Subject: [PATCH net-2.6.25 0/7] Make ipv4_devconf (all and default) live in net namespaces
Posted by Pavel Emelianov on Tue, 11 Dec 2007 17:44:09 GMT
View Forum Message <> Reply to Message

The ipv4_devconf_(all) and ipv4_devconf_dflt are currently
global, but should be per-namespace.

This set moves them on the struct net. Or, more precisely,
on the struct netns_ipv4, which in turn is on the struct net.

There are two minor things that are to be done additionally
to this set:

1. The snmp_seq_show() needs the IPV4_DEVCONF_ALL(FORWARDING)
   value, but since this entry is still global no valid struct
   net can be get in it, so I use the init_net's one. After
   snmp is made per-namespace, this will be fixed easily.

2. The rt_fill_info() needs the IPV4_DEVCONF_ALL(MC_FORWARDING),
   but the routing code is not tuned to work inside namespaces
   yet, so I use the init_net in it as well. Denis is currently
   working on ipv4 routing, so this will be prepared shortly.

Happily, all the other places of devconf-s usage can provide a
struct net pointer.

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

Subject: [PATCH net-2.6.25 1/7] Add the netns_ipv4 struct
Posted by Pavel Emelianov on Tue, 11 Dec 2007 17:45:58 GMT
View Forum Message <> Reply to Message

The ipv4 will store its parameters inside this structure.
This one is empty now, but it will be eventually filled.

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

---

diff --git a/include/net/net_namespace.h b/include/net/net_namespace.h
index 18da0af..d04ddf2 100644
--- a/include/net/net_namespace.h
+++ b/include/net/net_namespace.h
@@ -10,6 +10,7 @@

 #include <net/netns/unix.h>

```
 #include <net/netns/packet.h>
+#include <net/netns/ipv4.h>

 struct proc_dir_entry;
 struct net_device;
@@ -46,6 +47,7 @@ struct net {

  struct netns_packet packet;
  struct netns_unix unx;
+ struct netns_ipv4 ipv4;
 };

 #ifdef CONFIG_NET
diff --git a/include/net/netns/ipv4.h b/include/net/netns/ipv4.h
new file mode 100644
index 0000000..ce830d5
--- /dev/null
+++ b/include/net/netns/ipv4.h
@@ -0,0 +1,9 @@
+/*
+ * ipv4 in net namespaces
+ */
+
+#ifndef __NETNS_IPV4_H__
+#define __NETNS_IPV4_H__
+struct netns_ipv4 {
+};
+#endif
--
1.5.3.4
```

Subject: [PATCH net-2.6.25 2/7] Make __devinet_sysctl_register return an error
Posted by Pavel Emelianov on Tue, 11 Dec 2007 17:48:44 GMT
View Forum Message <> Reply to Message

Currently, this function is void, so failures in creating
sysctls for new/renamed devices are not reported to anywhere.

Fixing this is another complex (needed?) task, but this
return value is needed during the namespaces creation to
handle the case, when we failed to create "all" and "default"
entries.

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

```
diff --git a/net/ipv4/devinet.c b/net/ipv4/devinet.c
index 872883e..bfb0fb0 100644
--- a/net/ipv4/devinet.c
+++ b/net/ipv4/devinet.c
@@ -1455,8 +1460,8 @@ static struct devinet_sysctl_table {
  },
 };

-static void __devinet_sysctl_register(char *dev_name, int ctl_name,
-  struct ipv4_devconf *p)
+static int __devinet_sysctl_register(struct net *net, char *dev_name,
+  int ctl_name, struct ipv4_devconf *p)
 {
  int i;
  struct devinet_sysctl_table *t;
@@ -1498,14 +1504,14 @@ static void __devinet_sysctl_register(char *dev_name, int ctl_name,
   goto free_procname;

  p->sysctl = t;
- return;
+ return 0;

 free_procname:
  kfree(t->dev_name);
 free:
  kfree(t);
 out:
- return;
+ return -ENOBUFS;
 }

 static void __devinet_sysctl_unregister(struct ipv4_devconf *cnf)
--
1.5.3.4
```

Subject: [PATCH net-2.6.25 3/7] Pass the net pointer to the arp_req_set_proxy()
Posted by Pavel Emelianov on Tue, 11 Dec 2007 17:50:59 GMT
View Forum Message <> Reply to Message

This one will need to set the IPV4_DEVCONF_ALL(PROXY_ARP), but
there's no ways to get the net right in place, so we have to
pull one from the inet_ioctl's struct sock.

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

```
diff --git a/include/net/arp.h b/include/net/arp.h
index f026645..36482bf 100644
--- a/include/net/arp.h
+++ b/include/net/arp.h
@@ -11,7 +11,7 @@ extern struct neigh_table arp_tbl;

 extern void arp_init(void);
 extern int arp_find(unsigned char *haddr, struct sk_buff *skb);
-extern int arp_ioctl(unsigned int cmd, void __user *arg);
+extern int arp_ioctl(struct net *net, unsigned int cmd, void __user *arg);
 extern void     arp_send(int type, int ptype, __be32 dest_ip,
     struct net_device *dev, __be32 src_ip,
     unsigned char *dest_hw, unsigned char *src_hw, unsigned char *th);
diff --git a/net/ipv4/af_inet.c b/net/ipv4/af_inet.c
index 7f8b27f..a69b0c6 100644
--- a/net/ipv4/af_inet.c
+++ b/net/ipv4/af_inet.c
@@ -798,7 +798,7 @@ int inet_ioctl(struct socket *sock, unsigned int cmd, unsigned long arg)
   case SIOCDARP:
   case SIOCGARP:
   case SIOCSARP:
-   err = arp_ioctl(cmd, (void __user *)arg);
+   err = arp_ioctl(sk->sk_net, cmd, (void __user *)arg);
    break;
   case SIOCGIFADDR:
   case SIOCSIFADDR:
diff --git a/net/ipv4/arp.c b/net/ipv4/arp.c
index 48e7bf6..41a4d73 100644
--- a/net/ipv4/arp.c
+++ b/net/ipv4/arp.c
@@ -953,7 +953,7 @@ out_of_mem:
  * Set (create) an ARP cache entry.
  */

-static int arp_req_set_proxy(struct net_device *dev, int on)
+static int arp_req_set_proxy(struct net *net, struct net_device *dev, int on)
 {
  if (dev == NULL) {
   IPV4_DEVCONF_ALL(PROXY_ARP) = on;
@@ -966,7 +966,8 @@ static int arp_req_set_proxy(struct net_device *dev, int on)
  return -ENXIO;
 }

-static int arp_req_set_public(struct arpreq *r, struct net_device *dev)
+static int arp_req_set_public(struct net *net, struct arpreq *r,
+  struct net_device *dev)
 {
  __be32 ip = ((struct sockaddr_in *)&r->arp_pa)->sin_addr.s_addr;
```

```
  __be32 mask = ((struct sockaddr_in *)&r->arp_netmask)->sin_addr.s_addr;
@@ -985,17 +986,18 @@ static int arp_req_set_public(struct arpreq *r, struct net_device *dev)
   return 0;
  }

- return arp_req_set_proxy(dev, 1);
+ return arp_req_set_proxy(net, dev, 1);
 }

-static int arp_req_set(struct arpreq *r, struct net_device * dev)
+static int arp_req_set(struct net *net, struct arpreq *r,
+  struct net_device * dev)
 {
  __be32 ip;
  struct neighbour *neigh;
  int err;

  if (r->arp_flags & ATF_PUBL)
-  return arp_req_set_public(r, dev);
+  return arp_req_set_public(net, r, dev);

  ip = ((struct sockaddr_in *)&r->arp_pa)->sin_addr.s_addr;
  if (r->arp_flags & ATF_PERM)
@@ -1081,7 +1083,8 @@ static int arp_req_get(struct arpreq *r, struct net_device *dev)
  return err;
 }

-static int arp_req_delete_public(struct arpreq *r, struct net_device *dev)
+static int arp_req_delete_public(struct net *net, struct arpreq *r,
+  struct net_device *dev)
 {
  __be32 ip = ((struct sockaddr_in *) &r->arp_pa)->sin_addr.s_addr;
  __be32 mask = ((struct sockaddr_in *)&r->arp_netmask)->sin_addr.s_addr;
@@ -1092,17 +1095,18 @@ static int arp_req_delete_public(struct arpreq *r, struct net_device
*dev)
  if (mask)
   return -EINVAL;

- return arp_req_set_proxy(dev, 0);
+ return arp_req_set_proxy(net, dev, 0);
 }

-static int arp_req_delete(struct arpreq *r, struct net_device * dev)
+static int arp_req_delete(struct net *net, struct arpreq *r,
+  struct net_device * dev)
 {
  int err;
  __be32 ip;
```

```
  struct neighbour *neigh;

  if (r->arp_flags & ATF_PUBL)
-  return arp_req_delete_public(r, dev);
+  return arp_req_delete_public(net, r, dev);

  ip = ((struct sockaddr_in *)&r->arp_pa)->sin_addr.s_addr;
  if (dev == NULL) {
@@ -1132,7 +1136,7 @@ static int arp_req_delete(struct arpreq *r, struct net_device * dev)
  * Handle an ARP layer I/O control request.
  */

-int arp_ioctl(unsigned int cmd, void __user *arg)
+int arp_ioctl(struct net *net, unsigned int cmd, void __user *arg)
 {
  int err;
  struct arpreq r;
@@ -1180,10 +1184,10 @@ int arp_ioctl(unsigned int cmd, void __user *arg)

  switch (cmd) {
  case SIOCDARP:
-  err = arp_req_delete(&r, dev);
+  err = arp_req_delete(net, &r, dev);
   break;
  case SIOCSARP:
-  err = arp_req_set(&r, dev);
+  err = arp_req_set(net, &r, dev);
   break;
  case SIOCGARP:
   err = arp_req_get(&r, dev);
--
1.5.3.4
```

Subject: [PATCH net-2.6.25 4/7] Store the net pointer on devinet's ctl tables
Posted by Pavel Emelianov on Tue, 11 Dec 2007 17:53:55 GMT
View Forum Message <> Reply to Message

Some handers and strategies of devinet sysctl tables need
to know the net to propagate the ctl change to all the
net devices.

I use the (currently unused) extra2 pointer on the tables
to get it.

Holding the reference on the struct net is not possible,
because otherwise we'll get a net->ctl_table->net circular
dependency. But since the ctl tables are unregistered during

the net destruction, this is safe to get it w/o additional
protection.

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

```
diff --git a/net/ipv4/devinet.c b/net/ipv4/devinet.c
index 872883e..bfb0fb0 100644
--- a/net/ipv4/devinet.c
+++ b/net/ipv4/devinet.c
@@ -1237,12 +1237,12 @@ errout:

 #ifdef CONFIG_SYSCTL

-static void devinet_copy_dflt_conf(int i)
+static void devinet_copy_dflt_conf(struct net *net, int i)
 {
  struct net_device *dev;

  read_lock(&dev_base_lock);
- for_each_netdev(&init_net, dev) {
+ for_each_netdev(net, dev) {
   struct in_device *in_dev;
   rcu_read_lock();
   in_dev = __in_dev_get_rcu(dev);
@@ -1253,7 +1253,7 @@ static void devinet_copy_dflt_conf(int i)
  read_unlock(&dev_base_lock);
 }

-static void inet_forward_change(void)
+static void inet_forward_change(struct net *net)
 {
  struct net_device *dev;
  int on = IPV4_DEVCONF_ALL(FORWARDING);
@@ -1262,7 +1262,7 @@ static void inet_forward_change(void)
  IPV4_DEVCONF_DFLT(FORWARDING) = on;

  read_lock(&dev_base_lock);
- for_each_netdev(&init_net, dev) {
+ for_each_netdev(net, dev) {
   struct in_device *in_dev;
   rcu_read_lock();
   in_dev = __in_dev_get_rcu(dev);
@@ -1283,12 +1283,13 @@ static int devinet_conf_proc(ctl_table *ctl, int write,

  if (write) {
   struct ipv4_devconf *cnf = ctl->extra1;
```

```
+  struct net *net = ctl->extra2;
   int i = (int *)ctl->data - cnf->data;

   set_bit(i, cnf->state);

   if (cnf == &ipv4_devconf_dflt)
-   devinet_copy_dflt_conf(i);
+   devinet_copy_dflt_conf(net, i);
  }

  return ret;
@@ -1299,6 +1300,7 @@ static int devinet_conf_sysctl(ctl_table *table, int __user *name, int
nlen,
          void __user *newval, size_t newlen)
 {
  struct ipv4_devconf *cnf;
+ struct net *net;
  int *valp = table->data;
  int new;
  int i;
@@ -1334,12 +1336,13 @@ static int devinet_conf_sysctl(ctl_table *table, int __user *name, int
nlen,
  *valp = new;

  cnf = table->extra1;
+ net = table->extra2;
  i = (int *)table->data - cnf->data;

  set_bit(i, cnf->state);

  if (cnf == &ipv4_devconf_dflt)
-  devinet_copy_dflt_conf(i);
+  devinet_copy_dflt_conf(net, i);

  return 1;
 }
@@ -1353,8 +1356,10 @@ static int devinet_sysctl_forward(ctl_table *ctl, int write,
  int ret = proc_dointvec(ctl, write, filp, buffer, lenp, ppos);

  if (write && *valp != val) {
+  struct net *net = ctl->extra2;
+
   if (valp == &IPV4_DEVCONF_ALL(FORWARDING))
-   inet_forward_change();
+   inet_forward_change(net);
   else if (valp != &IPV4_DEVCONF_DFLT(FORWARDING))
    rt_cache_flush(0);
  }
```

```
@@ -1478,6 +1483,7 @@ static void __devinet_sysctl_register(char *dev_name, int ctl_name,
  for (i = 0; i < ARRAY_SIZE(t->devinet_vars) - 1; i++) {
   t->devinet_vars[i].data += (char *)p - (char *)&ipv4_devconf;
   t->devinet_vars[i].extra1 = p;
+  t->devinet_vars[i].extra2 = net;
  }

  /*
@@ -1525,8 +1531,8 @@ static void devinet_sysctl_register(struct in_device *idev)
 {
  neigh_sysctl_register(idev->dev, idev->arp_parms, NET_IPV4,
    NET_IPV4_NEIGH, "ipv4", NULL, NULL);
- __devinet_sysctl_register(idev->dev->name, idev->dev->ifindex,
-   &idev->cnf);
+ __devinet_sysctl_register(idev->dev->nd_net, idev->dev->name,
+   idev->dev->ifindex, &idev->cnf);
 }

 static void devinet_sysctl_unregister(struct in_device *idev)
@@ -1547,6 +1553,7 @@ static struct ctl_table ctl_forward_entry[] = {
   .proc_handler = devinet_sysctl_forward,
   .strategy = devinet_conf_sysctl,
   .extra1  = &ipv4_devconf,
+  .extra2  = &init_net,
  },
  { },
 };
@@ -1566,9 +1573,9 @@ void __init devinet_init(void)
  rtnl_register(PF_INET, RTM_DELADDR, inet_rtm_deladdr, NULL);
  rtnl_register(PF_INET, RTM_GETADDR, NULL, inet_dump_ifaddr);
 #ifdef CONFIG_SYSCTL
- __devinet_sysctl_register("all", NET_PROTO_CONF_ALL,
+ __devinet_sysctl_register(&init_net, "all", NET_PROTO_CONF_ALL,
    &ipv4_devconf);
- __devinet_sysctl_register("default", NET_PROTO_CONF_DEFAULT,
+ __devinet_sysctl_register(&init_net, "default", NET_PROTO_CONF_DEFAULT,
    &ipv4_devconf_dflt);
  register_sysctl_paths(net_ipv4_path, ctl_forward_entry);
 #endif
--
1.5.3.4
```

Subject: [PATCH net-2.6.25 5/7] Move the devinet pointers on the struct net
Posted by Pavel Emelianov on Tue, 11 Dec 2007 17:57:18 GMT
View Forum Message <> Reply to Message

This is the core.

Add all and default pointers on the netns_ipv4 and register
a new pernet subsys to initialize them.

Also add the ctl_table_header to register the
net.ipv4.ip_forward ctl.

I don't allocate additional memory for init_net, but use
global devinets.

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

```
diff --git a/include/net/netns/ipv4.h b/include/net/netns/ipv4.h
index ce830d5..e06d7cf 100644
--- a/include/net/netns/ipv4.h
+++ b/include/net/netns/ipv4.h
@@ -4,6 +4,12 @@

 #ifndef __NETNS_IPV4_H__
 #define __NETNS_IPV4_H__
+struct ctl_table_header;
+struct ipv4_devconf;
+
 struct netns_ipv4 {
+ struct ctl_table_header *forw_hdr;
+ struct ipv4_devconf *devconf_all;
+ struct ipv4_devconf *devconf_dflt;
 };
 #endif
diff --git a/net/ipv4/devinet.c b/net/ipv4/devinet.c
index bfb0fb0..996f07e 100644
--- a/net/ipv4/devinet.c
+++ b/net/ipv4/devinet.c
@@ -62,6 +62,7 @@
 #include <net/route.h>
 #include <net/ip_fib.h>
 #include <net/rtnetlink.h>
+#include <net/net_namespace.h>

 struct ipv4_devconf ipv4_devconf = {
  .data = {
@@ -1498,7 +1499,7 @@ static int __devinet_sysctl_register(struct net *net, char *dev_name,
  devinet_ctl_path[DEVINET_CTL_PATH_DEV].procname = t->dev_name;
  devinet_ctl_path[DEVINET_CTL_PATH_DEV].ctl_name = ctl_name;

- t->sysctl_header = register_sysctl_paths(devinet_ctl_path,
```

```
+ t->sysctl_header = register_net_sysctl_table(net, devinet_ctl_path,
    t->devinet_vars);
  if (!t->sysctl_header)
   goto free_procname;
@@ -1558,27 +1559,113 @@ static struct ctl_table ctl_forward_entry[] = {
  { },
 };

-static __initdata struct ctl_path net_ipv4_path[] = {
+static __net_initdata struct ctl_path net_ipv4_path[] = {
  { .procname = "net", .ctl_name = CTL_NET, },
  { .procname = "ipv4", .ctl_name = NET_IPV4, },
  { },
 };

+static __net_init int devinet_init_net(struct net *net)
+{
+ int err;
+ struct ctl_table *tbl;
+ struct ipv4_devconf *all, *dflt;
+ struct ctl_table_header *forw_hdr;
+
+ err = -ENOMEM;
+ all = &ipv4_devconf;
+ dflt = &ipv4_devconf_dflt;
+ tbl = ctl_forward_entry;
+
+ if (net != &init_net) {
+  all = kmemdup(all, sizeof(ipv4_devconf), GFP_KERNEL);
+  if (all == NULL)
+   goto err_alloc_all;
+
+  dflt = kmemdup(dflt, sizeof(ipv4_devconf_dflt), GFP_KERNEL);
+  if (dflt == NULL)
+   goto err_alloc_dflt;
+
+  tbl = kmemdup(tbl, sizeof(ctl_forward_entry), GFP_KERNEL);
+  if (tbl == NULL)
+   goto err_alloc_ctl;
+
+  tbl[0].data = &all->data[NET_IPV4_CONF_FORWARDING - 1];
+  tbl[0].extra1 = all;
+  tbl[0].extra2 = net;
+ }
+
+#ifdef CONFIG_SYSCTL
+ err = __devinet_sysctl_register(net, "all",
+   NET_PROTO_CONF_ALL, all);
```

```
+ if (err < 0)
+  goto err_reg_all;
+
+ err = __devinet_sysctl_register(net, "default",
+   NET_PROTO_CONF_DEFAULT, dflt);
+ if (err < 0)
+  goto err_reg_dflt;
+
+ err = -ENOMEM;
+ forw_hdr = register_net_sysctl_table(net, net_ipv4_path, tbl);
+ if (forw_hdr == NULL)
+  goto err_reg_ctl;
+#endif
+
+ net->ipv4.forw_hdr = forw_hdr;
+ net->ipv4.devconf_all = all;
+ net->ipv4.devconf_dflt = dflt;
+ return 0;
+
+#ifdef CONFIG_SYSCTL
+err_reg_ctl:
+ __devinet_sysctl_unregister(dflt);
+err_reg_dflt:
+ __devinet_sysctl_unregister(all);
+err_reg_all:
+ if (tbl != ctl_forward_entry)
+  kfree(tbl);
+#endif
+err_alloc_ctl:
+ if (dflt != &ipv4_devconf_dflt)
+  kfree(dflt);
+err_alloc_dflt:
+ if (all != &ipv4_devconf)
+  kfree(all);
+err_alloc_all:
+ return err;
+}
+
+static __net_exit void devinet_exit_net(struct net *net)
+{
+ struct ctl_table *tbl;
+
+ tbl = net->ipv4.forw_hdr->ctl_table_arg;
+#ifdef CONFIG_SYSCTL
+ unregister_net_sysctl_table(net->ipv4.forw_hdr);
+ __devinet_sysctl_unregister(net->ipv4.devconf_dflt);
+ __devinet_sysctl_unregister(net->ipv4.devconf_all);
+#endif
```

```
+ kfree(tbl);
+ kfree(net->ipv4.devconf_dflt);
+ kfree(net->ipv4.devconf_all);
+}
+
+static __net_initdata struct pernet_operations devinet_ops = {
+ .init = devinet_init_net,
+ .exit = devinet_exit_net,
+};
+
 void __init devinet_init(void)
 {
+ register_pernet_subsys(&devinet_ops);
+
  register_gifconf(PF_INET, inet_gifconf);
  register_netdevice_notifier(&ip_netdev_notifier);

  rtnl_register(PF_INET, RTM_NEWADDR, inet_rtm_newaddr, NULL);
  rtnl_register(PF_INET, RTM_DELADDR, inet_rtm_deladdr, NULL);
  rtnl_register(PF_INET, RTM_GETADDR, NULL, inet_dump_ifaddr);
-#ifdef CONFIG_SYSCTL
- __devinet_sysctl_register(&init_net, "all", NET_PROTO_CONF_ALL,
-   &ipv4_devconf);
- __devinet_sysctl_register(&init_net, "default", NET_PROTO_CONF_DEFAULT,
-   &ipv4_devconf_dflt);
- register_sysctl_paths(net_ipv4_path, ctl_forward_entry);
-#endif
 }

 EXPORT_SYMBOL(in_dev_finish_destroy);
--
1.5.3.4
```

Subject: [PATCH net-2.6.25 6/7] Switch users of ipv4_devconf_dflt to use the pernet one

Posted by Pavel Emelianov on Tue, 11 Dec 2007 17:59:58 GMT

View Forum Message <> Reply to Message

They are all collected in the net/ipv4/devinet.c file and
mostly use the IPV4_DEVCONF_DFLT macro.

So I add the net parameter to it and patch users accordingly.

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

```
diff --git a/net/ipv4/devinet.c b/net/ipv4/devinet.c
index 996f07e..cd957f4 100644
--- a/net/ipv4/devinet.c
+++ b/net/ipv4/devinet.c
@@ -83,7 +83,8 @@ static struct ipv4_devconf ipv4_devconf_dflt = {
  },
 };

-#define IPV4_DEVCONF_DFLT(attr) IPV4_DEVCONF(ipv4_devconf_dflt, attr)
+#define IPV4_DEVCONF_DFLT(net, attr) \
+ IPV4_DEVCONF((*net->ipv4.devconf_dflt), attr)

 static const struct nla_policy ifa_ipv4_policy[IFA_MAX+1] = {
  [IFA_LOCAL]      = { .type = NLA_U32 },
@@ -164,7 +165,8 @@ static struct in_device *inetdev_init(struct net_device *dev)
  if (!in_dev)
   goto out;
  INIT_RCU_HEAD(&in_dev->rcu_head);
- memcpy(&in_dev->cnf, &ipv4_devconf_dflt, sizeof(in_dev->cnf));
+ memcpy(&in_dev->cnf, dev->nd_net->ipv4.devconf_dflt,
+    sizeof(in_dev->cnf));
  in_dev->cnf.sysctl = NULL;
  in_dev->dev = dev;
  if ((in_dev->arp_parms = neigh_parms_alloc(dev, &arp_tbl)) == NULL)
@@ -1248,7 +1250,7 @@ static void devinet_copy_dflt_conf(struct net *net, int i)
   rcu_read_lock();
   in_dev = __in_dev_get_rcu(dev);
   if (in_dev && !test_bit(i, in_dev->cnf.state))
-   in_dev->cnf.data[i] = ipv4_devconf_dflt.data[i];
+   in_dev->cnf.data[i] = net->ipv4.devconf_dflt->data[i];
   rcu_read_unlock();
  }
  read_unlock(&dev_base_lock);
@@ -1260,7 +1262,7 @@ static void inet_forward_change(struct net *net)
  int on = IPV4_DEVCONF_ALL(FORWARDING);

  IPV4_DEVCONF_ALL(ACCEPT_REDIRECTS) = !on;
- IPV4_DEVCONF_DFLT(FORWARDING) = on;
+ IPV4_DEVCONF_DFLT(net, FORWARDING) = on;

  read_lock(&dev_base_lock);
  for_each_netdev(net, dev) {
@@ -1289,7 +1291,7 @@ static int devinet_conf_proc(ctl_table *ctl, int write,

   set_bit(i, cnf->state);

-  if (cnf == &ipv4_devconf_dflt)
+  if (cnf == net->ipv4.devconf_dflt)
```

```
    devinet_copy_dflt_conf(net, i);
  }

@@ -1342,7 +1344,7 @@ static int devinet_conf_sysctl(ctl_table *table, int __user *name, int
nlen,

  set_bit(i, cnf->state);

- if (cnf == &ipv4_devconf_dflt)
+ if (cnf == net->ipv4.devconf_dflt)
    devinet_copy_dflt_conf(net, i);

  return 1;
@@ -1361,7 +1363,7 @@ static int devinet_sysctl_forward(ctl_table *ctl, int write,

    if (valp == &IPV4_DEVCONF_ALL(FORWARDING))
      inet_forward_change(net);
-   else if (valp != &IPV4_DEVCONF_DFLT(FORWARDING))
+   else if (valp != &IPV4_DEVCONF_DFLT(net, FORWARDING))
      rt_cache_flush(0);
  }

--
1.5.3.4
```

Subject: [PATCH net-2.6.25 7/7] Switch users of ipv4_devconf(_all) to use the pernet one

Posted by Pavel Emelianov on Tue, 11 Dec 2007 18:02:18 GMT

View Forum Message <> Reply to Message

These are scattered over the code, but almost all the
"critical" places already have the proper struct net
at hand except for snmp proc showing function and routing
rtnl handler.

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

```
diff --git a/include/linux/inetdevice.h b/include/linux/inetdevice.h
index 962a062..b3c5081 100644
--- a/include/linux/inetdevice.h
+++ b/include/linux/inetdevice.h
@@ -44,7 +44,8 @@ struct in_device
 };

 #define IPV4_DEVCONF(cnf, attr) ((cnf).data[NET_IPV4_CONF_ ## attr - 1])
```

```
-#define IPV4_DEVCONF_ALL(attr) IPV4_DEVCONF(ipv4_devconf, attr)
+#define IPV4_DEVCONF_ALL(net, attr) \
+ IPV4_DEVCONF((*(net)->ipv4.devconf_all), attr)

 static inline int ipv4_devconf_get(struct in_device *in_dev, int index)
 {
@@ -71,11 +72,14 @@ static inline void ipv4_devconf_setall(struct in_device *in_dev)
  ipv4_devconf_set((in_dev), NET_IPV4_CONF_ ## attr, (val))

 #define IN_DEV_ANDCONF(in_dev, attr) \
- (IPV4_DEVCONF_ALL(attr) && IN_DEV_CONF_GET((in_dev), attr))
+ (IPV4_DEVCONF_ALL(in_dev->dev->nd_net, attr) && \
+  IN_DEV_CONF_GET((in_dev), attr))
 #define IN_DEV_ORCONF(in_dev, attr) \
- (IPV4_DEVCONF_ALL(attr) || IN_DEV_CONF_GET((in_dev), attr))
+ (IPV4_DEVCONF_ALL(in_dev->dev->nd_net, attr) || \
+  IN_DEV_CONF_GET((in_dev), attr))
 #define IN_DEV_MAXCONF(in_dev, attr) \
- (max(IPV4_DEVCONF_ALL(attr), IN_DEV_CONF_GET((in_dev), attr)))
+ (max(IPV4_DEVCONF_ALL(in_dev->dev->nd_net, attr), \
+      IN_DEV_CONF_GET((in_dev), attr)))

 #define IN_DEV_FORWARD(in_dev)  IN_DEV_CONF_GET((in_dev), FORWARDING)
 #define IN_DEV_MFORWARD(in_dev)  IN_DEV_ANDCONF((in_dev), MC_FORWARDING)
diff --git a/net/ipv4/arp.c b/net/ipv4/arp.c
index 41a4d73..cd4c95a 100644
--- a/net/ipv4/arp.c
+++ b/net/ipv4/arp.c
@@ -861,7 +861,7 @@ static int arp_process(struct sk_buff *skb)

  n = __neigh_lookup(&arp_tbl, &sip, dev, 0);

- if (IPV4_DEVCONF_ALL(ARP_ACCEPT)) {
+ if (IPV4_DEVCONF_ALL(dev->nd_net, ARP_ACCEPT)) {
   /* Unsolicited ARP is not accepted by default.
      It is possible, that this option should be enabled for some
      devices (strip is candidate)
@@ -956,7 +956,7 @@ out_of_mem:
 static int arp_req_set_proxy(struct net *net, struct net_device *dev, int on)
 {
  if (dev == NULL) {
-  IPV4_DEVCONF_ALL(PROXY_ARP) = on;
+  IPV4_DEVCONF_ALL(net, PROXY_ARP) = on;
   return 0;
  }
  if (__in_dev_get_rtnl(dev)) {
diff --git a/net/ipv4/devinet.c b/net/ipv4/devinet.c
index cd957f4..e94de54 100644
```

```
--- a/net/ipv4/devinet.c
+++ b/net/ipv4/devinet.c
@@ -1259,9 +1259,9 @@ static void devinet_copy_dflt_conf(struct net *net, int i)
 static void inet_forward_change(struct net *net)
 {
  struct net_device *dev;
- int on = IPV4_DEVCONF_ALL(FORWARDING);
+ int on = IPV4_DEVCONF_ALL(net, FORWARDING);

- IPV4_DEVCONF_ALL(ACCEPT_REDIRECTS) = !on;
+ IPV4_DEVCONF_ALL(net, ACCEPT_REDIRECTS) = !on;
  IPV4_DEVCONF_DFLT(net, FORWARDING) = on;

  read_lock(&dev_base_lock);
@@ -1361,7 +1361,7 @@ static int devinet_sysctl_forward(ctl_table *ctl, int write,
  if (write && *valp != val) {
   struct net *net = ctl->extra2;

-  if (valp == &IPV4_DEVCONF_ALL(FORWARDING))
+  if (valp == &IPV4_DEVCONF_ALL(net, FORWARDING))
    inet_forward_change(net);
   else if (valp != &IPV4_DEVCONF_DFLT(net, FORWARDING))
    rt_cache_flush(0);
diff --git a/net/ipv4/igmp.c b/net/ipv4/igmp.c
index c560a93..d3d5906 100644
--- a/net/ipv4/igmp.c
+++ b/net/ipv4/igmp.c
@@ -130,12 +130,12 @@
  */

 #define IGMP_V1_SEEN(in_dev) \
- (IPV4_DEVCONF_ALL(FORCE_IGMP_VERSION) == 1 || \
+ (IPV4_DEVCONF_ALL(in_dev->dev->nd_net, FORCE_IGMP_VERSION) == 1 || \
   IN_DEV_CONF_GET((in_dev), FORCE_IGMP_VERSION) == 1 || \
   ((in_dev)->mr_v1_seen && \
    time_before(jiffies, (in_dev)->mr_v1_seen)))
 #define IGMP_V2_SEEN(in_dev) \
- (IPV4_DEVCONF_ALL(FORCE_IGMP_VERSION) == 2 || \
+ (IPV4_DEVCONF_ALL(in_dev->dev->nd_net, FORCE_IGMP_VERSION) == 2 || \
   IN_DEV_CONF_GET((in_dev), FORCE_IGMP_VERSION) == 2 || \
   ((in_dev)->mr_v2_seen && \
    time_before(jiffies, (in_dev)->mr_v2_seen)))
diff --git a/net/ipv4/ipmr.c b/net/ipv4/ipmr.c
index 1187928..9947f52 100644
--- a/net/ipv4/ipmr.c
+++ b/net/ipv4/ipmr.c
@@ -849,7 +849,7 @@ static void mrtsock_destruct(struct sock *sk)
 {
```

```
  rtnl_lock();
  if (sk == mroute_socket) {
-  IPV4_DEVCONF_ALL(MC_FORWARDING)--;
+  IPV4_DEVCONF_ALL(sk->sk_net, MC_FORWARDING)--;

   write_lock_bh(&mrt_lock);
   mroute_socket=NULL;
@@ -898,7 +898,7 @@ int ip_mroute_setsockopt(struct sock *sk,int optname,char __user
*optval,int opt
    mroute_socket=sk;
    write_unlock_bh(&mrt_lock);

-   IPV4_DEVCONF_ALL(MC_FORWARDING)++;
+   IPV4_DEVCONF_ALL(sk->sk_net, MC_FORWARDING)++;
   }
   rtnl_unlock();
   return ret;
diff --git a/net/ipv4/proc.c b/net/ipv4/proc.c
index ce34b28..41734db 100644
--- a/net/ipv4/proc.c
+++ b/net/ipv4/proc.c
@@ -309,7 +309,8 @@ static int snmp_seq_show(struct seq_file *seq, void *v)
   seq_printf(seq, " %s", snmp4_ipstats_list[i].name);

  seq_printf(seq, "\nIp: %d %d",
-     IPV4_DEVCONF_ALL(FORWARDING) ? 1 : 2, sysctl_ip_default_ttl);
+     IPV4_DEVCONF_ALL(&init_net, FORWARDING) ? 1 : 2,
+     sysctl_ip_default_ttl);

  for (i = 0; snmp4_ipstats_list[i].name != NULL; i++)
   seq_printf(seq, " %lu",
diff --git a/net/ipv4/route.c b/net/ipv4/route.c
index cc064e6..499e1e3 100644
--- a/net/ipv4/route.c
+++ b/net/ipv4/route.c
@@ -2615,7 +2615,7 @@ static int rt_fill_info(struct sk_buff *skb, u32 pid, u32 seq, int event,
   __be32 dst = rt->rt_dst;

   if (MULTICAST(dst) && !LOCAL_MCAST(dst) &&
-      IPV4_DEVCONF_ALL(MC_FORWARDING)) {
+      IPV4_DEVCONF_ALL(&init_net, MC_FORWARDING)) {
    int err = ipmr_get_route(skb, r, nowait);
    if (err <= 0) {
     if (!nowait) {
--
1.5.3.4
```

Subject: Re: [PATCH net-2.6.25 1/7] Add the netns_ipv4 struct
Posted by davem on Sun, 16 Dec 2007 21:29:43 GMT
View Forum Message <> Reply to Message

From: Pavel Emelyanov <xemul@openvz.org>
Date: Tue, 11 Dec 2007 20:45:58 +0300

> The ipv4 will store its parameters inside this structure.
> This one is empty now, but it will be eventually filled.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

Applied.

Subject: Re: [PATCH net-2.6.25 2/7] Make __devinet_sysctl_register return an error
Posted by davem on Sun, 16 Dec 2007 21:30:15 GMT
View Forum Message <> Reply to Message

From: Pavel Emelyanov <xemul@openvz.org>
Date: Tue, 11 Dec 2007 20:48:44 +0300

> Currently, this function is void, so failures in creating
> sysctls for new/renamed devices are not reported to anywhere.
>
> Fixing this is another complex (needed?) task, but this
> return value is needed during the namespaces creation to
> handle the case, when we failed to create "all" and "default"
> entries.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

Applied.

Subject: Re: [PATCH net-2.6.25 3/7] Pass the net pointer to the arp_req_set_proxy()
Posted by davem on Sun, 16 Dec 2007 21:30:49 GMT
View Forum Message <> Reply to Message

From: Pavel Emelyanov <xemul@openvz.org>
Date: Tue, 11 Dec 2007 20:50:59 +0300

> This one will need to set the IPV4_DEVCONF_ALL(PROXY_ARP), but
> there's no ways to get the net right in place, so we have to
> pull one from the inet_ioctl's struct sock.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

Applied.

Subject: Re: [PATCH net-2.6.25 4/7] Store the net pointer on devinet's ctl tables
Posted by davem on Sun, 16 Dec 2007 21:31:23 GMT
View Forum Message <> Reply to Message

From: Pavel Emelyanov <xemul@openvz.org>
Date: Tue, 11 Dec 2007 20:53:55 +0300

> Some handers and strategies of devinet sysctl tables need
> to know the net to propagate the ctl change to all the
> net devices.
>
> I use the (currently unused) extra2 pointer on the tables
> to get it.
>
> Holding the reference on the struct net is not possible,
> because otherwise we'll get a net->ctl_table->net circular
> dependency. But since the ctl tables are unregistered during
> the net destruction, this is safe to get it w/o additional
> protection.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

Applied.

Subject: Re: [PATCH net-2.6.25 5/7] Move the devinet pointers on the struct net
Posted by davem on Sun, 16 Dec 2007 21:31:56 GMT
View Forum Message <> Reply to Message

From: Pavel Emelyanov <xemul@openvz.org>
Date: Tue, 11 Dec 2007 20:57:18 +0300

> This is the core.
>
> Add all and default pointers on the netns_ipv4 and register
> a new pernet subsys to initialize them.
>
> Also add the ctl_table_header to register the
> net.ipv4.ip_forward ctl.
>
> I don't allocate additional memory for init_net, but use
> global devinets.
>

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

Applied.

Subject: Re: [PATCH net-2.6.25 6/7] Switch users of ipv4_devconf_dflt to use the pernet one
Posted by davem on Sun, 16 Dec 2007 21:32:23 GMT
View Forum Message <> Reply to Message

From: Pavel Emelyanov <xemul@openvz.org>
Date: Tue, 11 Dec 2007 20:59:58 +0300

> They are all collected in the net/ipv4/devinet.c file and
> mostly use the IPV4_DEVCONF_DFLT macro.
>
> So I add the net parameter to it and patch users accordingly.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

Applied.

Subject: Re: [PATCH net-2.6.25 7/7] Switch users of ipv4_devconf(_all) to use the pernet one
Posted by davem on Sun, 16 Dec 2007 21:32:57 GMT
View Forum Message <> Reply to Message

From: Pavel Emelyanov <xemul@openvz.org>
Date: Tue, 11 Dec 2007 21:02:18 +0300

> These are scattered over the code, but almost all the
> "critical" places already have the proper struct net
> at hand except for snmp proc showing function and routing
> rtnl handler.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

Applied.
