Subject: Re: [patch 1/4] Network namespaces: cleanup of dev_base list use
Posted by ebiederm on Mon, 26 Jun 2006 21:02:49 GMT
View Forum Message <> Reply to Message

Andrey Savochkin <saw@swsoft.com> writes:

> Eric,
>
> On Mon, Jun 26, 2006 at 10:26:23AM -0600, Eric W. Biederman wrote:
>>
>>
> [snip]
>> It is a big enough problem that I don't think we want to gate on
>> that development but we need to be ready to take advantage of it when
>> it happens.
>
> Well, ok, implicit namespace reference will take advantage of it
> if it happens.

And if fact in that case we don't have to do anything special because
the process pointer will always be correct.

>> >> However short of always having code always execute in the proper
>> >> context I'm not comfortable with implicit parameters to functions.
>> >> Not that this the contents of this patch should address this but the
>> >> later patches should.
>> >
>> > We just have too many layers in networking code, and FIB/routing
>> > illustrates it well.
>>
>> I don't follow this comment. How does a lot of layers affect
>> the choice of implicit or explicit parameters? If you are maintaining
>> a patch outside the kernel I could see how there could be a win for
>> touching the least amount of code possible but for merged code that
>> you only have to go through once I don't see how the number of layers
>> affects things.
>
> I agree that implicit vs explicit parameters is a topic for discussion.
> From what you see from my patch, I vote for implicit ones in this case :)

Yes. I tend to be against implicit namespaces references mostly because
the explicit ones tend to make the code clearer.

> I was talking about layers because they imply changing more code,
> and usually imply adding more parameters to functions and passing these
> additional parameters to next layers.
> In "routing" code it goes from routing entry points, to routing cache, to
> general FIB functions, to table-specific code (FIB hash).

Yes. Although as I recall you don't have to pass anything down very far.
Because most functions once you have done the table lookup operate
on just a subset of the table, when they are getting the real work done.

> These additional parameters bloat the code to some extent.
> Sometimes it's possible to save here and there by fetching the parameter
> (namespace pointer) indirectly from structures you already have at hand,
> but it can't be done universally.
>
> One of the properties of implicit argument which I especially like
> is that both input and output paths are absolutely symmetric in how
> the namespace pointer is extracted.

There is an element of that. In the output path for the most part everything
works implicitly because you are in the proper context.

I need to dig out my code and start comparing to what you have done.

Eric
