Subject: Re: [patch 1/4] Network namespaces: cleanup of dev_base list use
Posted by Andrey Savochkin on Mon, 26 Jun 2006 20:14:37 GMT
View Forum Message <> Reply to Message

Eric,

On Mon, Jun 26, 2006 at 10:26:23AM -0600, Eric W. Biederman wrote:
> Andrey Savochkin <saw@swsoft.com> writes:
>
> > On Mon, Jun 26, 2006 at 09:13:52AM -0600, Eric W. Biederman wrote:
> >>
> >> There is another topic for discussion in this patch as well.
> >> How much of the context should be implicit and how much
> >> should be explicit.
> >>
> >> If the changes from netchannels had already been implemented, and all of
> >> the network processing was happening in a process context then I would
> >> trivially agree that implicit would be the way to go.
> >
>
[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.

>
> >> 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 :)

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).

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.

>
> As I recall for most of the FIB/routing code once you have removed
> the global variable accesses and introduce namespace checks in
> the hash table (because allocating hash tables at runtime isn't sane)
> the rest of the code was agnostic about what was going on. So I think
> you have touched everything that needs touching. So I don't see
> a code size or complexity argument there.

Andrey