Subject: Re: [patch 1/2][NETNS][RFD] store the network namespace pointer in the dst_entry structure
Posted by Daniel Lezcano on Tue, 11 Dec 2007 16:14:47 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:

```
> Daniel Lezcano <dlezcano@fr.ibm.com> writes:
>
>> Store the network namespace pointer in the dst_entry structure when it is
>> allocated.
>> The different protocols redefine the route object as a derivate object from
>> dst_entry. So using the dst_entry to store the network namespace pointer will
>> allow to take into account the ipv4, ipv6, dccp protocols in one shot through
>> the different route objects, rtable, rt6_info, ...
>>
>> ---
>>  include/net/dst.h       |    3 ++-
>>  net/core/dst.c          |    3 ++-
>>  net/decnet/dn_route.c   |    4 ++--
>>  net/ipv4/route.c        |   14 +++++++-------
>>  net/ipv6/route.c        |   18 ++++++++++--------
>>  net/xfrm/xfrm_policy.c  |    2 +-
>>  6 files changed, 24 insertions(+), 20 deletions(-)
>>
>> Index: linux-2.6-netns/include/net/dst.h
>> ===================================================================
>> --- linux-2.6-netns.orig/include/net/dst.h
>> +++ linux-2.6-netns/include/net/dst.h
>> @@ -81,6 +81,7 @@ struct dst_entry
>>    struct dn_route  *dn_next;
>>   };
>>   char   info[0];
>> + struct net  *net;
>
> Unless I'm missing something you just place that net pointer in
> the middle of a variable length array.  Weird I don't see us
> using that array.
```

yep, right, thanks.

```
> Could you please place the struct net *net pointer up by the
> network device pointer.
>>  };
>
> I know we need a net pointer in struct rt_table, because it
> is a hash table that we can't dynamically allocate so we need
> to place a network namespace pointer as part of the hash key.
>
```

> For the ipv6 fib tables I don't recall needing a net pointer
> as we didn't have a hash table and could instead have separate
> roots for different namespaces.

Yes don't need for the hash table but we used it to pass the network namespace parameter to the underlying function which need the net parameter.

We are facing two problems when removing the fl_net field from flowi:

 * The first one is the fl_net is used as a key. This problem can be handled simply in moving the netns to the rtable.

 * The second one is the usage made by the fl_net to pass through the different function calls the network namespace pointer without changing all functions signature. This problem can be solved if we put the netns pointer in the dst_entry structure, so when we are in ipv4, we use container_of on rtable and when we are in ipv6, we use the container_of on rt6_info. So everywhere with the flowi, we can retrieve the netns.

Here is a example for ipv4:

```
static inline int fib_lookup(const struct flowi *flp, struct fib_result
*res)
{
 struct rtable *rt = container_of(flp, struct rtable, fl);
 struct net *net = rt->u.dst.net;
 struct fib_table *local_table = net->ip_fib_local_table;
 struct fib_table *main_table = net->ip_fib_main_table;
 if (local_table->tb_lookup(local_table, flp, res) &&
     main_table->tb_lookup(main_table, flp, res))
  return -ENETUNREACH;
 return 0;
}
```

Other one for ipv6:

```
static struct rt6_info *ip6_pol_route_lookup(struct fib6_table *table,
          struct flowi *fl, int flags)
{
 struct rt6_info *rt  = container_of(flp, struct rt6_info, fl);
 struct net *net = rt.u.dst.net;
 struct fib6_node *fn;
 struct rt6_info *rt;

 read_lock_bh(&table->tb6_lock);
 fn = fib6_lookup(&table->tb6_root, &fl->fl6_dst, &fl->fl6_src);
restart:
 rt = fn->leaf;
 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);
 return rt;

}
```

> I find this slightly odd as I didn't wind up needing to add
> a struct net pointer in struct dst in my proof of concept tree
> and struct dst doesn't have a struct flowi so that would not
> have prevented it.

The idea is to put the net in the dst_entry because it is accessible
from rtable or rt6_info and these ones contain a flowi field.