ISSN: 2167-1435
v31 n3069

Software Defined Networking (SDN) Network Virtualization for the IS Curriculum?

Joseph Packy Laverty
laverty@rmu.edu

David Wood
wood@rmu.edu

John Turchek
turchek@rmu.edu

Robert Morris University
Moon Township, PA 15108, USA

Abstract

Network technologies, enterprise architectures, and cloud architectures are recommended topics in the IS curriculum. Over 70% of today's servers are virtual machines. Cloud computing is the ultimate "system" to teach in an information system curriculum. The TCP/IP and OSI network models were adequate to serve industry and the IS curriculum in an age when static, client-server architectures were dominant. Current network models and technologies used to manage large-scale, complex networks, integrate cloud architecture resources and multi-tenant applications (Open Networking Foundation, 2014a), provide no user program control or customized automation services, are vendor-specific, and fail to address the complexity of different traffic classes mix, mobile devices, BYOD, individual users or applications. This paper provides a) an overview of limitations of current network models and technologies b) an overview of new enterprise network and cloud network standards introduced in 2012, such as, Software-Defined Networking (SDN), and OpenFlow, and OpenStack, and c) considers the inclusion of these new open standards in the IS network curriculum.

Keywords: Software-Defined Networking, SDN, OpenFlow, OpenStack, Network Virtualization, Cloud Computing, IS Curriculum

1. INTRODUCTION

Data communication and networking are important components for ABET CAC Accreditation (ABET CAC, 2014) and the IS 2010 Model Curriculum (Topi, Valacich, Wright, Kaiser, Nunamaker, Sipior, & deVreede, 2010). The IS 2010 Model Curriculum Guidelines further recommend that the IS networking curriculum should be covered from a high-level of abstraction using an enterprise or IT infrastructure approach. Schünemann (2004) described abstraction and model-building as two related, important techniques for understanding complex domains of interest. Abstraction is a process of separating essentials from details by using various strategies, such as decomposition, classification or generalization. Conceptual Models, e.g., the TCP/IP and OSI network models, are one of many abstraction models used to classify components of real-world systems, connections and interactions (Schünemann, 2004).

The TCP/IP and OSI network models categorized network services and protocols using a layered taxonomy. Network protocols are mechanisms

ISSN: 2167-1435
v31 n3069

that define procedures and data to perform a specific network service without any benefits of fundamental abstraction, such as decomposition or top-down design. These traditional network models were adequate to serve industry and the IS curriculum in an age when static, client-server architectures were dominant.

While the IS2010 recommended an "enterprise" approach to the network curriculum, it did not define the concept of "enterprise". Enterprise technologies may include topics such as web and mobile applications, network services, transaction processing, web services, distributed databases. Cloud computing architectures are based on virtual machines (servers), have the ability to cluster processing resources, and share storage areas networks.

Current network conceptual models and technologies used to manage large-scale, complex networks, integrate cloud architecture resources and multi-tenant applications (Open Networking Foundation, 2014a), provide no user program control or customized automation services, are vendor-specific, and fail to address the complexity of different traffic classes mix, mobile devices, BYOD, individual users or applications.

Software-Defined Networking (Open Networking Foundation, 2014b) is an emerging open, enterprise-level network model that provides a framework to address the limitations of current network technologies and automate network management. Software-Defined Networking (SDN), OpenFlow, Network Functions Virtualization (NFV) and Network Virtualization (NV) are software-based abstractions that will permit enterprise and data center networks to be more scalable, programmable,dynamic, automated, and better aligned and support of business and IT requirements. (Open Networking Foundation, 2014c) (SDNCentral LLC., 2014a) (Pate, 2013), SDNCentral LLC, 2014b), (Openflow, 2014), (SDNCentral,2014c), (Cannistra, R. & DeCusatis, C.,2013).

OpenStack provides a framework to integrate network resources using SDN with compute and storage resources in cloud-based architectures (OpenSource.com, 2014a). Over 70% of today's servers are virtual machines (Seetharaman, 2013). Cloud computing is the ultimate "system" to teach in an information system curriculum.

This paper provides a) an overview of limitations of current network models and technologies, b) an overview of new enterprise network and cloud network standards introduced in 2012, such as, Software-Defined Networking (SDN), and OpenFlow, and OpenStack, and c) considers the inclusion of these new open standards in the IS network curriculum

2. CURRENT ENTERPRISE NETWORK TECHNOLOGIES AND LOGICAL MODELS

Network device manufacturers, e.g., producers of switches, routers, firewalls, etc., have abstracted the conceptual requirements TCP/IP or OSI protocols mechanisms using a network plane model. Network protocols are a conceptual set of rules used by network packets; whereas, network planes represent the software "implementation" of these rules. The network plane model categorizes L1, L2, L3, and L4 activities and data stores used to process network packets received and forwarded to network devices. Proprietary network device software is normally stored in firmware and is tightly coupled with the hardware. The following describes the activities, databases, and protocols for the data/forwarding, control, and infrastructure planes. See also Figure 1. (Peplnjak, 2013), (Salisbury, 2012), (Shenker, 2012), (Fratto, 2014).

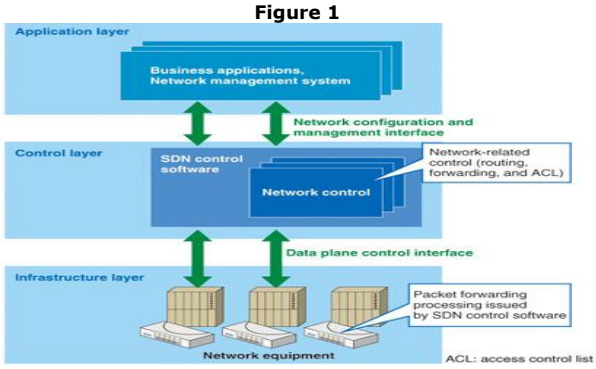
Control plane
The control plane is where forwarding or routing decisions are made. The control plane will use the information collected from the data planes and will interact with its neighbors (adjacencies) to construct shared routing or forwarding information databases. This information is used to enable a network device to interact with other devices and networks not directly attached to the network device. These activities implement procedural logic of routing protocols, e.g., RIP, OSPF, EGRIP, BGP, IEEE 802.1D, etc., the structure of control plan databases/tables, e.g., IP routing table, static routing table, link-state databases, and proprietary algorithm designed to maximize the performance of the device firmware and hardware. Other functions performed by the control plane include processing firewall ACLs, QoS policies, VLAN tags, and VPN identities (Pepinjak, 2013), (Pepinjak, 2014), (Salisbury, 2012), (Shenker, 2012).

It is not the responsibility of the control plane to execute packet and frame forwarding. Rather, it

ISSN: 2167-1435
v31 n3069

is the responsibility of the control plane to maintain a global view or topology of the network and determine the best route for the desired quality of service and security restrictions. The control layer will transmit and update the current and active forwarding plane forwarding tables. Forwarding tables used by the data plane stores data used to forward outbound frames and packets. The data/forwarding plane sends or receives network updates, e.g., route discovery, destination unreachable, etc. to the control plane as the global view of the network. The control plane gathers and processes network intelligence, e.g., hello packets, but the network plane broadcasts and executes the requests for this intelligence. The network intelligence processing will in turn directly affect the forwarding behaviors of the data plane pf the network device (Pepinjak, 2013), (Pepinjak, 2014), (Salisbury, 2012), Shenker, 2012), (Fratto, 2014).

Overview of SDN architecture.
Figure 1



Source: Nakajima, Y. (2014).

The relationship between the control and data planes is similar to a university department chair and a faculty member. Using a variety of data sources a department chair may develop various academic policies and procedures. These policies and guidelines are made available to faculty members to conduct and manage their courses. Each semester begins with a classroom of new students. Based on the department chair's policies, procedures and technical requirements of the course, the faculty member will either forward a student to an outcome of a letter grade or pass/fail. Based on student and course outcome data the faculty member will report to the department chair recommended changes for policies and procedures to guide future actions.

An objective of the department chair is to provide a "global education view" to guide the faculty member from multiple data sources, e.g., university policies, accreditation standards, previous course outcomes assessment. After a global education view has been established, changes and updates may be infrequent. Compared to the department chair the faculty member is the workhorse who actually educates the student, i.e., executes the policy. Many other details may need to be managed by the faculty member in a timely fashion.

Data plane

Similar to the responsibilities of a faculty member, the data/forwarding plane is workhorse of the switch, router or firewall device. While the control plane maintains the forwarding plan, the data plane must execute this forwarding plan. The data plane must perform the functionality of L1 physical connectivity, L2 frame decoding, error detection and correction, store-and forwarding, packet/frame forwarding, VLAN, QoS, and L3 NAT and neighbor management, ARP, and ICMP error signaling. Within a physical network device, the data plane will execute many functions of lower layers of the OSI and TCP/IP models. Speed of execution and performance to forward incoming frames and packets to outgoing ports are very important. This partially explains the historical dependency on vendor-specific firmware and network devices (Pepinjak, 2013), (Pepinjak, 2014), Tanno, 2013a) (Virtual LAN, 2014), (Cisco, 2014).

Management plane

The Management/Policy plane provides interfaces for network administrators to configure, e.g., CL1, GUI, HTTP, SSH, XML, or API, and monitor, e.g., SNMP, a network device. The configuration plane is accessed normally by a vendor-specific client or tool. Through the network device the management plane network administrator may select, configure multiple networking protocols, network services, and review other status information (Pepinjak, 2013), (Pepinjak, 2014).

3. LIMITATIONS OF CURRENT NETWORK TECHNOLOGIES

The following discusses the challenges and limitations of current network technologies to manage large-scale, complex networks and cloud architectures.

Hardware-dependency

To improve data plane performance, legacy network devices are constructed of proprietary network processors and instruction sets.

ISSN: 2167-1435
v31 n3069

Management control and data plane layer application functionalities are stored in the network device firmware and are tightly-coupled with vendor-specific with network device hardware (Open Networking Foundation, 2014a, p6).

Network client configuration tools are often limited to vendor-specific hardware. As a result, organizations establish business policies to limit the providers of network devices to reduce network complexity. As a result, today's networks remain vendor-specific, complex, inflexible, and expensive to maintain. (Shenker, 2012), Tanno (2013b).

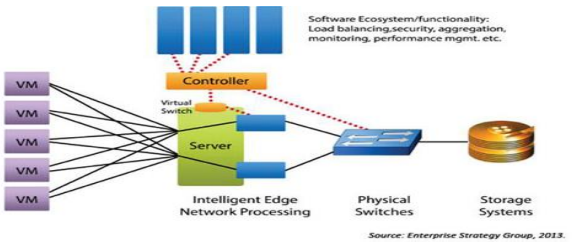
Lack of a program interface and automation
There is very limited ability for users to innovate, customize, or automate network devices through a programmatic interface in current network technologies. Compatibility, interoperability, and cooperation between manufacturers are limited (Open Networking Foundation, 2014a).

Cloud architectures and virtualization
Traditional client-server networks were designed based on a hierarchy of Ethernet switches, often connected to a variety of WAN technologies. Prior to virtualization, applications resided on a single server and were accessed via a static network path. Once configured, the most important network objective was availability of bandwidth to meet the organization's response policies. Storage resources were frequently attached directly to an application server (Shenker, 2012), (Tanno, 2013a), (Tanno, 2013b).

Over 70% of today's servers are virtual machines. VM-VM network traffic is poorly managed, lacks quality of service, and is poorly provisioned for both large-scale networks and between cloud servers (Seetharaman, 2013). The flexibility of virtualization and cloud technologies require dynamic provisioning and migration of network resources, quality of service management, and load balancing.

Enterprise-level hypervisors will support one or more virtual network adaptors ports in a VM guest, IP protocols, internal virtual switches, internal virtual networks and VLANS. Configuration of hypervisor-based virtual network devices, operational and security are similar to any physical network device. Hypervisor virtual switches will normally provide an uplink to connect guests to an external edge, switch or router (Xen Networking, 2014), (VMware, 2014). The management of virtual network devices is often separated from physical networks and WANs and relies on the edge network device to manage this connection. See Figure 2.

Cloud, Virtual, Storage Area Network Architectures
Figure 2



Emulex. (2013).

Cloud Computing architectures and virtualization technologies have created new challenges to traditional network administration and network protocols. Installing or moving a physical server is not an everyday event. Provisioning new VMs or the mobility of VMs may be a simple task for the virtualization infrastructure manager, but will be a complex, labor intensive task, and may take hours or days for a network administrator (Open Networking Foundation, 2014a), (Mahler, 2013), (Tanno, 2013a), (Tanno, 2013b).

Assume a virtual machine is provisioned or migrated to a different network location which necessitates a change in IP address or route. DHCP and DNS services may help reduce some complexity of these changes. In a large-scale network of hundreds or thousands routers, switches, firewalls and configuration sets, it may take hours or days to change the enterprise network using vendor-specific network device management tools. While VM mobility using hypervisor-specific tools may be simple, it does not mean the reconfiguration of the virtual and physical networks will be as simple. Some dynamic routing protocols, e.g., OSPF, may take a significant amount of time for the global view of the network to converge and learn since the control plane's global network view is distributed among network devices (Cisco, 2999), (Cisco, 2005).

Cloud architectures provide agility, flexibility and scalability, e.g., provisioning of additional VMs or SAN devices. Many changes in a cluster pool will

ISSN: 2167-1435
v31 n3069

also require a change in the network configuration. Due to the complexity of current requirements, network administrators often require significant lead time and testing. This may reduce the benefits of agility, flexibility, and scalability (Open Networking Foundation, 2014b).

Multi-tenant applications

Multi-tenant applications require a network infrastructure that supports diverse groups of users and organizations with different application, network performance, and security requirements. Multi-tenant application management is limited by current network protocols, which rely on VLAN's and/or subnets. L2 network devices do not scale well due to the need to track large number of MAC addresses, and L3 does not permit sharing of IP addresses between tenants (Seetharaman, S., 2013). Virtualization hypervisors often employ VLAN (IEEE 802.1Q) tags to simply switched Ethernet network administration and to provide identification of packets traveling through trunk lines. VLAN tagging supplements switch forwarding rules to specific VMs or a specific individual multi-tenant application. The maximum number of VLAN tags available is 4096, which limits the number of VMs and multi-tenant applications connected a virtual or physical switch (Mahler, 2013), (Tanno, 2013a), (Tanno, 2013b), (Virtual Lan, 2014), (Cisco, 2014), (ONF Overview, (2014).

Mixed network traffic loads

Historically, networks were designed to support data traffic and file transfer requirements. Data traffic bottlenecks were easily resolved by allocating more bandwidth - the more bandwidth the better. However, VoIP and videoconferencing requires the balancing of bandwidth, latency, and quality of service. Balancing different traffic classes to prevent bottlenecks may be improved by allocating extra bandwidth or prioritizing packet delivery. However, traditional network devices are not dynamic or automated. Therefore, simply increasing reserved bandwidth may waste bandwidth during off-peak times. (Open Networking Foundation, 2014a), O3bnetworks, 2014).

Quality of service, granularity and manual administration

Many network devices provide quality of service settings which enables the network administrators to customize and prioritize the transmission of packets. For example, existing protocols permit a network administrator to configure an application with a higher priority using the System Initialization Protocol (SIP). For example, SIP may be used to control multimedia sessions or assign a lower priority to FTP. Indirectly, it is also possible to configure subnets and VLANs to support quality of service policies. However, subnets and VLAN are complex to administer, require manual configuration, provides limited automation and are static. Due to the lack of program control, the network administrator is unable to dynamically modify priorities by application, user, device, network, or time of day (Eli, 2013).

With SDN, quality of service management may be customized by other factors such as type-of-device, e.g., customer mobile versus BYOD, user, or application program. Dynamic allocation of bandwidth and quality of service are also important to manage large data set transfers commonly encountered in big data, data analytics applications, and other massively parallel applications (O3bnetworks, 2014, p6).

Mobile devices, BYOD, and IoT

Today's complex networks need to support mobile customer apps, Bring your Own Device (BYOD) policies, and the emerging Internet of Things (IoT). To accommodate these new challenges, there is a need to provide dynamic bandwidth, quality of service, and security to protect corporate data. While network administrators may use existing protocols, planning and managing these new technologies are more difficult and uncertain. (Open Networking Foundation, 2014a, 2014d).

Lack of centralized network policies

Current network administration lacks a centralized policy and administration system that is not vendor or device specific. To implement a network-wide policy, thousands of network devices and protocols must be configured using a variety of manual, vendor-specific management tools. Network policies applied to all vendor's hardware devices, virtualized servers, and WAN connections would provide for consistency, reliability and reusability. As the complexity of network administration increases, the probability of security breaches, network failures, resistance to change and innovation may increase. (Open Networking Foundation, 2014a).

ISSN: 2167-1435
v31 n3069

4. COPING WITH ENTERPRISE NETWORK COMPLEXITY

The TCP/IP and OSI conceptual models have provided an abstract framework to facilitate the implementation of traditional networks. However, the current status of network administration remains complex and dependent on vendor hardware, software, and administration tools. On the other hand, application programming, interoperability, and virtualization are also complex systems. Unlike network protocols, high-level and object-program languages, virtualization, and web services have benefited modular design, abstraction, open layered-models, and interfaces. (Schünemann, 20014), (Shenker, 2012), (Tanno, 2013a, 2013b).

The evolution of programming languages provides an excellent example of the benefits of abstraction. Early machine languages were very complex and used no abstractions. Early machine language programmers had to be aware of the details of a hardware platform and specify where each bit was stored. Higher-level programming languages used compilers and the operating system layers as a form of abstraction to create machine-independent interfaces to hide the complexity of the hardware. Later, object-oriented programming languages further hide the complexity of programming by coding reusable classes and interfaces (Eli, 2013).

Most programming language families are based on common syntax and semantic rules, modularity and abstraction. Object-oriented programming languages, e.g., Java, Python, Ruby on Rails, etc., are platform-independent and open-source. On the other hand, network protocols are managed by vendor-specific software that is executed on a vendor-specific network device operating system and hardware.

Virtualization permits multiple virtual hosts to share a common physical processing and storage pool of resources. The virtualization layer, or hypervisor, is an abstraction layer that simplifies the provision, administration, and scaling of virtual hosts (VMs), applications, and storage (SANs).

Modularity and abstraction are strategies designed to decrease complexity. Modularity also promotes reusability of components. Application programmers often applied modularity by decomposing requirements into separate executable program modules and provided an interface using program calls and APIs.

During the 80s and 90s, the lack of interoperability standards limited the use program calls and APIs. For example, the ability to re-use executable program modules was limited by differences in hardware, program language, and operating systems. The acceptance of open web service protocols enabled application programmers to reduce application complexity and promoted platform interoperability.

5. INTRODUCTION TO OPENSTACK, SDN AND OPENFLOW

Computing and storage resources have been abstracted and virtualized. The abstraction and the virtualization of current network technologies may also reduce network complexity, administrations and other limitations. Since 2011, the Open Network Foundation (ONF) (ONF Overview, 2014), a user-driven organization, has been dedicated to the promotion and adoption of Software-Defined Networking (SDN) through open enterprise network standards (Open Networking Foundation, 2014a). SDN decouples the control plane functions from the data forwarding plane functions and provides a programmable interface. A Software-Defined Network model defines functionality by software concepts, rather than hardware (Shenker, 2012).

Understanding the relationship between SDN and OpenStack technologies is important. OpenStack is an open-source, cloud management system that provides a level of abstraction to build and manage private and public cloud computing platforms (OpenSource.com, 2014a). Cloud Computing architectures manage three large pools resources 1) computing (virtual machines), 2) networking, and 3) storage (SANs). Virtualized storage and computing resources have emerged into mature technologies. SDN and OpenFlow now provide a network architecture for dynamic and programmable network administration (OpenSource.com, 2014b). SDN and OpenFlow are current, accepted solutions to overcome network limitations facing modern enterprise cloud architectures by network vendor and the open source community. OpenStack permits business and middleware applications to communicate and manage cloud compute,

storage and network resources through an API interface. This following discussion focuses on relationship between OpenStack and SDN architectures.

SDN separates and abstracts the management, control and data/forwarding planes into a layered architecture to support large-scale enterprise networks and virtual resources. The Software-Defined Networking Architecture has three layers: a) application layer, b) control layer, and c) infrastructure layer

SDN Application Layer

The application layer will execute layer 4-7 network applications or business services either as a dedicated host, VM, or SDN software controller. VMWare VSX or a Citrix's XEN hypervisor or SDN applications may use API interfaces to communicate with the Control Layer requesting that a new VM be automatically configure the enterprise network. The centralized control layer would then use OpenFlow to communicate configuration requests to the SDN infrastructure layer of various physical network devices (Open Networking Foundation, 2014), (Mahler, D., 2013), (Seetharaman, S., 2013).

To avoid confusion, it is important to understand that the SDN Application Layer is not a hypervisor or a virtual machine management system, e.g., VMWare VSphere. It is not a storage area network management system, e.g., NetApp, EMC, or UniSphere. Rather, the SDN open-source application layer directly interfaces network resources with these proprietary applications or an OpenStack cloud management system.

The SDN Virtualization Abstraction Layer specifies a standard for interfaces and connections between the SDN Application Layer and the SDN Control Layer using APIs. The Linux Foundation OpenDayLight Project , sponsored by IBM, Cisco, Jupiter, etc., provides software and the OpenDayLight APIs to enable virtual applications and the OpenStack cloud management system to communicate between theses layers (OpenDayLight, 2014), (Seetharaman, 2013), (Linux Foundation, 2014).

SDN Control Layer

The SDN Control Layer provides the functionality of the network control plane. The separation (decoupling) and centralization of the control plane logic in SDN is a major change that simplifies dynamic management of a large-scale enterprise network. See Table 1.

Table 1 Differences between the SDN Control Layer and Traditional Control Planes

The SDN Control Layer based on software rather than hardware.
The SDN Control Layer functionality and data is centralized and separated from each physical network device.
The SDN Control Layer is programmable by the user.
The granularity of network administration may be defined by the programmer.
The SDN Control Layer is open-source, and not vendor or hardware dependent.

Based on L2 and L3 protocols and tables, especially the IP routing table, static routing table, and link state databases, the SDN Control Layer will process network data to make decisions based on the global view or topology of the network as well as processing and communicating quality-of-service, firewall rules and other network policies.

The OpenDayLight Project also introduced Network Function Virtualization (NFV), which enabled the agile placement of L4-L7 network services in either the SDN Control Layer or Application Layer. Initial SDN standards executed L4-L7 services in the control layer. NFV enabled L4-L7 services to be provided by a VM and treats Network Functions Virtualization as a Service (NFVaaS). Examples of L4-L7 services include load balancers, firewalls, intrusion detection systems and WAN accelerators (Open Networking Foundation, 2014), (OpenDayLight, 2013), (Seetharaman, S., 2013). The potential to separate network administration responsibilities between the Control Layer and the Application Layer was important to manage processing overhead caused by the increased number of endpoints caused by rapid growth of mobile device connections, BYOD, and multi-tenant applications (Casado, Koponen, Shenker, & Tootoonchian, 2012). However, the infrastructure layer or data plane for a specific device remains responsible to execute and implement the policy.

The SDN Service Abstraction Layer (SAL) specifies the network communication standards for network protocols and APIs between the SDN Controller Layer and the Infrastructure Layer. To enable this interface, ONF has proposed the OpenFlow protocol.

OpenFlow is an open source network protocol. OpenFlow is analogous to SOAP. OpenFlow has been designed to provide communication and to

facilitate interoperability of applications. OpenFlow enables SDN controllers to communicate configuration and forwarding information to management plane of network devices. Any network device can share information using OpenFlow. SDN applications often store OpenFlow packets in data structure called an Open Table, or other customized data structure. SDN also supports other network management protocols and APIs. SDN, supports NETCONF/Yang, Cisco's NetFlow protocol, SNMP, and other vendor-based XML APIs(Open Networking Foundation, 2014d), MacVittie, 2013), (OpenSource.com, 2014b), (Seetharaman, 2013), (Linux Foundation, 2014).

SDN Infrastructure Layer

The SDN Infrastructure Layer provides for network data/forwarding plane functions. While the infrastructure layer may still be vendor-dependent and tightly-coupled, an OpenFlow-enabled network device can communicate with a centralized, software-based controller to manage vendor network hardware. The infrastructure layer or data plane is still responsible for maintaining the device's forwarding table and executing the forwarding plan.

The infrastructure layer must also perform the functionality of L1 physical connectivity, L2 frame decoding, error detection and correction, store-and forwarding, packet/frame forwarding, VLAN , QoS, and L3 NAT management, ARP, neighbor management, and ICMP error signaling. The data plane is tightly-coupled with the device hardware and these functionalities are directly executed by individual network devices based on the information provided the Control Layer. However, SDN enables a physical network device may be replaced by software-based, OpenFlow-enabled network device, e.g., Open vSwitches (Linux Foundation, 2014). Future network devices may be delivered as software and executed on a virtual machine, rather than a physical device.

OpenFlow is key-enabler of SDN, but it has been limited to Ethernet and optical L2 technologies. Future implementations of OpenFlow will need to extend the SDN model to other L2 and WAN technologies.

6. IMPLICATIONS FOR THE IS CURRICULUM

The SDN architecture is in its infancy and is continuously evolving. Current implementations of SDN application and control layer software rarely fully implements the existing SDN standards. The overhead of communicating between a SDN centralized controller and OpenFlow switches and routers has slightly decreased overall network performance. Recommendations to improve future SDN implementations are to improve standards for application layer functionality, such as, granularity of service, dynamic customization, security and better support for mobile device and BOYD end points (Shenker, 2012), (Seetharaman, 2013), (Porras, 2013), (Casado et al, 2012), (Linux Foundation, 2014). APIs and OpenFlow are supported by most program languages. There are no limitations to the innovation of SDN controllers.

While the current IS curriculum includes coverage the TCP/IP and OSI conceptual network models, these traditional models are not related to vendor-specific network devices within the context of complex, large-scale networks. It is understandable that IS educators avoid applying network models to vendor-specific devices or configuration procedures, e.g., Cisco-like CCNA education. One can appreciate the resistance of network device manufacturers to adopt a new enterprise network model and protocols, which will render their network devices to commodity status and challenge their dominance of the market place.

Industry and IS education has relied on a collection of isolated protocols, each solving a specific network problem. Until recently, industry has failed to introduce or accept new conceptual models and network protocols to meet the challenges of cloud computing, multi-tenant applications, distributed data centers, mobile apps, and BYOD. To better understand the limitations of current network practices and emerging open source network standards, it is recommended that the IS network curriculum include coverage of the traditional network plane model and the Software-Defined Networking Architecture.

The IS Model Curriculum has encouraged the inclusion of enterprise infrastructure, cloud computing and virtualization technologies. Mobile application development courses are increasing in popularity. These technologies require a dynamic, automated, programmable enterprise network. If 70% of todays' servers are virtual machines, the IS curriculum must consider academic content to support the third-leg of a cloud architecture, e.g., enterprise and

ISSN: 2167-1435
v31 n3069

data center networks. Cloud computing is the ultimate "system" to teach in an information system curriculum. If some network device manufacturers have resisted an open source solution to handle the emerging network administration challenges, the IS curriculum should not.

The introduction of OpenStack, Software-Defined Networking, OpenFlow, virtual network devices (Open vSwitch) within the IS curriculum provides a way to introduce large-scale, complex networks, reinforce cloud computing and virtualization, and the challenges of enterprise infrastructure systems. More exciting are the academic opportunities to present these concepts with any microcomputer operating system and virtualization manager. Physical devices, physical cabling, and department budgets are not needed. All IS network students would need is a laptop. For example, a free SUSE OpenStack download is available to provide a functional, cloud-based architecture to be used in an educational context (OpenStack Portal, 2014).

Software-defined networking is not just another protocol. It provides an open interface that users and non-vendors can customize, manage, and innovate physical and virtual network management through an open, programmable interface and protocol. As SQL provided an open data retrieval language to standardize database interfaces, it is expected that Software-defined network protocols will standardize and change the way organizations will managed complex, large-scale networks, data centers and cloud-based systems for the future (Guendert, 2014).

7.REFERENCES

ABET CAC. (2014). Criteria for Accrediting Computing Programs, 2014-2015. ABET Retrieved from http://www.abet.org/cac-criteria-2014-2015/

Cannistra, R. & DeCusatis, C. (2013). Software Defined Networking Lab Initiatives Proceedings Enterprise 2013 Computing Community Conference

Casado, M., Koponen, T. Shenker, Scott & Tootoonchian, A. (2012). Fabric: A Retrospective on Evolving SDN. ACM Proceedings for Hot topics in software defined networks pages, pages 85-90

Cisco. (1999). Introduction to ISPF. Retrieved on 7/3/2014 from http://ws.edu.isoc.org/workshops/2004/CEDIA2/material/b2-1up.pdf.

Cisco. (2005). OSPF Design Guide. Retrieved on 6/12/2014 from http://www.cisco.com/c/en/us/support/docs/ip/open-shortest-path-first-ospf/7039-1.html

Cisco. (2012). Fundamentals of VXLAN. Retrieved on 6/15/2014 from https://www.youtube.com/watch?v=j7on2iLk5ls

Cisco. (2014). VLAN Tagging - Understanding VLANs Ethernet Frames. Retrieved on 7/10/2014 from http://www.firewall.cx/networking-topics/vlan-networks/219-vlan-tagging.html

Drutskoy, D., Keller, E. & Rexford, J. (2013). Scalable Network Virtualization in Software-Defined Networks. *IEEE Internet Computing.* Retrieved on 7/15/2014 from https://ngn.cs.colorado.edu/docs/drutsky_ieee_ic_flown_2013.pdf

Eli. (2013). Software Defined Networking (SDN) Introduction. Retrieved on 6/15/2014 from https://www.youtube.com/watch?v=2BJyIIIYU8E

Emulex. (2013). Software-defined Networking. Retrieved on 6/15/2014 from http://www.emulex.com/solutions/network-connectivity-solutions/software-defined-networking/

Ferro, G. (2014). ONF Extends their SDN to Embrace NETCONF and YANG. Retrieved on 6/20/2014 http://etherealmind.com/onf-extends-their-sdn-to-embrace-netconf-and-yang/

Fratto, M. (2014). What is the difference between Control plane and Data plane in the context of networking equipment like routers /switches? Retrieved on 6/15/2014 http://www.quora.com/What-is-the-difference-between-Control-plane-and-Data-plane-in-the-context-of-networking-equipment-like-routers-switches

Guendert, S. (2014). Software Defined Networking, Payments Systems, and

ISSN: 2167-1435
v31 n3069

zEnterprise. Proceedings Enterprise 2014 Computing Community Conference

Linux Foundation. (2014). About SDN and NFV. Retrieved on 7/160/2014 from http://www.opendaylight.org/project/technical-overview

Linux Foundation. (2014). About SDN and NFV. Retrieved on 7/160/2014 from http://www.opendaylight.org/project/technical-overview

MacVittie, L. (2013). F5 Application Layer SDN: Now with Extreme Programmability. Retrieved on 7/2/2014 from https://devcentral.f5.com/articles/f5-application-layer-sdn-now-with-extreme-programmability

Mahler, D. (2013). Introduction to Open vSwitch (OVS). Retrieved on 6/15/2014 from https://www.youtube.com/watch?v=rYW7kQRyUvA

NETCONF. (2014). Retrieved on 6/20/2014 from http://en.wikipedia.org/wiki/NETCONF

Nakajima, Y. (2014). Standardization Progress in Software Defined Networking/OpenFlow. Retrieved on 6/15/2014 from https://www.ntt-review.jp/archive/ntttechnical.php?contents=ntr201302gls.html

O3bnetworks (2014). What is Network Latency and Why Does It Matter? Retrieved on 6/15/2014 from http://www.o3bnetworks.com/media/40980/white%20paper_latency%20matters.pdf

ONF Overview. (2014). Retrieved on 6/20/2014 from https://www.opennetworking.org/about/onf-overview

Open Networking Foundation. (2014a). Software-Defined Networking: The New Norm for Network. Retrieve on 6/15/2014 from https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

Open Networking Foundation. (2014b). Software-Defined Networking (SDN) Definition. Retrieve on 6/15/2014 from https://www.opennetworking.org/sdn-resources/sdn-definition

Open Networking Foundation. (2014c). OpenFlow-enabled SDN and Network Functions Virtualization. Retrieved on 6/15/2014 from https://www.opennetworking.org/images/stories/downloads/sdn-resources/solution-briefs/sb-sdn-nvf-solution.pdf

Open Networking Foundation. (2014d). OpenFlow-enabled Transport SDN. Retrieved on 6/15/2014 from https://www.opennetworking.org/images/stories/downloads/sdn-resources/solution-briefs/sb-of-enabled-transport-sdn.pdf

OpenDayLight. (2014). Retrieved on 7/16/2014 from http://www.opendaylight.org/

OpenFlow (2014). Retrieved on 7/3/2014 - http://en.wikipedia.org/wiki/OpenFlow

OpenSource.com. (2014a). What is OpenStack? Retrieved on 7/8/2014 from http://opensource.com/resources/what-is-openstack

OpenSource.com. (2014b). OpenStack: The Open Source Cloud Operating System. Retrieved on 7/8/2014 from http://www.openstack.org/software/

OpenStack Portal. (2014). SUSE OpenStack Download Site. Retrieved on 7/8/2014 from http://en.opensuse.org/Portal:OpenStack

Pate, P. (2013). NFV and SDN: What's the Difference? Retrieved on 6/15/2014 from http://www.sdncentral.com/technology/nfv-and-sdn-whats-the-difference/2013/03/

Peplnjak, I. (2013). Management, Control and Data Planes in Network Devices and Systems. Retrieved on 6/15/2014 from http://blog.ipspace.net/2013/08/management-control-and-data-planes-in.html

Peplnjak, I. (2014). Software Defined Networking and Security. Retrieved on 6/15/2014 from https://www.troopers.de/wp-content/uploads/2014/03/TROOPERS14-Security_and_SDN-A_perfect_fit_or_oil-and-water-Ivan_Pepelnjak.pdf

ISSN: 2167-1435
v31 n3069

Porras, P. (2013). Toward a More Secure SDN Control Layer — SRI International's View. Retrieved on 7/14/2014 from http://www.sdncentral.com/author/porrascsl-sri-com/

Salisbury, B. (2012). The Control Plane, Data Plane and Forwarding Plane in Networks. Retrieved on 7/2/2014 from http://networkstatic.net/the-control-plane-data-plane-and-forwarding-plane-in-networks/

Schünemann, U. (2004). Model <-> Abstraction. Retrieve on 6/15/2014 from http://web.cs.mun.ca/~ulf/mod/rel.html

SDNCentral LLC. (2014a). What's NFV – Network Functions Virtualization? Retrieved on 6/15/2014 from http://www.sdncentral.com/whats-network-functions-virtualization-nfv/

SDNCentral LLC. (2014b). Which is Better – SDN or NFV? Retrieved on 6/15/2014 from http://www.sdncentral.com/which-is-better-sdn-or-nfv/

SDNCentral LLC. (2014c). What is OpenFlow? Retrieved on 6/20/2014 http://www.sdncentral.com/what-is-openflow/

Seetharaman, S. (2013). Understanding and Deploying Virtual Networks. SDN Hub. Retrieved on 7/14/2014 from http://www.slideshare.net/sdnhub/understanding-and-deploying-network-virtualization

Sequeira, A. (2013). Introducing VMware NSX – The Platform for Network Virtualization. Retrieved on 6/28/2014 from http://cto.vmware.com/introducing-vmware-nsx-the-platform-for-network-virtualization/

Shenker, S. (2012). Gentle Introduction to Software-Defined Networking. Retrieved on 6/15/2014 from https://www.youtube.com/watch?feature=player_detailpage&v=eXsCQdshMr4&t=168

Tanno, D. (2013a). Introduction to Software Defined Networks - Part 1, Virtualization Basics. Retrieved on 7/15/2014 from https://www.youtube.com/watch?v=p5WJBnKX44Y

Tanno, D. (2013b). Introduction to Software Defined Networks - Part 2, The Future of Networking. Retrieved on 7/15/2014 from https://www.youtube.com/watch?v=pXJyUjRK3AI

Topi, H., Valacich, J., Wright, R., Kaiser, K., Nunamaker, Jr., J.F., Sipior, J. & deVreede, G. (2010). *IS 2010 Curriculum Guidelines for Undergraduate Degree Programs in Information Systems.* ACM.

Virtual LAN (2014). Retrieved on 7/10/2014 from http://en.wikipedia.org/wiki/Virtual_LAN

Vmware (2014). VMware Virtual Networking Concepts. Retrieved on 7/2/2014 from http://www.vmware.com/files/pdf/virtual_networking_concepts.pdf

Xen Networking. (2014). Retrieved on 6/15/2014 - http://wiki.xen.org/wiki/Xen_Networking#Virtual_Network_Interfaces