Subject: Re: [patch 2/6] [Network namespace] Network device sharing by view
Posted by dev on Tue, 27 Jun 2006 09:54:51 GMT
View Forum Message <> Reply to Message

>> My point is that if you make namespace tagging at routing time, and
>> your packets are being routed only once, you lose the ability
>> to have separate routing tables in each namespace.
>
>
> Right. What is the advantage of having separate the routing tables ?
it is impossible to have bridged networking, tun/tap and many other
features without it. I even doubt that it is possible to introduce
private netfilter rules w/o virtualization of routing.

The question is do we want to have fully featured namespaces which allow
to create isolated virtual environments with semantics and behaviour of
standalone linux box or do we want to introduce some hacks with new
rules/restrictions to meet ones goals only?

From my POV, fully virtualized namespaces are the future. 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.

Thanks,
Kirill

Subject: Re: [patch 2/6] [Network namespace] Network device sharing by view
Posted by Herbert Poetzl on Tue, 27 Jun 2006 16:09:08 GMT
View Forum Message <> Reply to Message

On Tue, Jun 27, 2006 at 01:54:51PM +0400, Kirill Korotaev wrote:
> >>My point is that if you make namespace tagging at routing time, and
> >>your packets are being routed only once, you lose the ability
> >>to have separate routing tables in each namespace.
> >
> >
> >Right. What is the advantage of having separate the routing tables ?

> it is impossible to have bridged networking, tun/tap and many other
> features without it. I even doubt that it is possible to introduce
> private netfilter rules w/o virtualization of routing.

why? iptables work quite fine on a typical linux
system when you 'delegate' certain functionality
to certain chains (i.e. doesn't require access to
all of them)

> The question is do we want to have fully featured namespaces which
> allow to create isolated virtual environments with semantics and
> behaviour of standalone linux box or do we want to introduce some
> hacks with new rules/restrictions to meet ones goals only?

well, soemtimes 'hacks' are not only simpler but also
a much better solution for a given problem than the
straight forward approach ...

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 :)

> 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?

best,
Herbert

> Thanks,
> Kirill

Subject: Re: [patch 2/6] [Network namespace] Network device sharing by view
Posted by ebiederm on Tue, 27 Jun 2006 16:29:39 GMT
View Forum Message <> Reply to Message

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

> On Tue, Jun 27, 2006 at 01:54:51PM +0400, Kirill Korotaev wrote:
>> >>My point is that if you make namespace tagging at routing time, and
>> >>your packets are being routed only once, you lose the ability
>> >>to have separate routing tables in each namespace.
>> >
>> >
>> >Right. What is the advantage of having separate the routing tables ?
>
>> it is impossible to have bridged networking, tun/tap and many other

>> features without it. I even doubt that it is possible to introduce
>> private netfilter rules w/o virtualization of routing.
>
> why? iptables work quite fine on a typical linux
> system when you 'delegate' certain functionality
> to certain chains (i.e. doesn't require access to
> _all_ of them)
>
>> The question is do we want to have fully featured namespaces which
>> allow to create isolated virtual environments with semantics and
>> behaviour of standalone linux box or do we want to introduce some
>> hacks with new rules/restrictions to meet ones goals only?
>
> well, soemtimes 'hacks' are not only simpler but also
> a much better solution for a given problem than the
> straight forward approach ...

Well I would like to see a hack that qualifies. I watched the
linux-vserver irc channel for a while and almost every network problem
was caused by the change in semantics vserver provides.

In this case when you allow a guest more than one IP your hack while
easy to maintain becomes much more complex. Especially as you address
each case people care about one at a time.

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.

> 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.

>> 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.

Eric

Subject: Re: [patch 2/6] [Network namespace] Network device sharing by view
Posted by Herbert Poetzl on Tue, 27 Jun 2006 23:07:23 GMT
View Forum Message <> Reply to Message

On Tue, Jun 27, 2006 at 10:29:39AM -0600, Eric W. Biederman wrote:
> Herbert Poetzl <herbert@13thfloor.at> writes:
>
> > On Tue, Jun 27, 2006 at 01:54:51PM +0400, Kirill Korotaev wrote:
> >> >>My point is that if you make namespace tagging at routing time, and
> >> >>your packets are being routed only once, you lose the ability
> >> >>to have separate routing tables in each namespace.
> >> >
> >> >
> >> >Right. What is the advantage of having separate the routing tables ?
> >
> >> it is impossible to have bridged networking, tun/tap and many other
> >> features without it. I even doubt that it is possible to introduce
> >> private netfilter rules w/o virtualization of routing.
> >
> > why? iptables work quite fine on a typical linux
> > system when you 'delegate' certain functionality
> > to certain chains (i.e. doesn't require access to
> > _all_ of them)
> >
> >> The question is do we want to have fully featured namespaces which
> >> allow to create isolated virtual environments with semantics and
> >> behaviour of standalone linux box or do we want to introduce some
> >> hacks with new rules/restrictions to meet ones goals only?
> >
> > well, soemtimes 'hacks' are not only simpler but also
> > a much better solution for a given problem than the
> > straight forward approach ...
>
> Well I would like to see a hack that qualifies.

> 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 ...

> 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 ...

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

hmm?

> 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 ...

> >> 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

best,
Herbert

> Eric

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

Subject: Re: [patch 2/6] [Network namespace] Network device sharing by view
Posted by Sam Vilain on Wed, 28 Jun 2006 06:30:41 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> 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.
>

Yes, it does break what some people consider to be a sanity condition
when you don't have loopback anymore within a guest. I once experimented
with using 127.* addresses for per-guest loopback devices with vserver
to fix this, but that couldn't work without fixing glibc to not make
assumptions deep in the bowels of the resolver. I logged a fault with
gnu.org and you can guess where it went :-).

I don't think it's just the performance issue, though. Consider also
that if you only have one set of interfaces to manage, the overall
configuration of the network stack is simpler. `ip addr list' on the
host shows all the addresses on the system, you only have one routing
table to manage, one set of iptables, etc.

That being said, perhaps if each guest got its own interface, and from
some suitably privileged context you could see them all, perhaps it
would be nicer and maybe just as fast. Perhaps then *devices* could get
their own routing namespaces, and routing namespaces could get iptables
namespaces, or something like that, to give the most options.

> 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?
>

It used to just bind to the first one. Don't know if it still does.

Sam.

Subject: Re: [patch 2/6] [Network namespace] Network device sharing by view
Posted by Cedric Le Goater on Wed, 28 Jun 2006 10:14:28 GMT

View Forum Message <> Reply to Message

Hi !

Eric W. Biederman wrote:

[...]

> 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.

just one comment on the 'guest to guest communication' topic :

guest to guest communication is an important factor in consolidation
scenarios, where containers are packed on one server. This for maintenance
issues or priority issues on a HPC cluster for example. This case of
container migration is problably the most interesting and the performance
should be more than acceptable. May be not a top priority for the moment.

thanks,

C.

Subject: Re: [patch 2/6] [Network namespace] Network device sharing by view
Posted by Herbert Poetzl on Wed, 28 Jun 2006 14:11:48 GMT
View Forum Message <> Reply to Message

On Tue, Jun 27, 2006 at 10:07:29PM -0600, Eric W. Biederman wrote:
> 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.

not at all, you probably looked at a different
code, binding to INADDR_ANY actually _is_ the
default inside a guest, the only difference here
is that INADDR_ANY maps to a subset of _all_
available IPs ...

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

this does not happen either, as I said several
times, networking happens as on a _normal_
linux system, local traffic uses loop back, while
outbound traffic uses the appropriate network
interfaces

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

this 'semantic' change is intentional, and it
would be quite easy to change that (by putting
the host in a network context too) but as the
mechanism is isolation the host, similar to the
chroot() semantic for filesystems, sees _all_
the interfaces and IPs and therefore can also
bind to all of them ...

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

that does not happen either ...

IMHO you should have a closer look (or ask more
questions) before making false assumptions

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

there are no real exceptions, we have a legacy
mapping which basically 'remaps' localhost to
the first assigned IP (to make guest local traffic
secure without messing with the network stack)
but this can be avoided completely

> 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.

assuming that it will not be slower, although it
will now pass two network stacks and the bridging
code is non-productive too, let's see how it goes
but do not ignore the overhead just because it
might simplify the implementation ...

> 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.

that is a good argument and I think I'm perfectly
fine with this, given that the implementation
allows that (i.e. the network stack can handle
two interfaces with the same IP assigned and will
choose the local interface over the remote one
when the traffic will be between guests)

> Similarly for any critical guest communication to the outside world
> I can give the guest a real network adapter.

with a single MAC assigned, that is, I presume?

> 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.

guess that's what this discussion is about,
finding out the various aspects how isolation
and/or vitrtualization can be accomplished and
what features we consider common/useful enough
for mainline ... for me that is still in the
brainstorming phase, although several 'working
prototypes' already exist. IMHO the next step
is to collect a set of representative use cases

and test them with each implementation, regarding
performance, usability and practicability

> 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.

I'm thinking of all the various implementations
and 'prototypes' we did and tested, I agree
this might be confusing ...

> >> 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.

not necessarily, but I _know_ that the overhead
added at layer 3 is unmeasureable, and it still
needs to be proven that this is true for a layer
2 solution (which I'd actually prefer, because
it solves the protocol _and_ setup issues)

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

that is your opinion, I really doubt that you
will have less maintenance when you apply policy
to the guests ...

example here (just to clarify):

 - let's assume we have eth0 on the host and in
   guest A and B, with the following setup:

   eth0(H) 192.168.0.1/24
   eth0(A) 10.0.1.1/16 10.0.1.2/16
   eth0(B) 10.0.2.1/16

 - now what keeps guest B from jsut assigning
   10.0.2.2/16 to eth0? you need some kind of
   mechanism to prevent that, and/or to block
   the packets using inappropriate IPs

 * in the first case, i.e. you prevent assigning
   certain IPs inside a guest, you get a semantic
   change in the behaviour compared to a normal
   system, but there is no additional overhead
   on the communication

 * in the second case, you have to maintain the
   policy mechanism and keep it in sync with the
   guest configuration (somehow), and of course
   you have to verify every communication

 - OTOH, if you do not care about collisions
   basically assuming the point "that's like
   a hub on a network, if there are two guests
   with the same ip, it will be trouble, but
   that's okay" then this becomes a real issue
   for providers with potentially 'evil' customers

best,

Herbert

> >> >> 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

Subject: Re: [patch 2/6] [Network namespace] Network device sharing by view
Posted by Herbert Poetzl on Wed, 28 Jun 2006 14:15:48 GMT
View Forum Message <> Reply to Message

On Wed, Jun 28, 2006 at 06:31:05PM +1200, Sam Vilain wrote:
> Eric W. Biederman wrote:
> > 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.
> >
>
> Yes, it does break what some people consider to be a sanity condition
> when you don't have loopback anymore within a guest. I once experimented
> with using 127.* addresses for per-guest loopback devices with vserver
> to fix this, but that couldn't work without fixing glibc to not make
> assumptions deep in the bowels of the resolver. I logged a fault with
> gnu.org and you can guess where it went :-).

this is what the lo* patches address, by providing
the required loopback isolation and providing lo
inside a guest (i.e. it looks and feels like a

normal system, except that you cannot modify the
interfaces from inside)

> I don't think it's just the performance issue, though. Consider also
> that if you only have one set of interfaces to manage, the overall
> configuration of the network stack is simpler. `ip addr list' on the
> host shows all the addresses on the system, you only have one routing
> table to manage, one set of iptables, etc.
>
> That being said, perhaps if each guest got its own interface, and from
> some suitably privileged context you could see them all, perhaps it
> would be nicer and maybe just as fast. Perhaps then *devices* could get
> their own routing namespaces, and routing namespaces could get iptables
> namespaces, or something like that, to give the most options.
>
> > 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?
> >
>
> It used to just bind to the first one. Don't know if it still does.

no, it _alway_ binds to INADDR_ANY and checks
against other sockets (in the same context)
comparing the lists of assigned IPs (the subset)

so all checks happen at bind/connect time and
always against the set of IPs, only exception is
a performance optimization we do for single IP
guests (where INADDR_ANY gets rewritten to the
single IP)

best,
Herbert

> Sam.

Subject: Re: [patch 2/6] [Network namespace] Network device sharing by view
Posted by ebiederm on Wed, 28 Jun 2006 15:36:40 GMT

View Forum Message <> Reply to Message

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

> On Wed, Jun 28, 2006 at 06:31:05PM +1200, Sam Vilain wrote:
>> Eric W. Biederman wrote:
>> > 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.
>> >
>>
>> Yes, it does break what some people consider to be a sanity condition
>> when you don't have loopback anymore within a guest. I once experimented
>> with using 127.* addresses for per-guest loopback devices with vserver
>> to fix this, but that couldn't work without fixing glibc to not make
>> assumptions deep in the bowels of the resolver. I logged a fault with
>> gnu.org and you can guess where it went :-).
>
> this is what the lo* patches address, by providing
> the required loopback isolation and providing lo
> inside a guest (i.e. it looks and feels like a
> normal system, except that you cannot modify the
> interfaces from inside)

Ok. This is new. How do you talk between guests now?
Before those patches it was through IP addresses on the loopback interface
as I recall.

>> > 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?
>> >
>>
>> It used to just bind to the first one. Don't know if it still does.
>
> no, it _alway_ binds to INADDR_ANY and checks
> against other sockets (in the same context)
> comparing the lists of assigned IPs (the subset)
>
> so all checks happen at bind/connect time and
> always against the set of IPs, only exception is
> a performance optimization we do for single IP
> guests (where INADDR_ANY gets rewritten to the
> single IP)

What is the mechanism there?

My rough extrapolation says this mechanism causes problems when
migrating between machines. In particular it sounds like
only one process can bind to *:80, even if it is only allowed
to accept connections from a subset of those IPs.

So if on another machine I bound something to *:80 and only allowed to

use a different set of IPs and then attempted to migrate it, the
migration would fail because I could not restart the application,
with all of it's layer 3 resources.

To be clear I assume when I migrate I always take my IP address or
addresses with me.

Eric

Subject: Re: [patch 2/6] [Network namespace] Network device sharing by view
Posted by ebiederm on Wed, 28 Jun 2006 16:10:57 GMT
View Forum Message <> Reply to Message

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

>> 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.
>
> assuming that it will not be slower, although it
> will now pass two network stacks and the bridging
> code is non-productive too, let's see how it goes
> but do not ignore the overhead just because it
> might simplify the implementation ...

Sure. Mostly I have set it aside because the overhead
is not horrible and it is a very specific case that
can be heavily optimized if the core infrastructure is
solid.

>> 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.
>
> that is a good argument and I think I'm perfectly
> fine with this, given that the implementation
> allows that (i.e. the network stack can handle
> two interfaces with the same IP assigned and will
> choose the local interface over the remote one

> when the traffic will be between guests)

Yep. That exists today. The network stack prefers routes
as specific as possible.

>> Similarly for any critical guest communication to the outside world
>> I can give the guest a real network adapter.
>
> with a single MAC assigned, that is, I presume?

Yes.

>
> guess that's what this discussion is about,
> finding out the various aspects how isolation
> and/or vitrtualization can be accomplished and
> what features we consider common/useful enough
> for mainline ... for me that is still in the
> brainstorming phase, although several 'working
> prototypes' already exist. IMHO the next step
> is to collect a set of representative use cases
> and test them with each implementation, regarding
> performance, usability and practicability

I am fairly strongly convinced a layer 2 solution will
do fine. So for me it is a matter of proving that
and ensuring a good implementation.

> not necessarily, but I _know_ that the overhead
> added at layer 3 is unmeasureable, and it still
> needs to be proven that this is true for a layer
> 2 solution (which I'd actually prefer, because
> it solves the protocol _and_ setup issues)

That is a good perspective. Layer 3 is free, is layer 2 also free?
Unless the cache miss penalty is a killer layer 2 should come very
close. Of course VJ recently gave some evidence that packet processing
is dominated by cache misses.

>> >From what I have seen of layer 3 solutions it is a
>> bloody maintenance nightmare, and an inflexible mess.
>
> that is your opinion, I really doubt that you
> will have less maintenance when you apply policy
> to the guests ...

Yes and mostly of the layer 3 things that I implemented.
At a moderately fundamental level I see layer 3 implementations

being a special case that is a tangent from the rest of the
networking code. So I don't see a real synthesis with what
the rest of the networking stack is doing. Plus all of the
limitations that come with a layer 3 implementation.

> example here (just to clarify):
>
> - let's assume we have eth0 on the host and in
>   guest A and B, with the following setup:
>
>   eth0(H) 192.168.0.1/24
>   eth0(A) 10.0.1.1/16 10.0.1.2/16
>   eth0(B) 10.0.2.1/16
>
> - now what keeps guest B from jsut assigning
>   10.0.2.2/16 to eth0? you need some kind of
>   mechanism to prevent that, and/or to block
>   the packets using inappropriate IPs
>
> * in the first case, i.e. you prevent assigning
>   certain IPs inside a guest, you get a semantic
>   change in the behaviour compared to a normal
>   system, but there is no additional overhead
>   on the communication
>
> * in the second case, you have to maintain the
>   policy mechanism and keep it in sync with the
>   guest configuration (somehow), and of course
>   you have to verify every communication
>
> - OTOH, if you do not care about collisions
>   basically assuming the point "that's like
>   a hub on a network, if there are two guests
>   with the same ip, it will be trouble, but
>   that's okay" then this becomes a real issue
>   for providers with potentially 'evil' customers

So linux when serving as a router has strong filter capabilities.

So we can either use the strong network filtering linux already has
making work for the host administrator who has poorly behaved customers.
Or we can simply not give those poorly behaved guests CAP_NET_ADMIN,
and assign the IP address at guest startup before dropping the
capability. At which point the guest cannot misbehave.

Eric

Subject: Re: [patch 2/6] [Network namespace] Network device sharing by view
Posted by Herbert Poetzl on Wed, 28 Jun 2006 17:18:37 GMT
View Forum Message <> Reply to Message

On Wed, Jun 28, 2006 at 09:36:40AM -0600, Eric W. Biederman wrote:
> Herbert Poetzl <herbert@13thfloor.at> writes:
>
> > On Wed, Jun 28, 2006 at 06:31:05PM +1200, Sam Vilain wrote:
> >> Eric W. Biederman wrote:
> >> > 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.
> >> >
> >>
> >> Yes, it does break what some people consider to be a sanity
> >> condition when you don't have loopback anymore within a guest. I
> >> once experimented with using 127.* addresses for per-guest loopback
> >> devices with vserver to fix this, but that couldn't work without
> >> fixing glibc to not make assumptions deep in the bowels of the
> >> resolver. I logged a fault with gnu.org and you can guess where it
> >> went :-).
> >
> > this is what the lo* patches address, by providing
> > the required loopback isolation and providing lo
> > inside a guest (i.e. it looks and feels like a
> > normal system, except that you cannot modify the
> > interfaces from inside)
>
> Ok. This is new. How do you talk between guests now?

> Before those patches it was through IP addresses on the loopback
> interface as I recall.

no, that was probably your assumption, the IPs are
assigned (in a perfectly normal way) to the interfaces
(e.g. eth0 carries some IPs for guest A and B, eth1
carries others for guest C). the way the linux network
stack works, local addresses (i.e. those of A,B and C)
will automatically communicate via loopback (as they
are local) while outbound traffic will use the proper
interface (nothing is changed here)

the difference in the lo patches is, that we allow to
use the 'localhost' ip range (127.x.x.x) by isolating
traffic (in this range) on the loopback interface
(which typically allows to have 127.0.0.1 and lo

visible inside a guest)

> >> > 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?
> >> >
> >>
> >> It used to just bind to the first one. Don't know if it still does.
> >
> > no, it _alway_ binds to INADDR_ANY and checks
> > against other sockets (in the same context)
> > comparing the lists of assigned IPs (the subset)
> >
> > so all checks happen at bind/connect time and
> > always against the set of IPs, only exception is
> > a performance optimization we do for single IP
> > guests (where INADDR_ANY gets rewritten to the
> > single IP)
>
> What is the mechanism there?
>
> My rough extrapolation says this mechanism causes problems when
> migrating between machines.

that might be, as we do not consider migration such
important as other folks do :)

> In particular it sounds like only one process can bind to *:80, even
> if it is only allowed to accept connections from a subset of those
> IPs.

no, guest A,B and C can all bind to *:80 and coexist
quite fine, given that they do not have any IP in
the intersection of their subsets (which is checked
at bind time)

> So if on another machine I bound something to *:80 and only allowed to
> use a different set of IPs and then attempted to migrate it, the
> migration would fail because I could not restart the application,
> with all of it's layer 3 resources.

actually I do not see why, unless the destination
has a conflict on the ip subset, in which case you
would end up with a migrated, but not working guest :)

> To be clear I assume when I migrate I always take my IP address or
> addresses with me.

that's fine, the only requirement would be that the
host has a superset of the IP addresses used by the
guests ...

HTC,
Herbert

> Eric