Software-Defined Networking and Open Networking

Understanding Foundational Concepts and Constructs v1.1

Victor Lama
Dell Network Solutions
Enterprise Campus and Data Center
G500 Banking & Securities

Date	Version	Description
07-11-2016	1.0	Initial Release
08-01-2016	1.1	Minor font and syntax changes.

Table of Contents

Introduction

The last several years have a seen a sea-change of innovation in data center networking. An entirely new lexicon has emerged that includes terms such as *software-defined*, *underlays*, *overlays*, *programmatic control* and *centralized controllers*. They fall under the rubric of what is termed Software Defined Networking (SDN) and they represent a major paradigm shift regarding the manner in which networks are considered and deployed.

Without an understanding of some basic SDN concepts, appreciating the significance of these developments can present a challenge to even the most seasoned IT professionals. In fact, even networking professionals who don't understand the architectural implications of SDN and do not have a background in fundamental software constructs may find themselves overwhelmed by the ecosystem of software-based networking solutions, where they fit into the new architecture, and the corresponding jargon that is used to describe their functionality.

The goal of this white paper is to begin the process of demystification by taking a taxonomic approach to understanding the *lingua franca* of the world of SDN and Open Networking, especially as it pertains to Dell. The reader can then expand that knowledge through further reading.

For ease of reading, the information in this paper is presented in a question-and-answer format. Some questions are designed to help seasoned networkers refine their knowledge, while others are meant for IT generalists, who are intellectually curious and seek to broaden their horizons. To ensure that non-networking focused IT professionals can appreciate certain architectural principles associated with SDN, some basic legacy networking concepts will also be touched upon.

Section 1.0 of this paper offers some general knowledge regarding SDN and Dell's Open Networking initiatives. Terms and phrases that appear with relative ubiquity in the course of SDN discussions are referenced and deconstructed in this section. Section 2.0 focuses on specific vendor offerings from Dell's third-party ecosystem of Open Networking partners.

Note: This white paper may be updated periodically. It is advisable to look for the latest version.

1.0 General SDN and Open Networking Concepts

The following section is an FAQ-formatted overview of some of the most important foundational concepts around SDN and Dell Open Networking, their relationship to each other, the architectural constructs they exploit, the innovations they offer, and the lexicon used by computer scientists and vendors to describe all of the above. This section will also cover some general networking concepts.

1.1 What is SDN?

For the last 25 years, TCP/IP networks have been built using well-known distributed communication protocols, such as the Spanning Tree Protocol and the Routing Information Protocol. As Professor Scott Shenker, one of SDN's pioneers, explains, without the necessary software abstractions in place, computer scientists have been forced to "master the complexity" of writing distributed algorithms on vertically integrated packet forwarders. SDN addresses these shortcomings by centralizing control and providing a standardized interface to configure network forwarding state in a dynamic and programmatic fashion. Stated otherwise, SDN is about creating programmable networks.

This topic will be addressed in much more depth in subsequent questions.

1.2 What is Open Networking?

Within the context of Dell Networking, Open Networking refers to the disaggregation of a network switch's operating system (OS) from the underlying hardware. A switch that does not ship with a vertically integrated OS from the hardware vendor is commonly referred to as a bare-metal switch or a White Box. When the bare-metal switch is provided by an established network hardware vendor, like Dell, it is referred to as a Brite-box (a concatenation of *white* and *brand* name box).

In general, Brite-box vendors offer a "safer" approach to disaggregated networking. By choosing Dell, one can take advantage of a global footprint, a world-class supply chain, and a 24x7 follow-the-sun support model. As an industry pioneer in the disaggregated model, Dell offers operating systems from several different vendors that can be loaded onto a bare-metal Dell switch, such as Cumulus Linux, Big Switch Networks Switch Light or IP Infusion's OcNOS. Dell also offers its latest Linux-based OS10 operating system as part of its disaggregated model.

1.3 What exactly is disaggregation and why is this development in the networking industry significant?

The decoupling of an operating system from the underlying hardware is known as disaggregation. This allows network architects the flexibility to deploy the operating system and hardware platform of their choice, independent of each other. This is similar to the manner in which an x86 server from Dell or HP can load an OS from multiple vendors, like Microsoft or Red Hat. The opposite of this would be a vertically integrated switch from vendors like Cisco Systems that come with the proprietary NX-OS software, or a Dell Networking switch with Dell OS9 loaded.

Disaggregation in networking has opened up the market to fierce competition among third-party software vendors whose objective is to bring differentiated solutions to market before the other. Competition breeds innovation and better economics for the consumer.

A software-driven network design also has its advantages in terms of agility and the ease with which it can be adapted to support future network demands. Since the network's persona (its forwarding and processing logic) is abstracted from the underlying hardware, the network can be redesigned by changing the software that defines it. Meanwhile, all the networking hardware can remain the same.

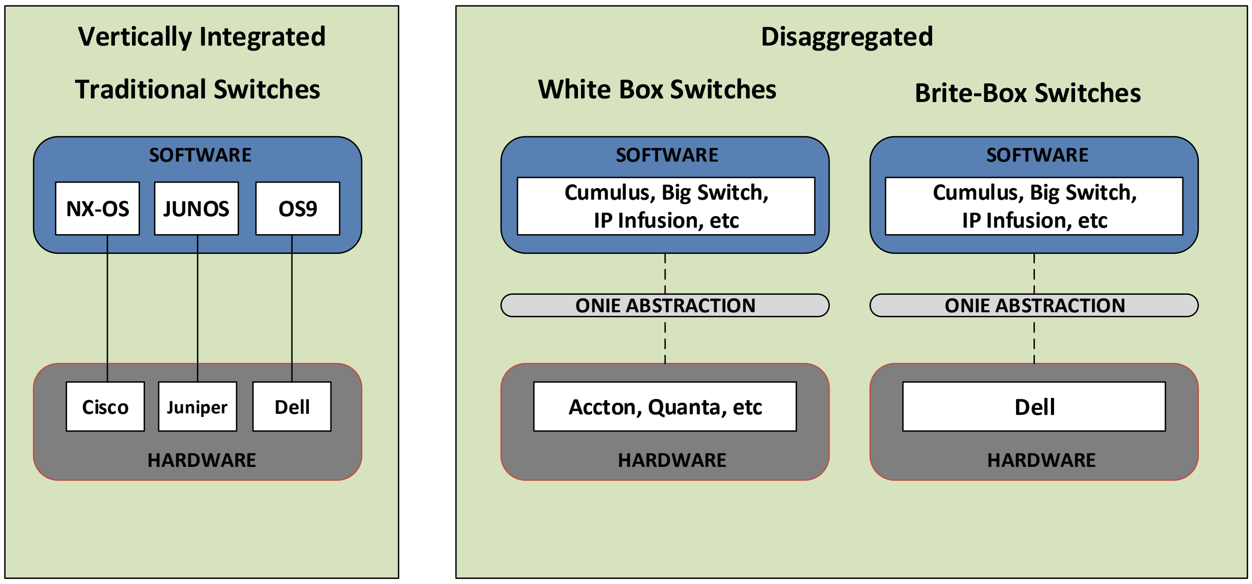


Figure 1-1 – Vertically Integrated and Disaggregated: White Box and Brite Box

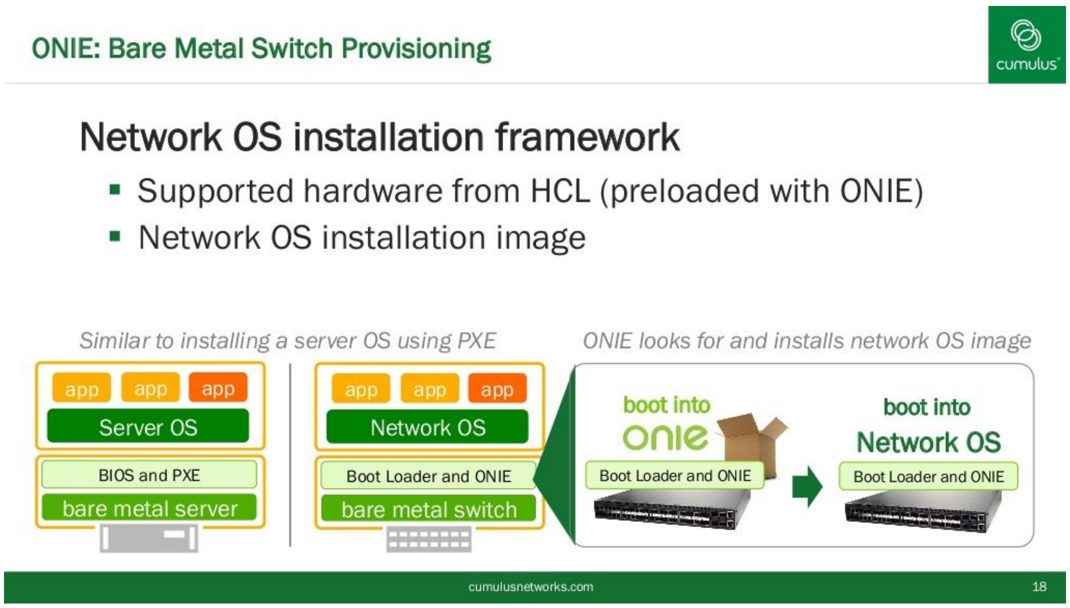
1.4 Is there any special requirement for installing a third party operating system on Dell Networking bare-metal switches?

Yes, a piece of Linux-based software called the Open Networking Install Environment (ONIE - pronounced "oh-nee") is required. This boot loader or "shim" sits between the network switching chipset and the operating system, thereby abstracting the underlying hardware. ONIE enables a bare-metal network switch ecosystem, where end users have a choice among different network operating systems from which to choose. Among the founding members of the ONIE project are Cumulus, Big Switch Networks, Dell and Broadcom. ONIE was contributed to the Open Compute Project in 2013.

All Dell Networking switches with the "ON" designation are equipped to leverage ONIE and load third party switch operating systems. Today, that includes 1, 10, 40, and 100G (multi-rate) switches that are all built using commoditized hardware from network chipset manufacturers, such as Broadcom.

1.5 How does ONIE work?

ONIE is a small operating system based on Linux that boots on a switch, discovers which network installer images are available (whether on a local network or locally stored), loads the image, and then provides an environment so that the installer can load the network OS onto the switch. This is illustrated in the figure below. Take note that this example is provided by Cumulus, but ONIE is leveraged for other third party OS solutions, too.



ONIE: Bare Metal Switch Provisioning

Network OS installation framework

- Supported hardware from HCL (preloaded with ONIE)
- Network OS installation image

Figure 1-2 – Open Networking Install Environment Behavior

1.6 So-called off-the-shelf "merchant silicon" features highly in discussions about Open Networking. What is it and why is it relevant?

Merchant silicon refers to the Application Specific Integrated Chips (aka ASICs, Network Processing Unit [NPU] or chipset) used for TCP/IP switching that are manufactured by vendors, like Broadcom and Cavium. Over the last few years, advancements in merchant silicon's feature sets, functionality and scalability have placed it on par with specially-designed ASICs, thereby lending feasibility to the disaggregated model. In fact, Cisco Systems, which arguably has represented the gold standard in proprietary switching ASICs, is now including a Broadcom chipset to perform the foundational L3/L2switching in the Nexus 9000 Series, one of its highest-performing data center switch platforms.

Spirited discussions can be had with regard to feature parity between merchant silicon ASICs and proprietary ASICs, but one thing is certain: advances in the former have allowed for a departure from the vertically integrated hardware-software model and enabled innovation and competition among networking operating system vendors.

1.7 Explanations of SDN typically consist of references to a control plane, a data plane and a management plane. What are they?

A network's architecture consists of three conceptual layers of functionality in which the tasks of gathering intelligence (control plane), forwarding data (data plane), and managing devices (management plane) are addressed.

When forwarding traffic, a switch or router has two tasks to perform that are handled by the control and data planes. First, it has to understand the topology of the network, meaning the manner in which network devices are connected to each other and the role that each plays. For example, in a layer 2[1] network that is running the Spanning Tree Protocol (STP), a switch will have to exchange Bridge Protocol Data Units (BPDUs) with other switches to determine which one is the root bridge, which path to the root bridge should remain enabled, and which ones need to be blocked to remove any bridging loops.

Similarly, in a layer 3[2] network that is running, say, the Open Shortest Path First (OSPF) routing protocol, messages will have to be exchanged between routers to calculate a loop-free path across the networks in the domain. For example, each

[1] "Layer 2" refers to an environment in which frames are *switched* between hosts on the same VLAN/subnet using only their source and destination MAC addresses.

[2] "Layer 3" refers to an environment in which frames are *routed* between hosts on different VLANs/subnets using their source and/or destination IP addresses.

router on a multi-access network must determine which one is the Designated Router (DR), which is the Backup DR (BDR) and which is a DROTHER. Then OSPF Link State Advertisements (LSAs) will be exchanged between them to populate the OSPF Link-State Database. Once the Dijkstra algorithm is run, the calculated shortest path routes to all the discovered networks in the domain will populate the Routing Information Base (RIB).

In short, the switch's control plane provides the layer 2 and layer 3 data sets (network and host reachability information) that inform the data plane. A network control plane that has an instance running on every switch is said to be distributed.

Second, the switch has to process the user data it receives on the ingress interface and make a forwarding decision. But it may first need to filter the incoming packets according to a configured security policy. Then it will have to analyze the frame's header to glean the necessary source and destination address information and consult the tables and databases that the control plane populated for guidance on how to forward it. At that point, an encapsulation and packet header rewrite operation may have to take place. Then the packet will need to be queued according to a Quality of Service (QoS) policy as it's scheduled for transmission at the outgoing interface. In short, the mechanisms and processing logic used to forward frames exist in the switch's data plane, which is sometimes referred to as the forwarding plane.

Lastly, the management plane concerns itself with those protocols and mechanisms used to manage the switch itself, such as Telnet, SSH, SNMP and NTP. The management plane is sometimes described as a subset of the control plane.

1.8 Discussions around SDN network design oftentimes focus on what are called leaf and spine or Clos architectures – what are they?

Clos architectures are not new. In fact, the name comes from the scientist, Charles Clos, who designed them to scale telecommunications networks back in 1952. Clos architectures lend themselves to efficient horizontal scaling, path resilience, deterministic traffic flows and predictable latency and jitter. The availability of high-density, low profile, fixed configuration, multi-rate switches have made Clos architectures relevant once again.

A leaf-and-spine architecture replicates the internal architecture of a chassis crossbar switch, which is referred to as the switch fabric because of the ability of an interface to send data to any other interface (full mesh). Picture a matrix or the mesh of a fabric with intersecting horizontal and vertical cross stitches.

One of the benefits related to this meshed fabric design involves resiliency and the shrunken failure domain ("blast radius") that results when one of the spine (core) switches fails. In the legacy Fat Tree (multi-tiered) design with only 2 core ("spine") switches, a failure of one of them results in a loss of 50% of the network's forwarding capacity.

ARCHITECTURAL NOTES:

- ToR switches are s6000-ON (40G x 32 ports) Switches
- Spine switches are s6100-ON (40G x 64 ports)
- **All switches running Dell Networking OS9**

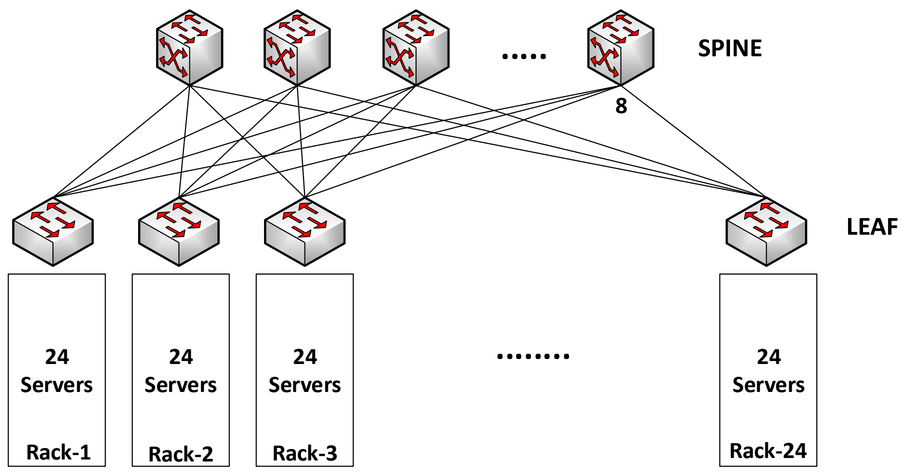


Figure 1-3 – Sample Clos Network with 576 40G QSFP+ Servers

Compare that with a Clos network that leverages L2 or L3 multi-pathing between a leaf switch and multiple (more than 2) spine switches, such as the one depicted in Figure 1-3 above. If one of the 8 spine switches fails, the network will experience a corresponding loss of only **12.5%** of "backplane" capacity.

The following article offers an excellent overview (with historical context) of Clos networks.

http://www.networkworld.com/article/2226122/cisco-subnet/clos-networks--what-s-old-is-new-again.html

1.9 Are Dell's Open Networking solutions considered SDN?

As one may suspect, the answer depends on interpretation. According to the Open Networking Foundation (ONF), which describes itself as "an organization dedicated to the promotion and adoption of Software-Defined Networking (SDN) through open standards development," SDN is defined as a set of standards-based software abstractions that separate the control plane from the data plane and provide programmatic provisioning of the physical and virtual switches. The ultimate objective is to be able to create programmable networks.

Dell Networking's Big Cloud Fabric (BCF) solution from Big Switch Networks is an example of the canonical SDN architecture described by the ONF. On the other hand, Dell Networking's Cumulus Linux or OS10 offerings do not provide such control and data plane separation, but do offer programmatic capabilities through the Linux API and a wide ecosystem of existing Linux-based packages/tools.

Different interpretations of Open Networking and SDN notwithstanding, there is a common thread that runs through the fabrics of all the aforementioned solutions: **the use of software abstractions to create tractable layers of modularity and programmability.** This accelerates innovation at each layer while maintaining a consistent interface.

For example, a computer's architecture is typically represented as consisting of the following five levels of abstraction: hardware, firmware, assembler, operating system (kernel) and processes (applications). Scientists and engineers have leveraged these abstractions for decades to divide responsibility for development and to accelerate innovation at all levels.

The OSI 7-layer model is another good example of layered abstractions. Each layer can have its existing protocols further developed, and new ones added, without impacting the layer above or below it; this is absolutely essential. Imagine how complex (read: impossible) it would be to write an application or add functionality if every change directly impacted – or depended on – the specifics of the physical transport! The Internet, which has undergone profound changes at each layer, would have never survived had the OSI's **data plane** abstractions never existed.

Unfortunately, many such abstractions have been absent in the network **control plane**. This has changed, however, over the last decade thanks to research and development in academic circles. The result is a [r]evolutionary network engineering paradigm known as SDN.

1.10 What are some of the abstractions used in SDN?

In the classic SDN model (Figure 1-4 below), the control plane is abstracted (separated) from the data plane and centralized. Furthermore, within the SDN controller itself, there are several layers of abstraction that enable a division of responsibilities among them. The centralized control plane's software is typically hosted by a cluster of x86 servers (at least two) for redundancy.

The control program, the "northmost" software construct in the SDN controller stack, is presented with an abstracted view (virtual topology) of the network by the Network Hypervisor. The control program is acted upon by an operator or by an external application through a northbound Application Programming Interface (API). A set of network connectivity and service requirements are instantiated by the control program and presented to the Network Hypervisor. The requirements articulated by the control program may be related to a routing or traffic engineering application, or perhaps a security-related primitive to provide tenant isolation, like VLANs or VRFs. **Regardless, the control program is only responsible for expressing the desired outcome without having to actually implement anything. Instead, it relegates the implementation to a lower level of software.**

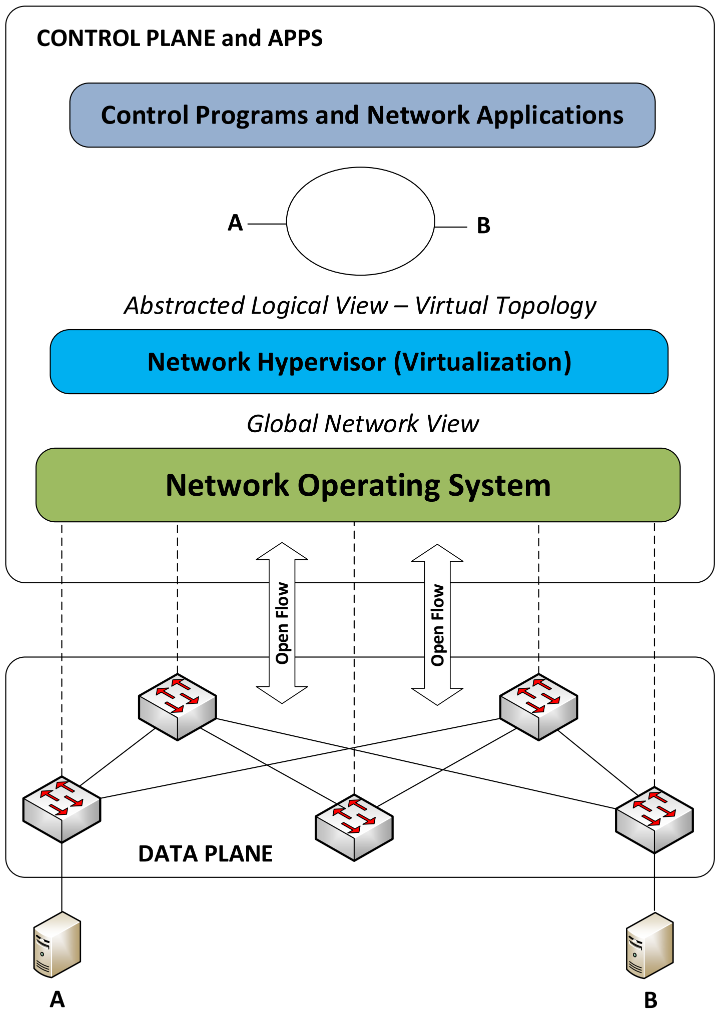

CONTROL PLANE and APPS
Control Programs and Network Applications
A
B
Abstracted Logical View – Virtual Topology
Network Hypervisor (Virtualization)
Global Network View
Network Operating System
Open Flow
Open Flow
DATA PLANE
A
B

Figure 1-4 – Classic SDN Architecture with Control Plane Abstractions

The Network Hypervisor translates (or compiles) the control program's requirements into low-level messages to program the physical network. Since the Network Hypervisor has a global view of the network, it knows how to orchestrate among the physical elements and it knows how to configure the network to execute the requirements presented to it by the control program. The Network Hypervisor is also responsible for discovering all the physical and virtual network endpoints and maintaining state information regarding any topology changes of the underlying network, including the addition or removal of switches, routers, and interconnecting links and their capabilities.

The southbound interface on the SDN controller that interacts directly with the physical and virtual switches is hosted by a Network Operating System (NOS). The NOS has an authoritative view of the entire network, including all the endpoints, and is responsible for conveying forwarding state from the Network Hypervisor to the switches. It also keeps track of traffic statistics, topology changes and interface state.

The NOS pushes the desired forwarding state to the switches by leveraging a low-level protocol known as OpenFlow to program a switch's flow tables. OpenFlow leverages a secure communications channel between the controller and the switches (TLS) and uses TCP as its transport (Port 6653). As a result, the network fabric gets programmed with the forwarding rules that must be honored to implement the control program's requirements. These rules, also known as flow entries, populate a switch's flow-table(s), i.e., forwarding memory, which is stored in Ternary Content Addressable Memory (TCAM).

The OpenFlow standard does not specify the low-level details of how it should be implemented. Like most protocol specifications, the implementation details are left to the vendor's imagination. Moreover, the manner in which a switch's flow tables are populated is directly related to the question of scalability. As such, different operating models have come into existence as the availability of a switch's flow tables have been made available to OpenFlow.

In the days of OpenFlow 1.0, data plane programming was accomplished "**reactively**." In practice, this means that the forwarding memory is treated like a cache that gets gradually populated as packets arrive. If at the time of the packet's arrival there is no entry in a switch's flow table, it will execute an OpenFlow "Packet-in" operation to consult the controller, which would then reactively program a flow entry into the switch's TCAM.

Depending on the size of the network and its level of activity, the number of Packet-in messages and responses could easily choke the communication bus between the switch and the controller, thereby creating a bottleneck and associated scalability limitations. There is also a concomitant CPU overload consideration to be made regarding the controller's ability to respond to so many queries.

Reactive flow control is an effort to work around the limitations of early OpenFlow implementations and not an inherent requirement of the protocol itself. Because the amount of forwarding memory (TCAM) exposed by these early implementations was so small, SDN controllers could not program all necessary forwarding rules at once and were in effect forced to cache them dynamically.

On the other hand, OpenFlow versions 1.3 and later allow for the utilization of multiple flow tables as part of its pipeline processing schema. Therefore, the network control plane can converge **proactively** and populate the tables with the forwarding rules in a preemptive manner. As such, once a policy is configured (e.g., adding an ACL from the CLI) or a network change occurs (e.g., a link comes up), the controller proactively updates all of the necessary switches and forwarding memory with the new forwarding rules.

A proactive OpenFlow implementation means that common case packet forwarding operations do not even involve the controller and the data plane/control plane channel is no longer a bottleneck, the controller's CPU is no longer being overwhelmed, and overall packet processing latency is reduced. Proactive forwarding is only feasible with switches that expose multiple tables of forwarding memory.

In either case, upon receipt of a frame, the switch's data plane will try to match the information found in the header with the information found in its flow tables. If a match is found, an associated action against the frame will be taken – either it will be forwarded or dropped – and the relevant counters will be incremented. If a match is not found, the frame will be punted to the controller.

As per the OpenFlow 1.5 specification, OpenFlow-enabled switches are required to support up to 12 match-header fields (e.g., source/destination MAC, source/destination IP, TCP ports, etc.), but up to 38 optional fields are defined, including MPLS (Multi-protocol Label Switching) labels, TTL counters and QoS markings.

In a similar manner to the way a server's operating system compiles high-level source program requirements into a low-level machine language to program a server's CPU, the Network Hypervisor compiles the control program's requirements and the NOS leverages OpenFlow to program the switch's flow tables. This is why Dr. Martin Casado, OpenFlow's primary developer, once described his creation as something akin to an x86 instruction set for a network. With this basic architecture and set of abstractions in place, it is up to the industry to develop control programs and applications that will exploit them. **Therefore, OpenFlow should be viewed as a way of implementing an SDN network; it's a means to an end and not the end in and of itself.**

As alluded to previously, not all abstractions deal directly with separating the control plane from the data plane. In the Open Networking (disaggregated) model, an abstraction in the form of a Linux program called the Open Networking Install Environment (ONIE) exists within the switching hardware itself. The ONIE bootloader allows different third party operating systems to be loaded onto a bare-metal switch that uses industry standard merchant silicon, such as Broadcom or Cavium ASICs.

That third-party OS may be the engine behind an SDN solution, as in the case of Big Switch Networks – or perhaps not, as described earlier with Cumulus Linux. The network shown in Figure 1-5 below is an example of a typical controller-based SDN architecture. Notice that it is the exact same topology of switches that were used in the Clos network in Figure 1-3, which happened to have Dell Networking OS9 running on them, and no controllers. However, because the Dell switches were ON-enabled, all one had to do was uninstall OS9, install Big Switch Networks' Switch Light operating system for Big Cloud Fabric (BCF), and deploy the BCF centralized controllers. With relatively minimal disruption, a legacy network is converted to a next-generation SDN without having to remove or replace any switching hardware. **That is the power of abstractions!**

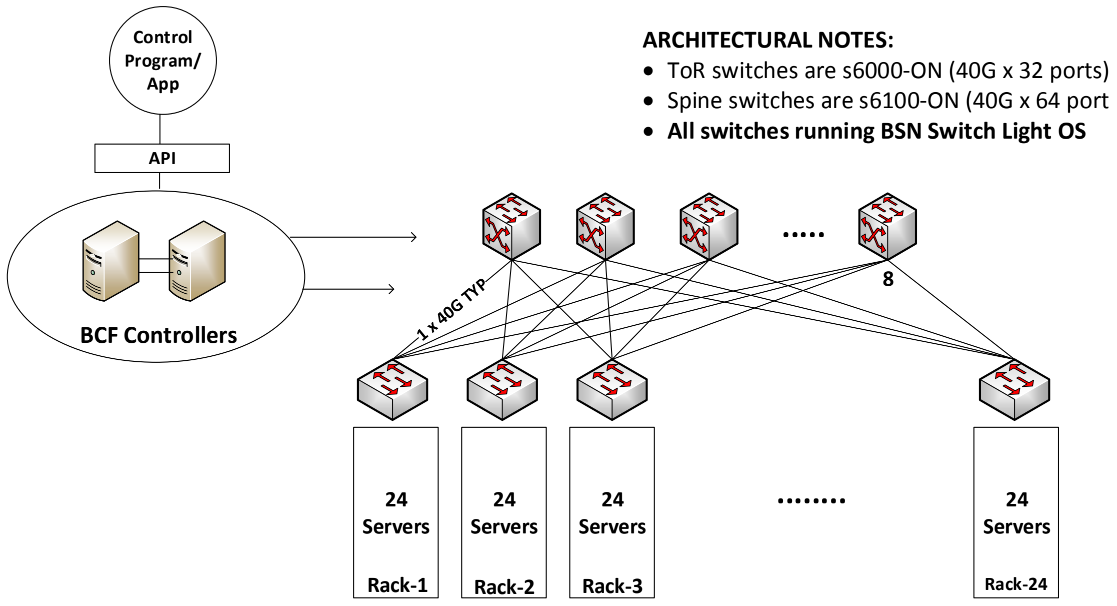


Figure 1-5 – Sample Big Switch BCF SDN Clos-based Architecture

Yet another abstraction, this time provided by Dell's OS10 SDN/DevOps-inspired Operating System, is known as the Switch Abstraction Interface (SAI, pronounced

”sigh”). The SAI takes the ONIE model to the next logical step by providing a standard C-based API to program switching ASICs, thus allowing different vendor chipsets (other than Broadcom) to be deployed with various different third party operating systems.

1.11 SDN is oftentimes equated with OpenFlow. Is that the only “southbound” protocol used by SDN controllers?

Although OpenFlow was the original SDN protocol used to provide programmatic control of the data plane, SDN architectures have evolved since then. They now include a litany of other design approaches and protocols that exploit the original abstractions that the original SDN architecture defined. OpenFlow itself has been considerably developed since its inception to include more functionality, such as the ability to support multiple tables and a management protocol known as OF-Config or the OpenFlow Configuration and Management Protocol. OF-Config is a companion protocol to OpenFlow and leverages YANG modules (RFC 6020) and NETCONF (RFC 6241) to model and manipulate configurable system attributes and operational state for both physical and virtual switches that run OpenFlow. Parenthetically, the OVSDB protocol (Open vSwitch Database) is a configuration and management protocol for OVS.

An SDN controller from vendors like Cisco take an approach that deviates from the classic SDN model. For example, Cisco’s Application Centric Infrastructure (ACI) controller, known as the Application Policy Infrastructure Controller (APIC), does not fully centralize the control plane (logically speaking) nor does its southbound protocol assume a completely ”dumb” switch.

In the view of Cisco’s developers, the built-in intelligence that has traditionally defined a switch’s control plane has value to offer: specifically, in delivering programmable networks by sharing responsibility with the controller as part of a hybrid approach. Instead of using OpenFlow to program a switch’s flow tables with low-level match-action instructions, Cisco’s APIC uses a higher-level protocol called OpFlex to convey a set of policy-based requirements from the APIC controller to the switch, which will in turn leverage its own policy engine to implement them.

Cisco’s approach illustrates the difference between an *imperative* programming model and a *declarative* one. In the former, a control program will define a desired end state as well the exact steps that need to be executed to achieve it; whereas in the latter, which is the programming model leveraged by Cisco’s APIC, the desired end state is defined without directives on how to implement it. Instead, the implementation details are left for an intelligent forwarding device to determine. One way to think about this is to consider the idea that the lowest compiled state for a set of programming instructions has been moved further "south" in the declarative model from the SDN controller's Network Operating System (centralized control plane) to the switch's operating system (distributed control plane).

Moreover, service provider-oriented SDN solutions leverage a new MP-BGP (Multi-Protocol Border Gateway Protocol) NLRI (Network Layer Reachability Information) and address family known as BGP-LS (BGP Link State) to communicate IGP (Interior Gateway Protocol - OSPF or IS-IS) link state information from an IGP speaker to a centralized controller for the purpose of establishing MPLS-based Label Switch Paths (LSP) within and across different domains as part of an end-to-end MPLS-TE (Traffic Engineering) solution. Yet another extension to MP-BGP known as Ethernet Virtual Private Network (EVPN) offers a new address family to convey L2 MAC address information for endpoints as a control plane alternative to VxLAN's (Virtual Extensible LAN) default "flood-and-learn" approach.

SDx Central published a report in 2015 that includes some of the most popular SDN controller solutions and the southbound protocols they use. As illustrated in the list below, OpenFlow is just one of many southbound protocols to deliver programmable networks.

URL to Report: https://www.sdxcentral.com/reports/sdn-controllers-report-2015/

From the report:

- Brocade SDN Controller – OpenFlow, OVSDB, BGP and NETCONF
- Cisco APIC SDN Controller – OpFlex
- Cisco Virtual Topology System – NETCONF, RESTCONF, MP-BGP EVPN
- Ericsson SDN Controller – OpenFlow, OVSDB, BGP, NETCONF, PCEP, BGP-LS
- Juniper Contrail – BGP, XMPP, NETCONF, OVSDB
- NEC Programmable Flow PF6800 SDN Controller – OpenFlow
- Avaya SDN Fx Controller – OpenFlow, OVSDB, OF-Config, NETCONF
- Nuage Networks SDN Controller – OpenFlow, OVSDB, BGP
- ODL Lithium – OpenFlow, OVSDB, BGP, NETCONF
- VMware NSX (not in report) – User World Agent, netcpa, RabbitMQ, vsfwd (NSX-vSphere). OpenFlow and ovsdb (NSX-Multi-Hypervisor)

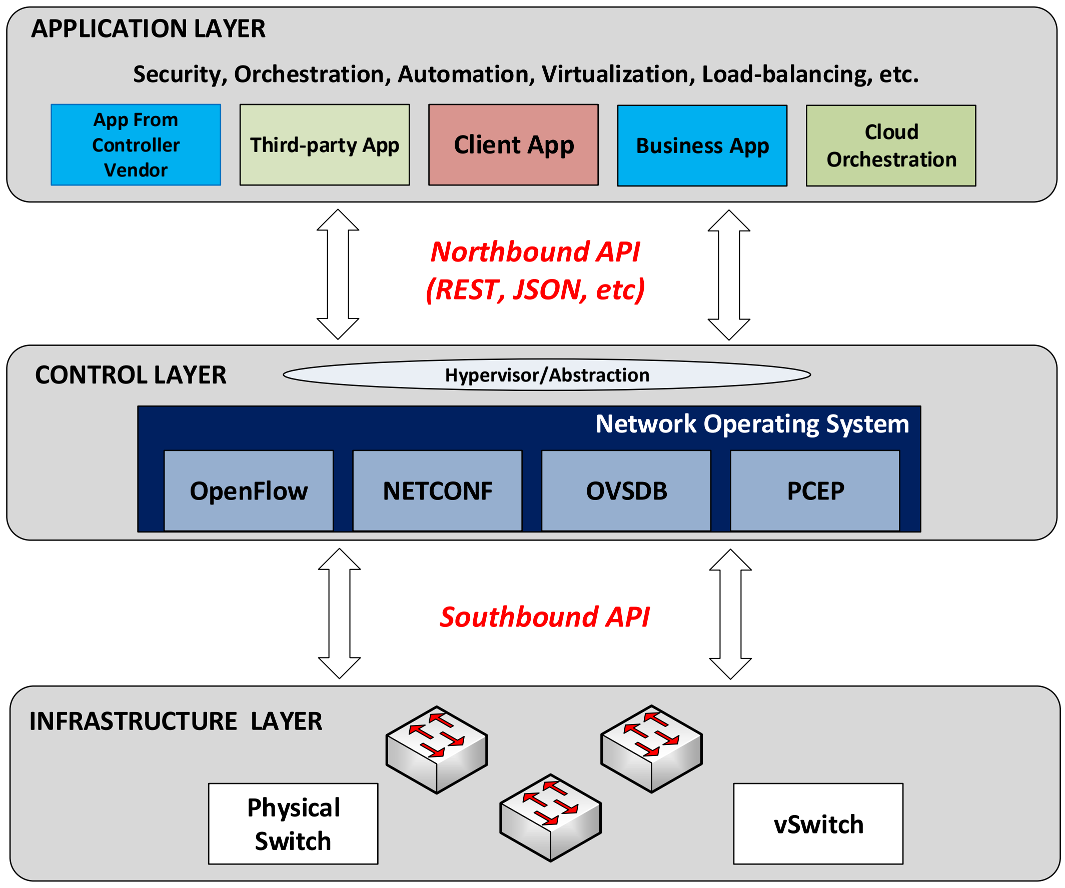


Figure 1-6 Generalized View of Classic SDN Architecture

1.12 What is the Northbound API used for in an SDN Network?

During the early days of SDN, it was the Southbound API, and in particular OpenFlow, that garnered the lion's share of the industry's attention. That's understandable given that OpenFlow was the foundational technology that enabled programmatic networks. The implications of separating the control plane from the data plane and leveraging abstractions to dynamically program the network were foreign concepts that network engineers spent a good amount of time trying to grasp and appreciate. As such, conversations around applications took a backseat and were nebulous at best, since no one could point to a specific use case or a vendor that was actually shipping a product that anyone had in production. Much has changed since then, and the application layer that consumes the Northbound API has been demystified.

The Northbound API makes the network's control information (Control Plane) available to higher level abstractions, such as applications. Everything below the Northbound API (i.e. the control and forwarding planes) still delivers the same traditional networking functionalities. The network basically continues to forward packets, as it always has, although the implementation is different. What is new, insofar as the application is concerned, is the architecture. Instead of control information being distributed across multiple network devices, thereby requiring complex distributed algorithms, it is now abstracted from the hardware, centralized and made available to the applications via the API. The network can thus be programmed with software applications instead of command line interface (CLI).

The application could be designed to deliver traditional network services, such as firewalls or load balancers and basic L2 and L3 forwarding, or complex orchestration across cloud resources (storage, compute and network), like OpenStack. SDN applications can instantly change network configuration to align it with business objectives or customer Service Level Agreements (SLA), such as forwarding packets over the least expensive path (think SD-WAN), dynamically adapting QoS based on available bandwidth and user subscription, dropping unwanted packets and tracking back to their source for containment, etc. Security, traffic engineering, multi-tenancy management, and network monitoring are just a few examples of the applications that are leveraged by the Northbound API.

Some vendors sell a controller with built-in applications, like Big Switch's Monitoring Fabric, Cisco's ACI and vMware's NSX. All of them, however, expose an API (RESTful) that software developers can write to with the hope that a massive ecosystem of software can be developed around it, thereby making their API a de facto standard.

1.13 SDN-based solutions are oftentimes categorized in terms of underlay and overlay networks. What are they and what is the difference?

An underlay network is defined as the underlying physical network – the transport – upon which virtual networks (overlays) forward data. Each network type has implications with regard to the possession and exploitation of topological awareness, forwarding intelligence and path selection.

Overlay networks put flow-based path selection at the endpoints, such as hypervisors or their hardware-based analogs. In other words, the forwarding intelligence resides at the edge of the network. Overlays can be used when one does not have control of the underlying infrastructure. A good example of an overlay network is a Virtual Private Network (VPN), like the one telecommuters use every day to access their company's intranet. A laptop with a VPN client installed is one endpoint of the tunnel while the VPN concentrator at company headquarters acts as the other. The underlay is provided by the ISPs that connect the two ends.

At the sending end, the traffic payload is encapsulated in another frame. A separate tunnel (outside) header is appended to it whose source and destination IP addresses are those of the tunnel endpoints. At the distant end, the original frame is de-encapsulated and forwarded onto the private network. The original packet travels through the physical underlay without its payload, including its original (inside) header, ever being examined or acted upon by the intermediate switching nodes. This is why it is said to have traveled through a tunnel: insofar as the original frame is concerned, its headers were examined twice, once at the encapsulating end, and again at the de-encapsulating end – with no intermediate hops in between.

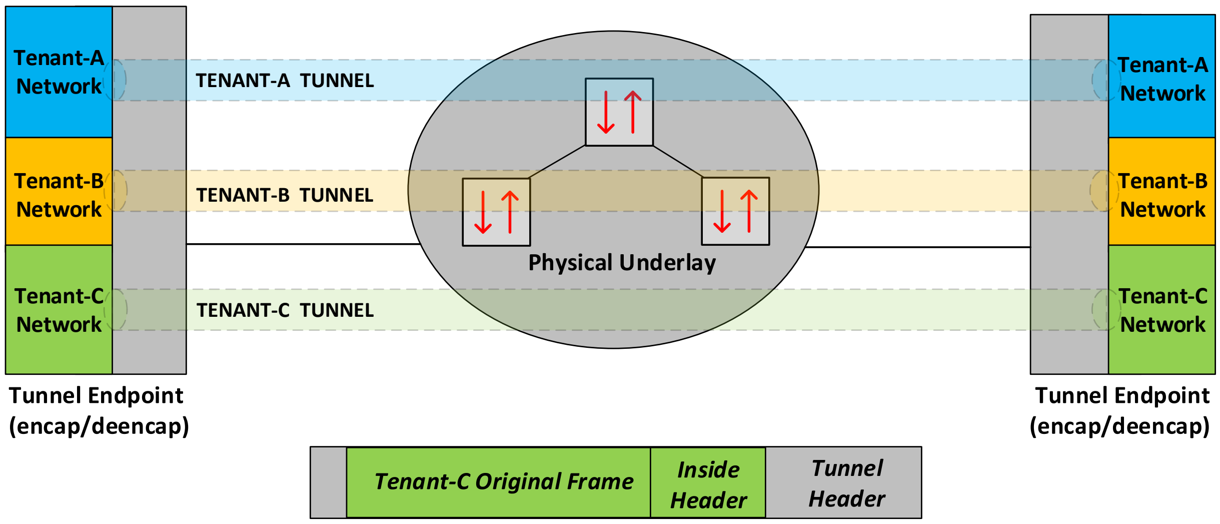


Figure 1-7 – Overlay Networks in a Multi-Tenant Environment. The underlay forwards traffic based on tunnel header information. The tunnel endpoint is a hypervisor or a physical switch.

SDN and Network Virtualization have brought overlay networks inside the data center. They are used in cloud-based XaaS applications for multi-tenancy network isolation and to provide Layer-2 adjacency across routed boundaries. The two most common overlay tunneling technologies used in SDN solutions today are VxLAN and GRE (Generic Route Encapsulation). In each case, the hypervisor acts as the virtual tunnel endpoints for virtual workloads. For physical servers with no hypervisor installed, a hardware-based endpoint is used, such as the VxLAN Tunnel Endpoint (VTEP) functionality available in the Broadcom Trident-2 (and later) chipset.

Take note that the underlay fabric has no knowledge of the existence of any private networks that sit "behind" the tunnel endpoints. That relieves it from having to create, store or change network forwarding state. That is of particular value in a dynamic cloud environment with migrating workloads whose identity is routinely being decoupled from its location courtesy of the logical tunnels. All the underlay has to do is route packets between static endpoints, which is something that IP fabrics have been doing very well for many years.

By the same token, the overlay network has no knowledge of the underlay network's topology, forwarding mechanisms, routing protocols, security and quality of service policies, etc. It can be said that the two are orthogonal to each other, although the underlay provides transport services to the overlay. Therefore, the transport fabric needs to be reliable, robust, redundant and resilient.

There are SDN solutions available that are designed for a more collaborative approach between overlays and underlays, in which the overlay is allowed to exploit underlay information to conduct efficient path monitoring and construct highly-reliable overlay topologies. Cisco's ACI (Application Centric Infrastructure) and Big Switch Network's Big Cloud Fabric (BCF) solutions are examples of this approach.

1.14 What problems does SDN solve and what are some real-world use cases?

As Professor Scott Shenker, one of the pioneers of SDN development, once remarked, ***"[OpenFlow] doesn't let you do anything you couldn't do on a network before."*** Given that this comes from one of the most passionate and prolific voices in SDN evangelism, this is not an altogether mundane comment.

So what exactly does OpenFlow-based SDN provide? **SDN establishes a set of clearly defined layers of functionality that comprise a set of principles upon which to build networks. These principles are embodied in the following:**

- Separation of the control plane from the data plane
- A centralized view of the entire network, including endpoints
- Software abstractions that allow for modularity within the control plane stack
- The concept of a network operating system that abstracts the installation of state in network switches from the logic and applications that control the behavior of the network
- A standardized programmatic interface (API) for the data plane

This clearly defined architecture sets the stage for the development of control programs and applications that can exploit these constructs to solve the network challenges that are addressed sub-optimally – if at all – today, as well as the requirements for future use cases. With that, the network has been put on par with its compute and storage counterparts.

In terms of the usefulness of an OpenFlow-based SDN is concerned, given the SDN controller's centralized view of the entire network, which includes the identity (IP/MAC addresses) and location (switch/hostname/port) of all the physical and virtual endpoints, deploying legacy networking functionality (segmentation, isolation, quality-of-service, security and load balancing, etc.), as well as managing and monitoring the network, can be simplified.

The so-called "killer app" for SDN is Network Virtualization. It's worth mentioning that the availability to leverage virtualized networking constructs is not new at all. For example, Virtual Local Area Networks (VLANs) have been in existence for over 25 years. And network engineers have been deploying virtual device contexts for routers and firewalls, Virtual Routing and Forwarding instances (VRFs), logical routers, data-path virtualization solutions, like Virtual Private Networks (VPNs) and Multi-protocol Label Switching (MPLS), control plane virtualization and other forms of network virtualization for many years. In fact, a well-known Cisco Press book titled *Network Virtualization* by Victor Moreno and Kumar Reddy was published in 2006 – long before anyone heard of SDN and OpenFlow.

However, within the context of SDN, Network Virtualization (NV) refers to the creation of virtualized network entities (routers, switches, firewalls, and load balancers) using one of two methods.

- **The first is through software emulation plus overlays. This model is referred to as hypervisor-based or overlay-based network virtualization, which can use a centralized controller – i.e. vMware NSX/Nicira NVP, OpenStack, Big Switch BCF P+V, etc.**

- **The second method is to virtualize the physical network itself via the classic SDN model, which utilizes a centralized controller and a southbound API for the data plane, such as OpenFlow – i.e. Big Switch BCF, NEC Programmable Flow, Brocade SDN Controller, Cisco APIC, etc.**

In either case, the virtual networks exist as part of a software abstraction that is decoupled from the underlying physical hardware. The hardware acts as the actual forwarding engine, the substrate upon which the virtual constructs are built. In the overlay model, that hardware is typically a virtualized server running hypervisor software, in which case the physical network is irrelevant. In the classic SDN approach (sometimes referred to as an underlay model), the underlying hardware is the physical network itself, in which case a network virtualization app/control program can leverage SDN's abstractions to create from it a shared pool of network transport that can be "sliced" into different virtualized networks.

The compute analog to NV is Server Virtualization, where the familiar attributes of a physical server are decoupled and reproduced in software (the hypervisor) as vCPU, vRAM, vNIC, etc. And just like a virtual machine, a virtual network can be instantiated, manipulated, saved and deleted with a few clicks of the mouse.

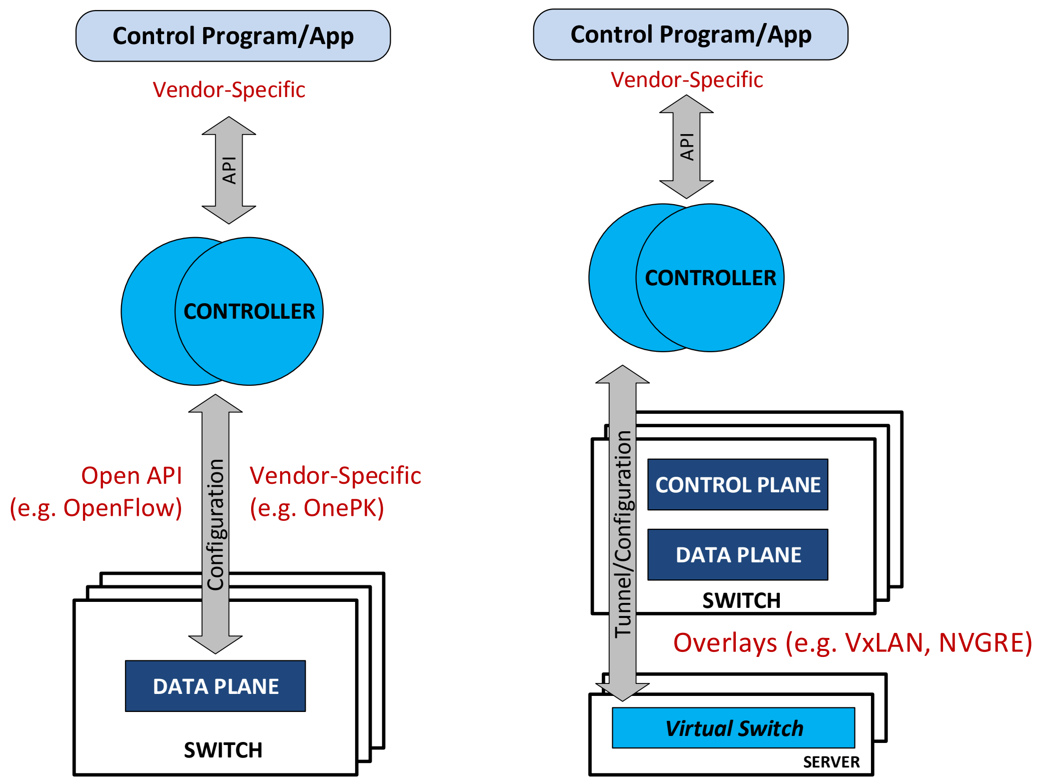


Figure 1-8 – Two Approaches to Network Virtualization: Classic SDN (left) and Overlay SDN (right). Note that the underlay switches are inconsequential in the overlay model.

There is a growing number of use cases and applicability for SDN, such as SD-WAN, network monitoring and tap aggregation, service insertion (IDS/IPS, firewalls, load-balancers), campus applications, automation and orchestration and others. The list grows as more control programs/applications are written and the technology matures.

2.0 Dell OS10 and Third-Party OS Vendor Solutions

The following sections will give a high-level overview of Dell Networking's Open Networking-enabled software solutions, including the Dell OS10 operating system. They are part of Dell's disaggregated software-hardware model. Given that each operating system defines a certain architecture and design approach, the choice of which one to deploy is a function of the specific architectural and operational requirements.

2.1 Dell OS10

In January of 2016, Dell Networking released the Base version (Open Edition) of its next-generation data center networking operating system called OS10. The "10" reflects the next numerical progression from Dell's legacy OS9 software train, but the two OSes are completely different. OS9 is a mature, tried and tested network switch operating system with a NetBSD kernel and a wide range of "table stakes" data center and campus networking technologies. As Dell Networking's flagship, legacy operating system, OS9 has a global footprint and a long history of success.

On the other hand, Dell OS10 is slated for release in general purpose enterprise deployments (Enterprise Edition) by CY16Q4. It is new and very different from OS9. Also, like the third-party software vendor solutions that will be described later in this section, Dell OS10 leverages the Open Networking Install Environment (ONIE) that comes with a Dell "ON" switch, such as the S4048-ON.

From the Dell OS10 FAQ Document – June 2016:

The OS10 Open Edition includes the Linux distribution, SAI API (NPU abstraction), and CPS API (Control Plane Abstraction) over REST and Python wrappers. The Open Edition enables native Linux applications (management, monitoring), third party applications such as Quagga and Bird, and the integration of custom applications via the CPS API.

The Open Edition is a development environment suitable for lab environments to prototype, develop, and test new applications. The Open Edition is also suitable for customers that require a high level of customization and have in-house capabilities for both development and support for production environments.

The OS10 Enterprise Edition includes the Open Edition capabilities plus a full Dell Networking protocol stack and management infrastructure providing a CLI to configure and monitor the platform. The Dell protocol stack comes fully integrated with the system software for the underlying hardware.

OS10 Enterprise Edition is designed for mainstream production networking environments, especially for organizations transitioning to a DevOps operational model.

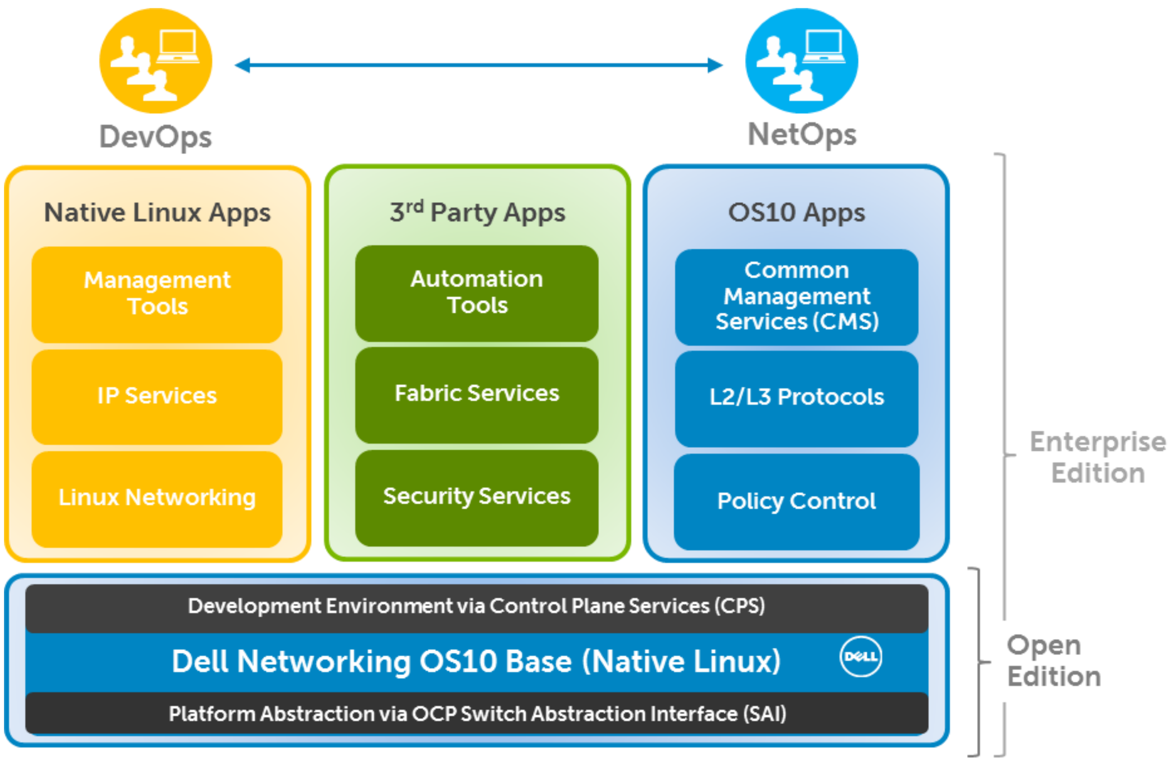


Figure 2-1 Dell OS10 Software Stack with Unmodifed Linux Kernel

2.1.1 What are Dell Networking OS10's most salient features and innovations?

The OS10 Base Module has been available for free since March of 2016 and runs an unmodified Debian Jesse Linux kernel. Linux is one of the most widely-used operating systems and can provide deployment and operational homogeneity across multiple IT layers, including networking, storage and compute. The OS10 Base Module can leverage the wide array of software packages and tools that exist within the Linux ecosystem today, like Quagga and Bird for routing TCP/IP, and it also provides a rich environment for developing homegrown applications and leveraging DevOps automation and management tools, such as Puppet, Chef and Kubernetes.

Dell OS10 is a very modular operating system that consists of several layers of abstraction. Of these, the most notable are the Switch Abstraction Interface (SAI), the Control Plane Services layer (CPS) and the Common Management Services interface (CMS). Combined, they make application development and network programmability (configuration and management) simpler and more streamlined. Recall that the very purpose of software abstractions is to create tractable layers of functionality that can be developed independently of each other, and without consideration for the underlying complexities and detail. Recall the abstracted control plane in the ONF's SDN model that was covered in section 1.10. This modularity makes OS10 extremely flexible in terms of its programmability and supported deployment scenarios.

2.1.2 Control Plane Services

The benefits of Dell OS10 Control Plane Services revolve around platform stability, scalability, feature development velocity, extensibility, and integration with external systems and applications. The CPS is the chief enabler of a network operating system that is purposefully designed for cloud/DevOps/SDN environments. The CPS layer is an inter-application framework that provides a stateful, distributed database service that also leverages a publish/subscribe messaging paradigm. OS10 applications use the CPS API to communicate with each other, just as custom-written applications use the CPS API to communicate with the OS10 components and services. As such, an application expresses a desire (subscribe) to receive state information regarding one of the OS10 device's subsystems, and then consumes that information in order to take a prescribed action.

To understand how the CPS infrastructure operates, consider the example of a Temperature Control (TC) application. This simple application will monitor an OS10-based switch's temperature and the speed of its cooling fans. The TC application will subscribe to event information regarding the temperature of the switch by registering its interest in such an event with the CPS layer. When the temperature exceeds a pre-configured threshold, the Platform Abstraction Service (PAS, which is a higher layer abstraction of the actual hardware drivers) will publish that information. In response to the notification, the TC application can speed up the rotation of the fans and/or send out an urgent email to the network operations center team, or even manipulate the data center's environmental controls to compensate for the cooling system's failure.

The CPS API has been leveraged by Dell Engineering in partnership with other vendors and clients to deliver innovative solutions. The following are two examples.

➢ **Silverpeak Accelerated Route Convergence**

The objective of the convergence application is to accelerate route convergence and failover after a link failure. In the topology under test, the SD-WAN edge appliance was connected to both a broadband Internet service and an MPLS cloud through a Dell switch that was running OS10. Typically, in SD-WAN solutions, delay-sensitive/tier 1 traffic travels over a highly-reliable and private MPLS link, while less demanding traffic is sent over a public broadband connection. The application must register with the CPS and subscribe to event information regarding the state of the physical link that is connected to the broadband provider. Once the link is deliberately failed, an event notification is sent to the CPS, which in turn publishes it for consumption by the application. The application, in turn, will shut down the link that the SD-WAN appliance uses for Internet-bound traffic, thereby triggering an immediate reconvergence and traffic switchover to the MPLS link.

- **OS10 Integration with Kubernetes Networking**

The client had an existing Kubernetes container environment that was leveraging the Tectonic distribution of CoreOS, which uses Flannel to provide an overlay network for inter-pod communications. Dell provided a solution for Kubernetes networking, where the customer was able do away with overlay networks and allowed them to deploy a Kubernetes cluster on an existing L3 fabric. The solution is a simple app that runs on a top-of-rack Dell OS10 switch. The app extracts the necessary network information about the Kubernetes cluster that is stored in the distributed key value database known as *etcd*. The CPS API is leveraged to program the container subnets on to the kernel and the ASIC by pointing those subnets to the server that hosts the particular subnet. Then BGP advertises those subnets to a peer router, thereby ensuring that they are reachable.

2.1.3 Switch Abstraction Interface

Under the hood, Dell OS10 employs the Switch Abstraction Interface (SAI), which provides a common interface for operating system software to communicate with and program the underlying networking ASIC/NPU. That operating system software can be the kernel itself or processes that are running in user space, while the chipset may be from any one of several vendors whose SDK conforms to the SAI standard. The SAI allows Dell to quickly integrate new ASIC/NPUs for customers reducing the time it takes to provide them with the latest chipset technologies.

This is not trivial, as it represents the next step in the evolution of disaggregation. The end-result is that ASICs from multiple vendors may be deployed (Broadcom, Cavium, etc.) with the operating system software of choice, thereby allowing network architects to build a network that meets their needs without any deployment constraints imposed by proprietary solutions.

2.1.4 Common Management Services

Dell OS10 exposes an API for use by external management systems such as a NETCONF client. NETCONF is a switch management protocol that was first developed by the Internet Engineering Task Force (IETF) in 2006 and later revised in 2011. It provides mechanisms to install, manipulate and delete configuration files on network devices. The impetus for its development was the realization by the industry that SNMP was only being used as a means to monitor the network and not write to a system's configuration. Network engineers preferred vendor CLI and vendor-provided scripts over unwieldy SNMP workflows.

In OS10, the NETCONF server leverages the data modeling structures that are defined by Yang data modules. Yang modules explicitly and precisely determine the syntax and semantics of the data that can be externally managed and manipulated by the NETCONF protocol. In other words, Yang provides a well-defined abstraction of the different elements of a switch's subsystems that can be configured by an external management system, such as the attributes of physical and virtual interfaces, IP routing protocols, platform management related items, etc. Yang also makes a distinction between configuration data and operational state data for the purpose of streamlining management.

Through the Hello process and the subsequent exchange of *capabilities* that occurs during session initialization between a NETCONF client (network management station) and NETCONF server (OS10 device), Yang advertises the specific elements in the device's subsystems that can be configured, monitored or otherwise manipulated by the management station. Notifications and administrative actions that are available to NETCONF are also made known at that time.

A concrete example of how OS10's NETCONF agent can make life easier for network operators involves the data stores that it defines. NETCONF defines several types of data stores in the network device, including the startup, running and candidate data stores. The running data store is where the device's current running configuration can be found and the startup data store includes the configuration that will be executed upon starting the device. These two data stores are well-known to network engineers. On the other hand, the candidate data store, although less familiar to some, is comparatively the more interesting element in terms of operations. The candidate data store holds a potential configuration that can be executed once it is committed to the running configuration. This is a very handy tool to have at one's disposal when making configuration changes that may inadvertently cause an outage. The candidate data configuration will be withdrawn after a set period of time has lapsed if it is not committed, thereby restoring the network to a previously operational state.

NETCONF also works symbiotically with the candidate data store to make network configuration changes to many devices at once in what is described as network-wide transactional operations. The change is first made to the each device's candidate data store and only after all devices agree to the feasibility of the new configuration will the candidate configurations be executed. If something fails and the *confirm-commit* message is not executed, the changes will *rollback* to a previous functioning state.

OS10 meets the demands that Cloud/DevOps environments make of modern network operating systems, such as programmatic access to a device's configuration, automation, the ability to execute unordered configurations (relegating ordering to the intelligence within the device itself), configuration validation, the ability to execute network-wide configuration changes at once, a standardized data structure to advertise configurable elements and the ability to separate configuration and state information.

2.2 Cumulus Linux

The Cumulus operating system for Dell Networking ON-based switches is most suitable for enterprises whose personnel are familiar with Linux. The Cumulus administrator interface is not quite an “industry-standard” command line interface (CLI) to which network engineers have become accustomed, but it is similar. One should be familiar with Linux file manipulation and basic command sets for managing images, as well as with Linux-based networking constructs to be effective. In short, there is indeed a learning curve for the “average” network administrator, but for engineers who want to capitalize on the value of deploying a Linux operating system in their environments and also possess the drive to venture slightly out of their comfort zone, the learning curve is certainly manageable.

One of the biggest benefits offered by Cumulus is the ability to manage the network with the same automation tools that are used to manage a Linux-based server farm, such as Ansible, Puppet and Chef.

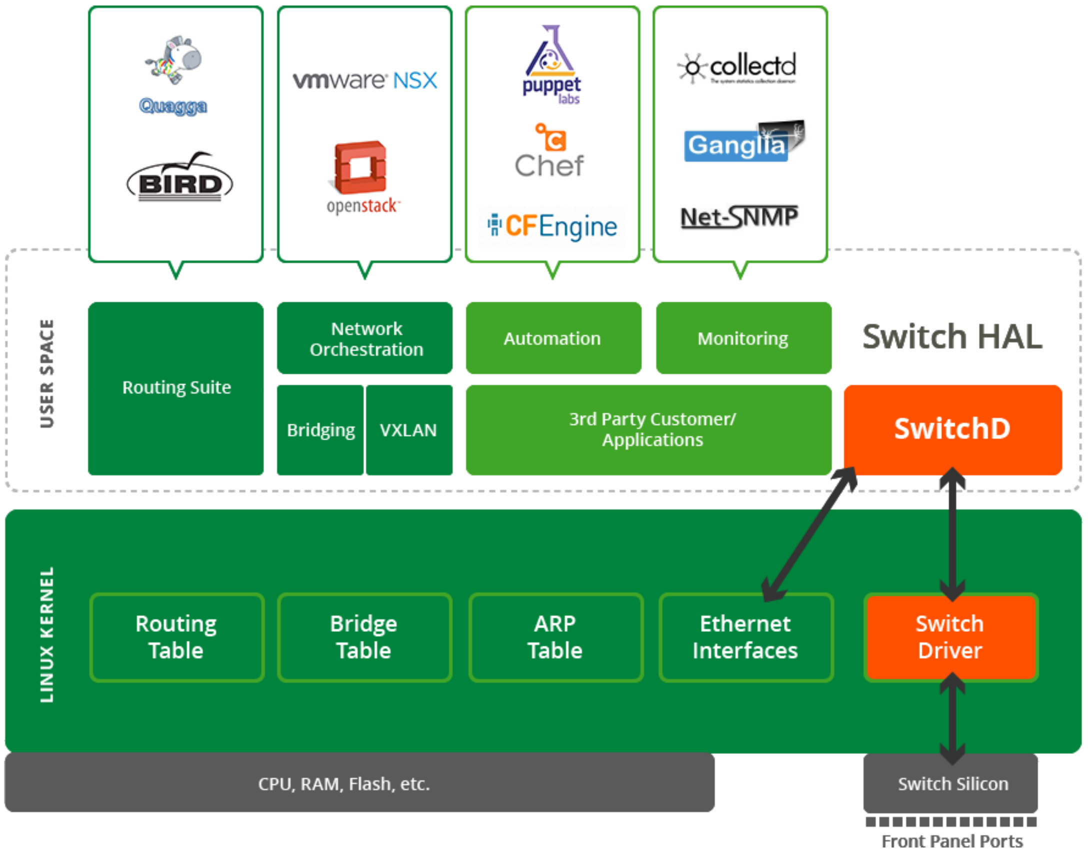


Figure 2-2 Cumulus Linux Network Operating System Architecture (from Cumulus website)

2.2.1 What is Cumulus Linux?

Cumulus Linux is a networking focused Linux distribution that is deeply rooted in Debian. Cumulus Linux replaces the vertically integrated operating system that a networking vendor would normally provide with their hardware. Switches running Cumulus Linux provide standard networking functions such as bridging, routing, VLANs, MLAGs, IPv4/IPv6, OSPF, BGP, access control, VRFs, and VxLAN overlays.

2.2.2 Is Cumulus Linux an overlay or underlay solution?

Insofar as Cumulus Linux is simply an operating system that runs on the physical networking hardware itself, it can loosely be categorized as an underlay, although it does not support an SDN architecture with a centralized control plane, as described earlier in this paper. It does support network overlays by providing the ability to enable the VxLAN gateway feature in the Broadcom chip.

2.2.3 Does Cumulus Linux run OpenFlow or any other "southbound" protocol for the purpose of communicating with a centralized SDN controller?

No. According to Cumulus CEO JR Rivers, *"The only way you can truly be successful in meeting the customer needs around OpenFlow is to be truly focused on a great OpenFlow agent that lives on the switch platform...In general, when customers want to use OpenFlow, Cumulus will say, go talk to Big Switch."*

2.2.4 Can Cumulus Linux be deployed in an NSX/VxLAN environment?

Absolutely. The value offered by virtual networks comes from the fact that they are abstracted from the underlying physical network. Therefore, a virtualization overlay, like VxLAN, which is also the overlay engine behind NSX, is orthogonal to the switching hardware and the operating system they run. The VxLAN Tunnel Endpoint (VTEP) resides on the virtualized server's hypervisor. This is where the encapsulation/de-encapsulation of the tunnel header takes place. The underlying network only needs to have the ability to forward IP traffic between those tunnel endpoints.

There is an exception and it involves the case in which a server is not running a virtualization platform and therefore has no hypervisor. In that case, the server will require a physical switch to act as the VTEP to allow for communication between the physical and virtual server environments. Broadcom's Trident-2 chipset fully supports that functionality from a hardware perspective, but the operating system running on the switch will have to offer the ability to expose that functionality through CLI. Cumulus Linux version 2.0 and later fully support VxLAN and NSX.

2.2.5 Can Dell switches running Cumulus Linux be deployed in an OpenStack environment?

Yes. In fact, there is a detailed design guide on Cumulus' website for deploying Cumulus Linux with OpenStack.

URL to Design Guide:

https://cumulusnetworks.com/media/resources/validated-design-guides/VMware-vSphere-Cumulus-Linux-Validated-Design-Guide.pdf

2.2.6 On which Dell Networking switches can Cumulus Linux be deployed?

Dell Networking switches with the "ON" designation are capable of running third-party operating systems. The Cumulus Linux HCL is actively updated as more vendor hardware platforms are tested and certified. As of the writing of this paper, the Dell Networking S6000-ON, S4048-ON, S3048-ON, S6100-ON and the Z9100-ON are on the Cumulus HCL.

2.2.7 What Services and Support model are in place for a customer who buys a Dell switch with Cumulus Linux?

As always, the hardware is fully supported by Dell and that does not change at all in the disaggregated model. Typically, the customer is responsible for installing the operating system and configuring the switch. However, the Dell Networking SE, along with their counterparts at Cumulus, are resources who can be leveraged to give the customer a smooth out-of-box experience. As of today, Dell Services does not provide any services SKU for a Cumulus Linux rollout, but services can be purchased from Cumulus.

2.3 Big Switch Networks (BSN)

Contrary to what the name suggests, Big Switch Networks does not sell switches. The company sells networking software that is loaded onto an Open Networking-enabled switch as part of a disaggregated deployment model. Dell offers two solutions from BSN: Big Cloud Fabric (BCF) and Big Monitoring Fabric (BMF). Both solutions involve a classic controller-based SDN architecture, and the operating system for both is BSN's Switch Light OS.

2.3.1 Big Cloud Fabric (BCF)

BCF is an SDN-based networking solution that closely resembles the classic SDN model that was described in the previous section. The architecture includes a pair of centralized controllers that present a northbound API for consumption by orchestration and cloud management tools, such as CloudStack, OpenStack and vMware. There is also a CLI and GUI interface for human interaction, but they are both REST API clients that translate command inputs into REST calls.

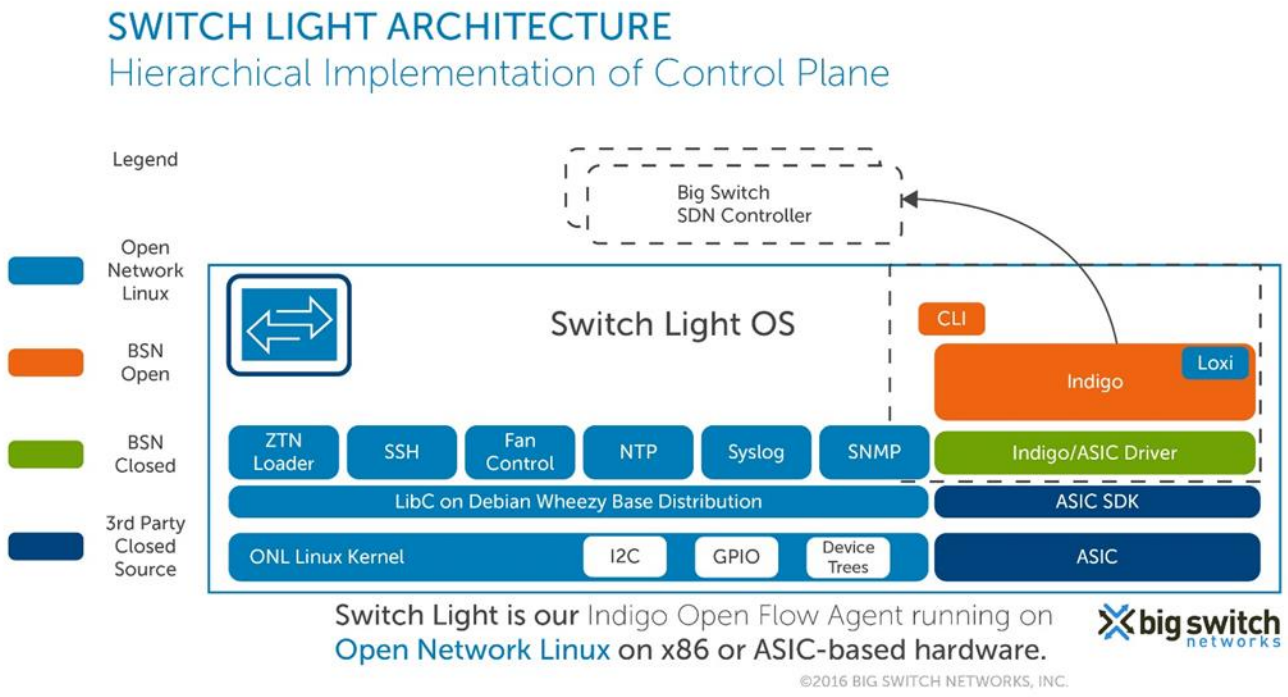

SWITCH LIGHT ARCHITECTURE
Hierarchical Implementation of Control Plane
Legend
Open Network Linux
BSN Open
BSN Closed
3rd Party Closed Source
Big Switch SDN Controller
Switch Light OS
CLI
Indigo
Loxi
ZTN Loader
SSH
Fan Control
NTP
Syslog
SNMP
Indigo/ASIC Driver
LibC on Debian Wheezy Base Distribution
ASIC SDK
ONL Linux Kernel
I2C
GPIO
Device Trees
ASIC
Switch Light is our Indigo Open Flow Agent running on Open Network Linux on x86 or ASIC-based hardware.
big switch networks
©2016 BIG SWITCH NETWORKS, INC.

Figure 2-3 Big Switch Networks - Switch Light OS Decomposed

The centralized controllers also leverage OpenFlow's southbound API to program the data plane with the necessary flow information to implement the policies that are instantiated by the control program. Because the Switch Light OS completely displaces any other software on the switch, it does not have to share the hardware tables with any other protocol data structures. That, coupled with the fact that newer versions of OpenFlow (1.3 and higher) allow for the use of multiple flow tables, means that BCF can operate in **proactive** OpenFlow mode, thereby allowing the fabric to scale.

The data plane is a leaf and spine (Clos) topology that – as of the writing of this paper – scales to 32 leaf switches (48 x 10G ports typically) and 6 spine switches (32 x 40G typically). The best way to think of the BCF architecture is to imagine a decomposed chassis-based switch with a pair of supervisor modules (aka Route Processor Modules, L2/L3 engines, etc.) and line cards. The BCF centralized controllers would be the equivalent of the redundant supervisor modules (active/standby), the leaf switches would comprise the 10G access line cards, and the inter-switch links and spine switches would comprise the non-blocking backplane. See figure 2-4 below. The L2/L3 boundaries of the Clos network are arbitrary and policy-driven, not topology driven.

In other words, the inter-switch link ports are non-committal in terms of their function; they may be part of a L2 or L3 forwarding mechanism.

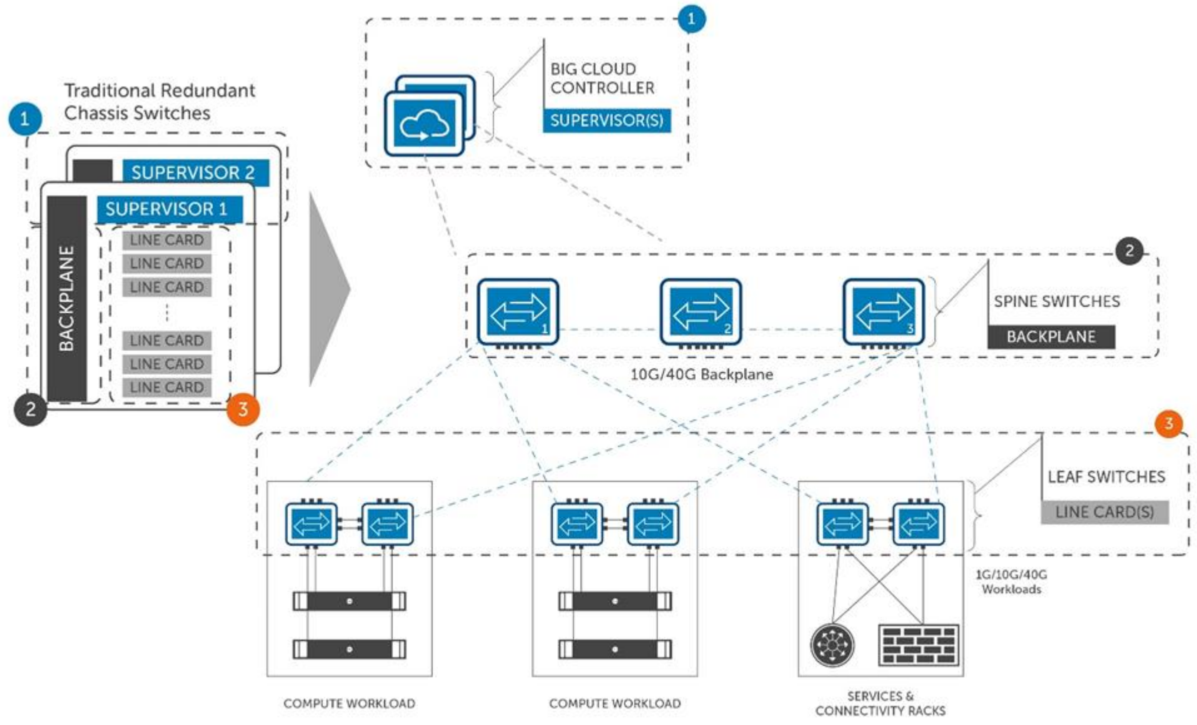


Figure 2-4 Big Cloud Fabric from Big Switch Networks. Comparison to Chassis-based switch.

As mentioned in the previous section, network virtualization is one of the prime use cases for SDN. In the BCF solution, a virtual network, which is comprised of L2 and L3 entities, is called a Tenant. The L2 broadcast domain (aka VLAN) of the Tenant network is referred to as a Logical Segment and the L3 component is called a Logical Router, which is also described as a VRF or Tenant Router. Tenants communicate with each other and with external devices through another L3 construct known as a System Router. Tenant routing – meaning between and within a segment – takes place at the leaf/access layer, which acts as a distributed router in hardware in which each leaf acts as the default gateway for any logical segment that it hosts. Inter-Tenant and external routing take place at the spine.

Starting with version 2.6 of the Switch Light OS, BCF can integrate with vMware NSX to provide visibility to the network underlay on which the NSX overlay tunnels "ride," as well as analytics to gage performance and reliability. By consuming an API provided by vCenter, the BCF controller can learn of the activity occurring on the overlay (i.e., which ESXi hosts are connected to which leaf switch and the virtual networks that they host, the creation of VTEPs, visibility to the VMS connected to the VTEPs, and virtual overlay network troubleshooting and analytics). This is what was being referred to earlier in this paper with regard to the value of integrating between the underlay and overlay networks.

BCF also integrates with OpenStack Neutron via a user space software agent for KVM-based virtual switches that adds functionality and is said to enhance performance of Open vSwitch (OVS).

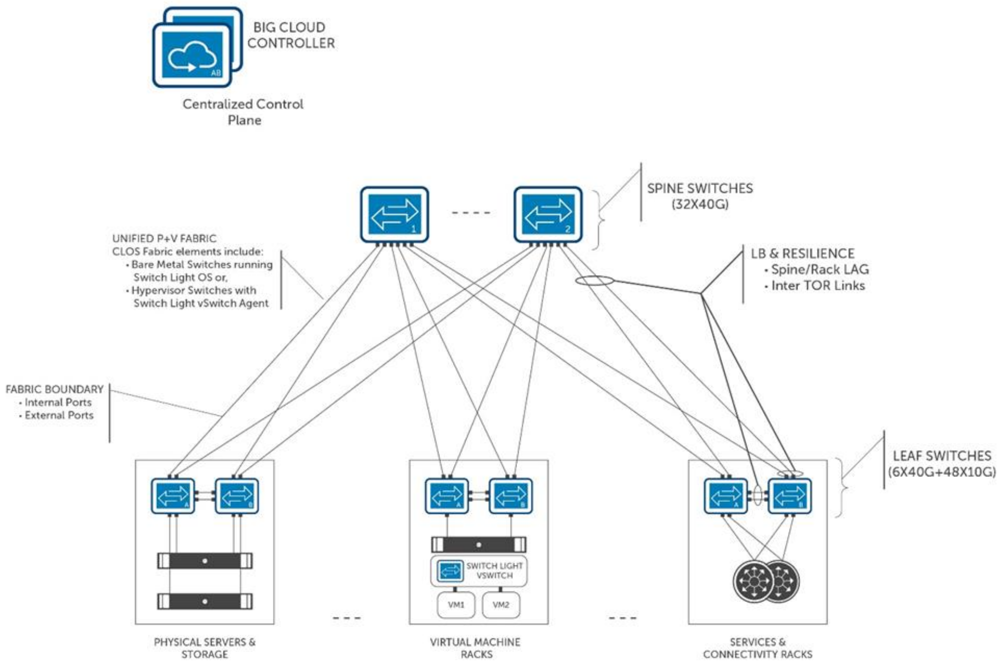

BIG CLOUD CONTROLLER
Centralized Control Plane
SPINE SWITCHES (32X40G)
UNIFIED P+V FABRIC
CLOS Fabric elements include:
• Bare Metal Switches running Switch Light OS or,
• Hypervisor Switches with Switch Light vSwitch Agent
LB & RESILIENCE
• Spine/Rack LAG
• Inter TOR Links
FABRIC BOUNDARY
• Internal Ports
• External Ports
LEAF SWITCHES (6X40G+48X10G)
SWITCH LIGHT VSWITCH
VM1
VM2
PHYSICAL SERVERS & STORAGE
VIRTUAL MACHINE RACKS
SERVICES & CONNECTIVITY RACKS

Figure 2-5 Big Cloud Fabrics Architecture Overall View

2.3.2 Big Monitoring Fabric (BMF)

The BMF solution from Big Switch Networks involves the deployment of a similar architecture to the BCF solution, but it is used as a means to interconnect a production network to a separate fabric whose only purpose is to provide "pervasive visibility and security." In other words, the network itself is a packet broker. It uses the same Switch Light OS with some modifications, redundant controllers and an unfolded Clos architecture. The entire solution can also be thought of as a big switch chassis to which the production network's monitoring taps and SPAN ports are connected, as well as the monitoring tools and performance analyzers. BMF leverages the ability of OpenFlow to match a wide scope of abstracted control plane information to capture traffic flows and forward them programmatically to a repository (farm) of network monitoring tools.

Enterprises who want to test the SDN waters may leverage this solution as a low-risk opportunity to introduce SDN into their environments and learn about how to leverage its capabilities and how to manage and maintain them while keeping their existing legacy production network in place.

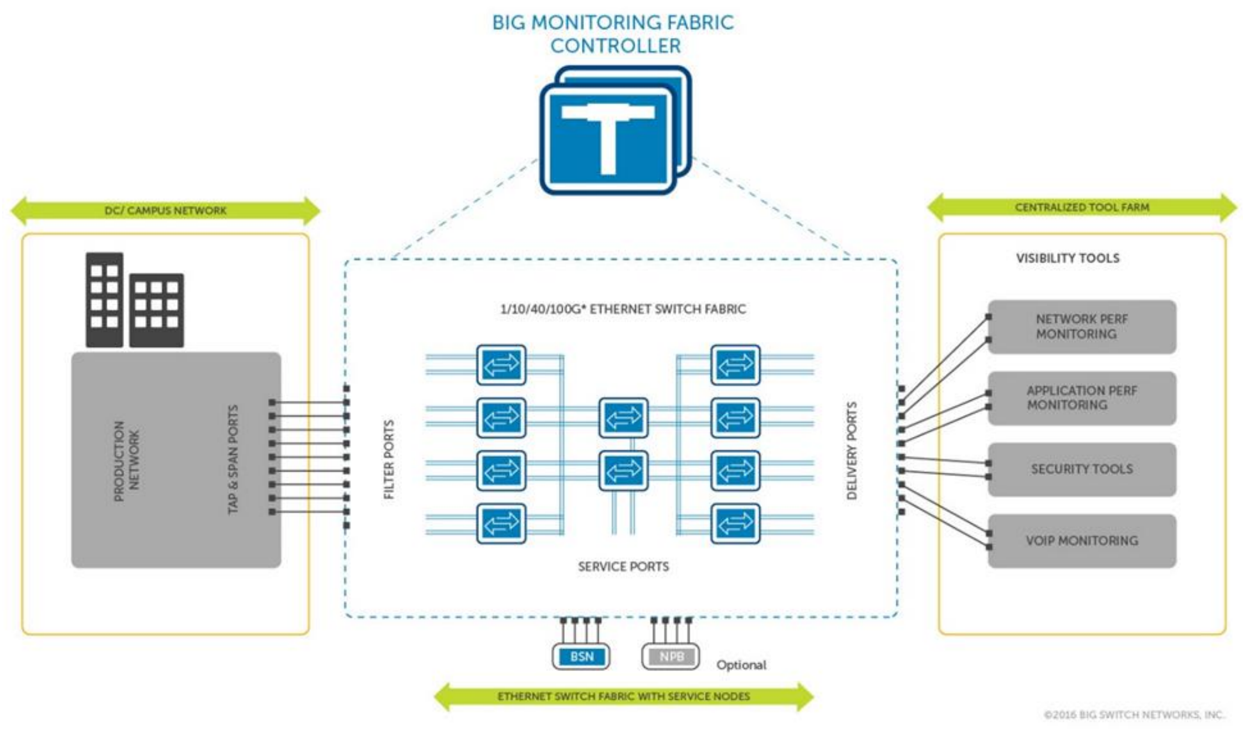


Figure 2-6 Big Monitoring Fabric Architecture Overall View

2.4 IP Infusion

OcNOS is the name of the operating system from IP Infusion that can be loaded onto a Dell Networking ON-based switch. The main attraction for this operating system is the vast MPLS-based services that it provides. This enables Dell to offer commoditized WAN edge and service provider based solutions. MPLS is a packet-switched data transport technology that is typically leveraged by enterprises to connect a data center to remote sites or as a data center interconnect (DCI) solution. MPLS can offer L3 or L2 connectivity, in which case, in its simplest form, an MPLS service can be thought of as a big switch (L2) or router (L3) that sits between sites.

OcNOS offers a relatively newer MPLS-based control plane solution known as EVPN – or Ethernet Virtual Private Network. One use case that is of particular relevance in the field of virtual networking is as a control plane for VxLAN. By default (according to IETF RFC 7348), VxLAN uses a multicast-based flood-and-learn approach (data plane learning) to VTEP and endpoint discovery. The overlay broadcast, unknown unicast, and multicast traffic (BUM) is encapsulated into multicast VXLAN packets and transported to remote VTEP switches through the underlay using multicast forwarding. Flooding in such a deployment can present a challenge for the scalability of the solution. The requirement to enable multicast capabilities in the underlay network also presents a challenge because some organizations do not want to enable multicast in their data centers or WAN networks.

EVPN offers a standards-based control plane solution to the VxLAN data plane that is more intelligent, efficient and scalable. It leverages a newly-established MP-BGP address family (L2VPN) and NLRI for advertising MAC addresses and mapping them to IP addresses. EVPN inherently supports multitenancy, privacy and route isolation.

One can find a solutions guide for VxLAN and EVPN using OcNOS at the following link:

http://www.ipinfusion.com/sites/default/files/OcNOS%20Solution%20Guide_VxLAN-EVPN.pdf

Since EVPN is an evolutionary technology that relies on other foundational technologies, to fully understand EVPN, one should familiarize themselves with the following RFCs:

- RFC 4271 - Border Gateway Protocol 4 (BGP-4): https://tools.ietf.org/html/rfc4271
- RFC 4760 - Multiprotocol Extensions for BGP-4: https://tools.ietf.org/html/rfc4760
- RFC 4364 - BGP/MPLS IP VPNs: https://tools.ietf.org/html/rfc4364#page-15

Finally, the OcNOS management plane can support a variety of management interfaces, such as "industry-standard" CLI, SNMP, REST, NETCONF and SAF IMM-OI.

2.5 Pluribus Networks

Open Netvisor Linux (ONVL), which is based on Canonical's Ubuntu Linux distribution, is the name of the network switch operating system that Dell offers from Pluribus Networks. Pluribus' software-centric solution is called Virtualization-Centric Fabric or VCF. The fabric is typically a Clos architecture whose switches are coalesced into a single management domain ("fabric") that can be managed through CLI or a C, RESTful API, as well as by DevOps tools, such as Ansible. Each switch runs an instance of a proprietary distributed database clustering software to create the management domain and it is used for configuration and state management across the physical network.

A VCF network does not leverage a centralized controller like BCF and BMF, nor is it an overlay solution, like NSX. Instead, it employs a distributed control plane across the fabric. All L2 and L3 control plane protocols, data plane forwarding mechanisms, multipathing and loop mitigation considerations are the same as they are in so-called legacy networks. The inter-switch links between leaf and spine may be L2 trunks or L3 interfaces. VCF has come a long way in its support for table stakes technologies and it continues to travel down that path, with support for VRF and other network virtualization technologies on the roadmap.

VCF's value comes from its ability to provide deep analytics and visibility (Insight Analytics) to existing and archived traffic flows across the network without having to deploy a separate tool farm, as one does with a typical packet broker-based solution. Analyzed production traffic flows "inline" and the analytics engines run on the switches themselves, with no need to purchase separate appliances. Traffic analysis can be done via a GUI or through CLI.

Specifically, the fabric analytics engine provides the following visibility:

- Telemetry – inspects every individual TCP connection and client-server aggregated connection fabric statistics.
- vFlow - filtering fabric-wide data center switching traffic on a granular flow level and applying security/QoS actions or forwarding decisions on each defined flow.
- vPort - tracking endpoints/VMs on a global, fabric-wide endpoint table.

Conclusion

This paper aimed to provide the necessary background, technical information and historical context to understand and appreciate SDN's foundational concepts and their relationship to Open Networking. While other vendors focus on a particular solution set and approach to delivering programmatic networks, Dell Networking offers engineers and architects the ability to choose a path that best suits their needs and meets their requirements through its championing of the disaggregated hardware/software model.