Subject: [PATCH 4/12 net-2.6.26] [ICMP]: Store sock rather than socket for ICMP flow control.
Posted by den on Fri, 29 Feb 2008 13:40:50 GMT
View Forum Message <> Reply to Message

Basically, there is no difference, what to store: socket or sock. Though,
sock looks better as there will be 1 less dereferrence on the fast path.

Signed-off-by: Denis V. Lunev <den@openvz.org>
Acked-by: Daniel Lezcano <dlezcano@fr.ibm.com>
---
 net/ipv4/icmp.c | 27 ++++++++++++++++++---------
 net/ipv6/icmp.c | 25 ++++++++++++++++++-------
 2 files changed, 26 insertions(+), 26 deletions(-)

```
diff --git a/net/ipv4/icmp.c b/net/ipv4/icmp.c
index 831b6ad..3a4da43 100644
--- a/net/ipv4/icmp.c
+++ b/net/ipv4/icmp.c
@@ -229,14 +229,14 @@ static const struct icmp_control icmp_pointers[NR_ICMP_TYPES+1];
  *
  * On SMP we have one ICMP socket per-cpu.
  */
-static DEFINE_PER_CPU(struct socket *, __icmp_socket) = NULL;
-#define icmp_socket __get_cpu_var(__icmp_socket)
+static DEFINE_PER_CPU(struct sock *, __icmp_sk) = NULL;
+#define icmp_sk  __get_cpu_var(__icmp_sk)

 static inline int icmp_xmit_lock(void)
 {
  local_bh_disable();

- if (unlikely(!spin_trylock(&icmp_socket->sk->sk_lock.slock))) {
+ if (unlikely(!spin_trylock(&icmp_sk->sk_lock.slock))) {
    /* This can happen if the output path signals a
     * dst_link_failure() for an outgoing ICMP packet.
     */
@@ -248,7 +248,7 @@ static inline int icmp_xmit_lock(void)

 static inline void icmp_xmit_unlock(void)
 {
- spin_unlock_bh(&icmp_socket->sk->sk_lock.slock);
+ spin_unlock_bh(&icmp_sk->sk_lock.slock);
 }

 /*
@@ -349,7 +349,7 @@ static void icmp_push_reply(struct icmp_bxm *icmp_param,
  struct sock *sk;
```

```
  struct sk_buff *skb;

- sk = icmp_socket->sk;
+ sk = icmp_sk;
  if (ip_append_data(sk, icmp_glue_bits, icmp_param,
        icmp_param->data_len+icmp_param->head_len,
        icmp_param->head_len,
@@ -378,7 +378,7 @@ 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_socket->sk;
+ struct sock *sk = icmp_sk;
  struct inet_sock *inet = inet_sk(sk);
  struct ipcm_cookie ipc;
  struct rtable *rt = (struct rtable *)skb->dst;
@@ -546,7 +546,7 @@ 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_socket->sk)->tos = tos;
+ inet_sk(icmp_sk)->tos = tos;
  ipc.addr = iph->saddr;
  ipc.opt = &icmp_param.replyopts;

@@ -1146,13 +1146,13 @@ static void __exit icmp_exit(void)
  int i;

  for_each_possible_cpu(i) {
-  struct socket *sock;
+  struct sock *sk;

-  sock = per_cpu(__icmp_socket, i);
-  if (sock == NULL)
+  sk = per_cpu(__icmp_sk, i);
+  if (sk == NULL)
    continue;
-  per_cpu(__icmp_socket, i) = NULL;
-  sock_release(sock);
+  per_cpu(__icmp_sk, i) = NULL;
+  sock_release(sk->sk_socket);
  }
 }

@@ -1169,8 +1169,7 @@ int __init icmp_init(void)
   if (err < 0)
    goto fail;
```

```
-  per_cpu(__icmp_socket, i) = sock;
-  sk = sock->sk;
+  per_cpu(__icmp_sk, i) = sk = sock->sk;
   sk->sk_allocation = GFP_ATOMIC;

   /* Enough space for 2 64K ICMP packets, including
diff --git a/net/ipv6/icmp.c b/net/ipv6/icmp.c
index b9b13a7..875bdc7 100644
--- a/net/ipv6/icmp.c
+++ b/net/ipv6/icmp.c
@@ -80,8 +80,8 @@ EXPORT_SYMBOL(icmpv6msg_statistics);
  *
  * On SMP we have one ICMP socket per-cpu.
  */
-static DEFINE_PER_CPU(struct socket *, __icmpv6_socket) = NULL;
-#define icmpv6_socket __get_cpu_var(__icmpv6_socket)
+static DEFINE_PER_CPU(struct sock *, __icmpv6_sk) = NULL;
+#define icmpv6_sk __get_cpu_var(__icmpv6_sk)

 static int icmpv6_rcv(struct sk_buff *skb);

@@ -94,7 +94,7 @@ static __inline__ int icmpv6_xmit_lock(void)
 {
  local_bh_disable();

- if (unlikely(!spin_trylock(&icmpv6_socket->sk->sk_lock.slock))) {
+ if (unlikely(!spin_trylock(&icmpv6_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.
@@ -107,7 +107,7 @@ static __inline__ int icmpv6_xmit_lock(void)

 static __inline__ void icmpv6_xmit_unlock(void)
 {
- spin_unlock_bh(&icmpv6_socket->sk->sk_lock.slock);
+ spin_unlock_bh(&icmpv6_sk->sk_lock.slock);
 }

 /*
@@ -392,7 +392,7 @@ void icmpv6_send(struct sk_buff *skb, int type, int code, __u32 info,
  if (icmpv6_xmit_lock())
   return;

- sk = icmpv6_socket->sk;
+ sk = icmpv6_sk;
  np = inet6_sk(sk);

  if (!icmpv6_xrlim_allow(sk, type, &fl))
```

```
@@ -538,7 +538,7 @@ static void icmpv6_echo_reply(struct sk_buff *skb)
  if (icmpv6_xmit_lock())
   return;

- sk = icmpv6_socket->sk;
+ sk = icmpv6_sk;
  np = inet6_sk(sk);

  if (!fl.oif && ipv6_addr_is_multicast(&fl.fl6_dst))
@@ -776,7 +776,7 @@ drop_no_count:
 }

 /*
- * Special lock-class for __icmpv6_socket:
+ * Special lock-class for __icmpv6_sk:
  */
 static struct lock_class_key icmpv6_socket_sk_dst_lock_key;

@@ -786,8 +786,9 @@ int __init icmpv6_init(void)
  int err, i, j;

  for_each_possible_cpu(i) {
+  struct socket *sock;
   err = sock_create_kern(PF_INET6, SOCK_RAW, IPPROTO_ICMPV6,
-          &per_cpu(__icmpv6_socket, i));
+          &sock);
   if (err < 0) {
    printk(KERN_ERR
           "Failed to initialize the ICMP6 control socket "
@@ -796,12 +797,12 @@ int __init icmpv6_init(void)
    goto fail;
   }

-  sk = per_cpu(__icmpv6_socket, i)->sk;
+  per_cpu(__icmpv6_sk, i) = sk = sock->sk;
   sk->sk_allocation = GFP_ATOMIC;
   /*
    * Split off their lock-class, because sk->sk_dst_lock
    * gets used from softirqs, which is safe for
-   * __icmpv6_socket (because those never get directly used
+   * __icmpv6_sk (because those never get directly used
    * via userspace syscalls), but unsafe for normal sockets.
    */
   lockdep_set_class(&sk->sk_dst_lock,
@@ -829,7 +830,7 @@ int __init icmpv6_init(void)
  for (j = 0; j < i; j++) {
   if (!cpu_possible(j))
    continue;
```

```
-  sock_release(per_cpu(__icmpv6_socket, j));
+  sock_release(per_cpu(__icmpv6_sk, j)->sk_socket);
  }

  return err;
@@ -840,7 +841,7 @@ void icmpv6_cleanup(void)
  int i;

  for_each_possible_cpu(i) {
-  sock_release(per_cpu(__icmpv6_socket, i));
+  sock_release(per_cpu(__icmpv6_sk, i)->sk_socket);
  }
  inet6_del_protocol(&icmpv6_protocol, IPPROTO_ICMPV6);
 }
--
1.5.3.rc5
```