Subject: [PATCH net-2.6.26 2/6][SOCK][NETNS]: Add a struct net argument to sock_prot_inuse_add and _get.
Posted by Pavel Emelianov on Mon, 31 Mar 2008 12:12:01 GMT
View Forum Message <> Reply to Message

This counter is about to become per-proto-and-per-net, so we'll need
two arguments to determine which cell in this "table" to work with.

All the places, but proc already pass proper net to it - proc will be
tuned a bit later.

Some indentation with spaces in proc files is done to keep the file
coding style consistent.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

---
 include/net/sock.h | 7 ++++---
 include/net/udp.h | 2 +-
 net/core/sock.c | 4 ++--
 net/ipv4/inet_hashtables.c | 8 ++++----
 net/ipv4/inet_timewait_sock.c | 2 +-
 net/ipv4/proc.c | 11 +++++++----
 net/ipv4/raw.c | 4 ++--
 net/ipv4/udp.c | 2 +-
 net/ipv6/inet6_hashtables.c | 4 ++--
 net/ipv6/ipv6_sockglue.c | 10 ++++++----
 net/ipv6/proc.c | 8 ++++----
 11 files changed, 34 insertions(+), 28 deletions(-)

diff --git a/include/net/sock.h b/include/net/sock.h
index 2a3344f..f4fdd10 100644
--- a/include/net/sock.h
+++ b/include/net/sock.h
@@ -635,10 +635,11 @@ static inline void sk_refcnt_debug_release(const struct sock *sk)

 #ifdef CONFIG_PROC_FS
 /* Called with local bh disabled */
-extern void sock_prot_inuse_add(struct proto *prot, int inc);
-extern int sock_prot_inuse_get(struct proto *proto);
+extern void sock_prot_inuse_add(struct net *net, struct proto *prot, int inc);
+extern int sock_prot_inuse_get(struct net *net, struct proto *proto);
 #else
-static void inline sock_prot_inuse_add(struct proto *prot, int inc)
+static void inline sock_prot_inuse_add(struct net *net, struct proto *prot,
+ int inc)
 {
 }

```
 #endif
diff --git a/include/net/udp.h b/include/net/udp.h
index 24a41fa..3e55a99 100644
--- a/include/net/udp.h
+++ b/include/net/udp.h
@@ -115,7 +115,7 @@ static inline void udp_lib_unhash(struct sock *sk)
  write_lock_bh(&udp_hash_lock);
  if (sk_del_node_init(sk)) {
   inet_sk(sk)->num = 0;
-  sock_prot_inuse_add(sk->sk_prot, -1);
+  sock_prot_inuse_add(sock_net(sk), sk->sk_prot, -1);
  }
  write_unlock_bh(&udp_hash_lock);
 }
diff --git a/net/core/sock.c b/net/core/sock.c
index c1ae56e..6f36ab9 100644
--- a/net/core/sock.c
+++ b/net/core/sock.c
@@ -1949,13 +1949,13 @@ struct prot_inuse {
 static DECLARE_BITMAP(proto_inuse_idx, PROTO_INUSE_NR);
 static DEFINE_PER_CPU(struct prot_inuse, prot_inuse);

-void sock_prot_inuse_add(struct proto *prot, int val)
+void sock_prot_inuse_add(struct net *net, struct proto *prot, int val)
 {
  __get_cpu_var(prot_inuse).val[prot->inuse_idx] += val;
 }
 EXPORT_SYMBOL_GPL(sock_prot_inuse_add);

-int sock_prot_inuse_get(struct proto *prot)
+int sock_prot_inuse_get(struct net *net, struct proto *prot)
 {
  int cpu, idx = prot->inuse_idx;
  int res = 0;
diff --git a/net/ipv4/inet_hashtables.c b/net/ipv4/inet_hashtables.c
index 1b6ff51..32ca2f8 100644
--- a/net/ipv4/inet_hashtables.c
+++ b/net/ipv4/inet_hashtables.c
@@ -288,7 +288,7 @@ unique:
  sk->sk_hash = hash;
  BUG_TRAP(sk_unhashed(sk));
  __sk_add_node(sk, &head->chain);
- sock_prot_inuse_add(sk->sk_prot, 1);
+ sock_prot_inuse_add(sock_net(sk), sk->sk_prot, 1);
  write_unlock(lock);

  if (twp) {
@@ -332,7 +332,7 @@ void __inet_hash_nolisten(struct sock *sk)
```

```
  write_lock(lock);
  __sk_add_node(sk, list);
- sock_prot_inuse_add(sk->sk_prot, 1);
+ sock_prot_inuse_add(sock_net(sk), sk->sk_prot, 1);
  write_unlock(lock);
 }
 EXPORT_SYMBOL_GPL(__inet_hash_nolisten);
@@ -354,7 +354,7 @@ static void __inet_hash(struct sock *sk)

  inet_listen_wlock(hashinfo);
  __sk_add_node(sk, list);
- sock_prot_inuse_add(sk->sk_prot, 1);
+ sock_prot_inuse_add(sock_net(sk), sk->sk_prot, 1);
  write_unlock(lock);
  wake_up(&hashinfo->lhash_wait);
 }
@@ -387,7 +387,7 @@ void inet_unhash(struct sock *sk)
  }

  if (__sk_del_node_init(sk))
-  sock_prot_inuse_add(sk->sk_prot, -1);
+  sock_prot_inuse_add(sock_net(sk), sk->sk_prot, -1);
  write_unlock_bh(lock);
 out:
  if (sk->sk_state == TCP_LISTEN)
diff --git a/net/ipv4/inet_timewait_sock.c b/net/ipv4/inet_timewait_sock.c
index f12bc24..a741378 100644
--- a/net/ipv4/inet_timewait_sock.c
+++ b/net/ipv4/inet_timewait_sock.c
@@ -91,7 +91,7 @@ void __inet_twsk_hashdance(struct inet_timewait_sock *tw, struct sock *sk,

  /* Step 2: Remove SK from established hash. */
  if (__sk_del_node_init(sk))
-  sock_prot_inuse_add(sk->sk_prot, -1);
+  sock_prot_inuse_add(sock_net(sk), sk->sk_prot, -1);

  /* Step 3: Hash TW into TIMEWAIT chain. */
  inet_twsk_add_node(tw, &ehead->twchain);
diff --git a/net/ipv4/proc.c b/net/ipv4/proc.c
index d63474c..8156c26 100644
--- a/net/ipv4/proc.c
+++ b/net/ipv4/proc.c
@@ -53,14 +53,17 @@ static int sockstat_seq_show(struct seq_file *seq, void *v)
 {
  socket_seq_show(seq);
  seq_printf(seq, "TCP: inuse %d orphan %d tw %d alloc %d mem %d\n",
-     sock_prot_inuse_get(&tcp_prot),
```

```
+     sock_prot_inuse_get(&init_net, &tcp_prot),
      atomic_read(&tcp_orphan_count),
      tcp_death_row.tw_count, atomic_read(&tcp_sockets_allocated),
      atomic_read(&tcp_memory_allocated));
- seq_printf(seq, "UDP: inuse %d mem %d\n", sock_prot_inuse_get(&udp_prot),
+ seq_printf(seq, "UDP: inuse %d mem %d\n",
+     sock_prot_inuse_get(&init_net, &udp_prot),
      atomic_read(&udp_memory_allocated));
- seq_printf(seq, "UDPLITE: inuse %d\n", sock_prot_inuse_get(&udplite_prot));
- seq_printf(seq, "RAW: inuse %d\n", sock_prot_inuse_get(&raw_prot));
+ seq_printf(seq, "UDPLITE: inuse %d\n",
+     sock_prot_inuse_get(&init_net, &udplite_prot));
+ seq_printf(seq, "RAW: inuse %d\n",
+     sock_prot_inuse_get(&init_net, &raw_prot));
  seq_printf(seq,  "FRAG: inuse %d memory %d\n",
    ip_frag_nqueues(&init_net), ip_frag_mem(&init_net));
  return 0;
diff --git a/net/ipv4/raw.c b/net/ipv4/raw.c
index 3492050..11d7f75 100644
--- a/net/ipv4/raw.c
+++ b/net/ipv4/raw.c
@@ -93,7 +93,7 @@ void raw_hash_sk(struct sock *sk)

  write_lock_bh(&h->lock);
  sk_add_node(sk, head);
- sock_prot_inuse_add(sk->sk_prot, 1);
+ sock_prot_inuse_add(sock_net(sk), sk->sk_prot, 1);
  write_unlock_bh(&h->lock);
 }
 EXPORT_SYMBOL_GPL(raw_hash_sk);
@@ -104,7 +104,7 @@ void raw_unhash_sk(struct sock *sk)

  write_lock_bh(&h->lock);
  if (sk_del_node_init(sk))
-  sock_prot_inuse_add(sk->sk_prot, -1);
+  sock_prot_inuse_add(sock_net(sk), sk->sk_prot, -1);
  write_unlock_bh(&h->lock);
 }
 EXPORT_SYMBOL_GPL(raw_unhash_sk);
diff --git a/net/ipv4/udp.c b/net/ipv4/udp.c
index 9143645..03bd706 100644
--- a/net/ipv4/udp.c
+++ b/net/ipv4/udp.c
@@ -231,7 +231,7 @@ gotit:
  if (sk_unhashed(sk)) {
   head = &udptable[snum & (UDP_HTABLE_SIZE - 1)];
   sk_add_node(sk, head);
-  sock_prot_inuse_add(sk->sk_prot, 1);
```

```
+  sock_prot_inuse_add(sock_net(sk), sk->sk_prot, 1);
  }
  error = 0;
 fail:
diff --git a/net/ipv6/inet6_hashtables.c b/net/ipv6/inet6_hashtables.c
index 340c7d4..580014a 100644
--- a/net/ipv6/inet6_hashtables.c
+++ b/net/ipv6/inet6_hashtables.c
@@ -43,7 +43,7 @@ void __inet6_hash(struct sock *sk)
  }

  __sk_add_node(sk, list);
- sock_prot_inuse_add(sk->sk_prot, 1);
+ sock_prot_inuse_add(sock_net(sk), sk->sk_prot, 1);
  write_unlock(lock);
 }
 EXPORT_SYMBOL(__inet6_hash);
@@ -204,7 +204,7 @@ unique:
  BUG_TRAP(sk_unhashed(sk));
  __sk_add_node(sk, &head->chain);
  sk->sk_hash = hash;
- sock_prot_inuse_add(sk->sk_prot, 1);
+ sock_prot_inuse_add(sock_net(sk), sk->sk_prot, 1);
  write_unlock(lock);

  if (twp != NULL) {
diff --git a/net/ipv6/ipv6_sockglue.c b/net/ipv6/ipv6_sockglue.c
index d3d93d7..4195ac9 100644
--- a/net/ipv6/ipv6_sockglue.c
+++ b/net/ipv6/ipv6_sockglue.c
@@ -155,10 +155,11 @@ static int do_ipv6_setsockopt(struct sock *sk, int level, int optname,

    if (sk->sk_protocol == IPPROTO_TCP) {
     struct inet_connection_sock *icsk = inet_csk(sk);
+    struct net *net = sock_net(sk);

     local_bh_disable();
-    sock_prot_inuse_add(sk->sk_prot, -1);
-    sock_prot_inuse_add(&tcp_prot, 1);
+    sock_prot_inuse_add(net, sk->sk_prot, -1);
+    sock_prot_inuse_add(net, &tcp_prot, 1);
     local_bh_enable();
     sk->sk_prot = &tcp_prot;
     icsk->icsk_af_ops = &ipv4_specific;
@@ -167,12 +168,13 @@ static int do_ipv6_setsockopt(struct sock *sk, int level, int optname,
     tcp_sync_mss(sk, icsk->icsk_pmtu_cookie);
    } else {
     struct proto *prot = &udp_prot;
```

```
+    struct net *net = sock_net(sk);

     if (sk->sk_protocol == IPPROTO_UDPLITE)
      prot = &udplite_prot;
     local_bh_disable();
-    sock_prot_inuse_add(sk->sk_prot, -1);
-    sock_prot_inuse_add(prot, 1);
+    sock_prot_inuse_add(net, sk->sk_prot, -1);
+    sock_prot_inuse_add(net, prot, 1);
     local_bh_enable();
     sk->sk_prot = prot;
     sk->sk_socket->ops = &inet_dgram_ops;
diff --git a/net/ipv6/proc.c b/net/ipv6/proc.c
index 364dc33..4b9d5a9 100644
--- a/net/ipv6/proc.c
+++ b/net/ipv6/proc.c
@@ -36,13 +36,13 @@ static struct proc_dir_entry *proc_net_devsnmp6;
 static int sockstat6_seq_show(struct seq_file *seq, void *v)
 {
  seq_printf(seq, "TCP6: inuse %d\n",
-         sock_prot_inuse_get(&tcpv6_prot));
+         sock_prot_inuse_get(&init_net, &tcpv6_prot));
  seq_printf(seq, "UDP6: inuse %d\n",
-         sock_prot_inuse_get(&udpv6_prot));
+         sock_prot_inuse_get(&init_net, &udpv6_prot));
  seq_printf(seq, "UDPLITE6: inuse %d\n",
-   sock_prot_inuse_get(&udplitev6_prot));
+   sock_prot_inuse_get(&init_net, &udplitev6_prot));
  seq_printf(seq, "RAW6: inuse %d\n",
-         sock_prot_inuse_get(&rawv6_prot));
+         sock_prot_inuse_get(&init_net, &rawv6_prot));
  seq_printf(seq, "FRAG6: inuse %d memory %d\n",
         ip6_frag_nqueues(&init_net), ip6_frag_mem(&init_net));
  return 0;
--
1.5.3.4
```

---

## Subject: Re: [PATCH net-2.6.26 2/6][SOCK][NETNS]: Add a struct net argument to sock_prot_inuse_add and _get.

Posted by davem on Tue, 01 Apr 2008 03:19:07 GMT

View Forum Message <> Reply to Message

From: Pavel Emelyanov <xemul@openvz.org>
Date: Mon, 31 Mar 2008 16:12:01 +0400

> This counter is about to become per-proto-and-per-net, so we'll need
> two arguments to determine which cell in this "table" to work with.

>
> All the places, but proc already pass proper net to it - proc will be
> tuned a bit later.
>
> Some indentation with spaces in proc files is done to keep the file
> coding style consistent.
>
> Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

Applied.

---