Subject: Re: [RFC PATCH 0/31] An introduction and A path for merging network namespace work
Posted by ebiederm on Wed, 28 Feb 2007 19:45:10 GMT
View Forum Message <> Reply to Message

Daniel Lezcano <dlezcano@fr.ibm.com> writes:

> Hi Eric,
>
> Do you plan to propose to merge into mainline your patchset ?

I'm hung up at the moment in the sysfs support. Network device renaming is broken in 2.6.21-rc2 at the moment.

Then I would like to see the best of etun/veth merged.

After that yes I would like to propose getting a network namespace implementation into mainline. Which would like be based on the patchset I posted.

> Shouldn't we ask netdev guys what they think about the explicit network
> namespace parameter into function you did versus the implicit network context
> using the push_net_ns/pop_net_ns function ?

It is an important question.

My impression is that in the larger context it seems to be a minor detail.

>From what I have seen of Dmitry's patches and from what I have seen of my own. When not talking functions parameters we make roughly the same set of code changes. For example in my git tree without referencing Dmitry's work, I made roughly the same set of changes to the fib code as he did.

I currently prefer my register_pernet_subsys infrastructure as it is much easier to deal with then gradually accumulating the code into the namespace initialization. There is nothing to clever about it, so we should be fine.

I think my current per_net() function is questionable (it borders on being too clever) but it is very straight forward to use. In fact I am probably over using it a bit and making my network namespace data structures a little too big. If you look at the slab or the initialization messages you will see I have exceeded page size. Oops.

The big advantage of my pernet work (as opposed to other techniques) is that it makes compiling out any the effect of my code possible,

and it allows for pernet variables with file scope. If continuing to support compiling out the pernet code isn't a requirement we could get less clever solution, and just pass a pointer around.

If we do start passing a pointer around there becomes the question of how do we support modules. Which is particularly important in the IPv6 case.

So long as my per_net() function doesn't cause problems I suspect it is easier to work with than to work without.

As long as we are supporting compiling things out I think using net_t instead of a raw pointer makes a lot of sense. I really like the fact that using an empty type when we compile things out so gcc can just optimize everything away, instead of having to ifdef everything.

The big practical difference between the approaches comes down to push_net_ns/pop_net_ns, or doing something else to get the argument where it belongs. The fact that push_net_ns/pop_net_ns cannot be universally used in the network stack is a pain. It means that a function that can used on both the receive and the transmit path has to have a network namespace argument.

The verification that we are doing the right thing with push_net_ns/pop_net_ns is also harder as we have to check that we have the proper value for the entire call chain instead of a check that is simply local. For doing the conversion it is not a big pain, as we need to audit the call chain anyway. For dealing with future changes it could be a problem if to verify every little patch we had to check the entire call chain.

The biggest argument against explicit parameters that I could see in earlier conversations was that you could not compile them out. Now that I have gotten clever and can compile my explicit parameters out that argument goes away.

The big downside of the explicit parameters is that in some cases you get a lot of noise patches just to get the value where you need it. So it more difficult to merge and a little more difficult to maintain out of the tree.

However for long term maintenance there is a big advantage of explicit parameters as you only need to test a patch to see if it is locally correct.

So unless explicit parameters hurt performance my impression is that
they are the better solution.

Dmitry? Daniel? What do you think.

Eric

_____

Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers