Subject: [patch 32/38][IPV6] rt6_info - dynamically allocate rt6_info
Posted by Daniel Lezcano on Mon, 03 Dec 2007 16:17:08 GMT
View Forum Message <> Reply to Message

The rt6_info structure are no longer a global static variable but
a dynamically allocated structure.
In order to facilitate future multiple instanciations of rt6_info
structures when multiple network namespaces will be supported, a
global static variable is still present as template for rt6_info
structure initialization.

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

```
---
 net/ipv6/route.c |   67 +++++++++++++++++++++++++++++++++++++++++++++++--------------------
 1 file changed, 47 insertions(+), 20 deletions(-)

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
@@ -132,7 +132,7 @@ static struct dst_ops ip6_dst_blackhole_
  .entry_size  = sizeof(struct rt6_info),
 };

-static struct rt6_info __ip6_null_entry = {
+static struct rt6_info ip6_null_entry_template = {
  .u = {
   .dst = {
    .__refcnt = ATOMIC_INIT(1),
@@ -143,7 +143,6 @@ static struct rt6_info __ip6_null_entry
    .input  = ip6_pkt_discard,
    .output  = ip6_pkt_discard_out,
    .ops  = &ip6_dst_ops,
-   .path  = (struct dst_entry*)&__ip6_null_entry,
   }
  },
  .rt6i_flags = (RTF_REJECT | RTF_NONEXTHOP),
@@ -151,35 +150,34 @@ static struct rt6_info __ip6_null_entry
  .rt6i_ref = ATOMIC_INIT(1),
 };

-struct rt6_info *ip6_null_entry = &__ip6_null_entry;
+struct rt6_info *ip6_null_entry;

 #ifdef CONFIG_IPV6_MULTIPLE_TABLES

 static int ip6_pkt_prohibit(struct sk_buff *skb);
 static int ip6_pkt_prohibit_out(struct sk_buff *skb);

-struct rt6_info __ip6_prohibit_entry = {
+struct rt6_info ip6_prohibit_entry_template = {
  .u = {
   .dst = {
-   .__refcnt = ATOMIC_INIT(1),
-   .__use  = 1,
-   .obsolete = -1,
-   .error  = -EACCES,
-   .metrics = { [RTAX_HOPLIMIT - 1] = 255, },
-   .input  = ip6_pkt_prohibit,
-   .output  = ip6_pkt_prohibit_out,
-   .ops  = &ip6_dst_ops,
-   .path  = (struct dst_entry*)&__ip6_prohibit_entry,
+   .__refcnt       = ATOMIC_INIT(1),
+   .__use          = 1,
+   .obsolete       = -1,
+   .error          = -EACCES,
+   .metrics        = { [RTAX_HOPLIMIT - 1] = 255, },
+   .input          = ip6_pkt_prohibit,
+   .output         = ip6_pkt_prohibit_out,
+   .ops            = &ip6_dst_ops,
   }
  },
- .rt6i_flags = (RTF_REJECT | RTF_NONEXTHOP),
- .rt6i_metric = ~(u32) 0,
- .rt6i_ref = ATOMIC_INIT(1),
+ .rt6i_flags     = (RTF_REJECT | RTF_NONEXTHOP),
+ .rt6i_metric    = ~(u32) 0,
+ .rt6i_ref       = ATOMIC_INIT(1),
 };

-struct rt6_info *ip6_prohibit_entry = &__ip6_prohibit_entry;
+struct rt6_info *ip6_prohibit_entry;

-static struct rt6_info __ip6_blk_hole_entry = {
+static struct rt6_info ip6_blk_hole_entry_template = {
  .u = {
   .dst = {
    .__refcnt = ATOMIC_INIT(1),
@@ -190,7 +188,6 @@ static struct rt6_info __ip6_blk_hole_en
    .input  = dst_discard,
    .output  = dst_discard,
    .ops  = &ip6_dst_ops,
-   .path  = (struct dst_entry*)&__ip6_blk_hole_entry,
   }
  },
  .rt6i_flags = (RTF_REJECT | RTF_NONEXTHOP),
@@ -198,7 +195,7 @@ static struct rt6_info __ip6_blk_hole_en
  .rt6i_ref = ATOMIC_INIT(1),
 };

-struct rt6_info *ip6_blk_hole_entry = &__ip6_blk_hole_entry;
+struct rt6_info *ip6_blk_hole_entry;

 #endif

@@ -2534,6 +2531,32 @@ 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
+
  ip6_dst_ops.kmem_cachep =
   kmem_cache_create("ip6_dst_cache", sizeof(struct rt6_info), 0,
       SLAB_HWCACHE_ALIGN|SLAB_PANIC, NULL);
@@ -2566,4 +2589,8 @@ void ip6_route_cleanup(void)
  rt6_ifdown(NULL);
  fib6_gc_cleanup();
  kmem_cache_destroy(ip6_dst_ops.kmem_cachep);
+
+ kfree(ip6_null_entry);
+ kfree(ip6_prohibit_entry);
+ kfree(ip6_blk_hole_entry);
 }
```

--