Subject: Re: [patch 2/6] [Network namespace] Network device sharing by view
Posted by ebiederm on Wed, 28 Jun 2006 04:07:29 GMT
View Forum Message <> Reply to Message

Herbert Poetzl <herbert@13thfloor.at> writes:

> On Tue, Jun 27, 2006 at 10:29:39AM -0600, Eric W. Biederman wrote:
>> Herbert Poetzl <herbert@13thfloor.at> writes:
>
>> I watched the linux-vserver irc channel for a while and almost
>> every network problem was caused by the change in semantics
>> vserver provides.
>
> the problem here is not the change in semantics compared
> to a real linux system (as there basically is none) but
> compared to _other_ technologies like UML or QEMU, which
> add the need for bridging and additional interfaces, while
> Linux-VServer only focuses on the IP layer ...

Not being able to bind to INADDR_ANY is a huge semantic change.
Unless things have changed recently you get that change when
you have two IP addresses in Linux-Vserver.

Talking to the outsider world through the loop back interface
is a noticeable semantics change.

Having to be careful of who uses INADDR_ANY on the host
when you have guests is essentially a semantics change.

Being able to talk to the outside world with a server
bound only to the loopback IP is a weird semantic
change.

And I suspect I missed something, it is weird peculiar and
I don't care to remember all of the exceptions.

Have a few more network interfaces for a layer 2 solution
is fundamental. Believing without proof and after arguments
to the contrary that you have not contradicted that a layer 2
solution is inherently slower is non-productive. Arguing
that a layer 2 only solution most prove itself on guest to guest
communication is also non-productive.

So just to sink one additional nail in the coffin of the silly
guest to guest communication issue. For any two guests where
fast communication between them is really important I can run
an additional interface pair that requires no routing or bridging.
Given that the implementation of the tunnel device is essentially

the same as the loopback interface and that I make only one
trip through the network stack there will be no performance overhead.
Similarly for any critical guest communication to the outside world
I can give the guest a real network adapter.

That said I don't think those things will be necessary and that if
they are it is an optimization opportunity to make various bits
of the network stack faster.

Bridging or routing between guests is an exercise in simplicity
and control not a requirement.

>> In this case when you allow a guest more than one IP your hack
>> while easy to maintain becomes much more complex.
>
> why? a set of IPs is quite similar to a single IP (which
> is actually a subset), so no real change there, only
> IP_ANY means something different for a guest ...

Which simply filtering at bind time makes impossible.

With a guest with 4 IPs
10.0.0.1 192.168.0.1 172.16.0.1 127.0.0.1
How do you make INADDR_ANY work with just filtering at bind time?

The host has at least the additional IPs.
10.0.0.2 192.168.0.2 172.16.0.2 127.0.0.1

Herbert I suspect we are talking about completely different
implementations otherwise I can't possibly see how we have
such different perceptions of their capabilities.

I am talking precisely about filter IP addresses at connect
or bind time that a guest can use. Which as I recall is
what vserver implements. If you are thinking of your ngnet
implementation that would explain things.

>> Especially as you address each case people care about one at a time.
>
> hmm?

Multiple IPs, IPv6, additional protocols, firewalls. etc.

>> In one shot this goes the entire way. Given how many people miss that
>> you do the work at layer 2 than at layer 3 I would not call this the
>> straight forward approach. The straight forward implementation yes,
>> but not the straight forward approach.
>

> seems I lost you here ...

>> > for example, you won't have multiple routing tables
>> > in a kernel where this feature is disabled, no?
>> > so why should it affect a guest, or require modified
>> > apps inside a guest when we would decide to provide
>> > only a single routing table?
>> >
>> >> From my POV, fully virtualized namespaces are the future.
>> >
>> > the future is already there, it's called Xen or UML, or QEMU :)
>>
>> Yep. And now we need it to run fast.
>
> hmm, maybe you should try to optimize linux for Xen then,
> as I'm sure it will provide the optimal virtualization
> and has all the features folks are looking for (regarding
> virtualization)
>
> I thought we are trying to figure a light-weight subset
> of isolation and virtualization technologies and methods
> which make sense to have in mainline ...

And you presume doing things at layer 2 is more expensive than
layer 3.

>From what I have seen of layer 3 solutions it is a
bloody maintenance nightmare, and an inflexible mess.

>> >> It is what makes virtualization solution usable (w/o apps
>> >> modifications), provides all the features and doesn't require much
>> >> efforts from people to be used.
>> >
>> > and what if they want to use virtualization inside
>> > their guests? where do you draw the line?
>>
>> The implementation doesn't have any problems with guests inside
>> of guests.
>>
>> The only reason to restrict guests inside of guests is because
>> the we aren't certain which permissions make sense.
>
> well, we have not even touched the permission issues yet

Agreed, permissions have not discussed but the point is that is the only
reason to keep from nesting the networking stack the way I have described
it.

Eric