Subject: [patch 12/38][IPV6] ip6_fib - move the fib table to the network namespace
Posted by Daniel Lezcano on Mon, 03 Dec 2007 16:16:48 GMT
View Forum Message <> Reply to Message

Move the global definition of the fib table to the network namespace
structure and make their access 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 | 9 +++
 net/ipv6/ip6_fib.c | 110 ++++++++++++++++++++++++++++++++----------
 2 files changed, 80 insertions(+), 39 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
@@ -44,6 +44,15 @@ struct net {
  struct fib_rules_ops *fib4_rules_ops;
 #endif /* CONFIG_IP_MULTIPLE_TABLES */

+ /* ipv6 routing table */
+#if defined(CONFIG_IPV6) || defined(CONFIG_IPV6_MODULE)
+ struct hlist_head   *fib_table_hash;
+ struct fib6_table   *fib6_main_tbl;
+#ifdef CONFIG_IPV6_MULTIPLE_TABLES
+ struct fib6_table   *fib6_local_tbl;
+#endif /* CONFIG_IPV6_MULTIPLE_TABLES */
+#endif /* CONFIG_IPV6 */
+
  struct sock   *rtnl;   /* rtnetlink socket */

  /* List of all packet sockets. */
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
@@ -166,14 +166,11 @@ static __inline__ void rt6_release(struc
   dst_free(&rt->u.dst);
 }

-static struct fib6_table *fib6_main_tbl;
-
 #ifdef CONFIG_IPV6_MULTIPLE_TABLES
 #define FIB_TABLE_HASHSZ 256
 #else
 #define FIB_TABLE_HASHSZ 1

```
 #endif
-static struct hlist_head *fib_table_hash;

 static void fib6_link_table(struct fib6_table *tb)
 {
@@ -191,13 +188,11 @@ static void fib6_link_table(struct fib6_
   * No protection necessary, this is the only list mutatation
   * operation, tables never disappear once they exist.
   */
- hlist_add_head_rcu(&tb->tb6_hlist, &fib_table_hash[h]);
+ hlist_add_head_rcu(&tb->tb6_hlist, &init_net.fib_table_hash[h]);
 }

 #ifdef CONFIG_IPV6_MULTIPLE_TABLES

-static struct fib6_table *fib6_local_tbl;
-
 static struct fib6_table *fib6_alloc_table(u32 id)
 {
  struct fib6_table *table;
@@ -232,6 +227,7 @@ struct fib6_table *fib6_new_table(u32 id
 struct fib6_table *fib6_get_table(u32 id)
 {
  struct fib6_table *tb;
+ struct hlist_head *head;
  struct hlist_node *node;
  unsigned int h;

@@ -239,7 +235,8 @@ struct fib6_table *fib6_get_table(u32 id
   id = RT6_TABLE_MAIN;
  h = id & (FIB_TABLE_HASHSZ - 1);
  rcu_read_lock();
- hlist_for_each_entry_rcu(tb, node, &fib_table_hash[h], tb6_hlist) {
+ head = &init_net.fib_table_hash[h];
+ hlist_for_each_entry_rcu(tb, node, head, tb6_hlist) {
   if (tb->tb6_id == id) {
    rcu_read_unlock();
    return tb;
@@ -252,8 +249,8 @@ struct fib6_table *fib6_get_table(u32 id

 static void __init fib6_tables_init(void)
 {
- fib6_link_table(fib6_main_tbl);
- fib6_link_table(fib6_local_tbl);
+ fib6_link_table(init_net.fib6_main_tbl);
+ fib6_link_table(init_net.fib6_local_tbl);
 }
```

```
 #else
@@ -265,18 +262,18 @@ struct fib6_table *fib6_new_table(u32 id

 struct fib6_table *fib6_get_table(u32 id)
 {
- return fib6_main_tbl;
+ return init_net.fib6_main_tbl;
 }

 struct dst_entry *fib6_rule_lookup(struct flowi *fl, int flags,
        pol_lookup_t lookup)
 {
- return (struct dst_entry *) lookup(fib6_main_tbl, fl, flags);
+ return (struct dst_entry *) lookup(init_net.fib6_main_tbl, fl, flags);
 }

 static void __init fib6_tables_init(void)
 {
- fib6_link_table(fib6_main_tbl);
+ fib6_link_table(init_net.fib6_main_tbl);
 }

 #endif
@@ -357,6 +354,7 @@ static int inet6_dump_fib(struct sk_buff
  struct fib6_walker_t *w;
  struct fib6_table *tb;
  struct hlist_node *node;
+ struct hlist_head *head;
  int res = 0;

  if (net != &init_net)
@@ -390,7 +388,8 @@ static int inet6_dump_fib(struct sk_buff

  for (h = s_h; h < FIB_TABLE_HASHSZ; h++, s_e = 0) {
   e = 0;
-  hlist_for_each_entry(tb, node, &fib_table_hash[h], tb6_hlist) {
+  head = &init_net.fib_table_hash[h];
+  hlist_for_each_entry(tb, node, head, tb6_hlist) {
    if (e < s_e)
     goto next;
    res = fib6_dump_table(tb, skb, cb);
@@ -1363,12 +1362,13 @@ void fib6_clean_all(int (*func)(struct r
 {
  struct fib6_table *table;
  struct hlist_node *node;
+ struct hlist_head *head;
  unsigned int h;
```

```
  rcu_read_lock();
  for (h = 0; h < FIB_TABLE_HASHSZ; h++) {
-  hlist_for_each_entry_rcu(table, node, &fib_table_hash[h],
-      tb6_hlist) {
+  head = &init_net.fib_table_hash[h];
+  hlist_for_each_entry_rcu(table, node, head, tb6_hlist) {
    write_lock_bh(&table->tb6_lock);
    fib6_clean_tree(&table->tb6_root, func, prune, arg);
    write_unlock_bh(&table->tb6_lock);
@@ -1467,42 +1467,74 @@ void fib6_run_gc(unsigned long dummy)
  spin_unlock_bh(&fib6_gc_lock);
 }

-void __init fib6_init(void)
+static int fib6_net_init(struct net *net)
 {
- fib6_node_kmem = kmem_cache_create("fib6_nodes",
-        sizeof(struct fib6_node),
-        0, SLAB_HWCACHE_ALIGN|SLAB_PANIC,
-        NULL);
+ int ret;

- fib_table_hash = kzalloc(sizeof(*fib_table_hash)*FIB_TABLE_HASHSZ, GFP_KERNEL);
- if (!fib_table_hash)
-  panic("IPV6: Failed to allocate fib_table_hash.\n");
-
- fib6_main_tbl = kzalloc(sizeof(*fib6_main_tbl), GFP_KERNEL);
- if (!fib6_main_tbl)
-  panic("IPV6: Failed to allocate fib6_main_tbl.\n");
-
- fib6_main_tbl->tb6_id = RT6_TABLE_MAIN;
- fib6_main_tbl->tb6_root.leaf = &ip6_null_entry;
- fib6_main_tbl->tb6_root.fn_flags = RTN_ROOT | RTN_TL_ROOT | RTN_RTINFO;
+ if (net != &init_net)
+  return -EPERM;
+
+ ret = -ENOMEM;
+ net->fib_table_hash = kzalloc(sizeof(*net->fib_table_hash)*FIB_TABLE_HASHSZ,
+         GFP_KERNEL);
+ if (!net->fib_table_hash)
+  goto out;
+
+ net->fib6_main_tbl = kzalloc(sizeof(*net->fib6_main_tbl), GFP_KERNEL);
+ if (!net->fib6_main_tbl)
+  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.fn_flags = RTN_ROOT | RTN_TL_ROOT | RTN_RTINFO;

 #ifdef CONFIG_IPV6_MULTIPLE_TABLES
- fib6_local_tbl = kzalloc(sizeof(*fib6_local_tbl), GFP_KERNEL);
- if (!fib6_local_tbl)
-  panic("IPV6: Failed to allocate fib6_local_tbl.\n");
-
- fib6_local_tbl->tb6_id = RT6_TABLE_LOCAL;
- fib6_local_tbl->tb6_root.leaf = &ip6_null_entry;
- fib6_local_tbl->tb6_root.fn_flags = RTN_ROOT | RTN_TL_ROOT | RTN_RTINFO;
+ net->fib6_local_tbl = kzalloc(sizeof(*net->fib6_local_tbl), GFP_KERNEL);
+ if (!net->fib6_local_tbl) {
+  kfree(net->fib6_main_tbl);
+  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.fn_flags = RTN_ROOT | RTN_TL_ROOT | RTN_RTINFO;
 #endif

  fib6_tables_init();

- __rtnl_register(PF_INET6, RTM_GETROUTE, NULL, inet6_dump_fib);
+out_fib6_main_tbl:
+ kfree(net->fib_table_hash);
+out:
+ return ret;
+ }
+
+static void fib6_net_exit(struct net *net)
+{
+#ifdef CONFIG_IPV6_MULTIPLE_TABLES
+ kfree(net->fib6_local_tbl);
+#endif
+ kfree(net->fib6_main_tbl);
+ kfree(net->fib_table_hash);
+}
+
+static struct pernet_operations fib6_net_ops = {
+ .init = fib6_net_init,
+ .exit = fib6_net_exit,
+};
+
+void __init fib6_init(void)
+{
+ fib6_node_kmem = kmem_cache_create("fib6_nodes",
+        sizeof(struct fib6_node),
+        0, SLAB_HWCACHE_ALIGN|SLAB_PANIC,
```

```
+       NULL);
+
+ register_pernet_subsys(&fib6_net_ops);
+       __rtnl_register(PF_INET6, RTM_GETROUTE, NULL, inet6_dump_fib);
 }

 void fib6_gc_cleanup(void)
 {
  del_timer(&ip6_fib_timer);
+ unregister_pernet_subsys(&fib6_net_ops);
  kmem_cache_destroy(fib6_node_kmem);
 }

--
```

Subject: Re: [patch 12/38][IPV6] ip6_fib - move the fib table to the network namespace
Posted by Brian Haley on Tue, 04 Dec 2007 19:28:07 GMT
View Forum Message <> Reply to Message

Hi Daniel,

Daniel Lezcano wrote:
> Move the global definition of the fib table to the network namespace
> structure and make their access to the initial network namespace.
....
> -void __init fib6_init(void)
> +static int fib6_net_init(struct net *net)
> {
> - fib6_node_kmem = kmem_cache_create("fib6_nodes",
> -   sizeof(struct fib6_node),
> -   0, SLAB_HWCACHE_ALIGN|SLAB_PANIC,
> -   NULL);
> + int ret;
>
> - fib_table_hash = kzalloc(sizeof(*fib_table_hash)*FIB_TABLE_HASHSZ, GFP_KERNEL);
> - if (!fib_table_hash)
> - panic("IPV6: Failed to allocate fib_table_hash.\n");
> -
> - fib6_main_tbl = kzalloc(sizeof(*fib6_main_tbl), GFP_KERNEL);
> - if (!fib6_main_tbl)
> - panic("IPV6: Failed to allocate fib6_main_tbl.\n");
> -

> - fib6_main_tbl->tb6_id = RT6_TABLE_MAIN;
> - fib6_main_tbl->tb6_root.leaf = &ip6_null_entry;
> - fib6_main_tbl->tb6_root.fn_flags = RTN_ROOT | RTN_TL_ROOT | RTN_RTINFO;
> + if (net != &init_net)
> + return -EPERM;
> +
> + ret = -ENOMEM;
> + net->fib_table_hash = kzalloc(sizeof(*net->fib_table_hash)*FIB_TABLE_HASHSZ,
> + GFP_KERNEL);
> + if (!net->fib_table_hash)
> + goto out;

So originally, the fib_table_hash was global with no allocation
necessary. Then in patch 11 you changed it to be dynamic and panic() on
failure. Now it just gracefully returns. Do you really want to do that
for the init_net namespace? Won't the next routine that tries to access
this NULL pointer oops? Same for fib6_main_table, fib6_local_table,
fib6_stats, etc. Or did I miss where the callers error-out on an ENOMEM?

-Brian

Subject: Re: [patch 12/38][IPV6] ip6_fib - move the fib table to the network namespace
Posted by Daniel Lezcano on Wed, 05 Dec 2007 10:05:00 GMT
View Forum Message <> Reply to Message

Brian Haley wrote:
> Hi Daniel,
>
> Daniel Lezcano wrote:
>> Move the global definition of the fib table to the network namespace
>> structure and make their access to the initial network namespace.
> ....
>> -void __init fib6_init(void)
>> +static int fib6_net_init(struct net *net)
>> {
>> -   fib6_node_kmem = kmem_cache_create("fib6_nodes",
>> -   sizeof(struct fib6_node),
>> -   0, SLAB_HWCACHE_ALIGN|SLAB_PANIC,
>> -   NULL);
>> +   int ret;
>>
>> -   fib_table_hash =

```
>> kzalloc(sizeof(*fib_table_hash)*FIB_TABLE_HASHSZ, GFP_KERNEL);
>> -    if (!fib_table_hash)
>> -        panic("IPV6: Failed to allocate fib_table_hash.\n");
>> -
>> -    fib6_main_tbl = kzalloc(sizeof(*fib6_main_tbl), GFP_KERNEL);
>> -    if (!fib6_main_tbl)
>> -        panic("IPV6: Failed to allocate fib6_main_tbl.\n");
>> -
>> -    fib6_main_tbl->tb6_id = RT6_TABLE_MAIN;
>> -    fib6_main_tbl->tb6_root.leaf = &ip6_null_entry;
>> -    fib6_main_tbl->tb6_root.fn_flags = RTN_ROOT | RTN_TL_ROOT |
>> RTN_RTINFO;
>> +    if (net != &init_net)
>> +        return -EPERM;
>> +
>> +    ret = -ENOMEM;
>> +    net->fib_table_hash =
>> kzalloc(sizeof(*net->fib_table_hash)*FIB_TABLE_HASHSZ,
>> +                    GFP_KERNEL);
>> +    if (!net->fib_table_hash)
>> +        goto out;
>
> So originally, the fib_table_hash was global with no allocation
> necessary.  Then in patch 11 you changed it to be dynamic and panic() on
> failure.  Now it just gracefully returns.  Do you really want to do that
> for the init_net namespace?  Won't the next routine that tries to access
> this NULL pointer oops?  Same for fib6_main_table, fib6_local_table,
> fib6_stats, etc.  Or did I miss where the callers error-out on an ENOMEM?
```

Yes, you are right, this is not consistent and we should not panic
neither ignore the error. I should have modified the ip6_route_init and
returned an error in case of register_pernet_subsys failure. So the ipv6
initialization can fails safely.

I will post some modifications around that for net-2.6.25 for netdev@.

Thanks for catching that.

 -- Daniel