Subject: Re: [patch 2/6] [Network namespace] Network device sharing by view
Posted by ebiederm on Mon, 26 Jun 2006 22:11:02 GMT
View Forum Message <> Reply to Message

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

> On Mon, Jun 26, 2006 at 02:37:15PM -0600, Eric W. Biederman wrote:
>> Herbert Poetzl <herbert@13thfloor.at> writes:
>>
>> > On Mon, Jun 26, 2006 at 01:35:15PM -0600, Eric W. Biederman wrote:
>> >> Herbert Poetzl <herbert@13thfloor.at> writes:
>> >
>> > yes, but you will not be able to apply policy on
>> > the parent, restricting the child networking in a
>> > proper way without jumping through hoops ...
>>
>> ? I don't understand where you are coming from.
>> There is no restriction on where you can apply policy.
>
> in a fully hierarchical system (not that I really think
> this is required here) you would be able to 'delegate'
> certain addresses (ranges) to a namespace (the child)
> below the current one (the parent) with the ability to
> limit/control the input/output (which is required for
> security)

All of that is possible with the current design.
It is merely a matter of using the features the kernel
currently has.

The trick is know that a child namespace only has a loopback
by default, and has to have a network interface added from
the parent to be able to talk to anything.

>> >> I really do not believe we have a hotpath issue, if this
>> >> is implemented properly. Benchmarks of course need to be taken,
>> >> to prove this.
>> >
>> > I'm fine with proper testing and good numbers here
>> > but until then, I can only consider it a prototype
>>
>> We are taking the first steps to get this all sorted out.
>> I think what we have is more than a prototype but less then
>> the final implementation. Call it the very first draft version.
>
> what we are desperately missing here is a proper way
> to testing this, maybe the network folks can come up

> with some test equipment/ideas here ...
>

>> That assumes on the wire stuff is noticeably slower.
>> You can achieve over 1GB/s on some networks.
>
> well, have you ever tried how much you can achieve
> over loopback :)

Not recently.

> well, local is fine, but you cannot utilize that
> on-wire which basically means that you would have
> either to 'map' the MAC on transmission (to the
> real one) which would basically make the approach
> useless, or to use addresses which are fine within
> a certain range of routers ...

I believe on the wire is fine as well. Certainly I had
no problems when I tested it. I do agree that it increases
the chance for a mac address collision so should be handled
carefully. But we are talking a number with almost as many random
bits as a UUID.

As I recall the rule from the birthday paradox is something like: if
you have N possibilities you get a 50% chance of collision when
you have sqrt(N) items present. sqrt(2**40) = 2**20 ~= 1 Million.

So as long as you have a good random generator the odds of a collision
are quite small until you have used a million local mac addresses.

Now while I can see some small chance of that happening on a very
crowded local area network, using lots of logical servers the
kernel arp cache and switch mac cache limits would start causing
real problems long before you got that far. So you would need to
start routing at which point the practical problem goes away.

>
> well, for loopback that would mean half the bandwidth
> and twice the latency, no?

Not at all. The usual bottle neck is copying the data. The data
only gets put in a skb once and then pointers to the skb are
passed around. So you get copied in once and copied out once.
We are certainly not going to add an extra copy to it.

For practical purposes the fast path through the network stack
is a series of hash table looks.

That added to the fact that we don't make a full trip through
the network stack on both sides (unless someone is running tcp dump)
for example.

>> If it does we have a lot more to optimize in the network stack than
>> just this code.
>
> why? duplicate stack traversal takes roughly twice
> the time, or am I wrong here? if so, please enlighten
> me ...

The network stack is how we decide what goes where. Sending
and receiving packets should always have hardware as the bottleneck,
not software. So software should be a tiny percentage of the time
it takes to send or receive any packet.

With packets limited to 1.5K and below things aren't always that
clear cut but the ideal remains. But basically anything we can
do besides remove the copy in and the copy out of the network stack
we should do.

The copy in and the copy out are fundamentally hard to remove without
modifying page tables which because of tlb invalidates can be even
more expensive than a copy.

The best you can hope for on a loopback scenario is a single copy
from one user space buffer to another skipping the intermediate
kernel buffer. But that is tricky for an entirely different set
of reasons.

Eric