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