Subject: [PATCH rhel6] Allow ipv6 proxies and arp proxies be shown with iproute2
Posted by Tony Zelenoff on Thu, 26 Jan 2012 14:49:52 GMT
View Forum Message <> Reply to Message

Add ability to return neighbour proxies list to caller if
it sent full ndmsg structure and has NTF_PROXY flag set.

Before this patch (and before iproute2 patches):
$ ip neigh add proxy 2001::1 dev eth0
$ ip -6 neigh show
$

After it and with applied iproute2 patches:
$ ip neigh add proxy 2001::1 dev eth0
$ ip -6 neigh show
2001::1 dev eth0 proxy
$

Compatibility with old versions of iproute2 is not broken,
kernel checks for incoming structure size and properly
works if old structure is came.

CC: davem@davemloft.net
CC: kuznet@ms2.inr.ac.ru
CC: netdev@vger.kernel.org
CC: xemul@parallels.com
Signed-off-by: Tony Zelenoff <antonz@parallels.com>
---
 net/core/neighbour.c | 90 ++--
 1 files changed, 87 insertions(+), 3 deletions(-)

diff --git a/net/core/neighbour.c b/net/core/neighbour.c
index 6dbcf51..009aad0 100644
--- a/net/core/neighbour.c
+++ b/net/core/neighbour.c
@@ -2111,6 +2111,35 @@ nla_put_failure:
  return -EMSGSIZE;
 }

+static int pneigh_fill_info(struct sk_buff *skb, struct pneigh_entry *pn,
+   u32 pid, u32 seq, int type, unsigned int flags,
+   struct neigh_table *tbl)
+{
+ struct nlmsghdr *nlh;
+ struct ndmsg *ndm;
+
+ nlh = nlmsg_put(skb, pid, seq, type, sizeof(*ndm), flags);
+ if (nlh == NULL)

```
+  return -EMSGSIZE;
+
+ ndm = nlmsg_data(nlh);
+ ndm->ndm_family  = tbl->family;
+ ndm->ndm_pad1    = 0;
+ ndm->ndm_pad2    = 0;
+ ndm->ndm_flags  = pn->flags | NTF_PROXY;
+ ndm->ndm_type  = NDA_DST;
+ ndm->ndm_ifindex = pn->dev->ifindex;
+ ndm->ndm_state  = NUD_NONE;
+
+ NLA_PUT(skb, NDA_DST, tbl->key_len, pn->key);
+
+ return nlmsg_end(skb, nlh);
+
+nla_put_failure:
+ nlmsg_cancel(skb, nlh);
+ return -EMSGSIZE;
+}
+
 static void neigh_update_notify(struct neighbour *neigh)
 {
  call_netevent_notifiers(NETEVENT_NEIGH_UPDATE, neigh);
@@ -2156,23 +2185,78 @@ out:
  return rc;
 }

+static int pneigh_dump_table(struct neigh_table *tbl, struct sk_buff *skb,
+        struct netlink_callback *cb)
+{
+ struct pneigh_entry *n;
+ struct net *net = sock_net(skb->sk);
+ int rc, h, s_h = cb->args[3];
+ int idx, s_idx = idx = cb->args[4];
+
+ read_lock_bh(&tbl->lock);
+
+ for (h = 0; h <= PNEIGH_HASHMASK; h++) {
+  if (h < s_h)
+   continue;
+  if (h > s_h)
+   s_idx = 0;
+  for (n = tbl->phash_buckets[h], idx = 0; n; n = n->next) {
+   if (dev_net(n->dev) != net)
+    continue;
+   if (idx < s_idx)
+    goto next;
+   if (pneigh_fill_info(skb, n, NETLINK_CB(cb->skb).pid,
```

```
+         cb->nlh->nlmsg_seq,
+         RTM_NEWNEIGH,
+         NLM_F_MULTI, tbl) <= 0) {
+     read_unlock_bh(&tbl->lock);
+     rc = -1;
+     goto out;
+    }
+  next:
+    idx++;
+  }
+ }
+
+ read_unlock_bh(&tbl->lock);
+ rc = skb->len;
+out:
+ cb->args[3] = h;
+ cb->args[4] = idx;
+ return rc;
+
+}
+
 static int neigh_dump_info(struct sk_buff *skb, struct netlink_callback *cb)
 {
  struct neigh_table *tbl;
  int t, family, s_t;
+ int proxy = 0;
+ int err = 0;

  read_lock(&neigh_tbl_lock);
  family = ((struct rtgenmsg *) nlmsg_data(cb->nlh))->rtgen_family;
+
+ /* check for full ndmsg structure presence, family member is
+  * the same for both structures */
+ if (nlmsg_len(cb->nlh) == sizeof(struct ndmsg) &&
+     ((struct ndmsg *) nlmsg_data(cb->nlh))->ndm_flags == NTF_PROXY)
+  proxy = 1;
+
  s_t = cb->args[0];

- for (tbl = neigh_tables, t = 0; tbl; tbl = tbl->next, t++) {
+ for (tbl = neigh_tables, t = 0; tbl && (err >= 0);
+      tbl = tbl->next, t++) {
   if (t < s_t || (family && tbl->family != family))
    continue;
   if (t > s_t)
    memset(&cb->args[1], 0, sizeof(cb->args) -
       sizeof(cb->args[0]));
-  if (neigh_dump_table(tbl, skb, cb) < 0)
```

```
-   break;
+  if (proxy) {
+   err = pneigh_dump_table(tbl, skb, cb);
+   continue;
+  }
+  err = neigh_dump_table(tbl, skb, cb);
  }
  read_unlock(&neigh_tbl_lock);

--
1.7.1
```