloc_name provides me that functionality.
>
> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

Applied to net-2.6.24, thanks.

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 15/16] net: Implement network device movement between namespaces
Posted by davem on Wed, 12 Sep 2007 11:54:36 GMT
View Forum Message <> Reply to Message

From: ebiederm@xmission.com (Eric W. Biederman)
Date: Sat, 08 Sep 2007 15:38:46 -0600

>
> This patch introduces NETIF_F_NETNS_LOCAL a flag to indicate
> a network device is local to a single network namespace and
> should never be moved. Useful for pseudo devices that we
> need an instance in each network namespace (like the loopback
> device) and for any device we find that cannot handle multiple
> network namespaces so we may trap them in the initial network
> namespace.
>
> This patch introduces the function dev_change_net_namespace
> a function used to move a network device from one network
> namespace to another. To the network device nothing
> special appears to happen, to the components of the network
> stack it appears as if the network device was unregistered
> in the network namespace it is in, and a new device
> was registered in the network namespace the device
> was moved to.
>
> This patch sets up a namespace device destructor that
> upon the exit of a network namespace moves all of the
> movable network devices to the initial network namespace
> so they are not lost.
>

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

Applied to net-2.6.24

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 16/16] net: netlink support for moving devices between network namespaces.
Posted by davem on Wed, 12 Sep 2007 11:57:40 GMT
View Forum Message <> Reply to Message

From: ebiederm@xmission.com (Eric W. Biederman)
Date: Sat, 08 Sep 2007 15:43:44 -0600

>
> The simplest thing to implement is moving network devices between
> namespaces. However with the same attribute IFLA_NET_NS_PID we can
> easily implement creating devices in the destination network
> namespace as well. However that is a little bit trickier so this
> patch sticks to what is simple and easy.
>
> A pid is used to identify a process that happens to be a member
> of the network namespace we want to move the network device to.
>
> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

Applied to net-2.6.24, thanks.

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 17/16] net: Disable netfilter sockopts when not in the initial network namespace
Posted by davem on Wed, 12 Sep 2007 11:59:03 GMT
View Forum Message <> Reply to Message

From: ebiederm@xmission.com (Eric W. Biederman)
Date: Sat, 08 Sep 2007 15:47:12 -0600

>
> Until we support multiple network namespaces with netfilter only allow
> netfilter configuration in the initial network namespace.

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

Applied to net-2.6.24, thanks.

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

---

Subject: Re: [PATCH 17/16] net: Disable netfilter sockopts when not in the initial network namespace
Posted by davem on Wed, 12 Sep 2007 12:03:11 GMT
View Forum Message <> Reply to Message

I added the following patch to net-2.6.24 to kill a warning
since net_alloc() has no users (yet).

commit f444fa9b5d70b3d431e1554e0975e012514c39f3
Author: David S. Miller <davem@kimchee.(none)>
Date: Wed Sep 12 14:01:08 2007 +0200

    [NET]: #if 0 out net_alloc() for now.

    We will undo this once it is actually used.

    Signed-off-by: David S. Miller <davem@davemloft.net>

```
diff --git a/net/core/net_namespace.c b/net/core/net_namespace.c
index f259a9b..1fc513c 100644
--- a/net/core/net_namespace.c
+++ b/net/core/net_namespace.c
@@ -32,10 +32,12 @@ void net_unlock(void)
  mutex_unlock(&net_list_mutex);
 }

+#if 0
 static struct net *net_alloc(void)
 {
  return kmem_cache_alloc(net_cachep, GFP_KERNEL);
 }
+#endif

 static void net_free(struct net *net)
 {
```

_____
Containers mailing list
Containers@lists.linux-foundation.org

Subject: Re: [PATCH 07/16] net: Make /proc/net per network namespace
Posted by Daniel Lezcano on Wed, 12 Sep 2007 12:12:04 GMT
View Forum Message <> Reply to Message

David Miller wrote:
> From: ebiederm@xmission.com (Eric W. Biederman)
> Date: Sat, 08 Sep 2007 15:20:36 -0600
>
>> This patch makes /proc/net per network namespace. It modifies the global
>> variables proc_net and proc_net_stat to be per network namespace.
>> The proc_net file helpers are modified to take a network namespace argument,
>> and all of their callers are fixed to pass &init_net for that argument.
>> This ensures that all of the /proc/net files are only visible and
>> usable in the initial network namespace until the code behind them
>> has been updated to be handle multiple network namespaces.
>>
>> Making /proc/net per namespace is necessary as at least some files
>> in /proc/net depend upon the set of network devices which is per
>> network namespace, and even more files in /proc/net have contents
>> that are relevant to a single network namespace.
>>
>> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
>
> Patch applied, thanks.
> _____
> Containers mailing list
> Containers@lists.linux-foundation.org
> https://lists.linux-foundation.org/mailman/listinfo/containers
>

Hi Dave,

it seems the fs/proc/proc_net.c was not added to the git repository.

Regards.

 -- Daniel

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: Re: [PATCH 17/16] net: Disable netfilter sockopts when not in the initial

network namespace
Posted by ebiederm on Wed, 12 Sep 2007 12:16:25 GMT
View Forum Message <> Reply to Message

David Miller <davem@davemloft.net> writes:

> I added the following patch to net-2.6.24 to kill a warning
> since net_alloc() has no users (yet).

Reasonable, and thanks for merging these.

Having a solid place to start helps a lot.

I will see if I can get the /proc races fixed shortly.

Eric

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: Re: [PATCH 07/16] net: Make /proc/net per network namespace
Posted by davem on Wed, 12 Sep 2007 12:19:32 GMT
View Forum Message <> Reply to Message

From: Daniel Lezcano <dlezcano@fr.ibm.com>
Date: Wed, 12 Sep 2007 14:12:04 +0200

> it seems the fs/proc/proc_net.c was not added to the git repository.

Fixed, thanks for catching that.

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers

Subject: Re: [PATCH 06/16] net: Add a network namespace parameter to struct sock
Posted by den on Thu, 20 Sep 2007 12:55:40 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> Sockets need to get a reference to their network namespace,
> or possibly a simple hold if someone registers on the network
> namespace notifier and will free the sockets when the namespace
> is going to be destroyed.

>
> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
> ---
> include/net/inet_timewait_sock.h | 1 +
> include/net/sock.h | 3 +++
> 2 files changed, 4 insertions(+), 0 deletions(-)
>
> diff --git a/include/net/inet_timewait_sock.h b/include/net/inet_timewait_sock.h
> index 47d52b2..abaff05 100644
> --- a/include/net/inet_timewait_sock.h
> +++ b/include/net/inet_timewait_sock.h
> @@ -115,6 +115,7 @@ struct inet_timewait_sock {
> #define tw_refcnt __tw_common.skc_refcnt
> #define tw_hash   __tw_common.skc_hash
> #define tw_prot   __tw_common.skc_prot
> +#define tw_net   __tw_common.skc_net

This place is a very tricky, indeed. If we keep the namespace until
timewait bucket death - we'll keep the namespace alive at least 5
minutes after all process death.

If we stop a VE (in terms of OpenVz) and restart it, we'll 100% have an
OLD namespace with all buckets shown :(So, in OpenVz we use a number
of VE instead of pointer to a VE. Additionally, on VE death we can wipe
all TW buckets. VE start stop from outside world looks very much like a
computer power on/off.

Regards,
 Den

Subject: Re: [PATCH 06/16] net: Add a network namespace parameter to struct sock
Posted by Daniel Lezcano on Thu, 20 Sep 2007 13:20:01 GMT
View Forum Message <> Reply to Message

Denis V. Lunev wrote:
> Eric W. Biederman wrote:
>> Sockets need to get a reference to their network namespace,
>> or possibly a simple hold if someone registers on the network
>> namespace notifier and will free the sockets when the namespace
>> is going to be destroyed.
>>
>> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
>> ---
>> include/net/inet_timewait_sock.h | 1 +
>> include/net/sock.h | 3 +++

>> 2 files changed, 4 insertions(+), 0 deletions(-)
>>
>> diff --git a/include/net/inet_timewait_sock.h b/include/net/inet_timewait_sock.h
>> index 47d52b2..abaff05 100644
>> --- a/include/net/inet_timewait_sock.h
>> +++ b/include/net/inet_timewait_sock.h
>> @@ -115,6 +115,7 @@ struct inet_timewait_sock {
>> #define tw_refcnt __tw_common.skc_refcnt
>> #define tw_hash __tw_common.skc_hash
>> #define tw_prot __tw_common.skc_prot
>> +#define tw_net __tw_common.skc_net
>
>
> This place is a very tricky, indeed. If we keep the namespace until
> timewait bucket death - we'll keep the namespace alive at least 5
> _minutes_ after all process death.

Yes, that's right. And for me that makes totally sense. The namespace
should not be destroyed until it is referenced somewhere.

> If we stop a VE (in terms of OpenVz) and restart it, we'll 100% have an
> _OLD_ namespace with all buckets shown :(
> So, in OpenVz we use a number
> of VE instead of pointer to a VE. Additionally, on VE death we can wipe
> all TW buckets. VE start stop from outside world looks very much like a
> computer power on/off.

That makes sense too. But if you wipe out the sockets when stopping the
VE where is the problem with the restart ?

Subject: Re: [PATCH 06/16] net: Add a network namespace parameter to struct sock
Posted by den on Fri, 21 Sep 2007 05:04:48 GMT
View Forum Message <> Reply to Message

Daniel Lezcano wrote:
>> This place is a very tricky, indeed. If we keep the namespace until
>> timewait bucket death - we'll keep the namespace alive at least 5
>> _minutes_ after all process death.
>
> Yes, that's right. And for me that makes totally sense. The namespace
> should not be destroyed until it is referenced somewhere.

If all incoming interfaces are stopped, sure they do, no incoming
packets will be. So, it is completely pointless to keep TW bucket for 5
minutes. This is a resources wastage.

>> If we stop a VE (in terms of OpenVz) and restart it, we'll 100% have an
>> _OLD_ namespace with all buckets shown :(So, in OpenVz we use a number
>> of VE instead of pointer to a VE. Additionally, on VE death we can wipe
>> all TW buckets. VE start stop from outside world looks very much like a
>> computer power on/off.
>
> That makes sense too. But if you wipe out the sockets when stopping the
> VE where is the problem with the restart ?
>
>

classical egg/chicken problem. If TW bucket holds namespace, how to
decide when to destroy it? :(

Subject: Re: [PATCH 06/16] net: Add a network namespace parameter to struct sock
Posted by ebiederm on Fri, 21 Sep 2007 05:58:56 GMT
View Forum Message <> Reply to Message

"Denis V. Lunev" <den@sw.ru> writes:

> Daniel Lezcano wrote:
>>> This place is a very tricky, indeed. If we keep the namespace until
>>> timewait bucket death - we'll keep the namespace alive at least 5
>>> _minutes_ after all process death.
>>
>> Yes, that's right. And for me that makes totally sense. The namespace
>> should not be destroyed until it is referenced somewhere.
>
> If all incoming interfaces are stopped, sure they do, no incoming
> packets will be. So, it is completely pointless to keep TW bucket for 5
> minutes. This is a resources wastage.

Agreed, at least in principle.

>>> If we stop a VE (in terms of OpenVz) and restart it, we'll 100% have an
>>> _OLD_ namespace with all buckets shown :(So, in OpenVz we use a number
>>> of VE instead of pointer to a VE. Additionally, on VE death we can wipe
>>> all TW buckets. VE start stop from outside world looks very much like a
>>> computer power on/off.
>>
>> That makes sense too. But if you wipe out the sockets when stopping the
>> VE where is the problem with the restart ?
>>
>>
>
> classical egg/chicken problem. If TW bucket holds namespace, how to

> decide when to destroy it? :(

TW bucket must have a reference to a namespace because otherwise
we cannot interpret them.

However if need be we can just do hold_net, release_net style reference
counting, if we know that when the namespace exits we will flush all
of those sockets.

I looked and it doesn't appear that I am actually initializing
this field in my current patchset. :(
- So either my skim through my code is wrong.
- Something got dropped in keeping the patches up to date.
- This was never addressed :(

I would be a good idea to see if we can make certain that we are
initializing the field right now (at least to &init_net). That
way we won't get into a subtle problem later when we try and use it.

Eric

Subject: Re: [PATCH 06/16] net: Add a network namespace parameter to struct sock
Posted by Daniel Lezcano on Fri, 21 Sep 2007 07:30:22 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> "Denis V. Lunev" <den@sw.ru> writes:
>
>> Daniel Lezcano wrote:
>>>> This place is a very tricky, indeed. If we keep the namespace until
>>>> timewait bucket death - we'll keep the namespace alive at least 5
>>>> _minutes_ after all process death.
>>> Yes, that's right. And for me that makes totally sense. The namespace
>>> should not be destroyed until it is referenced somewhere.
>> If all incoming interfaces are stopped, sure they do, no incoming
>> packets will be. So, it is completely pointless to keep TW bucket for 5
>> minutes. This is a resources wastage.
>
> Agreed, at least in principle.
>>>> If we stop a VE (in terms of OpenVz) and restart it, we'll 100% have an
>>>> _OLD_ namespace with all buckets shown :(So, in OpenVz we use a number
>>>> of VE instead of pointer to a VE. Additionally, on VE death we can wipe
>>>> all TW buckets. VE start stop from outside world looks very much like a
>>>> computer power on/off.
>>> That makes sense too. But if you wipe out the sockets when stopping the
>>> VE where is the problem with the restart ?

>>>
>>>
>> classical egg/chicken problem. If TW bucket holds namespace, how to
>> decide when to destroy it? :(
>
> TW bucket must have a reference to a namespace because otherwise
> we cannot interpret them.
>
> However if need be we can just do hold_net, release_net style reference
> counting, if we know that when the namespace exits we will flush all
> of those sockets.
>
> I looked and it doesn't appear that I am actually initializing
> this field in my current patchset. :(
> - So either my skim through my code is wrong.
> - Something got dropped in keeping the patches up to date.
> - This was never addressed :(
> I would be a good idea to see if we can make certain that we are
> initializing the field right now (at least to &init_net). That
> way we won't get into a subtle problem later when we try and use it.

With Denis's remark I looked at the code and I noticed that too.
I am currently doing some testing to check that. I will provide a
patchset to hold a network namespace reference for the timewait socket
and to wipe out timewait socket for the network namespace in a few hours.

BTW, the orphan sockets will lead to a similar problem ...

-- Daniel