Subject: [PATCH] Make helper to get dst entry and "use" is
Posted by Pavel Emelianov on Sat, 10 Nov 2007 14:29:52 GMT
View Forum Message <> Reply to Message

There are many places that get the dst entry, increase the
__use counter and set the "lastuse" time stamp.

Make a helper for this.

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

```
diff --git a/include/net/dst.h b/include/net/dst.h
index e9ff4a4..2f65e89 100644
--- a/include/net/dst.h
+++ b/include/net/dst.h
@@ -143,6 +143,13 @@ static inline void dst_hold(struct dst_entry * dst)
  atomic_inc(&dst->__refcnt);
 }

+static inline void dst_use(struct dst_entry *dst, unsigned long time)
+{
+ dst_hold(dst);
+ dst->__use++;
+ dst->lastuse = time;
+}
+
 static inline
 struct dst_entry * dst_clone(struct dst_entry * dst)
 {
diff --git a/net/decnet/dn_route.c b/net/decnet/dn_route.c
index 97eee5e..66663e5 100644
--- a/net/decnet/dn_route.c
+++ b/net/decnet/dn_route.c
@@ -293,9 +293,7 @@ static int dn_insert_route(struct dn_route *rt, unsigned hash, struct
dn_route *
         dn_rt_hash_table[hash].chain);
    rcu_assign_pointer(dn_rt_hash_table[hash].chain, rth);

-   rth->u.dst.__use++;
-   dst_hold(&rth->u.dst);
-   rth->u.dst.lastuse = now;
+   dst_use(&rth->u.dst, now);
    spin_unlock_bh(&dn_rt_hash_table[hash].lock);

    dnrt_drop(rt);
@@ -308,9 +306,7 @@ static int dn_insert_route(struct dn_route *rt, unsigned hash, struct
```

```
dn_route *
  rcu_assign_pointer(rt->u.dst.dn_next, dn_rt_hash_table[hash].chain);
  rcu_assign_pointer(dn_rt_hash_table[hash].chain, rt);

- dst_hold(&rt->u.dst);
- rt->u.dst.__use++;
- rt->u.dst.lastuse = now;
+ dst_use(&rt->u.dst, now);
  spin_unlock_bh(&dn_rt_hash_table[hash].lock);
  *rp = rt;
  return 0;
@@ -1182,9 +1178,7 @@ static int __dn_route_output_key(struct dst_entry **pprt, const struct
flowi *fl
        (flp->mark == rt->fl.mark) &&
        (rt->fl.iif == 0) &&
        (rt->fl.oif == flp->oif)) {
-    rt->u.dst.lastuse = jiffies;
-    dst_hold(&rt->u.dst);
-    rt->u.dst.__use++;
+    dst_use(&rt->u.dst, jiffies);
     rcu_read_unlock_bh();
     *pprt = &rt->u.dst;
     return 0;
@@ -1456,9 +1450,7 @@ int dn_route_input(struct sk_buff *skb)
       (rt->fl.oif == 0) &&
       (rt->fl.mark == skb->mark) &&
       (rt->fl.iif == cb->iif)) {
-   rt->u.dst.lastuse = jiffies;
-   dst_hold(&rt->u.dst);
-   rt->u.dst.__use++;
+   dst_use(&rt->u.dst, jiffies);
    rcu_read_unlock();
    skb->dst = (struct dst_entry *)rt;
    return 0;
diff --git a/net/ipv4/route.c b/net/ipv4/route.c
index c95b270..4565183 100644
--- a/net/ipv4/route.c
+++ b/net/ipv4/route.c
@@ -851,9 +851,7 @@ restart:
     */
   rcu_assign_pointer(rt_hash_table[hash].chain, rth);

-  rth->u.dst.__use++;
-  dst_hold(&rth->u.dst);
-  rth->u.dst.lastuse = now;
+  dst_use(&rth->u.dst, now);
   spin_unlock_bh(rt_hash_lock_addr(hash));
```

```
    rt_drop(rt);
@@ -1930,9 +1928,7 @@ int ip_route_input(struct sk_buff *skb, __be32 daddr, __be32 saddr,
        rth->fl.oif == 0 &&
        rth->fl.mark == skb->mark &&
        rth->fl.fl4_tos == tos) {
-   rth->u.dst.lastuse = jiffies;
-   dst_hold(&rth->u.dst);
-   rth->u.dst.__use++;
+   dst_use(&rth->u.dst, jiffies);
    RT_CACHE_STAT_INC(in_hit);
    rcu_read_unlock();
    skb->dst = (struct dst_entry*)rth;
@@ -2326,9 +2322,7 @@ int __ip_route_output_key(struct rtable **rp, const struct flowi *flp)
        rth->fl.mark == flp->mark &&
        !((rth->fl.fl4_tos ^ flp->fl4_tos) &
         (IPTOS_RT_MASK | RTO_ONLINK))) {
-   rth->u.dst.lastuse = jiffies;
-   dst_hold(&rth->u.dst);
-   rth->u.dst.__use++;
+   dst_use(&rth->u.dst, jiffies);
    RT_CACHE_STAT_INC(out_hit);
    rcu_read_unlock_bh();
    *rp = rth;
diff --git a/net/ipv6/route.c b/net/ipv6/route.c
index 973a97a..6ecb5e6 100644
--- a/net/ipv6/route.c
+++ b/net/ipv6/route.c
@@ -544,12 +544,8 @@ restart:
  rt = rt6_device_match(rt, fl->oif, flags);
  BACKTRACK(&fl->fl6_src);
 out:
- dst_hold(&rt->u.dst);
+ dst_use(&rt->u.dst, jiffies);
  read_unlock_bh(&table->tb6_lock);
-
- rt->u.dst.lastuse = jiffies;
- rt->u.dst.__use++;
-
  return rt;

 }
--
1.5.3.4
```