
Subject: Re: [PATCH] [NETNS45] network namespace locking rules
Posted by [dlunev](#) on Fri, 28 Sep 2007 16:33:41 GMT
[View Forum Message](#) <> [Reply to Message](#)

Daniel, I've sent you last letter from Eric regarding change below. By the way, you've been in CC: for that thread :)

I'll rebase this to #49 on monday.

Regards,
Den

Daniel Lezcano wrote:

> Denis V. Lunev wrote:

>> Current locking for network namespace list/initialization is broken.

>> for_each_net is called under single rtnl_lock in

>> register_netdevice_notifier.

>>

>> Locking:

>> net_mutex -> rtnl_lock() -> dev_base_lock

>> Reasoning:

>> - net_mutex holds serialization of the addition/removal of

>> subsystems/modules and the creation/destruction of network

>> namespaces as a whole

>> - loopback device is one of such subsystems and it takes

>> rtnl_lock inside

>> - per/namespace RTNL netlink socket requires an iteration over

>> namespace list inside rtnl_unlock, which is called inside net_mutex

>> Resume:

>> net_namespace_list is guarded by both rtnl_lock & net_mutex and

>> can be safely iterated under any of them

>>

>> Signed-off-by: Denis V. Lunev <den@openvz.org>

>>

>> -----

>

> [cut]

>>

>> -void net_lock(void)

>> -{

>> - mutex_lock(&net_list_mutex);

>> -}

>> -

>> -void net_unlock(void)

>> -{

>> - mutex_unlock(&net_list_mutex);

>> -}

>> -

```

>
> net_list_mutex is no more needed, right ? The removing of its
> declaration is missing (already done in #netns49)
>
>> static struct net *net_alloc(void)
>> {
>>     return kmem_cache_alloc(net_cachep, GFP_KERNEL);
>> @@ -71,9 +62,9 @@ static void cleanup_net(struct work_struct *work)
>>     mutex_lock(&net_mutex);
>>
>>     /* Don't let anyone else find us. */
>> - net_lock();
>> + rtnl_lock();
>>     list_del(&net->list);
>> - net_unlock();
>> + rtnl_unlock();
>>
>>     /* Run all of the network namespace exit methods */
>>     pernet_count = 0;
>> @@ -193,9 +184,9 @@ struct net *copy_net_ns(unsigned long flags,
>> struct net *old_net)
>>     if (err)
>>         goto out_unlock;
>>
>> - net_lock();
>> + rtnl_lock();
>>     list_add_tail(&new_net->list, &net_namespace_list);
>> - net_unlock();
>> + rtnl_unlock();
>>
>>
>> out_unlock:
>> @@ -220,14 +211,13 @@ static int __init net_ns_init(void)
>>     mutex_lock(&net_mutex);
>>     err = setup_net(&init_net);
>>
>> - net_lock();
>> + rtnl_lock();
>>     list_add_tail(&init_net.list, &net_namespace_list);
>> - net_unlock();
>> + rtnl_unlock();
>>
>>     mutex_unlock(&net_mutex);
>>     if (err)
>>         panic("Could not setup the initial network namespace");
>> -
>>     return 0;
>> }

```

```

>>
>
> Eric did already these changes in #netns49.
>
>> diff --git a/net/core/rtnetlink.c b/net/core/rtnetlink.c
>> index 82ebc23..e610313 100644
>> --- a/net/core/rtnetlink.c
>> +++ b/net/core/rtnetlink.c
>> @@ -73,16 +73,24 @@ void __rtnl_unlock(void)
>> void rtnl_unlock(void)
>> {
>>     struct net *net;
>> -    mutex_unlock(&rtnl_mutex);
>> -
>> -    net_lock();
>> +
>> +retry:
>>     for_each_net(net) {
>>         struct sock *rtnl = net->rtnl;
>> +
>> +        if (rtnl == NULL || rtnl->sk_receive_queue.qlen == 0)
>> +            continue;
>> +
>> +        get_net(net);
>> +        mutex_unlock(&rtnl_mutex);
>>         if (rtnl && rtnl->sk_receive_queue.qlen)
>             ~~~~~~
>
> this statement is always true with the previous one.
>
>>         rtnl->sk_data_ready(rtnl, 0);
>> -    }
>> -    net_unlock();
>> +    mutex_lock(&rtnl_mutex);
>> +    put_net(net);
>>
>> +    goto retry;
>
> Why do you need to return to the beginning of the list ?
>
>> +    }
>> +    mutex_unlock(&rtnl_mutex);
>>     netdev_run_todo();
>> }
>
> Denis, can you explain why this part must be modified ?
>
>

```

>

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>
