


[Home](#) [Archives](#) [Profile](#) [Subscribe](#)
[Main](#)

08/27/2012

Network Virtualization - Networking's 21st century equivalent to the space race

This post was written with Patrick Mullaney.

Overview

Similar to the milestones in the [space race timeline](#), I believe that Monday July 23rd (the day VMware announced plans to acquire Nicira) will be viewed by networking enthusiasts as a significant milestone in the evolution of our discipline. Just to keep this in perspective, two other significant milestones will probably be Cisco's Insieme announcement, but not for the reason you might think and Oracle's announcement of plans to acquire Xsigo.

Of course chronologically the announcements were Insieme, Nicira and Xsigo. And since Insieme seems to be more about [Cisco entering the storage market](#) than about anything else, I'd like to think that a causal link exists between the three announcements in the same order. However, I have absolutely no proof of this and while further speculation might be a great topic for a techno-drama, it would ultimately be irresponsible. As a result, I'll spend the rest of this post providing a high level overview of Network Virtualization, how it relates to Software Defined Networking and this may allow you to speculate on what the next major milestones in the NV race may be...

Getting down to Brass-Tacks

Network Virtualization (NV) and its big brother Software Defined Networking (SDN) both attempt to address a set of [use cases](#) that have proven difficult to handle while using traditional Networking gear. These challenges range from the more esoteric (e.g., the Virtual Patch Panel) to the more well known (e.g., automating L2 extension to support ease of VM migration). As described by [Greg Ferro](#) the basic idea is to separate the traditionally monolithic switch platform into three separate constructs known as the Data Plane, the Control Plane and the Management Plane. As shown below, once this is done, the Management and Control plane elements can span multiple Data Plane elements thereby allowing them to be programmed based on the Management


[Erik Smith](#)
[Follow](#)

[ARCHIVES](#)

SEARCH

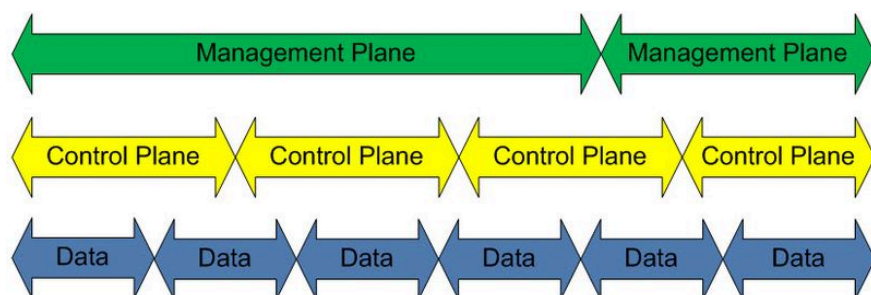
[CATEGORIES](#)

[Subscribe to this blog's feed](#)

DISCLAIMER

This is not an official EMC blog.
The opinions expressed here are my personal opinions.
Content published here is not read or approved in advance by EMC and does not necessarily reflect the views and opinions of EMC nor does it constitute any official communication of EMC.

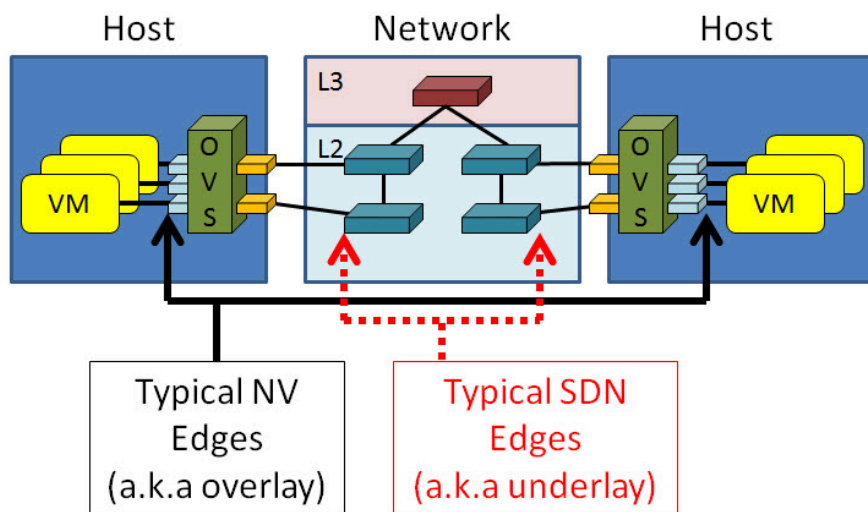
and Control planes greater understanding of the end to end connectivity requirements.



BTW, as described in David Black's [nvo3 BoF presentation](#) these relationships present some very interesting challenges to the management applications that will operate in this space.

Network Virtualization and Software Defined Networking

If the diagram above was a little bit too abstract for you, the diagram below should make the relationship between Network Virtualization (NV) and Software Defined Networking (SDN) a bit easier to understand. Please understand, because I am trying to make the topic as approachable as possible, I'm using a very simple topology. As a result, some of the options, such as a device that participates in both the underlay and overlay (e.g., a NV gateway) is not being shown. I'll save that for a later post.



The Underlay Network

Before you can use a Network Virtualization Overlay(NVO), you need to have network connectivity between the NV edges. This connectivity is provided by the underlay network and today can be provided by anything from a L2 LAN segment (e.g., a single L2 switch, or Ethernet Fabric such as Brocade VCS or Juniper's QFabric) to a L3 ECMP based network. Whatever you choose, the underlay must be capable of providing an adequate amount of bandwidth to support the bandwidth requirements of the overlay. These bandwidth requirements can be met by either statically configuring the underlay to provide the appropriate bandwidth or could be dynamically met by using an SDN approach to steer flows over links that provide an adequate amount of bandwidth. This steering function could be provided by the Management and Control Planes by using OpenFlow to configure the Data Plane.

I should point out that all of the work I have done with NV (so far) has used the "statically configured" underlay approach and this has worked fine for the very

simple Proof of Concept (PoC) work that I have been participating in. This is a roundabout way of saying that I haven't done much work with the "traditional?" SDN space nor tried to scale up the PoC configuration and as a result, I don't have a good fact-based opinion on how much of a need there will actually be for these SDN based steering functions.

That having been said, my advice to anyone trying to setup a NV based PoC would be to:

1. setup a statically configured underlay. I'm using an L2 topology.
2. Verify end to end connectivity between all of the NV Edges, ping works fine..
3. Once items 1 and 2 have been completed, mentally wrap the underlay network in a box, stick the box under your desk and forget about it!
4. Once the [overlay network](#) is up and you are sending and receiving packets via the overlay, then you can worry about optimizing the underlay.

My point is you can and probably should (at least initially) treat the underlay and overlay as different entities and avoid being overwhelmed by the complexity of trying to configure both at the same time.

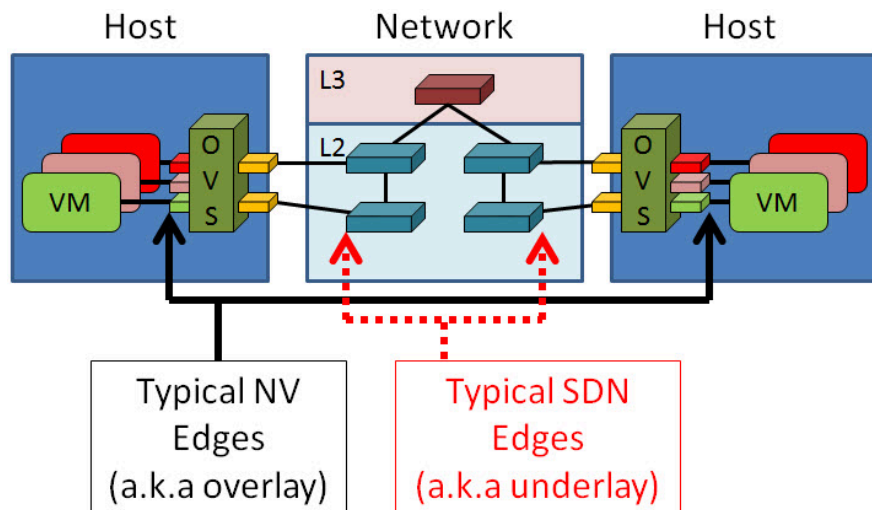
The Overlay Network

The Overlay Network I have been using is STT based. The Management and Control Plane functionality is being provided by Nicira's Network Virtualization Platform (NVP) and the Data Plane is Open vSwitch based.

Before I go any further, it's important to note that I gained access to the NVP software for free as a part of a PoC that I'm merely participating in.

When I was first introduced to NVP, I had a hard time understanding some of the basic NV concepts and terminology, so I am going to focus on those aspects for now. To start with, based on some work that I've done with VMware's vCloud Director recently, I think if you understand vCDNI or even just the concept of a distributed vSwitch, you'll have a pretty easy time understanding the type of functionality that NVP enables.

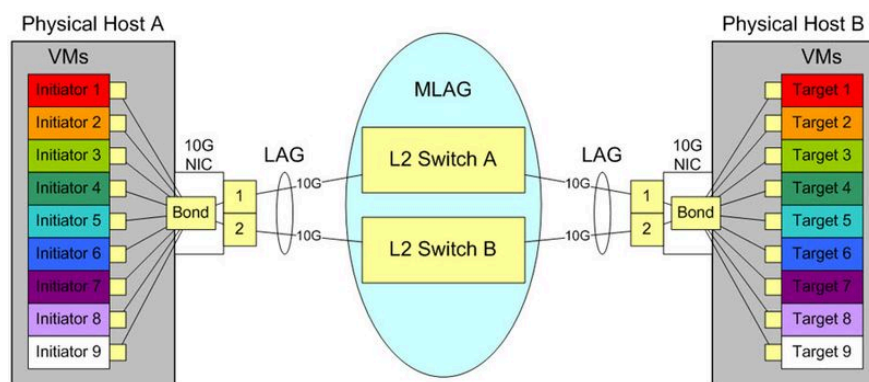
At a high level, NVP allows you to create distributed vSwitches (referred to as Transport Zones in NVP) that you can then attach tenants to. A tenant is simply a group of network devices that only have visibility to other network devices that also belong to that tenant. In NVP, tenants are attached to Virtual Interfaces on one or more Logical Switches. Each Logical Switch is associated with a Transport Zone. The easiest way to visualize a tenant is to think of each one as a different company that is utilizing the same physical infrastructure as another company. To make these relationships a bit less abstract, refer to the following diagram that is a slightly modified version of the one I provided above.



Note that both the VMs and the ports they connect to on the OVS instance have been color coded. Each color is intended to represent a different tenant or company. Obviously these different companies will want isolation from one another and the tenant concept allows for this and allows for them to also be managed separately. From a protocol perspective, this isolation is enforced by including a tenant ID in the header of each frame passed between the NV edges. Regardless of the tenant ID, these frames can use some or all of the links between the two OVS instances shown.

The tenant ID can be something as simple as a VLAN tag, MPLS label or be part of an encapsulation format that has been specifically designed to support NV such as [NVGRE](#), [VXLAN](#) or [STT](#). Each of these approaches has their pros and cons and are being actively debated in the IETF working group called [nvo3](#). As I mentioned previously, we're using STT and the performance results have been impressive. For a interesting performance comparison of the different encapsulation methods, see [this post](#) by Martin Cassado. During the PoC we also performed some basic performance testing with STT and it supports the data that Martin provides in his post.

More specifically, we created a configuration similar to the Physical topology shown below.



Physical Topology

The physical topology consisted of a couple of hosts that each had a single dual port 10GbE NIC. Each NIC was connected to a different 10G Ethernet switch.

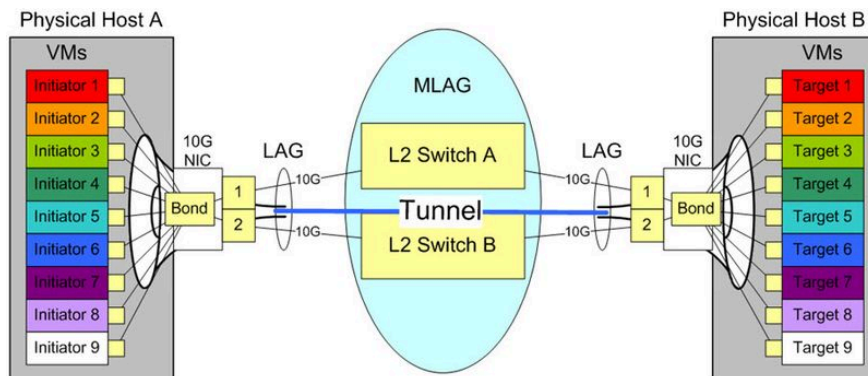
The Ethernet switches were configured to be a part of the same MLAG and this allowed us to setup and use bonding on the hosts.

We then created a total of 9 “VMs” on each Physical host and loaded NetPerf onto both the Initiator and Target VMs. I am intentionally being vague about the VM type we used but I'll tell you it wasn't VMware based.

Logical Topology

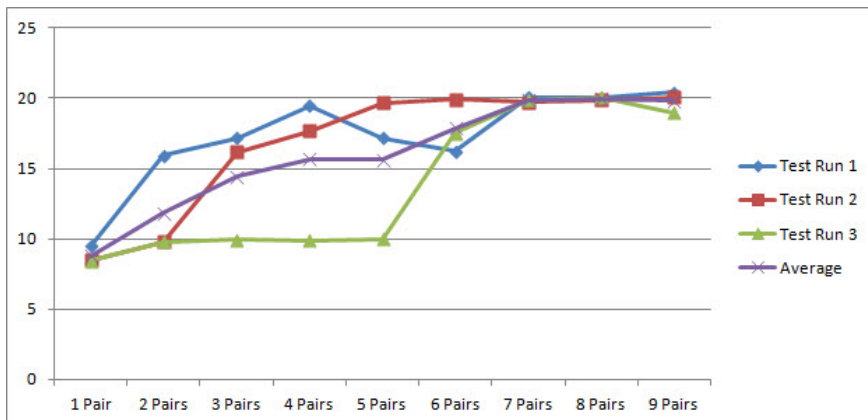
Once the physical Topology was configured I mentally wrapped it in a box, stuck it under my desk and forgot about it! :-)

Next we used NVP to create the Transport Zone (STT based) and then added each VM to a Logical Switch as a member of the same tenant. The Logical Topology that resulted from this work is shown below.



Again, I am not showing all of the detail in the diagram above, but the addition of a tunnel between the two hosts should give you a good idea of the concepts involved.

Finally we ran a series of tests and created the following table based on these results



The Y-axis is in Gbps and represents the throughput observed coming out of the adapter. These numbers were generated by NetPerf but were verified by using an XGIG analyzer. The X-axis represents the number of pairs of VMs that were running traffic during the test.

Note that a single pair of VMs was able to generate about 8Gbps and that as we scaled this up, the amount of bandwidth consumed approached the maximum available bandwidth of 20Gbps.

Also note that test 3 had an interesting result. I believe this was the result of multiple flows being stacked on the same physical link while the other link remained under utilized. Perhaps this is one of those areas where SDN could help.

Also note that while running with 9 pairs on test run 1, we apparently achieve greater than 20Gbps and this was the result of my manual (staggered start) testing process and the short duration of the tests.

Finally I am just including this data to help solidify some of the concepts that were introduced. Since the PoC is still underway, I am not able to share any additional detail about the configuration used or the parameters used, perhaps in future post..

Wrapping up

So that's NV in a nutshell. In future posts I'll be digging into some of the concepts in a bit more detail but hopefully this gets you off to a good start as you begin investigating NV for yourself.

Thanks for reading!

Posted at 10:36 AM | [Permalink](#)



Comments

You can follow this conversation by subscribing to the [comment feed](#) for this post.



Hi Eric,

Thanks for posting this. I like the way you build this up from simple constructs.

BTW, you mentioned for test 3:

"Also note that test 3 had an interesting result. I believe this was the result of multiple flows being stacked on the same physical link while the other link remained under utilized. Perhaps this is one of those areas where SDN could help."

I wanted to note that this is one good reason why the Brocade VCS Fabric has the Brocade ISL Trunk. The B-ISL Trunk automatically eliminates "hot spots" found in MLAG/LAG that are created by hashing techniques where an entire flow has to go on a physical link. Instead, B-ISL trunks uses frame stiping in hardware so all flows are uniformly distributed across all physical links. This means no hot spots and very high trunk utilization and its automatically configured so you don't have to program OpenFlow or an equivalent :-)

Thanks for posting this content. It's very useful.

Posted by: [BRCDbrems](#) | [09/21/2012 at 07:14 PM](#)



Hi Brook, thanks! Glad you found it useful..

I completely agree that the underlay network could provide significant value add in these kinds of environments. BTW, based on some test results that I've seen, Brocade's frame based trunking approach has proven to allow for greater utilization than any of the other standards based approaches. As long as the customer is using VCS end to end, I can see value in following this kind of approach.

One problem (that I am actually working on right now) has to do with fully utilizing a bond when there is only a single Tunnel End Point on each physical server. Due to the way that Bonding/teaming is done, all of the traffic in a single tunnel will end up utilizing the same physical link. Does Brocade have a solution for this particular problem?

Regards, Erik

Posted by: [Erik](#) | [09/25/2012 at 10:46 AM](#)



Hey there I am so thrilled I found your site, I really found you by accident, while I was browsing on Askjeeve for something else, Regardless I am here now and would just like to say kudos for a fantastic post and a all round thrilling blog (I also love the theme/design), I don't have time to read it all at the minute but I have saved it and also included your RSS feeds, so when I have time I will be back to read a lot more, Please do keep up the superb work.

Posted by: [Hank](#) | [07/22/2013 at 07:31 AM](#)

The comments to this entry are closed.

[Brass Tacks](#) [Powered by TypePad](#)