Subject: Re: [patch 2/6] [Network namespace] Network device sharing by view
Posted by Andrey Savochkin on Mon, 26 Jun 2006 09:47:11 GMT
View Forum Message <> Reply to Message

Hi Daniel,

It's good that you kicked off network namespace discussion.
Although I wish you'd Cc'ed someone at OpenVZ so I could notice it earlier :).

Indeed, the first point to agree in this discussion is device list.
In your patch, you essentially introduce a data structure parallel
to the main device list, creating a "view" of this list.
I see a fundamental problem with this approach.
When a device presents an skb to the protocol layer, it needs to know to which
namespace this skb belongs.
Otherwise you would never get rid of problems with bind: what to do if device
eth1 is visible in namespace1, namespace2, and root namespace, and each
namespace has a socket bound to 0.0.0.0:80?

We have to conclude that each device should be visible only in one namespace.
In this case, instead of introducing net_ns_dev and net_ns_dev_list
structures, we can simply have a separate dev_base list head in each namespace.
Moreover, separate device list in each namespace will be in line with
making namespace isolation complete. Complete isolation will allow each
namespace to set up own tun/tap devices, have own routes, netfilter tables,
and so on.

My follow-up messages will contain the first set of patches with network
namespaces implemented in the same way as network isolation in OpenVZ.
This patchset introduces namespaces for device list and IPv4 FIB/routing.
Two technical issues are omitted to make the patch idea clearer: device moving
between namespaces, and selective routing cache flush + garbage collection.

If this patchset is agreeable, the next patchset will finalize integration
with nsproxy, add namespaces to socket lookup code and neighbour
cache, and introduce a simple device to pass traffic between namespaces.
Then we will turn to less obvious matters including netlink messages,
network statistics, representation of network information in proc and sysfs,
tuning of parameters through sysctl, IPv6 and other protocols, and
per-namespace netfilters.

Best regards
Andrey

Subject: Re: [patch 2/6] [Network namespace] Network device sharing by view
Posted by Herbert Poetzl on Mon, 26 Jun 2006 13:02:03 GMT

On Mon, Jun 26, 2006 at 01:47:11PM +0400, Andrey Savochkin wrote:
> Hi Daniel,
>
> It's good that you kicked off network namespace discussion Although I.
> wish you'd Cc'ed someone at OpenVZ so I could notice it earlier :) .

> Indeed, the first point to agree in this discussion is device list.
> In your patch, you essentially introduce a data structure parallel
> to the main device list, creating a "view" of this list.

> I see a fundamental problem with this approach. When a device presents
> an skb to the protocol layer, it needs to know to which namespace this
> skb belongs.

> Otherwise you would never get rid of problems with bind: what to do if
> device eth1 is visible in namespace1, namespace2, and root namespace,
> and each namespace has a socket bound to 0.0.0.0:80?

this is something which isn't a fundamental problem at
all, and IMHO there are at least three options here
(probably more)

- check at 'bind' time if the binding would overlap
  and give the 'proper' error (as it happens right
  now on the host)
  (this is how Linux-VServer currently handles the
  network isolation, and yes, it works quite fine :)

- allow arbitrary binds and 'tag' the packets according
  to some 'host' policy (e.g. iptables or tc)
  (this is how the Linux-VServer ngnet was designed)

- deliver packets to _all_ bound sockets/destinations
  (this is probably a more unusable but quite thinkable
  solution)

> We have to conclude that each device should be visible only in one
> namespace.

I disagree here, especially some supervisor context or
the host context should be able to 'see' and probably
manipulate _all_ of the devices

> In this case, instead of introducing net_ns_dev and net_ns_dev_list
> structures, we can simply have a separate dev_base list head in each
> namespace. Moreover, separate device list in each namespace will be in
> line with making namespace isolation complete.

> Complete isolation will allow each namespace to set up own tun/tap
> devices, have own routes, netfilter tables, and so on.

tun/tap devices are quite possible with this approach
too, I see no problem here ...

for iptables and routes, I'm worried about the required
'policy' to make them secure, i.e. how do you ensure
that the packets 'leaving' guest X do not contain
'evil' packets and/or disrupt your host system?

> My follow-up messages will contain the first set of patches with
> network namespaces implemented in the same way as network isolation
> in OpenVZ.

hmm, you probably mean 'network virtualization' here

> This patchset introduces namespaces for device list and IPv4
> FIB/routing. Two technical issues are omitted to make the patch idea
> clearer: device moving between namespaces, and selective routing cache
> flush + garbage collection.
>
> If this patchset is agreeable, the next patchset will finalize
> integration with nsproxy, add namespaces to socket lookup code and
> neighbour cache, and introduce a simple device to pass traffic between
> namespaces.

passing traffic 'between' namespaces should happen via
lo, no? what kind of 'device' is required there, and
what overhead does it add to the networking?

TIA,
Herbert

> Then we will turn to less obvious matters including
> netlink messages, network statistics, representation of network
> information in proc and sysfs, tuning of parameters through sysctl,
> IPv6 and other protocols, and per-namespace netfilters.
>
> Best regards
>  Andrey

---

Subject: Re: [patch 2/6] [Network namespace] Network device sharing by view
Posted by ebiederm on Mon, 26 Jun 2006 14:05:24 GMT
View Forum Message <> Reply to Message

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

> On Mon, Jun 26, 2006 at 01:47:11PM +0400, Andrey Savochkin wrote:
>> Hi Daniel,
>>
>> It's good that you kicked off network namespace discussion Although I.
>> wish you'd Cc'ed someone at OpenVZ so I could notice it earlier :)  .
>
>> Indeed, the first point to agree in this discussion is device list.
>> In your patch, you essentially introduce a data structure parallel
>> to the main device list, creating a "view" of this list.
>
>> I see a fundamental problem with this approach. When a device presents
>> an skb to the protocol layer, it needs to know to which namespace this
>> skb belongs.
>
>> Otherwise you would never get rid of problems with bind: what to do if
>> device eth1 is visible in namespace1, namespace2, and root namespace,
>> and each namespace has a socket bound to 0.0.0.0:80?
>
> this is something which isn't a fundamental problem at
> all, and IMHO there are at least three options here
> (probably more)

I agree that there are other implementations that can be used for
containers.  However when you think namespaces this is what you need.

For several reasons.
1) So you can use AF_PACKET safely.
   This allows a network namespace to use DHCP and all of the other
   usual network autoconfiguration tools.  0.0.0.0:80 is just
   a special subset of that.

2) It means the existing network stack can be used without
   logic changes.  All that is needed is a lookup of the appropriate
   context.  This is very straight forward to audit.

3) Since all of the network stack is trivially available all of
   the advanced network stack features like iptables are easily
   available.

4) There is no retraining or other rules for user to learn.
   Because people understand what is going on it is more likely
   a setup will be secure.  Most of the other implementations
   don't quite act like a normal network setup and the special
   rules can be hard to learn.

>  - check at 'bind' time if the binding would overlap

>  and give the 'proper' error (as it happens right
>  now on the host)
>  (this is how Linux-VServer currently handles the
>  network isolation, and yes, it works quite fine :)

It works yes but it limits you to a subset of the network
stack.  And has serious problems with concepts like INADDR_ANY.
PF_PACKET is not an option.

> - allow arbitrary binds and 'tag' the packets according
>  to some 'host' policy (e.g. iptables or tc)
>  (this is how the Linux-VServer ngnet was designed)

A little more general but very weird.

> - deliver packets to _all_ bound sockets/destinations
>  (this is probably a more unusable but quite thinkable
>  solution)
>
>> We have to conclude that each device should be visible only in one
>> namespace.
>
> I disagree here, especially some supervisor context or
> the host context should be able to 'see' and probably
> manipulate _all_ of the devices

This part really is necessary.  This does not preclude managing
a network namespace from outside of the namespace.

>> In this case, instead of introducing net_ns_dev and net_ns_dev_list
>> structures, we can simply have a separate dev_base list head in each
>> namespace. Moreover, separate device list in each namespace will be in
>> line with making namespace isolation complete.
>
>> Complete isolation will allow each namespace to set up own tun/tap
>> devices, have own routes, netfilter tables, and so on.
>
> tun/tap devices are quite possible with this approach
> too, I see no problem here ...
>
> for iptables and routes, I'm worried about the required
> 'policy' to make them secure, i.e. how do you ensure
> that the packets 'leaving' guest X do not contain
> 'evil' packets and/or disrupt your host system?

In the traditional ways.  When you control the router and/or the switch
someone is directly connected to.

We don't need to reinvent the wheel if we do this properly.

>> This patchset introduces namespaces for device list and IPv4
>> FIB/routing. Two technical issues are omitted to make the patch idea
>> clearer: device moving between namespaces, and selective routing cache
>> flush + garbage collection.
>>
>> If this patchset is agreeable, the next patchset will finalize
>> integration with nsproxy, add namespaces to socket lookup code and
>> neighbour cache, and introduce a simple device to pass traffic between
>> namespaces.
>
> passing traffic 'between' namespaces should happen via
> lo, no? what kind of 'device' is required there, and
> what overhead does it add to the networking?

Definitely not.  lo is a local loopback interface.

What is needed is a two headed device that is the cousin of lo.
But with one network interface in each network namespace.

Note even connecting network namespaces is optional.

Eric

---

Subject: Re: [patch 2/6] [Network namespace] Network device sharing by view
Posted by Andrey Savochkin on Mon, 26 Jun 2006 14:08:03 GMT
View Forum Message <> Reply to Message

Hi Herbert,

On Mon, Jun 26, 2006 at 03:02:03PM +0200, Herbert Poetzl wrote:
> On Mon, Jun 26, 2006 at 01:47:11PM +0400, Andrey Savochkin wrote:
>
> > I see a fundamental problem with this approach. When a device presents
> > an skb to the protocol layer, it needs to know to which namespace this
> > skb belongs.
>
> > Otherwise you would never get rid of problems with bind: what to do if
> > device eth1 is visible in namespace1, namespace2, and root namespace,
> > and each namespace has a socket bound to 0.0.0.0:80?
>
> this is something which isn't a fundamental problem at
> all, and IMHO there are at least three options here
> (probably more)
>
>  - check at 'bind' time if the binding would overlap

> and give the 'proper' error (as it happens right
> now on the host)
> (this is how Linux-VServer currently handles the
> network isolation, and yes, it works quite fine :)

I'm not comfortable with this as a permanent mainstream solution.
It means that network namespaces are actually not namespaces: you can't run some program (e.g., apache) with default configs in a new namespace without regards to who runs what in other namespaces.
In other words, name "0.0.0.0:80" creates a collision in your implementation, so socket "names" do not form isolated spaces.

>
> - allow arbitrary binds and 'tag' the packets according
> to some 'host' policy (e.g. iptables or tc)
> (this is how the Linux-VServer ngnet was designed)
>
> - deliver packets to _all_ bound sockets/destinations
> (this is probably a more unusable but quite thinkable
> solution)

Deliver TCP packets to all sockets?
How many connections do you expect to be established in this case?

>
> > We have to conclude that each device should be visible only in one
> > namespace.
>
> I disagree here, especially some supervisor context or
> the host context should be able to 'see' and probably
> manipulate _all_ of the devices

Right, manipulating all devices from some supervisor context is useful.

But this shouldn't necessarily be done by regular ip/ifconfig tools.
Besides, it could be quite confusing if in ifconfig output in the supervisor context you see 325 "tun0" devices coming from different namespaces :)

So I'm all for network namespace management mechanisms not bound to existing tools/APIs.

>
> > Complete isolation will allow each namespace to set up own tun/tap
> > devices, have own routes, netfilter tables, and so on.
>
> tun/tap devices are quite possible with this approach
> too, I see no problem here ...

>
> for iptables and routes, I'm worried about the required
> 'policy' to make them secure, i.e. how do you ensure
> that the packets 'leaving' guest X do not contain
> 'evil' packets and/or disrupt your host system?

Sorry, I don't get your point.
How do you ensure that packets leaving your neighbor's computer
do not disrupt your system?
>From my point of view, network namespaces are just neighbors.

>
> > My follow-up messages will contain the first set of patches with
> > network namespaces implemented in the same way as network isolation
> > in OpenVZ.
>
> hmm, you probably mean 'network virtualization' here

I meant isolation between different network contexts/namespaces.

>
> > This patchset introduces namespaces for device list and IPv4
> > FIB/routing. Two technical issues are omitted to make the patch idea
> > clearer: device moving between namespaces, and selective routing cache
> > flush + garbage collection.
> >
> > If this patchset is agreeable, the next patchset will finalize
> > integration with nsproxy, add namespaces to socket lookup code and
> > neighbour cache, and introduce a simple device to pass traffic between
> > namespaces.
>
> passing traffic 'between' namespaces should happen via
> lo, no? what kind of 'device' is required there, and
> what overhead does it add to the networking?

OpenVZ provides 2 options.

1) A packet appears right inside some namespace, without any additional
   overhead. Usually this implies that either all packets from this device
   belong to this namespace, i.e. simple device->namespace assignment.
   However, there is nothing conceptually wrong with having
   namespace-aware device drivers or netfilter modules selecting namespaces
   for each incoming packet. It all depends on how you want packets go
   through various network layers, and how much network management abilities
   you want to have in non-root namespaces.
   My point is that for network namespaces being real namespaces, decision
   making should be done somewhere before socket lookup.

2) Parent network namespace acts as a router forwarding packets to child
namespaces. This scheme is the preferred one in OpenVZ for various
reasons, most important being the simplicity of migration of network
namespaces. In this case flexibility has the cost of going through
packet handling layers two times.
Technically, this is implemented via a simple netdevice doing
netif_rx in hard_xmit.

Regards

Andrey

---

Subject: Re: [patch 2/6] [Network namespace] Network device sharing by view
Posted by Daniel Lezcano on Mon, 26 Jun 2006 14:56:32 GMT
View Forum Message <> Reply to Message

Andrey Savochkin wrote:
> Hi Daniel,

Hi Andrey,

>
> It's good that you kicked off network namespace discussion.
> Although I wish you'd Cc'ed someone at OpenVZ so I could notice it earlier :).

devel@openvz.org ?

> When a device presents an skb to the protocol layer, it needs to know to which
> namespace this skb belongs.
> Otherwise you would never get rid of problems with bind: what to do if device
> eth1 is visible in namespace1, namespace2, and root namespace, and each
> namespace has a socket bound to 0.0.0.0:80?

Exact. But, the idea was to retrieve the namespace from the routes.

IMHO, I think there are roughly 2 network isolation implementation:

- make all network ressources private to the namespace

- keep a "flat" model where network ressources have a new identifier
which is the network namespace pointer. The idea is to move only some
network informations private to the namespace (eg port range, stats, ...)

Daniel.

---

Subject: Re: [patch 2/6] [Network namespace] Network device sharing by view
Posted by Herbert Poetzl on Mon, 26 Jun 2006 18:28:48 GMT
View Forum Message <> Reply to Message

On Mon, Jun 26, 2006 at 06:08:03PM +0400, Andrey Savochkin wrote:
> Hi Herbert,
>
> On Mon, Jun 26, 2006 at 03:02:03PM +0200, Herbert Poetzl wrote:
> > On Mon, Jun 26, 2006 at 01:47:11PM +0400, Andrey Savochkin wrote:
> >
> > > I see a fundamental problem with this approach. When a device
> > > presents an skb to the protocol layer, it needs to know to which
> > > namespace this skb belongs.
> >
> > > Otherwise you would never get rid of problems with bind: what to
> > > do if device eth1 is visible in namespace1, namespace2, and root
> > > namespace, and each namespace has a socket bound to 0.0.0.0:80?
> >
> > this is something which isn't a fundamental problem at
> > all, and IMHO there are at least three options here
> > (probably more)
> >
> >  - check at 'bind' time if the binding would overlap
> >    and give the 'proper' error (as it happens right
> >    now on the host)
> >    (this is how Linux-VServer currently handles the
> >    network isolation, and yes, it works quite fine :)
>
> I'm not comfortable with this as a permanent mainstream solution.
> It means that network namespaces are actually not namespaces: you
> can't run some program (e.g., apache) with default configs in a new
> namespace without regards to who runs what in other namespaces.

not at all, maybe you should take a closer look at the
current Linux-VServer implementation, which is quite
simple and _does_ allow guests to bind to IP_ANY quite
fine, only the host (which has all priviledes) has to
be careful with binding to 0.0.0.0 ...

> In other words, name "0.0.0.0:80" creates a collision in your
> implementation, so socket "names" do not form isolated spaces.
>
> >
> >  - allow arbitrary binds and 'tag' the packets according
> >    to some 'host' policy (e.g. iptables or tc)
> >    (this is how the Linux-VServer ngnet was designed)
> >
> >  - deliver packets to _all_ bound sockets/destinations
> >    (this is probably a more unusable but quite thinkable

> >    solution)
>
> Deliver TCP packets to all sockets?
> How many connections do you expect to be established in this case?

well, roughly the same number of connections you'll
get when you have two boxes with the same IP on the
same subnet :)

in other words, if there are more than one guest
with the same ip and port open, then we have some
kind of misconfiguration (i.e. policy is required)

> > > We have to conclude that each device should be visible only in one
> > > namespace.
> >
> > I disagree here, especially some supervisor context or
> > the host context should be able to 'see' and probably
> > manipulate _all_ of the devices
>
> Right, manipulating all devices from some supervisor context is useful.
>
> But this shouldn't necessarily be done by regular ip/ifconfig tools.
> Besides, it could be quite confusing if in ifconfig output in the
> supervisor context you see 325 "tun0" devices coming from
> different namespaces :)

isolation would not provide more than _one_ tun0
interfaces, virtualization OTOH will ...

> So I'm all for network namespace management mechanisms not bound
> to existing tools/APIs.

well, I'm not against new APIs/tools, but I prefer
to keep it simple, and elegant, which often includes
reusing existing APIs and tools ...

> > > Complete isolation will allow each namespace to set up own tun/tap
> > > devices, have own routes, netfilter tables, and so on.
> >
> > tun/tap devices are quite possible with this approach
> > too, I see no problem here ...
> >
> > for iptables and routes, I'm worried about the required
> > 'policy' to make them secure, i.e. how do you ensure
> > that the packets 'leaving' guest X do not contain
> > 'evil' packets and/or disrupt your host system?
>

> Sorry, I don't get your point.
> How do you ensure that packets leaving your neighbor's computer
> do not disrupt your system?

by having a strong 'policy' on the router/switch
which will (hopefully) reject everything sent in error
or to disrupt/harm other boxes ...

> >From my point of view, network namespaces are just neighbors.

yes, but you _need_ policy there the same way you need
it for resources, i.e. you cannot simply allow everyone
to do everything with his network interface, especially
if that interface is _shared_ with all others ...

> > > My follow-up messages will contain the first set of patches with
> > > network namespaces implemented in the same way as network isolation
> > > in OpenVZ.
> >
> > hmm, you probably mean 'network virtualization' here
>
> I meant isolation between different network contexts/namespaces.

well, isolation is basically what we do in Linux-VServer
by allowing to bind to certain IPs (or ranges) instead
of binding _all_ available IPs ... this can be extended
for routing and iptables as well, and does not require
any 'virtualization' which would give each guest it's own
set of interfaces, routes, iptables etc ... and it is
usually more lightweight too ..

> > > This patchset introduces namespaces for device list and IPv4
> > > FIB/routing. Two technical issues are omitted to make the patch
> > > idea clearer: device moving between namespaces, and selective
> > > routing cache flush + garbage collection.
> > >
> > > If this patchset is agreeable, the next patchset will finalize
> > > integration with nsproxy, add namespaces to socket lookup code and
> > > neighbour cache, and introduce a simple device to pass traffic
> > > between namespaces.
> >
> > passing traffic 'between' namespaces should happen via
> > lo, no? what kind of 'device' is required there, and
> > what overhead does it add to the networking?
>
> OpenVZ provides 2 options.
>

> 1) A packet appears right inside some namespace, without any additional
> overhead. Usually this implies that either all packets from this
> device belong to this namespace, i.e. simple device->namespace
> assignment. However, there is nothing conceptually wrong with
> having namespace-aware device drivers or netfilter modules
> selecting namespaces for each incoming packet. It all depends on
> how you want packets go through various network layers, and how
> much network management abilities you want to have in non-root
> namespaces. My point is that for network namespaces being real
> namespaces, decision making should be done somewhere before socket
> lookup.

well, I doubt that many providers will be able to put
roughly hundred or more network interface cards into
their machines, plus a proper switch to do the policy :)

> 2) Parent network namespace acts as a router forwarding packets to child
> namespaces. This scheme is the preferred one in OpenVZ for various
> reasons, most important being the simplicity of migration of network
> namespaces. In this case flexibility has the cost of going through
> packet handling layers two times.

> Technically, this is implemented via a simple netdevice doing
> netif_rx in hard_xmit.

which results in a duplicate stack traversal and kernel
side policy to decide which goes where ... i.e. at least
twice as much overhead than any isolation would have

best,
Herbert

> Regards
>
> Andrey

---

Subject: Re: [patch 2/6] [Network namespace] Network device sharing by view
Posted by ebiederm on Mon, 26 Jun 2006 18:59:41 GMT
View Forum Message <> Reply to Message

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

> On Mon, Jun 26, 2006 at 06:08:03PM +0400, Andrey Savochkin wrote:
>
> not at all, maybe you should take a closer look at the
> current Linux-VServer implementation, which is quite
> simple and _does_ allow guests to bind to IP_ANY quite

> fine, only the host (which has all priviledes) has to
> be careful with binding to 0.0.0.0 ...

It works, and is a reasonable implementation.  However the
semantics change.

The real practical problem is that you loose power, and
the ability to migrate applications.  Not that this precludes
you from loading a security module and doing what you do now.

>> > > We have to conclude that each device should be visible only in one
>> > > namespace.
>> >
>> > I disagree here, especially some supervisor context or
>> > the host context should be able to 'see' and probably
>> > manipulate _all_ of the devices
>>
>> Right, manipulating all devices from some supervisor context is useful.
>>
>> But this shouldn't necessarily be done by regular ip/ifconfig tools.
>> Besides, it could be quite confusing if in ifconfig output in the
>> supervisor context you see 325 "tun0" devices coming from
>> different namespaces :)
>
> isolation would not provide more than _one_ tun0
> interfaces, virtualization OTOH will ...

Think layer 2 isolation not layer 3 isolation.

>> So I'm all for network namespace management mechanisms not bound
>> to existing tools/APIs.
>
> well, I'm not against new APIs/tools, but I prefer
> to keep it simple, and elegant, which often includes
> reusing existing APIs and tools ...

And knowledge.  Except for the single IP per guest case filtering
at BIND time starts show some surprising semantics.

> by having a strong 'policy' on the router/switch
> which will (hopefully) reject everything sent in error
> or to disrupt/harm other boxes ...

And linux has a software router and switch capabilities so
those can easily be used unmodified.

>> >From my point of view, network namespaces are just neighbors.
>

> yes, but you _need_ policy there the same way you need
> it for resources, i.e. you cannot simply allow everyone
> to do everything with his network interface, especially
> if that interface is _shared_ with all others ...

Agreed.  And the network stack seems to have a perfectly good
set of utilities to handle that already.

>
> well, isolation is basically what we do in Linux-VServer
> by allowing to bind to certain IPs (or ranges) instead
> of binding _all_ available IPs ... this can be extended
> for routing and iptables as well, and does not require
> any 'virtualization' which would give each guest it's own
> set of interfaces, routes, iptables etc ... and it is
> usually more lightweight too ..

I disagree with the cost.  Done properly we should have the cost
of the existing networking stack plus the cost of an extra pointer
dereference when we look at global variables.

This is layer 2 isolation.  So we can use protocols like DHCP,
unmodified.

In the normally accepted definition it isn't virtualization because
we aren't emulating anything.

>> > > This patchset introduces namespaces for device list and IPv4
>> > > FIB/routing. Two technical issues are omitted to make the patch
>> > > idea clearer: device moving between namespaces, and selective
>> > > routing cache flush + garbage collection.
>> > >
>> > > If this patchset is agreeable, the next patchset will finalize
>> > > integration with nsproxy, add namespaces to socket lookup code and
>> > > neighbour cache, and introduce a simple device to pass traffic
>> > > between namespaces.
>> >
>> > passing traffic 'between' namespaces should happen via
>> > lo, no? what kind of 'device' is required there, and
>> > what overhead does it add to the networking?
>>
>> OpenVZ provides 2 options.
>>
>>  1) A packet appears right inside some namespace, without any additional
>>     overhead. Usually this implies that either all packets from this
>>     device belong to this namespace, i.e. simple device->namespace
>>     assignment. However, there is nothing conceptually wrong with
>>     having namespace-aware device drivers or netfilter modules

>> selecting namespaces for each incoming packet. It all depends on
>> how you want packets go through various network layers, and how
>> much network management abilities you want to have in non-root
>> namespaces. My point is that for network namespaces being real
>> namespaces, decision making should be done somewhere before socket
>> lookup.
>
> well, I doubt that many providers will be able to put
> roughly hundred or more network interface cards into
> their machines, plus a proper switch to do the policy :)

Well switches exist. But yes because physical hardware
is limited this is a limited policy.

>> 2) Parent network namespace acts as a router forwarding packets to child
>> namespaces. This scheme is the preferred one in OpenVZ for various
>> reasons, most important being the simplicity of migration of network
>> namespaces. In this case flexibility has the cost of going through
>> packet handling layers two times.
>
>> Technically, this is implemented via a simple netdevice doing
>> netif_rx in hard_xmit.
>
> which results in a duplicate stack traversal and kernel
> side policy to decide which goes where ... i.e. at least
> twice as much overhead than any isolation would have

Not twice because you don't traverse the entire network stack.
Just up to the routing layer and then across, and there are
optimization possibilities that should keep it down to a single
traversal of the network stack.

Note: We are not encouraging saying that the linux-vserver implementation
must die. Only that we are solving something with much larger scope.

If the first case does not at least pass packets as fast as the existing
network stack I doubt we will be allowed to merge it. By making the nic
drivers smarter we can have a single driver that creates multiple
network interfaces simply by looking at the destination mac address,
sort of like the bonding driver in reverse. That will trivially
remove the extra network stack traversals if we don't want
to apply before we let the packet out on the wire.

And there is not requirement that after the namespace is setup we
leave any applications on the inside with CAP_NET_ADMIN so we don't
need to worry about user space applications changing the network
configurations if we don't want to.

Eric