Subject: [PATCH 5/12 net-2.6.26] [ICMP]: Pass proper ICMP socket into icmp(v6)_xmit_(un)lock.
Posted by den on Fri, 29 Feb 2008 13:40:51 GMT
View Forum Message <> Reply to Message

We have to get socket lock inside icmp(v6)_xmit_lock/unlock. The socket
is get from global variable now. When this code became namespaces, one
should pass a namespace and get socket from it.

Though, above is useless. Socket is available in the caller, just pass
it inside. This saves a bit of code now and saves more later.

add/remove: 0/0 grow/shrink: 1/3 up/down: 1/-169 (-168)
function old new delta
icmp_rcv 718 719 +1
icmpv6_rcv 2343 2303 -40
icmp_send 1566 1518 -48
icmp_reply 549 468 -81

Signed-off-by: Denis V. Lunev <den@openvz.org>
Acked-by: Daniel Lezcano <dlezcano@fr.ibm.com>
---
net/ipv4/icmp.c | 19 ++++++++++---------
net/ipv6/icmp.c | 24 ++++++++++++++----------
2 files changed, 22 insertions(+), 21 deletions(-)

diff --git a/net/ipv4/icmp.c b/net/ipv4/icmp.c
index 3a4da43..9bcf263 100644
--- a/net/ipv4/icmp.c
+++ b/net/ipv4/icmp.c
@@ -232,11 +232,11 @@ static const struct icmp_control icmp_pointers[NR_ICMP_TYPES+1];
static DEFINE_PER_CPU(struct sock *, __icmp_sk) = NULL;
#define icmp_sk __get_cpu_var(__icmp_sk)

-static inline int icmp_xmit_lock(void)
+static inline int icmp_xmit_lock(struct sock *sk)
{
local_bh_disable();

- if (unlikely(!spin_trylock(&icmp_sk->sk_lock.slock))) {
+ if (unlikely(!spin_trylock(&sk->sk_lock.slock))) {
/* This can happen if the output path signals a
* dst_link_failure() for an outgoing ICMP packet.
*/
@@ -246,9 +246,9 @@ static inline int icmp_xmit_lock(void)
return 0;
}

```
-static inline void icmp_xmit_unlock(void)
+static inline void icmp_xmit_unlock(struct sock *sk)
 {
- spin_unlock_bh(&icmp_sk->sk_lock.slock);
+ spin_unlock_bh(&sk->sk_lock.slock);
 }

 /*
@@ -387,7 +387,7 @@ static void icmp_reply(struct icmp_bxm *icmp_param, struct sk_buff
*skb)
  if (ip_options_echo(&icmp_param->replyopts, skb))
   return;

- if (icmp_xmit_lock())
+ if (icmp_xmit_lock(sk))
   return;

  icmp_param->data.icmph.checksum = 0;
@@ -415,7 +415,7 @@ static void icmp_reply(struct icmp_bxm *icmp_param, struct sk_buff
*skb)
   icmp_push_reply(icmp_param, &ipc, rt);
  ip_rt_put(rt);
 out_unlock:
- icmp_xmit_unlock();
+ icmp_xmit_unlock(sk);
 }


@@ -440,6 +440,7 @@ 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;

  if (!rt)
   goto out;
@@ -507,7 +508,7 @@ void icmp_send(struct sk_buff *skb_in, int type, int code, __be32 info)
   }
  }

- if (icmp_xmit_lock())
+ if (icmp_xmit_lock(sk))
   return;

  /*
@@ -546,7 +547,7 @@ void icmp_send(struct sk_buff *skb_in, int type, int code, __be32 info)
  icmp_param.data.icmph.checksum  = 0;
  icmp_param.skb   = skb_in;
```

```
  icmp_param.offset = skb_network_offset(skb_in);
- inet_sk(icmp_sk)->tos = tos;
+ inet_sk(sk)->tos = tos;
  ipc.addr = iph->saddr;
  ipc.opt = &icmp_param.replyopts;

@@ -654,7 +655,7 @@ route_done:
 ende:
  ip_rt_put(rt);
 out_unlock:
- icmp_xmit_unlock();
+ icmp_xmit_unlock(sk);
 out:;
 }

diff --git a/net/ipv6/icmp.c b/net/ipv6/icmp.c
index 875bdc7..18f220a 100644
--- a/net/ipv6/icmp.c
+++ b/net/ipv6/icmp.c
@@ -90,11 +90,11 @@ static struct inet6_protocol icmpv6_protocol = {
  .flags  = INET6_PROTO_NOPOLICY|INET6_PROTO_FINAL,
 };

-static __inline__ int icmpv6_xmit_lock(void)
+static __inline__ int icmpv6_xmit_lock(struct sock *sk)
 {
  local_bh_disable();

- if (unlikely(!spin_trylock(&icmpv6_sk->sk_lock.slock))) {
+ if (unlikely(!spin_trylock(&sk->sk_lock.slock))) {
   /* This can happen if the output path (f.e. SIT or
    * ip6ip6 tunnel) signals dst_link_failure() for an
    * outgoing ICMP6 packet.
@@ -105,9 +105,9 @@ static __inline__ int icmpv6_xmit_lock(void)
  return 0;
 }

-static __inline__ void icmpv6_xmit_unlock(void)
+static __inline__ void icmpv6_xmit_unlock(struct sock *sk)
 {
- spin_unlock_bh(&icmpv6_sk->sk_lock.slock);
+ spin_unlock_bh(&sk->sk_lock.slock);
 }

 /*
@@ -389,12 +389,12 @@ void icmpv6_send(struct sk_buff *skb, int type, int code, __u32 info,
  fl.fl_icmp_code = code;
  security_skb_classify_flow(skb, &fl);
```

```
- if (icmpv6_xmit_lock())
-  return;
-
  sk = icmpv6_sk;
  np = inet6_sk(sk);

+ if (icmpv6_xmit_lock(sk))
+  return;
+
  if (!icmpv6_xrlim_allow(sk, type, &fl))
   goto out;

@@ -498,7 +498,7 @@ out_put:
 out_dst_release:
  dst_release(dst);
 out:
- icmpv6_xmit_unlock();
+ icmpv6_xmit_unlock(sk);
 }

 EXPORT_SYMBOL(icmpv6_send);
@@ -535,12 +535,12 @@ static void icmpv6_echo_reply(struct sk_buff *skb)
  fl.fl_icmp_type = ICMPV6_ECHO_REPLY;
  security_skb_classify_flow(skb, &fl);

- if (icmpv6_xmit_lock())
-  return;
-
  sk = icmpv6_sk;
  np = inet6_sk(sk);

+ if (icmpv6_xmit_lock(sk))
+  return;
+
  if (!fl.oif && ipv6_addr_is_multicast(&fl.fl6_dst))
   fl.oif = np->mcast_oif;

@@ -584,7 +584,7 @@ out_put:
   in6_dev_put(idev);
  dst_release(dst);
 out:
- icmpv6_xmit_unlock();
+ icmpv6_xmit_unlock(sk);
 }

 static void icmpv6_notify(struct sk_buff *skb, int type, int code, __be32 info)
--
```

1.5.3.rc5

_____

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