# Software Defined Networking:

What it is and what you should know!

**Jeff Dixon**
**4/5/2016**

**Table of Contents**

**Abstract**

It's well known that technology changes and creates new and increasing demands at an incredible pace. There have been few dramatic changes in how networks are designed over the years, however, that is quickly changing with the advent of Software Defined Networking (SDN). This paper will serve to provide a comprehensive guide to explore and explain the advancements of Software Defined Networking. The purpose is aimed to further inform and educate readers about SDN, what it is, and what you need to know moving forward, learning today for the challenges of tomorrow.

Industry respected journals, presentations, and various other sources will be reviewed and discussed in order to provide the greatest breadth of information possible in exploring this topic. SDN brings many exciting changes to the current capabilities of networking and looks to solve many of the problems that are currently faced. Networks are becoming larger and smarter and SDN provides a new means to build and support these growing networks, much like the way we now virtualize servers. We will also cover the differences with SDN as opposed to Network Virtualization (NV) and Network Function Virtualization (NFV), as well as other related technologies and how each falls into place.

While the networks of the past have been able to scale to unimagined levels, a more efficient and innovative technology is needed to keep up with the continued growth and demands put forth.  Networks have begun to shift toward SDN to fill these needs and it's highly likely that we will see this type of networking being put into place more every day as it matures and improves. Anyone either just getting into the networking field or already there will benefit from establishing a better understanding of this technology which is the primary goal of this paper.

**Chapter 1: Introduction**

Software Defined Networking, it sounds interesting but do you really know what is it, how it works, or how you can benefit from it? Whether you're starting from ground zero or you already have a good working knowledge, there is sure to be something for everyone. This paper was composed using numerous resources to develop a full and detailed insight and understanding of the ever evolving world of Software Defined Networking (SDN). The purpose of this study was to answer the following questions: what is Software Defined Networking, how SDN is designed and how it operates, what benefits and new possibilities are introduced, and some of the various technologies that exist around SDN. The significance of this study was in bringing together such a large array of information and resources to inform readers about SDN and provide additional insight among all skill levels. While vender neutrality and standardization are significant concepts entwined with SDN, specific definitions of what qualifies as SDN has changed and varied somewhat between vendors. SDN technology is still somewhat in its infancy and is always evolving. Never-the-less, the basic underlying concepts that SDN is built around will remain the same. Organizations are still working on finding new and expanded ways to develop these solutions and as they grow we will begin to see it deployed more frequently and in more locations.

**Chapter 2: SDN Explained**

Let us start by answering the most fundamental question. what exactly is Software

Defined Networking? Software Defined Networking (SDN) is most commonly defined as a

network that has decoupled the control plane from the forwarding plane [1]. To describe this in

another way, the network is controlled and managed by a central software platform, separate

from the hardware that is forwarding the traffic. Traditionally, each network device contains

both a control and forwarding plane. This means it must be configured and perform

independently of any other device on the network. While this method works well, it has

numerous drawbacks and below are some of these limitations:

- Management is complicated, tedious, and error prone. Every device is configured separately.

- Management complexity increases as the network grows.

- Network intelligence is limited at best, each device has little to no knowledge of the overall network flow or the specific needs of applications passing traffic

- Typically lacks any network automation capabilities.

SDN is posed to help remove and reduce these problems as well as bring many new capabilities

to our networks. Complex tasks that were once done manually or through limited and difficult

management tools can now be accomplished with great speed, automation, and efficiency

through SDN [2]. The main goal behind SDN was to create an agile and flexible network that

could accommodate rapid changes based on the needs and demands of businesses and

applications, all from a centralized programmable platform [3]. Another major concept of SDN

was that it was based upon an open platform strategy. The idea was to create a standardized

technology that works across different network hardware and software systems, encourages

innovation, and increases flexibility [2]. This helps to remove vendor lock-in and removes the need to understand how to program different command sets for each vendor. In addition to simplifying management and helping to open the door to new hardware solutions. SDN allows you to have a controller from company A and manage network devices, be it generic white boxes or various other devices, from different companies. This allows you to choose different vendors for management apart from the hardware passing your traffic. This open platform creates many advantages, cost savings, performance capabilities, and other aspects that were previously not possible.

Starting off, let's look at how traditional networks differentiate from SDN and their planes of operation. All networks, traditional or SDN, functioning at layer 3 (L3) have two primary planes of operation. The control plane and the data plane. The control plane is in essence the brains of the device and makes all the important decisions about traffic flow. The control plane is responsible for transmitting routing information in addition to building the routing and ARP tables [4]. Routing is done with static routes, Interior Gateway Protocols (IGP) such as OSPF, EIGRP, IS-IS, and Exterior Gateway Protocols (EGP) typically BGP. These protocols are able to establish adjacencies and use various algorithms to determine the path traffic should take [5]. With traditional networks each layer 3 device independently operates its own control and data plane as seen in Figure 1. Since the control plane operates individually on each device it is not able to obtain a complete view of the entire network.

**Figure 1. Traditional Network Device**
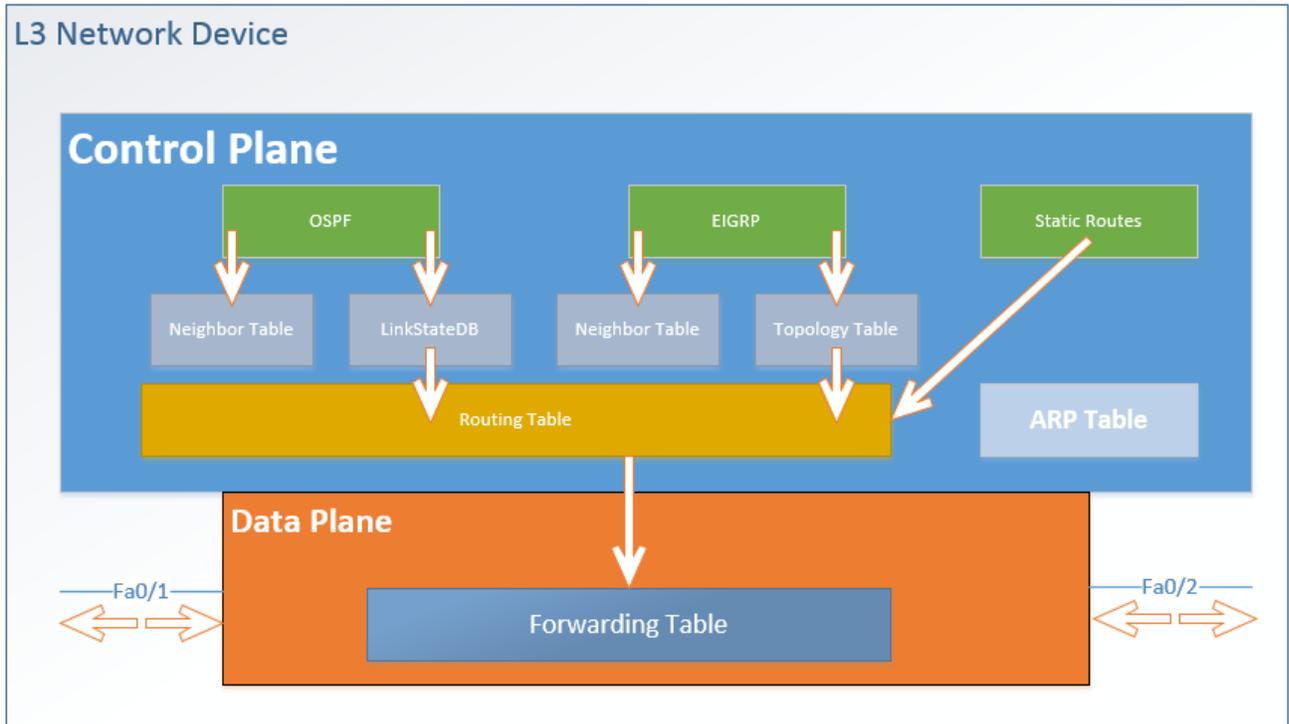


Depending on the routing protocols in use layer 3 devices may, at best, have limited knowledge

of its neighbors, devices directly attached, and limited awareness of networks in the same area

or autonomous system. With SDN, the control plane has now been separated for

programmability and management. This separation is the most significant aspect in terms of

how SDN's differ from traditional networks. SDN's control plane is logically centralized in

software, running on one or more controllers and is able to see and manage traffic flows for

every device and interface that is a part of its network. Changes to networks, VLAN's, routing,

traffic flows, QoS, etc. are able to be made quickly and even automated across all devices from

a central point of control. Changes that would take days or weeks, sometimes longer, can now

take place in minutes or seconds. Complexity is reduced and the need to independently

configure each device is removed. The control plane sends the information needed to create

forwarding tables and topology updates, from a network wide perspective, on to the data plane as it occurs [5].

The data plane, also known as the forwarding plane, is where the bulk of the work happens. That is to say, the majority of all packets pass only through the data plane at wire speed along with the help of application-specific integrated circuits (ASICs). It's also responsible for managing quality of service (QoS), filtering, encapsulation, queuing, and policing [5]. The data plane is sometimes considered the "fast path" due to the speed at which it forwards packets, while the control plane, in turn, would be considered the "slow path". This is due to the additional overhead in route processing that occurs at the control layer [5]. Because the data plane passes the majority of the traffic and the control plane creates relatively few changes, the performance requirements in the data plane are significantly higher. ASICs, which are purpose built circuits, are built into network hardware to allow the data plane to meet these demanding requirements. Although able to run on some existing hardware, SDN opens the door to moving toward white box style, generic hardware, for processing in the data plane.

Though not always depicted or mentioned, the management plane is another layer that is always present. The management plane is exactly what it sounds like, it provides access to manage a device or interface. Typically, a protocol such as telnet, SSH, SNMP, etc. is used to access, configure, monitor, and/or troubleshoot. Many of these protocols that you know and love are still present in SDN. Yet, as new management protocols emerge and further develop we may start to see some of these older protocols no longer in use. Some of the newer management protocols SDN has brought include OF-Config used by OpenFlow and Netconf, as well as others not mentioned. In traditional networks, the management plane is contained at

the hardware level. With SDN, we see the management plane present at the forwarding, control, and application planes. Since the planes are separated, this creates multiple new management points that didn't exist prior. Figure 3, which appears later in the section on architecture, better depicts the management plane and its function in an SDN environment. Though present at each plane, the majority of management functions are set to take place at the SDN controller level. The idea is to remove as much, and possibly all, of the management that was previously done at the hardware layer and move the management into software via the centralized controller(s).

When looking at the network architecture, SDN changes things up from the traditional design. To note, there are slightly different architectural designs for SDN available depending on the source. I will describe most common and standard representation, however, you may see vendors that have chosen to implement this in a modified or hybrid approach. Figure 2 below is an architecture from the Open Network Foundation (ONF). ONF is the body that introduced and maintains the OpenFlow standard. One point of confusion that often occurs in these discussions is naming. In Figure 2, we see the bottom layer (plane) referred to as the infrastructure later.  However, this lower level is often shown as the data or forwarding plane. No matter the name, just think of this as where the hardware resides that is forwarding your traffic (routers, switches, etc.).

**Figure 2. ONF/SDN Architecture** [6]
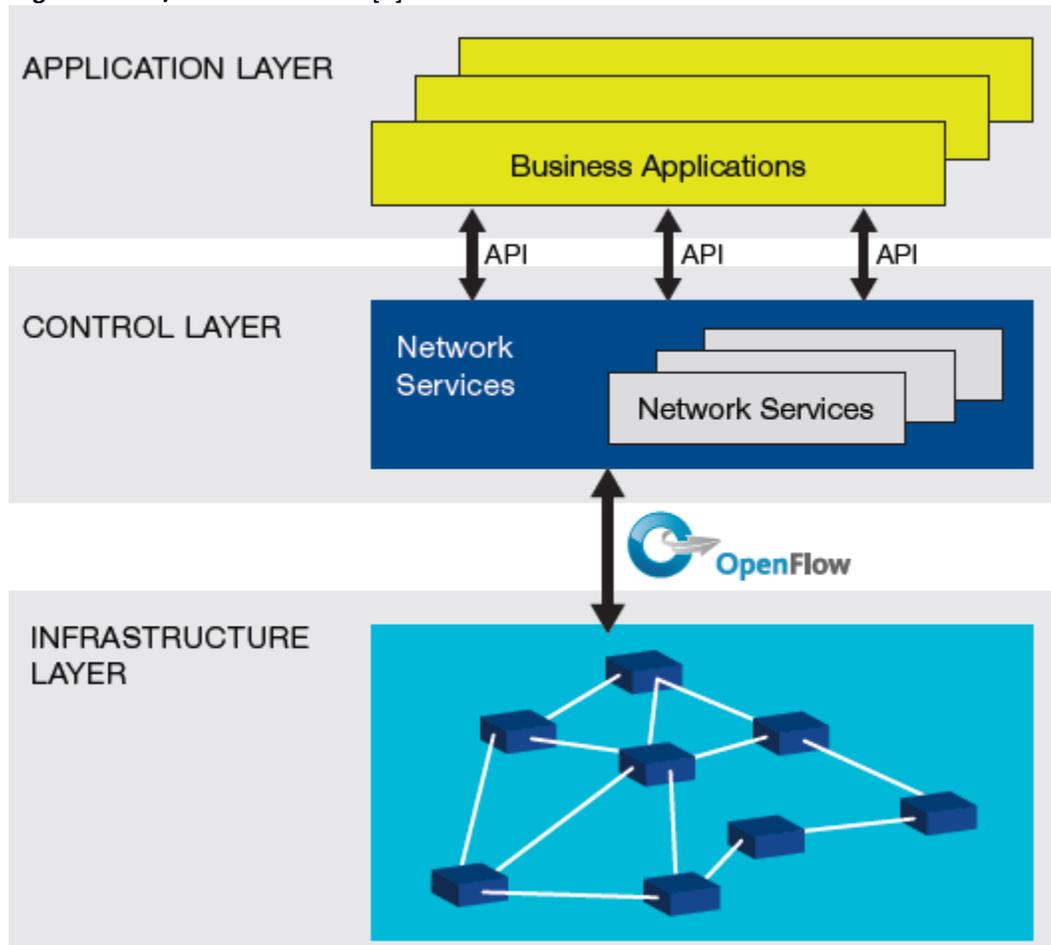


Moving up in the architecture, we have the control plane, where the SDN Controller resides.

The control plane is where all the magic happens! Applications pass requirements and changes

down to the controller, where it complies by managing the flow of traffic and device

configuration at the forwarding plane. The controller acts as the brains of the network, sending

and receiving data between every plane to create a highly adaptive and programmable

network. The controller provides a logically centralized entity, that may consist of one or more

physical or virtual instances, yet operates as a single unit [7]. A controller platform typically

uses a collection of "pluggable" modules that allow it to perform various network functions.

Additional modules can then be added or changed to enhance the capabilities of the controller. [8] OpenDaylight (ODL) is a popular example of an open source SDN controller. The ODL project is sponsored by many of the top networking vendors in the industry [9]. ODL is often used with OpenFlow for the southbound API but supports several others as well, including Netconf. Another essential function of controllers is that they help to translate business policies into network policies [2]. This is helpful because business policies are typically not written in very specific, technical terms. A business policy may be formed in the realm of, ABCOrg permits the telephony system to run from the cloud.  It wouldn't, however, specify the specific requirements necessary to make this happen.  IT policies on the other hand, are typically very technical. These policies normally contain very specific statements, such as, allow traffic from IP 172.16.0.1 on port 8443, block inbound ICMP, or assigning voice traffic to VLAN100 with a DSCP value of 46.  Business policies can be set with the aid of applications (in the application plane) that pass their requirements down to the controllers (control plane) through API's. The controllers then help to interpret these into specific changes that are applied to flows and configurations in the forwarding plane.  Additionally, controllers seek to bridge the gap between open, programmable network elements, and the applications that communicate with them. They help to automate the setup and management of the entire infrastructure, including the network, services, and applications. [2] The application layer or application plane, present only in SDN, sits above the control plane. In the application plane, programs are added that are able to communicate behaviors and resource requirements to the SDN Controller, as well as, build abstracted views of the network from collected information [10]. Between each of these layers or planes is a northbound and southbound API, also referred to as an interface. The

Northbound interface communicates between the control plane and the application plane [7]. These are arguably the most critical APIs in the environment. This is because the northbound interface directly facilitates the application's capabilities, this is a central feature that brings much of the value to SDN [11]. There are multiple northbound APIs in use at this time, as the rush to innovate and develop new functionalities, along with heavy competition, sometimes drives vendors away from standardization. Ideally, as the technology matures, companies may standardize on a single API. The southbound interface communicates between the controller (control plane) and the networking hardware (forwarding plane). [7] Although many protocols exist, OpenFlow was the first and is the most well-known southbound interface [12]. Additionally, north and southbound API's are both expected to be implemented in an open, vender-neutral, and interoperable manner [13]. While not often mentioned, there are two more API's worth noting that can be present. The westbound API, which acts as a conduit between the control planes of different network domains [14]. Followed by the eastbound API, which essentially is a means for the control plane of an SDN domain to communicate with the control plane of a non-SDN domain (e.g. MPLS) [7].

Expanding on the main architecture components we've covered thus far we can see their interworking's in more detail as shown in Figure 3. As discussed previously, we can see how the management plane runs vertically beside each of the data, control, and application layers. At the application layer we have the individual SDN applications. These each have a programmed logic containing SDN behaviors and requirements along with one or more Northbound Interfaces (NBI). The applications use the NBI to communicate down to the SDN Controller and specifically to a NBI Agent. Additionally, the controller contains control logic for

various tasks and capabilities and a southbound interface (CDPI). The southbound interface enforces behavior down to the network elements in the data plane and reports back status updates and configurations. Inside the data plane, resides a datapath. This is a logical device that may contain one or more physical devices. Here the southbound interface terminates at the agent and forwarding devices report and perform as instructed from above. [13]

**Figure 3. SDN Architecture ONF detailed** [13]
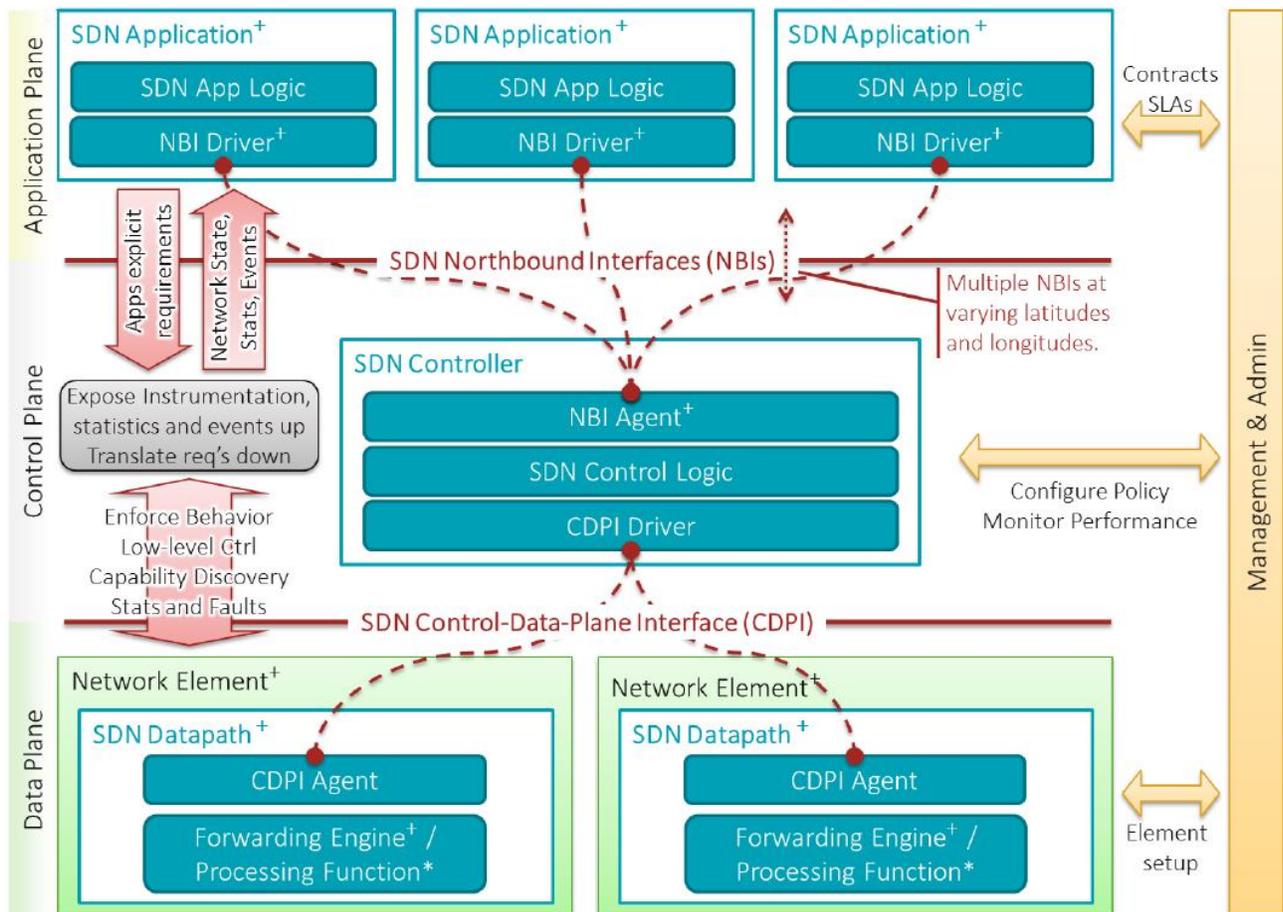


One additional concept to address before moving forward is SDN Orchestration. SDN orchestration "is the ability to program automated behaviors in a network to coordinate the required networking and software elements to support applications and services." [15] Infonetics Research published a great paper in Feb 2015 entitled "The Evolution of SDN and

NFV Orchestration" The link to the paper is listed in the bibliography [16] and is highly recommend. It offers a detailed insight into the current and future of SDN Orchestration. In their research, orchestration is described in this way: "Orchestration increases automation by coordinating the interactions and services flows among various parts of the network, giving guidance to other orchestrators and controllers." [16]

With an established groundwork for what SDN is, along with how it's structured, let us review in more detail the uses, benefits, and possibilities that have everyone so excited about this paradigm shift in networking.

Some of the key benefits provided by SDN include:

- Allows centralized automation of provisioning, configuration and remediation tasks across the network. [2]

- Accelerates application deployment and delivery, dramatically reducing IT costs through Policy-enabled workflow automation.

- Enables cloud architectures, adding security, automation, and performance [17]

- Creates a standardized way of communicating across the network, vendors, and device types. [18]

- Supports dynamic movement, replication and allocation of virtual resources [19]

- Eases the administrative burden of configuration and provisioning of functions such as QoS and security [19]

- Able to more easily deploy and scale network functionality [19]

- Able to perform traffic engineering with an end-to-end view of the network [19]

- Allows network functionality evolve more rapidly based on a software development lifecycle [19]

- Implementation of more effective security functionality [19]

- Reduces complexity [19]

- Networks can be treated as a single programmable entity [7]

- Centralized management through controller software with a global view of the network. [3]

- Reduction in operating expenses due to automation and reduced management time. [3]

- Enables innovation by allowing new types of applications, services, and business models adding value and potentially new revenue streams. [3]

The use cases for SDN at times seem almost unlimited! New ideas for SDN are being thought up and developed every day. Never-the-less, popularity and hype alone are typically not enough for most companies to make such a significant shift in networking strategies. Tangible, quantifiable, and proven benefits, such as ROI, are needed for these technologies to be embraced. [2]

Below are just a few of the more common use cases for SDN:

- Automating QoS network wide for unified communications sessions [17]

- Dynamically identifying trusted traffic flows at the network edge to improve network performance and right-size costly service appliances by routing the flow around stateful services such as firewalls, IPS sensors, or network address translation. [20]

- Automated provisioning of network bandwidth to accommodate scheduled data transfers. [20]

- Isolating network flows based on controller policies to segment traffic. [20]

- Simplifies the convergence of multiple data centers from different institutions with programmatic controls and service orchestration. [20]

- Reduce security threats that emerge. An application with packet inspection could work to detect malware and automatically quarantine the device to a designated network segment. [20]

- Dynamically encrypting select traffic flows to protect sensitive information. [20]

- Implementing applications that determine a user's service tier and adjusts QoS markings for the appropriate traffic class. [20]

- Extending routing protocols to consider not just path selection parameters like speed but also the actual costs of using a particular path over another to reduce or contain costs. [20]

- Carrier and service providers can off bandwidth on demand. This could allow carrier links to request additional bandwidth when needed and provide WAN optimization and bandwidth calendaring. [3]

- Creating self-healing networks by dynamically moving traffic to a new device/path when a fault is detected and automatically redirecting traffic back once the fault is resolved. [18]

- Facilitate development and operations (DevOps) through the automation of application updates and deployments and the automation of IT infrastructure [2].

Of the many benefits and shifts in SDN, management is clearly one of the more significant. While SDN has many advantages in this arena, it also creates several new management considerations and challenges that should be acknowledged. From a management perspective, SDN requires, not all, but many of the same demands as traditional networks. However, SDN adds several new aspects that must also be addressed. If that's the case, how is management simplified, easier, and faster? Although SDN does add new management points, the centralized management, speed, programmability, and numerous other benefits far exceed the addition of these new matters. It is important, however, that they not go overlooked. IEEE Xplore recently published a journal covering the management requirements and challenges for SDN which describes some of the differences in traditional and SDN management [21]. This next section highlights some of these key considerations:

| Traditional Network | Software Defined Network |
|---|---|
| Uses well known protocols to manage and configure each device and track changes | Configurations change rapidly and can be comparatively complex. Management interfaces should be included at each plane for configuration and troubleshooting. |
| Fault tolerance enabled by providing alternative and/or redundant routes and links. | The forwarding plane still demands redundant or alternate paths. However, since the control plane is separated, one must now consider how to handle the loss of connection to the control plane. If the controller is a single node, there is a single point of failure. If multiple nodes exist, configuration must be made to ensure a graceful connection to an alternate controller during failure. |
| Traditional networks are not considered programmable networks (PN). PN's allow software independent from the network hardware to control device behaviors and traffic flows. | Network programmability is a key feature of SDN. To maintain this capability, proper tools and methods should be set to control versioning, coordinate deployments, and provide verification of network software. |
| Uses Quality of Service (QoS), bandwidth and reservation configurations to manage traffic performance. | SDN must also consider the performance of network applications and potential bottlenecks or delays that may occur from passing traffic from separated planes. A control model must also be decided on that best suits the network (e.g. centralized, distributed, or hierarchical.) |
| Security must control and protect network access to and from devices and locations. It should also try to prevent intrusions, spoofing, DoS, and similar attacks | Resource isolation and security between planes needs to be considered, particularly control traffic. Ensure network applications do not conflict with one another and provide isolation. |
| Management of some higher level protocols may be adjusted | SDN should provide highly flexibly management interfaces or systems at each plane for quick adaptability |
| Network planning focuses on aspects such as need for L2 or L3 device, | SDN Planning has somewhat different concerns from traditional networks. Software used may be a key |

| capacity and performance needs, and topology location. | consideration to ensure compatibility and functionality. Hardware will typically be either a generic forwarding device or a control device if needed. The management plane may also offer tools that assist in planning |
| --- | --- |
| Monitoring focuses on resource utilization, outages, and alarms. Traffic visualization is possible due to well-known predictable routing behaviors. | Has same focus as traditional networks but at every level including applications. However, with new SDN forwarding and control protocols, behaviors may become less predictable and present new challenges in monitoring and visualizing network activity. |

**Chapter Three: Open and Related Technologies**

Next we're going to dive into two very closely related technologies: Network Virtualization (NV) and Network Function Virtualization (NFV). While these may be implemented on their own, more often than not, we are finding them integrated as part of SDN solutions. This makes it an important item to understand when discussing SDN. Following this we'll finish up with a discussion of some of the popular open SDN protocols in use.

Network Virtualization (NV) and Network Function Virtualization(NFV) are not the same as SDN, yet we see them in many SDN implementations and solutions. Though they are closely related, they accomplish fairly different tasks from SDN. The industry often creates some confusion by using these terms interchangeably or by changing how they define them, however, after this section you should have a good understanding of each of these concepts. In one way or another, we have strived to create virtualization in networking for many years. We have several capabilities already at our disposal such as, Virtual LAN's (VLAN's), Virtual Routing and Forwarding (VRF), Virtual Port-Channels (vPC), and others. Before going much further, let's

set a definition for virtualization. "Virtualization is the basic act of decoupling an infrastructure service from the physical assets on which that service operates." [22] Although they operate differently, you will often see NV and even SDN likened to Server Virtualization. Without getting too deep, we can describe Server Virtualization in this way:  Server Virtualization (SV) uses a hypervisor running on a physical host, that in turn presents virtual CPU, RAM, NIC, Storage, and other resources to a virtual machine. Server Virtualization revolutionized the way we create and manage servers. In turn, it brought about many new technologies and services that were not previously possible. In this accomplishment, NFV and SDN are often predicted as being able to achieve the same advancements for networking. Network Virtualization (NV) is in many ways where things all started, leading up to NFV and SDN.  NV differs from SV in that it does not run a hypervisor directly on top of a physical network appliance such as a switch or router. Instead, NV encapsulates traffic in order to simulate a virtual networking platform. This comparison is illustrated further through Figure 4 below [23].

**Figure 4. SV vs NV** [22]
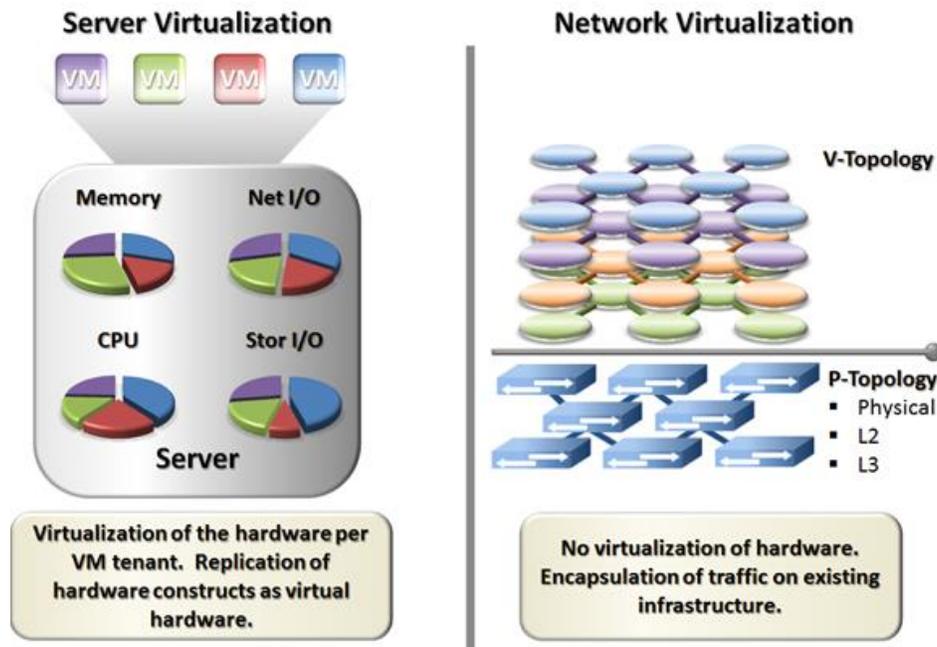
Moving on to the next evolution we have Network Functions Virtualization (NFV), which is essentially a progression of NV. Although not proper, you may see the terms used interchangeably. The aim of NFV is to virtualize layer 4 through layer 7 services. Network functions are virtualized info software applications that are able to run on, physical or virtual, x86 type servers. [24] The diagram in Figure 4 below helps to illustrate the idea behind NFV and compares it to that of traditional networks. We see in this depiction that the core and edge routers have been removed and the functionality has been enabled instead from a VM. NFV can reduce the need for dedicated network devices (routers, firewalls, etc) by allowing the functions previously performed by those devices (NAT, Firewall Rules, Routing, etc) to run on a server platform.

**Figure 4. SV vs NV** [24]
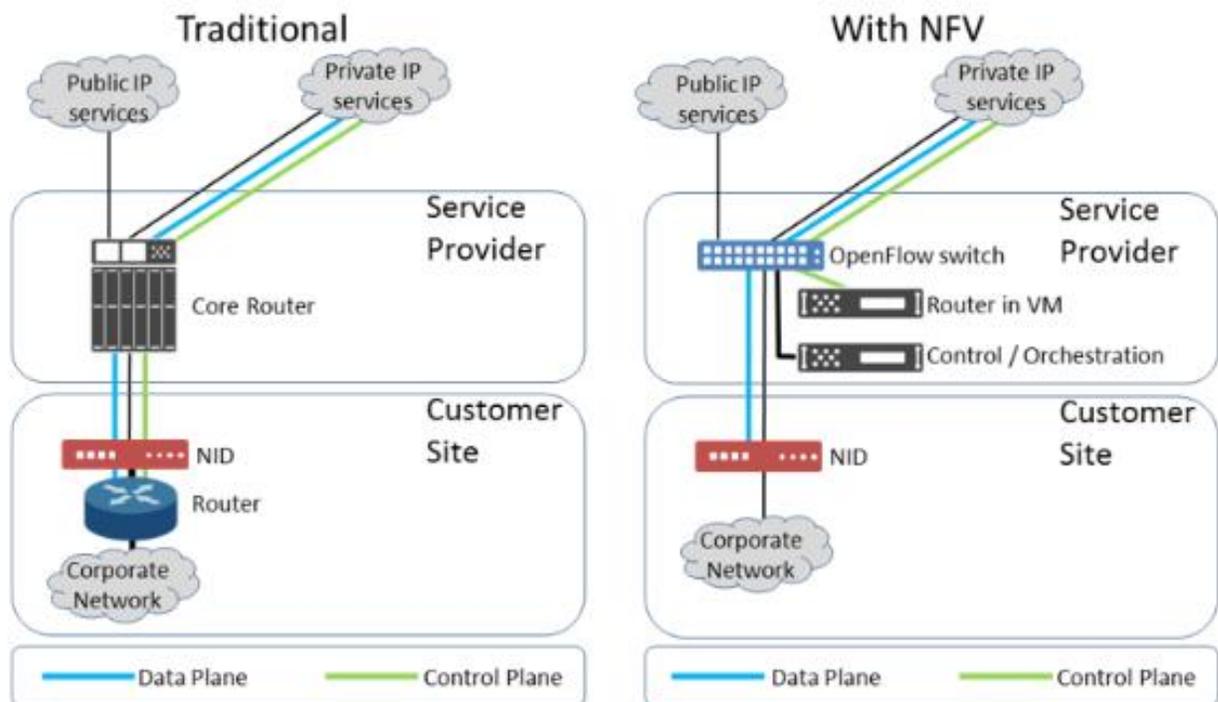
The need to constantly make changes to the network infrastructure and to test various solutions was the initial driver behind SDN [24]. As new applications like cloud and big data began to evolve, so did the need for more efficient east-west traffic between servers. SDN moved the control plane out and allowed for better performance for east-west traffic while reducing north-south traffic. A change that was able to greatly speed up operations and help meet the new network demands created in many industries. Steaming from a different set of demands, NFV was instead driven my major carriers that had difficulty scaling their networks due cost and complexity. [24] With NFV, the focus was on standardization and virtualization of devices. Whereas, SDN was focused on separating the control plane.

No matter which of these concepts we're discussing, NV, NFV, SDN, etc. there is a common drive to move away from solutions that result in vendor lock-in and more towards an open (nonproprietary) and standardized platform. In this idea, you could have a SDN solution from a particular vendor and manage switches, routers, etc from various other hardware providers seamlessly with a common set of controls, capabilities, and protocols. The thought behind this is that it will create better products, lower costs, drive more innovation, and create a higher return on investment. [2] While SDN is based around this idea of an open platform, we are not entirely there yet. Many solutions at this point may still require you to use their components if you want to use that vendor's solution. In addition, moving to SDN does not always mean you must forklift your network and start fresh. SDN can often run on your current infrastructure while adding new capabilities.

The most common SDN/NFV protocol in use at this time is OpenFlow. It should be clarified that OpenFlow itself is not SDN or NFV, it is simply one piece of the overall solution.

OpenFlow is maintained by the Open Networking Foundation (ONF) and was one of the first standard communications interfaces for use between the control and forwarding layers in SDN. [25] To describe it another way, OpenFlow is a protocol that allows the SDN controller to interact directly, with both physical and virtual, network devices (switches, routers, etc). [26] While other southbound protocols exist, OpenFlow is currently the only open standards-based protocol available. OpenFlow, as the name implies, uses flows to identify network traffic. These flows are based on pre-defined match rules that can be statically or dynamically programmed. This design creates a very granular control that enables the network to respond to real-time changes at all levels. [27] OpenFlow is supported by many of the major networking vendors at this time. If you would like to experiment with OpenFlow and SDN, there are some OpenFlow Network Simulator and Emulator applications available. The most popular among the research and education community is Mininet (http://mininet.org/) and NS-3 (https://www.nsnam.org/). These are free open source projects that can be used for SDN research and testing. Unfortunately, these are both noted to have a number of drawbacks and limitations. EstiNet (http://www.estinet.com) is a different commercial option and considered the best simulator available. More information about OpenFlow can be found at the ONF website (www.opennetworking.org).

OpenFlow isn't able to do everything on its own however, it utilizes two more protocols for configuration and management. These are the OpenFlow Management and Configuration Protocol (OF-Config) and the Open vSwitch Database Management Protocol (OVSDB). OF-Config uses a configuration point to issue commands down to an OpenFlow switch. The configuration point can be located with the SDN controller or within network management products.

Configuration points can manage multiple switches and switches can be managed by multiple configuration points. The configuration point provides each logical switch with the IP and port numbers of the controllers that will manage the device flows. In addition, it specifies if TCP or TLS will be used to communicate between the two devices and configures certificates for use between them. Configuration points can also perform other functions such as discovering resources that have been allocated to logical switches, configuring tunnels, setting port parameters, turning ports on and off, and retrieving switch status. If a configuration fails, it receives an error code and is able to roll back the configuration. [28] If you recall, OpenFlow operates only on flow tables, which allows it to specify how packets are routed. OpenFlow, however, is not able to make configuration changes to the devices itself. In order to make these changes OF-Config relies on the Netconf protocol (RFC 6241) when it needs to send and receive configuration data to the switch. To be noted, Netconf is also capable of running on its own from a controller. However, that design is outside of our scope at this time. The OF-Config/Netconfig function is illustrated further in Figure 5 below.



**Figure 5. Of-Config**
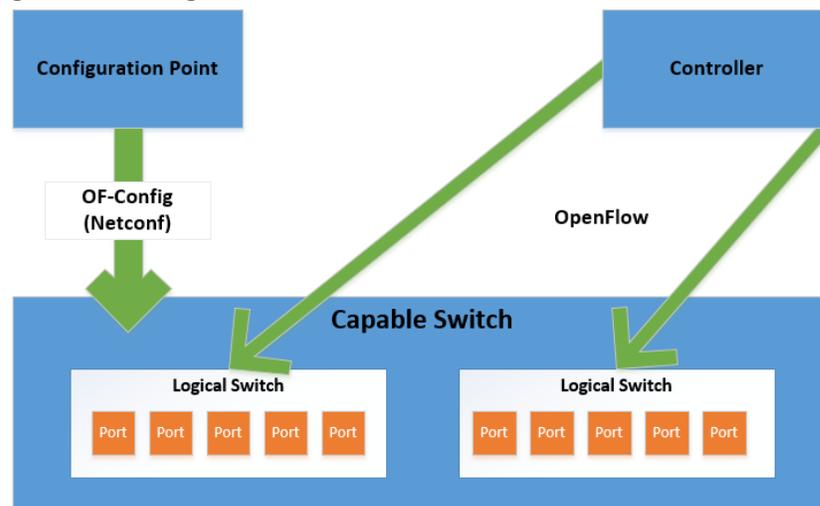
**Configuration Point:** Configures set of Capable Switches
**Capable Switch:** Managed entity containing set of Logical Switches
**Logical Switch:** Set of resources managed by controller

The second protocol, Open vSwitch Database Management Protocol (OVSDB), is an open source

protocol that was specifically designed for the management of Open vSwitch implementations.

[28] Open vSwitch is an open source multilayer virtual switch. [29] If you are not familiar with

virtual switches, they were essentially born from server virtualization. In order to present

network ports to a virtual server running in a hypervisor, a virtualized switch was created to

connect virtual NIC's from the VM and interface back to the physical network ports on the host.

Figure 6 shows this concept in more detail. Each VM Host would run a hypervisor, typically in a

cluster configuration. The hypervisor platform would present a virtual switch to the VM's to

allow connection from a vNIC. There are three main virtual switches in use today, VMware

virtual switch, Cisco Nexus 1000v, and the Open vSwitch (OVS).

**Figure 6. Virtual Switch**
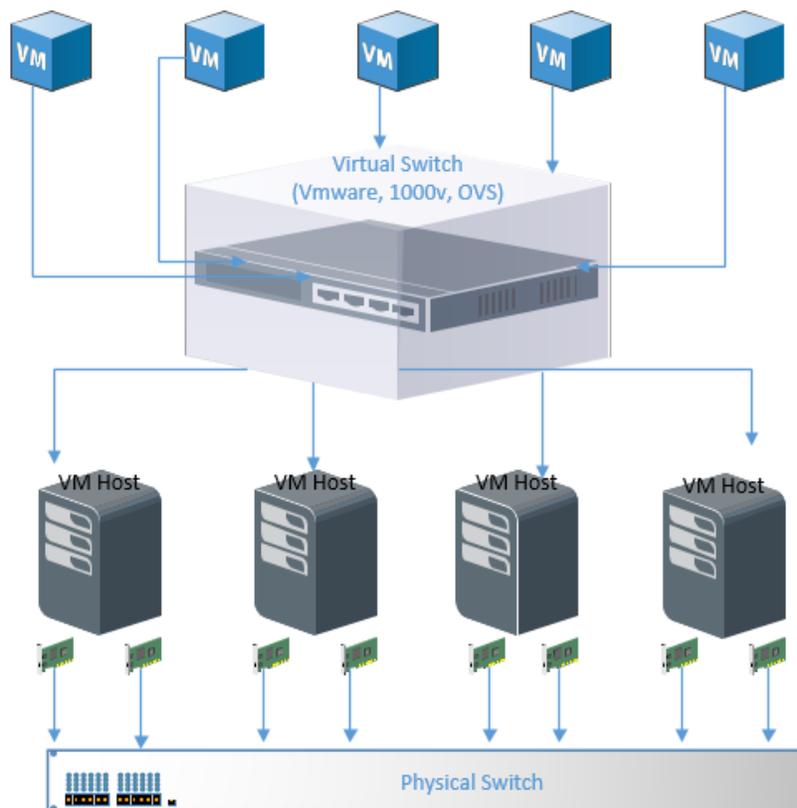
OpenFlow, OpenDaylight, and Open vSwitch, clearly the push is for open architecture and while these are not the only solutions in place, these currently remain main focus in SDN for open standards.

**Chapter Four: Conclusions**

There is no question that virtualization and software defined technologies are the way of the future. In fact, we are already seeing a concept described as Software Defined Anything (SDx) arise. Companies are striving to apply these concepts to all areas of compute, networking, storage, and security. Although it has yet to do so, it is very possible and believed by many that SDN will change the face of networking for everyone, just as server virtualization has done for compute. Solutions like Cisco ACI and VMware NSX are already taking SDN to new levels and expanding what SDN is and is capable of everyday. SDN's promise of an open sourced, vendor agnostic, centralized, and programmable software controlled network is certainly enough to get anyone excited about the possibilities and changes such a solution could bring. The possibility of a new landscape for networking is getting closer and one may suggest keeping a close eye as it continues to develop in the near future.

**Bibliography**

[1] N., & Sood, M. (dec 2014). Software defined network — Architectures. *Parallel, Distributed and Grid Computing (PDGC), 2014 International Conference,* 451-456. Retrieved April 4, 2016, from http://ieeexplore.ieee.org.jproxy.lib.ecu.edu/xpls/icp.jsp?arnumber=7030788

[2] Underdahl, B., & Kinghorn, G. (2015). Software Defined Networking FOR DUMMIES. Retrieved April 4, 2016, from http://www.cisco.com/c/dam/en/us/solutions/collateral/data-center-virtualization/application-centric-infrastructure/sdnfordummies.pdf

[3] What's Software-Defined Networking (SDN)? (n.d.). Retrieved April 04, 2016, from https://www.sdxcentral.com/resources/sdn/what-the-definition-of-software-defined-networking-sdn/

[4] CEF (Cisco Express Forwarding) | Networklessons.com. (2014). Retrieved April 04, 2016, from https://networklessons.com/switching/cef-cisco-express-forwarding/

[5] Salisbury, B. (2012, September 27). The Control Plane, Data Plane and Forwarding Plane in Networks. Retrieved April 04, 2016, from http://networkstatic.net/the-control-plane-data-plane-and-forwarding-plane-in-networks/

[6] OpenFlow-enabled SDN and Network Functions Virtualization. (n.d.). *ONF Solution Brief*. Retrieved April 5, 2016, from https://www.opennetworking.org/images/stories/downloads/sdn-resources/solution-briefs/sb-sdn-nvf-solution.pdf

[7] Jarschel, M., Zinner, T., & Hobfeld, T. (n.d.). Interfaces, attributes, and use cases: A compass for SDN. *IEEE Communications Magazine*. Retrieved April 4, 2016, from http://ieeexplore.ieee.org.jproxy.lib.ecu.edu/xpls/icp.jsp?arnumber=6829966

[8] What are SDN Controllers (or SDN Controller Platforms)? (n.d.). Retrieved April 04, 2016, from https://www.sdxcentral.com/resources/sdn/sdn-controllers

[9] What is the OpenDaylight Project (ODL) ? (n.d.). Retrieved April 04, 2016, from https://www.sdxcentral.com/resources/sdn/opendaylight-project/

[10] Understanding the SDN Architecture - Definition -. (n.d.). Retrieved April 04, 2016, from https://www.sdxcentral.com/resources/sdn/inside-sdn-architecture/

[11] What are SDN Northbound APIs? (n.d.). Retrieved April 04, 2016, from https://www.sdxcentral.com/resources/sdn/north-bound-interfaces-api/

[12] What are SDN Southbound APIs? - Where they are used. (n.d.). Retrieved April 04, 2016, from https://www.sdxcentral.com/resources/sdn/southbound-interface-api/

[13] SDN Architecture Overview. (n.d.). *Open Networking Foundation*. Retrieved April 5, 2016, from https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/SDN-architecture-overview-1.0.pdf

[14] Jain, S. (n.d.). B4: Experience with a Globally-Deployed. *ACM SIGCOMM Computer Communication Review*. Retrieved April 4, 2016, from http://delivery.acm.org.jproxy.lib.ecu.edu/10.1145/2490000/2486019/p3-jain.pdf?ip=150.216.68.200&id=2486019&acc=PUBLIC&key=A79D83B43E50B5B8.D2E2D13E69DBEDD9.4D4702B0C3E38B35.4D4702B0C3E38B35&CFID=597838163&CFTOKEN=18424837&__acm__=1459920010_78db92719215ba84d1499b5a979701ac

[15] What is definition of SDN Orchestration? (n.d.). Retrieved April 04, 2016, from https://www.sdxcentral.com/resources/sdn/what-is-sdn-orchestration/

[16] Howard, M. (2015, February). The Evolution of SDN and NFV Orchestration - juniper.net. Retrieved April 4, 2016, from https://www.juniper.net/assets/us/en/local/pdf/analyst-reports/2000604-en.pdf

[17] Tourrilhes, J. (nov 2014). SDN and OpenFlow Evolution: A Standards Perspective. *Computer,* 22-29. Retrieved April 4, 2016, from http://ieeexplore.ieee.org.jproxy.lib.ecu.edu/xpls/icp.jsp?arnumber=6965280

[18] Moore, P. (2015, March 4). What is the difference between HTTP and HTTPS? - Quora. Retrieved April 4, 2016, from https://www.quora.com/What-is-the-difference-between-HTTP-and-HTTPS

[19] SDN 101: An Introduction to Software Defined Networking. (n.d.). Retrieved April 4, 2016, from https://www.citrix.com/content/dam/citrix/en_us/documents/oth/sdn-101-an-introduction-to-software-defined-networking.pdf

[20] Software-Defined Networking ... - Cisco Systems. (2013). Retrieved April 4, 2016, from http://www.cisco.com/c/dam/en_us/solutions/industries/docs/gov/cis13090_sdn_sled_white_paper.pdf

[21] Wickboldt, J. A., De Jesus, W. P., Isolani, P. H., Both, C. B., Rochol, J., & Granville, L. Z. (january 2015). Software-defined networking: Management requirements and challenges. *IEEE Communications Magazine, 53*(1), 278-285. Retrieved April 4, 2016, from http://ieeexplore.ieee.org.jproxy.lib.ecu.edu/xpls/abs_all.jsp?arnumber=7010546

[22] Hedlund, B. (2013, March 28). What is Network Virtualization? Retrieved April 04, 2016, from http://bradhedlund.com/2013/05/28/what-is-network-virtualization/

[23] Onisick, J. (2013, June 08). What Network Virtualization Isn't. Retrieved April 04, 2016, from http://www.definethecloud.net/what-network-virtualization-isnt/

[24] Fruehe, J. (August 27). The Battle of SDN vs. NFV. Retrieved April 04, 2016, from http://www.moorinsightsstrategy.com/the-battle-of-sdn-vs-nfv

[25] OpenFlow. (n.d.). Retrieved April 04, 2016, from https://www.opennetworking.org/sdn-resources/openflow

[26] What is OpenFlow? Definition and how it relates to SDN. (n.d.). Retrieved April 04, 2016, from https://www.sdxcentral.com/resources/sdn/what-is-openflow/

[27] Pitt, D. (2015, April 15). Understanding OpenFlow, VXLAN and Cisco's ACI. Retrieved April 04, 2016, from http://www.networkcomputing.com/networking/understanding-openflow-vxlan-and-ciscos-aci/551182896

[28] Jacobs, D. (2013, April). OpenFlow configuration protocols: Understanding OF-Config and OVSDB. Retrieved April 04, 2016, from http://searchsdn.techtarget.com/tip/OpenFlow-configuration-protocols-Understanding-OF-Config-and-OVSDB

[29] Production Quality, Multilayer Open Virtual Switch. (n.d.). Retrieved April 04, 2016, from http://openvswitch.org/