vNOX: VMware Network Operating System for SDDC

Rahul Kulkarni, Anupam Chanda
VMware Research and Development

Abstract

Network Virtualization at the Edge Hypervisor, Network Overlay Model and Software based Network Services have been proven to work for virtualized workloads and the SDDC paradigm. There is also an opportunity at VMware to apply SDN principles to the datacenter physical fabric underlay by building VMware networking solutions for management and control of the underlay for Agility, Scale and Quality of Service. This entails us to provide a holistic network service offering which includes both the physical and virtual networking components. A key enabler for this is the availability of commodity programmable white box network hardware for building a physical network fabric in a datacenter. In this paper we propose building SDN solutions using commodity network devices integrated with the VMware SDDC stack. The benefits include: ability to develop a differentiated network service offering for vCHS customers, an opportunity to capture a still nascent business for white box based network fabric management, technical advantage in having both physical and virtual networking solutions provided by the VMware SDDC software stack, and increased feature velocity by being able to drive innovation in the ecosystem by controlling the commodity network fabric. In this paper we provide a blueprint for vNOX (VMware Network Operating System), which describes a holistic solution for managing and controlling a datacenter network. We believe that VMware solutions such as vCHS, vRack, MARVIN would immediately benefit from the vNOX solution

1. Introduction

The networking industry is at an inflection point where the SDN model has fundamentally impacted the way virtual networks are provisioned, managed and consumed. VMware is leading that charge with NSX and its model for network virtualization. A similar force is rapidly taking shape for physical networks at different layers: open white box network switches as opposed to closed box proprietary switches, open SDN operating system running on the white boxes as opposed to closed vendor operating systems, SDN controllers managing an integrated physical and virtual network fabric as opposed to multi-vendor control, Cloud Management Systems managing networking for both physical and virtual infrastructure riding on such SDN controllers. Other industry initiatives like the Open Compute Project [1] and disaggregated hardware models such as the Intel Rack Scale Architecture (RSA) [2] are redefining physical networking in the SDN context. Open Source projects like OpenStack Neutron [3], and Open Daylight [4] using SDN vendor plugins/controllers are already controlling both physical and virtual network infrastructure together as an integrated fabric. On the Cloud platform front Google's Andromeda [5] is by far the most sophisticated and impressive offering for their cloud platform network services. The Andromeda architecture is based on a holistic physical and virtual networking approach towards offering a cloud network service.

As the networking landscape is getting redefined, it is imperative and critical for VMware to influence and establish leadership in defining an SDN platform for the emerging generation of datacenter physical networks.

In this proposal, we outline an architecture named vNOX, which can control and manage the data center physical network fabric. We propose two approaches for vNOX. First, a complete ground up approach in that VMware provides all the components for management, control, data and service planes of the physical fabric. Second, a hybrid approach where vNOX leverages the dataplane elements from external vendors and integrates with the VMware management, control and service plane software.

Large IaaS vendors such as Google and Amazon [6] have already embraced such models to reduce the management complexity and the cost inherent in proprietary networking equipment. If VMware offerings like vCHS and the ecosystem do not have such CAPEX and OPEX reduction opportunities, our customers (and VMware by extension) will be at an inherent disadvantage against IaaS providers.

2. Vision

vNOX is a virtual and physical network operating system for SDDC built on a commodity network fabric. The key components to realize the vNOX vision are a Central vNOX Controller to control and manage the physical fabric, and a vNOX aware operating system (VMware/Vendor) running on programmable network devices for data plane services integrated with such a controller. One can envision a future where we have NSX augmented with vNOX for managing and controlling both the underlay and the overlay network fabric.

To realize the vNOX vision here are our guiding principles:

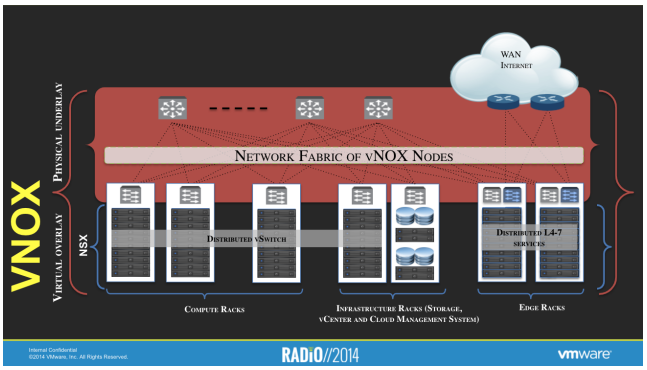
- ***Innovate*** to meet Application Service Level Objectives:
  - Offer Network as a Service (vCHS) with QoS guarantees for latency and throughput using a central Traffic Engineering engine.
  - Balance physical network flows and workload placements for optimized network usage.
  - Enable Distributed Resource Scheduler and Network IO control algorithms to work in conjunction with the intelligence from the underlay and overlay network fabric.

- ***Provide*** a unified OAM view of the virtual and the physical network:
  - Powerful operational insights due to visibility into both the underlay and overlay by correlating Application (V) and Tunnel (P) flows.
  - Analytics and data mining for traffic patterns using metadata in tunnel encapsulation for operational insights.
  - Identical management APIs and tools for provisioning both physical & virtual networks.

- ***Extend*** NSX services/functions on the physical fabric:
  - Dynamic load balancing in vNOX nodes front-ending an elastic NSX services cluster.
  - Distributed layer 3 routing on network devices.

- Tunnel termination for on-ramp/off-ramp gateway functions, IDS/IPS, virus scanners for external Internet facing traffic.
- Compute intensive cryptographic services, which can be leveraged at edge VPN gateways using hardware offloads.

The vNOX vision of a unified network software platform for both physical and virtual network elements can take out the barriers to enterprise and cloud provider adoption of white box clouds. With extensions to NSX, vCAC, vCOPS to manage the datacenter network fabric, VMware can provide a seamless end-to-end solution as compared to fragmented multi vendor offerings from OpenStack and other commodity cloud solutions. We can neutralize "specialized" hardware centric solutions from network vendors like Cisco/Insieme by providing a software-controlled data center for physical and virtual network elements using commodity hardware.



The major enabler for the vNOX vision is the emergence of network white box reference designs for top-of-rack (ToR) and Aggregation switches (leaf and spine switches) from major vendors like Broadcom, Intel and others [7] which has led to ODMs like Quanta manufacturing switches based on such designs.

- The white box consists of an x86 based System on Chip control card and an integrated switching silicon in multiple form factors (combinations of 64 to 128 10G/40G ports) and optionally data plane acceleration chips for security and packet processing for L4-7 services.
- Future roadmap entails stitching together multichip form factors for increased port density, enhanced programmable capabilities, etc.

Other industry forces driving in this direction are initiatives like the Open Compute initiative from Facebook to provide open network specifications for network hardware and software. An interesting side note from OCP networking software charter indicates an interest in a switch hypervisor [8].

Execution towards this requires working on multiple fronts. We need to work with chip vendors and ODMs to develop white boxes to VMware specifications. The types of devices can be bucketed in two categories: ToR/fabric switches, and edge gateway switches for services. The first category of switches in current deployments has 64-128 10/40 Gb ports with low compute requirements and basic switching/routing capabilities and the second category of edge switches which can be used to run NSX edge services can have additional compute horsepower [e.g., 16 core 2.5GHz CPUs] with high PCIe IO bandwidth to the CPU and additional dataplane packet processing chips for L4-7 services for high throughput/low latency packet processing. These switches can be viewed as general-purpose packet processing engines for software network services leveraging hardware offloads for compute intensive workloads. These are not to be viewed as proprietary middle boxes but rather off-the-shelf packet processing chip engines to assist with the software services deployed on a scale-out cluster to avoid choke points at on-ramp/off-ramp, cloud and service gateways.

3. vNOX Design

In this section we present a design of the vNOX operating system for the datacenter physical fabric. The design is divided into four major components, which relate to the physical fabric control, management, data and service planes. We would like to point out that the vNOX integration for the two classes of devices (ToR/spine switches and edge switches) is different in that the ToR/spines provide network underlay connectivity (L2 and L3) to the hypervisors and the Edge Switches. The edge switches themselves participate in the overlay with the hypervisors and can

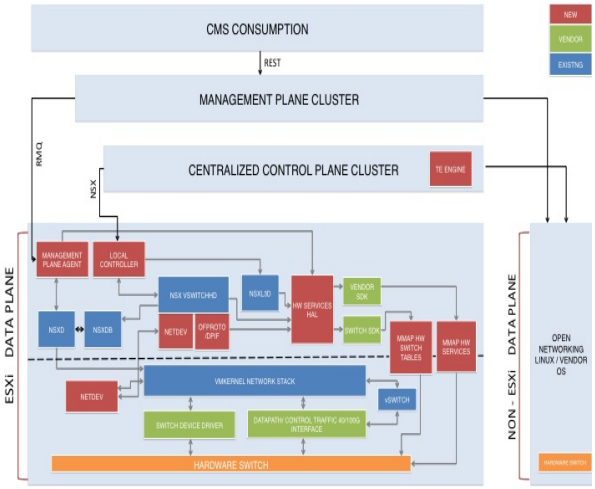


Fig.1 vNOX platform

provide services like VXLAN-VLAN termination, load balancing, VPN termination, etc. Given the different roles in the network of these device types, the implications to the vNOX integration will slightly vary.

a) Control Plane [CP]

vNOX proposes two approaches for control plane design for the physical fabric. First, a traditional CP and hybrid/decoupled CP approach. Traditional switch/router control planes are tightly coupled to individual boxes in that all the control protocols for Layer 2, Layer 3 and other state for ACLs, VLANs, etc. are running on the control processor and maintain state. Network virtualization has helped abstract out a lot of the state and simplified physical fabric configurations. But, there is still control plane state tied to the physical box for routing/switching control protocols, QoS, etc. The decoupled or hybrid approach is to separate the control elements like routing protocols from the physical box and manage the forwarding and QoS logic from a control cluster essentially rendering the physical fabric elements as forwarding entities only [line cards in a logical DC Chassis]. There is also a lack of visibility to tie-in logical and physical network requirements for Quality of Service guarantees. ECMP, DSCP, rate limiting, etc. is best effort and is hard to adapt to dynamic environments for network SLA guarantees. SDN controllers for the physical fabric are essentially built around the decoupled control plane principles.

The vNOX architecture proposes two key approaches in this aspect: A scale out **Decoupled Control Plane** and a **Traffic Engineering (TE) Engine** for the physical fabric.

i. **Decoupled Control Plane:**

Simplifying the control plane on the ToR and Spine switches will lead to a manageable dataplane fabric with forwarding decisions made by a Central Controller similar to an SDN controller. The switches would still need some bootstrapping software for topology discovery/changes and a management network for reachability but that does not preclude us from decoupling the rest of the aspects of the control plane from the physical box like BGP/OSPF, etc. Similar in principle to the NSX edge architecture for vDR (virtual distributed router) wherein BGP peering is done on a scale out distributed system and the forwarding information is downloaded to individual FIBs of the virtual edges. A related Microsoft proposal presented at Nanog called BrainSlug [9] captures the essence of this very well. An identical architecture needs to be designed for the physical fabric elements wherein the decoupled control plane can then control the physical ToR/spines/edges in the physical dataplane for data forwarding and edge services. This simplification of software on individual boxes will reduce OPEX significantly.

ii. Traditional Control Plane:

The decoupled control plane model is a non-trivial problem to solve. The architecture makes a recommendation of a decoupled CP but does not mandate it. vNOX can also function with a traditional approach where control protocols for routing run within individual switch/routers.

iii. Traffic Engineering Engine [TEE]:

The TEE in the CCP is responsible for Traffic Engineering of the entire physical fabric consisting of ToRs, spines, and edge gateway switches in a DC (vCHS). The goal of the TE engine is to optimally utilize the physical network resources for the tenant VM workloads it is servicing. It does so by dynamically correlating virtual and physical network flows, identifying large flows using tunnel metadata and rebalancing them on the network elements for efficient usage. Since the CCP has end-to-end visibility into the entire physical and virtual network, the TE engine in the CCP can make intelligent forwarding decisions for SLA guarantees. Specifically correlating logical and physical (tunnel) flows can be extremely useful for providing application SLAs. This is difficult to achieve today with automation and monitoring tools. Some examples such as vMotion or transfer of large datasets in a vSAN cluster can be critical and would require guaranteed latency/bandwidth in the network. These scenarios can be monitored and serviced efficiently by a TE engine for the physical fabric in the CCP. Related work in this area can be found in Hadera [10]. Given VMware's vantage point in the Data Center, a TEE, which can optimise traffic flows by correlating VM traffic to physical flows for network performance and tenant SLOs, can be a key differentiator for offering network as a service.

iv. CP Integration:

vNOX devices that participate in the overlay network with the hypervisors, integrate at the control plane with the network controller which also controls the edge software switches. The merchant silicon on such a vNOX device has the ability to terminate VXLAN tunnels [11]. As such the data plane integration of these devices with other data plane elements in the SDDC (i.e., hypervisors, gateways, service nodes) would be with VXLAN tunnels.

For the control plane integration with the network controller, we plan to implement a local control plane (LCP) agent on the vNOX device. We define an interface between the central control plane (CCP) and the LCP agent. We have a couple of choices in designing this interface between CCP and LCP.

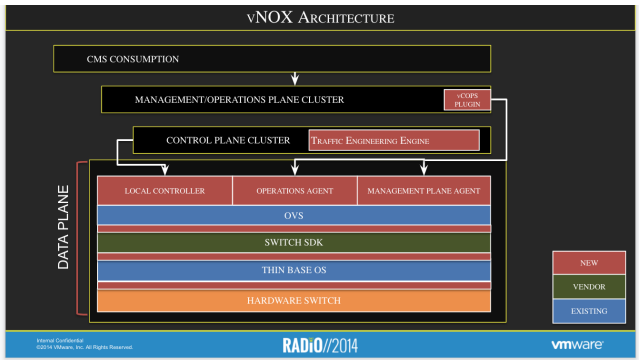


Fig 2: Architecture of a vNOX node

The first approach is to use the OVSDB protocol [12] to integrate CCP and LCP. In this approach we run an OVSDB server on the vNOX node, and the CCP sends/receives transactions to this OVSDB server. These transactions exchange control plane information between the CCP and the LCP. E.g., for a vNOX device that acts as a layer 2 gateway for a logical switch, such control plane information includes the MAC addresses of the VMs for the logical switch and their locations (the data plane IP addresses of the hypervisors on which such VMs are located), and the MAC addresses of the physical stations attached to the vNOX device. The MAC address and location of a VM enables the vNOX device to forward a packet from a physical station destined to the VM's MAC address to the hypervisor's VXLAN tunnel. Similarly, in the reverse direction, the hypervisor can forward a packet destined to the MAC address of the physical station to the VXLAN tunnel terminating on the vNOX device

This approach has the benefit of openness: it relies on the open OVSDB protocol, and an open OVSDB database schema that VMware has designed in partnership with its hardware partners. In VMworld 2013, several hardware partners (Arista, Brocade, Cumulus Networks, Dell, HP, and Juniper) demonstrated L2 gateway integration with the NSX controller using the said approach.

For the vNOX solution, the details of using the OVSDB mechanism are as follows. We will run the open-sourced ovsdb-server on the hardware device. The LCP would listen to changes to the database hosted by the ovsdb-server (notifications sent by the OVSDB protocol). In response to the changes, the LCP would program the local forwarding engine using the white box SDK. The SDK also provides notifications to the LCP on local forwarding entry changes/learnt state, etc. The LCP would publish relevant information to the OVSDB database for ovsdb-server to send update notifications to the CCP.

While the OVSDB based approach has the benefit of openness, it can inhibit feature velocity. Any modification to the database schema requires notifications to and acknowledgement from our hardware partners. So, we propose an alternate design for the vNOX nodes. In this design we would use a VMware-proprietary interface between the CCP and LCP. In essence, it would convey the same (or similar) information as with the OVSDB based design, but it provides the flexibility that we can arbitrarily update the interface between the CCP and the LCP. This flexibility can provide us with feature velocity and differentiators as compared to the devices and solutions from our hardware partners. Additionally, one concern with the OVSDB protocol is that it tends to be verbose, and can be a source of a lot of notifications to its client. This is a concern for the scalability of the solution. A proprietary in-house protocol is better suited to address this challenge and to make the solution more scalable. Finally, this approach gives us the ability to deliver advanced features without

necessarily divulging our intellectual property (unlike the OVSDB approach that relies on an open-sourced database schema).

The design details of the second approach are very similar to those of the OVSDB based approach. The only difference is the interface between the CCP and the LCP.

As a side note we would like to point out that for the above approach to work with an external vendor, the vendor has to provide the vNOX LCP the set of APIs required to program the dataplane and the vNOX LCP has to be ported to run on the vendor switch operating system. An alternative way would be to use a proprietary OVSDB extended schema

b) Services Plane

As prescribed earlier the vNOX platform recommends edge switches to have additional computing horsepower and much richer hardware capabilities for supporting dataplane processing. As a reference the Broadcom OCP specification calls out support for a dataplane processor module that is used to provide acceleration for cryptographic dataplane functions, deep packet inspection, and modification capabilities. Intel DPDK software also has demonstrated fast dataplane processing on commodity x86 processors. For high throughput choke points e.g., multiple 40G-100G (future) North-South interfaces in the datacenter, the use cases for a commodity data plane packet processing module (e.g., Broadcom XLP432, Cavium OCTEON) augmented to a commodity switch can be justified and can perform better than an x86 at line rate speeds. There is also an NSX effort underway for a bare metal services platform for high performance gateway services. The Edge Switch platform can also be viewed as a "bare metal server" for hosting such edge services.

The edge commodity switches can be inserted into the network using the Service Insertion platform (vSIP) framework in NSX, which integrates third party services. These edge switches will be treated as physical appliance based services: these physical appliances are commodity switches running VMware services bare metal or in a VM. The edge switch services will subscribe to the service discovery model for vSIP and participate in both edge based service transport (L2) and cloud based transport (L3) called out in the vSIP framework specification.

The vNOX node implements a Service Plane Agent (SPA) for integration into the vSIP and provides services like Layer 2 gateway service, Layer 3 gateway service, and Layer 4-7 services (e.g., load balancer), security services, etc. which can be offloaded to additional dataplane hardware. The SPA in turn integrates with the Hardware Abstraction Layer (HAL) on the vNOX node, which interfaces with the underlying hardware device SDK.

c) Management and Operations

i. **Management Plane:**

Network Devices of both types (overlay and underlay) would benefit by having management plane integration with the vSphere stack. The design involves integrating with a centralized management plane cluster (MP) via a management plane agent (MPA) running on the vNOX node. In current deployments of ESX hosts, MP and MPA communicate over a message bus (RabbitMQ). Our design involves leveraging this existing implementation, and integrating the white box devices over RabbitMQ message bus.

The MPA exposes the ports on the switch as first class objects, and implements a number of interfaces that the MP can use to control the switch. Some examples of such interfaces are the following:

- Marking ports administratively up or down.
- Configuring ports in trunk or access mode.
- Managing vlan configurations of ports.
- Configuring ports in LAG mode (for HA).
- Enabling/disabling STP on ports.
- Configuring LACP, BFD or other link monitoring protocols.
- Configuring QoS properties on ports.
- Configuring routing.

For edge devices the management plane integration involves registering the services with the vSIP framework. This allows the management plane cluster to configure, manage and monitor these services.

The requirement here is for the MPA to run on the vNOX node. For whiteboxes running VMware OS we intend to develop the MPA to run on the vNOX node, third party vendors need to port the MPA on their switch operating systems and integrate it with the underlying management APIs for the switch

ii. **Operations:**

For operations purposes a vNOX node runs agents that integrate with vSphere's centralized operations and management collector framework, e.g., integrating with vCOPS. These devices can report real time traffic patterns, and flow monitoring. This would allow us to present statistics like link and fabric utilization (physical network utilization), as well as logical flow monitoring akin to waypoints in the network (e.g., a given logical flow goes through which physical elements). Additionally, identifying hot spots in the physical network fabric would be possible because of this integration. Further, the datasets collected in the network fabric could be fed into machine learning algorithms to predict network usage and what-if scenarios. This will result in powerful operational insights of data center network resources

d) DataPlane

The data plane operating system running on the vNOX nodes could be one of the three choices: VMware provided OS, open source OS like Open Networking Linux, or a vendor provided OS. The flexibility, control and IP over the software may diminish from the former to the later. The pros and cons of these various approaches are discussed in the next section. The rest of this section describes the ESXi-based dataplane design for an Open vSwitch based hardware switch. Most of the principles should be similar for other Operating Systems.

The three key components of a hardware switch are: 1. OS running on the switch control processor, 2. Hardware switch SDK and a driver and 3. A switch abstraction in software to expose the properties of a network device (ports, VLAN, L2/L3 protocols, ACLs, QoS).

For an ESXi based vNOX node, ESXi is ported to run on the specific x86 based switch control processor platform. ESXi then needs the switch SDK to be compiled and integrated into the OS. The hardware switch SDK has two components: a kernel component, which is a driver for the switch, and a userworld component, where the bulk of the intelligence for switch programmability resides. For example the SDK exposes APIs for programming the Layer2/Layer3 TCAM/BST tables, flow tables, port management, etc. During ESXi boot the switch driver discovers and exposes a PCIe endpoint to the userworld SDK and memory maps the device memory. The SDK goes through an initialization sequence for the switch to bring it out of reset, load the required firmware, initialize the ASIC, setup the physical ports, initialize the various hardware tables and complete the boot sequence and hand over the switch for consumption. The driver will be ported in VMkernel and the SDK ported and compiled into the ESXi user library. Both the driver and the SDK have an OS abstraction layer for integration, this layer has to be integrated into vmklinux for the driver and ESXi userworld APIs for the user SDK. VMKernel network layer will treat the network ports of the switch (e.g., 48x10 Gbps, 4x40 Gbps) as uplink ports similar to NICs ports. Open vSwitch will be integrated into the vendor SDK via a hardware abstraction layer (HAL). The control path component of Open vSwitch (netdev) will be developed to talk to the VMKernel uplink layer, which then interfaces with the driver

to manage the physical ports. The data path component of OVS (ofproto) will be implemented and then integrated into the HAL, which in turn interfaces with the vendor SDK for consumption of the switch hardware resources. The control traffic interface from the hardware switch (40-100G) is treated as an uplink into a vSwitch in VMKernel, which can send and receive control or services traffic to the VMs or userworld applications. OVSDB and the LCP interface to the controller are used for the control plane integration. MPA is used to interface to the management plane cluster.

Operating system choices for DataPlane

A key underpinning of vNOX is the operating system and software for the network white boxes to be integrated with the rest of the VMware SDDC/NSX stack. The two obvious choices are open source Linux and ESXi. A key advantage with running Linux is that the switch vendor SDKs for commodity switches are ported and well tested under Linux with little effort for bringup. There is also a recent open source effort called Open Network Linux [13] under the OCP charter, which provides a base OS for commodity bare metal platforms (without any switching functions). It still requires a vendor switch SDK to be integrated either in a binary format or compiled in. The goal of OCP and ONL is take a commodity switch and integrate it with an open source software stack: ONIE [14] for upgrades and boot, ONL for basic OS, vendor SDK on top, and finally shrink-wrap additional software differentiators on top of this stack and you have a switch platform. Given this background Linux would definitely be a great advantage to start off with. There is also an additional advantage of Linux to be able to run on non-x86 control processors. The downside is that these open source efforts are still at a very early stage and immature in nature. An additional concern for VMware would be the Linux integration into the rest of the northbound components, which are specific to ESXi and any additional intelligence we add in the VMKernel Network Stack for services such as vMotion, etc.

The ESXi route for dataplane elements would require us to port such vendor SDKs. It would also require us to bring up the switching silicon under an ESXi environment akin to a vmklinux driver albeit somewhat more complex. A key advantage of ESXi is that we could run VMware network services, e.g., Layer 3 routing, load balancers, security services, and utilize the hardware capabilities of the switch silicon. Another advantage would be easier integration with the management stack, which would be a challenge with a Linux OS. A third advantage is to run any special purpose virtual machines for telemetry, intrusion detection/prevention, etc. But, given the roadmap of a multi hypervisor story for NSX the advantages of ESXi could be less compelling.

The vNOX architecture does not make a clear recommendation on the OS choice. For the PoC effort we will be using ESXi and any learning based on that will determine the best OS choice for the dataplane elements.

In the context of ESXi dataplane performance requirements for a hardware switch, we would like to point out that the forwarding engines for network switches are completely in hardware. E.g., a 64 port x 10 Gigabit switch will forward traffic completely in hardware based on the programmed switch tables. In addition any layer 4/7 network services will utilize the hardware capabilities for packet processing in lieu of software only services. E.g., a load balancer service could program the switch hardware to load balance across a set of ports based on rule matches in the packet header. The network traffic coming into the control processor, which runs the network stack is usually only control traffic (e.g., protocol traffic for OSPF) from the PCIe control interface usually in the range of 40-100G in current switches. The control processors on the switches are fully capable of such workloads and the CPU specifications can be customised based on requirements.

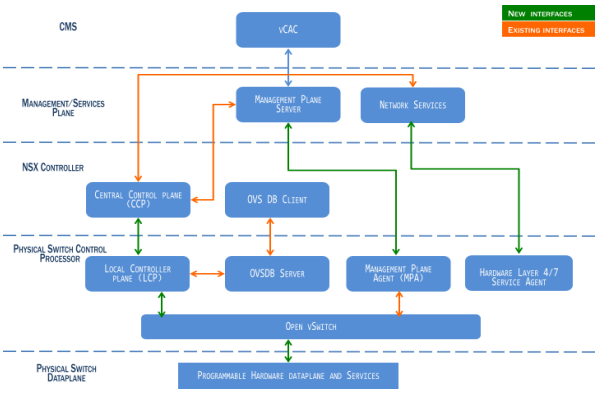


Fig.3 vNOX abstract view

4. Use cases

The initial use cases targeted for vNOX are internal to VMware. We are actively engaged with the vCHS datacenter architecture team to understand their use cases.

vCHS motivation for vNOX is driven primarily based on:

- Deploying a network fabric with low cost network hardware.
- Providing Network as a Service that can meet application SLO.
- Single pane of glass for Physical And Virtual Networks.
- Operational insights for the network fabric.
- Tenant metering and usage.
- Having the ability to manage and provision the network hardware with the same set of VMware tools used for SDDC.
- Ability to drive software requirements without vendor dependency.
- Infeasibility of a switch vendor to provide/integrate a certain feature available in a NSX/SDDC suite, e.g., vCOPS, vCAC.

5. Implementation Plan

We are developing the vNOX platform for multivendor white box switches mainly Broadcom and Intel. The initial PoC milestone is to get a vNOX based network switch built on ESXi and Open vSwitch integrated with the switch SDK. The PoC [15] is being developed based on an Intel platform called SeaCliffTrail [16] which uses the Fulcrum switching chipset.

Near Term Goals:

- ESXi based vNOX node running on Intel SeaCliffTrail Platform controlling the hardware switch.
- Ability to switch VM traffic connected to the vNOX node based on specific flows/rules added from CLI.
- vNOX node integrated with the NSX controller.

6. Conclusion

A datacenter operating system should have a universal view of the physical and virtual network elements to deliver a comprehensive solution. It is critical for vCHS to offer highly agile, resilient networking services to customers, which are built upon low cost commodity network hardware and VMware software. We believe with vNOX we can build that service. A holistic yet decoupled vNOX platform for data center physical and virtual networks will pave the way for the next generation of intelligent network fabrics for SDDC.

Acknowledgments

We would like to acknowledge Madhukar Krishnarao, Bernie Haverkamp and Aishwarya Sohoni for working on the PoC. For valuable inputs and suggestions we would like to acknowledge Shoby Cherian, T.Sridhar, and Rob Mills from EE; Joe Carvalho, Rob Nafus, Jason Lochhead, and Matt Probst from the vCHS architecture team; Duncan Epping from the Marvin engineering team; Saman Amarsinghe from the CTO office; Ben Pfaff for shepherding the paper and Ben Basler, Andrew Lambeth, Serge Maskalik, Radha Popuri, Romain Lenglet, Rajiv Krishnamurthy, Rajiv Ramanathan, Ganesan ChandraShekhar, Ram Singh, and Guolin Yang from NSBU for their feedback. Raj Yavatkar from the vRack team.

References

1. http://opencompute.org/projects/networking/
2. https://intel.activeevents.com/sf13/connect/.../SF13_CLDS001_100.pdf
3. https://wiki.openstack.org/wiki/Neutron
4. http://www.opendaylight.org/
5. http://googlecloudplatform.blogspot.com/2014/04/enter-andromeda-zone-google-cloud-platforms-latest-networking-stack.html
6. http://www.theregister.co.uk/2013/11/18/cisco_cloud_problem/
7. http://www.opencompute.org/wiki/Networking/SpecsAndDesigns
8. http://opencompute.org/assets/Uploads/OCP-Network-Project-Charter- V1-12.pdf
9. http://www.nanog.org/sites/default/files/wed.general.brainslug.lapukhov.20.pdf
10. https://www.usenix.org/legacy/event/nsdi10/tech/full_papers/al-fares.pdf
11. http://tools.ietf.org/html/draft-mahalingam-dutt-dcops-vxlan-00
12. http://tools.ietf.org/html/draft-pfaff-ovsdb-proto-02
13. http://opennetlinux.org/
14. http://www.onie.org/
15. https://wiki.eng.vmware.com/EToR
16. http://www.adiengineering.com/products/seacliff-trail
17. http://www.sdncentral.com/news/white-box-week-enterprise-might-care/2013/11/
18. http://cseweb.ucsd.edu/~vahdat/papers/scale-out-micro10.pdf