
Subject: [PATCH] [NETNS45] network namespace locking rules

Posted by [den](#) on Fri, 28 Sep 2007 14:36:10 GMT

[View Forum Message](#) <> [Reply to Message](#)

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>

diff --git a/include/net/net_namespace.h b/include/net/net_namespace.h

index b8186ea..2845992 100644

--- a/include/net/net_namespace.h

+++ b/include/net/net_namespace.h

```
@@ -174,8 +174,21 @@ static inline void release_net(struct net *net)
    atomic_dec(&net->use_count);
}
```

```
-extern void net_lock(void);
```

```
-extern void net_unlock(void);
```

```
+/*
```

```
+ * 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
+ */

#define for_each_net(VAR) \
    list_for_each_entry(VAR, &net_namespace_list, list)
diff --git a/net/core/net_namespace.c b/net/core/net_namespace.c
index 026e39a..07682a2 100644
--- a/net/core/net_namespace.c
+++ b/net/core/net_namespace.c
@@ -10,6 +10,7 @@

/*
 * Our network namespace constructor/destructor lists
+ * Locking rules are described in details in include/net/net_namespace.h
 */

static LIST_HEAD(pernet_list);
@@ -24,16 +25,6 @@ static struct kmem_cache *net_cachep;
struct net init_net;
EXPORT_SYMBOL_GPL(init_net);

-void net_lock(void)
-{
- mutex_lock(&net_list_mutex);
-}
-
-void net_unlock(void)
-{
- mutex_unlock(&net_list_mutex);
-}
-
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;
}

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)
    rtnl->sk_data_ready(rtnl, 0);
- }
- net_unlock();
+ mutex_lock(&rtnl_mutex);
+ put_net(net);

+ goto retry;
+ }
+ mutex_unlock(&rtnl_mutex);
  netdev_run_todo();
}

```

Subject: Re: [PATCH] [NETNS45] network namespace locking rules

Posted by [Daniel Lezcano](#) on Fri, 28 Sep 2007 15:10:36 GMT

[View Forum Message](#) <> [Reply to Message](#)

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>

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>

Subject: Re: [PATCH] [NETNS45] network namespace locking rules
Posted by [ebiederm](#) on Fri, 28 Sep 2007 16:54:41 GMT
[View Forum Message](#) <> [Reply to Message](#)

"Denis V. Lunev" <den@openvz.org> writes:

> Current locking for network namespace list/initialization is broken.
> for_each_net is called under single rtnl_lock in
> register_netdevice_notifier.

As of 984e617a3e1974022b8f671427a76ffbe886f75b this issue has
been addressed in net-2.6.24

The only remaining part to address is rtnl_unlock().

My current hypothesis is that rtnl_unlock() only needs to process
the packets that are queued while we had the rtnl_lock held,
to maintain the current semantics.

So the retry may not be necessary as it may be possible to prove
that the only extra packets that could come in come from another
thread taking the rtnl_lock and they will those packets in their
rtnl_unlock() if we don't.

I will review this later today, and add the retry if necessary.

One way or another I think we agree on how to get the locking correct.
The other details are a bit tricky but look usable.

Eric

Subject: Re: [PATCH] [NETNS45] network namespace locking rules

Posted by [dlunev](#) on Fri, 28 Sep 2007 17:02:13 GMT

[View Forum Message](#) <> [Reply to Message](#)

Eric W. Biederman wrote:

> "Denis V. Lunev" <den@openvz.org> writes:

>

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

>> for_each_net is called under single rtnl_lock in

>> register_netdevice_notifier.

>

> As of 984e617a3e1974022b8f671427a76ffbe886f75b this issue has

> been addressed in net-2.6.24

>

> The only remaining part to address is rtnl_unlock().

>

> My current hypothesis is that rtnl_unlock() only needs to process

> the packets that are queued while we had the rtnl_lock held,

> to maintain the current semantics.

>

> So the retry may not be necessary as it may be possible to prove

> that the only extra packets that could come in come from another

> thread taking the rtnl_lock and they will those packets in their

> rtnl_unlock() if we don't.

>

> I will review this later today, and add the retry if necessary.

>

> One way or another I think we agree on how to get the locking correct.

> The other details are a bit tricky but look usable.

>

> Eric

>

Unfortunately, the answer is 'no'. data_ready callback takes rtnl inside, so we can have new packets from other namespaces during the processing.

No way, but restart :(

Regards,
Den
