Subject: [PATCH net-2.6.25][NEIGH] Make neigh_add_timer symmetrical to neigh_del_timer
Posted by Pavel Emelianov on Thu, 20 Dec 2007 09:50:37 GMT
View Forum Message <> Reply to Message

The neigh_del_timer() looks sane - it removes the timer and
(conditionally) puts the neighbor. I expected, that the
neigh_add_timer() is symmetrical to the del one - i.e. it
holds the neighbor and arms the timer - but it turned out
that it was not so.

I think, that making them look symmetrical makes the code
more readable.

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

```
diff --git a/net/core/neighbour.c b/net/core/neighbour.c
index 4b6dd1e..9a283fc 100644
--- a/net/core/neighbour.c
+++ b/net/core/neighbour.c
@@ -165,6 +165,16 @@ static int neigh_forced_gc(struct neigh_table *tbl)
  return shrunk;
 }

+static void neigh_add_timer(struct neighbour *n, unsigned long when)
+{
+ neigh_hold(n);
+ if (unlikely(mod_timer(&n->timer, when))) {
+  printk("NEIGH: BUG, double timer add, state is %x\n",
+         n->nud_state);
+  dump_stack();
+ }
+}
+
 static int neigh_del_timer(struct neighbour *n)
 {
  if ((n->nud_state & NUD_IN_TIMER) &&
@@ -716,15 +726,6 @@ static __inline__ int neigh_max_probes(struct neighbour *n)
   p->ucast_probes + p->app_probes + p->mcast_probes);
 }

-static inline void neigh_add_timer(struct neighbour *n, unsigned long when)
-{
- if (unlikely(mod_timer(&n->timer, when))) {
-  printk("NEIGH: BUG, double timer add, state is %x\n",
```

```
-         n->nud_state);
-  dump_stack();
- }
-}
-
 /* Called when a timer expires for a neighbour entry. */

 static void neigh_timer_handler(unsigned long arg)
@@ -856,7 +857,6 @@ int __neigh_event_send(struct neighbour *neigh, struct sk_buff *skb)
    atomic_set(&neigh->probes, neigh->parms->ucast_probes);
    neigh->nud_state     = NUD_INCOMPLETE;
    neigh->updated = jiffies;
-   neigh_hold(neigh);
    neigh_add_timer(neigh, now + 1);
   } else {
    neigh->nud_state = NUD_FAILED;
@@ -869,7 +869,6 @@ int __neigh_event_send(struct neighbour *neigh, struct sk_buff *skb)
   }
  } else if (neigh->nud_state & NUD_STALE) {
   NEIGH_PRINTK2("neigh %p is delayed.\n", neigh);
-  neigh_hold(neigh);
   neigh->nud_state = NUD_DELAY;
   neigh->updated = jiffies;
   neigh_add_timer(neigh,
@@ -1013,13 +1012,11 @@ int neigh_update(struct neighbour *neigh, const u8 *lladdr, u8 new,

  if (new != old) {
   neigh_del_timer(neigh);
-  if (new & NUD_IN_TIMER) {
-   neigh_hold(neigh);
+  if (new & NUD_IN_TIMER)
    neigh_add_timer(neigh, (jiffies +
       ((new & NUD_REACHABLE) ?
        neigh->parms->reachable_time :
        0)));
-  }
   neigh->nud_state = new;
  }
```