Subject: [PATCH 0/12] L2 network namespace (v3)
Posted by Mishin Dmitry on Wed, 17 Jan 2007 15:51:14 GMT
View Forum Message <> Reply to Message

This is an update of L2 network namespaces patches. They are applicable
to Cedric's 2.6.20-rc4-mm1-lxc2 tree.

Changes:
 - updated to 2.6.20-rc4-mm1-lxc2
 - current network context is per-CPU now
 - fixed compilation without CONFIG_NET_NS

Changed current context definition should fix all mentioned by Cedric issues:
 - the nsproxy backpointer is unnecessary now - thus removed;
 - the push_net_ns() and pop_net_ns() use per-CPU variable now;
 - there is no race on ->nsproxy between push_net_ns() and
   exit_task_namespaces() because they deals with differrent pointers.

==========
L2 network namespaces

The most straightforward concept of network virtualization is complete
separation of namespaces, covering device list, routing tables, netfilter
tables, socket hashes, and everything else.

On input path, each packet is tagged with namespace right from the
place where it appears from a device, and is processed by each layer
in the context of this namespace.
Non-root namespaces communicate with the outside world in two ways: by
owning hardware devices, or receiving packets forwarded them by their parent
namespace via pass-through device.

This complete separation of namespaces is very useful for at least two
purposes:
  - allowing users to create and manage by their own various tunnels and
    VPNs, and
  - enabling easier and more straightforward live migration of groups of
    processes with their environment.

--
Thanks,
Dmitry.

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

Subject: [PATCH 1/12] L2 network namespace (v3): current network namespace operations
Posted by Mishin Dmitry on Wed, 17 Jan 2007 15:57:40 GMT
View Forum Message <> Reply to Message

Added functions and macros required to operate with network namespaces.
They are required in order to switch network namespace for incoming packets and
to not extend current network interface by additional network namespace argue.

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

---
include/linux/net_namespace.h | 49 ++--
kernel/fork.c | 1
kernel/sched.c | 5 ++++
net/core/net_namespace.c | 3 ++
4 files changed, 56 insertions(+), 2 deletions(-)

```
--- linux-2.6.20-rc4-mm1.net_ns.orig/include/linux/net_namespace.h
+++ linux-2.6.20-rc4-mm1.net_ns/include/linux/net_namespace.h
@@ -33,7 +33,34 @@ static inline void put_net_ns(struct net
  kref_put(&ns->kref, free_net_ns);
 }

-#else
+DECLARE_PER_CPU(struct net_namespace *, exec_net_ns);
+#define current_net_ns  (__get_cpu_var(exec_net_ns))
+
+static inline void init_current_net_ns(int cpu)
+{
+ get_net_ns(&init_net_ns);
+ per_cpu(exec_net_ns, cpu) = &init_net_ns;
+}
+
+static inline struct net_namespace *push_net_ns(struct net_namespace *to)
+{
+ struct net_namespace *orig;
+
+ orig = current_net_ns;
+ get_net_ns(to);
+ current_net_ns = to;
+ put_net_ns(orig);
+ return orig;
+}
+
+static inline void pop_net_ns(struct net_namespace *to)
+{
+ (void)push_net_ns(to);
+}
```

```
+
+#define net_ns_match(target, ns) ((target) == (ns))
+
+#else /* CONFIG_NET_NS */

 #define INIT_NET_NS(net_ns)

@@ -58,6 +85,24 @@ static inline int copy_net_ns(int flags,
 static inline void put_net_ns(struct net_namespace *ns)
 {
 }
-#endif
+
+#define current_net_ns   NULL
+
+static inline void init_current_net_ns(int cpu)
+{
+}
+
+static inline struct net_namespace *push_net_ns(struct net_namespace *ns)
+{
+ return NULL;
+}
+
+static inline void pop_net_ns(struct net_namespace *ns)
+{
+}
+
+#define net_ns_match(target, ns) ((void)(ns), 1)
+
+#endif /* !CONFIG_NET_NS */

 #endif /* _LINUX_NET_NAMESPACE_H */
--- linux-2.6.20-rc4-mm1.net_ns.orig/kernel/fork.c
+++ linux-2.6.20-rc4-mm1.net_ns/kernel/fork.c
@@ -1719,6 +1719,7 @@ asmlinkage long sys_unshare(unsigned lon
   if (new_net) {
    net = current->nsproxy->net_ns;
    current->nsproxy->net_ns = new_net;
+   pop_net_ns(new_net);
    new_net = net;
   }

--- linux-2.6.20-rc4-mm1.net_ns.orig/kernel/sched.c
+++ linux-2.6.20-rc4-mm1.net_ns/kernel/sched.c
@@ -53,6 +53,7 @@
 #include <linux/tsacct_kern.h>
 #include <linux/kprobes.h>
```

```
 #include <linux/delayacct.h>
+#include <linux/net_namespace.h>
 #include <asm/tlb.h>

 #include <asm/unistd.h>
@@ -1824,6 +1825,9 @@ static inline void finish_task_switch(st
   kprobe_flush_task(prev);
   put_task_struct(prev);
  }
+
+ (void)push_net_ns(current->nsproxy->net_ns);
+
 }

 /**
@@ -7066,6 +7070,7 @@ void __init sched_init(void)
    // delimiter for bitsearch
    __set_bit(MAX_PRIO, array->bitmap);
   }
+  init_current_net_ns(i);
  }

  set_load_weight(&init_task);
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/core/net_namespace.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/core/net_namespace.c
@@ -18,6 +18,9 @@ struct net_namespace init_net_ns = {

 #ifdef CONFIG_NET_NS

+DEFINE_PER_CPU(struct net_namespace *, exec_net_ns);
+EXPORT_PER_CPU_SYMBOL_GPL(exec_net_ns);
+
 /*
  * Clone a new ns copying an original net ns, setting refcount to 1
  * @old_ns: namespace to clone
```


Subject: [PATCH 2/12] L2 network namespace (v3): network devices virtualization
Posted by Mishin Dmitry on Wed, 17 Jan 2007 15:59:17 GMT
View Forum Message <> Reply to Message

Added ability to have per-namespace network devices.

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

```
---
 include/linux/net_namespace.h |    8 +++-
 include/linux/netdevice.h     |    8 ++++
 net/core/dev.c                |   77 +++++++++++++++++++++++++++++++++++++-------
 net/core/net-sysfs.c          |   23 ++++++++++++
 net/core/net_namespace.c      |   11 ++++++
 5 files changed, 114 insertions(+), 13 deletions(-)

--- linux-2.6.20-rc4-mm1.net_ns.orig/include/linux/net_namespace.h
+++ linux-2.6.20-rc4-mm1.net_ns/include/linux/net_namespace.h
@@ -7,7 +7,9 @@
 #include <linux/errno.h>

 struct net_namespace {
- struct kref kref;
+ struct kref  kref;
+ struct net_device *dev_base_p, **dev_tail_p;
+ unsigned int  hash;
 };

 extern struct net_namespace init_net_ns;
@@ -60,6 +62,8 @@ static inline void pop_net_ns(struct net

 #define net_ns_match(target, ns) ((target) == (ns))

+#define net_ns_hash(ns)   ((ns)->hash)
+
 #else /* CONFIG_NET_NS */

 #define INIT_NET_NS(net_ns)
@@ -103,6 +107,8 @@ static inline void pop_net_ns(struct net

 #define net_ns_match(target, ns) ((void)(ns), 1)

+#define net_ns_hash(ns)   (0)
+
 #endif /* !CONFIG_NET_NS */

 #endif /* _LINUX_NET_NAMESPACE_H */
--- linux-2.6.20-rc4-mm1.net_ns.orig/include/linux/netdevice.h
+++ linux-2.6.20-rc4-mm1.net_ns/include/linux/netdevice.h
@@ -389,6 +389,7 @@ struct net_device
  int   promiscuity;
  int   allmulti;

+ struct net_namespace *net_ns;
```

```
  /* Protocol specific pointers */

@@ -567,9 +568,16 @@ struct packet_type {

 #include <linux/interrupt.h>
 #include <linux/notifier.h>
+#include <linux/net_namespace.h>

 extern struct net_device  loopback_dev;  /* The loopback */
+#ifndef CONFIG_NET_NS
 extern struct net_device  *dev_base;  /* All devices */
+#define dev_base_ns(dev)  dev_base
+#else
+#define dev_base   (current_net_ns->dev_base_p)
+#define dev_base_ns(dev)  (dev->net_ns->dev_base_p)
+#endif
 extern rwlock_t    dev_base_lock;  /* Device list lock */

 extern int    netdev_boot_setup_check(struct net_device *dev);
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/core/dev.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/core/dev.c
@@ -90,6 +90,7 @@
 #include <linux/if_ether.h>
 #include <linux/netdevice.h>
 #include <linux/etherdevice.h>
+#include <linux/net_namespace.h>
 #include <linux/notifier.h>
 #include <linux/skbuff.h>
 #include <net/sock.h>
@@ -174,20 +175,27 @@ static spinlock_t net_dma_event_lock;
  * unregister_netdevice(), which must be called with the rtnl
  * semaphore held.
  */
+#ifndef CONFIG_NET_NS
 struct net_device *dev_base;
 static struct net_device **dev_tail = &dev_base;
-DEFINE_RWLOCK(dev_base_lock);
-
+#define dev_tail_ns(dev) dev_tail
 EXPORT_SYMBOL(dev_base);
+#else
+#define dev_tail_ns(dev) (dev->net_ns->dev_tail_p)
+#endif
+
+DEFINE_RWLOCK(dev_base_lock);
 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];

-static inline struct hlist_head *dev_name_hash(const char *name)
+static inline struct hlist_head *dev_name_hash(const char *name,
+       struct net_namespace *ns)
 {
  unsigned hash = full_name_hash(name, strnlen(name, IFNAMSIZ));
+ hash ^= net_ns_hash(ns);
  return &dev_name_head[hash & ((1<<NETDEV_HASHBITS)-1)];
 }

@@ -212,10 +220,12 @@ DEFINE_PER_CPU(struct softnet_data, soft
 extern int netdev_sysfs_init(void);
 extern int netdev_register_sysfs(struct net_device *);
 extern void netdev_unregister_sysfs(struct net_device *);
+extern int netdev_rename_sysfs(struct net_device *);
 #else
 #define netdev_sysfs_init()   (0)
 #define netdev_register_sysfs(dev) (0)
 #define netdev_unregister_sysfs(dev) do { } while(0)
+#define netdev_rename_sysfs(dev) (0)
 #endif


@@ -475,10 +485,13 @@ __setup("netdev=", netdev_boot_setup);
 struct net_device *__dev_get_by_name(const char *name)
 {
  struct hlist_node *p;
+ struct net_namespace *ns = current_net_ns;

- hlist_for_each(p, dev_name_hash(name)) {
+ hlist_for_each(p, dev_name_hash(name, ns)) {
   struct net_device *dev
    = hlist_entry(p, struct net_device, name_hlist);
+  if (!net_ns_match(dev->net_ns, ns))
+   continue;
   if (!strncmp(dev->name, name, IFNAMSIZ))
    return dev;
  }
@@ -751,10 +764,11 @@ int dev_change_name(struct net_device *d
  else
   strlcpy(dev->name, newname, IFNAMSIZ);

- err = device_rename(&dev->dev, dev->name);
+ err = netdev_rename_sysfs(dev);
  if (!err) {
   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(dev->name,
+       current_net_ns));
   raw_notifier_call_chain(&netdev_chain,
     NETDEV_CHANGENAME, dev);
  }
@@ -1481,8 +1495,11 @@ gso:
   spin_lock(&dev->queue_lock);
   q = dev->qdisc;
   if (q->enqueue) {
+   struct net_namespace *orig_net_ns;
+   orig_net_ns = push_net_ns(dev->net_ns);
    rc = q->enqueue(skb, q);
    qdisc_run(dev);
+   pop_net_ns(orig_net_ns);
    spin_unlock(&dev->queue_lock);

    rc = rc == NET_XMIT_BYPASS ? NET_XMIT_SUCCESS : rc;
@@ -1678,7 +1695,10 @@ static void net_tx_action(struct softirq
    clear_bit(__LINK_STATE_SCHED, &dev->state);

    if (spin_trylock(&dev->queue_lock)) {
+    struct net_namespace *orig_net_ns;
+    orig_net_ns = push_net_ns(dev->net_ns);
     qdisc_run(dev);
+    pop_net_ns(orig_net_ns);
     spin_unlock(&dev->queue_lock);
    } else {
     netif_schedule(dev);
@@ -1765,6 +1785,7 @@ int netif_receive_skb(struct sk_buff *sk
 {
  struct packet_type *ptype, *pt_prev;
  struct net_device *orig_dev;
+ struct net_namespace *orig_net_ns;
  int ret = NET_RX_DROP;
  __be16 type;

@@ -1783,6 +1804,8 @@ int netif_receive_skb(struct sk_buff *sk
  if (!orig_dev)
   return NET_RX_DROP;

+ orig_net_ns = push_net_ns(skb->dev->net_ns);
+
  __get_cpu_var(netdev_rx_stat).total++;

  skb->h.raw = skb->nh.raw = skb->data;
@@ -1851,6 +1874,7 @@ ncls:
```

```
 out:
  rcu_read_unlock();
+ pop_net_ns(orig_net_ns);
  return ret;
 }

@@ -2878,6 +2902,7 @@ int register_netdevice(struct net_device
 {
  struct hlist_head *head;
  struct hlist_node *p;
+ struct net_namespace *ns;
  int ret;

  BUG_ON(dev_boot_phase);
@@ -2895,6 +2920,12 @@ int register_netdevice(struct net_device
  spin_lock_init(&dev->ingress_lock);
 #endif

+ ns = NULL;
+#ifdef CONFIG_NET_NS
+ BUG_ON(!dev->net_ns);
+ ns = dev->net_ns;
+#endif
+
  dev->iflink = -1;

  /* Init, if this function is available */
@@ -2917,10 +2948,12 @@ int register_netdevice(struct net_device
   dev->iflink = dev->ifindex;

  /* Check for existence of name */
- head = dev_name_hash(dev->name);
+ head = dev_name_hash(dev->name, ns);
  hlist_for_each(p, head) {
   struct net_device *d
    = hlist_entry(p, struct net_device, name_hlist);
+  if (!net_ns_match(d->net_ns, ns))
+   continue;
   if (!strncmp(d->name, dev->name, IFNAMSIZ)) {
    ret = -EEXIST;
     goto out;
@@ -2980,8 +3013,8 @@ int register_netdevice(struct net_device
  dev->next = NULL;
  dev_init_scheduler(dev);
  write_lock_bh(&dev_base_lock);
- *dev_tail = dev;
- dev_tail = &dev->next;
+ *dev_tail_ns(dev) = dev;
```

```
+ dev_tail_ns(dev) = &dev->next;
  hlist_add_head(&dev->name_hlist, head);
  hlist_add_head(&dev->index_hlist, dev_index_hash(dev->ifindex));
  dev_hold(dev);
@@ -3195,6 +3228,10 @@ struct net_device *alloc_netdev(int size
  if (sizeof_priv)
   dev->priv = netdev_priv(dev);

+#ifdef CONFIG_NET_NS
+ get_net_ns(current_net_ns);
+ dev->net_ns = current_net_ns;
+#endif
  setup(dev);
  strcpy(dev->name, name);
  return dev;
@@ -3211,6 +3248,15 @@ EXPORT_SYMBOL(alloc_netdev);
  */
 void free_netdev(struct net_device *dev)
 {
+#ifdef CONFIG_NET_NS
+ struct net_namespace *ns;
+
+ ns = dev->net_ns;
+ if (ns != NULL) {
+  put_net_ns(ns);
+  dev->net_ns = NULL;
+ }
+#endif
 #ifdef CONFIG_SYSFS
  /*  Compatibility with error handling in drivers */
  if (dev->reg_state == NETREG_UNINITIALIZED) {
@@ -3221,6 +3267,13 @@ void free_netdev(struct net_device *dev)
  BUG_ON(dev->reg_state != NETREG_UNREGISTERED);
  dev->reg_state = NETREG_RELEASED;

+#ifdef CONFIG_NET_NS
+ if (ns != NULL && ns != &init_net_ns) {
+  kfree((char *)dev - dev->padded);
+  return;
+ }
+#endif
+
  /* will free via device release */
  put_device(&dev->dev);
 #else
@@ -3268,13 +3321,13 @@ int unregister_netdevice(struct net_devi
   dev_close(dev);
```

```
  /* And unlink it from device chain. */
- for (dp = &dev_base; (d = *dp) != NULL; dp = &d->next) {
+ for (dp = &dev_base_ns(dev); (d = *dp) != NULL; dp = &d->next) {
   if (d == dev) {
    write_lock_bh(&dev_base_lock);
    hlist_del(&dev->name_hlist);
    hlist_del(&dev->index_hlist);
-   if (dev_tail == &dev->next)
-    dev_tail = dp;
+   if (dev_tail_ns(dev) == &dev->next)
+    dev_tail_ns(dev) = dp;
    *dp = d->next;
    write_unlock_bh(&dev_base_lock);
    break;
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/core/net-sysfs.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/core/net-sysfs.c
@@ -453,6 +453,12 @@ static struct class net_class = {

 void netdev_unregister_sysfs(struct net_device * net)
 {
+#ifdef CONFIG_NET_NS
+ if (net->net_ns != &init_net_ns)
+  /* not supported yet: sysfs virtualization is required */
+  return;
+#endif
+
  device_del(&(net->dev));
 }

@@ -462,6 +468,12 @@ int netdev_register_sysfs(struct net_dev
  struct device *dev = &(net->dev);
  struct attribute_group **groups = net->sysfs_groups;

+#ifdef CONFIG_NET_NS
+ if (net->net_ns != &init_net_ns)
+  /* not supported yet: sysfs virtualization is required */
+  return 0;
+#endif
+
  device_initialize(dev);
  dev->class = &net_class;
  dev->platform_data = net;
@@ -481,6 +493,17 @@ int netdev_register_sysfs(struct net_dev
  return device_add(dev);
 }

+int netdev_rename_sysfs(struct net_device *net)
+{
```

```
+#ifdef CONFIG_NET_NS
+ if (net->net_ns != &init_net_ns)
+  /* not supported yet: sysfs virtualization is required */
+  return 0;
+#endif
+
+ return device_rename(&net->dev, net->name);
+}
+
 int netdev_sysfs_init(void)
 {
  return class_register(&net_class);
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/core/net_namespace.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/core/net_namespace.c
@@ -9,11 +9,14 @@
 #include <linux/version.h>
 #include <linux/nsproxy.h>
 #include <linux/net_namespace.h>
+#include <linux/net.h>

 struct net_namespace init_net_ns = {
  .kref = {
   .refcount = ATOMIC_INIT(2),
  },
+ .dev_base_p = NULL,
+ .dev_tail_p = &init_net_ns.dev_base_p,
 };

 #ifdef CONFIG_NET_NS
@@ -35,6 +38,9 @@ static struct net_namespace *clone_net_n
   return NULL;

  kref_init(&ns->kref);
+ ns->dev_base_p = NULL;
+ ns->dev_tail_p = &ns->dev_base_p;
+ ns->hash = net_random();
  return ns;
 }

@@ -73,6 +79,11 @@ void free_net_ns(struct kref *kref)
  struct net_namespace *ns;

  ns = container_of(kref, struct net_namespace, kref);
+ if (ns->dev_base_p != NULL) {
+  printk("NET_NS: BUG: namespace %p has devices! ref %d\n",
+    ns, atomic_read(&ns->kref.refcount));
+  return;
+ }
```

```
 kfree(ns);
}
```

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

Subject: [PATCH 3/12] L2 network namespace (v3): loopback device virtualization
Posted by Mishin Dmitry on Wed, 17 Jan 2007 16:00:32 GMT
View Forum Message <> Reply to Message

Added per-namespace network loopback device

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

```
---
 drivers/net/loopback.c        |  112 ++++++++++++++++++++++++++++++++-------------
 include/linux/net_namespace.h |    2
 include/linux/netdevice.h     |    6 +-
 net/core/dev.c                |    3 -
 net/core/net_namespace.c      |   16 ++++++
 net/ipv4/devinet.c            |    2
 net/ipv6/addrconf.c           |    2
 net/ipv6/route.c              |    6 +-
 8 files changed, 108 insertions(+), 41 deletions(-)

--- linux-2.6.20-rc4-mm1.net_ns.orig/drivers/net/loopback.c
+++ linux-2.6.20-rc4-mm1.net_ns/drivers/net/loopback.c
@@ -62,7 +62,12 @@ struct pcpu_lstats {
  unsigned long packets;
  unsigned long bytes;
 };
-static DEFINE_PER_CPU(struct pcpu_lstats, pcpu_lstats);
+#ifndef CONFIG_NET_NS
+struct pcpu_lstats  *pcpu_lstats_p;
+#define pcpu_lstats_ptr(ns)  pcpu_lstats_p
+#else
+#define pcpu_lstats_ptr(ns)  (ns->pcpu_lstats_p)
+#endif

 #define LOOPBACK_OVERHEAD (128 + MAX_HEADER + 16 + 16)

@@ -154,7 +159,8 @@ static int loopback_xmit(struct sk_buff
  dev->last_rx = jiffies;

  /* it's OK to use __get_cpu_var() because BHs are off */
```

```
- lb_stats = &__get_cpu_var(pcpu_lstats);
+ lb_stats = per_cpu_ptr(pcpu_lstats_ptr(dev->net_ns),
+       smp_processor_id());
  lb_stats->bytes += skb->len;
  lb_stats->packets++;

@@ -163,11 +169,9 @@ static int loopback_xmit(struct sk_buff
  return 0;
 }

-static struct net_device_stats loopback_stats;
-
 static struct net_device_stats *get_stats(struct net_device *dev)
 {
- struct net_device_stats *stats = &loopback_stats;
+ struct net_device_stats *stats = netdev_priv(dev);
  unsigned long bytes = 0;
  unsigned long packets = 0;
  int i;
@@ -175,7 +179,7 @@ static struct net_device_stats *get_stat
  for_each_possible_cpu(i) {
   const struct pcpu_lstats *lb_stats;

-  lb_stats = &per_cpu(pcpu_lstats, i);
+  lb_stats = per_cpu_ptr(pcpu_lstats_ptr(dev->net_ns), i);
   bytes   += lb_stats->bytes;
   packets += lb_stats->packets;
  }
@@ -200,40 +204,80 @@ static const struct ethtool_ops loopback
  .get_rx_csum  = always_on,
 };

-/*
- * The loopback device is special. There is only one instance and
- * it is statically allocated. Don't do this for other devices.
- */
-struct net_device loopback_dev = {
- .name    = "lo",
- .get_stats  = &get_stats,
- .priv   = &loopback_stats,
- .mtu   = (16 * 1024) + 20 + 20 + 12,
- .hard_start_xmit = loopback_xmit,
- .hard_header  = eth_header,
- .hard_header_cache = eth_header_cache,
- .header_cache_update = eth_header_cache_update,
- .hard_header_len = ETH_HLEN, /* 14 */
- .addr_len  = ETH_ALEN, /* 6 */
- .tx_queue_len  = 0,
```

```
- .type   = ARPHRD_LOOPBACK, /* 0x0001*/
- .rebuild_header  = eth_rebuild_header,
- .flags   = IFF_LOOPBACK,
- .features   = NETIF_F_SG | NETIF_F_FRAGLIST
+struct net_device *loopback_dev_p;
+EXPORT_SYMBOL(loopback_dev_p);
+
+struct pcpu_lstats *pcpu_lstats_p;
+EXPORT_SYMBOL(pcpu_lstats_p);
+
+
+void loopback_dev_dtor(struct net_device *dev)
+{
+ free_percpu(pcpu_lstats_ptr(dev->net_ns));
+ free_netdev(dev);
+}
+
+void loopback_dev_ctor(struct net_device *dev)
+{
+ dev->mtu  = (16 * 1024) + 20 + 20 + 12;
+ dev->hard_start_xmit = loopback_xmit;
+ dev->hard_header = eth_header;
+ dev->hard_header_cache = eth_header_cache;
+ dev->header_cache_update = eth_header_cache_update;
+ dev->hard_header_len = ETH_HLEN; /* 14 */
+ dev->addr_len  = ETH_ALEN; /* 6 */
+ dev->tx_queue_len = 0;
+ dev->type  = ARPHRD_LOOPBACK; /* 0x0001*/
+ dev->rebuild_header = eth_rebuild_header;
+ dev->flags  = IFF_LOOPBACK;
+ dev->features   = NETIF_F_SG | NETIF_F_FRAGLIST
 #ifdef LOOPBACK_TSO
       | NETIF_F_TSO
 #endif
       | NETIF_F_NO_CSUM | NETIF_F_HIGHDMA
-      | NETIF_F_LLTX,
- .ethtool_ops  = &loopback_ethtool_ops,
-};
+      | NETIF_F_LLTX;
+ dev->ethtool_ops = &loopback_ethtool_ops;
+ dev->get_stats  = &get_stats;
+ dev->destructor  = loopback_dev_dtor;
+}

 /* Setup and register the loopback device. */
-static int __init loopback_init(void)
+int loopback_init(void)
 {
```

```
- return register_netdev(&loopback_dev);
-};
+ struct net_namespace *ns = current_net_ns;
+ int err;

-module_init(loopback_init);
+ err = -ENOMEM;
+ loopback_dev_ptr = alloc_netdev(sizeof(struct net_device_stats), "lo",
+       loopback_dev_ctor);
+ if (loopback_dev_ptr == NULL)
+  goto out;
+ pcpu_lstats_ptr(ns) = alloc_percpu(struct pcpu_lstats);
+ if (pcpu_lstats_ptr(ns) == NULL)
+  goto out_stats;
+ /*
+  * loopback device doesn't hold active reference: it doesn't prevent
+  * stopping of net_namespace. So, just put old namespace.
+  */
+#ifdef CONFIG_NET_NS
+ put_net_ns(loopback_dev_ptr->net_ns);
+ loopback_dev_ptr->net_ns = ns;
+#endif
+ err = register_netdev(loopback_dev_ptr);
+ if (err)
+  goto out_register;
+ return 0;

-EXPORT_SYMBOL(loopback_dev);
+out_register:
+ free_percpu(ns->pcpu_lstats_p);
+#ifdef CONFIG_NET_NS
+ /* in order to avoid put in free_netdev() */
+ loopback_dev_ptr->net_ns = NULL;
+#endif
+out_stats:
+ free_netdev(loopback_dev_ptr);
+out:
+ return err;
+}
+
+module_init(loopback_init);
--- linux-2.6.20-rc4-mm1.net_ns.orig/include/linux/net_namespace.h
+++ linux-2.6.20-rc4-mm1.net_ns/include/linux/net_namespace.h
@@ -9,6 +9,8 @@
 struct net_namespace {
  struct kref  kref;
  struct net_device *dev_base_p, **dev_tail_p;
+ struct net_device *loopback_dev_p;
```

```
+ struct pcpu_lstats *pcpu_lstats_p;
  unsigned int  hash;
 };

--- linux-2.6.20-rc4-mm1.net_ns.orig/include/linux/netdevice.h
+++ linux-2.6.20-rc4-mm1.net_ns/include/linux/netdevice.h
@@ -570,16 +570,20 @@ struct packet_type {
 #include <linux/notifier.h>
 #include <linux/net_namespace.h>

-extern struct net_device  loopback_dev;  /* The loopback */
+extern struct net_device  *loopback_dev_p;
 #ifndef CONFIG_NET_NS
+#define loopback_dev_ptr  loopback_dev_p
 extern struct net_device  *dev_base;  /* All devices */
 #define dev_base_ns(dev)  dev_base
 #else
+#define loopback_dev_ptr  (current_net_ns->loopback_dev_p)
 #define dev_base   (current_net_ns->dev_base_p)
 #define dev_base_ns(dev)  (dev->net_ns->dev_base_p)
 #endif
+#define loopback_dev   (*loopback_dev_ptr) /* The loopback */
 extern rwlock_t    dev_base_lock;  /* Device list lock */

+extern int   loopback_init(void);
 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);
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/core/dev.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/core/dev.c
@@ -3253,7 +3253,8 @@ void free_netdev(struct net_device *dev)

  ns = dev->net_ns;
  if (ns != NULL) {
-  put_net_ns(ns);
+  if (dev != ns->loopback_dev_p)
+   put_net_ns(ns);
   dev->net_ns = NULL;
  }
 #endif
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/core/net_namespace.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/core/net_namespace.c
@@ -10,6 +10,7 @@
 #include <linux/nsproxy.h>
 #include <linux/net_namespace.h>
 #include <linux/net.h>
+#include <linux/netdevice.h>
```

```
 struct net_namespace init_net_ns = {
  .kref = {
@@ -17,6 +18,8 @@ struct net_namespace init_net_ns = {
  },
  .dev_base_p = NULL,
  .dev_tail_p = &init_net_ns.dev_base_p,
+ .loopback_dev_p = NULL,
+ .pcpu_lstats_p = NULL,
 };

 #ifdef CONFIG_NET_NS
@@ -41,7 +44,19 @@ static struct net_namespace *clone_net_n
  ns->dev_base_p = NULL;
  ns->dev_tail_p = &ns->dev_base_p;
  ns->hash = net_random();
+
+ if ((push_net_ns(ns)) != old_ns)
+  BUG();
+ if (loopback_init())
+  goto out_loopback;
+ pop_net_ns(old_ns);
  return ns;
+
+out_loopback:
+ pop_net_ns(old_ns);
+ BUG_ON(atomic_read(&ns->kref.refcount) != 1);
+ kfree(ns);
+ return NULL;
 }

 /*
@@ -79,6 +94,7 @@ void free_net_ns(struct kref *kref)
  struct net_namespace *ns;

  ns = container_of(kref, struct net_namespace, kref);
+ unregister_netdev(ns->loopback_dev_p);
  if (ns->dev_base_p != NULL) {
   printk("NET_NS: BUG: namespace %p has devices! ref %d\n",
     ns, atomic_read(&ns->kref.refcount));
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/ipv4/devinet.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/ipv4/devinet.c
@@ -198,7 +198,7 @@ static void inetdev_destroy(struct in_de
  ASSERT_RTNL();

  dev = in_dev->dev;
- if (dev == &loopback_dev)
+ if (dev == loopback_dev_p)
   return;
```

```
  in_dev->dead = 1;
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/ipv6/addrconf.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/ipv6/addrconf.c
@@ -2360,7 +2360,7 @@ static int addrconf_ifdown(struct net_de

  ASSERT_RTNL();

- if (dev == &loopback_dev && how == 1)
+ if (dev == loopback_dev_p && how == 1)
   how = 0;

  rt6_ifdown(dev);
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/ipv6/route.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/ipv6/route.c
@@ -124,7 +124,6 @@ struct rt6_info ip6_null_entry = {
   .dst = {
    .__refcnt = ATOMIC_INIT(1),
    .__use  = 1,
-   .dev  = &loopback_dev,
    .obsolete = -1,
    .error  = -ENETUNREACH,
    .metrics = { [RTAX_HOPLIMIT - 1] = 255, },
@@ -150,7 +149,6 @@ struct rt6_info ip6_prohibit_entry = {
   .dst = {
    .__refcnt = ATOMIC_INIT(1),
    .__use  = 1,
-   .dev  = &loopback_dev,
    .obsolete = -1,
    .error  = -EACCES,
    .metrics = { [RTAX_HOPLIMIT - 1] = 255, },
@@ -170,7 +168,6 @@ struct rt6_info ip6_blk_hole_entry = {
   .dst = {
    .__refcnt = ATOMIC_INIT(1),
    .__use  = 1,
-   .dev  = &loopback_dev,
    .obsolete = -1,
    .error  = -EINVAL,
    .metrics = { [RTAX_HOPLIMIT - 1] = 255, },
@@ -2469,7 +2466,10 @@ void __init ip6_route_init(void)
 #endif
 #ifdef CONFIG_IPV6_MULTIPLE_TABLES
  fib6_rules_init();
+ ip6_prohibit_entry.u.dst.dev = &loopback_dev;
+ ip6_blk_hole_entry.u.dst.dev = &loopback_dev;
 #endif
+ ip6_null_entry.u.dst.dev = &loopback_dev;
 }
```

void ip6_route_cleanup(void)

_____

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

Subject: [PATCH 4/12] L2 network namespace (v3): devinet sysctl's checks
Posted by Mishin Dmitry on Wed, 17 Jan 2007 16:01:57 GMT
View Forum Message <> Reply to Message

This patch prohibits sysctl's registrations for devices, initialized in non-init
network namespace. This is required due to absense of the sysctl virtualization.

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

---
net/ipv4/devinet.c | 26 +++++++++++++++++++----------
1 file changed, 16 insertions(+), 10 deletions(-)

--- linux-2.6.20-rc4-mm1.net_ns.orig/net/ipv4/devinet.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/ipv4/devinet.c
@@ -160,15 +160,17 @@ struct in_device *inetdev_init(struct ne
 /* Reference in_dev->dev */
 dev_hold(dev);
#ifdef CONFIG_SYSCTL
- neigh_sysctl_register(dev, in_dev->arp_parms, NET_IPV4,
- NET_IPV4_NEIGH, "ipv4", NULL, NULL);
+ if (current_net_ns == &init_net_ns)
+ neigh_sysctl_register(dev, in_dev->arp_parms, NET_IPV4,
+ NET_IPV4_NEIGH, "ipv4", NULL, NULL);
#endif

 /* Account for reference dev->ip_ptr (below) */
 in_dev_hold(in_dev);

#ifdef CONFIG_SYSCTL
- devinet_sysctl_register(in_dev, &in_dev->cnf);
+ if (current_net_ns == &init_net_ns)
+ devinet_sysctl_register(in_dev, &in_dev->cnf);
#endif
 ip_mc_init_dev(in_dev);
 if (dev->flags & IFF_UP)
@@ -211,13 +213,15 @@ static void inetdev_destroy(struct in_de
 }

#ifdef CONFIG_SYSCTL

```
- devinet_sysctl_unregister(&in_dev->cnf);
+ if (current_net_ns == &init_net_ns)
+  devinet_sysctl_unregister(&in_dev->cnf);
 #endif

  dev->ip_ptr = NULL;

 #ifdef CONFIG_SYSCTL
- neigh_sysctl_unregister(in_dev->arp_parms);
+ if (current_net_ns == &init_net_ns)
+  neigh_sysctl_unregister(in_dev->arp_parms);
 #endif
  neigh_parms_release(&arp_tbl, in_dev->arp_parms);
  arp_ifdown(dev);
@@ -1105,11 +1109,13 @@ static int inetdev_event(struct notifier
   inetdev_changename(dev, in_dev);

 #ifdef CONFIG_SYSCTL
-  devinet_sysctl_unregister(&in_dev->cnf);
-  neigh_sysctl_unregister(in_dev->arp_parms);
-  neigh_sysctl_register(dev, in_dev->arp_parms, NET_IPV4,
-        NET_IPV4_NEIGH, "ipv4", NULL, NULL);
-  devinet_sysctl_register(in_dev, &in_dev->cnf);
+  if (current_net_ns == &init_net_ns) {
+   devinet_sysctl_unregister(&in_dev->cnf);
+   neigh_sysctl_unregister(in_dev->arp_parms);
+   neigh_sysctl_register(dev, in_dev->arp_parms, NET_IPV4,
+         NET_IPV4_NEIGH, "ipv4", NULL, NULL);
+   devinet_sysctl_register(in_dev, &in_dev->cnf);
+  }
 #endif
   break;
  }
```

_____

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

Subject: [PATCH 5/12] L2 network namespace (v3): IPv4 routing
Posted by Mishin Dmitry on Wed, 17 Jan 2007 16:03:35 GMT
View Forum Message <> Reply to Message

Make FIBs per-namespace and adds additional key (net namespace) to lookups in
routing cache.

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

```
---
 include/linux/net_namespace.h |   12 +++
 include/net/flow.h            |    3
 include/net/ip_fib.h          |   46 ++++++++++---
 net/core/fib_rules.c          |   42 ++++++++++---
 net/core/net_namespace.c      |   11 +++
 net/ipv4/fib_frontend.c       |  135 ++++++++++++++++++++++++++++++++++++++--------
 net/ipv4/fib_hash.c           |   10 +--
 net/ipv4/fib_rules.c          |   86 ++++++++++++++++++++++++----
 net/ipv4/fib_semantics.c      |  100 ++++++++++++++++++++++++++++-----
 net/ipv4/fib_trie.c           |   18 ++++-
 net/ipv4/route.c              |   32 +++++++++
 11 files changed, 396 insertions(+), 99 deletions(-)

--- linux-2.6.20-rc4-mm1.net_ns.orig/include/linux/net_namespace.h
+++ linux-2.6.20-rc4-mm1.net_ns/include/linux/net_namespace.h
@@ -11,6 +11,18 @@ struct net_namespace {
  struct net_device *dev_base_p, **dev_tail_p;
  struct net_device *loopback_dev_p;
  struct pcpu_lstats *pcpu_lstats_p;
+#ifndef CONFIG_IP_MULTIPLE_TABLES
+ struct fib_table *fib4_local_table, *fib4_main_table;
+#else
+ struct list_head fib_rules_ops_list;
+ struct fib_rules_ops *fib4_rules_ops;
+#endif
+ struct hlist_head *fib4_tables;
+ struct hlist_head *fib4_hash, *fib4_laddrhash;
+ unsigned  fib4_hash_size, fib4_info_cnt;
+#ifdef CONFIG_IP_FIB_TRIE
+ int   fib4_trie_last_dflt;
+#endif
  unsigned int  hash;
 };

--- linux-2.6.20-rc4-mm1.net_ns.orig/include/net/flow.h
+++ linux-2.6.20-rc4-mm1.net_ns/include/net/flow.h
@@ -82,6 +82,9 @@ struct flowi {
 #define fl_mh_type uli_u.mht.type
 #endif
  __u32           secid; /* used by xfrm; see secid.txt */
+#ifdef CONFIG_NET_NS
+ struct net_namespace *net_ns;
+#endif
 } __attribute__((__aligned__(BITS_PER_LONG/8)));

 #define FLOW_DIR_IN 0
--- linux-2.6.20-rc4-mm1.net_ns.orig/include/net/ip_fib.h
```

```
+++ linux-2.6.20-rc4-mm1.net_ns/include/net/ip_fib.h
@@ -18,6 +18,7 @@

 #include <net/flow.h>
 #include <linux/seq_file.h>
+#include <linux/net_namespace.h>
 #include <net/fib_rules.h>

 struct fib_config {
@@ -171,14 +172,21 @@ struct fib_table {

 #ifndef CONFIG_IP_MULTIPLE_TABLES

-extern struct fib_table *ip_fib_local_table;
-extern struct fib_table *ip_fib_main_table;
+#ifndef CONFIG_NET_NS
+extern struct fib_table *ip_fib_local_table_static;
+extern struct fib_table *ip_fib_main_table_static;
+#define ip_fib_local_table_ns()  ip_fib_local_table_static
+#define ip_fib_main_table_ns()  ip_fib_main_table_static
+#else
+#define ip_fib_local_table_ns()  (current_net_ns->fib4_local_table)
+#define ip_fib_main_table_ns()  (current_net_ns->fib4_main_table)
+#endif

 static inline struct fib_table *fib_get_table(u32 id)
 {
  if (id != RT_TABLE_LOCAL)
-  return ip_fib_main_table;
- return ip_fib_local_table;
+  return ip_fib_main_table_ns();
+ return ip_fib_local_table_ns();
 }

 static inline struct fib_table *fib_new_table(u32 id)
@@ -188,21 +196,29 @@ static inline struct fib_table *fib_new_

 static inline int fib_lookup(const struct flowi *flp, struct fib_result *res)
 {
- if (ip_fib_local_table->tb_lookup(ip_fib_local_table, flp, res) &&
-     ip_fib_main_table->tb_lookup(ip_fib_main_table, flp, res))
+ struct fib_table *tb;
+
+ tb = ip_fib_local_table_ns();
+ if (!tb->tb_lookup(tb, flp, res))
+  return 0;
+ tb = ip_fib_main_table_ns();
+ if (tb->tb_lookup(tb, flp, res))
```

```
   return -ENETUNREACH;
  return 0;
 }

 static inline void fib_select_default(const struct flowi *flp, struct fib_result *res)
 {
+ struct fib_table *tb;
+
+ tb = ip_fib_main_table_ns();
  if (FIB_RES_GW(*res) && FIB_RES_NH(*res).nh_scope == RT_SCOPE_LINK)
-  ip_fib_main_table->tb_select_default(ip_fib_main_table, flp, res);
+  tb->tb_select_default(ip_fib_main_table_ns(), flp, res);
 }

 #else /* CONFIG_IP_MULTIPLE_TABLES */
-#define ip_fib_local_table fib_get_table(RT_TABLE_LOCAL)
-#define ip_fib_main_table fib_get_table(RT_TABLE_MAIN)
+#define ip_fib_local_table_ns() fib_get_table(RT_TABLE_LOCAL)
+#define ip_fib_main_table_ns() fib_get_table(RT_TABLE_MAIN)

 extern int fib_lookup(struct flowi *flp, struct fib_result *res);

@@ -215,6 +231,10 @@ extern void fib_select_default(const str
 /* Exported by fib_frontend.c */
 extern struct nla_policy rtm_ipv4_policy[];
 extern void  ip_fib_init(void);
+#ifdef CONFIG_NET_NS
+extern int ip_fib_struct_init(void);
+extern void ip_fib_struct_cleanup(struct net_namespace *);
+#endif
 extern int inet_rtm_delroute(struct sk_buff *skb, struct nlmsghdr* nlh, void *arg);
 extern int inet_rtm_newroute(struct sk_buff *skb, struct nlmsghdr* nlh, void *arg);
 extern int inet_rtm_getroute(struct sk_buff *skb, struct nlmsghdr* nlh, void *arg);
@@ -230,6 +250,9 @@ extern int ip_fib_check_default(__be32 g
 extern int fib_sync_down(__be32 local, struct net_device *dev, int force);
 extern int fib_sync_up(struct net_device *dev);
 extern __be32  __fib_res_prefsrc(struct fib_result *res);
+#ifdef CONFIG_NET_NS
+extern void fib_hashtable_destroy(struct net_namespace *);
+#endif

 /* Exported by fib_hash.c */
 extern struct fib_table *fib_hash_init(u32 id);
@@ -237,7 +260,10 @@ extern struct fib_table *fib_hash_init(u
 #ifdef CONFIG_IP_MULTIPLE_TABLES
 extern int fib4_rules_dump(struct sk_buff *skb, struct netlink_callback *cb);

-extern void __init fib4_rules_init(void);
```

```
+#ifdef CONFIG_NET_NS
+extern void fib4_rules_cleanup(struct net_namespace *);
+#endif
+extern int fib4_rules_init(void);

 #ifdef CONFIG_NET_CLS_ROUTE
 extern u32 fib_rules_tclass(struct fib_result *res);
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/core/fib_rules.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/core/fib_rules.c
@@ -13,7 +13,12 @@
 #include <linux/list.h>
 #include <net/fib_rules.h>

-static LIST_HEAD(rules_ops);
+#ifndef CONFIG_NET_NS
+static struct list_head rules_ops_static;
+#define rules_ops_ns()  rules_ops_static
+#else
+#define rules_ops_ns()  (current_net_ns->fib_rules_ops_list)
+#endif
 static DEFINE_SPINLOCK(rules_mod_lock);

 static void notify_rule_change(int event, struct fib_rule *rule,
@@ -22,10 +27,12 @@ static void notify_rule_change(int event

 static struct fib_rules_ops *lookup_rules_ops(int family)
 {
+ struct list_head *ops_list;
  struct fib_rules_ops *ops;

+ ops_list = &rules_ops_ns();
  rcu_read_lock();
- list_for_each_entry_rcu(ops, &rules_ops, list) {
+ list_for_each_entry_rcu(ops, ops_list, list) {
   if (ops->family == family) {
    if (!try_module_get(ops->owner))
     ops = NULL;
@@ -47,6 +54,7 @@ static void rules_ops_put(struct fib_rul
 int fib_rules_register(struct fib_rules_ops *ops)
 {
  int err = -EEXIST;
+ struct list_head *ops_list;
  struct fib_rules_ops *o;

  if (ops->rule_size < sizeof(struct fib_rule))
@@ -57,12 +65,13 @@ int fib_rules_register(struct fib_rules_
      ops->action == NULL)
   return -EINVAL;
```

```
+ ops_list = &rules_ops_ns();
  spin_lock(&rules_mod_lock);
- list_for_each_entry(o, &rules_ops, list)
+ list_for_each_entry(o, ops_list, list)
   if (ops->family == o->family)
    goto errout;

- list_add_tail_rcu(&ops->list, &rules_ops);
+ list_add_tail_rcu(&ops->list, ops_list);
  err = 0;
 errout:
  spin_unlock(&rules_mod_lock);
@@ -85,10 +94,12 @@ static void cleanup_ops(struct fib_rules
 int fib_rules_unregister(struct fib_rules_ops *ops)
 {
  int err = 0;
+ struct list_head *ops_list;
  struct fib_rules_ops *o;

+ ops_list = &rules_ops_ns();
  spin_lock(&rules_mod_lock);
- list_for_each_entry(o, &rules_ops, list) {
+ list_for_each_entry(o, ops_list, list) {
   if (o == ops) {
    list_del_rcu(&o->list);
    cleanup_ops(ops);
@@ -131,6 +142,14 @@ int fib_rules_lookup(struct fib_rules_op

  rcu_read_lock();

+ err = -EINVAL;
+ if (ops->rules_list->next == NULL) {
+  if (net_ratelimit())
+   printk(" *** NULL head, ops %p, list %p\n",
+     ops, ops->rules_list);
+  goto out;
+ }
+
  list_for_each_entry_rcu(rule, ops->rules_list, list) {
   if (!fib_rule_match(rule, ops, fl, flags))
    continue;
@@ -141,6 +160,12 @@ int fib_rules_lookup(struct fib_rules_op
    arg->rule = rule;
    goto out;
   }
+  if (rule->list.next == NULL) {
+   if (net_ratelimit())
```

```
+    printk(" *** NULL, ops %p, list %p, item %p\n",
+      ops, ops->rules_list, rule);
+   goto out;
+  }
  }

  err = -ENETUNREACH;
@@ -439,19 +464,21 @@ static int fib_rules_event(struct notifi
        void *ptr)
 {
  struct net_device *dev = ptr;
+ struct list_head *ops_list;
  struct fib_rules_ops *ops;

  ASSERT_RTNL();
  rcu_read_lock();

+ ops_list = &rules_ops_ns();
  switch (event) {
  case NETDEV_REGISTER:
-  list_for_each_entry(ops, &rules_ops, list)
+  list_for_each_entry(ops, ops_list, list)
    attach_rules(ops->rules_list, dev);
   break;

  case NETDEV_UNREGISTER:
-  list_for_each_entry(ops, &rules_ops, list)
+  list_for_each_entry(ops, ops_list, list)
    detach_rules(ops->rules_list, dev);
   break;
  }
@@ -467,6 +494,7 @@ static struct notifier_block fib_rules_n

 static int __init fib_rules_init(void)
 {
+ INIT_LIST_HEAD(&rules_ops_ns());
  return register_netdevice_notifier(&fib_rules_notifier);
 }

--- linux-2.6.20-rc4-mm1.net_ns.orig/net/core/net_namespace.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/core/net_namespace.c
@@ -11,6 +11,7 @@
 #include <linux/net_namespace.h>
 #include <linux/net.h>
 #include <linux/netdevice.h>
+#include <net/ip_fib.h>

 struct net_namespace init_net_ns = {
```

```
  .kref = {
@@ -36,7 +37,7 @@ static struct net_namespace *clone_net_n
 {
  struct net_namespace *ns;

- ns = kmalloc(sizeof(struct net_namespace), GFP_KERNEL);
+ ns = kzalloc(sizeof(struct net_namespace), GFP_KERNEL);
  if (!ns)
   return NULL;

@@ -47,12 +48,19 @@ static struct net_namespace *clone_net_n

  if ((push_net_ns(ns)) != old_ns)
   BUG();
+#ifdef CONFIG_IP_MULTIPLE_TABLES
+ INIT_LIST_HEAD(&ns->fib_rules_ops_list);
+#endif
+ if (ip_fib_struct_init())
+  goto out_fib4;
  if (loopback_init())
   goto out_loopback;
  pop_net_ns(old_ns);
  return ns;

 out_loopback:
+ ip_fib_struct_cleanup(ns);
+out_fib4:
  pop_net_ns(old_ns);
  BUG_ON(atomic_read(&ns->kref.refcount) != 1);
  kfree(ns);
@@ -100,6 +108,7 @@ void free_net_ns(struct kref *kref)
     ns, atomic_read(&ns->kref.refcount));
   return;
  }
+ ip_fib_struct_cleanup(ns);
  kfree(ns);
 }

--- linux-2.6.20-rc4-mm1.net_ns.orig/net/ipv4/fib_frontend.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/ipv4/fib_frontend.c
@@ -50,18 +50,24 @@
 #define FFprint(a...) printk(KERN_DEBUG a)

 #ifndef CONFIG_IP_MULTIPLE_TABLES
-
-struct fib_table *ip_fib_local_table;
-struct fib_table *ip_fib_main_table;
-
```

```
+#ifndef CONFIG_NET_NS
+struct fib_table *ip_fib_local_table_static;
+struct fib_table *ip_fib_main_table_static;
+#endif
 #define FIB_TABLE_HASHSZ 1
-static struct hlist_head fib_table_hash[FIB_TABLE_HASHSZ];
-
 #else
-
 #define FIB_TABLE_HASHSZ 256
-static struct hlist_head fib_table_hash[FIB_TABLE_HASHSZ];
+#endif

+#ifndef CONFIG_NET_NS
+static struct hlist_head fib_table_hash_static[FIB_TABLE_HASHSZ];
+#define fib_table_hash_ns(ns)  fib_table_hash_static
+#else
+#define fib_table_hash_ns(ns)  (ns->fib4_tables)
+#endif
+#define fib_table_hash_current() fib_table_hash_ns(current_net_ns)
+
+#ifdef CONFIG_IP_MULTIPLE_TABLES
 struct fib_table *fib_new_table(u32 id)
 {
  struct fib_table *tb;
@@ -76,21 +82,23 @@ struct fib_table *fib_new_table(u32 id)
  if (!tb)
   return NULL;
  h = id & (FIB_TABLE_HASHSZ - 1);
- hlist_add_head_rcu(&tb->tb_hlist, &fib_table_hash[h]);
+ hlist_add_head_rcu(&tb->tb_hlist, &fib_table_hash_current()[h]);
  return tb;
 }

 struct fib_table *fib_get_table(u32 id)
 {
  struct fib_table *tb;
+ struct hlist_head *list;
  struct hlist_node *node;
  unsigned int h;

  if (id == 0)
   id = RT_TABLE_MAIN;
  h = id & (FIB_TABLE_HASHSZ - 1);
+ list = &fib_table_hash_current()[h];
  rcu_read_lock();
- hlist_for_each_entry_rcu(tb, node, &fib_table_hash[h], tb_hlist) {
+ hlist_for_each_entry_rcu(tb, node, list, tb_hlist) {
```

```
   if (tb->tb_id == id) {
    rcu_read_unlock();
    return tb;
@@ -101,15 +109,17 @@ struct fib_table *fib_get_table(u32 id)
 }
 #endif /* CONFIG_IP_MULTIPLE_TABLES */

-static void fib_flush(void)
+static void fib_flush_ns(struct net_namespace *ns)
 {
  int flushed = 0;
  struct fib_table *tb;
+ struct hlist_head *list;
  struct hlist_node *node;
  unsigned int h;

  for (h = 0; h < FIB_TABLE_HASHSZ; h++) {
-  hlist_for_each_entry(tb, node, &fib_table_hash[h], tb_hlist)
+  list = &fib_table_hash_ns(ns)[h];
+  hlist_for_each_entry(tb, node, list, tb_hlist)
    flushed += tb->tb_flush(tb);
  }

@@ -117,6 +127,11 @@ static void fib_flush(void)
   rt_cache_flush(-1);
 }

+static inline void fib_flush(void)
+{
+ fib_flush_ns(current_net_ns);
+}
+
 /*
  * Find the first device with a given source address.
  */
@@ -125,14 +140,15 @@ struct net_device * ip_dev_find(__be32 a
 {
  struct flowi fl = { .nl_u = { .ip4_u = { .daddr = addr } } };
  struct fib_result res;
+ struct fib_table *tb;
  struct net_device *dev = NULL;

 #ifdef CONFIG_IP_MULTIPLE_TABLES
  res.r = NULL;
 #endif

- if (!ip_fib_local_table ||
-     ip_fib_local_table->tb_lookup(ip_fib_local_table, &fl, &res))
```

```
+ tb = ip_fib_local_table_ns();
+ if (!tb || tb->tb_lookup(tb, &fl, &res))
   return NULL;
  if (res.type != RTN_LOCAL)
   goto out;
@@ -149,6 +165,7 @@ unsigned inet_addr_type(__be32 addr)
 {
  struct flowi  fl = { .nl_u = { .ip4_u = { .daddr = addr } } };
  struct fib_result res;
+ struct fib_table *tb;
  unsigned ret = RTN_BROADCAST;

  if (ZERONET(addr) || BADCLASS(addr))
@@ -160,10 +177,10 @@ unsigned inet_addr_type(__be32 addr)
  res.r = NULL;
 #endif

- if (ip_fib_local_table) {
+ tb = ip_fib_local_table_ns();
+ if (tb) {
   ret = RTN_UNICAST;
-  if (!ip_fib_local_table->tb_lookup(ip_fib_local_table,
-         &fl, &res)) {
+  if (!tb->tb_lookup(tb, &fl, &res)) {
    ret = res.type;
    fib_res_put(&res);
   }
@@ -582,6 +599,7 @@ int inet_dump_fib(struct sk_buff *skb, s
  unsigned int h, s_h;
  unsigned int e = 0, s_e;
  struct fib_table *tb;
+ struct hlist_head *list;
  struct hlist_node *node;
  int dumped = 0;

@@ -594,7 +612,8 @@ int inet_dump_fib(struct sk_buff *skb, s

  for (h = s_h; h < FIB_TABLE_HASHSZ; h++, s_e = 0) {
   e = 0;
-  hlist_for_each_entry(tb, node, &fib_table_hash[h], tb_hlist) {
+  list = &fib_table_hash_current()[h];
+  hlist_for_each_entry(tb, node, list, tb_hlist) {
    if (e < s_e)
     goto next;
    if (dumped)
@@ -896,25 +915,91 @@ static struct notifier_block fib_netdev_
  .notifier_call =fib_netdev_event,
 };
```

```
-void __init ip_fib_init(void)
+#ifdef CONFIG_NET_NS
+int inline ip_fib_struct_init(void)
+{
+ struct hlist_head *tables;
+ unsigned int i;
+
+ tables = kmalloc(FIB_TABLE_HASHSZ * sizeof(*tables), GFP_KERNEL);
+ if (tables == NULL)
+  return -ENOMEM;
+ for (i = 0; i < FIB_TABLE_HASHSZ; i++)
+  INIT_HLIST_HEAD(&tables[i]);
+ fib_table_hash_current() = tables;
+
+#ifdef CONFIG_IP_FIB_TRIE
+ current_net_ns->fib4_trie_last_dflt = -1;
+#endif
+#ifndef CONFIG_IP_MULTIPLE_TABLES
+ ip_fib_local_table_ns() = fib_hash_init(RT_TABLE_LOCAL);
+ hlist_add_head_rcu(&ip_fib_local_table_ns()->tb_hlist,
+   &fib_table_hash_current()[0]);
+ ip_fib_main_table_ns()  = fib_hash_init(RT_TABLE_MAIN);
+ hlist_add_head_rcu(&ip_fib_main_table_ns()->tb_hlist,
+   &fib_table_hash_current()[0]);
+#else
+ if (fib4_rules_init()) {
+  kfree(tables);
+  fib_table_hash_current() = NULL;
+  return -ENOMEM;
+ }
+#endif
+ return 0;
+}
+
+#else /* !defined(CONFIG_NET_NS) */
+
+int inline ip_fib_struct_init(void)
 {
  unsigned int i;

  for (i = 0; i < FIB_TABLE_HASHSZ; i++)
-  INIT_HLIST_HEAD(&fib_table_hash[i]);
+  INIT_HLIST_HEAD(&fib_table_hash_static[i]);
 #ifndef CONFIG_IP_MULTIPLE_TABLES
- ip_fib_local_table = fib_hash_init(RT_TABLE_LOCAL);
- hlist_add_head_rcu(&ip_fib_local_table->tb_hlist, &fib_table_hash[0]);
- ip_fib_main_table  = fib_hash_init(RT_TABLE_MAIN);
```

```
- hlist_add_head_rcu(&ip_fib_main_table->tb_hlist, &fib_table_hash[0]);
+ ip_fib_local_table_ns() = fib_hash_init(RT_TABLE_LOCAL);
+ hlist_add_head_rcu(&ip_fib_local_table_ns()->tb_hlist,
+       &fib_table_hash_current()[0]);
+ ip_fib_main_table_ns()  = fib_hash_init(RT_TABLE_MAIN);
+ hlist_add_head_rcu(&ip_fib_main_table_ns()->tb_hlist,
+       &fib_table_hash_current()[0]);
+ return 0;
 #else
- fib4_rules_init();
+ return fib4_rules_init();
 #endif
+}
+#endif /* CONFIG_NET_NS */
+
+void __init ip_fib_init(void)
+{
+ ip_fib_struct_init();

  register_netdevice_notifier(&fib_netdev_notifier);
  register_inetaddr_notifier(&fib_inetaddr_notifier);
  nl_fib_lookup_init();
 }

+#ifdef CONFIG_NET_NS
+void ip_fib_struct_cleanup(struct net_namespace *ns)
+{
+ rtnl_lock();
+#ifdef CONFIG_IP_MULTIPLE_TABLES
+ fib4_rules_cleanup(ns);
+#endif
+ /*
+  * FIB should already be empty since there is no netdevice,
+  * but clear it anyway
+  */
+ fib_flush_ns(ns);
+ rt_cache_flush(0);
+#ifdef CONFIG_IP_MULTIPLE_TABLES
+ kfree(ns->fib4_tables);
+ ns->fib4_tables = NULL;
+#endif
+ fib_hashtable_destroy(ns);
+ rtnl_unlock();
+}
+#endif /* CONFIG_NET_NS */
+
 EXPORT_SYMBOL(inet_addr_type);
 EXPORT_SYMBOL(ip_dev_find);
```

```
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/ipv4/fib_hash.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/ipv4/fib_hash.c
@@ -629,7 +629,9 @@ static int fn_flush_list(struct fn_zone
   list_for_each_entry_safe(fa, fa_node, &f->fn_alias, fa_list) {
    struct fib_info *fi = fa->fa_info;

-   if (fi && (fi->fib_flags&RTNH_F_DEAD)) {
+   if (fi == NULL)
+    continue;
+   if (fi->fib_flags&RTNH_F_DEAD) {
     write_lock_bh(&fib_hash_lock);
     list_del(&fa->fa_list);
     if (list_empty(&f->fn_alias)) {
@@ -757,7 +759,7 @@ static int fn_hash_dump(struct fib_table
  return skb->len;
 }

-#ifdef CONFIG_IP_MULTIPLE_TABLES
+#if defined(CONFIG_IP_MULTIPLE_TABLES) || defined(CONFIG_NET_NS)
 struct fib_table * fib_hash_init(u32 id)
 #else
 struct fib_table * __init fib_hash_init(u32 id)
@@ -810,7 +812,7 @@ struct fib_iter_state {
 static struct fib_alias *fib_get_first(struct seq_file *seq)
 {
  struct fib_iter_state *iter = seq->private;
- struct fn_hash *table = (struct fn_hash *) ip_fib_main_table->tb_data;
+ struct fn_hash *table = (struct fn_hash *) ip_fib_main_table_ns()->tb_data;

  iter->bucket    = 0;
  iter->hash_head = NULL;
@@ -949,7 +951,7 @@ static void *fib_seq_start(struct seq_fi
  void *v = NULL;

  read_lock(&fib_hash_lock);
- if (ip_fib_main_table)
+ if (ip_fib_main_table_ns())
   v = *pos ? fib_get_idx(seq, *pos - 1) : SEQ_START_TOKEN;
  return v;
 }
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/ipv4/fib_rules.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/ipv4/fib_rules.c
@@ -32,7 +32,7 @@
 #include <net/ip_fib.h>
 #include <net/fib_rules.h>

-static struct fib_rules_ops fib4_rules_ops;
+static struct fib_rules_ops fib4_rules_ops_static;
```

```
 struct fib4_rule
 {
@@ -76,7 +76,12 @@ static struct fib4_rule local_rule = {
  },
 };

-static LIST_HEAD(fib4_rules);
+#ifndef CONFIG_NET_NS
+static LIST_HEAD(fib4_rules_static);
+#define fib4_rules_ops_ns() fib4_rules_ops_static
+#else
+#define fib4_rules_ops_ns() (*current_net_ns->fib4_rules_ops)
+#endif

 #ifdef CONFIG_NET_CLS_ROUTE
 u32 fib_rules_tclass(struct fib_result *res)
@@ -92,7 +97,7 @@ int fib_lookup(struct flowi *flp, struct
  };
  int err;

- err = fib_rules_lookup(&fib4_rules_ops, flp, 0, &arg);
+ err = fib_rules_lookup(&fib4_rules_ops_ns(), flp, 0, &arg);
  res->r = arg.rule;

  return err;
@@ -285,11 +290,13 @@ int fib4_rules_dump(struct sk_buff *skb,
 static u32 fib4_rule_default_pref(void)
 {
  struct list_head *pos;
+ struct list_head *fib4_rules;
  struct fib_rule *rule;

- if (!list_empty(&fib4_rules)) {
-  pos = fib4_rules.next;
-  if (pos->next != &fib4_rules) {
+ fib4_rules = fib4_rules_ops_ns().rules_list;
+ if (!list_empty(fib4_rules)) {
+  pos = fib4_rules->next;
+  if (pos->next != fib4_rules) {
    rule = list_entry(pos->next, struct fib_rule, list);
    if (rule->pref)
     return rule->pref - 1;
@@ -306,7 +313,7 @@ static size_t fib4_rule_nlmsg_payload(st
        + nla_total_size(4); /* flow */
 }

-static struct fib_rules_ops fib4_rules_ops = {
```

```
+static struct fib_rules_ops fib4_rules_ops_static = {
  .family  = AF_INET,
  .rule_size = sizeof(struct fib4_rule),
  .action  = fib4_rule_action,
@@ -318,15 +325,68 @@ static struct fib_rules_ops fib4_rules_o
  .nlmsg_payload = fib4_rule_nlmsg_payload,
  .nlgroup = RTNLGRP_IPV4_RULE,
  .policy  = fib4_rule_policy,
- .rules_list = &fib4_rules,
  .owner  = THIS_MODULE,
 };

-void __init fib4_rules_init(void)
+#ifndef CONFIG_NET_NS
+
+int fib4_rules_init(void)
+{
+ fib4_rules_ops_static.rules_list = &fib4_rules_static,
+ list_add_tail(&local_rule.common.list, &fib4_rules_static);
+ list_add_tail(&main_rule.common.list, &fib4_rules_static);
+ list_add_tail(&default_rule.common.list, &fib4_rules_static);
+ fib_rules_register(&fib4_rules_ops_static);
+ return 0;
+}
+
+#else
+
+static int fib4_rule_create(struct fib4_rule *orig, struct list_head *head)
+{
+ struct fib4_rule *p;
+
+ p = kmalloc(sizeof(*p), GFP_KERNEL);
+ if (p == NULL)
+  return -1;
+ memcpy(p, orig, sizeof(*p));
+ list_add_tail_rcu(&p->common.list, head);
+ return 0;
+}
+
+int fib4_rules_init(void)
 {
- list_add_tail(&local_rule.common.list, &fib4_rules);
- list_add_tail(&main_rule.common.list, &fib4_rules);
- list_add_tail(&default_rule.common.list, &fib4_rules);
+ struct fib_rules_ops *ops;
+ struct list_head *rules;
+
+ ops = kmalloc(sizeof(*ops) + sizeof(*rules), GFP_KERNEL);
```

```
+ if (ops == NULL)
+  goto out;
+ memcpy(ops, &fib4_rules_ops_static, sizeof(*ops));
+ rules = (struct list_head *)(ops + 1);
+ INIT_LIST_HEAD(rules);
+ ops->rules_list = rules;
+ current_net_ns->fib4_rules_ops = ops;
+
+ fib_rules_register(ops);
+
+ if (fib4_rule_create(&local_rule, rules) ||
+     fib4_rule_create(&main_rule, rules) ||
+     fib4_rule_create(&default_rule, rules))
+  goto out_rule;
+
+ return 0;

- fib_rules_register(&fib4_rules_ops);
+out_rule:
+ fib_rules_unregister(ops); /* list cleanup is inside */
+ kfree(ops);
+out:
+ return -ENOMEM;
 }
+
+void fib4_rules_cleanup(struct net_namespace *ns)
+{
+ fib_rules_unregister(ns->fib4_rules_ops);
+}
+
+#endif
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/ipv4/fib_semantics.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/ipv4/fib_semantics.c
@@ -51,10 +51,21 @@
 #define FSprintk(a...)

 static DEFINE_SPINLOCK(fib_info_lock);
-static struct hlist_head *fib_info_hash;
-static struct hlist_head *fib_info_laddrhash;
-static unsigned int fib_hash_size;
-static unsigned int fib_info_cnt;
+#ifndef CONFIG_NET_NS
+static struct hlist_head *fib_info_hash_static;
+static struct hlist_head *fib_info_laddrhash_static;
+static unsigned int fib_hash_size_static;
+static unsigned int fib_info_cnt_static;
+#define fib_info_hash(ns) fib_info_hash_static
+#define fib_info_laddrhash(ns) fib_info_laddrhash_static
```

```
+#define fib_hash_size(ns) fib_hash_size_static
+#define fib_info_cnt(ns) fib_info_cnt_static
+#else
+#define fib_info_hash(ns) ((ns)->fib4_hash)
+#define fib_info_laddrhash(ns) ((ns)->fib4_laddrhash)
+#define fib_hash_size(ns) ((ns)->fib4_hash_size)
+#define fib_info_cnt(ns) ((ns)->fib4_info_cnt)
+#endif

 #define DEVINDEX_HASHBITS 8
 #define DEVINDEX_HASHSIZE (1U << DEVINDEX_HASHBITS)
@@ -154,7 +165,7 @@ void free_fib_info(struct fib_info *fi)
    dev_put(nh->nh_dev);
   nh->nh_dev = NULL;
  } endfor_nexthops(fi);
- fib_info_cnt--;
+ fib_info_cnt(current_net_ns)--;
  kfree(fi);
 }

@@ -197,9 +208,10 @@ static __inline__ int nh_comp(const stru
  return 0;
 }

-static inline unsigned int fib_info_hashfn(const struct fib_info *fi)
+static inline unsigned int fib_info_hashfn(const struct fib_info *fi,
+  struct net_namespace *ns)
 {
- unsigned int mask = (fib_hash_size - 1);
+ unsigned int mask = (fib_hash_size(ns) - 1);
  unsigned int val = fi->fib_nhs;

  val ^= fi->fib_protocol;
@@ -209,15 +221,16 @@ static inline unsigned int fib_info_hash
  return (val ^ (val >> 7) ^ (val >> 12)) & mask;
 }

-static struct fib_info *fib_find_info(const struct fib_info *nfi)
+static noinline struct fib_info *fib_find_info(const struct fib_info *nfi)
 {
+ struct net_namespace *ns = current_net_ns;
  struct hlist_head *head;
  struct hlist_node *node;
  struct fib_info *fi;
  unsigned int hash;

- hash = fib_info_hashfn(nfi);
- head = &fib_info_hash[hash];
```

```
+ hash = fib_info_hashfn(nfi, ns);
+ head = &fib_info_hash(ns)[hash];

  hlist_for_each_entry(fi, node, head, fib_hash) {
   if (fi->fib_nhs != nfi->fib_nhs)
@@ -237,11 +250,13 @@ static struct fib_info *fib_find_info(co

 static inline unsigned int fib_devindex_hashfn(unsigned int val)
 {
- unsigned int mask = DEVINDEX_HASHSIZE - 1;
+ unsigned int r, mask = DEVINDEX_HASHSIZE - 1;

- return (val ^
+ r = val ^
    (val >> DEVINDEX_HASHBITS) ^
-   (val >> (DEVINDEX_HASHBITS * 2))) & mask;
+   (val >> (DEVINDEX_HASHBITS * 2));
+ r ^= net_ns_hash(current_net_ns);
+ return r & mask;
 }

 /* Check, that the gateway is already configured.
@@ -592,9 +607,10 @@ out:
  return 0;
 }

-static inline unsigned int fib_laddr_hashfn(__be32 val)
+static inline unsigned int fib_laddr_hashfn(__be32 val,
+       struct net_namespace *ns)
 {
- unsigned int mask = (fib_hash_size - 1);
+ unsigned int mask = (fib_hash_size(ns) - 1);

  return ((__force u32)val ^ ((__force u32)val >> 7) ^ ((__force u32)val >> 14)) & mask;
 }
@@ -623,17 +639,18 @@ static void fib_hash_move(struct hlist_h
       struct hlist_head *new_laddrhash,
       unsigned int new_size)
 {
+ struct net_namespace *ns = current_net_ns;
  struct hlist_head *old_info_hash, *old_laddrhash;
- unsigned int old_size = fib_hash_size;
+ unsigned int old_size = fib_hash_size(ns);
  unsigned int i, bytes;

  spin_lock_bh(&fib_info_lock);
- old_info_hash = fib_info_hash;
- old_laddrhash = fib_info_laddrhash;
```

```
- fib_hash_size = new_size;
+ old_info_hash = fib_info_hash(ns);
+ old_laddrhash = fib_info_laddrhash(ns);
+ fib_hash_size(ns) = new_size;

  for (i = 0; i < old_size; i++) {
-  struct hlist_head *head = &fib_info_hash[i];
+  struct hlist_head *head = &old_info_hash[i];
   struct hlist_node *node, *n;
   struct fib_info *fi;

@@ -643,15 +660,15 @@ static void fib_hash_move(struct hlist_h

    hlist_del(&fi->fib_hash);

-   new_hash = fib_info_hashfn(fi);
+   new_hash = fib_info_hashfn(fi, ns);
    dest = &new_info_hash[new_hash];
    hlist_add_head(&fi->fib_hash, dest);
   }
  }
- fib_info_hash = new_info_hash;
+ fib_info_hash(ns) = new_info_hash;

  for (i = 0; i < old_size; i++) {
-  struct hlist_head *lhead = &fib_info_laddrhash[i];
+  struct hlist_head *lhead = &old_laddrhash[i];
   struct hlist_node *node, *n;
   struct fib_info *fi;

@@ -661,12 +678,12 @@ static void fib_hash_move(struct hlist_h

    hlist_del(&fi->fib_lhash);

-   new_hash = fib_laddr_hashfn(fi->fib_prefsrc);
+   new_hash = fib_laddr_hashfn(fi->fib_prefsrc, ns);
    ldest = &new_laddrhash[new_hash];
    hlist_add_head(&fi->fib_lhash, ldest);
   }
  }
- fib_info_laddrhash = new_laddrhash;
+ fib_info_laddrhash(ns) = new_laddrhash;

  spin_unlock_bh(&fib_info_lock);

@@ -675,9 +692,23 @@ static void fib_hash_move(struct hlist_h
  fib_hash_free(old_laddrhash, bytes);
 }
```

```
+#ifdef CONFIG_NET_NS
+void fib_hashtable_destroy(struct net_namespace *ns)
+{
+ unsigned int bytes;
+
+ bytes = ns->fib4_hash_size * sizeof(struct hlist_head *);
+ fib_hash_free(ns->fib4_hash, bytes);
+ ns->fib4_hash = NULL;
+ fib_hash_free(ns->fib4_laddrhash, bytes);
+ ns->fib4_laddrhash = NULL;
+}
+#endif
+
 struct fib_info *fib_create_info(struct fib_config *cfg)
 {
  int err;
+ struct net_namespace *ns = current_net_ns;
  struct fib_info *fi = NULL;
  struct fib_info *ofi;
  int nhs = 1;
@@ -702,8 +733,8 @@ struct fib_info *fib_create_info(struct
 #endif

  err = -ENOBUFS;
- if (fib_info_cnt >= fib_hash_size) {
-  unsigned int new_size = fib_hash_size << 1;
+ if (fib_info_cnt(ns) >= fib_hash_size(ns)) {
+  unsigned int new_size = fib_hash_size(ns) << 1;
   struct hlist_head *new_info_hash;
   struct hlist_head *new_laddrhash;
   unsigned int bytes;
@@ -723,14 +754,14 @@ struct fib_info *fib_create_info(struct
    fib_hash_move(new_info_hash, new_laddrhash, new_size);
   }

-  if (!fib_hash_size)
+  if (!fib_hash_size(ns))
    goto failure;
  }

  fi = kzalloc(sizeof(*fi)+nhs*sizeof(struct fib_nh), GFP_KERNEL);
  if (fi == NULL)
   goto failure;
- fib_info_cnt++;
+ fib_info_cnt(ns)++;

  fi->fib_protocol = cfg->fc_protocol;
```

```
  fi->fib_flags = cfg->fc_flags;
@@ -837,11 +868,11 @@ link_it:
  atomic_inc(&fi->fib_clntref);
  spin_lock_bh(&fib_info_lock);
  hlist_add_head(&fi->fib_hash,
-         &fib_info_hash[fib_info_hashfn(fi)]);
+         &fib_info_hash(ns)[fib_info_hashfn(fi, ns)]);
  if (fi->fib_prefsrc) {
   struct hlist_head *head;

-  head = &fib_info_laddrhash[fib_laddr_hashfn(fi->fib_prefsrc)];
+  head = &fib_info_laddrhash(ns)[fib_laddr_hashfn(fi->fib_prefsrc, ns)];
   hlist_add_head(&fi->fib_lhash, head);
  }
  change_nexthops(fi) {
@@ -1043,15 +1074,16 @@ nla_put_failure:

 int fib_sync_down(__be32 local, struct net_device *dev, int force)
 {
+ struct net_namespace *ns = current_net_ns;
  int ret = 0;
  int scope = RT_SCOPE_NOWHERE;

  if (force)
   scope = -1;

- if (local && fib_info_laddrhash) {
-  unsigned int hash = fib_laddr_hashfn(local);
-  struct hlist_head *head = &fib_info_laddrhash[hash];
+ if (local && fib_info_laddrhash(ns)) {
+  unsigned int hash = fib_laddr_hashfn(local, ns);
+  struct hlist_head *head = &fib_info_laddrhash(ns)[hash];
   struct hlist_node *node;
   struct fib_info *fi;

--- linux-2.6.20-rc4-mm1.net_ns.orig/net/ipv4/fib_trie.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/ipv4/fib_trie.c
@@ -172,7 +172,17 @@ static struct tnode *halve(struct trie *
 static void tnode_free(struct tnode *tn);

 static struct kmem_cache *fn_alias_kmem __read_mostly;
-static struct trie *trie_local = NULL, *trie_main = NULL;
+#ifndef CONFIG_NET_NS
+static struct trie *trie_local_static = NULL, *trie_main_static = NULL;
+static int trie_last_dflt_static = -1;
+#define trie_local  trie_local_static
+#define trie_main  trie_main_static
+#define trie_last_dflt  trie_last_dflt_static
```

```
+#else
+#define trie_local  ((struct trie *)ip_fib_local_table_ns()->tb_data)
+#define trie_main  ((struct trie *)ip_fib_main_table_ns()->tb_data)
+#define trie_last_dflt  (current_net_ns->fib4_trie_last_dflt)
+#endif


 /* rcu_read_lock needs to be hold by caller from readside */
@@ -1743,8 +1753,6 @@ static int fn_trie_flush(struct fib_tabl
  return found;
 }

-static int trie_last_dflt = -1;
-
 static void
 fn_trie_select_default(struct fib_table *tb, const struct flowi *flp, struct fib_result *res)
 {
@@ -1929,7 +1937,7 @@ out:

 /* Fix more generic FIB names for init later */

-#ifdef CONFIG_IP_MULTIPLE_TABLES
+#if defined(CONFIG_IP_MULTIPLE_TABLES) || defined(CONFIG_NET_NS)
 struct fib_table * fib_hash_init(u32 id)
 #else
 struct fib_table * __init fib_hash_init(u32 id)
@@ -1962,10 +1970,12 @@ struct fib_table * __init fib_hash_init(

  trie_init(t);

+#ifndef CONFIG_NET_NS
  if (id == RT_TABLE_LOCAL)
   trie_local = t;
  else if (id == RT_TABLE_MAIN)
   trie_main = t;
+#endif

  if (id == RT_TABLE_LOCAL)
   printk(KERN_INFO "IPv4 FIB: Using LC-trie version %s\n", VERSION);
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/ipv4/route.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/ipv4/route.c
@@ -269,6 +269,7 @@ struct rt_cache_iter_state {
  int bucket;
 };

+static struct rtable *rt_cache_get_next(struct seq_file *seq, struct rtable *r);
 static struct rtable *rt_cache_get_first(struct seq_file *seq)
 {
```

```
  struct rtable *r = NULL;
@@ -281,21 +282,28 @@ static struct rtable *rt_cache_get_first
     break;
    rcu_read_unlock_bh();
   }
+ if (r && !net_ns_match(r->fl.net_ns, current_net_ns))
+  r = rt_cache_get_next(seq, r);
   return r;
 }

 static struct rtable *rt_cache_get_next(struct seq_file *seq, struct rtable *r)
 {
   struct rt_cache_iter_state *st = rcu_dereference(seq->private);
+ struct net_namespace *ns = current_net_ns;

+next:
   r = r->u.rt_next;
   while (!r) {
     rcu_read_unlock_bh();
     if (--st->bucket < 0)
-      break;
+      goto out;
     rcu_read_lock_bh();
     r = rt_hash_table[st->bucket].chain;
   }
+ if (!net_ns_match(r->fl.net_ns, ns))
+  goto next;
+out:
   return r;
 }

@@ -571,7 +579,11 @@ static inline int compare_keys(struct fl
     (*(u16 *)&fl1->nl_u.ip4_u.tos ^
      *(u16 *)&fl2->nl_u.ip4_u.tos) |
     (fl1->oif ^ fl2->oif) |
+#ifdef CONFIG_NET_NS
+  (fl1->iif ^ fl2->iif) | (fl1->net_ns != fl2->net_ns)) == 0;
+#else
     (fl1->iif ^ fl2->iif)) == 0;
+#endif
 }

 #ifdef CONFIG_IP_ROUTE_MULTIPATH_CACHED
@@ -1133,6 +1145,7 @@ void ip_rt_redirect(__be32 old_gw, __be3
   struct rtable *rth, **rthp;
   __be32  skeys[2] = { saddr, 0 };
   int  ikeys[2] = { dev->ifindex, 0 };
+ struct net_namespace *ns = current_net_ns;
```

```
  struct netevent_redirect netevent;

  if (!in_dev)
@@ -1164,6 +1177,7 @@ void ip_rt_redirect(__be32 old_gw, __be3

     if (rth->fl.fl4_dst != daddr ||
         rth->fl.fl4_src != skeys[i] ||
+        !net_ns_match(rth->fl.net_ns, ns) ||
         rth->fl.oif != ikeys[k] ||
         rth->fl.iif != 0) {
      rthp = &rth->u.rt_next;
@@ -1653,6 +1667,9 @@ static int ip_route_input_mc(struct sk_b
  dev_hold(rth->u.dst.dev);
  rth->idev = in_dev_get(rth->u.dst.dev);
  rth->fl.oif = 0;
+#ifdef CONFIG_NET_NS
+ rth->fl.net_ns = current_net_ns;
+#endif
  rth->rt_gateway = daddr;
  rth->rt_spec_dst= spec_dst;
  rth->rt_type = RTN_MULTICAST;
@@ -1795,6 +1812,9 @@ static inline int __mkroute_input(struct
  dev_hold(rth->u.dst.dev);
  rth->idev = in_dev_get(rth->u.dst.dev);
  rth->fl.oif  = 0;
+#ifdef CONFIG_NET_NS
+ rth->fl.net_ns = current_net_ns;
+#endif
  rth->rt_spec_dst= spec_dst;

  rth->u.dst.input = ip_forward;
@@ -2037,6 +2057,9 @@ local_input:
  rth->u.dst.dev = &loopback_dev;
  dev_hold(rth->u.dst.dev);
  rth->idev = in_dev_get(rth->u.dst.dev);
+#ifdef CONFIG_NET_NS
+ rth->fl.net_ns = current_net_ns;
+#endif
  rth->rt_gateway = daddr;
  rth->rt_spec_dst= spec_dst;
  rth->u.dst.input= ip_local_deliver;
@@ -2092,6 +2115,7 @@ int ip_route_input(struct sk_buff *skb,
  struct rtable * rth;
  unsigned hash;
  int iif = dev->ifindex;
+ struct net_namespace *ns = current_net_ns;

  tos &= IPTOS_RT_MASK;
```

```
  hash = rt_hash(daddr, saddr, iif);
@@ -2101,6 +2125,7 @@ int ip_route_input(struct sk_buff *skb,
       rth = rcu_dereference(rth->u.rt_next)) {
   if (rth->fl.fl4_dst == daddr &&
       rth->fl.fl4_src == saddr &&
+      net_ns_match(rth->fl.net_ns, ns) &&
       rth->fl.iif == iif &&
       rth->fl.oif == 0 &&
       rth->fl.mark == skb->mark &&
@@ -2236,6 +2261,9 @@ static inline int __mkroute_output(struc
  rth->u.dst.dev = dev_out;
  dev_hold(dev_out);
  rth->idev = in_dev_get(dev_out);
+#ifdef CONFIG_NET_NS
+ rth->fl.net_ns = current_net_ns;
+#endif
  rth->rt_gateway = fl->fl4_dst;
  rth->rt_spec_dst= fl->fl4_src;

@@ -2557,6 +2585,7 @@ int __ip_route_output_key(struct rtable
 {
  unsigned hash;
  struct rtable *rth;
+ struct net_namespace *ns = current_net_ns;

  hash = rt_hash(flp->fl4_dst, flp->fl4_src, flp->oif);

@@ -2565,6 +2594,7 @@ int __ip_route_output_key(struct rtable
   rth = rcu_dereference(rth->u.rt_next)) {
   if (rth->fl.fl4_dst == flp->fl4_dst &&
       rth->fl.fl4_src == flp->fl4_src &&
+      net_ns_match(rth->fl.net_ns, ns) &&
       rth->fl.iif == 0 &&
       rth->fl.oif == flp->oif &&
       rth->fl.mark == flp->mark &&
```

Subject: [PATCH 6/12] L2 network namespace (v3): socket hashes
Posted by Mishin Dmitry on Wed, 17 Jan 2007 16:05:35 GMT
View Forum Message <> Reply to Message

Socket hash lookups are made within namespace. Hash tables are common for all
namespaces, with additional permutation of indexes. Asynchronous events
should be run in proper namespace.

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

---
include/linux/ipv6.h | 3 ++-
include/net/inet6_hashtables.h | 6 ++++--
include/net/inet_hashtables.h | 36 ++++++++++++++++++++++++++----------
include/net/inet_sock.h | 6 ++++--
include/net/inet_timewait_sock.h | 2 ++
include/net/sock.h | 2 ++
include/net/udp.h | 4 ++++
net/core/sock.c | 6 ++++++
net/ipv4/af_inet.c | 11 +++++++++++
net/ipv4/inet_connection_sock.c | 19 +++++++++++++++----
net/ipv4/inet_hashtables.c | 30 ++++++++++++++++++++++++------
net/ipv4/inet_timewait_sock.c | 17 +++++++++++++++--
net/ipv4/raw.c | 2 ++
net/ipv4/tcp_timer.c | 9 +++++++++
net/ipv4/udp.c | 27 +++++++++++++++++++++------
net/ipv6/inet6_connection_sock.c | 2 ++
net/ipv6/inet6_hashtables.c | 25 +++++++++++++++++++------
net/ipv6/raw.c | 4 ++++
net/ipv6/udp.c | 10 +++++++++-
19 files changed, 172 insertions(+), 49 deletions(-)

--- linux-2.6.20-rc4-mm1.net_ns.orig/include/linux/ipv6.h
+++ linux-2.6.20-rc4-mm1.net_ns/include/linux/ipv6.h
@@ -459,10 +459,11 @@ static inline struct raw6_sock *raw6_sk(
#define inet_v6_ipv6only(__sk) 0
#endif /* defined(CONFIG_IPV6) || defined(CONFIG_IPV6_MODULE) */

-#define INET6_MATCH(__sk, __hash, __saddr, __daddr, __ports, __dif)\
+#define INET6_MATCH(__sk, __hash, __saddr, __daddr, __ports, __dif, __ns)\
(((__sk)->sk_hash == (__hash)) && \
((*((__portpair *)&(inet_sk(__sk)->dport))) == (__ports)) && \
((__sk)->sk_family == AF_INET6) && \
+ net_ns_match((__sk)->sk_net_ns, __ns) && \
ipv6_addr_equal(&inet6_sk(__sk)->daddr, (__saddr)) && \
ipv6_addr_equal(&inet6_sk(__sk)->rcv_saddr, (__daddr)) && \
(!((__sk)->sk_bound_dev_if) || ((__sk)->sk_bound_dev_if == (__dif))))
--- linux-2.6.20-rc4-mm1.net_ns.orig/include/net/inet6_hashtables.h
+++ linux-2.6.20-rc4-mm1.net_ns/include/net/inet6_hashtables.h
@@ -26,11 +26,13 @@ struct inet_hashinfo;

/* I have no idea if this is a good hash for v6 or not. -DaveM */
static inline unsigned int inet6_ehashfn(const struct in6_addr *laddr, const u16 lport,
- const struct in6_addr *faddr, const __be16 fport)
+ const struct in6_addr *faddr, const __be16 fport,

```
+    struct net_namespace *ns)
 {
  unsigned int hashent = (lport ^ (__force u16)fport);

  hashent ^= (__force u32)(laddr->s6_addr32[3] ^ faddr->s6_addr32[3]);
+ hashent ^= net_ns_hash(ns);
  hashent ^= hashent >> 16;
  hashent ^= hashent >> 8;
  return hashent;
@@ -44,7 +46,7 @@ static inline int inet6_sk_ehashfn(const
  const struct in6_addr *faddr = &np->daddr;
  const __u16 lport = inet->num;
  const __be16 fport = inet->dport;
- return inet6_ehashfn(laddr, lport, faddr, fport);
+ return inet6_ehashfn(laddr, lport, faddr, fport, current_net_ns);
 }

 extern void __inet6_hash(struct inet_hashinfo *hashinfo, struct sock *sk);
--- linux-2.6.20-rc4-mm1.net_ns.orig/include/net/inet_hashtables.h
+++ linux-2.6.20-rc4-mm1.net_ns/include/net/inet_hashtables.h
@@ -78,6 +78,7 @@ struct inet_bind_bucket {
  signed short  fastreuse;
  struct hlist_node node;
  struct hlist_head owners;
+ struct net_namespace *net_ns;
 };

 #define inet_bind_bucket_for_each(tb, node, head) \
@@ -142,30 +143,34 @@ extern struct inet_bind_bucket *
 extern void inet_bind_bucket_destroy(struct kmem_cache *cachep,
         struct inet_bind_bucket *tb);

-static inline int inet_bhashfn(const __u16 lport, const int bhash_size)
+static inline int inet_bhashfn(const __u16 lport,
+         struct net_namespace *ns,
+         const int bhash_size)
 {
- return lport & (bhash_size - 1);
+ return (lport ^ net_ns_hash(ns)) & (bhash_size - 1);
 }

 extern void inet_bind_hash(struct sock *sk, struct inet_bind_bucket *tb,
       const unsigned short snum);

 /* These can have wildcards, don't try too hard. */
-static inline int inet_lhashfn(const unsigned short num)
+static inline int inet_lhashfn(const unsigned short num,
+         struct net_namespace *ns)
```

```
 {
- return num & (INET_LHTABLE_SIZE - 1);
+ return (num ^ net_ns_hash(ns)) & (INET_LHTABLE_SIZE - 1);
 }

 static inline int inet_sk_listen_hashfn(const struct sock *sk)
 {
- return inet_lhashfn(inet_sk(sk)->num);
+ return inet_lhashfn(inet_sk(sk)->num, current_net_ns);
 }

 /* Caller must disable local BH processing. */
 static inline void __inet_inherit_port(struct inet_hashinfo *table,
           struct sock *sk, struct sock *child)
 {
- const int bhash = inet_bhashfn(inet_sk(child)->num, table->bhash_size);
+ const int bhash = inet_bhashfn(inet_sk(child)->num, current_net_ns,
+     table->bhash_size);
  struct inet_bind_hashbucket *head = &table->bhash[bhash];
  struct inet_bind_bucket *tb;

@@ -313,29 +318,33 @@ typedef __u64 __bitwise __addrpair;
       (((__force __u64)(__be32)(__daddr)) << 32) | \
       ((__force __u64)(__be32)(__saddr)));
 #endif /* __BIG_ENDIAN */
-#define INET_MATCH(__sk, __hash, __cookie, __saddr, __daddr, __ports, __dif)\
+#define INET_MATCH(__sk, __hash, __cookie, __saddr, __daddr, __ports, __dif, __ns)\
  (((__sk)->sk_hash == (__hash))    && \
   ((*((__addrpair *)&(inet_sk(__sk)->daddr))) == (__cookie)) && \
   ((*((__portpair *)&(inet_sk(__sk)->dport))) == (__ports)) && \
+  net_ns_match((__sk)->sk_net_ns, __ns)    && \
   (!((__sk)->sk_bound_dev_if) || ((__sk)->sk_bound_dev_if == (__dif))))
-#define INET_TW_MATCH(__sk, __hash, __cookie, __saddr, __daddr, __ports, __dif)\
+#define INET_TW_MATCH(__sk, __hash, __cookie, __saddr, __daddr, __ports, __dif, __ns)\
  (((__sk)->sk_hash == (__hash))    && \
   ((*((__addrpair *)&(inet_twsk(__sk)->tw_daddr))) == (__cookie)) && \
   ((*((__portpair *)&(inet_twsk(__sk)->tw_dport))) == (__ports)) && \
+  net_ns_match((__sk)->sk_net_ns, __ns)    && \
   (!((__sk)->sk_bound_dev_if) || ((__sk)->sk_bound_dev_if == (__dif))))
 #else /* 32-bit arch */
 #define INET_ADDR_COOKIE(__name, __saddr, __daddr)
-#define INET_MATCH(__sk, __hash, __cookie, __saddr, __daddr, __ports, __dif) \
+#define INET_MATCH(__sk, __hash, __cookie, __saddr, __daddr, __ports, __dif, __ns)\
  (((__sk)->sk_hash == (__hash))    && \
   (inet_sk(__sk)->daddr  == (__saddr))  && \
   (inet_sk(__sk)->rcv_saddr == (__daddr))  && \
   ((*((__portpair *)&(inet_sk(__sk)->dport))) == (__ports)) && \
+  net_ns_match((__sk)->sk_net_ns, __ns)   && \
```

```
   (!((__sk)->sk_bound_dev_if) || ((__sk)->sk_bound_dev_if == (__dif))))
-#define INET_TW_MATCH(__sk, __hash,__cookie, __saddr, __daddr, __ports, __dif) \
+#define INET_TW_MATCH(__sk, __hash,__cookie, __saddr, __daddr, __ports, __dif, __ns)\
   (((__sk)->sk_hash == (__hash))    && \
    (inet_twsk(__sk)->tw_daddr == (__saddr))  && \
    (inet_twsk(__sk)->tw_rcv_saddr == (__daddr))  && \
    ((*((__portpair *)&(inet_twsk(__sk)->tw_dport))) == (__ports)) && \
+  net_ns_match((__sk)->sk_net_ns, __ns)   && \
    (!((__sk)->sk_bound_dev_if) || ((__sk)->sk_bound_dev_if == (__dif))))
 #endif /* 64-bit arch */

@@ -355,22 +364,23 @@ static inline struct sock *
  const __portpair ports = INET_COMBINED_PORTS(sport, hnum);
  struct sock *sk;
  const struct hlist_node *node;
+ struct net_namespace *ns = current_net_ns;
  /* Optimize here for direct hit, only listening connections can
   * have wildcards anyways.
   */
- unsigned int hash = inet_ehashfn(daddr, hnum, saddr, sport);
+ unsigned int hash = inet_ehashfn(daddr, hnum, saddr, sport, ns);
  struct inet_ehash_bucket *head = inet_ehash_bucket(hashinfo, hash);

  prefetch(head->chain.first);
  read_lock(&head->lock);
  sk_for_each(sk, node, &head->chain) {
-  if (INET_MATCH(sk, hash, acookie, saddr, daddr, ports, dif))
+  if (INET_MATCH(sk, hash, acookie, saddr, daddr, ports, dif, ns))
    goto hit; /* You sunk my battleship! */
  }

  /* Must check for a TIME_WAIT'er before going to listener hash. */
  sk_for_each(sk, node, &(head + hashinfo->ehash_size)->chain) {
-  if (INET_TW_MATCH(sk, hash, acookie, saddr, daddr, ports, dif))
+  if (INET_TW_MATCH(sk, hash, acookie, saddr, daddr, ports, dif, ns))
    goto hit;
  }
  sk = NULL;
--- linux-2.6.20-rc4-mm1.net_ns.orig/include/net/inet_sock.h
+++ linux-2.6.20-rc4-mm1.net_ns/include/net/inet_sock.h
@@ -168,9 +168,11 @@ static inline void inet_sk_copy_descenda
 extern int inet_sk_rebuild_header(struct sock *sk);

 static inline unsigned int inet_ehashfn(const __be32 laddr, const __u16 lport,
-     const __be32 faddr, const __be16 fport)
+     const __be32 faddr, const __be16 fport,
+     struct net_namespace *ns)
 {
```

```
  unsigned int h = ((__force __u32)laddr ^ lport) ^ ((__force __u32)faddr ^ (__force __u32)fport);
+ h ^= net_ns_hash(ns);
  h ^= h >> 16;
  h ^= h >> 8;
  return h;
@@ -184,7 +186,7 @@ static inline int inet_sk_ehashfn(const
  const __be32 faddr = inet->daddr;
  const __be16 fport = inet->dport;

- return inet_ehashfn(laddr, lport, faddr, fport);
+ return inet_ehashfn(laddr, lport, faddr, fport, current_net_ns);
 }

 #endif /* _INET_SOCK_H */
--- linux-2.6.20-rc4-mm1.net_ns.orig/include/net/inet_timewait_sock.h
+++ linux-2.6.20-rc4-mm1.net_ns/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_ns  __tw_common.skc_net_ns
  volatile unsigned char tw_substate;
  /* 3 bits hole, try to pack */
  unsigned char  tw_rcv_wscale;
@@ -201,6 +202,7 @@ static inline void inet_twsk_put(struct
   printk(KERN_DEBUG "%s timewait_sock %p released\n",
          tw->tw_prot->name, tw);
 #endif
+  put_net_ns(tw->tw_net_ns);
   kmem_cache_free(tw->tw_prot->twsk_prot->twsk_slab, tw);
   module_put(owner);
  }
--- linux-2.6.20-rc4-mm1.net_ns.orig/include/net/sock.h
+++ linux-2.6.20-rc4-mm1.net_ns/include/net/sock.h
@@ -119,6 +119,7 @@ struct sock_common {
  atomic_t  skc_refcnt;
  unsigned int  skc_hash;
  struct proto  *skc_prot;
+ struct net_namespace *skc_net_ns;
 };

 /**
@@ -195,6 +196,7 @@ 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_ns  __sk_common.skc_net_ns
  unsigned char  sk_shutdown : 2,
```

```
     sk_no_check : 2,
     sk_userlocks : 4;
--- linux-2.6.20-rc4-mm1.net_ns.orig/include/net/udp.h
+++ linux-2.6.20-rc4-mm1.net_ns/include/net/udp.h
@@ -53,6 +53,10 @@ struct udp_skb_cb {
 extern struct hlist_head udp_hash[UDP_HTABLE_SIZE];
 extern rwlock_t udp_hash_lock;

+static inline int udp_hashfn(u16 num, struct net_namespace *ns)
+{
+ return (num ^ net_ns_hash(ns)) & (UDP_HTABLE_SIZE - 1);
+}

 /* Note: this must match 'valbool' in sock_setsockopt */
 #define UDP_CSUM_NOXMIT  1
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/core/sock.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/core/sock.c
@@ -845,6 +845,10 @@ struct sock *sk_alloc(int family, gfp_t
     */
    sk->sk_prot = sk->sk_prot_creator = prot;
    sock_lock_init(sk);
+#ifdef CONFIG_NET_NS
+   get_net_ns(current_net_ns);
+   sk->sk_net_ns = current_net_ns;
+#endif
   }

   if (security_sk_alloc(sk, family, priority))
@@ -884,6 +888,7 @@ void sk_free(struct sock *sk)
         __FUNCTION__, atomic_read(&sk->sk_omem_alloc));

  security_sk_free(sk);
+ put_net_ns(sk->sk_net_ns);
  if (sk->sk_prot_creator->slab != NULL)
   kmem_cache_free(sk->sk_prot_creator->slab, sk);
  else
@@ -919,6 +924,7 @@ struct sock *sk_clone(const struct sock
   lockdep_set_class(&newsk->sk_callback_lock,
       af_callback_keys + newsk->sk_family);

+  (void) get_net_ns(newsk->sk_net_ns);
   newsk->sk_dst_cache = NULL;
   newsk->sk_wmem_queued = 0;
   newsk->sk_forward_alloc = 0;
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/ipv4/af_inet.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/ipv4/af_inet.c
@@ -367,10 +367,18 @@ out_rcu_unlock:
 int inet_release(struct socket *sock)
```

```
 {
  struct sock *sk = sock->sk;
+ struct net_namespace *ns, *orig_net_ns;

  if (sk) {
   long timeout;

+  /* Need to change namespace here since protocol ->close
+   * operation may send packets.
+   */
+  get_net_ns(sk->sk_net_ns);
+  ns = sk->sk_net_ns;
+  orig_net_ns = push_net_ns(ns);
+
   /* Applications forget to leave groups before exiting */
   ip_mc_drop_socket(sk);

@@ -387,6 +395,9 @@ int inet_release(struct socket *sock)
    timeout = sk->sk_lingertime;
   sock->sk = NULL;
   sk->sk_prot->close(sk, timeout);
+
+  pop_net_ns(orig_net_ns);
+  put_net_ns(ns);
  }
  return 0;
 }
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/ipv4/inet_connection_sock.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/ipv4/inet_connection_sock.c
@@ -43,10 +43,12 @@ int inet_csk_bind_conflict(const struct
  struct sock *sk2;
  struct hlist_node *node;
  int reuse = sk->sk_reuse;
+ struct net_namespace *ns = current_net_ns;

  sk_for_each_bound(sk2, node, &tb->owners) {
   if (sk != sk2 &&
       !inet_v6_ipv6only(sk2) &&
+      net_ns_match(sk2->sk_net_ns, ns) &&
       (!sk->sk_bound_dev_if ||
        !sk2->sk_bound_dev_if ||
        sk->sk_bound_dev_if == sk2->sk_bound_dev_if)) {
@@ -75,6 +77,7 @@ int inet_csk_get_port(struct inet_hashin
  struct inet_bind_hashbucket *head;
  struct hlist_node *node;
  struct inet_bind_bucket *tb;
+ struct net_namespace *ns = current_net_ns;
  int ret;
```

```
  local_bh_disable();
@@ -85,11 +88,15 @@ int inet_csk_get_port(struct inet_hashin
   int rover = net_random() % (high - low) + low;

   do {
-   head = &hashinfo->bhash[inet_bhashfn(rover, hashinfo->bhash_size)];
+   head = &hashinfo->bhash[inet_bhashfn(rover, ns,
+        hashinfo->bhash_size)];
    spin_lock(&head->lock);
-   inet_bind_bucket_for_each(tb, node, &head->chain)
+   inet_bind_bucket_for_each(tb, node, &head->chain) {
+    if (!net_ns_match(tb->net_ns, ns))
+     continue;
     if (tb->port == rover)
      goto next;
+   }
    break;
   next:
    spin_unlock(&head->lock);
@@ -112,11 +119,15 @@ int inet_csk_get_port(struct inet_hashin
    */
   snum = rover;
  } else {
-  head = &hashinfo->bhash[inet_bhashfn(snum, hashinfo->bhash_size)];
+  head = &hashinfo->bhash[inet_bhashfn(snum, ns,
+       hashinfo->bhash_size)];
   spin_lock(&head->lock);
-  inet_bind_bucket_for_each(tb, node, &head->chain)
+  inet_bind_bucket_for_each(tb, node, &head->chain) {
+   if (!net_ns_match(tb->net_ns, ns))
+    continue;
    if (tb->port == snum)
     goto tb_found;
+  }
  }
  tb = NULL;
  goto tb_not_found;
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/ipv4/inet_hashtables.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/ipv4/inet_hashtables.c
@@ -36,6 +36,10 @@ struct inet_bind_bucket *inet_bind_bucke
  if (tb != NULL) {
   tb->port      = snum;
   tb->fastreuse = 0;
+#ifdef CONFIG_NET_NS
+  get_net_ns(current_net_ns);
+  tb->net_ns = current_net_ns;
+#endif
```

```
   INIT_HLIST_HEAD(&tb->owners);
   hlist_add_head(&tb->node, &head->chain);
  }
@@ -49,6 +53,7 @@ void inet_bind_bucket_destroy(struct kme
 {
  if (hlist_empty(&tb->owners)) {
   __hlist_del(&tb->node);
+  put_net_ns(tb->net_ns);
   kmem_cache_free(cachep, tb);
  }
 }
@@ -66,7 +71,8 @@ void inet_bind_hash(struct sock *sk, str
  */
 static void __inet_put_port(struct inet_hashinfo *hashinfo, struct sock *sk)
 {
- const int bhash = inet_bhashfn(inet_sk(sk)->num, hashinfo->bhash_size);
+ const int bhash = inet_bhashfn(inet_sk(sk)->num, current_net_ns,
+     hashinfo->bhash_size);
  struct inet_bind_hashbucket *head = &hashinfo->bhash[bhash];
  struct inet_bind_bucket *tb;

@@ -130,12 +136,15 @@ static struct sock *inet_lookup_listener
           const int dif)
 {
  struct sock *result = NULL, *sk;
+ struct net_namespace *ns = current_net_ns;
  const struct hlist_node *node;
  int hiscore = -1;

  sk_for_each(sk, node, head) {
   const struct inet_sock *inet = inet_sk(sk);

+  if (!net_ns_match(sk->sk_net_ns, ns))
+   continue;
   if (inet->num == hnum && !ipv6_only_sock(sk)) {
    const __be32 rcv_saddr = inet->rcv_saddr;
    int score = sk->sk_family == PF_INET ? 1 : 0;
@@ -168,14 +177,16 @@ struct sock *__inet_lookup_listener(stru
 {
  struct sock *sk = NULL;
  const struct hlist_head *head;
+ struct net_namespace *ns = current_net_ns;

  read_lock(&hashinfo->lhash_lock);
- head = &hashinfo->listening_hash[inet_lhashfn(hnum)];
+ head = &hashinfo->listening_hash[inet_lhashfn(hnum, ns)];
  if (!hlist_empty(head)) {
   const struct inet_sock *inet = inet_sk((sk = __sk_head(head)));
```

```
   if (inet->num == hnum && !sk->sk_node.next &&
       (!inet->rcv_saddr || inet->rcv_saddr == daddr) &&
+       net_ns_match(sk->sk_net_ns, ns) &&
       (sk->sk_family == PF_INET || !ipv6_only_sock(sk)) &&
       !sk->sk_bound_dev_if)
     goto sherry_cache;
@@ -202,7 +213,8 @@ static int __inet_check_established(stru
  int dif = sk->sk_bound_dev_if;
  INET_ADDR_COOKIE(acookie, saddr, daddr)
  const __portpair ports = INET_COMBINED_PORTS(inet->dport, lport);
- unsigned int hash = inet_ehashfn(daddr, lport, saddr, inet->dport);
+ struct net_namespace *ns = current_net_ns;
+ unsigned int hash = inet_ehashfn(daddr, lport, saddr, inet->dport, ns);
  struct inet_ehash_bucket *head = inet_ehash_bucket(hinfo, hash);
  struct sock *sk2;
  const struct hlist_node *node;
@@ -215,7 +227,7 @@ static int __inet_check_established(stru
  sk_for_each(sk2, node, &(head + hinfo->ehash_size)->chain) {
   tw = inet_twsk(sk2);

-  if (INET_TW_MATCH(sk2, hash, acookie, saddr, daddr, ports, dif)) {
+  if (INET_TW_MATCH(sk2, hash, acookie, saddr, daddr, ports, dif, ns)) {
     if (twsk_unique(sk, sk2, twp))
      goto unique;
     else
@@ -226,7 +238,7 @@ static int __inet_check_established(stru

  /* And established part... */
  sk_for_each(sk2, node, &head->chain) {
-  if (INET_MATCH(sk2, hash, acookie, saddr, daddr, ports, dif))
+  if (INET_MATCH(sk2, hash, acookie, saddr, daddr, ports, dif, ns))
     goto not_unique;
  }

@@ -274,6 +286,7 @@ int inet_hash_connect(struct inet_timewa
 {
  struct inet_hashinfo *hinfo = death_row->hashinfo;
  const unsigned short snum = inet_sk(sk)->num;
+ struct net_namespace *ns = current_net_ns;
   struct inet_bind_hashbucket *head;
   struct inet_bind_bucket *tb;
  int ret;
@@ -292,7 +305,8 @@ int inet_hash_connect(struct inet_timewa
     local_bh_disable();
   for (i = 1; i <= range; i++) {
     port = low + (i + offset) % range;
-    head = &hinfo->bhash[inet_bhashfn(port, hinfo->bhash_size)];
```

```
+    head = &hinfo->bhash[inet_bhashfn(port, ns,
+      hinfo->bhash_size)];
     spin_lock(&head->lock);

     /* Does not bother with rcv_saddr checks,
@@ -300,6 +314,8 @@ int inet_hash_connect(struct inet_timewa
      * unique enough.
      */
    inet_bind_bucket_for_each(tb, node, &head->chain) {
+    if (!net_ns_match(tb->net_ns, ns))
+     continue;
      if (tb->port == port) {
       BUG_TRAP(!hlist_empty(&tb->owners));
       if (tb->fastreuse >= 0)
@@ -347,7 +363,7 @@ ok:
   goto out;
   }

-  head = &hinfo->bhash[inet_bhashfn(snum, hinfo->bhash_size)];
+  head = &hinfo->bhash[inet_bhashfn(snum, ns, hinfo->bhash_size)];
   tb  = inet_csk(sk)->icsk_bind_hash;
  spin_lock_bh(&head->lock);
  if (sk_head(&tb->owners) == sk && !sk->sk_bind_node.next) {
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/ipv4/inet_timewait_sock.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/ipv4/inet_timewait_sock.c
@@ -31,7 +31,7 @@ void __inet_twsk_kill(struct inet_timewa
  write_unlock(&ehead->lock);

  /* Disassociate with bind bucket. */
- bhead = &hashinfo->bhash[inet_bhashfn(tw->tw_num, hashinfo->bhash_size)];
+ bhead = &hashinfo->bhash[inet_bhashfn(tw->tw_num, current_net_ns, hashinfo->bhash_size)];
  spin_lock(&bhead->lock);
  tb = tw->tw_tb;
  __hlist_del(&tw->tw_bind_node);
@@ -65,7 +65,8 @@ void __inet_twsk_hashdance(struct inet_t
     Note, that any socket with inet->num != 0 MUST be bound in
     binding cache, even if it is closed.
   */
- bhead = &hashinfo->bhash[inet_bhashfn(inet->num, hashinfo->bhash_size)];
+ bhead = &hashinfo->bhash[inet_bhashfn(inet->num, current_net_ns,
+     hashinfo->bhash_size)];
  spin_lock(&bhead->lock);
  tw->tw_tb = icsk->icsk_bind_hash;
  BUG_TRAP(icsk->icsk_bind_hash);
@@ -109,6 +110,10 @@ struct inet_timewait_sock *inet_twsk_all
   tw->tw_hash     = sk->sk_hash;
   tw->tw_ipv6only     = 0;
   tw->tw_prot     = sk->sk_prot_creator;
```

```
+#ifdef CONFIG_NET_NS
+  get_net_ns(current_net_ns);
+  tw->tw_net_ns     = current_net_ns;
+#endif
   atomic_set(&tw->tw_refcnt, 1);
   inet_twsk_dead_node_init(tw);
   __module_get(tw->tw_prot->owner);
@@ -125,6 +130,7 @@ static int inet_twdr_do_twkill_work(stru
 {
  struct inet_timewait_sock *tw;
  struct hlist_node *node;
+ struct net_namespace *orig_net_ns;
  unsigned int killed;
  int ret;

@@ -136,8 +142,10 @@ static int inet_twdr_do_twkill_work(stru
   */
  killed = 0;
  ret = 0;
+ orig_net_ns = current_net_ns;
 rescan:
  inet_twsk_for_each_inmate(tw, node, &twdr->cells[slot]) {
+  (void)push_net_ns(tw->tw_net_ns);
   __inet_twsk_del_dead_node(tw);
   spin_unlock(&twdr->death_lock);
   __inet_twsk_kill(tw, twdr->hashinfo);
@@ -160,6 +168,7 @@ rescan:

  twdr->tw_count -= killed;
  NET_ADD_STATS_BH(LINUX_MIB_TIMEWAITED, killed);
+ pop_net_ns(orig_net_ns);

  return ret;
 }
@@ -334,10 +343,12 @@ void inet_twdr_twcal_tick(unsigned long
  int n, slot;
  unsigned long j;
  unsigned long now = jiffies;
+ struct net_namespace *orig_net_ns;
  int killed = 0;
  int adv = 0;

  twdr = (struct inet_timewait_death_row *)data;
+ orig_net_ns = current_net_ns;

  spin_lock(&twdr->death_lock);
  if (twdr->twcal_hand < 0)
@@ -353,6 +364,7 @@ void inet_twdr_twcal_tick(unsigned long
```

```
    inet_twsk_for_each_inmate_safe(tw, node, safe,
            &twdr->twcal_row[slot]) {
+    (void)push_net_ns(tw->tw_net_ns);
     __inet_twsk_del_dead_node(tw);
     __inet_twsk_kill(tw, twdr->hashinfo);
     inet_twsk_put(tw);
@@ -380,6 +392,7 @@ out:
   del_timer(&twdr->tw_timer);
  NET_ADD_STATS_BH(LINUX_MIB_TIMEWAITKILLED, killed);
  spin_unlock(&twdr->death_lock);
+ pop_net_ns(orig_net_ns);
 }

 EXPORT_SYMBOL_GPL(inet_twdr_twcal_tick);
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/ipv4/raw.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/ipv4/raw.c
@@ -106,6 +106,7 @@ struct sock *__raw_v4_lookup(struct sock
        int dif)
 {
  struct hlist_node *node;
+ struct net_namespace *ns = current_net_ns;

  sk_for_each_from(sk, node) {
   struct inet_sock *inet = inet_sk(sk);
@@ -113,6 +114,7 @@ struct sock *__raw_v4_lookup(struct sock
   if (inet->num == num      &&
       !(inet->daddr && inet->daddr != raddr)   &&
       !(inet->rcv_saddr && inet->rcv_saddr != laddr) &&
+      net_ns_match(sk->sk_net_ns, ns)   &&
       !(sk->sk_bound_dev_if && sk->sk_bound_dev_if != dif))
    goto found; /* gotcha */
  }
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/ipv4/tcp_timer.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/ipv4/tcp_timer.c
@@ -171,7 +171,9 @@ static void tcp_delack_timer(unsigned lo
  struct sock *sk = (struct sock*)data;
  struct tcp_sock *tp = tcp_sk(sk);
  struct inet_connection_sock *icsk = inet_csk(sk);
+ struct net_namespace *orig_net_ns;

+ orig_net_ns = push_net_ns(sk->sk_net_ns);
  bh_lock_sock(sk);
  if (sock_owned_by_user(sk)) {
   /* Try again later. */
@@ -225,6 +227,7 @@ out:
 out_unlock:
  bh_unlock_sock(sk);
```

```
  sock_put(sk);
+ pop_net_ns(orig_net_ns);
 }

 static void tcp_probe_timer(struct sock *sk)
@@ -384,8 +387,10 @@ static void tcp_write_timer(unsigned lon
 {
  struct sock *sk = (struct sock*)data;
  struct inet_connection_sock *icsk = inet_csk(sk);
+ struct net_namespace *orig_net_ns;
  int event;

+ orig_net_ns = push_net_ns(sk->sk_net_ns);
  bh_lock_sock(sk);
  if (sock_owned_by_user(sk)) {
   /* Try again later */
@@ -419,6 +424,7 @@ out:
 out_unlock:
  bh_unlock_sock(sk);
  sock_put(sk);
+ pop_net_ns(orig_net_ns);
 }

 /*
@@ -447,9 +453,11 @@ static void tcp_keepalive_timer (unsigne
 {
  struct sock *sk = (struct sock *) data;
  struct inet_connection_sock *icsk = inet_csk(sk);
+ struct net_namespace *orig_net_ns;
  struct tcp_sock *tp = tcp_sk(sk);
  __u32 elapsed;

+ orig_net_ns = push_net_ns(sk->sk_net_ns);
  /* Only process if socket is not in use. */
  bh_lock_sock(sk);
  if (sock_owned_by_user(sk)) {
@@ -521,4 +529,5 @@ death:
 out:
  bh_unlock_sock(sk);
  sock_put(sk);
+ put_net_ns(orig_net_ns);
 }
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/ipv4/udp.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/ipv4/udp.c
@@ -114,13 +114,15 @@ DEFINE_RWLOCK(udp_hash_lock);

 static int udp_port_rover;
```

```
-static inline int __udp_lib_lport_inuse(__u16 num, struct hlist_head udptable[])
+static inline int __udp_lib_lport_inuse(__u16 num, struct hlist_head udptable[],
+       struct net_namespace *ns)
 {
  struct sock *sk;
  struct hlist_node *node;

- sk_for_each(sk, node, &udptable[num & (UDP_HTABLE_SIZE - 1)])
-  if (inet_sk(sk)->num == num)
+ sk_for_each(sk, node, &udptable[udp_hashfn(num, ns)])
+  if (inet_sk(sk)->num == num &&
+      net_ns_match(sk->sk_net_ns, ns))
    return 1;
  return 0;
 }
@@ -142,6 +144,7 @@ int __udp_lib_get_port(struct sock *sk,
  struct hlist_node *node;
  struct hlist_head *head;
  struct sock *sk2;
+ struct net_namespace *ns = current_net_ns;
  int    error = 1;

  write_lock_bh(&udp_hash_lock);
@@ -156,7 +159,7 @@ int __udp_lib_get_port(struct sock *sk,
   for (i = 0; i < UDP_HTABLE_SIZE; i++, result++) {
    int size;

-   head = &udptable[result & (UDP_HTABLE_SIZE - 1)];
+   head = &udptable[udp_hashfn(result, ns)];
    if (hlist_empty(head)) {
     if (result > sysctl_local_port_range[1])
      result = sysctl_local_port_range[0] +
@@ -180,7 +183,7 @@ int __udp_lib_get_port(struct sock *sk,
     result = sysctl_local_port_range[0]
      + ((result - sysctl_local_port_range[0]) &
        (UDP_HTABLE_SIZE - 1));
-   if (! __udp_lib_lport_inuse(result, udptable))
+   if (! __udp_lib_lport_inuse(result, udptable, ns))
     break;
   }
   if (i >= (1 << 16) / UDP_HTABLE_SIZE)
@@ -188,11 +191,12 @@ int __udp_lib_get_port(struct sock *sk,
 gotit:
   *port_rover = snum = result;
  } else {
-  head = &udptable[snum & (UDP_HTABLE_SIZE - 1)];
+  head = &udptable[udp_hashfn(snum, ns)];
```

```
   sk_for_each(sk2, node, head)
    if (inet_sk(sk2)->num == snum                       &&
        sk2 != sk                                       &&
+       net_ns_match(sk2->sk_net_ns, ns)                &&
        (!sk2->sk_reuse        || !sk->sk_reuse)        &&
        (!sk2->sk_bound_dev_if || !sk->sk_bound_dev_if
         || sk2->sk_bound_dev_if == sk->sk_bound_dev_if) &&
@@ -201,7 +205,7 @@ gotit:
  }
  inet_sk(sk)->num = snum;
  if (sk_unhashed(sk)) {
-  head = &udptable[snum & (UDP_HTABLE_SIZE - 1)];
+  head = &udptable[udp_hashfn(snum, ns)];
    sk_add_node(sk, head);
    sock_prot_inc_use(sk->sk_prot);
  }
@@ -240,13 +244,16 @@ static struct sock *__udp4_lib_lookup(__
 {
  struct sock *sk, *result = NULL;
  struct hlist_node *node;
+ struct net_namespace *ns = current_net_ns;
  unsigned short hnum = ntohs(dport);
  int badness = -1;

  read_lock(&udp_hash_lock);
- sk_for_each(sk, node, &udptable[hnum & (UDP_HTABLE_SIZE - 1)]) {
+ sk_for_each(sk, node, &udptable[udp_hashfn(hnum, ns)]) {
    struct inet_sock *inet = inet_sk(sk);

+   if (!net_ns_match(sk->sk_net_ns, ns))
+     continue;
    if (inet->num == hnum && !ipv6_only_sock(sk)) {
      int score = (sk->sk_family == PF_INET ? 1 : 0);
      if (inet->rcv_saddr) {
@@ -291,6 +298,7 @@ static inline struct sock *udp_v4_mcast_
 {
  struct hlist_node *node;
  struct sock *s = sk;
+ struct net_namespace *ns = current_net_ns;
  unsigned short hnum = ntohs(loc_port);

  sk_for_each_from(s, node) {
@@ -301,6 +309,7 @@ static inline struct sock *udp_v4_mcast_
        (inet->dport != rmt_port && inet->dport)  ||
        (inet->rcv_saddr && inet->rcv_saddr != loc_addr) ||
        ipv6_only_sock(s)     ||
+       !net_ns_match(sk->sk_net_ns, ns)   ||
        (s->sk_bound_dev_if && s->sk_bound_dev_if != dif))
```

```
    continue;
   if (!ip_mc_sf_allow(s, loc_addr, rmt_addr, dif))
@@ -1131,7 +1140,7 @@ static int __udp4_lib_mcast_deliver(stru
  int dif;

  read_lock(&udp_hash_lock);
- sk = sk_head(&udptable[ntohs(uh->dest) & (UDP_HTABLE_SIZE - 1)]);
+ sk = sk_head(&udptable[udp_hashfn(ntohs(uh->dest), current_net_ns)]);
  dif = skb->dev->ifindex;
  sk = udp_v4_mcast_next(sk, uh->dest, daddr, uh->source, saddr, dif);
  if (sk) {
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/ipv6/inet6_connection_sock.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/ipv6/inet6_connection_sock.c
@@ -31,10 +31,12 @@ int inet6_csk_bind_conflict(const struct
 {
  const struct sock *sk2;
  const struct hlist_node *node;
+ struct net_namespace *ns = current_net_ns;

  /* We must walk the whole port owner list in this case. -DaveM */
  sk_for_each_bound(sk2, node, &tb->owners) {
   if (sk != sk2 &&
+      net_ns_match(sk2->sk_net_ns, ns) &&
       (!sk->sk_bound_dev_if ||
        !sk2->sk_bound_dev_if ||
        sk->sk_bound_dev_if == sk2->sk_bound_dev_if) &&
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/ipv6/inet6_hashtables.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/ipv6/inet6_hashtables.c
@@ -65,17 +65,18 @@ struct sock *__inet6_lookup_established(
  struct sock *sk;
  const struct hlist_node *node;
  const __portpair ports = INET_COMBINED_PORTS(sport, hnum);
+ struct net_namespace *ns = current_net_ns;
  /* Optimize here for direct hit, only listening connections can
   * have wildcards anyways.
   */
- unsigned int hash = inet6_ehashfn(daddr, hnum, saddr, sport);
+ unsigned int hash = inet6_ehashfn(daddr, hnum, saddr, sport, ns);
  struct inet_ehash_bucket *head = inet_ehash_bucket(hashinfo, hash);

  prefetch(head->chain.first);
  read_lock(&head->lock);
  sk_for_each(sk, node, &head->chain) {
   /* For IPV6 do the cheaper port and family tests first. */
-  if (INET6_MATCH(sk, hash, saddr, daddr, ports, dif))
+  if (INET6_MATCH(sk, hash, saddr, daddr, ports, dif, ns))
    goto hit; /* You sunk my battleship! */
  }
```

```
  /* Must check for a TIME_WAIT'er before going to listener hash. */
@@ -83,6 +84,7 @@ struct sock *__inet6_lookup_established(
    const struct inet_timewait_sock *tw = inet_twsk(sk);

    if(*((__portpair *)&(tw->tw_dport)) == ports &&
+      net_ns_match(sk->sk_net_ns, ns)   &&
       sk->sk_family  == PF_INET6) {
     const struct inet6_timewait_sock *tw6 = inet6_twsk(sk);

@@ -107,12 +109,15 @@ struct sock *inet6_lookup_listener(struc
         const unsigned short hnum, const int dif)
 {
  struct sock *sk;
+ struct net_namespace *ns = current_net_ns;
  const struct hlist_node *node;
  struct sock *result = NULL;
  int score, hiscore = 0;

  read_lock(&hashinfo->lhash_lock);
- sk_for_each(sk, node, &hashinfo->listening_hash[inet_lhashfn(hnum)]) {
+ sk_for_each(sk, node, &hashinfo->listening_hash[inet_lhashfn(hnum, ns)]) {
+  if (!net_ns_match(sk->sk_net_ns, ns))
+   continue;
    if (inet_sk(sk)->num == hnum && sk->sk_family == PF_INET6) {
     const struct ipv6_pinfo *np = inet6_sk(sk);

@@ -172,8 +177,9 @@ static int __inet6_check_established(str
  const struct in6_addr *saddr = &np->daddr;
  const int dif = sk->sk_bound_dev_if;
  const __portpair ports = INET_COMBINED_PORTS(inet->dport, lport);
+ struct net_namespace *ns = current_net_ns;
  const unsigned int hash = inet6_ehashfn(daddr, inet->num, saddr,
-      inet->dport);
+       inet->dport, ns);
  struct inet_ehash_bucket *head = inet_ehash_bucket(hinfo, hash);
  struct sock *sk2;
  const struct hlist_node *node;
@@ -190,6 +196,7 @@ static int __inet6_check_established(str

    if(*((__portpair *)&(tw->tw_dport)) == ports    &&
       sk2->sk_family         == PF_INET6  &&
+      net_ns_match(sk2->sk_net_ns, ns)   &&
       ipv6_addr_equal(&tw6->tw_v6_daddr, saddr)  &&
       ipv6_addr_equal(&tw6->tw_v6_rcv_saddr, daddr) &&
       sk2->sk_bound_dev_if == sk->sk_bound_dev_if) {
@@ -203,7 +210,7 @@ static int __inet6_check_established(str

  /* And established part... */
```

```
  sk_for_each(sk2, node, &head->chain) {
-  if (INET6_MATCH(sk2, hash, saddr, daddr, ports, dif))
+  if (INET6_MATCH(sk2, hash, saddr, daddr, ports, dif, ns))
    goto not_unique;
  }

@@ -249,6 +256,7 @@ int inet6_hash_connect(struct inet_timew
 {
  struct inet_hashinfo *hinfo = death_row->hashinfo;
  const unsigned short snum = inet_sk(sk)->num;
+ struct net_namespace *ns = current_net_ns;
   struct inet_bind_hashbucket *head;
   struct inet_bind_bucket *tb;
  int ret;
@@ -266,7 +274,8 @@ int inet6_hash_connect(struct inet_timew
    local_bh_disable();
   for (i = 1; i <= range; i++) {
    port = low + (i + offset) % range;
-    head = &hinfo->bhash[inet_bhashfn(port, hinfo->bhash_size)];
+    head = &hinfo->bhash[inet_bhashfn(port, ns,
+      hinfo->bhash_size)];
     spin_lock(&head->lock);

     /* Does not bother with rcv_saddr checks,
@@ -274,6 +283,8 @@ int inet6_hash_connect(struct inet_timew
      * unique enough.
      */
    inet_bind_bucket_for_each(tb, node, &head->chain) {
+    if (!net_ns_match(tb->net_ns, ns))
+     continue;
      if (tb->port == port) {
       BUG_TRAP(!hlist_empty(&tb->owners));
       if (tb->fastreuse >= 0)
@@ -322,7 +333,7 @@ ok:
   goto out;
   }

-  head = &hinfo->bhash[inet_bhashfn(snum, hinfo->bhash_size)];
+  head = &hinfo->bhash[inet_bhashfn(snum, ns, hinfo->bhash_size)];
   tb   = inet_csk(sk)->icsk_bind_hash;
  spin_lock_bh(&head->lock);

--- linux-2.6.20-rc4-mm1.net_ns.orig/net/ipv6/raw.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/ipv6/raw.c
@@ -89,11 +89,15 @@ struct sock *__raw_v6_lookup(struct sock
 {
  struct hlist_node *node;
  int is_multicast = ipv6_addr_is_multicast(loc_addr);
```

```
+ struct net_namespace *ns = current_net_ns;

  sk_for_each_from(sk, node)
   if (inet_sk(sk)->num == num) {
    struct ipv6_pinfo *np = inet6_sk(sk);

+   if (!net_ns_match(sk->sk_net_ns, ns))
+    continue;
+
    if (!ipv6_addr_any(&np->daddr) &&
        !ipv6_addr_equal(&np->daddr, rmt_addr))
     continue;
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/ipv6/udp.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/ipv6/udp.c
@@ -63,13 +63,16 @@ static struct sock *__udp6_lib_lookup(st
 {
  struct sock *sk, *result = NULL;
  struct hlist_node *node;
+ struct net_namespace *ns = current_net_ns;
  unsigned short hnum = ntohs(dport);
  int badness = -1;

   read_lock(&udp_hash_lock);
- sk_for_each(sk, node, &udptable[hnum & (UDP_HTABLE_SIZE - 1)]) {
+ sk_for_each(sk, node, &udptable[udp_hashfn(hnum, ns)]) {
   struct inet_sock *inet = inet_sk(sk);

+  if (!net_ns_match(sk->sk_net_ns, ns))
+   continue;
   if (inet->num == hnum && sk->sk_family == PF_INET6) {
    struct ipv6_pinfo *np = inet6_sk(sk);
    int score = 0;
@@ -303,6 +306,7 @@ static struct sock *udp_v6_mcast_next(st
 {
  struct hlist_node *node;
  struct sock *s = sk;
+ struct net_namespace *ns = current_net_ns;
  unsigned short num = ntohs(loc_port);

  sk_for_each_from(s, node) {
@@ -314,6 +318,8 @@ static struct sock *udp_v6_mcast_next(st
     if (inet->dport != rmt_port)
      continue;
    }
+   if (!net_ns_match(sk->sk_net_ns, ns))
+    continue;
    if (!ipv6_addr_any(&np->daddr) &&
        !ipv6_addr_equal(&np->daddr, rmt_addr))
```

```
     continue;
@@ -345,7 +351,7 @@ static int __udp6_lib_mcast_deliver(stru
  int dif;

  read_lock(&udp_hash_lock);
- sk = sk_head(&udptable[ntohs(uh->dest) & (UDP_HTABLE_SIZE - 1)]);
+ sk = sk_head(&udptable[udp_hashfn(uh->dest, current_net_ns)]);
  dif = inet6_iif(skb);
  sk = udp_v6_mcast_next(sk, uh->dest, daddr, uh->source, saddr, dif);
  if (!sk) {
```

_____

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

Subject: [PATCH 7/12] allow proc_dir_entries to have destructor
Posted by Mishin Dmitry on Wed, 17 Jan 2007 16:10:51 GMT
View Forum Message <> Reply to Message

Destructor field added proc_dir_entries,
standard destructor kfree'ing data introduced.

Signed-off-by: Andrey Savochkin <saw@swsoft.com>

```
---
 fs/proc/generic.c       |   10 ++++++++--
 fs/proc/root.c          |    1 +
 include/linux/proc_fs.h |    4 ++++
 3 files changed, 13 insertions(+), 2 deletions(-)

Index: 2.6.20-rc4-mm1/fs/proc/generic.c
===================================================================
--- 2.6.20-rc4-mm1.orig/fs/proc/generic.c
+++ 2.6.20-rc4-mm1/fs/proc/generic.c
@@ -611,6 +611,11 @@ static struct proc_dir_entry *proc_creat
  return ent;
 }

+void proc_data_destructor(struct proc_dir_entry *ent)
+{
+ kfree(ent->data);
+}
+
 struct proc_dir_entry *proc_symlink(const char *name,
   struct proc_dir_entry *parent, const char *dest)
 {
@@ -623,6 +628,7 @@ struct proc_dir_entry *proc_symlink(cons
```

```
   ent->data = kmalloc((ent->size=strlen(dest))+1, GFP_KERNEL);
   if (ent->data) {
    strcpy((char*)ent->data,dest);
+   ent->destructor = proc_data_destructor;
    if (proc_register(parent, ent) < 0) {
     kfree(ent->data);
     kfree(ent);
@@ -701,8 +707,8 @@ void free_proc_entry(struct proc_dir_ent

  release_inode_number(ino);

- if (S_ISLNK(de->mode) && de->data)
-  kfree(de->data);
+ if (de->destructor)
+  de->destructor(de);
  kfree(de);
 }

Index: 2.6.20-rc4-mm1/fs/proc/root.c
===================================================================
--- 2.6.20-rc4-mm1.orig/fs/proc/root.c
+++ 2.6.20-rc4-mm1/fs/proc/root.c
@@ -167,6 +167,7 @@ EXPORT_SYMBOL(proc_symlink);
 EXPORT_SYMBOL(proc_mkdir);
 EXPORT_SYMBOL(create_proc_entry);
 EXPORT_SYMBOL(remove_proc_entry);
+EXPORT_SYMBOL(proc_data_destructor);
 EXPORT_SYMBOL(proc_root);
 EXPORT_SYMBOL(proc_root_fs);
 EXPORT_SYMBOL(proc_net);
Index: 2.6.20-rc4-mm1/include/linux/proc_fs.h
===================================================================
--- 2.6.20-rc4-mm1.orig/include/linux/proc_fs.h
+++ 2.6.20-rc4-mm1/include/linux/proc_fs.h
@@ -45,6 +45,8 @@ typedef int (read_proc_t)(char *page, ch
 typedef int (write_proc_t)(struct file *file, const char __user *buffer,
       unsigned long count, void *data);
 typedef int (get_info_t)(char *, char **, off_t, int);
+struct proc_dir_entry;
+typedef void (destroy_proc_t)(struct proc_dir_entry *);

 struct proc_dir_entry {
  unsigned int low_ino;
@@ -64,6 +66,7 @@ struct proc_dir_entry {
  read_proc_t *read_proc;
  write_proc_t *write_proc;
  atomic_t count;  /* use count */
+ destroy_proc_t *destructor;
```

```
 int deleted;  /* delete flag */
 void *set;
};
@@ -108,6 +111,7 @@ char *task_mem(struct mm_struct *, char
 extern struct proc_dir_entry *create_proc_entry(const char *name, mode_t mode,
       struct proc_dir_entry *parent);
 extern void remove_proc_entry(const char *name, struct proc_dir_entry *parent);
+extern void proc_data_destructor(struct proc_dir_entry *);

 extern struct vfsmount *proc_mnt;
 extern int proc_fill_super(struct super_block *,void *,int);
```

_____

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

Subject: [PATCH 8/12] net_device seq_file

Posted by Mishin Dmitry on Wed, 17 Jan 2007 16:11:52 GMT
View Forum Message <> Reply to Message

Library function to create a seq_file in proc filesystem,
showing some information for each netdevice.
This code is present in the kernel in about 10 instances, and
all of them can be converted to using introduced library function.

Signed-off-by: Andrey Savochkin <saw@swsoft.com>

```
---
 include/linux/netdevice.h |    7 +++
 net/core/dev.c            |   96 ++++++++++++++++++++++++++++++++++++++++++++++
 2 files changed, 103 insertions(+)

--- linux-2.6.20-rc4-mm1.net_ns.orig/include/linux/netdevice.h
+++ linux-2.6.20-rc4-mm1.net_ns/include/linux/netdevice.h
@@ -604,6 +604,13 @@ extern int  register_netdevice(struct ne
 extern int  unregister_netdevice(struct net_device *dev);
 extern void  free_netdev(struct net_device *dev);
 extern void  synchronize_net(void);
+#ifdef CONFIG_PROC_FS
+extern int  netdev_proc_create(char *name,
+    int (*show)(struct seq_file *,
+     struct net_device *, void *),
+    void *data, struct module *mod);
+void   netdev_proc_remove(char *name);
+#endif
 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);
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/core/dev.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/core/dev.c
@@ -2099,6 +2099,102 @@ static int dev_ifconf(char __user *arg)
 }

 #ifdef CONFIG_PROC_FS
+
+struct netdev_proc_data {
+ struct file_operations fops;
+ int (*show)(struct seq_file *, struct net_device *, void *);
+ void *data;
+};
+
+static void *netdev_proc_seq_start(struct seq_file *seq, loff_t *pos)
+{
+ struct net_device *dev;
+ loff_t off;
+
+ read_lock(&dev_base_lock);
+ if (*pos == 0)
+  return SEQ_START_TOKEN;
+ for (dev = dev_base, off = 1; dev; dev = dev->next, off++) {
+  if (*pos == off)
+   return dev;
+ }
+ return NULL;
+}
+
+static void *netdev_proc_seq_next(struct seq_file *seq, void *v, loff_t *pos)
+{
+ ++*pos;
+ return (v == SEQ_START_TOKEN) ? dev_base
+     : ((struct net_device *)v)->next;
+}
+
+static void netdev_proc_seq_stop(struct seq_file *seq, void *v)
+{
+ read_unlock(&dev_base_lock);
+}
+
+static int netdev_proc_seq_show(struct seq_file *seq, void *v)
+{
+ struct netdev_proc_data *p;
+
+ p = seq->private;
+ return (*p->show)(seq, v, p->data);
+}
```

```
+
+static struct seq_operations netdev_proc_seq_ops = {
+ .start = netdev_proc_seq_start,
+ .next  = netdev_proc_seq_next,
+ .stop  = netdev_proc_seq_stop,
+ .show  = netdev_proc_seq_show,
+};
+
+static int netdev_proc_open(struct inode *inode, struct file *file)
+{
+ int err;
+ struct seq_file *p;
+
+ err = seq_open(file, &netdev_proc_seq_ops);
+ if (!err) {
+  p = file->private_data;
+  p->private = (struct netdev_proc_data *)PDE(inode)->data;
+ }
+ return err;
+}
+
+int netdev_proc_create(char *name,
+  int (*show)(struct seq_file *, struct net_device *, void *),
+  void *data, struct module *mod)
+{
+ struct netdev_proc_data *p;
+ struct proc_dir_entry *ent;
+
+ p = kzalloc(sizeof(*p), GFP_KERNEL);
+ p->fops.owner = mod;
+ p->fops.open = netdev_proc_open;
+ p->fops.read = seq_read;
+ p->fops.llseek = seq_lseek;
+ p->fops.release = seq_release;
+ p->show = show;
+ p->data = data;
+ ent = create_proc_entry(name, S_IRUGO, proc_net);
+ if (ent == NULL) {
+  kfree(p);
+  return -EINVAL;
+ }
+ ent->data = p;
+ ent->destructor = proc_data_destructor;
+ smp_wmb();
+ ent->proc_fops = &p->fops;
+ return 0;
+}
+EXPORT_SYMBOL(netdev_proc_create);
```

```
+
+void netdev_proc_remove(char *name)
+{
+ proc_net_remove(name);
+}
+EXPORT_SYMBOL(netdev_proc_remove);
+
 /*
  * This is invoked by the /proc filesystem handler to display a device
  * in detail.
```

_____

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

Subject: [PATCH 9/12] L2 network namespace (v3): device to pass packets between namespaces

Posted by Mishin Dmitry on Wed, 17 Jan 2007 16:14:00 GMT
View Forum Message <> Reply to Message

A simple device to pass packets between a namespace and its child.

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

```
---
 drivers/net/Makefile     |   3
 drivers/net/veth.c       | 321 ++++++++++++++++++++++++++++++++++++++++++++++
 net/core/net_namespace.c |   1
 3 files changed, 325 insertions(+)

--- linux-2.6.20-rc4-mm1.net_ns.orig/drivers/net/Makefile
+++ linux-2.6.20-rc4-mm1.net_ns/drivers/net/Makefile
@@ -125,6 +125,9 @@ obj-$(CONFIG_SLIP) += slip.o
 obj-$(CONFIG_SLHC) += slhc.o

 obj-$(CONFIG_DUMMY) += dummy.o
+ifeq ($(CONFIG_NET_NS),y)
+obj-m += veth.o
+endif
 obj-$(CONFIG_IFB) += ifb.o
 obj-$(CONFIG_DE600) += de600.o
 obj-$(CONFIG_DE620) += de620.o
--- /dev/null
+++ linux-2.6.20-rc4-mm1.net_ns/drivers/net/veth.c
@@ -0,0 +1,321 @@
+/*
+ * Copyright (C) 2006  SWsoft
```

```
+ *
+ * Written by Andrey Savochkin <saw@sw.ru>,
+ * reusing code by Andrey Mirkin <amirkin@sw.ru>.
+ */
+#include <linux/list.h>
+#include <linux/spinlock.h>
+#include <linux/ctype.h>
+#include <asm/semaphore.h>
+#include <linux/netdevice.h>
+#include <linux/etherdevice.h>
+#include <linux/proc_fs.h>
+#include <linux/seq_file.h>
+#include <net/dst.h>
+#include <net/xfrm.h>
+
+struct veth_struct
+{
+ struct net_device *pair;
+ struct net_device_stats stats;
+};
+
+#define veth_from_netdev(dev) ((struct veth_struct *)(netdev_priv(dev)))
+
+/* ---------------------------------------------------------------- *
+ *
+ * Device functions
+ *
+ * ---------------------------------------------------------------- */
+
+static struct net_device_stats *get_stats(struct net_device *dev);
+static int veth_xmit(struct sk_buff *skb, struct net_device *dev)
+{
+ struct net_device_stats *stats;
+ struct veth_struct *entry;
+ struct net_device *rcv;
+ struct net_namespace *orig_net_ns;
+ int length;
+
+ stats = get_stats(dev);
+ entry = veth_from_netdev(dev);
+ rcv = entry->pair;
+
+ if (!(rcv->flags & IFF_UP))
+  /* Target namespace does not want to receive packets */
+  goto outf;
+
+ dst_release(skb->dst);
+ skb->dst = NULL;
```

```
+ secpath_reset(skb);
+ skb_orphan(skb);
+ nf_reset(skb);
+
+ orig_net_ns = push_net_ns(rcv->net_ns);
+ skb->dev = rcv;
+ skb->pkt_type = PACKET_HOST;
+ skb->protocol = eth_type_trans(skb, rcv);
+
+ length = skb->len;
+ stats->tx_bytes += length;
+ stats->tx_packets++;
+ stats = get_stats(rcv);
+ stats->rx_bytes += length;
+ stats->rx_packets++;
+
+ netif_rx(skb);
+ pop_net_ns(orig_net_ns);
+ return 0;
+
+outf:
+ stats->tx_dropped++;
+ kfree_skb(skb);
+ return 0;
+}
+
+static int veth_open(struct net_device *dev)
+{
+ return 0;
+}
+
+static int veth_close(struct net_device *dev)
+{
+ return 0;
+}
+
+static void veth_destructor(struct net_device *dev)
+{
+ free_netdev(dev);
+}
+
+static struct net_device_stats *get_stats(struct net_device *dev)
+{
+ return &veth_from_netdev(dev)->stats;
+}
+
+int veth_init_dev(struct net_device *dev)
+{
```

```
+ dev->hard_start_xmit = veth_xmit;
+ dev->open = veth_open;
+ dev->stop = veth_close;
+ dev->destructor = veth_destructor;
+ dev->get_stats = get_stats;
+
+ ether_setup(dev);
+
+ dev->tx_queue_len = 0;
+ return 0;
+}
+
+static void veth_setup(struct net_device *dev)
+{
+ dev->init = veth_init_dev;
+}
+
+static inline int is_veth_dev(struct net_device *dev)
+{
+ return dev->init == veth_init_dev;
+}
+
+/* ------------------------------------------------------------------ *
+ *
+ * Management interface
+ *
+ * ------------------------------------------------------------------ */
+
+struct net_device *veth_dev_alloc(char *name, char *addr)
+{
+ struct net_device *dev;
+
+ dev = alloc_netdev(sizeof(struct veth_struct), name, veth_setup);
+ if (dev != NULL) {
+  memcpy(dev->dev_addr, addr, ETH_ALEN);
+  dev->addr_len = ETH_ALEN;
+ }
+ return dev;
+}
+
+int veth_entry_add(char *parent_name, char *parent_addr,
+ struct net_namespace *parent_ns, char *child_name, char *child_addr,
+       struct net_namespace *child_ns)
+{
+ struct net_device *parent_dev, *child_dev;
+ int err;
+
+ err = -ENOMEM;
```

```
+ if ((parent_dev = veth_dev_alloc(parent_name, parent_addr)) == NULL)
+  goto out_alocp;
+ if ((child_dev = veth_dev_alloc(child_name, child_addr)) == NULL)
+  goto out_alocc;
+ veth_from_netdev(parent_dev)->pair = child_dev;
+ veth_from_netdev(child_dev)->pair = parent_dev;
+
+ /*
+  * About serialization, see comments to veth_pair_del().
+  */
+ rtnl_lock();
+ /* refcounts should be already upped, so, just put old ones */
+ put_net_ns(parent_dev->net_ns);
+ parent_dev->net_ns = parent_ns;
+ if ((err = register_netdevice(parent_dev)))
+  goto out_regp;
+
+ put_net_ns(child_dev->net_ns);
+ child_dev->net_ns = child_ns;
+ if ((err = register_netdevice(child_dev)))
+  goto out_regc;
+ rtnl_unlock();
+ return 0;
+
+out_regc:
+ unregister_netdevice(parent_dev);
+ rtnl_unlock();
+ free_netdev(child_dev);
+ return err;
+
+out_regp:
+ rtnl_unlock();
+ free_netdev(child_dev);
+out_alocc:
+ free_netdev(parent_dev);
+out_alocp:
+ return err;
+}
+
+static void veth_pair_del(struct net_device *parent_dev)
+{
+ struct net_device *child_dev;
+ struct net_namespace *parent_ns, *child_ns;
+
+ child_dev = veth_from_netdev(parent_dev)->pair;
+ get_net_ns(child_dev->net_ns);
+ child_ns = child_dev->net_ns;
+
```

```
+ dev_close(child_dev);
+ synchronize_net();
+ /*
+  * Now child_dev does not send or receives anything.
+  * This means child_dev->hard_start_xmit is not called anymore.
+  */
+ unregister_netdevice(parent_dev);
+ /*
+  * At this point child_dev has dead pointer to parent_dev.
+  * But this pointer is not dereferenced.
+  */
+ parent_ns = push_net_ns(child_ns);
+ unregister_netdevice(child_dev);
+ pop_net_ns(parent_ns);
+
+ put_net_ns(child_ns);
+}
+
+int veth_entry_del(char *parent_name)
+{
+ struct net_device *dev;
+
+ if ((dev = dev_get_by_name(parent_name)) == NULL)
+  return -ENODEV;
+
+ rtnl_lock();
+ veth_pair_del(dev);
+ dev_put(dev);
+ rtnl_unlock();
+
+ return 0;
+}
+
+void veth_entry_del_all(void)
+{
+ struct net_device **p, *dev;
+
+ rtnl_lock();
+ for (p = &dev_base; (dev = *p) != NULL; ) {
+  if (!is_veth_dev(dev)) {
+   p = &dev->next;
+   continue;
+  }
+
+  dev_hold(dev);
+  veth_pair_del(dev);
+  dev_put(dev);
+ }
```

```
+ rtnl_unlock();
+}
+
+/* ---------------------------------------------------------------- *
+ *
+ * Information in proc
+ *
+ * ---------------------------------------------------------------- */
+
+#ifdef CONFIG_PROC_FS
+
+#define ADDR_FMT "%02x:%02x:%02x:%02x:%02x:%02x"
+#define ADDR(x) (x)[0],(x)[1],(x)[2],(x)[3],(x)[4],(x)[5]
+#define ADDR_HDR "%-17s"
+
+static int veth_proc_show(struct seq_file *m,
+  struct net_device *dev, void *data)
+{
+ struct net_device *pair;
+
+ if (dev == SEQ_START_TOKEN) {
+  seq_puts(m, "Version: 1.0\n");
+  seq_printf(m, "%-*s " ADDR_HDR " %-*s " ADDR_HDR "\n",
+    IFNAMSIZ, "Name", "Address",
+    IFNAMSIZ, "PeerName", "PeerAddress");
+  return 0;
+ }
+
+ if (!is_veth_dev(dev))
+  return 0;
+
+ pair = veth_from_netdev(dev)->pair;
+ seq_printf(m, "%-*s " ADDR_FMT " %-*s " ADDR_FMT "\n",
+   IFNAMSIZ, dev->name, ADDR(dev->dev_addr),
+   IFNAMSIZ, pair->name, ADDR(pair->dev_addr));
+ return 0;
+}
+
+static int veth_proc_create(void)
+{
+ return netdev_proc_create("veth_list", &veth_proc_show, NULL,
+   THIS_MODULE);
+}
+
+static void veth_proc_remove(void)
+{
+ netdev_proc_remove("net/veth_list");
+}
```

```
+
+#else
+
+static inline int veth_proc_create(void) { return 0; }
+static inline void veth_proc_remove(void) { }
+
+#endif
+
+/* ---------------------------------------------------------------- *
+ *
+ * Module initialization
+ *
+ * ---------------------------------------------------------------- */
+
+int __init veth_init(void)
+{
+ veth_proc_create();
+ return 0;
+}
+
+void __exit veth_exit(void)
+{
+ veth_proc_remove();
+ veth_entry_del_all();
+}
+
+module_init(veth_init)
+module_exit(veth_exit)
+
+MODULE_DESCRIPTION("Virtual Ethernet Device");
+MODULE_LICENSE("GPL v2");
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/core/net_namespace.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/core/net_namespace.c
@@ -111,5 +111,6 @@ void free_net_ns(struct kref *kref)
  ip_fib_struct_cleanup(ns);
  kfree(ns);
 }
+EXPORT_SYMBOL_GPL(free_net_ns);

 #endif /* CONFIG_NET_NS */
```

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

device
Posted by Mishin Dmitry on Wed, 17 Jan 2007 16:15:15 GMT
View Forum Message <> Reply to Message

Temporary code to debug and play with pass-through device.
Create device pair by
 modprobe veth
	echo 'add veth1 0:1:2:3:4:1 eth0 0:1:2:3:4:2' >/proc/net/veth_ctl
and your shell will appear into a new namespace with `eth0' device.
Configure device in this namespace
	ip l s eth0 up
	ip a a 1.2.3.4/24 dev eth0
and in the root namespace
	ip l s veth1 up
	ip a a 1.2.3.1/24 dev veth1
to establish a communication channel between root namespace and the newly
created one.

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

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

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

--- linux-2.6.20-rc4-mm1.net_ns.orig/drivers/net/veth.c
+++ linux-2.6.20-rc4-mm1.net_ns/drivers/net/veth.c
@@ -12,6 +12,7 @@
 #include <linux/etherdevice.h>
 #include <linux/proc_fs.h>
 #include <linux/seq_file.h>
+#include <linux/syscalls.h>
 #include <net/dst.h>
 #include <net/xfrm.h>

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

 /* ----------------------------------------------------------------- *
  *
+ * Temporary interface to create veth devices
+ *
+ * ----------------------------------------------------------------- */
+
+#ifdef CONFIG_PROC_FS
```

```
+
+static int veth_debug_open(struct inode *inode, struct file *file)
+{
+ return 0;
+}
+
+static char *parse_addr(char *s, char *addr)
+{
+ int i, v;
+
+ for (i = 0; i < ETH_ALEN; i++) {
+  if (!isxdigit(*s))
+   return NULL;
+  *addr = 0;
+  v = isdigit(*s) ? *s - '0' : toupper(*s) - 'A' + 10;
+  s++;
+  if (isxdigit(*s)) {
+   *addr += v << 16;
+   v = isdigit(*s) ? *s - '0' : toupper(*s) - 'A' + 10;
+   s++;
+  }
+  *addr++ += v;
+  if (i < ETH_ALEN - 1 && ispunct(*s))
+   s++;
+ }
+ return s;
+}
+
+static ssize_t veth_debug_write(struct file *file, const char __user *user_buf,
+  size_t size, loff_t *ppos)
+{
+ char buf[128], *s, *parent_name, *child_name;
+ char parent_addr[ETH_ALEN], child_addr[ETH_ALEN];
+ struct net_namespace *parent_ns, *child_ns;
+ int err;
+
+ s = buf;
+ err = -EINVAL;
+ if (size >= sizeof(buf))
+  goto out;
+ err = -EFAULT;
+ if (copy_from_user(buf, user_buf, size))
+  goto out;
+ buf[size] = 0;
+
+ err = -EBADRQC;
+ if (!strncmp(buf, "add ", 4)) {
+  parent_name = buf + 4;
```

```
+  if ((s = strchr(parent_name, ' ')) == NULL)
+   goto out;
+  *s = 0;
+  if ((s = parse_addr(s + 1, parent_addr)) == NULL)
+   goto out;
+  if (!*s)
+   goto out;
+  child_name = s + 1;
+  if ((s = strchr(child_name, ' ')) == NULL)
+   goto out;
+  *s = 0;
+  if ((s = parse_addr(s + 1, child_addr)) == NULL)
+   goto out;
+
+  get_net_ns(current_net_ns);
+  parent_ns = current_net_ns;
+  if (*s == ' ') {
+   unsigned int id;
+   id = simple_strtoul(s + 1, &s, 0);
+   err = sys_bind_ns(id, NS_ALL);
+  } else
+   err = sys_unshare(CLONE_NEWNET2);
+  if (err)
+   goto out;
+  /* after bind_ns() or unshare_ns() namespace is changed */
+  get_net_ns(current_net_ns);
+  child_ns = current_net_ns;
+  err = veth_entry_add(parent_name, parent_addr, parent_ns,
+   child_name, child_addr, child_ns);
+  if (err) {
+   put_net_ns(child_ns);
+   put_net_ns(parent_ns);
+  } else
+   err = size;
+ }
+out:
+ return err;
+}
+
+static struct file_operations veth_debug_ops = {
+ .open = &veth_debug_open,
+ .write = &veth_debug_write,
+};
+
+static int veth_debug_create(void)
+{
+ proc_net_fops_create("veth_ctl", 0200, &veth_debug_ops);
+ return 0;
```

```
+}
+
+static void veth_debug_remove(void)
+{
+ proc_net_remove("veth_ctl");
+}
+
+#else
+
+static int veth_debug_create(void) { return -1; }
+static void veth_debug_remove(void) { }
+
+#endif
+
+/* ---------------------------------------------------------------- *
+ *
  * Information in proc
  *
  * ---------------------------------------------------------------- */
@@ -304,12 +422,15 @@ static inline void veth_proc_remove(void

 int __init veth_init(void)
 {
+ if (veth_debug_create())
+  return -EINVAL;
  veth_proc_create();
  return 0;
 }

 void __exit veth_exit(void)
 {
+ veth_debug_remove();
  veth_proc_remove();
  veth_entry_del_all();
 }
--- linux-2.6.20-rc4-mm1.net_ns.orig/fs/proc/array.c
+++ linux-2.6.20-rc4-mm1.net_ns/fs/proc/array.c
@@ -72,6 +72,7 @@
 #include <linux/highmem.h>
 #include <linux/file.h>
 #include <linux/times.h>
+#include <linux/net_namespace.h>
 #include <linux/cpuset.h>
 #include <linux/rcupdate.h>
 #include <linux/delayacct.h>
@@ -198,6 +199,13 @@ static inline char * task_state(struct t
  put_group_info(group_info);
```

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

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

 static int __init nshash_init(void)
 {
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/core/net_namespace.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/core/net_namespace.c
@@ -56,6 +56,8 @@ static struct net_namespace *clone_net_n
  if (loopback_init())
   goto out_loopback;
  pop_net_ns(old_ns);
+ printk(KERN_DEBUG "NET_NS: created new netcontext %p for %s "
+  "(pid=%d)\n", ns, current->comm, current->tgid);
  return ns;

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

Subject: [PATCH 11/12] L2 network namespace (v3): sockets proc view virtualization
Posted by Mishin Dmitry on Wed, 17 Jan 2007 16:16:44 GMT
View Forum Message <> Reply to Message

Only current net namespace sockets or all sockets in case of init_net_ns should
be visible through proc interface.

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

```
---
 include/net/af_unix.h |   21 ++++++++++++++++++----
 net/ipv4/tcp_ipv4.c   |    9 +++++++++
 net/ipv4/udp.c        |   13 ++++++++++--
 3 files changed, 37 insertions(+), 6 deletions(-)

--- linux-2.6.20-rc4-mm1.net_ns.orig/include/net/af_unix.h
+++ linux-2.6.20-rc4-mm1.net_ns/include/net/af_unix.h
@@ -19,9 +19,13 @@ extern atomic_t unix_tot_inflight;

 static inline struct sock *first_unix_socket(int *i)
 {
+ struct sock *sk;
+
  for (*i = 0; *i <= UNIX_HASH_SIZE; (*i)++) {
-  if (!hlist_empty(&unix_socket_table[*i]))
-   return __sk_head(&unix_socket_table[*i]);
+  for (sk = sk_head(&unix_socket_table[*i]); sk; sk = sk_next(sk))
+   if (net_ns_match(sk->sk_net_ns, current_net_ns) ||
+    net_ns_match(current_net_ns, &init_net_ns))
+    return sk;
  }
  return NULL;
 }
@@ -32,10 +36,19 @@ static inline struct sock *next_unix_soc
  /* More in this chain? */
  if (next)
   return next;
+ for (; next != NULL; next = sk_next(next)) {
+  if (!net_ns_match(next->sk_net_ns, current_net_ns) &&
+   !net_ns_match(current_net_ns, &init_net_ns))
+   continue;
+  return next;
```

```
+ }
  /* Look for next non-empty chain. */
  for ((*i)++; *i <= UNIX_HASH_SIZE; (*i)++) {
-  if (!hlist_empty(&unix_socket_table[*i]))
-   return __sk_head(&unix_socket_table[*i]);
+  for (next = sk_head(&unix_socket_table[*i]); next;
+       next = sk_next(next))
+   if (net_ns_match(next->sk_net_ns, current_net_ns) ||
+    net_ns_match(current_net_ns, &init_net_ns))
+    return next;
  }
  return NULL;
 }
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/ipv4/tcp_ipv4.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/ipv4/tcp_ipv4.c
@@ -1992,6 +1992,9 @@ get_req:
  }
 get_sk:
  sk_for_each_from(sk, node) {
+  if (!net_ns_match(sk->sk_net_ns, current_net_ns) &&
+   !net_ns_match(current_net_ns, &init_net_ns))
+   continue;
   if (sk->sk_family == st->family) {
    cur = sk;
    goto out;
@@ -2043,6 +2046,9 @@ static void *established_get_first(struc

   read_lock(&tcp_hashinfo.ehash[st->bucket].lock);
   sk_for_each(sk, node, &tcp_hashinfo.ehash[st->bucket].chain) {
+   if (!net_ns_match(sk->sk_net_ns, current_net_ns) &&
+    !net_ns_match(current_net_ns, &init_net_ns))
+    continue;
    if (sk->sk_family != st->family) {
     continue;
    }
@@ -2102,6 +2108,9 @@ get_tw:
   sk = sk_next(sk);

  sk_for_each_from(sk, node) {
+  if (!net_ns_match(sk->sk_net_ns, current_net_ns) &&
+   !net_ns_match(current_net_ns, &init_net_ns))
+   continue;
   if (sk->sk_family == st->family)
    goto found;
  }
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/ipv4/udp.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/ipv4/udp.c
@@ -1549,6 +1549,9 @@ static struct sock *udp_get_first(struct
```

```
  for (state->bucket = 0; state->bucket < UDP_HTABLE_SIZE; ++state->bucket) {
   struct hlist_node *node;
   sk_for_each(sk, node, state->hashtable + state->bucket) {
+    if (!net_ns_match(sk->sk_net_ns, current_net_ns) &&
+     !net_ns_match(current_net_ns, &init_net_ns))
+     continue;
     if (sk->sk_family == state->family)
      goto found;
   }
@@ -1565,8 +1568,14 @@ static struct sock *udp_get_next(struct
  do {
   sk = sk_next(sk);
 try_again:
-  ;
- } while (sk && sk->sk_family != state->family);
+  if (!sk)
+   break;
+  if (sk->sk_family != state->family)
+   continue;
+  if (net_ns_match(sk->sk_net_ns, current_net_ns) ||
+   net_ns_match(current_net_ns, &init_net_ns))
+   break;
+ } while (1);

  if (!sk && ++state->bucket < UDP_HTABLE_SIZE) {
   sk = sk_head(state->hashtable + state->bucket);
```

Subject: [PATCH 12/12] L2 network namespace (v3): L3 network namespace intro

Posted by Mishin Dmitry on Wed, 17 Jan 2007 16:18:25 GMT

View Forum Message <> Reply to Message

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

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

---
include/linux/net_namespace.h | 3 +++
net/core/net_namespace.c | 40 ++----------
2 files changed, 31 insertions(+), 12 deletions(-)

--- linux-2.6.20-rc4-mm1.net_ns.orig/include/linux/net_namespace.h

```
+++ linux-2.6.20-rc4-mm1.net_ns/include/linux/net_namespace.h
@@ -24,6 +24,9 @@ struct net_namespace {
  int   fib4_trie_last_dflt;
 #endif
  unsigned int  hash;
+#define NET_NS_LEVEL2  1
+#define NET_NS_LEVEL3  2
+ unsigned int  level;
 };

 extern struct net_namespace init_net_ns;
--- linux-2.6.20-rc4-mm1.net_ns.orig/net/core/net_namespace.c
+++ linux-2.6.20-rc4-mm1.net_ns/net/core/net_namespace.c
@@ -30,13 +30,19 @@ EXPORT_PER_CPU_SYMBOL_GPL(exec_net_ns);

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

+ /* level 3 namespaces are incomplete in order to have childs */
+ if (current_net_ns->level == NET_NS_LEVEL3)
+  return ERR_PTR(-EPERM);
+
  ns = kzalloc(sizeof(struct net_namespace), GFP_KERNEL);
  if (!ns)
   return NULL;
@@ -48,20 +54,25 @@ static struct net_namespace *clone_net_n

  if ((push_net_ns(ns)) != old_ns)
   BUG();
+ if (level ==  NET_NS_LEVEL2) {
 #ifdef CONFIG_IP_MULTIPLE_TABLES
- INIT_LIST_HEAD(&ns->fib_rules_ops_list);
+  INIT_LIST_HEAD(&ns->fib_rules_ops_list);
 #endif
- if (ip_fib_struct_init())
-  goto out_fib4;
+  if (ip_fib_struct_init())
+   goto out_fib4;
```

```
+ }
+ ns->level = level;
  if (loopback_init())
   goto out_loopback;
  pop_net_ns(old_ns);
- printk(KERN_DEBUG "NET_NS: created new netcontext %p for %s "
-  "(pid=%d)\n", ns, current->comm, current->tgid);
+ printk(KERN_DEBUG "NET_NS: created new netcontext %p, level %u, "
+  "for %s (pid=%d)\n", ns, (ns->level == NET_NS_LEVEL2) ?
+      2 : 3, current->comm, current->tgid);
  return ns;

 out_loopback:
- ip_fib_struct_cleanup(ns);
+ if (level ==  NET_NS_LEVEL2)
+  ip_fib_struct_cleanup(ns);
 out_fib4:
  pop_net_ns(old_ns);
  BUG_ON(atomic_read(&ns->kref.refcount) != 1);
@@ -75,13 +86,17 @@ out_fib4:
 int unshare_net_ns(unsigned long unshare_flags,
      struct net_namespace **new_net)
 {
+ unsigned int level;
+
  if (unshare_flags & (CLONE_NEWNET2|CLONE_NEWNET3)) {
   if (!capable(CAP_SYS_ADMIN))
    return -EPERM;

-  *new_net = clone_net_ns(current->nsproxy->net_ns);
-  if (!*new_net)
-   return -ENOMEM;
+  level = (unshare_flags & CLONE_NEWNET2) ? NET_NS_LEVEL2 :
+        NET_NS_LEVEL3;
+  *new_net = clone_net_ns(level, current->nsproxy->net_ns);
+  if (IS_ERR(*new_net))
+   return PTR_ERR(*new_net);
  }

  return 0;
@@ -110,7 +125,8 @@ void free_net_ns(struct kref *kref)
     ns, atomic_read(&ns->kref.refcount));
   return;
  }
- ip_fib_struct_cleanup(ns);
+ if (ns->level == NET_NS_LEVEL2)
+  ip_fib_struct_cleanup(ns);
  printk(KERN_DEBUG "NET_NS: net namespace %p destroyed\n", ns);
```

```
 kfree(ns);
}
```

_____

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

Subject: Re: [PATCH 1/12] L2 network namespace (v3): current network namespace operations
Posted by ebiederm on Thu, 18 Jan 2007 13:37:39 GMT
View Forum Message <> Reply to Message

Dmitry Mishin <dim@openvz.org> writes:

> On Wednesday 17 January 2007 23:16, Eric W. Biederman wrote:
>> Dmitry Mishin <dim@openvz.org> writes:
>>
>> > Added functions and macros required to operate with network namespaces.
>> > They are required in order to switch network namespace for incoming packets
> and
>> > to not extend current network interface by additional network namespace
> argue.
>>
>> Is exec_net only used in interrupt context?
> I tried to do so.
>
>> Or how do you ensure a sleeping function does not get called and the
>> kernel process comes back on another cpu?
> Seems that I forgot to remove it's usage at least in one place - in
> clone_net_ns(). If you caught more, please, let me know.

Sure. It was not clear from what I saw of the patch it was intended to
be restricted to only interrupt context. So if figured I would ask if
that was what you meant.

Eric

_____

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

Subject: Re: [PATCH 0/12] L2 network namespace (v3)
Posted by yoshfuji on Fri, 19 Jan 2007 00:07:45 GMT
View Forum Message <> Reply to Message

In article <200701171851.14734.dim@openvz.org> (at Wed, 17 Jan 2007 18:51:14 +0300),
Dmitry Mishin <dim@openvz.org> says:

> ==========
> L2 network namespaces
>
> The most straightforward concept of network virtualization is complete
> separation of namespaces, covering device list, routing tables, netfilter
> tables, socket hashes, and everything else.
>
> On input path, each packet is tagged with namespace right from the
> place where it appears from a device, and is processed by each layer
> in the context of this namespace.
> Non-root namespaces communicate with the outside world in two ways: by
> owning hardware devices, or receiving packets forwarded them by their parent
> namespace via pass-through device.

Can you handle multicast / broadcast and IPv6, which are very important?

--yoshfuji

Subject: Re: [PATCH 0/12] L2 network namespace (v3)
Posted by ebiederm on Fri, 19 Jan 2007 07:27:30 GMT
View Forum Message <> Reply to Message

YOSHIFUJI Hideaki / $B5HF#1QL@ (B <yoshfuji@linux-ipv6.org> writes:

> In article <200701171851.14734.dim@openvz.org> (at Wed, 17 Jan 2007 18:51:14
> +0300), Dmitry Mishin <dim@openvz.org> says:
>
>> ==========
>> L2 network namespaces
>>
>> The most straightforward concept of network virtualization is complete
>> separation of namespaces, covering device list, routing tables, netfilter
>> tables, socket hashes, and everything else.
>>
>> On input path, each packet is tagged with namespace right from the
>> place where it appears from a device, and is processed by each layer
>> in the context of this namespace.
>> Non-root namespaces communicate with the outside world in two ways: by
>> owning hardware devices, or receiving packets forwarded them by their parent
>> namespace via pass-through device.

>
> Can you handle multicast / broadcast and IPv6, which are very important?

The basic idea here is very simple.

Each network namespace appears to user space as a separate network stack,
with it's own set of routing tables etc.

All sockets and all network devices (the sources of packets) belong
to exactly one network namespace.

>From the socket or the network device a packet enters the network stack
you can infer the network namespace that it will be processed in.
Each network namespace should get it own complement of the data structures
necessary to process packets, and everything should work.

Talking between namespaces is accomplished either through an external network,
or through a special pseudo network device. The simplest to implement
is two network devices where all packets transmitted on one are received
on the other. Then by placing one network device in one namespace and
the other in another interface it looks like two machines connected by
a cross over cable.

Once you have that in a one namespace you can connect other namespaces
with the existing ethernet bridging or by configuring one of the
namespaces as a router and routing traffic between them.

Supporting IPv6 is roughly as difficult as supporting IPv4.

What needs to happen to convert code is all variables either need
a per network namespace instance or the data structures needs to be
modified to have a network namespace tag. For hash tables which
are hard to allocate dynamically tagging is the preferred conversion
method, for anything that is small enough duplication is preferred
as it allows the existing logic to be kept.

In the fast path the impact of all of the conversions should be very light,
to non-existent. In network stack initialization and cleanup there
is work todo because you are initializing and cleanup variables more often
then at module insertion and removal.

So my expectation is that once we get a framework established and merged
to allow network namespaces eventually the entire network stack will be
converted. Not just ipv4 and ipv6 but decnet, ipx, iptables, fair scheduling,
ethernet bridging and all of the other weird and twisty bits of the
linux network stack.

The primary practical hurdle is there is a lot of networking code in
the kernel.

I think I know a path by which we can incrementally merge support for
network namespaces without breaking anything. More to come on this
when I finish up my demonstration patchset in a week or so that
is complete enough to show what I am talking about.

I hope this helps but the concept into perspective.

As for Dmitry's patchset in particular it currently does not support
IPv6 and I don't know where it is with respect to the broadcast and
multicast but I don't see any immediate problems that would preclude
those from working. But any incompleteness is exactly that
incompleteness and an implementation problem not a fundamental design
issue.

Eric

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

Subject: Re: [PATCH 0/12] L2 network namespace (v3)
Posted by Mishin Dmitry on Fri, 19 Jan 2007 09:35:11 GMT
View Forum Message <> Reply to Message

On Friday 19 January 2007 10:27, Eric W. Biederman wrote:
> YOSHIFUJI Hideaki / $B5HF#1QL@ (B <yoshfuji@linux-ipv6.org> writes:
>
> > In article <200701171851.14734.dim@openvz.org> (at Wed, 17 Jan 2007 18:51:14
> > +0300), Dmitry Mishin <dim@openvz.org> says:
> >
> >> ==========
> >> L2 network namespaces
> >>
> >> The most straightforward concept of network virtualization is complete
> >> separation of namespaces, covering device list, routing tables, netfilter
> >> tables, socket hashes, and everything else.
> >>
> >> On input path, each packet is tagged with namespace right from the
> >> place where it appears from a device, and is processed by each layer
> >> in the context of this namespace.
> >> Non-root namespaces communicate with the outside world in two ways: by
> >> owning hardware devices, or receiving packets forwarded them by their parent
> >> namespace via pass-through device.

> >
> > Can you handle multicast / broadcast and IPv6, which are very important?
>
> The basic idea here is very simple.
>
> Each network namespace appears to user space as a separate network stack,
> with it's own set of routing tables etc.
>
> All sockets and all network devices (the sources of packets) belong
> to exactly one network namespace.
>
> >From the socket or the network device a packet enters the network stack
> you can infer the network namespace that it will be processed in.
> Each network namespace should get it own complement of the data structures
> necessary to process packets, and everything should work.
>
> Talking between namespaces is accomplished either through an external network,
> or through a special pseudo network device. The simplest to implement
> is two network devices where all packets transmitted on one are received
> on the other. Then by placing one network device in one namespace and
> the other in another interface it looks like two machines connected by
> a cross over cable.
>
> Once you have that in a one namespace you can connect other namespaces
> with the existing ethernet bridging or by configuring one of the
> namespaces as a router and routing traffic between them.
>
>
> Supporting IPv6 is roughly as difficult as supporting IPv4.
>
> What needs to happen to convert code is all variables either need
> a per network namespace instance or the data structures needs to be
> modified to have a network namespace tag. For hash tables which
> are hard to allocate dynamically tagging is the preferred conversion
> method, for anything that is small enough duplication is preferred
> as it allows the existing logic to be kept.
>
> In the fast path the impact of all of the conversions should be very light,
> to non-existent. In network stack initialization and cleanup there
> is work todo because you are initializing and cleanup variables more often
> then at module insertion and removal.
>
> So my expectation is that once we get a framework established and merged
> to allow network namespaces eventually the entire network stack will be
> converted. Not just ipv4 and ipv6 but decnet, ipx, iptables, fair scheduling,
> ethernet bridging and all of the other weird and twisty bits of the
> linux network stack.
Thanks Eric for such descriptive comment. I can only sign off on it :)

>
> The primary practical hurdle is there is a lot of networking code in
> the kernel.
>
> I think I know a path by which we can incrementally merge support for
> network namespaces without breaking anything. More to come on this
> when I finish up my demonstration patchset in a week or so that
> is complete enough to show what I am talking about.
>
> I hope this helps but the concept into perspective.
I'll be waiting it.

>
> As for Dmitry's patchset in particular it currently does not support
> IPv6 and I don't know where it is with respect to the broadcast and
> multicast but I don't see any immediate problems that would preclude
> those from working. But any incompleteness is exactly that
> incompleteness and an implementation problem not a fundamental design
> issue.
Broadcasts/multicasts are supported.

--
Thanks,
Dmitry.

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