Subject: [PATCH 0/12 net-2.6.25] [NETNS]: Routing namespacing on IP output path.
Posted by den on Tue, 22 Jan 2008 15:58:54 GMT
View Forum Message <> Reply to Message

This set introduces namespacing in the IP output path. The namespace is
added to all routing API functions except ones with a valid socket. This
is very intrusive.

Routing cache is virtualized as a part of this efforts, though the hash
function is not tuned to use namespace id. This not required to work in
initial namespace.

ICMP replies now also use correct namespace.

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

Subject: [PATCH 1/12 net-2.6.25] [IPV4]: Declarations cleanup in ip_fib.h.
Posted by den on Tue, 22 Jan 2008 15:59:18 GMT
View Forum Message <> Reply to Message

Two small issues fixed:
- fib_select_multipath is exported from fib_semantics.c rather than from
  fib_frontend.c. So, move the declaration below appropriate comment.
- struct rt_entry declaration is not used. Drop it.

Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 include/net/ip_fib.h | 4 +---
 1 files changed, 1 insertions(+), 3 deletions(-)

diff --git a/include/net/ip_fib.h b/include/net/ip_fib.h
index a859124..be70b33 100644
--- a/include/net/ip_fib.h
+++ b/include/net/ip_fib.h
@@ -222,15 +222,13 @@ extern const struct nla_policy rtm_ipv4_policy[];
 extern void ip_fib_init(void);
 extern int fib_validate_source(__be32 src, __be32 dst, u8 tos, int oif,
          struct net_device *dev, __be32 *spec_dst, u32 *itag);
-extern void fib_select_multipath(const struct flowi *flp, struct fib_result *res);
-
-struct rtentry;

 /* Exported by fib_semantics.c */
 extern int ip_fib_check_default(__be32 gw, struct net_device *dev);
 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);
+extern void fib_select_multipath(const struct flowi *flp, struct fib_result *res);

 /* Exported by fib_{hash|trie}.c */
 extern void fib_hash_init(void);
--
1.5.3.rc5
```

Subject: [PATCH 2/12 net-2.6.25] [IPV4]: Consolidate fib_select_default.
Posted by den on Tue, 22 Jan 2008 15:59:19 GMT
View Forum Message <> Reply to Message

The difference in the implementation of the fib_select_default when
CONFIG_IP_MULTIPLE_TABLES is (not) defined looks negligible. Consolidate it
and place into fib_frontend.c.

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

```
---
 include/net/ip_fib.h    |   10 +---------
 net/ipv4/fib_frontend.c |   14 ++++++++++++++
 net/ipv4/fib_rules.c    |   10 ----------
 3 files changed, 15 insertions(+), 19 deletions(-)

diff --git a/include/net/ip_fib.h b/include/net/ip_fib.h
index be70b33..39f944a 100644
--- a/include/net/ip_fib.h
+++ b/include/net/ip_fib.h
@@ -193,14 +193,6 @@ static inline int fib_lookup(struct net *net, const struct flowi *flp,
  return -ENETUNREACH;
 }

-static inline void fib_select_default(const struct flowi *flp,
-         struct fib_result *res)
-{
- struct fib_table *table = fib_get_table(&init_net, RT_TABLE_MAIN);
- if (FIB_RES_GW(*res) && FIB_RES_NH(*res).nh_scope == RT_SCOPE_LINK)
-  table->tb_select_default(table, flp, res);
-}
-
 #else /* CONFIG_IP_MULTIPLE_TABLES */
 extern int __net_init fib4_rules_init(struct net *net);
 extern void __net_exit fib4_rules_exit(struct net *net);
@@ -213,7 +205,6 @@ extern int fib_lookup(struct net *n, struct flowi *flp, struct fib_result *res);

 extern struct fib_table *fib_new_table(struct net *net, u32 id);
 extern struct fib_table *fib_get_table(struct net *net, u32 id);
-extern void fib_select_default(const struct flowi *flp, struct fib_result *res);
```

```
 #endif /* CONFIG_IP_MULTIPLE_TABLES */

@@ -222,6 +213,7 @@ extern const struct nla_policy rtm_ipv4_policy[];
 extern void  ip_fib_init(void);
 extern int fib_validate_source(__be32 src, __be32 dst, u8 tos, int oif,
          struct net_device *dev, __be32 *spec_dst, u32 *itag);
+extern void fib_select_default(const struct flowi *flp, struct fib_result *res);

 /* Exported by fib_semantics.c */
 extern int ip_fib_check_default(__be32 gw, struct net_device *dev);
diff --git a/net/ipv4/fib_frontend.c b/net/ipv4/fib_frontend.c
index 6761639..13bf01d 100644
--- a/net/ipv4/fib_frontend.c
+++ b/net/ipv4/fib_frontend.c
@@ -116,6 +116,20 @@ struct fib_table *fib_get_table(struct net *net, u32 id)
 }
 #endif /* CONFIG_IP_MULTIPLE_TABLES */

+void fib_select_default(const struct flowi *flp, struct fib_result *res)
+{
+ struct fib_table *tb;
+ int table = RT_TABLE_MAIN;
+#ifdef CONFIG_IP_MULTIPLE_TABLES
+ if (res->r == NULL || res->r->action != FR_ACT_TO_TBL)
+  return;
+ table = res->r->table;
+#endif
+ tb = fib_get_table(&init_net, table);
+ if (FIB_RES_GW(*res) && FIB_RES_NH(*res).nh_scope == RT_SCOPE_LINK)
+  tb->tb_select_default(tb, flp, res);
+}
+
 static void fib_flush(struct net *net)
 {
  int flushed = 0;
diff --git a/net/ipv4/fib_rules.c b/net/ipv4/fib_rules.c
index 1effb4a..19274d0 100644
--- a/net/ipv4/fib_rules.c
+++ b/net/ipv4/fib_rules.c
@@ -102,16 +102,6 @@ errout:
 }


-void fib_select_default(const struct flowi *flp, struct fib_result *res)
-{
- if (res->r && res->r->action == FR_ACT_TO_TBL &&
-     FIB_RES_GW(*res) && FIB_RES_NH(*res).nh_scope == RT_SCOPE_LINK) {
```

```
-  struct fib_table *tb;
-  if ((tb = fib_get_table(&init_net, res->r->table)) != NULL)
-   tb->tb_select_default(tb, flp, res);
- }
-}
-
 static int fib4_rule_match(struct fib_rule *rule, struct flowi *fl, int flags)
 {
  struct fib4_rule *r = (struct fib4_rule *) rule;
--
1.5.3.rc5
```

Subject: [PATCH 3/12 net-2.6.25] [NETNS]: Add netns parameter to fib_select_default.

Posted by den on Tue, 22 Jan 2008 15:59:20 GMT

View Forum Message <> Reply to Message

Currently fib_select_default calls fib_get_table() with the init_net. Prepare
it to provide a correct namespace to lookup default route.

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

```
---
 include/net/ip_fib.h    |    3 ++-
 net/ipv4/fib_frontend.c |    5 +++--
 net/ipv4/route.c        |    2 +-
 3 files changed, 6 insertions(+), 4 deletions(-)

diff --git a/include/net/ip_fib.h b/include/net/ip_fib.h
index 39f944a..9daa60b 100644
--- a/include/net/ip_fib.h
+++ b/include/net/ip_fib.h
@@ -213,7 +213,8 @@ extern const struct nla_policy rtm_ipv4_policy[];
 extern void  ip_fib_init(void);
 extern int fib_validate_source(__be32 src, __be32 dst, u8 tos, int oif,
           struct net_device *dev, __be32 *spec_dst, u32 *itag);
-extern void fib_select_default(const struct flowi *flp, struct fib_result *res);
+extern void fib_select_default(struct net *net, const struct flowi *flp,
+          struct fib_result *res);

 /* Exported by fib_semantics.c */
 extern int ip_fib_check_default(__be32 gw, struct net_device *dev);
diff --git a/net/ipv4/fib_frontend.c b/net/ipv4/fib_frontend.c
index 13bf01d..7e3e732 100644
--- a/net/ipv4/fib_frontend.c
+++ b/net/ipv4/fib_frontend.c
@@ -116,7 +116,8 @@ struct fib_table *fib_get_table(struct net *net, u32 id)
 }
```

```
 #endif /* CONFIG_IP_MULTIPLE_TABLES */

-void fib_select_default(const struct flowi *flp, struct fib_result *res)
+void fib_select_default(struct net *net,
+    const struct flowi *flp, struct fib_result *res)
 {
  struct fib_table *tb;
  int table = RT_TABLE_MAIN;
@@ -125,7 +126,7 @@ void fib_select_default(const struct flowi *flp, struct fib_result *res)
   return;
  table = res->r->table;
 #endif
- tb = fib_get_table(&init_net, table);
+ tb = fib_get_table(net, table);
  if (FIB_RES_GW(*res) && FIB_RES_NH(*res).nh_scope == RT_SCOPE_LINK)
   tb->tb_select_default(tb, flp, res);
 }
diff --git a/net/ipv4/route.c b/net/ipv4/route.c
index 27e0f81..4313255 100644
--- a/net/ipv4/route.c
+++ b/net/ipv4/route.c
@@ -2419,7 +2419,7 @@ static int ip_route_output_slow(struct rtable **rp, const struct flowi
*oldflp)
  else
 #endif
  if (!res.prefixlen && res.type == RTN_UNICAST && !fl.oif)
-  fib_select_default(&fl, &res);
+  fib_select_default(&init_net, &fl, &res);

  if (!fl.fl4_src)
   fl.fl4_src = FIB_RES_PREFSRC(res);
--
1.5.3.rc5
```

Subject: [PATCH 4/12 net-2.6.25] [NETNS]: Add namespace parameter to ip_dev_find.

Posted by den on Tue, 22 Jan 2008 15:59:21 GMT

View Forum Message <> Reply to Message

in_dev_find() need a namespace to pass it to fib_get_table(), so add an argument.

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

```
---
 drivers/infiniband/core/addr.c |    4 ++--
 drivers/infiniband/core/cma.c  |    2 +-
 include/linux/inetdevice.h     |    2 +-
```

```
 net/ipv4/fib_frontend.c       |    4 ++--
 net/ipv4/igmp.c               |    2 +-
 net/ipv4/ip_sockglue.c        |    2 +-
 net/ipv4/ipmr.c               |    2 +-
 net/ipv4/route.c              |    6 +++---
 8 files changed, 12 insertions(+), 12 deletions(-)

diff --git a/drivers/infiniband/core/addr.c b/drivers/infiniband/core/addr.c
index 0802b79..963177e 100644
--- a/drivers/infiniband/core/addr.c
+++ b/drivers/infiniband/core/addr.c
@@ -110,7 +110,7 @@ int rdma_translate_ip(struct sockaddr *addr, struct rdma_dev_addr
*dev_addr)
  __be32 ip = ((struct sockaddr_in *) addr)->sin_addr.s_addr;
  int ret;

- dev = ip_dev_find(ip);
+ dev = ip_dev_find(&init_net, ip);
  if (!dev)
   return -EADDRNOTAVAIL;

@@ -261,7 +261,7 @@ static int addr_resolve_local(struct sockaddr_in *src_in,
  __be32 dst_ip = dst_in->sin_addr.s_addr;
  int ret;

- dev = ip_dev_find(dst_ip);
+ dev = ip_dev_find(&init_net, dst_ip);
  if (!dev)
   return -EADDRNOTAVAIL;

diff --git a/drivers/infiniband/core/cma.c b/drivers/infiniband/core/cma.c
index b37045c..ef9efb3 100644
--- a/drivers/infiniband/core/cma.c
+++ b/drivers/infiniband/core/cma.c
@@ -1280,7 +1280,7 @@ static int iw_conn_req_handler(struct iw_cm_id *cm_id,
  atomic_inc(&conn_id->dev_remove);
  conn_id->state = CMA_CONNECT;

- dev = ip_dev_find(iw_event->local_addr.sin_addr.s_addr);
+ dev = ip_dev_find(&init_net, iw_event->local_addr.sin_addr.s_addr);
  if (!dev) {
   ret = -EADDRNOTAVAIL;
   cma_enable_remove(conn_id);
diff --git a/include/linux/inetdevice.h b/include/linux/inetdevice.h
index e74a2ee..8d9eaae 100644
--- a/include/linux/inetdevice.h
+++ b/include/linux/inetdevice.h
@@ -129,7 +129,7 @@ struct in_ifaddr
```

```
 extern int register_inetaddr_notifier(struct notifier_block *nb);
 extern int unregister_inetaddr_notifier(struct notifier_block *nb);

-extern struct net_device  *ip_dev_find(__be32 addr);
+extern struct net_device *ip_dev_find(struct net *net, __be32 addr);
 extern int  inet_addr_onlink(struct in_device *in_dev, __be32 a, __be32 b);
 extern int  devinet_ioctl(unsigned int cmd, void __user *);
 extern void  devinet_init(void);
diff --git a/net/ipv4/fib_frontend.c b/net/ipv4/fib_frontend.c
index 7e3e732..d282618 100644
--- a/net/ipv4/fib_frontend.c
+++ b/net/ipv4/fib_frontend.c
@@ -153,7 +153,7 @@ static void fib_flush(struct net *net)
  * Find the first device with a given source address.
  */

-struct net_device * ip_dev_find(__be32 addr)
+struct net_device * ip_dev_find(struct net *net, __be32 addr)
 {
  struct flowi fl = { .nl_u = { .ip4_u = { .daddr = addr } } };
  struct fib_result res;
@@ -164,7 +164,7 @@ struct net_device * ip_dev_find(__be32 addr)
  res.r = NULL;
 #endif

- local_table = fib_get_table(&init_net, RT_TABLE_LOCAL);
+ local_table = fib_get_table(net, RT_TABLE_LOCAL);
  if (!local_table || local_table->tb_lookup(local_table, &fl, &res))
   return NULL;
  if (res.type != RTN_LOCAL)
diff --git a/net/ipv4/igmp.c b/net/ipv4/igmp.c
index 928bc32..1f5314c 100644
--- a/net/ipv4/igmp.c
+++ b/net/ipv4/igmp.c
@@ -1395,7 +1395,7 @@ static struct in_device * ip_mc_find_dev(struct ip_mreqn *imr)
   return idev;
  }
  if (imr->imr_address.s_addr) {
-  dev = ip_dev_find(imr->imr_address.s_addr);
+  dev = ip_dev_find(&init_net, imr->imr_address.s_addr);
   if (!dev)
    return NULL;
   dev_put(dev);
diff --git a/net/ipv4/ip_sockglue.c b/net/ipv4/ip_sockglue.c
index 82817e5..754b0a5 100644
--- a/net/ipv4/ip_sockglue.c
+++ b/net/ipv4/ip_sockglue.c
@@ -594,7 +594,7 @@ static int do_ip_setsockopt(struct sock *sk, int level,
```

```
     err = 0;
     break;
    }
-   dev = ip_dev_find(mreq.imr_address.s_addr);
+   dev = ip_dev_find(&init_net, mreq.imr_address.s_addr);
    if (dev) {
     mreq.imr_ifindex = dev->ifindex;
     dev_put(dev);
diff --git a/net/ipv4/ipmr.c b/net/ipv4/ipmr.c
index 4198615..2212717 100644
--- a/net/ipv4/ipmr.c
+++ b/net/ipv4/ipmr.c
@@ -423,7 +423,7 @@ static int vif_add(struct vifctl *vifc, int mrtsock)
    return -ENOBUFS;
   break;
  case 0:
-  dev = ip_dev_find(vifc->vifc_lcl_addr.s_addr);
+  dev = ip_dev_find(&init_net, vifc->vifc_lcl_addr.s_addr);
   if (!dev)
    return -EADDRNOTAVAIL;
   dev_put(dev);
diff --git a/net/ipv4/route.c b/net/ipv4/route.c
index 4313255..674575b 100644
--- a/net/ipv4/route.c
+++ b/net/ipv4/route.c
@@ -2282,14 +2282,14 @@ static int ip_route_output_slow(struct rtable **rp, const struct flowi
*oldflp)
    goto out;

   /* It is equivalent to inet_addr_type(saddr) == RTN_LOCAL */
-  dev_out = ip_dev_find(oldflp->fl4_src);
+  dev_out = ip_dev_find(&init_net, oldflp->fl4_src);
   if (dev_out == NULL)
    goto out;

   /* I removed check for oif == dev_out->oif here.
      It was wrong for two reasons:
-     1. ip_dev_find(saddr) can return wrong iface, if saddr is
-        assigned to multiple interfaces.
+     1. ip_dev_find(net, saddr) can return wrong iface, if saddr
+        is assigned to multiple interfaces.
      2. Moreover, we are allowed to send packets with saddr
         of another iface. --ANK
    */
--
1.5.3.rc5
```

Subject: [PATCH 5/12 net-2.6.25] [NETNS]: Re-export init_net via EXPORT_SYMBOL.
Posted by den on Tue, 22 Jan 2008 15:59:22 GMT
View Forum Message <> Reply to Message

init_net is used added as a parameter to a lot of old API calls, f.e.
ip_dev_find. These calls were exported as EXPORT_SYMBOL. So, export init_net
as EXPORT_SYMBOL to keep networking API consistent.

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

diff --git a/net/core/net_namespace.c b/net/core/net_namespace.c
index 8023208..26e941d 100644
--- a/net/core/net_namespace.c
+++ b/net/core/net_namespace.c
@@ -18,7 +18,7 @@ static DEFINE_MUTEX(net_mutex);
 LIST_HEAD(net_namespace_list);

 struct net init_net;
-EXPORT_SYMBOL_GPL(init_net);
+EXPORT_SYMBOL(init_net);

 /*
  * setup_net runs the initializers for the network namespace object.
--
1.5.3.rc5

Subject: [PATCH 6/12 net-2.6.25] [NETNS]: Add namespace parameter to ip_route_output_slow.
Posted by den on Tue, 22 Jan 2008 15:59:23 GMT
View Forum Message <> Reply to Message

This function needs a net namespace to lookup devices, fib tables, etc. in,
so pass it there.

Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 net/ipv4/route.c | 22 ++++++++++----------
 1 files changed, 11 insertions(+), 10 deletions(-)

diff --git a/net/ipv4/route.c b/net/ipv4/route.c
index 674575b..c1f9950 100644
--- a/net/ipv4/route.c
+++ b/net/ipv4/route.c

```
@@ -2248,7 +2248,8 @@ static inline int ip_mkroute_output(struct rtable **rp,
  * Major route resolver routine.
  */

-static int ip_route_output_slow(struct rtable **rp, const struct flowi *oldflp)
+static int ip_route_output_slow(struct net *net, struct rtable **rp,
+    const struct flowi *oldflp)
 {
  u32 tos = RT_FL_TOS(oldflp);
  struct flowi fl = { .nl_u = { .ip4_u =
@@ -2260,7 +2261,7 @@ static int ip_route_output_slow(struct rtable **rp, const struct flowi
*oldflp)
          RT_SCOPE_UNIVERSE),
            } },
        .mark = oldflp->mark,
-       .iif = init_net.loopback_dev->ifindex,
+       .iif = net->loopback_dev->ifindex,
        .oif = oldflp->oif };
  struct fib_result res;
  unsigned flags = 0;
@@ -2282,7 +2283,7 @@ static int ip_route_output_slow(struct rtable **rp, const struct flowi
*oldflp)
    goto out;

   /* It is equivalent to inet_addr_type(saddr) == RTN_LOCAL */
-  dev_out = ip_dev_find(&init_net, oldflp->fl4_src);
+  dev_out = ip_dev_find(net, oldflp->fl4_src);
   if (dev_out == NULL)
    goto out;

@@ -2322,7 +2323,7 @@ static int ip_route_output_slow(struct rtable **rp, const struct flowi
*oldflp)


  if (oldflp->oif) {
-  dev_out = dev_get_by_index(&init_net, oldflp->oif);
+  dev_out = dev_get_by_index(net, oldflp->oif);
   err = -ENODEV;
   if (dev_out == NULL)
    goto out;
@@ -2356,15 +2357,15 @@ static int ip_route_output_slow(struct rtable **rp, const struct flowi
*oldflp)
    fl.fl4_dst = fl.fl4_src = htonl(INADDR_LOOPBACK);
   if (dev_out)
    dev_put(dev_out);
-  dev_out = init_net.loopback_dev;
+  dev_out = net->loopback_dev;
   dev_hold(dev_out);
```

```
-  fl.oif = init_net.loopback_dev->ifindex;
+  fl.oif = net->loopback_dev->ifindex;
   res.type = RTN_LOCAL;
   flags |= RTCF_LOCAL;
   goto make_route;
  }

- if (fib_lookup(&init_net, &fl, &res)) {
+ if (fib_lookup(net, &fl, &res)) {
   res.fi = NULL;
   if (oldflp->oif) {
    /* Apparently, routing tables are wrong. Assume,
@@ -2403,7 +2404,7 @@ static int ip_route_output_slow(struct rtable **rp, const struct flowi
*oldflp)
    fl.fl4_src = fl.fl4_dst;
   if (dev_out)
    dev_put(dev_out);
-  dev_out = init_net.loopback_dev;
+  dev_out = net->loopback_dev;
   dev_hold(dev_out);
   fl.oif = dev_out->ifindex;
   if (res.fi)
@@ -2419,7 +2420,7 @@ static int ip_route_output_slow(struct rtable **rp, const struct flowi
*oldflp)
  else
 #endif
  if (!res.prefixlen && res.type == RTN_UNICAST && !fl.oif)
-  fib_select_default(&init_net, &fl, &res);
+  fib_select_default(net, &fl, &res);

  if (!fl.fl4_src)
   fl.fl4_src = FIB_RES_PREFSRC(res);
@@ -2469,7 +2469,7 @@ int __ip_route_output_key(struct rtable **rp, const struct flowi *flp)
  }
  rcu_read_unlock_bh();

- return ip_route_output_slow(rp, flp);
+ return ip_route_output_slow(&init_net, rp, flp);
 }

 EXPORT_SYMBOL_GPL(__ip_route_output_key);
--
1.5.3.rc5
```

Subject: [PATCH 7/12 net-2.6.25] [NETNS]: Add namespace parameter to __ip_route_output_key.

This is only required to propagate it down to the ip_route_output_slow.

Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 include/net/route.h | 4 ++--
 net/ipv4/icmp.c | 4 ++--
 net/ipv4/route.c | 7 ++++---
 net/ipv4/xfrm4_policy.c | 2 +-
 4 files changed, 9 insertions(+), 8 deletions(-)

```
diff --git a/include/net/route.h b/include/net/route.h
index 5847e6f..3e3b14e 100644
--- a/include/net/route.h
+++ b/include/net/route.h
@@ -110,7 +110,7 @@ extern int  ip_rt_init(void);
 extern void  ip_rt_redirect(__be32 old_gw, __be32 dst, __be32 new_gw,
           __be32 src, struct net_device *dev);
 extern void  rt_cache_flush(int how);
-extern int  __ip_route_output_key(struct rtable **, const struct flowi *flp);
+extern int  __ip_route_output_key(struct net *, struct rtable **, const struct flowi *flp);
 extern int  ip_route_output_key(struct rtable **, struct flowi *flp);
 extern int  ip_route_output_flow(struct rtable **rp, struct flowi *flp, struct sock *sk, int flags);
 extern int  ip_route_input(struct sk_buff*, __be32 dst, __be32 src, u8 tos, struct net_device
*devin);
@@ -158,7 +158,7 @@ static inline int ip_route_connect(struct rtable **rp, __be32 dst,

  int err;
  if (!dst || !src) {
-  err = __ip_route_output_key(rp, &fl);
+  err = __ip_route_output_key(&init_net, rp, &fl);
   if (err)
    return err;
   fl.fl4_dst = (*rp)->rt_dst;
diff --git a/net/ipv4/icmp.c b/net/ipv4/icmp.c
index 7ed8c50..21422bf 100644
--- a/net/ipv4/icmp.c
+++ b/net/ipv4/icmp.c
@@ -569,7 +569,7 @@ void icmp_send(struct sk_buff *skb_in, int type, int code, __be32 info)
   struct rtable *rt2;

   security_skb_classify_flow(skb_in, &fl);
-  if (__ip_route_output_key(&rt, &fl))
+  if (__ip_route_output_key(&init_net, &rt, &fl))
    goto out_unlock;

   /* No need to clone since we're just using its address. */
```

```
@@ -592,7 +592,7 @@ void icmp_send(struct sk_buff *skb_in, int type, int code, __be32 info)
    goto out_unlock;

   if (inet_addr_type(&init_net, fl.fl4_src) == RTN_LOCAL)
-   err = __ip_route_output_key(&rt2, &fl);
+   err = __ip_route_output_key(&init_net, &rt2, &fl);
   else {
    struct flowi fl2 = {};
    struct dst_entry *odst;
diff --git a/net/ipv4/route.c b/net/ipv4/route.c
index c1f9950..cb035cc 100644
--- a/net/ipv4/route.c
+++ b/net/ipv4/route.c
@@ -2442,7 +2442,8 @@ make_route:
 out: return err;
 }

-int __ip_route_output_key(struct rtable **rp, const struct flowi *flp)
+int __ip_route_output_key(struct net *net, struct rtable **rp,
+     const struct flowi *flp)
 {
  unsigned hash;
  struct rtable *rth;
@@ -2469,7 +2470,7 @@ int __ip_route_output_key(struct rtable **rp, const struct flowi *flp)
  }
  rcu_read_unlock_bh();

- return ip_route_output_slow(&init_net, rp, flp);
+ return ip_route_output_slow(net, rp, flp);
 }

 EXPORT_SYMBOL_GPL(__ip_route_output_key);
@@ -2535,7 +2536,7 @@ int ip_route_output_flow(struct rtable **rp, struct flowi *flp, struct sock
*sk,
 {
  int err;

- if ((err = __ip_route_output_key(rp, flp)) != 0)
+ if ((err = __ip_route_output_key(&init_net, rp, flp)) != 0)
   return err;

  if (flp->proto) {
diff --git a/net/ipv4/xfrm4_policy.c b/net/ipv4/xfrm4_policy.c
index f04516c..3783e3e 100644
--- a/net/ipv4/xfrm4_policy.c
+++ b/net/ipv4/xfrm4_policy.c
@@ -36,7 +36,7 @@ static struct dst_entry *xfrm4_dst_lookup(int tos, xfrm_address_t *saddr,
  if (saddr)
```

```
   fl.fl4_src = saddr->a4;

- err = __ip_route_output_key(&rt, &fl);
+ err = __ip_route_output_key(&init_net, &rt, &fl);
  dst = &rt->u.dst;
  if (err)
   dst = ERR_PTR(err);
--
1.5.3.rc5
```

Subject: [PATCH 8/12 net-2.6.25] [NETNS]: Add namespace parameter to ip_route_output_flow.

Posted by den on Tue, 22 Jan 2008 15:59:25 GMT

View Forum Message <> Reply to Message

Needed to propagate it down to the __ip_route_output_key.

Signed_off_by: Denis V. Lunev <den@openvz.org>

```
---
 drivers/infiniband/hw/cxgb3/iwch_cm.c |    2 +-
 include/net/route.h                   |    6 +++---
 net/dccp/ipv4.c                       |    2 +-
 net/ipv4/af_inet.c                    |    2 +-
 net/ipv4/inet_connection_sock.c       |    2 +-
 net/ipv4/ip_output.c                  |    2 +-
 net/ipv4/raw.c                        |    2 +-
 net/ipv4/route.c                      |    7 ++++---
 net/ipv4/udp.c                        |    2 +-
 9 files changed, 14 insertions(+), 13 deletions(-)

diff --git a/drivers/infiniband/hw/cxgb3/iwch_cm.c b/drivers/infiniband/hw/cxgb3/iwch_cm.c
index 20ba372..ff3dee4 100644
--- a/drivers/infiniband/hw/cxgb3/iwch_cm.c
+++ b/drivers/infiniband/hw/cxgb3/iwch_cm.c
@@ -332,7 +332,7 @@ static struct rtable *find_route(struct t3cdev *dev, __be32 local_ip,
      }
  };

- if (ip_route_output_flow(&rt, &fl, NULL, 0))
+ if (ip_route_output_flow(&init_net, &rt, &fl, NULL, 0))
   return NULL;
  return rt;
 }
diff --git a/include/net/route.h b/include/net/route.h
index 3e3b14e..6b970d7 100644
--- a/include/net/route.h
+++ b/include/net/route.h
```

```
@@ -112,7 +112,7 @@ extern void  ip_rt_redirect(__be32 old_gw, __be32 dst, __be32 new_gw,
 extern void  rt_cache_flush(int how);
 extern int  __ip_route_output_key(struct net *, struct rtable **, const struct flowi *flp);
 extern int  ip_route_output_key(struct rtable **, struct flowi *flp);
-extern int  ip_route_output_flow(struct rtable **rp, struct flowi *flp, struct sock *sk, int flags);
+extern int  ip_route_output_flow(struct net *, struct rtable **rp, struct flowi *flp, struct sock *sk, int
flags);
 extern int  ip_route_input(struct sk_buff*, __be32 dst, __be32 src, u8 tos, struct net_device
*devin);
 extern unsigned short ip_rt_frag_needed(struct iphdr *iph, unsigned short new_mtu);
 extern void  ip_rt_send_redirect(struct sk_buff *skb);
@@ -167,7 +167,7 @@ static inline int ip_route_connect(struct rtable **rp, __be32 dst,
   *rp = NULL;
  }
  security_sk_classify_flow(sk, &fl);
- return ip_route_output_flow(rp, &fl, sk, flags);
+ return ip_route_output_flow(&init_net, rp, &fl, sk, flags);
 }

 static inline int ip_route_newports(struct rtable **rp, u8 protocol,
@@ -184,7 +184,7 @@ static inline int ip_route_newports(struct rtable **rp, u8 protocol,
   ip_rt_put(*rp);
   *rp = NULL;
   security_sk_classify_flow(sk, &fl);
-  return ip_route_output_flow(rp, &fl, sk, 0);
+  return ip_route_output_flow(&init_net, rp, &fl, sk, 0);
  }
  return 0;
 }
diff --git a/net/dccp/ipv4.c b/net/dccp/ipv4.c
index f450df2..9e38b0d 100644
--- a/net/dccp/ipv4.c
+++ b/net/dccp/ipv4.c
@@ -469,7 +469,7 @@ static struct dst_entry* dccp_v4_route_skb(struct sock *sk,
      };

  security_skb_classify_flow(skb, &fl);
- if (ip_route_output_flow(&rt, &fl, sk, 0)) {
+ if (ip_route_output_flow(&init_net, &rt, &fl, sk, 0)) {
   IP_INC_STATS_BH(IPSTATS_MIB_OUTNOROUTES);
   return NULL;
  }
diff --git a/net/ipv4/af_inet.c b/net/ipv4/af_inet.c
index bcf8c8a..09ca529 100644
--- a/net/ipv4/af_inet.c
+++ b/net/ipv4/af_inet.c
@@ -1113,7 +1113,7 @@ int inet_sk_rebuild_header(struct sock *sk)
  };
```

```
  security_sk_classify_flow(sk, &fl);
- err = ip_route_output_flow(&rt, &fl, sk, 0);
+ err = ip_route_output_flow(&init_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 1c2a32f..7801cce 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(&rt, &fl, sk, 0)) {
+ if (ip_route_output_flow(&init_net, &rt, &fl, sk, 0)) {
   IP_INC_STATS_BH(IPSTATS_MIB_OUTNOROUTES);
   return NULL;
  }
diff --git a/net/ipv4/ip_output.c b/net/ipv4/ip_output.c
index e57de0f..dc56e40 100644
--- a/net/ipv4/ip_output.c
+++ b/net/ipv4/ip_output.c
@@ -350,7 +350,7 @@ int ip_queue_xmit(struct sk_buff *skb, int ipfragok)
     * itself out.
     */
    security_sk_classify_flow(sk, &fl);
-   if (ip_route_output_flow(&rt, &fl, sk, 0))
+   if (ip_route_output_flow(&init_net, &rt, &fl, sk, 0))
     goto no_route;
   }
   sk_setup_caps(sk, &rt->u.dst);
diff --git a/net/ipv4/raw.c b/net/ipv4/raw.c
index 91a5218..85c0869 100644
--- a/net/ipv4/raw.c
+++ b/net/ipv4/raw.c
@@ -558,7 +558,7 @@ static int raw_sendmsg(struct kiocb *iocb, struct sock *sk, struct msghdr
*msg,
   }

   security_sk_classify_flow(sk, &fl);
-  err = ip_route_output_flow(&rt, &fl, sk, 1);
+  err = ip_route_output_flow(&init_net, &rt, &fl, sk, 1);
  }
  if (err)
   goto done;
diff --git a/net/ipv4/route.c b/net/ipv4/route.c
```

```
index cb035cc..58ad12a 100644
--- a/net/ipv4/route.c
+++ b/net/ipv4/route.c
@@ -2532,11 +2532,12 @@ static int ipv4_dst_blackhole(struct rtable **rp, struct flowi *flp, struct
sock
  return (rt ? 0 : -ENOMEM);
 }

-int ip_route_output_flow(struct rtable **rp, struct flowi *flp, struct sock *sk, int flags)
+int ip_route_output_flow(struct net *net, struct rtable **rp, struct flowi *flp,
+     struct sock *sk, int flags)
 {
  int err;

- if ((err = __ip_route_output_key(&init_net, rp, flp)) != 0)
+ if ((err = __ip_route_output_key(net, rp, flp)) != 0)
    return err;

  if (flp->proto) {
@@ -2559,7 +2560,7 @@ EXPORT_SYMBOL_GPL(ip_route_output_flow);

 int ip_route_output_key(struct rtable **rp, struct flowi *flp)
 {
- return ip_route_output_flow(rp, flp, NULL, 0);
+ return ip_route_output_flow(&init_net, rp, flp, NULL, 0);
 }

 static int rt_fill_info(struct sk_buff *skb, u32 pid, u32 seq, int event,
diff --git a/net/ipv4/udp.c b/net/ipv4/udp.c
index cb2411c..9092d80 100644
--- a/net/ipv4/udp.c
+++ b/net/ipv4/udp.c
@@ -660,7 +660,7 @@ 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(&rt, &fl, sk, 1);
+   err = ip_route_output_flow(&init_net, &rt, &fl, sk, 1);
    if (err) {
     if (err == -ENETUNREACH)
      IP_INC_STATS_BH(IPSTATS_MIB_OUTNOROUTES);
--
1.5.3.rc5
```

Subject: [PATCH 9/12 net-2.6.25] [NETNS]: Add namespace parameter to ip_route_output_key.

Needed to propagate it down to the ip_route_output_flow.

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

```
 drivers/infiniband/core/addr.c   |    4 ++--
 drivers/net/bonding/bond_main.c  |    2 +-
 include/net/route.h              |    2 +-
 net/atm/clip.c                   |    2 +-
 net/bridge/br_netfilter.c        |    2 +-
 net/ipv4/arp.c                   |    6 +++---
 net/ipv4/icmp.c                  |    4 ++--
 net/ipv4/igmp.c                  |    6 +++---
 net/ipv4/ip_gre.c                |   10 +++++++-----
 net/ipv4/ip_output.c             |    2 +-
 net/ipv4/ipip.c                  |    8 ++++----
 net/ipv4/ipmr.c                  |    4 ++--
 net/ipv4/ipvs/ip_vs_xmit.c       |    6 +++---
 net/ipv4/netfilter.c             |    6 +++---
 net/ipv4/netfilter/nf_nat_rule.c |    2 +-
 net/ipv4/route.c                 |    6 +++---
 net/ipv4/syncookies.c            |    2 +-
 net/ipv6/ip6_tunnel.c            |    4 ++--
 net/ipv6/sit.c                   |    4 ++--
 net/rxrpc/ar-peer.c              |    2 +-
 net/sctp/protocol.c              |    4 ++--
 21 files changed, 44 insertions(+), 44 deletions(-)

diff --git a/drivers/infiniband/core/addr.c b/drivers/infiniband/core/addr.c
index 963177e..a58ad8a 100644
--- a/drivers/infiniband/core/addr.c
+++ b/drivers/infiniband/core/addr.c
@@ -158,7 +158,7 @@ static void addr_send_arp(struct sockaddr_in *dst_in)

  memset(&fl, 0, sizeof fl);
  fl.nl_u.ip4_u.daddr = dst_ip;
- if (ip_route_output_key(&rt, &fl))
+ if (ip_route_output_key(&init_net, &rt, &fl))
   return;

  neigh_event_send(rt->u.dst.neighbour, NULL);
@@ -179,7 +179,7 @@ static int addr_resolve_remote(struct sockaddr_in *src_in,
  memset(&fl, 0, sizeof fl);
  fl.nl_u.ip4_u.daddr = dst_ip;
  fl.nl_u.ip4_u.saddr = src_ip;
- ret = ip_route_output_key(&rt, &fl);
+ ret = ip_route_output_key(&init_net, &rt, &fl);
```

```
  if (ret)
   goto out;

diff --git a/drivers/net/bonding/bond_main.c b/drivers/net/bonding/bond_main.c
index b0b2603..7a7be20 100644
--- a/drivers/net/bonding/bond_main.c
+++ b/drivers/net/bonding/bond_main.c
@@ -2513,7 +2513,7 @@ static void bond_arp_send_all(struct bonding *bond, struct slave
*slave)
   fl.fl4_dst = targets[i];
   fl.fl4_tos = RTO_ONLINK;

-  rv = ip_route_output_key(&rt, &fl);
+  rv = ip_route_output_key(&init_net, &rt, &fl);
   if (rv) {
    if (net_ratelimit()) {
     printk(KERN_WARNING DRV_NAME
diff --git a/include/net/route.h b/include/net/route.h
index 6b970d7..d9b876a 100644
--- a/include/net/route.h
+++ b/include/net/route.h
@@ -111,7 +111,7 @@ extern void  ip_rt_redirect(__be32 old_gw, __be32 dst, __be32 new_gw,
           __be32 src, struct net_device *dev);
 extern void  rt_cache_flush(int how);
 extern int  __ip_route_output_key(struct net *, struct rtable **, const struct flowi *flp);
-extern int  ip_route_output_key(struct rtable **, struct flowi *flp);
+extern int  ip_route_output_key(struct net *, struct rtable **, struct flowi *flp);
 extern int  ip_route_output_flow(struct net *, struct rtable **rp, struct flowi *flp, struct sock *sk, int
flags);
 extern int  ip_route_input(struct sk_buff*, __be32 dst, __be32 src, u8 tos, struct net_device
*devin);
 extern unsigned short ip_rt_frag_needed(struct iphdr *iph, unsigned short new_mtu);
diff --git a/net/atm/clip.c b/net/atm/clip.c
index 45e0862..86b885e 100644
--- a/net/atm/clip.c
+++ b/net/atm/clip.c
@@ -534,7 +534,7 @@ static int clip_setentry(struct atm_vcc *vcc, __be32 ip)
   unlink_clip_vcc(clip_vcc);
   return 0;
  }
- error = ip_route_output_key(&rt, &fl);
+ error = ip_route_output_key(&init_net, &rt, &fl);
  if (error)
   return error;
  neigh = __neigh_lookup(&clip_tbl, &ip, rt->u.dst.dev, 1);
diff --git a/net/bridge/br_netfilter.c b/net/bridge/br_netfilter.c
index 0e884fe..d4579cf 100644
--- a/net/bridge/br_netfilter.c
```

```
+++ b/net/bridge/br_netfilter.c
@@ -336,7 +336,7 @@ static int br_nf_pre_routing_finish(struct sk_buff *skb)
    if (err != -EHOSTUNREACH || !in_dev || IN_DEV_FORWARD(in_dev))
     goto free_skb;

-   if (!ip_route_output_key(&rt, &fl)) {
+   if (!ip_route_output_key(&init_net, &rt, &fl)) {
     /* - Bridged-and-DNAT'ed traffic doesn't
      *   require ip_forwarding. */
     if (((struct dst_entry *)rt)->dev == dev) {
diff --git a/net/ipv4/arp.c b/net/ipv4/arp.c
index a44ff1a..a3cfd04 100644
--- a/net/ipv4/arp.c
+++ b/net/ipv4/arp.c
@@ -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(&rt, &fl) < 0)
+ if (ip_route_output_key(&init_net, &rt, &fl) < 0)
   return 1;
  if (rt->u.dst.dev != dev) {
   NET_INC_STATS_BH(LINUX_MIB_ARPFILTER);
@@ -1002,7 +1002,7 @@ static int arp_req_set(struct net *net, struct arpreq *r,
   struct flowi fl = { .nl_u = { .ip4_u = { .daddr = ip,
         .tos = RTO_ONLINK } } };
   struct rtable * rt;
-  if ((err = ip_route_output_key(&rt, &fl)) != 0)
+  if ((err = ip_route_output_key(net, &rt, &fl)) != 0)
    return err;
   dev = rt->u.dst.dev;
   ip_rt_put(rt);
@@ -1109,7 +1109,7 @@ static int arp_req_delete(struct net *net, struct arpreq *r,
   struct flowi fl = { .nl_u = { .ip4_u = { .daddr = ip,
         .tos = RTO_ONLINK } } };
   struct rtable * rt;
-  if ((err = ip_route_output_key(&rt, &fl)) != 0)
+  if ((err = ip_route_output_key(net, &rt, &fl)) != 0)
    return err;
   dev = rt->u.dst.dev;
   ip_rt_put(rt);
diff --git a/net/ipv4/icmp.c b/net/ipv4/icmp.c
index 21422bf..c04aac5 100644
--- a/net/ipv4/icmp.c
+++ b/net/ipv4/icmp.c
@@ -404,7 +404,7 @@ static void icmp_reply(struct icmp_bxm *icmp_param, struct sk_buff
*skb)
       .tos = RT_TOS(ip_hdr(skb)->tos) } },
```

```
        .proto = IPPROTO_ICMP };
   security_skb_classify_flow(skb, &fl);
-  if (ip_route_output_key(&rt, &fl))
+  if (ip_route_output_key(&init_net, &rt, &fl))
    goto out_unlock;
  }
  if (icmpv4_xrlim_allow(rt, icmp_param->data.icmph.type,
@@ -598,7 +598,7 @@ void icmp_send(struct sk_buff *skb_in, int type, int code, __be32 info)
    struct dst_entry *odst;

    fl2.fl4_dst = fl.fl4_src;
-   if (ip_route_output_key(&rt2, &fl2))
+   if (ip_route_output_key(&init_net, &rt2, &fl2))
     goto out_unlock;

    /* Ugh! */
diff --git a/net/ipv4/igmp.c b/net/ipv4/igmp.c
index 1f5314c..994648b 100644
--- a/net/ipv4/igmp.c
+++ b/net/ipv4/igmp.c
@@ -301,7 +301,7 @@ static struct sk_buff *igmpv3_newpack(struct net_device *dev, int size)
        .nl_u = { .ip4_u = {
        .daddr = IGMPV3_ALL_MCR } },
        .proto = IPPROTO_IGMP };
-  if (ip_route_output_key(&rt, &fl)) {
+  if (ip_route_output_key(&init_net, &rt, &fl)) {
    kfree_skb(skb);
    return NULL;
   }
@@ -645,7 +645,7 @@ static int igmp_send_report(struct in_device *in_dev, struct ip_mc_list
*pmc,
   struct flowi fl = { .oif = dev->ifindex,
        .nl_u = { .ip4_u = { .daddr = dst } },
        .proto = IPPROTO_IGMP };
-  if (ip_route_output_key(&rt, &fl))
+  if (ip_route_output_key(&init_net, &rt, &fl))
    return -1;
  }
  if (rt->rt_src == 0) {
@@ -1401,7 +1401,7 @@ static struct in_device * ip_mc_find_dev(struct ip_mreqn *imr)
   dev_put(dev);
  }

- if (!dev && !ip_route_output_key(&rt, &fl)) {
+ if (!dev && !ip_route_output_key(&init_net, &rt, &fl)) {
   dev = rt->u.dst.dev;
   ip_rt_put(rt);
  }
```

```
diff --git a/net/ipv4/ip_gre.c b/net/ipv4/ip_gre.c
index a74983d..63f6917 100644
--- a/net/ipv4/ip_gre.c
+++ b/net/ipv4/ip_gre.c
@@ -480,7 +480,7 @@ out:
  fl.fl4_dst = eiph->saddr;
  fl.fl4_tos = RT_TOS(eiph->tos);
  fl.proto = IPPROTO_GRE;
- if (ip_route_output_key(&rt, &fl)) {
+ if (ip_route_output_key(&init_net, &rt, &fl)) {
   kfree_skb(skb2);
   return;
  }
@@ -493,7 +493,7 @@ out:
   fl.fl4_dst = eiph->daddr;
   fl.fl4_src = eiph->saddr;
   fl.fl4_tos = eiph->tos;
-  if (ip_route_output_key(&rt, &fl) ||
+  if (ip_route_output_key(&init_net, &rt, &fl) ||
       rt->u.dst.dev->type != ARPHRD_IPGRE) {
    ip_rt_put(rt);
    kfree_skb(skb2);
@@ -748,7 +748,7 @@ static int ipgre_tunnel_xmit(struct sk_buff *skb, struct net_device *dev)
       .saddr = tiph->saddr,
       .tos = RT_TOS(tos) } },
         .proto = IPPROTO_GRE };
-  if (ip_route_output_key(&rt, &fl)) {
+  if (ip_route_output_key(&init_net, &rt, &fl)) {
    tunnel->stat.tx_carrier_errors++;
    goto tx_error;
   }
@@ -921,7 +921,7 @@ static void ipgre_tunnel_bind_dev(struct net_device *dev)
       .tos = RT_TOS(iph->tos) } },
         .proto = IPPROTO_GRE };
   struct rtable *rt;
-  if (!ip_route_output_key(&rt, &fl)) {
+  if (!ip_route_output_key(&init_net, &rt, &fl)) {
    tdev = rt->u.dst.dev;
    ip_rt_put(rt);
   }
@@ -1177,7 +1177,7 @@ static int ipgre_open(struct net_device *dev)
       .tos = RT_TOS(t->parms.iph.tos) } },
         .proto = IPPROTO_GRE };
   struct rtable *rt;
-  if (ip_route_output_key(&rt, &fl))
+  if (ip_route_output_key(&init_net, &rt, &fl))
    return -EADDRNOTAVAIL;
   dev = rt->u.dst.dev;
```

```
   ip_rt_put(rt);
diff --git a/net/ipv4/ip_output.c b/net/ipv4/ip_output.c
index dc56e40..6a5b839 100644
--- a/net/ipv4/ip_output.c
+++ b/net/ipv4/ip_output.c
@@ -1377,7 +1377,7 @@ void ip_send_reply(struct sock *sk, struct sk_buff *skb, struct
ip_reply_arg *ar
         .dport = tcp_hdr(skb)->source } },
          .proto = sk->sk_protocol };
   security_skb_classify_flow(skb, &fl);
-  if (ip_route_output_key(&rt, &fl))
+  if (ip_route_output_key(&init_net, &rt, &fl))
    return;
  }

diff --git a/net/ipv4/ipip.c b/net/ipv4/ipip.c
index 160535b..da28158 100644
--- a/net/ipv4/ipip.c
+++ b/net/ipv4/ipip.c
@@ -405,7 +405,7 @@ out:
  fl.fl4_daddr = eiph->saddr;
  fl.fl4_tos = RT_TOS(eiph->tos);
  fl.proto = IPPROTO_IPIP;
- if (ip_route_output_key(&rt, &key)) {
+ if (ip_route_output_key(&init_net, &rt, &key)) {
   kfree_skb(skb2);
   return 0;
  }
@@ -418,7 +418,7 @@ out:
   fl.fl4_daddr = eiph->daddr;
   fl.fl4_src = eiph->saddr;
   fl.fl4_tos = eiph->tos;
-  if (ip_route_output_key(&rt, &fl) ||
+  if (ip_route_output_key(&init_net, &rt, &fl) ||
       rt->u.dst.dev->type != ARPHRD_TUNNEL) {
    ip_rt_put(rt);
    kfree_skb(skb2);
@@ -547,7 +547,7 @@ static int ipip_tunnel_xmit(struct sk_buff *skb, struct net_device *dev)
       .saddr = tiph->saddr,
       .tos = RT_TOS(tos) } },
         .proto = IPPROTO_IPIP };
-  if (ip_route_output_key(&rt, &fl)) {
+  if (ip_route_output_key(&init_net, &rt, &fl)) {
    tunnel->stat.tx_carrier_errors++;
    goto tx_error_icmp;
   }
@@ -668,7 +668,7 @@ static void ipip_tunnel_bind_dev(struct net_device *dev)
       .tos = RT_TOS(iph->tos) } },
```

```
        .proto = IPPROTO_IPIP };
   struct rtable *rt;
-  if (!ip_route_output_key(&rt, &fl)) {
+  if (!ip_route_output_key(&init_net, &rt, &fl)) {
    tdev = rt->u.dst.dev;
    ip_rt_put(rt);
   }
diff --git a/net/ipv4/ipmr.c b/net/ipv4/ipmr.c
index 2212717..a94f52c 100644
--- a/net/ipv4/ipmr.c
+++ b/net/ipv4/ipmr.c
@@ -1185,7 +1185,7 @@ static void ipmr_queue_xmit(struct sk_buff *skb, struct mfc_cache *c,
int vifi)
        .saddr = vif->local,
        .tos = RT_TOS(iph->tos) } },
         .proto = IPPROTO_IPIP };
-  if (ip_route_output_key(&rt, &fl))
+  if (ip_route_output_key(&init_net, &rt, &fl))
    goto out_free;
   encap = sizeof(struct iphdr);
  } else {
@@ -1194,7 +1194,7 @@ static void ipmr_queue_xmit(struct sk_buff *skb, struct mfc_cache *c,
int vifi)
            { .daddr = iph->daddr,
        .tos = RT_TOS(iph->tos) } },
         .proto = IPPROTO_IPIP };
-  if (ip_route_output_key(&rt, &fl))
+  if (ip_route_output_key(&init_net, &rt, &fl))
    goto out_free;
  }

diff --git a/net/ipv4/ipvs/ip_vs_xmit.c b/net/ipv4/ipvs/ip_vs_xmit.c
index 8436bf8..f63006c 100644
--- a/net/ipv4/ipvs/ip_vs_xmit.c
+++ b/net/ipv4/ipvs/ip_vs_xmit.c
@@ -78,7 +78,7 @@ __ip_vs_get_out_rt(struct ip_vs_conn *cp, u32 rtos)
        .tos = rtos, } },
    };

-   if (ip_route_output_key(&rt, &fl)) {
+   if (ip_route_output_key(&init_net, &rt, &fl)) {
     spin_unlock(&dest->dst_lock);
     IP_VS_DBG_RL("ip_route_output error, "
          "dest: %u.%u.%u.%u\n",
@@ -101,7 +101,7 @@ __ip_vs_get_out_rt(struct ip_vs_conn *cp, u32 rtos)
      .tos = rtos, } },
   };
```

```
-  if (ip_route_output_key(&rt, &fl)) {
+  if (ip_route_output_key(&init_net, &rt, &fl)) {
     IP_VS_DBG_RL("ip_route_output error, dest: "
          "%u.%u.%u.%u\n", NIPQUAD(cp->daddr));
     return NULL;
@@ -170,7 +170,7 @@ ip_vs_bypass_xmit(struct sk_buff *skb, struct ip_vs_conn *cp,

   EnterFunction(10);

- if (ip_route_output_key(&rt, &fl)) {
+ if (ip_route_output_key(&init_net, &rt, &fl)) {
    IP_VS_DBG_RL("ip_vs_bypass_xmit(): ip_route_output error, "
         "dest: %u.%u.%u.%u\n", NIPQUAD(iph->daddr));
    goto tx_error_icmp;
diff --git a/net/ipv4/netfilter.c b/net/ipv4/netfilter.c
index 6322155..9a904c6 100644
--- a/net/ipv4/netfilter.c
+++ b/net/ipv4/netfilter.c
@@ -33,7 +33,7 @@ int ip_route_me_harder(struct sk_buff *skb, unsigned addr_type)
    fl.nl_u.ip4_u.tos = RT_TOS(iph->tos);
    fl.oif = skb->sk ? skb->sk->sk_bound_dev_if : 0;
    fl.mark = skb->mark;
-   if (ip_route_output_key(&rt, &fl) != 0)
+   if (ip_route_output_key(&init_net, &rt, &fl) != 0)
     return -1;

    /* Drop old route. */
@@ -43,7 +43,7 @@ int ip_route_me_harder(struct sk_buff *skb, unsigned addr_type)
    /* non-local src, find valid iif to satisfy
     * rp-filter when calling ip_route_input. */
    fl.nl_u.ip4_u.daddr = iph->saddr;
-   if (ip_route_output_key(&rt, &fl) != 0)
+   if (ip_route_output_key(&init_net, &rt, &fl) != 0)
     return -1;

    odst = skb->dst;
@@ -187,7 +187,7 @@ EXPORT_SYMBOL(nf_ip_checksum);

 static int nf_ip_route(struct dst_entry **dst, struct flowi *fl)
 {
- return ip_route_output_key((struct rtable **)dst, fl);
+ return ip_route_output_key(&init_net, (struct rtable **)dst, fl);
 }

 static const struct nf_afinfo nf_ip_afinfo = {
diff --git a/net/ipv4/netfilter/nf_nat_rule.c b/net/ipv4/netfilter/nf_nat_rule.c
index 4391aec..5191822 100644
--- a/net/ipv4/netfilter/nf_nat_rule.c
```

```
+++ b/net/ipv4/netfilter/nf_nat_rule.c
@@ -97,7 +97,7 @@ static void warn_if_extra_mangle(__be32 dstip, __be32 srcip)
  struct flowi fl = { .nl_u = { .ip4_u = { .daddr = dstip } } };
  struct rtable *rt;

- if (ip_route_output_key(&rt, &fl) != 0)
+ if (ip_route_output_key(&init_net, &rt, &fl) != 0)
   return;

  if (rt->rt_src != srcip && !warned) {
diff --git a/net/ipv4/route.c b/net/ipv4/route.c
index 58ad12a..87076c6 100644
--- a/net/ipv4/route.c
+++ b/net/ipv4/route.c
@@ -2558,9 +2558,9 @@ int ip_route_output_flow(struct net *net, struct rtable **rp, struct flowi
*flp,

 EXPORT_SYMBOL_GPL(ip_route_output_flow);

-int ip_route_output_key(struct rtable **rp, struct flowi *flp)
+int ip_route_output_key(struct net *net, struct rtable **rp, struct flowi *flp)
 {
- return ip_route_output_flow(&init_net, rp, flp, NULL, 0);
+ return ip_route_output_flow(net, rp, flp, NULL, 0);
 }

 static int rt_fill_info(struct sk_buff *skb, u32 pid, u32 seq, int event,
@@ -2727,7 +2727,7 @@ static int inet_rtm_getroute(struct sk_buff *in_skb, struct nlmsghdr*
nlh, void
    },
    .oif = tb[RTA_OIF] ? nla_get_u32(tb[RTA_OIF]) : 0,
   };
-  err = ip_route_output_key(&rt, &fl);
+  err = ip_route_output_key(&init_net, &rt, &fl);
  }

  if (err)
diff --git a/net/ipv4/syncookies.c b/net/ipv4/syncookies.c
index 2da1be0..f470fe4 100644
--- a/net/ipv4/syncookies.c
+++ b/net/ipv4/syncookies.c
@@ -264,7 +264,7 @@ struct sock *cookie_v4_check(struct sock *sk, struct sk_buff *skb,
            { .sport = th->dest,
        .dport = th->source } } };
   security_req_classify_flow(req, &fl);
-  if (ip_route_output_key(&rt, &fl)) {
+  if (ip_route_output_key(&init_net, &rt, &fl)) {
    reqsk_free(req);
```

```
    goto out;
   }
diff --git a/net/ipv6/ip6_tunnel.c b/net/ipv6/ip6_tunnel.c
index 425c9ae..9031e52 100644
--- a/net/ipv6/ip6_tunnel.c
+++ b/net/ipv6/ip6_tunnel.c
@@ -533,7 +533,7 @@ ip4ip6_err(struct sk_buff *skb, struct inet6_skb_parm *opt,
  fl.fl4_dst = eiph->saddr;
  fl.fl4_tos = RT_TOS(eiph->tos);
  fl.proto = IPPROTO_IPIP;
- if (ip_route_output_key(&rt, &fl))
+ if (ip_route_output_key(&init_net, &rt, &fl))
    goto out;

  skb2->dev = rt->u.dst.dev;
@@ -545,7 +545,7 @@ ip4ip6_err(struct sk_buff *skb, struct inet6_skb_parm *opt,
    fl.fl4_dst = eiph->daddr;
    fl.fl4_src = eiph->saddr;
    fl.fl4_tos = eiph->tos;
-   if (ip_route_output_key(&rt, &fl) ||
+   if (ip_route_output_key(&init_net, &rt, &fl) ||
        rt->u.dst.dev->type != ARPHRD_TUNNEL) {
     ip_rt_put(rt);
     goto out;
diff --git a/net/ipv6/sit.c b/net/ipv6/sit.c
index 1c6fddb..e77239d 100644
--- a/net/ipv6/sit.c
+++ b/net/ipv6/sit.c
@@ -557,7 +557,7 @@ static int ipip6_tunnel_xmit(struct sk_buff *skb, struct net_device *dev)
        .tos = RT_TOS(tos) } },
          .oif = tunnel->parms.link,
          .proto = IPPROTO_IPV6 };
-   if (ip_route_output_key(&rt, &fl)) {
+   if (ip_route_output_key(&init_net, &rt, &fl)) {
     tunnel->stat.tx_carrier_errors++;
     goto tx_error_icmp;
    }
@@ -686,7 +686,7 @@ static void ipip6_tunnel_bind_dev(struct net_device *dev)
         .oif = tunnel->parms.link,
         .proto = IPPROTO_IPV6 };
    struct rtable *rt;
-   if (!ip_route_output_key(&rt, &fl)) {
+   if (!ip_route_output_key(&init_net, &rt, &fl)) {
     tdev = rt->u.dst.dev;
     ip_rt_put(rt);
    }
diff --git a/net/rxrpc/ar-peer.c b/net/rxrpc/ar-peer.c
index 90fa107..2abe208 100644
```

```
--- a/net/rxrpc/ar-peer.c
+++ b/net/rxrpc/ar-peer.c
@@ -57,7 +57,7 @@ static void rxrpc_assess_MTU_size(struct rxrpc_peer *peer)
   BUG();
  }

- ret = ip_route_output_key(&rt, &fl);
+ ret = ip_route_output_key(&init_net, &rt, &fl);
  if (ret < 0) {
   _leave(" [route err %d]", ret);
   return;
diff --git a/net/sctp/protocol.c b/net/sctp/protocol.c
index 3f7def2..1339742 100644
--- a/net/sctp/protocol.c
+++ b/net/sctp/protocol.c
@@ -454,7 +454,7 @@ static struct dst_entry *sctp_v4_get_dst(struct sctp_association *asoc,
      __FUNCTION__, NIPQUAD(fl.fl4_dst),
      NIPQUAD(fl.fl4_src));

- if (!ip_route_output_key(&rt, &fl)) {
+ if (!ip_route_output_key(&init_net, &rt, &fl)) {
   dst = &rt->u.dst;
  }

@@ -497,7 +497,7 @@ static struct dst_entry *sctp_v4_get_dst(struct sctp_association *asoc,
   if ((laddr->state == SCTP_ADDR_SRC) &&
       (AF_INET == laddr->a.sa.sa_family)) {
    fl.fl4_src = laddr->a.v4.sin_addr.s_addr;
-   if (!ip_route_output_key(&rt, &fl)) {
+   if (!ip_route_output_key(&init_net, &rt, &fl)) {
     dst = &rt->u.dst;
     goto out_unlock;
    }
--
1.5.3.rc5
```

Subject: Re: [PATCH 5/12 net-2.6.25] [NETNS]: Re-export init_net via EXPORT_SYMBOL.
Posted by Patrick McHardy on Tue, 22 Jan 2008 17:04:40 GMT
View Forum Message <> Reply to Message

Denis V. Lunev wrote:
> init_net is used added as a parameter to a lot of old API calls, f.e.
> ip_dev_find. These calls were exported as EXPORT_SYMBOL. So, export init_net
> as EXPORT_SYMBOL to keep networking API consistent.

I think this should go in 2.6.24 if still possible so people
don't have to find workarounds that will be obsolete one
version later.

Subject: Re: [PATCH 5/12 net-2.6.25] [NETNS]: Re-export init_net via EXPORT_SYMBOL.
Posted by dlunev on Tue, 22 Jan 2008 17:19:11 GMT
View Forum Message <> Reply to Message

Patrick McHardy wrote:
> Denis V. Lunev wrote:
>> init_net is used added as a parameter to a lot of old API calls, f.e.
>> ip_dev_find. These calls were exported as EXPORT_SYMBOL. So, export
>> init_net
>> as EXPORT_SYMBOL to keep networking API consistent.
>
>
> I think this should go in 2.6.24 if still possible so people
> don't have to find workarounds that will be obsolete one
> version later.
>
yep, sure :) should I send this one separate for 2.4?

Subject: Re: [PATCH 5/12 net-2.6.25] [NETNS]: Re-export init_net via EXPORT_SYMBOL.
Posted by davem on Wed, 23 Jan 2008 06:08:37 GMT
View Forum Message <> Reply to Message

From: "Denis V. Lunev" <dlunev@gmail.com>
Date: Tue, 22 Jan 2008 20:19:11 +0300

> Patrick McHardy wrote:
> > Denis V. Lunev wrote:
> >> init_net is used added as a parameter to a lot of old API calls, f.e.
> >> ip_dev_find. These calls were exported as EXPORT_SYMBOL. So, export
> >> init_net
> >> as EXPORT_SYMBOL to keep networking API consistent.
> >
> >
> > I think this should go in 2.6.24 if still possible so people
> > don't have to find workarounds that will be obsolete one
> > version later.
> >
> yep, sure :) should I send this one separate for 2.4?

I'll take care of this.

Subject: Re: [PATCH 0/12 net-2.6.25] [NETNS]: Routing namespacing on IP output path.
Posted by davem on Wed, 23 Jan 2008 06:09:52 GMT
View Forum Message <> Reply to Message

From: "Denis V. Lunev" <den@openvz.org>
Date: Tue, 22 Jan 2008 18:58:54 +0300

> This set introduces namespacing in the IP output path. The namespace is
> added to all routing API functions except ones with a valid socket. This
> is very intrusive.
>
> Routing cache is virtualized as a part of this efforts, though the hash
> function is not tuned to use namespace id. This not required to work in
> initial namespace.
>
> ICMP replies now also use correct namespace.
>
> Signed-off-by: Denis V. Lunev <den@openvz.org>

Only patches to 1-9 (out of 12) showed up in my mailbox and the netdev
list so that is what I applied :-)

Thanks!

Subject: [PATCH 10/12 net-2.6.25] [NETNS]: Correct namespace for connect-time routing.
Posted by den on Wed, 23 Jan 2008 07:46:25 GMT
View Forum Message <> Reply to Message

ip_route_connect and ip_route_newports are a part of routing API presented to
the socket layer. The namespace is available inside them through a socket.

Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 include/net/route.h | 8 +++++---
 1 files changed, 5 insertions(+), 3 deletions(-)

diff --git a/include/net/route.h b/include/net/route.h
index d9b876a..1985d82 100644
--- a/include/net/route.h
+++ b/include/net/route.h
@@ -33,6 +33,7 @@

```
 #include <linux/ip.h>
 #include <linux/cache.h>
 #include <linux/security.h>
+#include <net/sock.h>

 #ifndef __KERNEL__
 #warning This file is not supposed to be used outside of kernel.
@@ -157,8 +158,9 @@ static inline int ip_route_connect(struct rtable **rp, __be32 dst,
       .dport = dport } } };

  int err;
+ struct net *net = sk->sk_net;
  if (!dst || !src) {
-  err = __ip_route_output_key(&init_net, rp, &fl);
+  err = __ip_route_output_key(net, rp, &fl);
   if (err)
    return err;
   fl.fl4_dst = (*rp)->rt_dst;
@@ -167,7 +169,7 @@ static inline int ip_route_connect(struct rtable **rp, __be32 dst,
   *rp = NULL;
  }
  security_sk_classify_flow(sk, &fl);
- return ip_route_output_flow(&init_net, rp, &fl, sk, flags);
+ return ip_route_output_flow(net, rp, &fl, sk, flags);
 }

 static inline int ip_route_newports(struct rtable **rp, u8 protocol,
@@ -184,7 +186,7 @@ static inline int ip_route_newports(struct rtable **rp, u8 protocol,
   ip_rt_put(*rp);
   *rp = NULL;
   security_sk_classify_flow(sk, &fl);
-  return ip_route_output_flow(&init_net, rp, &fl, sk, 0);
+  return ip_route_output_flow(sk->sk_net, rp, &fl, sk, 0);
  }
  return 0;
 }
--
1.5.3.rc5
```

Subject: [PATCH 11/12 net-2.6.25] [NETNS]: Routing cache virtualization.
Posted by den on Wed, 23 Jan 2008 07:46:26 GMT
View Forum Message <> Reply to Message

Basically, this piece looks relatively easy. Namespace is already available
on the dst entry via device and the device is safe to dereferrence. Compare
it with one of a searcher and skip entry if appropriate.

The only exception is ip_rt_frag_needed. So, add namespace parameter to it.

Signed-off-by: Denis V. Lunev <den@openvz.org>
---
 include/net/route.h | 2 +-
 net/ipv4/icmp.c | 2 +-
 net/ipv4/route.c | 21 ++++++++++++++++-----
 3 files changed, 18 insertions(+), 7 deletions(-)

diff --git a/include/net/route.h b/include/net/route.h
index 1985d82..4eabf00 100644
--- a/include/net/route.h
+++ b/include/net/route.h
@@ -115,7 +115,7 @@ extern int __ip_route_output_key(struct net *, struct rtable **, const struct f
 extern int ip_route_output_key(struct net *, struct rtable **, struct flowi *flp);
 extern int ip_route_output_flow(struct net *, struct rtable **rp, struct flowi *flp, struct sock *sk, int flags);
 extern int ip_route_input(struct sk_buff*, __be32 dst, __be32 src, u8 tos, struct net_device *devin);
-extern unsigned short ip_rt_frag_needed(struct iphdr *iph, unsigned short new_mtu);
+extern unsigned short ip_rt_frag_needed(struct net *net, struct iphdr *iph, unsigned short new_mtu);
 extern void ip_rt_send_redirect(struct sk_buff *skb);

 extern unsigned inet_addr_type(struct net *net, __be32 addr);
diff --git a/net/ipv4/icmp.c b/net/ipv4/icmp.c
index c04aac5..052b278 100644
--- a/net/ipv4/icmp.c
+++ b/net/ipv4/icmp.c
@@ -696,7 +696,7 @@ static void icmp_unreach(struct sk_buff *skb)
         "and DF set.\n",
            NIPQUAD(iph->daddr));
    } else {
-    info = ip_rt_frag_needed(iph,
+   info = ip_rt_frag_needed(&init_net, iph,
            ntohs(icmph->un.frag.mtu));
     if (!info)
      goto out;
diff --git a/net/ipv4/route.c b/net/ipv4/route.c
index 87076c6..07dd295 100644
--- a/net/ipv4/route.c
+++ b/net/ipv4/route.c
@@ -648,6 +648,11 @@ static inline int compare_keys(struct flowi *fl1, struct flowi *fl2)
   (fl1->iif ^ fl2->iif)) == 0;
 }

+static inline int compare_netns(struct rtable *rt1, struct rtable *rt2)

```
+{
+ return rt1->u.dst.dev->nd_net == rt2->u.dst.dev->nd_net;
+}
+
 /*
  * Perform a full scan of hash table and free all entries.
  * Can be called by a softirq or a process.
@@ -961,7 +966,7 @@ restart:

  spin_lock_bh(rt_hash_lock_addr(hash));
  while ((rth = *rthp) != NULL) {
-  if (compare_keys(&rth->fl, &rt->fl)) {
+  if (compare_keys(&rth->fl, &rt->fl) && compare_netns(rth, rt)) {
    /* Put it first */
    *rthp = rth->u.dst.rt_next;
    /*
@@ -1415,7 +1420,8 @@ static __inline__ unsigned short guess_mtu(unsigned short old_mtu)
  return 68;
 }

-unsigned short ip_rt_frag_needed(struct iphdr *iph, unsigned short new_mtu)
+unsigned short ip_rt_frag_needed(struct net *net, struct iphdr *iph,
+     unsigned short new_mtu)
 {
  int i;
  unsigned short old_mtu = ntohs(iph->tot_len);
@@ -1438,7 +1444,8 @@ unsigned short ip_rt_frag_needed(struct iphdr *iph, unsigned short
new_mtu)
        rth->rt_dst  == daddr &&
        rth->rt_src  == iph->saddr &&
        rth->fl.iif == 0 &&
-       !(dst_metric_locked(&rth->u.dst, RTAX_MTU))) {
+       !(dst_metric_locked(&rth->u.dst, RTAX_MTU)) &&
+       rth->u.dst.dev->nd_net == net) {
     unsigned short mtu = new_mtu;

     if (new_mtu < 68 || new_mtu >= old_mtu) {
@@ -2049,7 +2056,9 @@ int ip_route_input(struct sk_buff *skb, __be32 daddr, __be32 saddr,
  struct rtable * rth;
  unsigned hash;
  int iif = dev->ifindex;
+ struct net *net;

+ net = skb->dev->nd_net;
  tos &= IPTOS_RT_MASK;
  hash = rt_hash(daddr, saddr, iif);

@@ -2061,7 +2070,8 @@ int ip_route_input(struct sk_buff *skb, __be32 daddr, __be32 saddr,
```

```
       rth->fl.iif == iif &&
       rth->fl.oif == 0 &&
       rth->fl.mark == skb->mark &&
-      rth->fl.fl4_tos == tos) {
+      rth->fl.fl4_tos == tos &&
+      rth->u.dst.dev->nd_net == net) {
    dst_use(&rth->u.dst, jiffies);
    RT_CACHE_STAT_INC(in_hit);
    rcu_read_unlock();
@@ -2459,7 +2469,8 @@ int __ip_route_output_key(struct net *net, struct rtable **rp,
       rth->fl.oif == flp->oif &&
       rth->fl.mark == flp->mark &&
       !((rth->fl.fl4_tos ^ flp->fl4_tos) &
-       (IPTOS_RT_MASK | RTO_ONLINK))) {
+       (IPTOS_RT_MASK | RTO_ONLINK)) &&
+      rth->u.dst.dev->nd_net == net) {
    dst_use(&rth->u.dst, jiffies);
    RT_CACHE_STAT_INC(out_hit);
    rcu_read_unlock_bh();
--
1.5.3.rc5
```

Subject: [PATCH 12/12 net-2.6.25] [NETNS]: Add namespace for ICMP replying code.
Posted by den on Wed, 23 Jan 2008 07:46:27 GMT
View Forum Message <> Reply to Message

All needed API is done, the namespace is available when required from the
device on the DST entry from the incoming packet. So, just replace init_net
with proper namespace.

Other protocols will follow.

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

```
---
 net/ipv4/icmp.c      |   21 +++++++++++++++--------
 net/ipv4/ip_output.c |    2 +-
 2 files changed, 14 insertions(+), 9 deletions(-)

diff --git a/net/ipv4/icmp.c b/net/ipv4/icmp.c
index 052b278..a6c092c 100644
--- a/net/ipv4/icmp.c
+++ b/net/ipv4/icmp.c
@@ -404,7 +404,7 @@ static void icmp_reply(struct icmp_bxm *icmp_param, struct sk_buff
*skb)
       .tos = RT_TOS(ip_hdr(skb)->tos) } },
         .proto = IPPROTO_ICMP };
```

```
   security_skb_classify_flow(skb, &fl);
-  if (ip_route_output_key(&init_net, &rt, &fl))
+  if (ip_route_output_key(rt->u.dst.dev->nd_net, &rt, &fl))
    goto out_unlock;
  }
  if (icmpv4_xrlim_allow(rt, icmp_param->data.icmph.type,
@@ -436,9 +436,11 @@ void icmp_send(struct sk_buff *skb_in, int type, int code, __be32 info)
  struct ipcm_cookie ipc;
  __be32 saddr;
  u8  tos;
+ struct net *net;

  if (!rt)
   goto out;
+ net = rt->u.dst.dev->nd_net;

  /*
   * Find the original header. It is expected to be valid, of course.
@@ -514,7 +516,7 @@ void icmp_send(struct sk_buff *skb_in, int type, int code, __be32 info)
   struct net_device *dev = NULL;

   if (rt->fl.iif && sysctl_icmp_errors_use_inbound_ifaddr)
-   dev = dev_get_by_index(&init_net, rt->fl.iif);
+   dev = dev_get_by_index(net, rt->fl.iif);

   if (dev) {
    saddr = inet_select_addr(dev, 0, RT_SCOPE_LINK);
@@ -569,7 +571,7 @@ void icmp_send(struct sk_buff *skb_in, int type, int code, __be32 info)
   struct rtable *rt2;

   security_skb_classify_flow(skb_in, &fl);
-  if (__ip_route_output_key(&init_net, &rt, &fl))
+  if (__ip_route_output_key(net, &rt, &fl))
    goto out_unlock;

   /* No need to clone since we're just using its address. */
@@ -591,14 +593,14 @@ void icmp_send(struct sk_buff *skb_in, int type, int code, __be32 info)
   if (xfrm_decode_session_reverse(skb_in, &fl, AF_INET))
    goto out_unlock;

-  if (inet_addr_type(&init_net, fl.fl4_src) == RTN_LOCAL)
-   err = __ip_route_output_key(&init_net, &rt2, &fl);
+  if (inet_addr_type(net, fl.fl4_src) == RTN_LOCAL)
+   err = __ip_route_output_key(net, &rt2, &fl);
   else {
    struct flowi fl2 = {};
    struct dst_entry *odst;
```

```
    fl2.fl4_dst = fl.fl4_src;
-   if (ip_route_output_key(&init_net, &rt2, &fl2))
+   if (ip_route_output_key(net, &rt2, &fl2))
     goto out_unlock;

    /* Ugh! */
@@ -666,6 +668,9 @@ static void icmp_unreach(struct sk_buff *skb)
  int hash, protocol;
  struct net_protocol *ipprot;
  u32 info = 0;
+ struct net *net;
+
+ net = skb->dst->dev->nd_net;

  /*
   * Incomplete header ?
@@ -696,7 +701,7 @@ static void icmp_unreach(struct sk_buff *skb)
         "and DF set.\n",
            NIPQUAD(iph->daddr));
    } else {
-    info = ip_rt_frag_needed(&init_net, iph,
+    info = ip_rt_frag_needed(net, iph,
           ntohs(icmph->un.frag.mtu));
     if (!info)
      goto out;
@@ -734,7 +739,7 @@ static void icmp_unreach(struct sk_buff *skb)
   */

  if (!sysctl_icmp_ignore_bogus_error_responses &&
-     inet_addr_type(&init_net, iph->daddr) == RTN_BROADCAST) {
+     inet_addr_type(net, iph->daddr) == RTN_BROADCAST) {
   if (net_ratelimit())
    printk(KERN_WARNING "%u.%u.%u.%u sent an invalid ICMP "
         "type %u, code %u "
diff --git a/net/ipv4/ip_output.c b/net/ipv4/ip_output.c
index 6a5b839..4fad239 100644
--- a/net/ipv4/ip_output.c
+++ b/net/ipv4/ip_output.c
@@ -1377,7 +1377,7 @@ void ip_send_reply(struct sock *sk, struct sk_buff *skb, struct
ip_reply_arg *ar
        .dport = tcp_hdr(skb)->source } },
        .proto = sk->sk_protocol };
   security_skb_classify_flow(skb, &fl);
-  if (ip_route_output_key(&init_net, &rt, &fl))
+  if (ip_route_output_key(sk->sk_net, &rt, &fl))
    return;
  }
```

--
1.5.3.rc5

Subject: Re: [PATCH 10/12 net-2.6.25] [NETNS]: Correct namespace for connect-time routing.
Posted by davem on Wed, 23 Jan 2008 07:51:16 GMT
View Forum Message <> Reply to Message

From: "Denis V. Lunev" <den@openvz.org>
Date: Wed, 23 Jan 2008 10:46:25 +0300

> ip_route_connect and ip_route_newports are a part of routing API presented to
> the socket layer. The namespace is available inside them through a socket.
>
> Signed-off-by: Denis V. Lunev <den@openvz.org>

Applied.

Subject: Re: [PATCH 11/12 net-2.6.25] [NETNS]: Routing cache virtualization.
Posted by davem on Wed, 23 Jan 2008 07:51:23 GMT
View Forum Message <> Reply to Message

From: "Denis V. Lunev" <den@openvz.org>
Date: Wed, 23 Jan 2008 10:46:26 +0300

> Basically, this piece looks relatively easy. Namespace is already available
> on the dst entry via device and the device is safe to dereferrence. Compare
> it with one of a searcher and skip entry if appropriate.
>
> The only exception is ip_rt_frag_needed. So, add namespace parameter to it.
>
> Signed-off-by: Denis V. Lunev <den@openvz.org>

Applied.

Subject: Re: [PATCH 12/12 net-2.6.25] [NETNS]: Add namespace for ICMP replying code.
Posted by davem on Wed, 23 Jan 2008 07:51:30 GMT
View Forum Message <> Reply to Message

From: "Denis V. Lunev" <den@openvz.org>
Date: Wed, 23 Jan 2008 10:46:27 +0300

> All needed API is done, the namespace is available when required from the

> device on the DST entry from the incoming packet. So, just replace init_net
> with proper namespace.
>
> Other protocols will follow.
>
> Signed-off-by: Denis V. Lunev <den@openvz.org>

Applied, thanks.

Subject: Re: [PATCH 12/12 net-2.6.25] [NETNS]: Add namespace for ICMP replying code.
Posted by Mathieu Lacage on Wed, 23 Jan 2008 09:16:29 GMT
View Forum Message <> Reply to Message

hi,

On Tue, 2008-01-22 at 23:51 -0800, David Miller wrote:
> From: "Denis V. Lunev" <den@openvz.org>
> Date: Wed, 23 Jan 2008 10:46:27 +0300
>
> > All needed API is done, the namespace is available when required from the
> > device on the DST entry from the incoming packet. So, just replace init_net
> > with proper namespace.
> >
> > Other protocols will follow.
> >
> > Signed-off-by: Denis V. Lunev <den@openvz.org>
>
> Applied, thanks.

I have been following the netns patches on this ML for a while but I still have not figured out in which tree the patches fed to David Miller are applied. I have attempted to grep the public trees 'davem/net-2.6' and 'davem/net-2.6.25' but without much success so far. Is there a public git tree I can clone which contains all the netns patches which David Miller state are 'Applied' ?

thank you,
Mathieu

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

---

Subject: Re: [PATCH 12/12 net-2.6.25] [NETNS]: Add namespace for ICMP replying code.
Posted by yoshfuji on Wed, 23 Jan 2008 09:30:36 GMT
View Forum Message <> Reply to Message

In article <1201079789.10778.26.camel@mistigri.inria.fr> (at Wed, 23 Jan 2008 10:16:29 +0100), Mathieu Lacage <mathieu.lacage@sophia.inria.fr> says:

> I have been following the netns patches on this ML for a while but I
> still have not figured out in which tree the patches fed to David Miller
> are applied. I have attempted to grep the public trees 'davem/net-2.6'
> and 'davem/net-2.6.25' but without much success so far. Is there a
> public git tree I can clone which contains all the netns patches which
> David Miller state are 'Applied' ?

I'm cloning from
 git://git.kernel.org/pub/scm/linux/kernel/git/davem/net-2.6.25.git

There may be some time-lag.

--yoshfuji

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