Subject: [patch 33/38][IPV6] rt6_info - move rt6_info structure inside the namespace
Posted by Daniel Lezcano on Mon, 03 Dec 2007 16:17:09 GMT
View Forum Message <> Reply to Message

The rt6_info structures are moved inside the network namespace
structure. All references to these structures are now relative
to the initial network namespace.

Signed-off-by: Daniel Lezcano <dlezcano@fr.ibm.com>
Signed-off-by: Benjamin Thery <benjamin.thery@bull.net>

```
---
 include/net/net_namespace.h |    3 +
 net/ipv6/addrconf.c         |   12 ++--
 net/ipv6/fib6_rules.c       |   13 ++--
 net/ipv6/ip6_fib.c          |   47 ++++++++-------
 net/ipv6/route.c            |  132 +++++++++++++++++++++++++-------------------
 5 files changed, 117 insertions(+), 90 deletions(-)

Index: linux-2.6-netns/include/net/net_namespace.h
===================================================================
--- linux-2.6-netns.orig/include/net/net_namespace.h
+++ linux-2.6-netns/include/net/net_namespace.h
@@ -47,9 +47,12 @@ struct net {
 	/* ipv6 routing table */
 #if defined(CONFIG_IPV6) || defined(CONFIG_IPV6_MODULE)
 	struct rt6_statistics	*rt6_stats;
+	struct rt6_info		*ip6_null_entry;
 	struct hlist_head	*fib_table_hash;
 	struct fib6_table	*fib6_main_tbl;
 #ifdef CONFIG_IPV6_MULTIPLE_TABLES
+	struct rt6_info		*ip6_prohibit_entry;
+	struct rt6_info		*ip6_blk_hole_entry;
 	struct fib6_table	*fib6_local_tbl;
 #endif /* CONFIG_IPV6_MULTIPLE_TABLES */

Index: linux-2.6-netns/net/ipv6/addrconf.c
===================================================================
--- linux-2.6-netns.orig/net/ipv6/addrconf.c
+++ linux-2.6-netns/net/ipv6/addrconf.c
@@ -4265,13 +4265,13 @@ int __init addrconf_init(void)
 	if (err)
 		return err;

-	ip6_null_entry->u.dst.dev = init_net.loopback_dev;
-	ip6_null_entry->rt6i_idev = in6_dev_get(init_net.loopback_dev);
+	init_net.ip6_null_entry->u.dst.dev = init_net.loopback_dev;
+	init_net.ip6_null_entry->rt6i_idev = in6_dev_get(init_net.loopback_dev);
 #ifdef CONFIG_IPV6_MULTIPLE_TABLES
```

```
- ip6_prohibit_entry->u.dst.dev = init_net.loopback_dev;
- ip6_prohibit_entry->rt6i_idev = in6_dev_get(init_net.loopback_dev);
- ip6_blk_hole_entry->u.dst.dev = init_net.loopback_dev;
- ip6_blk_hole_entry->rt6i_idev = in6_dev_get(init_net.loopback_dev);
+ init_net.ip6_prohibit_entry->u.dst.dev = init_net.loopback_dev;
+ init_net.ip6_prohibit_entry->rt6i_idev = in6_dev_get(init_net.loopback_dev);
+ init_net.ip6_blk_hole_entry->u.dst.dev = init_net.loopback_dev;
+ init_net.ip6_blk_hole_entry->rt6i_idev = in6_dev_get(init_net.loopback_dev);
 #endif

  register_netdevice_notifier(&ipv6_dev_notf);
Index: linux-2.6-netns/net/ipv6/fib6_rules.c
===================================================================
--- linux-2.6-netns.orig/net/ipv6/fib6_rules.c
+++ linux-2.6-netns/net/ipv6/fib6_rules.c
@@ -34,6 +34,7 @@ static struct fib_rules_ops fib6_rules_o
 struct dst_entry *fib6_rule_lookup(struct flowi *fl, int flags,
        pol_lookup_t lookup)
 {
+ struct net *net = fl->fl_net;
  struct fib_lookup_arg arg = {
   .lookup_ptr = lookup,
  };
@@ -45,8 +46,8 @@ struct dst_entry *fib6_rule_lookup(struc
  if (arg.result)
   return arg.result;

- dst_hold(&ip6_null_entry->u.dst);
- return &ip6_null_entry->u.dst;
+ dst_hold(&net->ip6_null_entry->u.dst);
+ return &net->ip6_null_entry->u.dst;
 }

 static int fib6_rule_action(struct fib_rule *rule, struct flowi *flp,
@@ -61,14 +62,14 @@ static int fib6_rule_action(struct fib_r
  case FR_ACT_TO_TBL:
   break;
  case FR_ACT_UNREACHABLE:
-  rt = ip6_null_entry;
+  rt = net->ip6_null_entry;
   goto discard_pkt;
  default:
  case FR_ACT_BLACKHOLE:
-  rt = ip6_blk_hole_entry;
+  rt = net->ip6_blk_hole_entry;
   goto discard_pkt;
  case FR_ACT_PROHIBIT:
-  rt = ip6_prohibit_entry;
```

```
+  rt = net->ip6_prohibit_entry;
   goto discard_pkt;
  }

@@ -76,7 +77,7 @@ static int fib6_rule_action(struct fib_r
  if (table)
   rt = lookup(table, flp, flags);

- if (rt != ip6_null_entry) {
+ if (rt != net->ip6_null_entry) {
   struct fib6_rule *r = (struct fib6_rule *)rule;

   /*
Index: linux-2.6-netns/net/ipv6/ip6_fib.c
===================================================================
--- linux-2.6-netns.orig/net/ipv6/ip6_fib.c
+++ linux-2.6-netns/net/ipv6/ip6_fib.c
@@ -77,8 +77,8 @@ static DEFINE_RWLOCK(fib6_walker_lock);
 #endif

 static void fib6_prune_clones(struct fib6_node *fn, struct rt6_info *rt);
-static struct rt6_info * fib6_find_prefix(struct fib6_node *fn);
-static struct fib6_node * fib6_repair_tree(struct fib6_node *fn);
+static struct rt6_info * fib6_find_prefix(struct net *net, struct fib6_node *fn);
+static struct fib6_node * fib6_repair_tree(struct net *net, struct fib6_node *fn);
 static int fib6_walk(struct fib6_walker_t *w);
 static int fib6_walk_continue(struct fib6_walker_t *w);

@@ -189,14 +189,14 @@ static void fib6_link_table(struct net *

 #ifdef CONFIG_IPV6_MULTIPLE_TABLES

-static struct fib6_table *fib6_alloc_table(u32 id)
+static struct fib6_table *fib6_alloc_table(struct net *net, u32 id)
 {
  struct fib6_table *table;

  table = kzalloc(sizeof(*table), GFP_ATOMIC);
  if (table != NULL) {
   table->tb6_id = id;
-  table->tb6_root.leaf = ip6_null_entry;
+  table->tb6_root.leaf = net->ip6_null_entry;
   table->tb6_root.fn_flags = RTN_ROOT | RTN_TL_ROOT | RTN_RTINFO;
  }

@@ -213,7 +213,7 @@ struct fib6_table *fib6_new_table(struct
  if (tb)
   return tb;
```

```
- tb = fib6_alloc_table(id);
+ tb = fib6_alloc_table(net, id);
  if (tb != NULL)
   fib6_link_table(net, tb);

@@ -681,6 +681,7 @@ void fib6_force_start_gc(void)
 int fib6_add(struct fib6_node *root, struct rt6_info *rt, struct nl_info *info)
 {
  struct fib6_node *fn, *pn = NULL;
+ struct net *net = info->net;
  int err = -ENOMEM;

  fn = fib6_add_1(root, &rt->rt6i_dst.addr, sizeof(struct in6_addr),
@@ -713,8 +714,8 @@ int fib6_add(struct fib6_node *root, str
    if (sfn == NULL)
     goto st_failure;

-   sfn->leaf = ip6_null_entry;
-   atomic_inc(&ip6_null_entry->rt6i_ref);
+   sfn->leaf = net->ip6_null_entry;
+   atomic_inc(&net->ip6_null_entry->rt6i_ref);
    sfn->fn_flags = RTN_ROOT;
    sfn->fn_sernum = fib6_new_sernum();

@@ -769,11 +770,11 @@ out:
    * super-tree leaf node we have to find a new one for it.
    */
   if (pn != fn && !pn->leaf && !(pn->fn_flags & RTN_RTINFO)) {
-   pn->leaf = fib6_find_prefix(pn);
+   pn->leaf = fib6_find_prefix(net, pn);
 #if RT6_DEBUG >= 2
    if (!pn->leaf) {
     BUG_TRAP(pn->leaf != NULL);
-    pn->leaf = ip6_null_entry;
+    pn->leaf = net->ip6_null_entry;
    }
 #endif
    atomic_inc(&pn->leaf->rt6i_ref);
@@ -789,7 +790,7 @@ out:
   */
 st_failure:
  if (fn && !(fn->fn_flags & (RTN_RTINFO|RTN_ROOT)))
-  fib6_repair_tree(fn);
+  fib6_repair_tree(net, fn);
  dst_free(&rt->u.dst);
  return err;
 #endif
```

```
@@ -854,7 +855,6 @@ static struct fib6_node * fib6_lookup_1(

   if (fn->fn_flags & RTN_ROOT)
    break;
-
   fn = fn->parent;
  }

@@ -955,10 +955,10 @@ struct fib6_node * fib6_locate(struct fi
  *
  */

-static struct rt6_info * fib6_find_prefix(struct fib6_node *fn)
+static struct rt6_info * fib6_find_prefix(struct net *net, struct fib6_node *fn)
 {
  if (fn->fn_flags&RTN_ROOT)
-  return ip6_null_entry;
+  return net->ip6_null_entry;

  while(fn) {
   if(fn->left)
@@ -977,7 +977,7 @@ static struct rt6_info * fib6_find_prefi
  * is the node we want to try and remove.
  */

-static struct fib6_node * fib6_repair_tree(struct fib6_node *fn)
+static struct fib6_node * fib6_repair_tree(struct net *net, struct fib6_node *fn)
 {
  int children;
  int nstate;
@@ -1004,11 +1004,11 @@ static struct fib6_node * fib6_repair_tr
       || (children && fn->fn_flags&RTN_ROOT)
 #endif
       ) {
-   fn->leaf = fib6_find_prefix(fn);
+   fn->leaf = fib6_find_prefix(net, fn);
 #if RT6_DEBUG >= 2
    if (fn->leaf==NULL) {
     BUG_TRAP(fn->leaf);
-    fn->leaf = ip6_null_entry;
+    fn->leaf = net->ip6_null_entry;
    }
 #endif
    atomic_inc(&fn->leaf->rt6i_ref);
@@ -1079,9 +1079,9 @@ static struct fib6_node * fib6_repair_tr
 static void fib6_del_route(struct fib6_node *fn, struct rt6_info **rtp,
       struct nl_info *info)
 {
```

```
+ struct net *net = info->net;
  struct fib6_walker_t *w;
  struct rt6_info *rt = *rtp;
- struct net *net = &init_net;

  RT6_TRACE("fib6_del_route\n");

@@ -1110,13 +1110,13 @@ static void fib6_del_route(struct fib6_n
  rt->u.dst.rt6_next = NULL;

  if (fn->leaf == NULL && fn->fn_flags&RTN_TL_ROOT)
-  fn->leaf = ip6_null_entry;
+  fn->leaf = net->ip6_null_entry;

  /* If it was last route, expunge its radix tree node */
  if (fn->leaf == NULL) {
   fn->fn_flags &= ~RTN_RTINFO;
   net->rt6_stats->fib_route_nodes--;
-  fn = fib6_repair_tree(fn);
+  fn = fib6_repair_tree(net, fn);
  }

  if (atomic_read(&rt->rt6i_ref) != 1) {
@@ -1128,7 +1128,7 @@ static void fib6_del_route(struct fib6_n
    */
   while (fn) {
    if (!(fn->fn_flags&RTN_RTINFO) && fn->leaf == rt) {
-    fn->leaf = fib6_find_prefix(fn);
+    fn->leaf = fib6_find_prefix(net, fn);
     atomic_inc(&fn->leaf->rt6i_ref);
     rt6_release(rt);
    }
@@ -1144,6 +1144,7 @@ static void fib6_del_route(struct fib6_n

 int fib6_del(struct rt6_info *rt, struct nl_info *info)
 {
+ struct net *net = info->net;
  struct fib6_node *fn = rt->rt6i_node;
  struct rt6_info **rtp;

@@ -1153,7 +1154,7 @@ int fib6_del(struct rt6_info *rt, struct
   return -ENOENT;
  }
 #endif
- if (fn == NULL || rt == ip6_null_entry)
+ if (fn == NULL || rt == net->ip6_null_entry)
   return -ENOENT;
```

```
  BUG_TRAP(fn->fn_flags&RTN_RTINFO);
@@ -1498,7 +1499,7 @@ static int fib6_net_init(struct net *net
    goto out_fib6_main_tbl;

  net->fib6_main_tbl->tb6_id = RT6_TABLE_MAIN;
- net->fib6_main_tbl->tb6_root.leaf = ip6_null_entry;
+ net->fib6_main_tbl->tb6_root.leaf = net->ip6_null_entry;
  net->fib6_main_tbl->tb6_root.fn_flags = RTN_ROOT | RTN_TL_ROOT | RTN_RTINFO;

 #ifdef CONFIG_IPV6_MULTIPLE_TABLES
@@ -1508,7 +1509,7 @@ static int fib6_net_init(struct net *net
    goto out_fib6_main_tbl;
  }
  net->fib6_local_tbl->tb6_id = RT6_TABLE_LOCAL;
- net->fib6_local_tbl->tb6_root.leaf = ip6_null_entry;
+ net->fib6_local_tbl->tb6_root.leaf = net->ip6_null_entry;
  net->fib6_local_tbl->tb6_root.fn_flags = RTN_ROOT | RTN_TL_ROOT | RTN_RTINFO;
 #endif

Index: linux-2.6-netns/net/ipv6/route.c
===================================================================
--- linux-2.6-netns.orig/net/ipv6/route.c
+++ linux-2.6-netns/net/ipv6/route.c
@@ -150,8 +150,6 @@ static struct rt6_info ip6_null_entry_te
  .rt6i_ref = ATOMIC_INIT(1),
 };

-struct rt6_info *ip6_null_entry;
-
 #ifdef CONFIG_IPV6_MULTIPLE_TABLES

 static int ip6_pkt_prohibit(struct sk_buff *skb);
@@ -175,8 +173,6 @@ struct rt6_info ip6_prohibit_entry_templ
  .rt6i_ref       = ATOMIC_INIT(1),
 };

-struct rt6_info *ip6_prohibit_entry;
-
 static struct rt6_info ip6_blk_hole_entry_template = {
  .u = {
   .dst = {
@@ -195,8 +191,6 @@ static struct rt6_info ip6_blk_hole_entr
  .rt6i_ref = ATOMIC_INIT(1),
 };

-struct rt6_info *ip6_blk_hole_entry;
-
 #endif
```

```
 /* allocate dst with ip6_dst_ops */
@@ -250,7 +244,8 @@ static inline int rt6_need_strict(struct
  * Route lookup. Any table->tb6_lock is implied.
  */

-static __inline__ struct rt6_info *rt6_device_match(struct rt6_info *rt,
+static __inline__ struct rt6_info *rt6_device_match(struct net *net,
+          struct rt6_info *rt,
           int oif,
           int strict)
 {
@@ -279,7 +274,7 @@ static __inline__ struct rt6_info *rt6_d
    return local;

   if (strict)
-   return ip6_null_entry;
+   return net->ip6_null_entry;
  }
  return rt;
 }
@@ -412,7 +407,8 @@ static struct rt6_info *find_rr_leaf(str
  return match;
 }

-static struct rt6_info *rt6_select(struct fib6_node *fn, int oif, int strict)
+static struct rt6_info *rt6_select(struct net *net,
+       struct fib6_node *fn, int oif, int strict)
 {
  struct rt6_info *match, *rt0;

@@ -440,7 +436,7 @@ static struct rt6_info *rt6_select(struc
  RT6_TRACE("%s() => %p\n",
     __FUNCTION__, match);

- return (match ? match : ip6_null_entry);
+ return (match ? match : net->ip6_null_entry);
 }

 #ifdef CONFIG_IPV6_ROUTE_INFO
@@ -523,9 +519,9 @@ int rt6_route_rcv(struct net_device *dev
 }
 #endif

-#define BACKTRACK(saddr) \
+#define BACKTRACK(__net, saddr)   \
 do { \
- if (rt == ip6_null_entry) { \
```

```
+ if (rt == __net->ip6_null_entry) { \
   struct fib6_node *pn; \
   while (1) { \
    if (fn->fn_flags & RTN_TL_ROOT) \
@@ -544,6 +540,7 @@ do { \
 static struct rt6_info *ip6_pol_route_lookup(struct fib6_table *table,
          struct flowi *fl, int flags)
 {
+ struct net *net = fl->fl_net;
  struct fib6_node *fn;
  struct rt6_info *rt;

@@ -551,8 +548,8 @@ static struct rt6_info *ip6_pol_route_lo
  fn = fib6_lookup(&table->tb6_root, &fl->fl6_dst, &fl->fl6_src);
 restart:
  rt = fn->leaf;
- rt = rt6_device_match(rt, fl->oif, flags);
- BACKTRACK(&fl->fl6_src);
+ rt = rt6_device_match(net, rt, fl->oif, flags);
+ BACKTRACK(net, &fl->fl6_src);
 out:
  dst_use(&rt->u.dst, jiffies);
  read_unlock_bh(&table->tb6_lock);
@@ -676,6 +673,7 @@ static struct rt6_info *rt6_alloc_clone(
 static struct rt6_info *ip6_pol_route(struct fib6_table *table, int oif,
         struct flowi *fl, int flags)
 {
+ struct net *net = fl->fl_net;
  struct fib6_node *fn;
  struct rt6_info *rt, *nrt;
  int strict = 0;
@@ -692,9 +690,9 @@ restart_2:
  fn = fib6_lookup(&table->tb6_root, &fl->fl6_dst, &fl->fl6_src);

 restart:
- rt = rt6_select(fn, oif, strict | reachable);
- BACKTRACK(&fl->fl6_src);
- if (rt == ip6_null_entry ||
+ rt = rt6_select(net, fn, oif, strict | reachable);
+ BACKTRACK(net, &fl->fl6_src);
+ if (rt == net->ip6_null_entry ||
     rt->rt6i_flags & RTF_CACHE)
   goto out;

@@ -712,7 +710,7 @@ restart:
  }

  dst_release(&rt->u.dst);
```

```
- rt = nrt ? : ip6_null_entry;
+ rt = nrt ? : net->ip6_null_entry;

  dst_hold(&rt->u.dst);
  if (nrt) {
@@ -1261,8 +1259,9 @@ static int __ip6_del_rt(struct rt6_info
 {
  int err;
  struct fib6_table *table;
+ struct net *net = rt->rt6i_dev->nd_net;

- if (rt == ip6_null_entry)
+ if (rt == net->ip6_null_entry)
   return -ENOENT;

  table = rt->rt6i_table;
@@ -1341,6 +1340,7 @@ static struct rt6_info *__ip6_route_redi
  struct ip6rd_flowi *rdfl = (struct ip6rd_flowi *)fl;
  struct rt6_info *rt;
  struct fib6_node *fn;
+ struct net *net = fl->fl_net;

  /*
   * Get the "current" route for this destination and
@@ -1377,8 +1377,8 @@ restart:
  }

  if (!rt)
-  rt = ip6_null_entry;
- BACKTRACK(&fl->fl6_src);
+  rt = net->ip6_null_entry;
+ BACKTRACK(fl->fl_net, &fl->fl6_src);
 out:
  dst_hold(&rt->u.dst);

@@ -1419,10 +1419,11 @@ void rt6_redirect(struct in6_addr *dest,
 {
  struct rt6_info *rt, *nrt = NULL;
  struct netevent_redirect netevent;
+ struct net *net = neigh->dev->nd_net;

  rt = ip6_route_redirect(dest, src, saddr, neigh->dev);

- if (rt == ip6_null_entry) {
+ if (rt == net->ip6_null_entry) {
   if (net_ratelimit())
    printk(KERN_DEBUG "rt6_redirect: source isn't a valid nexthop "
           "for redirect target\n");
```

```
@@ -1891,8 +1892,10 @@ struct rt6_info *addrconf_dst_alloc(stru

 static int fib6_ifdown(struct rt6_info *rt, void *arg)
 {
+ struct net *net = rt->rt6i_dev->nd_net;
+
  if (((void*)rt->rt6i_dev == arg || arg == NULL) &&
-     rt != ip6_null_entry) {
+     rt != net->ip6_null_entry) {
   RT6_TRACE("deleted by ifdown %p\n", rt);
   return -1;
  }
@@ -2513,15 +2516,63 @@ ctl_table ipv6_route_table[] = {

 static int ip6_route_net_init(struct net *net)
 {
+   int ret = 0;
+
+   ret = -ENOMEM;
+    net->ip6_null_entry = kzalloc(sizeof(*net->ip6_null_entry), GFP_KERNEL);
+    if (!net->ip6_null_entry)
+    goto out;
+
+    memcpy(net->ip6_null_entry, &ip6_null_entry_template,
+           sizeof(*net->ip6_null_entry));
+    net->ip6_null_entry->u.dst.path = (struct dst_entry*)net->ip6_null_entry;
+
+#ifdef CONFIG_IPV6_MULTIPLE_TABLES
+    net->ip6_prohibit_entry = kzalloc(sizeof(*net->ip6_prohibit_entry),
+          GFP_KERNEL);
+    if (!net->ip6_prohibit_entry)
+    goto out_ip6_prohibit_entry;
+
+    memcpy(net->ip6_prohibit_entry, &ip6_prohibit_entry_template,
+           sizeof(*net->ip6_prohibit_entry));
+    net->ip6_prohibit_entry->u.dst.path =
+    (struct dst_entry*)net->ip6_prohibit_entry;
+
+    net->ip6_blk_hole_entry = kzalloc(sizeof(*net->ip6_blk_hole_entry),
+          GFP_KERNEL);
+    if (!net->ip6_blk_hole_entry) {
+    kfree(net->ip6_prohibit_entry);
+    goto out_ip6_prohibit_entry;
+   }
+
+    memcpy(net->ip6_blk_hole_entry, &ip6_blk_hole_entry_template,
+           sizeof(*net->ip6_blk_hole_entry));
+    net->ip6_blk_hole_entry->u.dst.path =
```

```
+    (struct dst_entry*)net->ip6_blk_hole_entry;
+#endif
+
  proc_net_fops_create(net, "ipv6_route", 0, &ipv6_route_proc_fops);
  proc_net_fops_create(net, "rt6_stats", S_IRUGO, &rt6_stats_seq_fops);
- return 0;
+
+   ret = 0;
+out:
+   return ret;
+
+out_ip6_prohibit_entry:
+   kfree(net->ip6_null_entry);
+   goto out;
 }

 static void ip6_route_net_exit(struct net *net)
 {
  proc_net_remove(net, "ipv6_route");
  proc_net_remove(net, "rt6_stats");
+
+ kfree(net->ip6_null_entry);
+#ifdef CONFIG_IPV6_MULTIPLE_TABLES
+ kfree(net->ip6_prohibit_entry);
+ kfree(net->ip6_blk_hole_entry);
+#endif
 }

 static struct pernet_operations ip6_route_net_ops = {
@@ -2531,38 +2582,13 @@ static struct pernet_operations ip6_rout

 void __init ip6_route_init(void)
 {
-  ip6_null_entry = kzalloc(sizeof(*ip6_null_entry), GFP_KERNEL);
-  if (!ip6_null_entry)
-  panic("IPV6: cannot allocate ip6_null_entry\n");
-
-  memcpy(ip6_null_entry, &ip6_null_entry_template,
-         sizeof(*ip6_null_entry));
-  ip6_null_entry->u.dst.path = (struct dst_entry*)ip6_null_entry;
-
- #ifdef CONFIG_IPV6_MULTIPLE_TABLES
-  ip6_prohibit_entry = kzalloc(sizeof(*ip6_prohibit_entry), GFP_KERNEL);
-  if (!ip6_prohibit_entry)
-  panic("IPV6: cannot allocate ip6_prohibit_entry\n");
-
-  memcpy(ip6_prohibit_entry, &ip6_prohibit_entry_template,
-         sizeof(*ip6_prohibit_entry));
```

```
-  ip6_prohibit_entry->u.dst.path = (struct dst_entry*)ip6_prohibit_entry;
-
-  ip6_blk_hole_entry = kzalloc(sizeof(*ip6_blk_hole_entry), GFP_KERNEL);
-  if (!ip6_blk_hole_entry)
-  panic("IPV6: panic cannot allocate ip6_blk_hole_entry\n");
-
-  memcpy(ip6_blk_hole_entry, &ip6_blk_hole_entry_template,
-         sizeof(*ip6_blk_hole_entry));
-  ip6_blk_hole_entry->u.dst.path = (struct dst_entry*)ip6_blk_hole_entry;
- #endif
+ register_pernet_subsys(&ip6_route_net_ops);

  ip6_dst_ops.kmem_cachep =
   kmem_cache_create("ip6_dst_cache", sizeof(struct rt6_info), 0,
       SLAB_HWCACHE_ALIGN|SLAB_PANIC, NULL);
  ip6_dst_blackhole_ops.kmem_cachep = ip6_dst_ops.kmem_cachep;

- register_pernet_subsys(&ip6_route_net_ops);
  fib6_init();
 #ifdef CONFIG_XFRM
  xfrm6_init();
@@ -2578,8 +2604,6 @@ void __init ip6_route_init(void)

 void ip6_route_cleanup(void)
 {
- unregister_pernet_subsys(&ip6_route_net_ops);
-
 #ifdef CONFIG_IPV6_MULTIPLE_TABLES
  fib6_rules_cleanup();
 #endif
@@ -2590,7 +2614,5 @@ void ip6_route_cleanup(void)
  fib6_gc_cleanup();
  kmem_cache_destroy(ip6_dst_ops.kmem_cachep);

- kfree(ip6_null_entry);
- kfree(ip6_prohibit_entry);
- kfree(ip6_blk_hole_entry);
+ unregister_pernet_subsys(&ip6_route_net_ops);
 }

--
```