Subject: [PATCH 10/12] [NETNS]: Make icmp_sk per namespace.
Posted by den on Fri, 29 Feb 2008 13:40:56 GMT
View Forum Message <> Reply to Message

All preparations are done. Now just add a hook to perform an initialization
on namespace startup and replace icmp_sk macro with proper inline call.

Signed-off-by: Denis V. Lunev <den@openvz.org>
Acked-by: Daniel Lezcano <dlezcano@fr.ibm.com>

```
---
 include/net/netns/ipv4.h |    2 +
 net/ipv4/icmp.c          |   49 +++++++++++++++++++++++++++++++----------
 2 files changed, 34 insertions(+), 17 deletions(-)

diff --git a/include/net/netns/ipv4.h b/include/net/netns/ipv4.h
index a9b4f60..504fde1 100644
--- a/include/net/netns/ipv4.h
+++ b/include/net/netns/ipv4.h
@@ -26,6 +26,8 @@ struct netns_ipv4 {
  struct hlist_head *fib_table_hash;
  struct sock  *fibnl;

+ struct sock  **icmp_sk;
+
  struct netns_frags frags;
 #ifdef CONFIG_NETFILTER
  struct xt_table  *iptable_filter;
diff --git a/net/ipv4/icmp.c b/net/ipv4/icmp.c
index 97d97ad..b51f4b0 100644
--- a/net/ipv4/icmp.c
+++ b/net/ipv4/icmp.c
@@ -229,8 +229,10 @@ static const struct icmp_control icmp_pointers[NR_ICMP_TYPES+1];
  *
  * On SMP we have one ICMP socket per-cpu.
  */
-static struct sock **__icmp_sk = NULL;
-#define icmp_sk  (__icmp_sk[smp_processor_id()])
+static struct sock *icmp_sk(struct net *net)
+{
+ return net->ipv4.icmp_sk[smp_processor_id()];
+}

 static inline int icmp_xmit_lock(struct sock *sk)
 {
@@ -349,7 +351,7 @@ static void icmp_push_reply(struct icmp_bxm *icmp_param,
  struct sock *sk;
  struct sk_buff *skb;
```

```
- sk = icmp_sk;
+ sk = icmp_sk(rt->u.dst.dev->nd_net);
  if (ip_append_data(sk, icmp_glue_bits, icmp_param,
       icmp_param->data_len+icmp_param->head_len,
       icmp_param->head_len,
@@ -378,10 +380,11 @@ static void icmp_push_reply(struct icmp_bxm *icmp_param,

 static void icmp_reply(struct icmp_bxm *icmp_param, struct sk_buff *skb)
 {
- struct sock *sk = icmp_sk;
- struct inet_sock *inet = inet_sk(sk);
  struct ipcm_cookie ipc;
  struct rtable *rt = (struct rtable *)skb->dst;
+ struct net *net = rt->u.dst.dev->nd_net;
+ struct sock *sk = icmp_sk(net);
+ struct inet_sock *inet = inet_sk(sk);
  __be32 daddr;

  if (ip_options_echo(&icmp_param->replyopts, skb))
@@ -407,7 +410,7 @@ 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->u.dst.dev->nd_net, &rt, &fl))
+  if (ip_route_output_key(net, &rt, &fl))
    goto out_unlock;
  }
  if (icmpv4_xrlim_allow(rt, icmp_param->data.icmph.type,
@@ -440,11 +443,12 @@ void icmp_send(struct sk_buff *skb_in, int type, int code, __be32 info)
  __be32 saddr;
  u8  tos;
  struct net *net;
- struct sock *sk = icmp_sk;
+ struct sock *sk;

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

  /*
   * Find the original header. It is expected to be valid, of course.
@@ -1142,22 +1146,23 @@ static const struct icmp_control icmp_pointers[NR_ICMP_TYPES +
1] = {
  },
 };
```

```
-static void __exit icmp_exit(void)
+static void __net_exit icmp_sk_exit(struct net *net)
 {
  int i;

  for_each_possible_cpu(i)
-  sk_release_kernel(__icmp_sk[i]);
- kfree(__icmp_sk);
- __icmp_sk = NULL;
+  sk_release_kernel(net->ipv4.icmp_sk[i]);
+ kfree(net->ipv4.icmp_sk);
+ net->ipv4.icmp_sk = NULL;
 }

-int __init icmp_init(void)
+int __net_init icmp_sk_init(struct net *net)
 {
  int i, err;

- __icmp_sk = kzalloc(nr_cpu_ids * sizeof(struct sock *), GFP_KERNEL);
- if (__icmp_sk == NULL)
+ net->ipv4.icmp_sk =
+  kzalloc(nr_cpu_ids * sizeof(struct sock *), GFP_KERNEL);
+ if (net->ipv4.icmp_sk == NULL)
   return -ENOMEM;

  for_each_possible_cpu(i) {
@@ -1169,8 +1174,8 @@ int __init icmp_init(void)
   if (err < 0)
    goto fail;

-  __icmp_sk[i] = sk = sock->sk;
-  sk_change_net(sk, &init_net);
+  net->ipv4.icmp_sk[i] = sk = sock->sk;
+  sk_change_net(sk, net);

   sk->sk_allocation = GFP_ATOMIC;

@@ -1193,10 +1198,20 @@ int __init icmp_init(void)
  return 0;

 fail:
- icmp_exit();
+ icmp_sk_exit(net);
  return err;
 }

+static struct pernet_operations __net_initdata icmp_sk_ops = {
```

```
+	.init = icmp_sk_init,
+	.exit = icmp_sk_exit,
+};
+
+int __init icmp_init(void)
+{
+ return register_pernet_device(&icmp_sk_ops);
+}
+
 EXPORT_SYMBOL(icmp_err_convert);
 EXPORT_SYMBOL(icmp_send);
 EXPORT_SYMBOL(icmp_statistics);
--
1.5.3.rc5
```