Subject: Network virtualization/isolation
Posted by jamal on Sun, 03 Dec 2006 14:13:18 GMT
View Forum Message <> Reply to Message

I have removed the Re: just to add some freshness to the discussion

So i read quickly the rest of the discussions. I was almost suprised to find that i agree with Eric on a lot of opinions (we also agree that vindaloo is good for you i guess);->
The two issues that stood out for me (in addition to what i already said below):

1) the solution must ease the migration of containers; i didnt see anything about migrating them to another host across a network, but i assume that this is a given.
2) the socket level bind/accept filtering with multiple IPs. From reading what Herbert has, it seems they have figured a clever way to optimize this path albeit some challenges (speacial casing for raw filters) etc.

I am wondering if one was to use the two level muxing of the socket layer, how much more performance improvement the above scheme provides for #2?
Consider the case of L2 where by the time the packet hits the socket layer on incoming, the VE is already known; in such a case, the lookup would be very cheap. The advantage being you get rid of the speacial casing altogether. I dont see any issues with binds per multiple IPs etc using such a technique.

For the case of #1 above, wouldnt it be also easier if the tables for netdevices, PIDs etc were per VE (using the 2 level mux)?

In any case, folks, i hope i am not treading on anyones toes; i know each one of you has implemented and has users and i am trying to be as neutral as i can (but clearly biased;->).

cheers,
jamal

On Sun, 2006-03-12 at 07:26 -0500, jamal wrote:
> On Wed, 2006-14-11 at 16:17 +0100, Daniel Lezcano wrote:
> > The attached document describes the network isolation at the layer 2
> > and at the layer 3 ..
>
> Daniel,

>
> I apologize for taking this long to get back to you. The document (I
> hope) made it clear to me at least the difference between the two
> approaches. So thanks for taking the time to put it together.
>
> So here are my thoughts ...
> I havent read the rest of the thread so i may be repeating some of the
> discussion; i have time today, I will try to catchup with the
> discussion.
>
> * i think the L2 approach is the more complete of the two approaches:
>
> It caters to more applications: eg i can have network elements such as
> virtual bridges and routers. It doesnt seem like i can do that with the
> L3 approach. I think this in itself is a powerful enough reason to
> disqualify the L3 approach.
>
> Leading from the above, I dont have to make _a single line of code
> change_ to any of the network element management tools inside the
> container. i.e i can just run quagga and OSPF and BGP will work as is or
> the bridge daemon and STP will work as is or tc to control "real"
> devices or ip to control "real" ip addresses. Virtual routers and
> bridges are real world applications (if you want more info ask me or ask
> google, she knows).
>
> **** This wasnt clear to me from the doc on the L3 side of things, so
> please correct me:
> because of the pid virtualization in the L2 approach(openvz?) I can run
> all applications as is. They just dont know they are running on a
> virtual environment. To use an extreme example: if i picked apache as a
> binary compiled 10 years ago, it will run on the L2 approach but not on
> the L3 approach. Is this understanding correct? I find it hard to
> believe that the L3 approach wouldnt work this way - it may be just my
> reading into the doc.
>
> So lets say the approach taken is that of L2 (I am biased towards this
> because i want to be able to do virtual bridges and routers). The
> disadvantage of the L2 approach (or is it just the openvz?) approach is:
>
> - it is clear theres a lot more code needed to allow for the two level
> multiplexing every where. i.e first you mux to select the namespace then
> you do other things like find a pid, netdevice, ip address etc. I am
> also not sure how complete that code is; you clearly get everything
> attached to netdevices for free (eg networkc scheduler block) - which is
> nice in itself; but you may have to do the muxing code for other blocks.
> If my understanding is correct everything in the net subsystem has this
> mux levels already in place (at least with openvz)? I think each
> subsystem may have its own merit discussed (eg the L3 tables with the

> recent changes from Patrick allow up to 2^32 -1 tables, so a muxing
> layer at L3 maybe unnecessary)
> ---> To me this 2 level muxing looks like a clean design in that there
> is consistency (i.e no hack thats specific to just one sub-subsystem but
> not others). With this approach one could imagine hardware support that
> does the first level of muxing (selecting a namespace for you). This is
> clearly already happening with NICs supporting several unicast MAC
> addresses.
> I think the litmus test for this approach is the answer to the question:
> If i compiled in the containers in and do not use the namespaces, how
> much more overhead is there for the host path? I would hope that it is
> as close to 0 as possible. It should certainly be 0 if i dont compile in
> containers.
>
> - The desire for many MAC addresses. I dont think this is a killer
> issue. NICs are begining to show up which capabilities for many unicast
> MACs; many current have multicast hardware tables that can be used for
> stashing unicast MAC addresses; it has also been shown you can use
> multicast MAC addresses and get away with it if there is no conflict
> (protocols such as VRRP, CARP etc do this).
>
> - Manageability from the host side. It seems to be more complex with the
> L2 than with L3. But so what? These tools are written from scratch and
> there is no "backward compatibility" baggage.
>
> Ok, I am out of coffee for the last 10 minutes;-> But above sit my views
> worth about $0.02 Canadian (which is about $0.02 US these days).
>
> I will try later to catch up with the discussion that started from
> Daniels original posting.
>
> cheers,
> jamal
