Subject: [PATCH 0/11 net-2.6.26] UDP/ICMP/TCP for a namespace
Posted by den on Mon, 24 Mar 2008 14:33:05 GMT
View Forum Message <> Reply to Message

Hello, Dave!

This patch set finally enables TCP/UDP and ICMP inside a namespace.
In order to do this we fix ARP processing, IP options processing and
allow packets to flow to namespace aware protocols.

Finally, this makes namespace functional and alive :)

Signed-off-by: Denis V. Lunev <den@openvz.org>

Subject: [PATCH 10/11 net-2.6.26] [NETNS]: Allow to create sockets in non-initial namespace.
Posted by den on Mon, 24 Mar 2008 14:36:02 GMT
View Forum Message <> Reply to Message

Allow to create sockets in the namespace if the protocol ok with this.

Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 net/ipv4/af_inet.c | 24 ++++++++++++++++++++----
 1 files changed, 21 insertions(+), 3 deletions(-)

```
diff --git a/net/ipv4/af_inet.c b/net/ipv4/af_inet.c
index a9e241c..39ddb84 100644
--- a/net/ipv4/af_inet.c
+++ b/net/ipv4/af_inet.c
@@ -243,6 +243,23 @@ void build_ehash_secret(void)
 }
 EXPORT_SYMBOL(build_ehash_secret);

+static inline int inet_netns_ok(struct net *net, int protocol)
+{
+ int hash;
+ struct net_protocol *ipprot;
+
+ if (net == &init_net)
+  return 1;
+
+ hash = protocol & (MAX_INET_PROTOS - 1);
+ ipprot = rcu_dereference(inet_protos[hash]);
+
+ if (ipprot == NULL)
+  /* raw IP is OK */
```

```
+  return 1;
+ return ipprot->netns_ok;
+}
+
 /*
  * Create an inet socket.
  */
@@ -259,9 +276,6 @@ static int inet_create(struct net *net, struct socket *sock, int protocol)
  int try_loading_module = 0;
  int err;

- if (net != &init_net)
-  return -EAFNOSUPPORT;
-
  if (sock->type != SOCK_RAW &&
      sock->type != SOCK_DGRAM &&
      !inet_ehash_secret)
@@ -320,6 +334,10 @@ lookup_protocol:
  if (answer->capability > 0 && !capable(answer->capability))
   goto out_rcu_unlock;

+ err = -EAFNOSUPPORT;
+ if (!inet_netns_ok(net, protocol))
+  goto out_rcu_unlock;
+
  sock->ops = answer->ops;
  answer_prot = answer->prot;
  answer_no_check = answer->no_check;
--
1.5.3.rc5
```

Subject: [PATCH 1/11 2.6.26] [NETNS]: Process ARP in the context of the correct namespace.

Posted by den on Mon, 24 Mar 2008 14:36:02 GMT

View Forum Message <> Reply to Message

Get namespace from a device and pass it to the routing engine. Enable ARP
packet processing and device notifiers after that.

Signed-off-by: Denis V. Lunev <den@openvz.org>

```
---
 net/ipv4/arp.c |   23 +++++++++----------------
 1 files changed, 9 insertions(+), 14 deletions(-)

diff --git a/net/ipv4/arp.c b/net/ipv4/arp.c
index efe01df..6d90ec5 100644
--- a/net/ipv4/arp.c
```

```
+++ b/net/ipv4/arp.c
@@ -242,7 +242,7 @@ static int arp_constructor(struct neighbour *neigh)
   return -EINVAL;
  }

- neigh->type = inet_addr_type(&init_net, addr);
+ neigh->type = inet_addr_type(dev->nd_net, addr);

  parms = in_dev->arp_parms;
  __neigh_parms_put(neigh->parms);
@@ -341,14 +341,14 @@ static void arp_solicit(struct neighbour *neigh, struct sk_buff *skb)
  switch (IN_DEV_ARP_ANNOUNCE(in_dev)) {
  default:
  case 0:  /* By default announce any local IP */
-  if (skb && inet_addr_type(&init_net, ip_hdr(skb)->saddr) == RTN_LOCAL)
+  if (skb && inet_addr_type(dev->nd_net, ip_hdr(skb)->saddr) == RTN_LOCAL)
    saddr = ip_hdr(skb)->saddr;
   break;
  case 1:  /* Restrict announcements of saddr in same subnet */
   if (!skb)
    break;
   saddr = ip_hdr(skb)->saddr;
-  if (inet_addr_type(&init_net, saddr) == RTN_LOCAL) {
+  if (inet_addr_type(dev->nd_net, saddr) == RTN_LOCAL) {
    /* saddr should be known to target */
    if (inet_addr_onlink(in_dev, target, saddr))
     break;
@@ -424,7 +424,7 @@ static int arp_filter(__be32 sip, __be32 tip, struct net_device *dev)
  int flag = 0;
  /*unsigned long now; */

- if (ip_route_output_key(&init_net, &rt, &fl) < 0)
+ if (ip_route_output_key(dev->nd_net, &rt, &fl) < 0)
   return 1;
  if (rt->u.dst.dev != dev) {
   NET_INC_STATS_BH(LINUX_MIB_ARPFILTER);
@@ -477,7 +477,7 @@ int arp_find(unsigned char *haddr, struct sk_buff *skb)

  paddr = skb->rtable->rt_gateway;

- if (arp_set_predefined(inet_addr_type(&init_net, paddr), haddr, paddr, dev))
+ if (arp_set_predefined(inet_addr_type(dev->nd_net, paddr), haddr, paddr, dev))
   return 0;

  n = __neigh_lookup(&arp_tbl, &paddr, dev, 1);
@@ -709,6 +709,7 @@ static int arp_process(struct sk_buff *skb)
  u16 dev_type = dev->type;
  int addr_type;
```

```
  struct neighbour *n;
+ struct net *net = dev->nd_net;

  /* arp_rcv below verifies the ARP header and verifies the device
   * is ARP'able.
@@ -804,7 +805,7 @@ static int arp_process(struct sk_buff *skb)
  /* Special case: IPv4 duplicate address detection packet (RFC2131) */
  if (sip == 0) {
   if (arp->ar_op == htons(ARPOP_REQUEST) &&
-      inet_addr_type(&init_net, tip) == RTN_LOCAL &&
+      inet_addr_type(net, tip) == RTN_LOCAL &&
       !arp_ignore(in_dev, sip, tip))
    arp_send(ARPOP_REPLY, ETH_P_ARP, sip, dev, tip, sha,
      dev->dev_addr, sha);
@@ -834,7 +835,7 @@ static int arp_process(struct sk_buff *skb)
    goto out;
   } else if (IN_DEV_FORWARD(in_dev)) {
       if (addr_type == RTN_UNICAST  && rt->u.dst.dev != dev &&
-        (arp_fwd_proxy(in_dev, rt) || pneigh_lookup(&arp_tbl, &init_net, &tip, dev, 0))) {
+        (arp_fwd_proxy(in_dev, rt) || pneigh_lookup(&arp_tbl, net, &tip, dev, 0))) {
     n = neigh_event_ns(&arp_tbl, sha, &sip, dev);
     if (n)
      neigh_release(n);
@@ -864,7 +865,7 @@ static int arp_process(struct sk_buff *skb)
    */
   if (n == NULL &&
       arp->ar_op == htons(ARPOP_REPLY) &&
-      inet_addr_type(&init_net, sip) == RTN_UNICAST)
+      inet_addr_type(net, sip) == RTN_UNICAST)
    n = __neigh_lookup(&arp_tbl, &sip, dev, 1);
  }

@@ -911,9 +912,6 @@ static int arp_rcv(struct sk_buff *skb, struct net_device *dev,
 {
  struct arphdr *arp;

- if (dev->nd_net != &init_net)
-  goto freeskb;
-
  /* ARP header, plus 2 device addresses, plus 2 IP addresses.  */
  if (!pskb_may_pull(skb, arp_hdr_len(dev)))
   goto freeskb;
@@ -1198,9 +1196,6 @@ static int arp_netdev_event(struct notifier_block *this, unsigned long
event, vo
 {
  struct net_device *dev = ptr;

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

```
-  return NOTIFY_DONE;
-
  switch (event) {
  case NETDEV_CHANGEADDR:
   neigh_changeaddr(&arp_tbl, dev);
--
1.5.3.rc5
```

Subject: [PATCH 2/11 net-2.6.26] [NETNS]: /proc/net/arp namespacing.
Posted by den on Mon, 24 Mar 2008 14:36:03 GMT
View Forum Message <> Reply to Message

Seqfile operation showing /proc/net/arp are already namespace aware. All we
need is to register this file for each namespace.

Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 net/ipv4/arp.c | 20 ++++++++++++++++++--
 1 files changed, 18 insertions(+), 2 deletions(-)

```
diff --git a/net/ipv4/arp.c b/net/ipv4/arp.c
index 6d90ec5..832473e 100644
--- a/net/ipv4/arp.c
+++ b/net/ipv4/arp.c
@@ -1377,13 +1377,29 @@ static const struct file_operations arp_seq_fops = {
  .release = seq_release_net,
 };

-static int __init arp_proc_init(void)
+
+static int __net_init arp_net_init(struct net *net)
 {
- if (!proc_net_fops_create(&init_net, "arp", S_IRUGO, &arp_seq_fops))
+ if (!proc_net_fops_create(net, "arp", S_IRUGO, &arp_seq_fops))
   return -ENOMEM;
  return 0;
 }

+static void __net_exit arp_net_exit(struct net *net)
+{
+ proc_net_remove(net, "arp");
+}
+
+static struct pernet_operations arp_net_ops = {
+ .init = arp_net_init,
+ .exit = arp_net_exit,
+};
```

```
+
+static int __init arp_proc_init(void)
+{
+ return register_pernet_subsys(&arp_net_ops);
+}
+
 #else /* CONFIG_PROC_FS */

 static int __init arp_proc_init(void)
--
1.5.3.rc5
```

Subject: [PATCH 3/11 net-2.6.26] [NETNS]: Add namespace parameter to ip_options_compile.
Posted by den on Mon, 24 Mar 2008 14:36:04 GMT
View Forum Message <> Reply to Message

ip_options_compile uses inet_addr_type which requires a namespace. The packet
argument is optional, so parameter is the only way to obtain it. Pass the
init_net there for now.

Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 include/net/ip.h | 3 ++-
 net/ipv4/ip_input.c | 2 +-
 net/ipv4/ip_options.c | 7 ++++---
 3 files changed, 7 insertions(+), 5 deletions(-)

```
diff --git a/include/net/ip.h b/include/net/ip.h
index 9f50d4f..bcc3afa 100644
--- a/include/net/ip.h
+++ b/include/net/ip.h
@@ -347,7 +347,8 @@ extern int ip_forward(struct sk_buff *skb);
 extern void ip_options_build(struct sk_buff *skb, struct ip_options *opt, __be32 daddr, struct
rtable *rt, int is_frag);
 extern int ip_options_echo(struct ip_options *dopt, struct sk_buff *skb);
 extern void ip_options_fragment(struct sk_buff *skb);
-extern int ip_options_compile(struct ip_options *opt, struct sk_buff *skb);
+extern int ip_options_compile(struct net *net,
+         struct ip_options *opt, struct sk_buff *skb);
 extern int ip_options_get(struct ip_options **optp,
       unsigned char *data, int optlen);
 extern int ip_options_get_from_user(struct ip_options **optp,
diff --git a/net/ipv4/ip_input.c b/net/ipv4/ip_input.c
index e3a0c78..f3a7a08 100644
--- a/net/ipv4/ip_input.c
+++ b/net/ipv4/ip_input.c
```

```
@@ -286,7 +286,7 @@ static inline int ip_rcv_options(struct sk_buff *skb)
  opt = &(IPCB(skb)->opt);
  opt->optlen = iph->ihl*4 - sizeof(struct iphdr);

- if (ip_options_compile(opt, skb)) {
+ if (ip_options_compile(&init_net, opt, skb)) {
   IP_INC_STATS_BH(IPSTATS_MIB_INHDRERRORS);
   goto drop;
  }
diff --git a/net/ipv4/ip_options.c b/net/ipv4/ip_options.c
index aeed4e5..f0949b4 100644
--- a/net/ipv4/ip_options.c
+++ b/net/ipv4/ip_options.c
@@ -248,7 +248,8 @@ void ip_options_fragment(struct sk_buff * skb)
  * If opt == NULL, then skb->data should point to IP header.
  */

-int ip_options_compile(struct ip_options * opt, struct sk_buff * skb)
+int ip_options_compile(struct net *net,
+         struct ip_options * opt, struct sk_buff * skb)
 {
  int l;
  unsigned char * iph;
@@ -389,7 +390,7 @@ int ip_options_compile(struct ip_options * opt, struct sk_buff * skb)
      {
       __be32 addr;
       memcpy(&addr, &optptr[optptr[2]-1], 4);
-      if (inet_addr_type(&init_net, addr) == RTN_UNICAST)
+      if (inet_addr_type(net, addr) == RTN_UNICAST)
        break;
       if (skb)
        timeptr = (__be32*)&optptr[optptr[2]+3];
@@ -512,7 +513,7 @@ static int ip_options_get_finish(struct ip_options **optp,
  while (optlen & 3)
   opt->__data[optlen++] = IPOPT_END;
  opt->optlen = optlen;
- if (optlen && ip_options_compile(opt, NULL)) {
+ if (optlen && ip_options_compile(&init_net, opt, NULL)) {
   kfree(opt);
   return -EINVAL;
  }
--
1.5.3.rc5
```

Subject: [PATCH 4/11 net-2.6.25] [NETNS]: Add namespace parameter to ip_options_get(...).

Pass the init_net there for now.

Signed-off-by: Denis V. Lunev <den@openvz.org>
---
include/net/ip.h | 4 ++--
net/ipv4/ip_options.c | 14 ++++++++------
net/ipv4/ip_sockglue.c | 4 ++--
3 files changed, 12 insertions(+), 10 deletions(-)

```
diff --git a/include/net/ip.h b/include/net/ip.h
index bcc3afa..531270d 100644
--- a/include/net/ip.h
+++ b/include/net/ip.h
@@ -349,9 +349,9 @@ extern int ip_options_echo(struct ip_options *dopt, struct sk_buff *skb);
 extern void ip_options_fragment(struct sk_buff *skb);
 extern int ip_options_compile(struct net *net,
         struct ip_options *opt, struct sk_buff *skb);
-extern int ip_options_get(struct ip_options **optp,
+extern int ip_options_get(struct net *net, struct ip_options **optp,
      unsigned char *data, int optlen);
-extern int ip_options_get_from_user(struct ip_options **optp,
+extern int ip_options_get_from_user(struct net *net, struct ip_options **optp,
        unsigned char __user *data, int optlen);
 extern void ip_options_undo(struct ip_options * opt);
 extern void ip_forward_options(struct sk_buff *skb);
diff --git a/net/ipv4/ip_options.c b/net/ipv4/ip_options.c
index f0949b4..59f7ddf 100644
--- a/net/ipv4/ip_options.c
+++ b/net/ipv4/ip_options.c
@@ -507,13 +507,13 @@ static struct ip_options *ip_options_get_alloc(const int optlen)
         GFP_KERNEL);
 }

-static int ip_options_get_finish(struct ip_options **optp,
+static int ip_options_get_finish(struct net *net, struct ip_options **optp,
      struct ip_options *opt, int optlen)
 {
  while (optlen & 3)
   opt->__data[optlen++] = IPOPT_END;
  opt->optlen = optlen;
- if (optlen && ip_options_compile(&init_net, opt, NULL)) {
+ if (optlen && ip_options_compile(net, opt, NULL)) {
   kfree(opt);
   return -EINVAL;
  }
@@ -522,7 +522,8 @@ static int ip_options_get_finish(struct ip_options **optp,
```

```
  return 0;
 }

-int ip_options_get_from_user(struct ip_options **optp, unsigned char __user *data, int optlen)
+int ip_options_get_from_user(struct net *net, struct ip_options **optp,
+        unsigned char __user *data, int optlen)
 {
  struct ip_options *opt = ip_options_get_alloc(optlen);

@@ -532,10 +533,11 @@ int ip_options_get_from_user(struct ip_options **optp, unsigned char
__user *dat
    kfree(opt);
    return -EFAULT;
   }
- return ip_options_get_finish(optp, opt, optlen);
+ return ip_options_get_finish(net, optp, opt, optlen);
 }

-int ip_options_get(struct ip_options **optp, unsigned char *data, int optlen)
+int ip_options_get(struct net *net, struct ip_options **optp,
+     unsigned char *data, int optlen)
 {
  struct ip_options *opt = ip_options_get_alloc(optlen);

@@ -543,7 +545,7 @@ int ip_options_get(struct ip_options **optp, unsigned char *data, int
optlen)
    return -ENOMEM;
   if (optlen)
    memcpy(opt->__data, data, optlen);
- return ip_options_get_finish(optp, opt, optlen);
+ return ip_options_get_finish(net, optp, opt, optlen);
 }

 void ip_forward_options(struct sk_buff *skb)
diff --git a/net/ipv4/ip_sockglue.c b/net/ipv4/ip_sockglue.c
index bb3cbe5..1b86a50 100644
--- a/net/ipv4/ip_sockglue.c
+++ b/net/ipv4/ip_sockglue.c
@@ -176,7 +176,7 @@ int ip_cmsg_send(struct msghdr *msg, struct ipcm_cookie *ipc)
    switch (cmsg->cmsg_type) {
    case IP_RETOPTS:
     err = cmsg->cmsg_len - CMSG_ALIGN(sizeof(struct cmsghdr));
-    err = ip_options_get(&ipc->opt, CMSG_DATA(cmsg), err < 40 ? err : 40);
+    err = ip_options_get(&init_net, &ipc->opt, CMSG_DATA(cmsg), err < 40 ? err : 40);
     if (err)
      return err;
     break;
@@ -449,7 +449,7 @@ static int do_ip_setsockopt(struct sock *sk, int level,
```

```
   struct ip_options * opt = NULL;
   if (optlen > 40 || optlen < 0)
    goto e_inval;
-  err = ip_options_get_from_user(&opt, optval, optlen);
+  err = ip_options_get_from_user(&init_net, &opt, optval, optlen);
   if (err)
    break;
   if (inet->is_icsk) {
--
1.5.3.rc5
```

Subject: [PATCH 8/11 net-2.6.26] [NETNS]: Process netfilter hooks in initial namespace only.
Posted by den on Mon, 24 Mar 2008 14:36:06 GMT
View Forum Message <> Reply to Message

There were no packets in the namespace other than initial previously. This
will be changed in the neareast future. Netfilters are not namespace aware
and should be processed in the initial namespace only for now.

Signed-off-by: Denis V. Lunev <den@openvz.org>
CC: Patrick McHardy <kaber@trash.net>
---
 net/netfilter/core.c | 8 ++++++++
 1 files changed, 8 insertions(+), 0 deletions(-)

```
diff --git a/net/netfilter/core.c b/net/netfilter/core.c
index c4065b8..ec05684 100644
--- a/net/netfilter/core.c
+++ b/net/netfilter/core.c
@@ -165,6 +165,14 @@ int nf_hook_slow(int pf, unsigned int hook, struct sk_buff *skb,
  unsigned int verdict;
  int ret = 0;

+#ifdef CONFIG_NET_NS
+ struct net *net;
+
+ net = indev == NULL ? outdev->nd_net : indev->nd_net;
+ if (net != &init_net)
+  return 1;
+#endif
+
  /* We may already have this, but read-locks nest anyway */
  rcu_read_lock();

--
1.5.3.rc5
```

Subject: [PATCH 5/11 net-2.6.26] [NETNS]: Add namespace parameter to ip_cmsg_send.
Posted by den on Mon, 24 Mar 2008 14:36:06 GMT
View Forum Message <> Reply to Message

Pass the init_net there for now.

Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 include/net/ip.h | 3 ++-
 net/ipv4/ip_sockglue.c | 4 ++--
 net/ipv4/raw.c | 2 +-
 net/ipv4/udp.c | 2 +-
 4 files changed, 6 insertions(+), 5 deletions(-)

```
diff --git a/include/net/ip.h b/include/net/ip.h
index 531270d..6d7bcd5 100644
--- a/include/net/ip.h
+++ b/include/net/ip.h
@@ -362,7 +362,8 @@ extern int ip_options_rcv_srr(struct sk_buff *skb);
  */

 extern void ip_cmsg_recv(struct msghdr *msg, struct sk_buff *skb);
-extern int ip_cmsg_send(struct msghdr *msg, struct ipcm_cookie *ipc);
+extern int ip_cmsg_send(struct net *net,
+        struct msghdr *msg, struct ipcm_cookie *ipc);
 extern int ip_setsockopt(struct sock *sk, int level, int optname, char __user *optval, int optlen);
 extern int ip_getsockopt(struct sock *sk, int level, int optname, char __user *optval, int __user
*optlen);
 extern int compat_ip_setsockopt(struct sock *sk, int level,
diff --git a/net/ipv4/ip_sockglue.c b/net/ipv4/ip_sockglue.c
index 1b86a50..0857f2d 100644
--- a/net/ipv4/ip_sockglue.c
+++ b/net/ipv4/ip_sockglue.c
@@ -163,7 +163,7 @@ void ip_cmsg_recv(struct msghdr *msg, struct sk_buff *skb)
   ip_cmsg_recv_security(msg, skb);
 }

-int ip_cmsg_send(struct msghdr *msg, struct ipcm_cookie *ipc)
+int ip_cmsg_send(struct net *net, struct msghdr *msg, struct ipcm_cookie *ipc)
 {
  int err;
  struct cmsghdr *cmsg;
@@ -176,7 +176,7 @@ int ip_cmsg_send(struct msghdr *msg, struct ipcm_cookie *ipc)
   switch (cmsg->cmsg_type) {
   case IP_RETOPTS:
    err = cmsg->cmsg_len - CMSG_ALIGN(sizeof(struct cmsghdr));
-   err = ip_options_get(&init_net, &ipc->opt, CMSG_DATA(cmsg), err < 40 ? err : 40);
+   err = ip_options_get(net, &ipc->opt, CMSG_DATA(cmsg), err < 40 ? err : 40);
```

```
    if (err)
     return err;
    break;
diff --git a/net/ipv4/raw.c b/net/ipv4/raw.c
index b433b48..7d29a05 100644
--- a/net/ipv4/raw.c
+++ b/net/ipv4/raw.c
@@ -499,7 +499,7 @@ static int raw_sendmsg(struct kiocb *iocb, struct sock *sk, struct msghdr
*msg,
  ipc.oif = sk->sk_bound_dev_if;

  if (msg->msg_controllen) {
-  err = ip_cmsg_send(msg, &ipc);
+  err = ip_cmsg_send(&init_net, msg, &ipc);
   if (err)
    goto out;
   if (ipc.opt)
diff --git a/net/ipv4/udp.c b/net/ipv4/udp.c
index 8c1f5ea..9d69e9d 100644
--- a/net/ipv4/udp.c
+++ b/net/ipv4/udp.c
@@ -607,7 +607,7 @@ int udp_sendmsg(struct kiocb *iocb, struct sock *sk, struct msghdr *msg,

  ipc.oif = sk->sk_bound_dev_if;
  if (msg->msg_controllen) {
-  err = ip_cmsg_send(msg, &ipc);
+  err = ip_cmsg_send(&init_net, msg, &ipc);
   if (err)
    return err;
   if (ipc.opt)
--
1.5.3.rc5
```

Subject: [PATCH 6/11 net-2.6.26] [NETNS]: Process IP layer in the context of the correct namespace.

Posted by den on Mon, 24 Mar 2008 14:36:07 GMT

View Forum Message <> Reply to Message

Replace all the rest of the init_net with a proper net on the IP layer.

Signed-off-by: Denis V. Lunev <den@openvz.org>

```
---
 net/ipv4/ip_fragment.c |    5 ++++-
 net/ipv4/ip_input.c    |    6 ++++--
 net/ipv4/ip_options.c  |    2 +-
 net/ipv4/ip_output.c   |    2 +-
 net/ipv4/ip_sockglue.c |    7 ++++---
```

```
 5 files changed, 14 insertions(+), 8 deletions(-)

diff --git a/net/ipv4/ip_fragment.c b/net/ipv4/ip_fragment.c
index 3b2e5ad..8b448c4 100644
--- a/net/ipv4/ip_fragment.c
+++ b/net/ipv4/ip_fragment.c
@@ -204,8 +204,11 @@ static void ip_expire(unsigned long arg)

  if ((qp->q.last_in&FIRST_IN) && qp->q.fragments != NULL) {
   struct sk_buff *head = qp->q.fragments;
+  struct net *net;
+
+  net = container_of(qp->q.net, struct net, ipv4.frags);
   /* Send an ICMP "Fragment Reassembly Timeout" message. */
-  if ((head->dev = dev_get_by_index(&init_net, qp->iif)) != NULL) {
+  if ((head->dev = dev_get_by_index(net, qp->iif)) != NULL) {
    icmp_send(head, ICMP_TIME_EXCEEDED, ICMP_EXC_FRAGTIME, 0);
    dev_put(head->dev);
   }
diff --git a/net/ipv4/ip_input.c b/net/ipv4/ip_input.c
index f3a7a08..eb1fa27 100644
--- a/net/ipv4/ip_input.c
+++ b/net/ipv4/ip_input.c
@@ -160,6 +160,7 @@ int ip_call_ra_chain(struct sk_buff *skb)
  struct ip_ra_chain *ra;
  u8 protocol = ip_hdr(skb)->protocol;
  struct sock *last = NULL;
+ struct net_device *dev = skb->dev;

  read_lock(&ip_ra_lock);
  for (ra = ip_ra_chain; ra; ra = ra->next) {
@@ -170,7 +171,8 @@ int ip_call_ra_chain(struct sk_buff *skb)
    */
   if (sk && inet_sk(sk)->num == protocol &&
       (!sk->sk_bound_dev_if ||
-       sk->sk_bound_dev_if == skb->dev->ifindex)) {
+       sk->sk_bound_dev_if == dev->ifindex) &&
+      sk->sk_net == dev->nd_net) {
    if (ip_hdr(skb)->frag_off & htons(IP_MF | IP_OFFSET)) {
     if (ip_defrag(skb, IP_DEFRAG_CALL_RA_CHAIN)) {
      read_unlock(&ip_ra_lock);
@@ -286,7 +288,7 @@ static inline int ip_rcv_options(struct sk_buff *skb)
  opt = &(IPCB(skb)->opt);
  opt->optlen = iph->ihl*4 - sizeof(struct iphdr);

- if (ip_options_compile(&init_net, opt, skb)) {
+ if (ip_options_compile(dev->nd_net, opt, skb)) {
   IP_INC_STATS_BH(IPSTATS_MIB_INHDRERRORS);
```

```
   goto drop;
  }
diff --git a/net/ipv4/ip_options.c b/net/ipv4/ip_options.c
index 59f7ddf..87cc122 100644
--- a/net/ipv4/ip_options.c
+++ b/net/ipv4/ip_options.c
@@ -145,7 +145,7 @@ int ip_options_echo(struct ip_options * dopt, struct sk_buff * skb)
        __be32 addr;

        memcpy(&addr, sptr+soffset-1, 4);
-       if (inet_addr_type(&init_net, addr) != RTN_LOCAL) {
+       if (inet_addr_type(skb->dst->dev->nd_net, addr) != RTN_LOCAL) {
         dopt->ts_needtime = 1;
         soffset += 8;
        }
diff --git a/net/ipv4/ip_output.c b/net/ipv4/ip_output.c
index dc494ea..349fae5 100644
--- a/net/ipv4/ip_output.c
+++ b/net/ipv4/ip_output.c
@@ -351,7 +351,7 @@ int ip_queue_xmit(struct sk_buff *skb, int ipfragok)
      * itself out.
      */
     security_sk_classify_flow(sk, &fl);
-    if (ip_route_output_flow(&init_net, &rt, &fl, sk, 0))
+    if (ip_route_output_flow(sk->sk_net, &rt, &fl, sk, 0))
      goto no_route;
   }
   sk_setup_caps(sk, &rt->u.dst);
diff --git a/net/ipv4/ip_sockglue.c b/net/ipv4/ip_sockglue.c
index 0857f2d..b854431 100644
--- a/net/ipv4/ip_sockglue.c
+++ b/net/ipv4/ip_sockglue.c
@@ -449,7 +449,8 @@ static int do_ip_setsockopt(struct sock *sk, int level,
   struct ip_options * opt = NULL;
   if (optlen > 40 || optlen < 0)
    goto e_inval;
-  err = ip_options_get_from_user(&init_net, &opt, optval, optlen);
+  err = ip_options_get_from_user(sk->sk_net, &opt,
+           optval, optlen);
   if (err)
    break;
   if (inet->is_icsk) {
@@ -589,13 +590,13 @@ static int do_ip_setsockopt(struct sock *sk, int level,
     err = 0;
     break;
    }
-   dev = ip_dev_find(&init_net, mreq.imr_address.s_addr);
+   dev = ip_dev_find(sk->sk_net, mreq.imr_address.s_addr);
```

```
    if (dev) {
     mreq.imr_ifindex = dev->ifindex;
     dev_put(dev);
    }
   } else
-   dev = __dev_get_by_index(&init_net, mreq.imr_ifindex);
+   dev = __dev_get_by_index(sk->sk_net, mreq.imr_ifindex);


   err = -EADDRNOTAVAIL;
--
1.5.3.rc5
```

Subject: [PATCH 7/11 net-2.6.26] [NETNS]: Process INET socket layer in the correct namespace.

Posted by den on Mon, 24 Mar 2008 14:36:08 GMT

View Forum Message <> Reply to Message

Replace all the reast of the init_net with a proper net on the socket layer.

Signed-off-by: Denis V. Lunev <den@openvz.org>

```
---
 net/ipv4/af_inet.c             |    4 ++--
 net/ipv4/inet_connection_sock.c |    2 +-
 net/ipv4/raw.c                 |    4 ++--
 net/ipv4/udp.c                 |    4 ++--
 4 files changed, 7 insertions(+), 7 deletions(-)

diff --git a/net/ipv4/af_inet.c b/net/ipv4/af_inet.c
index 44f5ce1..a9e241c 100644
--- a/net/ipv4/af_inet.c
+++ b/net/ipv4/af_inet.c
@@ -446,7 +446,7 @@ int inet_bind(struct socket *sock, struct sockaddr *uaddr, int addr_len)
  if (addr_len < sizeof(struct sockaddr_in))
   goto out;

- chk_addr_ret = inet_addr_type(&init_net, addr->sin_addr.s_addr);
+ chk_addr_ret = inet_addr_type(sk->sk_net, addr->sin_addr.s_addr);

  /* Not specified by any standard per-se, however it breaks too
   * many applications when removed.  It is unfortunate since
@@ -1114,7 +1114,7 @@ int inet_sk_rebuild_header(struct sock *sk)
  };

  security_sk_classify_flow(sk, &fl);
- err = ip_route_output_flow(&init_net, &rt, &fl, sk, 0);
+ err = ip_route_output_flow(sk->sk_net, &rt, &fl, sk, 0);
```

```
 }
  if (!err)
   sk_setup_caps(sk, &rt->u.dst);
diff --git a/net/ipv4/inet_connection_sock.c b/net/ipv4/inet_connection_sock.c
index f9c5c4d..d13c5f1 100644
--- a/net/ipv4/inet_connection_sock.c
+++ b/net/ipv4/inet_connection_sock.c
@@ -333,7 +333,7 @@ struct dst_entry* inet_csk_route_req(struct sock *sk,
       .dport = ireq->rmt_port } } };

  security_req_classify_flow(req, &fl);
- if (ip_route_output_flow(&init_net, &rt, &fl, sk, 0)) {
+ if (ip_route_output_flow(sk->sk_net, &rt, &fl, sk, 0)) {
   IP_INC_STATS_BH(IPSTATS_MIB_OUTNOROUTES);
   return NULL;
  }
diff --git a/net/ipv4/raw.c b/net/ipv4/raw.c
index 7d29a05..3f68a93 100644
--- a/net/ipv4/raw.c
+++ b/net/ipv4/raw.c
@@ -499,7 +499,7 @@ static int raw_sendmsg(struct kiocb *iocb, struct sock *sk, struct msghdr
*msg,
  ipc.oif = sk->sk_bound_dev_if;

  if (msg->msg_controllen) {
-  err = ip_cmsg_send(&init_net, msg, &ipc);
+  err = ip_cmsg_send(sk->sk_net, msg, &ipc);
   if (err)
    goto out;
   if (ipc.opt)
@@ -553,7 +553,7 @@ static int raw_sendmsg(struct kiocb *iocb, struct sock *sk, struct msghdr
*msg,
   }

   security_sk_classify_flow(sk, &fl);
-  err = ip_route_output_flow(&init_net, &rt, &fl, sk, 1);
+  err = ip_route_output_flow(sk->sk_net, &rt, &fl, sk, 1);
  }
  if (err)
   goto done;
diff --git a/net/ipv4/udp.c b/net/ipv4/udp.c
index 9d69e9d..1f48687 100644
--- a/net/ipv4/udp.c
+++ b/net/ipv4/udp.c
@@ -607,7 +607,7 @@ int udp_sendmsg(struct kiocb *iocb, struct sock *sk, struct msghdr *msg,

  ipc.oif = sk->sk_bound_dev_if;
  if (msg->msg_controllen) {
```

```
-  err = ip_cmsg_send(&init_net, msg, &ipc);
+  err = ip_cmsg_send(sk->sk_net, msg, &ipc);
   if (err)
    return err;
   if (ipc.opt)
@@ -656,7 +656,7 @@ int udp_sendmsg(struct kiocb *iocb, struct sock *sk, struct msghdr *msg,
            { .sport = inet->sport,
        .dport = dport } } };
   security_sk_classify_flow(sk, &fl);
-  err = ip_route_output_flow(&init_net, &rt, &fl, sk, 1);
+  err = ip_route_output_flow(sk->sk_net, &rt, &fl, sk, 1);
   if (err) {
    if (err == -ENETUNREACH)
     IP_INC_STATS_BH(IPSTATS_MIB_OUTNOROUTES);
--
1.5.3.rc5
```

Subject: [PATCH 9/11 net-2.6.26] [NETNS]: Drop packets in the non-initial namespace on the per/protocol basis

Posted by den on Mon, 24 Mar 2008 14:36:10 GMT

View Forum Message <> Reply to Message

IP layer now can handle multiple namespaces normally. So, process such packets normally and drop them only if the transport layer is not aware about namespaces.

Signed-off-by: Denis V. Lunev <den@openvz.org>

```
---
 include/net/protocol.h |    3 ++-
 net/ipv4/ip_input.c    |    8 ++++----
 2 files changed, 6 insertions(+), 5 deletions(-)

diff --git a/include/net/protocol.h b/include/net/protocol.h
index ad8c584..8d024d7 100644
--- a/include/net/protocol.h
+++ b/include/net/protocol.h
@@ -39,7 +39,8 @@ struct net_protocol {
  int   (*gso_send_check)(struct sk_buff *skb);
  struct sk_buff        *(*gso_segment)(struct sk_buff *skb,
            int features);
- int   no_policy;
+ unsigned int  no_policy:1,
+    netns_ok:1;
 };

 #if defined(CONFIG_IPV6) || defined (CONFIG_IPV6_MODULE)
diff --git a/net/ipv4/ip_input.c b/net/ipv4/ip_input.c
```

```
index eb1fa27..2aeea5d 100644
--- a/net/ipv4/ip_input.c
+++ b/net/ipv4/ip_input.c
@@ -199,6 +199,8 @@ int ip_call_ra_chain(struct sk_buff *skb)

 static int ip_local_deliver_finish(struct sk_buff *skb)
 {
+ struct net *net = skb->dev->nd_net;
+
   __skb_pull(skb, ip_hdrlen(skb));

   /* Point into the IP datagram, just past the header. */
@@ -214,7 +216,8 @@ static int ip_local_deliver_finish(struct sk_buff *skb)
    raw = raw_local_deliver(skb, protocol);

    hash = protocol & (MAX_INET_PROTOS - 1);
-   if ((ipprot = rcu_dereference(inet_protos[hash])) != NULL) {
+   ipprot = rcu_dereference(inet_protos[hash]);
+   if (ipprot != NULL && (net == &init_net || ipprot->netns_ok)) {
     int ret;

     if (!ipprot->no_policy) {
@@ -375,9 +378,6 @@ int ip_rcv(struct sk_buff *skb, struct net_device *dev, struct packet_type
*pt,
   struct iphdr *iph;
   u32 len;

- if (dev->nd_net != &init_net)
-   goto drop;
-
   /* When the interface is in promisc. mode, drop all the crap
    * that it receives, do not try to analyse it.
    */
--
1.5.3.rc5
```

Subject: [PATCH 11/11 net-2.6.25] [NETNS]: Enable TCP/UDP/ICMP inside namespace.
Posted by den on Mon, 24 Mar 2008 14:36:12 GMT
View Forum Message <> Reply to Message

```
Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 net/ipv4/af_inet.c |    3 +++
 net/ipv4/udplite.c |    1 +
 2 files changed, 4 insertions(+), 0 deletions(-)
```

```
diff --git a/net/ipv4/af_inet.c b/net/ipv4/af_inet.c
index 39ddb84..06cfb0b 100644
--- a/net/ipv4/af_inet.c
+++ b/net/ipv4/af_inet.c
@@ -1302,17 +1302,20 @@ static struct net_protocol tcp_protocol = {
  .gso_send_check = tcp_v4_gso_send_check,
  .gso_segment = tcp_tso_segment,
  .no_policy = 1,
+ .netns_ok = 1,
 };

 static struct net_protocol udp_protocol = {
  .handler = udp_rcv,
  .err_handler = udp_err,
  .no_policy = 1,
+ .netns_ok = 1,
 };

 static struct net_protocol icmp_protocol = {
  .handler = icmp_rcv,
  .no_policy = 1,
+ .netns_ok = 1,
 };

 static int __init init_ipv4_mibs(void)
diff --git a/net/ipv4/udplite.c b/net/ipv4/udplite.c
index 8d42e34..e61ab2b 100644
--- a/net/ipv4/udplite.c
+++ b/net/ipv4/udplite.c
@@ -31,6 +31,7 @@ static struct net_protocol udplite_protocol = {
  .handler = udplite_rcv,
  .err_handler = udplite_err,
  .no_policy = 1,
+ .netns_ok = 1,
 };

 DEFINE_PROTO_INUSE(udplite)
--
1.5.3.rc5
```

Subject: Re: [PATCH 0/11 net-2.6.26] UDP/ICMP/TCP for a namespace
Posted by davem on Mon, 24 Mar 2008 22:44:19 GMT
View Forum Message <> Reply to Message

From: "Denis V. Lunev" <den@openvz.org>
Date: Mon, 24 Mar 2008 17:33:05 +0300

> This patch set finally enables TCP/UDP and ICMP inside a namespace.
> In order to do this we fix ARP processing, IP options processing and
> allow packets to flow to namespace aware protocols.
>
> Finally, this makes namespace functional and alive :)
>
> Signed-off-by: Denis V. Lunev <den@openvz.org>

Thanks for all of your hard work, applied.

I'll push this to my net-2.6.26 tree after some build
sanity checking.
