Subject: Re: [PATCH 0/12] L2 network namespace (v3)
Posted by ebiederm on Fri, 19 Jan 2007 07:27:30 GMT
View Forum Message <> Reply to Message

YOSHIFUJI Hideaki / $B5HF#1QL@ (B <yoshfuji@linux-ipv6.org> writes:

> In article <200701171851.14734.dim@openvz.org> (at Wed, 17 Jan 2007 18:51:14
> +0300), Dmitry Mishin <dim@openvz.org> says:
>
>> ==========
>> L2 network namespaces
>>
>> The most straightforward concept of network virtualization is complete
>> separation of namespaces, covering device list, routing tables, netfilter
>> tables, socket hashes, and everything else.
>>
>> On input path, each packet is tagged with namespace right from the
>> place where it appears from a device, and is processed by each layer
>> in the context of this namespace.
>> Non-root namespaces communicate with the outside world in two ways: by
>> owning hardware devices, or receiving packets forwarded them by their parent
>> namespace via pass-through device.
>
> Can you handle multicast / broadcast and IPv6, which are very important?

The basic idea here is very simple.

Each network namespace appears to user space as a separate network stack,
with it's own set of routing tables etc.

All sockets and all network devices (the sources of packets) belong
to exactly one network namespace.

>From the socket or the network device a packet enters the network stack
you can infer the network namespace that it will be processed in.
Each network namespace should get it own complement of the data structures
necessary to process packets, and everything should work.

Talking between namespaces is accomplished either through an external network,
or through a special pseudo network device. The simplest to implement
is two network devices where all packets transmitted on one are received
on the other. Then by placing one network device in one namespace and
the other in another interface it looks like two machines connected by
a cross over cable.

Once you have that in a one namespace you can connect other namespaces
with the existing ethernet bridging or by configuring one of the
namespaces as a router and routing traffic between them.

Supporting IPv6 is roughly as difficult as supporting IPv4.

What needs to happen to convert code is all variables either need
a per network namespace instance or the data structures needs to be
modified to have a network namespace tag. For hash tables which
are hard to allocate dynamically tagging is the preferred conversion
method, for anything that is small enough duplication is preferred
as it allows the existing logic to be kept.

In the fast path the impact of all of the conversions should be very light,
to non-existent. In network stack initialization and cleanup there
is work todo because you are initializing and cleanup variables more often
then at module insertion and removal.

So my expectation is that once we get a framework established and merged
to allow network namespaces eventually the entire network stack will be
converted. Not just ipv4 and ipv6 but decnet, ipx, iptables, fair scheduling,
ethernet bridging and all of the other weird and twisty bits of the
linux network stack.

The primary practical hurdle is there is a lot of networking code in
the kernel.

I think I know a path by which we can incrementally merge support for
network namespaces without breaking anything. More to come on this
when I finish up my demonstration patchset in a week or so that
is complete enough to show what I am talking about.

I hope this helps but the concept into perspective.

As for Dmitry's patchset in particular it currently does not support
IPv6 and I don't know where it is with respect to the broadcast and
multicast but I don't see any immediate problems that would preclude
those from working. But any incompleteness is exactly that
incompleteness and an implementation problem not a fundamental design
issue.

Eric

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers