ORNL/TM-2015/211

Multi-Tenant Isolation via Reconfigurable Networks

Ferrol Aderholdt
Blake Caldwell
Susan Hicks
Scott Koch
Thomas Naughton
Daniel Pelfrey
James Pogge
Stephen L. Scott
Galen Shipman
Lawrence Sorrillo

December 2014

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE US DEPARTMENT OF ENERGY

ORNL/TM-2015/211

Computing & Computational Sciences Directorate

DoD-HPC Program

Multi-Tenant Isolation via Reconfigurable Networks

Ferrol Aderholdt[2], Blake Caldwell[1], Susan Hicks[1], Scott Koch[1],
Thomas Naughton[1], Daniel Pelfrey[1], James Pogge[2],
Stephen L. Scott[1,2], Galen Shipman[2] and Lawrence Sorrillo[1]

[1] Oak Ridge National Laboratory
Oak Ridge, TN 37831

[2] Tennessee Technological University
Cookeville, TN, 38501

Date Published: December 2014

Prepared by
OAK RIDGE NATIONAL LABORATORY
P.O. Box 2008
Oak Ridge, Tennessee 37831-6285
managed by
UT-Battelle, LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

ACL	Access Control List
API	Application Programming Interface
ASIC	Application Specific Integrated Circuit
BGP	Border Gateway Protocol
CLI	Command Line Interface
CPU	Central Processing Unit
DNAT	Dynamic Network Address Translation
GRE	Generic Routing Encapsulation
HPC	high-performance computing
LACP	Link Aggregation Control Protocol
LXC	Linux Containers
MD5	Message Digest Algorithm V5
MLAG	Multichassis Link Aggregation
NAT	Network Address Translation
NFV	Network Function Virtualization
NIDS	Network Intrusion Detection System
OS	operating system
OSPF	Open Shortest Path First
OVS	Open Virtual Switch
QOS	Quality of Service
SDN	Software Defined Networking
SDM	Security Device Manager
SNAT	Source Network Address Translation
SNMP	Simple Network Management Protocol
VLAN	Virtual Local Area Network
VMS	Virtual Modular Switch
VNIC	Virtualized Network Interface Control
VPC	Virtual Port Channel
VRF	Virtual Routing and Forwarding
XMPP	Extensible Messaging and Presence Protocol
VXLAN	Virtual eXtensible Local Area Network

Executive Summary

Multi-Tenant Isolation via Reconfigurable Networks

High performance computing environments are often used for a wide variety of workloads ranging from simulation, data transformation and analysis, and complex workflows to name just a few. These systems may process data at various security levels but in so doing are often enclaved at the highest security posture. This approach places significant restrictions on the users of the system even when processing data at a lower security level and exposes data at higher levels of confidentiality to a much broader population than otherwise necessary. The traditional approach of isolation, while effective in establishing security enclaves poses significant challenges for the use of shared infrastructure in HPC environments. This report details current state-of-the-art in reconfigurable network enclaving through Software Defined Networking (SDN) and Network Function Virtualization (NFV) and their applicability to secure enclaves in HPC environments.

SDN and NFV methods are based on a solid foundation of system wide virtualization. The purpose of which is very straight forward, the system administrator can deploy networks that are more amenable to customer needs, and at the same time achieve increased scalability making it easier to increase overall capacity as needed without negatively affecting functionality. The network administration of both the server system and the virtual sub-systems is simplified allowing control of the infrastructure through well-defined APIs (Application Programming Interface). While SDN and NFV technologies offer significant promise in meeting these goals, they also provide the ability to address a significant component of the multi-tenant challenge in HPC environments, namely resource isolation. Traditional HPC systems are built upon scalable high-performance networking technologies designed to meet specific application requirements. Dynamic isolation of resources within these environments has remained difficult to achieve. SDN and NFV methodology provide us with relevant concepts and available open standards based APIs that isolate compute and storage resources within an otherwise common networking infrastructure. Additionally, the integration of the networking APIs within larger system frameworks such as OpenStack provide the tools necessary to establish isolated enclaves dynamically allowing the benefits of HPC while providing a controlled security structure surrounding these systems.

Key Points SDN and NFV provides the functionality necessary to configure distributed networking components on-demand, while at the same time providing desired performance, security, and reliability goals. The requirements of these open standards are largely driven by the cloud computing community. Adapting these standards to HPC systems can provide an increased level of flexibility with significantly higher performance than that of a typical cloud computing infrastructure. Reconfigurable networks are a key component of this flexibility providing a unique opportunity to achieve the performance and application scalability of leading edge HPC platforms while providing the ability to isolate applications within a shared infrastructure.

Recommendations Additional research into the application of SDN and NFV technologies within an HPC context is required. Leveraging large-scale orchestration frameworks such as OpenStack to manage HPC system components will broaden the applicability and improve the security of HPC systems. While our initial work focuses on leveraging SDN and NFV capabilities of Ethernet based networks for secure enclaves, the proposed techniques are readily adaptable to high-performance networking technologies utilized within HPC. Adopting SDN, NFV and broader orchestration technologies such as OpenStack for

on-demand network reconfiguration will require further development including scalable low-overhead tools that provide monitoring and auditing of networking components (including endpoints). All this development should be within the scope of compliance with applicable and necessary security policies.

The remainder of this report is structured as follows:

- Section 1 introduces software defined networking and network function virtualization and their role in addressing remote resource isolation in multi-tenant HPC systems.
- Section 2 provides background on the resource management and orchestration capabilities available through SDN and NFV. Relevant terminology in SDN and NFV is also detailed in this section.
- Section 3 details alternative architectures and available methods for implementing dynamically reconfiguring networks.
- Section 4 provides an overview of a number of SDN and NFV vendor technologies and the capabilities provided by them.
- Section 5 provides an overview of our secure enclave testbed and planned activities to assess SDN and NFV suitability and gaps for secure enclave resource isolation.
- Section 6 concludes the report and highlights key observations in the use of SDN and NFV to support resource management and orchestration of secure enclaves. This includes a summary of identified limitations in the current state-of-the-practice in SDN and NFV.

Chapter 1

Introduction

Server virtualization introduces immediate benefits in the improved sharing of centralized resources. This offers several benefits, which include the efficient utilization of hardware, saving power, cooling and cabinet space by only using components necessary for the application. Virtualization also aids in dynamic deployment of new services, which makes capacity planning and growth more manageable. This ultimately leads to improved flexibility for data managing the available resources. Additionally, reliability can be improved by leveraging virtual machine migration and other resilience capabilities, which can be used to move critical services away from failing hardware. Virtualization allows for new hardware to be integrated without requiring a redesign of the infrastructure, which avoids service interruptions (downtimes) when adding the new hardware.

As the use of virtualization becomes more ubiquitous, additional hardware support is emerging to assist with the multiplexing of the physical resources. Many hardware specific services such as data storage and networking were not initially easily realizable with available virtual machine technologies. However, newer hardware functionality is helping to improve performance when virtualizing these critical I/O services. In the specific case of HPC workloads, latency and bandwidth requirements place a higher performance demand on these virtualized services and the hardware used to realize them. The adapting of virtualization methods within the HPC community requires a more narrowly focused approach to virtualization. Streamlined techniques such as the use of Linux containers provides a virtualized environment rather than a complete virtual machine, enabling the flexibility desired within the HPC community without sacrificing performance system performance.

Historically, networking has followed a tiered service level design based on connectivity, throughput and Quality of Service (QoS) requirements. The use of routers, load balancers, switches, and firewalls are dominated by the type, number and level of service each connection is expected to provide. This model severely limits the flexibility required in multi-tenant systems, which seek to take advantage of the fast deployment capabilities made possible by virtualized systems. Additionally, the use of proprietary operating system (OS) and switch fabric hardware make the use of network appliances from multiple vendors complicated and time consuming. This lack of commonality in the networking layer causes burdens for maintenance and adds to operational costs. Even when considering the use of automated tools, careful planning is required to ensure minimal system disruption as these necessary configuration changes occur.

The growth of server virtualization is spurring increased interest in technologies that can be leveraged to aid with network virtualization. A key element of modern networking with virtualized resources is the combination of Software Defined Networking (SDN) and Network Function Virtualization (NFV). As industry standard APIs are developed using a common open source standard, network appliance operation moves seamlessly within the compute infrastructure.

The goal of this report is to present results from our investigation into mechanisms that can be used to implement reconfigurable networks. The intent is to leverage these networking technologies to facilitate isolation in multi-tenant environments. The report focuses on SDN and NFV to gain insights into their use in a high-performance computing (HPC) context. This includes a review of methods and technologies for implementing reconfigurable networks and a snapshot of key vendors that are providing products that support SDN and NFV. The report includes a presentation of design concepts and solutions that allow flexible implementation of reconfigurable networks within the topology outlined by the virtual containers and compute nodes specific to a HPC implementation of virtualized environments. Particular emphasis is placed on the near term challenges, such as designing, implementing, securing, and maintaining a dynamic reconfigurable network that meets performance, security, and operational requirements.

1.1 Report Outline

The remainder of the report is structured as follows, Chapter 2 provides background on the resource management and orchestration capabilities available through SDN and NFV. Relevant terminology in SDN and NFV is also detailed in this section. In Chapter 3 details are given about alternative architectures and available methods for implementing dynamically reconfiguring networks. Chapter 4 provides an overview of a number of SDN and NFV vendor technologies and their capabilities. An overview of our secure enclave testbed and planned activities to assess SDN and NFV suitability and for secure enclave resource isolation is presented in Chapter 5. Lastly, Chapter 6 concludes the report and highlights key observations in the use of SDN and NFV to support resource management and orchestration of secure enclaves. This includes a summary of identified limitations in the current state-of-the-practice in SDN and NFV.

Chapter 2

Background

2.1 SDN and Network Function Virtualization

Prior work in *programmable networks* laid the foundation for the current efforts into Software Defined Networking (SDN) [33]. Fundamentally, the SDN architectural model is based on the notion of decoupling the control and data channels. This separation enables the control portion to be managed in a more flexible manner without binding it to the actual data forwarding layer [30, 33], i.e., the control and data may be managed (even implemented) separately.

This separation can be leveraged by virtualized environments to allow more dynamic configuration of the network to meet the needs of applications. Allowing tenants (customers) to provision and configure dynamic networks can be beneficial for testing applications, or scaling an existing production environment or specific application. Virtualization saves time for the tenants since they don't have to wait for network administrators to provision and configure additional network resources. This saves time for both the network engineers and systems engineers. The system engineers can focus on adding to resource capacity, leaving the virtualization controller to handle the tenant flexibility needs. SDN works by separating the control plane and data plane in the network environment. The control plane handles the configuration and use management of available network resources including routing and monitoring functions. The control plane is responsible for QoS and security policy enforcement on the network connections. The data plane handles the actual flow of data between applications with connections and port sharing under direct management of the control plane between tenant compute nodes and any external network connections.

In a traditional network the routers, switches, firewalls, and load balancers are dedicated to a physical configuration. Often these pieces of hardware are from different vendors. Organizational network topologies are typically centered on these functions. Virtualized networking is focused on commodity servers that can perform all of these functions to various extents. The standardization of software based services as opposed to application specific physical appliances provides on-demand flexibility in provisioning the layout of the newly defined system. Routers, firewalls and load balancers can be rapidly deployed as needed in a virtualized environment. As network commodity servers improve in performance and lowered cost, additional network function virtualization can be realized. Advances in the switch fabric ASIC and corresponding controllers will allow MAC (media access control) functions such as layer 2 and layer 3 level control functions to be transferred to the virtualized network control resulting in a reduction in system cost coupled with increased deployment flexibility.

The large scale adaptation of SDN facilitates the dynamic reconfiguration of networks to meet the needs of both specific user requirements and applications. The incorporation of Network Function

Virtualization (NFV) on the other hand is changing how networks are scaled, enabling dynamically configured functions such as firewalls, and load balancers to optimize deployment time. Systems can be deployed and realized based on available CPU, network, and memory capacity in the virtual server farms. This deployment model saves the network engineers from having to focus on rack space, cooling, and cabling requirements associated with specific system expansion needs. NFV deployment reduces custom hardware support costs, however server support costs will increase.

System performance requirements are more easily realized using SDN and NFV. The use of SDN allows the user to quickly spin up network functions such firewalls and load balancers based on specific needs. Additionally SDN has the potential to assist in optimizing traffic flows within the network to reduce latency and network hot spots. For example, in a SDN environment, software could detect that tenant traffic is spread out and is pushing heavy traffic among nodes causing potential hot spots and requiring QoS to be enabled. Rearranging the host servers and changing the network to meet that service level agreement is possible with reconfigurable networks. The software provisions the network in an underutilized area with respect to virtual server resources and available network resources providing the hypervisors with the necessary resources to move tenant traffic over to the newly created service. Likewise the software could detect that a virtualized load balancer, or firewall, is nearing capacity and can spin up and configure replacement virtual services with additional capability. The combination of SDN and NFV allows higher functionality, while being able to monitor the network, and modify configurations as needed.

2.2 Terminology

This section reviews relevant terminology and background concepts. Standardizing of terminology is still being worked out in the network virtualization community and inconsistencies exist in the literature. For example, the available vendor documentation and associated research on both SDN and OpenFlow discuss basic capabilities and functions in application specific terminology, often focusing on specific use cases, rather than a generic capability.

SDN In the SDN architecture, the control and data planes are decoupled, network intelligence and state are logically centralized, and the underlying network infrastructure is abstracted from the applications. As a result, enterprises and carriers gain unprecedented programmability, automation, and network control, enabling them to build highly scalable, flexible networks that readily adapt to changing business needs [17].

OpenFlow OpenFlow is an open standard that enables researchers to run experimental protocols in the campus networks we use every day. OpenFlow is added as a feature to commercial Ethernet switches, routers and wireless access points and provides a standardized hook to allow researchers to run experiments, without requiring vendors to expose the internal workings of their network devices. OpenFlow is currently being implemented by major vendors, with OpenFlow-enabled switches now commercially available [27].

OpenDaylight OpenDaylight is a collaborative, open source project to advance Software-Defined Networking (SDN). OpenDaylight is a community-led, open, industry-supported framework, consisting of code and blueprints, for accelerating adoption, fostering new innovation, reducing risk and creating a more transparent approach to Software-Defined Networking [34].

Control plane and Data Plane In traditional networking, the control plane and data plane traffic shares the same path. In SDN, the control and data are separated to facilitate an abstract network design. Control plane traffic consists of L2, and L3 protocols, management traffic such as Simple Network Management Protocol (SNMP) and Secure Shell (SSH). The data plane is the traffic containing the data exchanged between applications, i.e., application data.

Abstraction The concept of network abstraction is primarily focused on supporting network policy and controls rather than specific methods that can be used to deploy the controls through physical hardware. In the context of SDN, it refers to connections, ports and data flow policies rather than the physical connection descriptions such as VLANs, IP addresses, and physical networking devices. This network abstraction layer facilities APIs that can be used to configure details about the network.

Decoupling The separation of the control plane, and the data plane, allows the network to be abstracted. The control plane is defined in general terms and manages policies. The data plane is the physical interface that acts on these policies, thus abstracting the network design, from the planning and manipulation of physical connections.

Northbound and Southbound interfaces The concept of North and Southbound traffic refers to the information exchanged between the decoupled control and data planes of the SDN. Northbound specifically refers to information from the data plane to the control plane, and Southbound refers to information from the control plane to the data plane. Restated, the Southbound interface involves the controller-to-switch interaction and is defined by protocols like OpenFlow [33]. Conversely, the Northbound interface involves the controller(s) and network services/application and the standards for this are less well defined [33].

In the context of OpenDaylight, a Northbound interface allows applications to gather information about the network used to modify the existing connection resources and capabilities such as bandwidth assigned to the network. The Southbound interface deals with the hardware, and network layers, control policies are translated to the data plane as instructions for connections, traffic management and security policies [35].

Agents and Controllers Tenants or applications interact with provisioning mechanisms that communicate with agents or controllers. This allows the administrator to reconfigure the network to meet tenant or application needs. An agent requests network resources through an API in the controller, which then provisions the requested resources. The available resources are reported back to the agent. This exchange allows the agents or tenants to dynamically provision resources during a heavy load, and then release resources when the demand is low.

Chapter 3

Methods of Implementing Reconfigurable Networks

There are a variety of methods for dynamically reconfiguring the network, each with different challenges and limitations. This section contrasts typical networking techniques with virtual networking methods.

3.1 Typical Networking Environment

In the typical environment, the system is analyzed and requirements documents are generated to realize the needs of the application. Initially the Layer 3 appliance is built out in the appropriate location and then specific security policies are added. The Layer 2 structures are built out depending on customer needs. In a rapidly changing environment, meeting the Layer 2 change requests presents a challenge. If a customer requires additional Layer 3 instances on top of the requested Layer 2 functions, it can force a complete redesign of existing security policies. Additionally, the physical connections must be considered for the deployment, taking into consideration:

1. *Are the tenant nodes connected?*
2. *Is there a need for external WAN and Internet connectivity?*
3. *As the system grows, how will load balancing be handled?*

The primary concern with traditional networking deployment models is that the physical connections have to change as rapidly as the user's needs change. This means a considerable amount of time is spent reallocating existing physical connections and facility (space) resources as well as the down time associated with reconfiguring the existing equipment or adding new appliances.

3.2 Static Networks Involving VRF and Preconfigured VLANS

The availability of preconfigured static Virtual Routing and Forwarding (VRF)s and corresponding Virtual Local Area Network (VLAN)s allows for tenants to be placed into separate service areas, where all traffic is carried between physical virtualization servers using different VLANs. This topology works well if the network is rigidly defined with fixed connections and port definitions. Access to external servers and infrastructure is handled through connection policies maintained directly on the server. This method does not allow the tenants to run services in an environment where inbound and outbound filtering is applied. This is especially true if each customer has unique and frequently changing network requirements. As the requirements change, the Access Control List (ACL)s, firewall rules, and both physical & virtual switch policies and configurations must change as well. Altering connectivity requires updating all of the traffic defining policies and connection information.

3.3 API

A common method used in reconfigurable networks is deployment automation. This methodology implies that all network appliances are managed by a centrally control system. The network administrators push out bulk changes and policies across the entire network without having to configure each device manually. In a secure environment, after verifying the correct permissions, tenants can request additional resources and these requests are pushed through the central network management system. The central management system configures the available resources with appropriate security policies and connection rules and then pushes out these changes where they are needed. All SDN methods have a need for a common interface to abstract the physical connections from the vendor specific (physical) device [5]. As each vendor uses a combination of proprietary hardware and appliance OS the associated application software commonality is accomplished through a vendor specific API.

The vendor API interacts with the OpenStack service software and acts as a common interface to translate SDN functionality into direct corresponding functions on the vendor hardware. The resulting control interface approaches the desired universal control layer envisioned by the virtual system designers without sacrificing the capabilities of the individual hardware. Additionally this method allows, within some parameters, the multiple vendor deployment capability desired by server farm administrators. This capability further eliminates the need to have one specific switch appliance vendor throughout the facility. Systems are streamlined in this virtual common platform approach. As an example, with a large group of tenants, this allows the central management system to limit the networks carried over a Layer 2 trunk to the physical virtual servers, which saves bandwidth by removing unneeded broadcast traffic. The existence of common connectivity templates in place allows each tenant to securely operate without interacting with other tenants. Further, access to external networks is provided to the tenant systems, while maintaining existing connection and security policies. Network access APIs are vendor specific, with only common functions called out specifically in the OpenFlow standard. In many cases vendors add functionality to increase performance, statistics or reliability outside of the standard in an effort to entice the use of their hardware. Additionally since switch fabric bandwidth and connection agility change the vendors port configuration, vendor syntax will often be different. The syntax for shutting down and reconfiguring ports will vary from vendor to vendor on the command line interface. The use of a vendor API allows the central management system to communicate with the physical appliance over a controller attached port, using a common control language. This allows the central control authority to authenticate and then make any necessary network configuration changes. The OpenFlow based API can make all changes at all layers, once the established security policies have been met. In addition to configuration APIs, some network appliances support third party configuration management software such as Chef, or Puppet. In most cases APIs will use a JSON like interface for human readable code and configuration data, but will have commands that are unique for their specific capabilities and platforms.

3.4 Traditional SDN

In a traditional SDN, there is no implied intelligence on the network appliance as all decisions with respect to the control plane and data plane originate from a logically central control authority. While this methodology offers flexibility, it does not however scale well, representing a recognized single point of failure. The use of traditional SDN through a single central controller works well for small deployments with high flexibility and low availability requirements [46]. Some of these limitations can be addressed by employing a clustered central controller architecture with an active/passive or active/active failover strategy. There are SDN architectures that adopt an active/active centralized controller architecture to address both scalability and resiliency requirements.

3.5 Hybrid SDN

Hybrid SDN uses a separate control plane like traditional SDN, but network devices also maintain control plane functions independently. In this model each network device still functions independently from the central controller, but also receives configurations from the central controller. The central controller handles traffic by reconfiguring the individual nodes as needed by each device. This allows a simple method for dynamically handling traffic hot-spots.

Applications can talk to the central controllers via API to get network health, or to make provisioning changes leveraging the abstraction concept. In a hybrid SDN control responsibility is both shared and dispersed, losing the central controller does not result in the loss of the entire network, only the management and configuration is crippled until the system is repaired. Individual network appliances can still be configured if central control is lost, the network is still capable of running the current applications with a temporarily frozen configuration and policies rule set. The hybrid SDN model scales better, and maintains high availability.

3.6 Overlay Network

Network overlays are accomplished by using tunneling or encapsulation techniques. This allows the extension of the network at Layer 2 from one location to another, increasing flexibility in terms of scaling the network as large as needed. The use of overlays also overcomes some of the intrinsic limitations of network appliances such as the 4096 VLAN limit. Overlay networks are beneficial in a data center environment due to low latency, higher bandwidth, and increased control over bandwidth utilization. Additionally, overlay methods extend Layer 2 networks across Layer 3 boundaries, either within the data center, or across WAN links. Keeping this local to the data center allows additional control flexibility unless dedicated paths are required. For example, if Service Level Agreements (SLAs) exist to a certain path across a provider's network. Overlay networks can extend across the WAN interface to other data centers as long as the connection is compatible in performance such as bandwidth, latency, and jitter. Essentially the network overlay is a network built on top of an existing network structure. Connectivity is accomplished through the creation of network tunnels, requiring endpoints within both connected domains which are configured to allow traffic transferred across the tunnel appearing as a contiguous Layer 2 domain.

The necessary overlay endpoints can be created manually, or via APIs. Similar overlay methods found within the data center are implemented by using encapsulation methods such as Virtual eXtensible Local Area Network (VXLAN). Protocols like VXLAN allow you to create virtualized Layer 2 networks across different Layer 3 networks and can scale up to 16 million logical networks.

3.7 OpenStack

Open Stack uses a hybrid SDN approach where the network appliances are considered stand-alone devices and function as separate entities from OpenStack. It is possible to have all Layer 2 and Layer 3 traffic preconfigured statically on the individual network device. In this configuration, Open Stack handles traffic between tenants. Some network vendors support OpenStack plugins that allow OpenStack to make port and VLAN configuration changes as part of their OpenStack interface API. OpenStack includes a network control node application called *Neutron* that facilitates SDN networking accomplished using the internal OpenStack routing engine for both inter- and intra-VLAN traffic. Neutron has the capability to communicate with the network via dynamic routing.

3.8 Implementing Neutron Routers

The use of multiple flat networks require bridge interfaces for each network connection, the addition of VLANS further complicates the setup by requiring switch and gateway configuration per instance. Neutron contains a plugin agent specifically to handle L3 connectivity. This agent allows both administrators and tenants to create routers that handle traffic between directly connected tenant network interfaces, either Generic Routing Encapsulation (GRE) or VLAN, and a single management or controller network node. Access to external provider networks, including WAN services are handled through this Neutron router structure. The external network is typically implemented as either a FLAT or VLAN provider network.

Nova compute nodes use both fixed and floating IP addresses. The fixed IP addresses are assigned to the compute instance on creation, and remain until the instance is terminated. Floating IP addresses are dynamically associated with the instance as needed. Floating IP addresses can be associated or disassociated with a instance at any time. A public or provider network involves a connection that is potentially outside of Neutron control. In a Nova network the use of 1:1 NAT translation allows for a customizable "floating" IP address implementation, it is common for the same IP that is used as the L2 address to also be used in the bridge to the hypervisor. This is accomplished by using the `iptables` configuration on the host by modifying the Source Network Address Translation (SNAT)/Dynamic Network Address Translation (DNAT) rules. Re-association of a floating IP address is accomplished by removing the rule from the `iptables` SNAT/DNAT rules list and re-associating on another instance, in this way the instance IPs remain static, only the NAT rules change.

Neutron routers act as gateways for each tenant instance using the Neutron L3 agent, instead of manipulating the `iptables` on the hypervisor. The `iptable` in the router handles the NAT translations, by instantiation of connections to Virtualized Network Interface Control (VNIC) devices connected to its ports. The floating IP addresses are procured from the provider network through pre-determined tables or using the Neutron DHCP agent. Containers[1] or VE can be instantiated without worrying about using redundant IP addresses on the same networks nor requiring the user to reset or manually load tables as part of a start up script. Access to the node within a container is only granted by using the network ID (namespace) and setting the connection in the routing tables. Attempts to access without proper credentials can be tagged and monitored easily in this configuration. This method limits the floating IP addresses to that of the WAN address space. The MAC addresses of the tenant NICs can have fixed IP addresses in the NAT tables as well as be associated with defined security group IDs. The Neutron L3 agent should be present on both the network and controller node. Once a container is established only the compute node

[1]Also referred to as Virtual Environments (VEs) in the other project report titled, "Review of Enabling Technologies to Facilitate Secure Compute Customization."

within the container have unfettered access to each other and controlled access to the external network. Nova nodes are simply added to the table in the router as created, no other management action is necessary. This also applies to VMs that perform other functions including additional routers for separate internal networks.

The use of routers in Neutron is possible using existing technology, however it is a fairly new development. Bottlenecks have been observed in the layering necessary to perform the function as it is currently being built up with existing software blocks rather than implemented as a stand-alone function. The redundancy factor is higher than normal to achieve otherwise simple NAT pairings. The beta release of LXD in Ubuntu 14.04 OpenStack and the re-writing of most of the access agents in Neutron is well underway. Preliminary reports suggest it is possible to have near zero latency network within a LXD-LXC structured environment. The list of current considerations for implementing virtual routers is given below.

A. When configuring the L3 agent using the agent config file, specifying an external network bridge, causes Neutron to associate the external NIC directly with the bridge. The attributes for "vlan" "segmentation ID", and "provider network" are ignored, Neutron assigns an IP address to its translation table from the provider network.
B. The gateway can be manually specified using the `gatewayexternalnetwork_id` attribute, otherwise Neutron looks for the gateway from the provider network if the attribute `external=true` is set otherwise, Neutron will stall if gateway not found.
C. If an external bridge is not set, Neutron uses the external interface into the Open Virtual Switch (OVS) bridge specified by the provider network from the Neutron Controller. Any subsequent network traffic is handled through the Open VSwitch flow rules present in the controller. This is the typical interface for controlling VLANs through OVS.
D. Traffic within a GRE based tenant network is limited to that network only, bridging is now through the router.
E. The Neutron router will allow directly connected tenant networks to communicate with each other freely, and the external provider network only if the router rules allow the connection. All tenant nodes are behind the Neutron router, and no longer have floating IP addresses, therefore there is no direct connection to them outside the Neutron router or within the DHCP namespace instance.

A test network (Figure 3.1) is proposed as a sandbox test using a Network Node, and Network Controller Node, and a Nova Compute Node. In the test there is only one Nova Node but more are possible, the container must be limited to one network Node and one Network Control Node however. All nodes have both Neutron control Agents and the OVS agent running, the controller node does not require the OVS agent. The Open vSwitch plugin can be replaced with a proprietary Neutron Switch interface provided by the physical switch manufacturer. The provider network is modeled using a network node instance. Connections to the networks are through bridging (br-ext) and using GRE tunnels (br-tun) set in the OVS router configuration file. Internally the VLAN interfaces are configured using bridging adapters configured using Neutron as shown in Figure 3.2.

3.9 LXC / LXD

LXD [25] is an extension of the successful LXC [24], which includes the use of Docker [15] support and similar services in establishing containers. It was envisioned by Open Stack Canonical to have the virtualization environment where LXC is the client support and LXD is the server. LXD will allow secure containers set ups for Linux based compute nodes, there is no support for other operating systems. Further the LXD/LXC pairing eliminates the redundant bridge structures that cause intrinsic delays in packet delivery by replacing the need for additional structures to perform the routing function directly within the hypervisor.

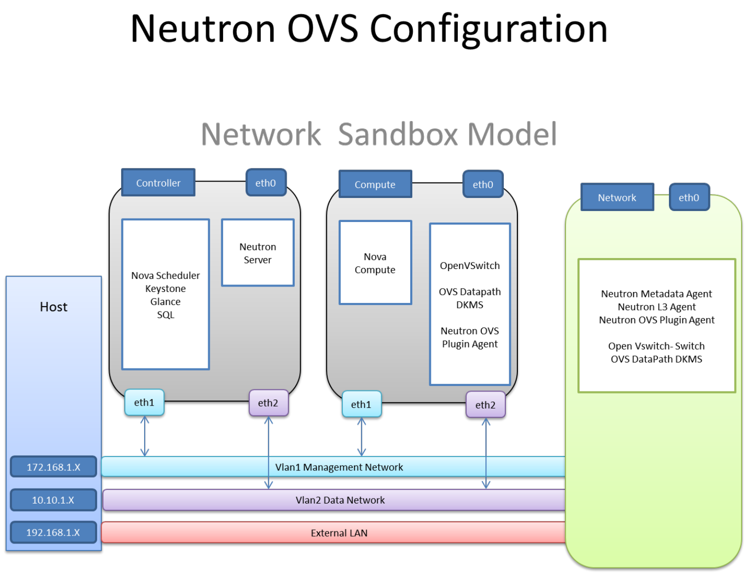

Figure 3.1. Neutron OVS SDN Router Configuration.

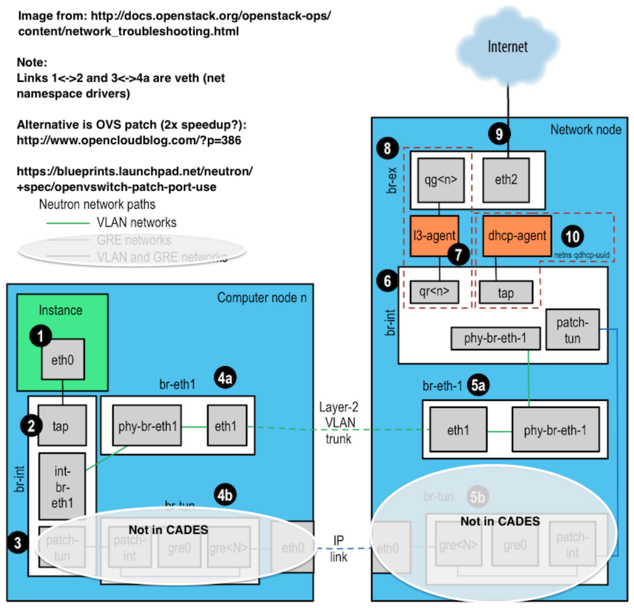

Figure 3.2. VNIC interface configuration.

Chapter 4

Key Vendors and their role in SDN

All network traffic between physical servers eventually must be connected through real network switching appliances. Each of these hardware devices are built to industry connection standards while providing proprietary hardware, software and capability to make them more desirable on the open market. In addition to the industry developers working on OpenFlow [43], OpenDaylight [35], and OpenStack [36], there are many key network vendors embracing these software platforms and tools as an industry standard. This section is a brief overview of the key players in this field and a discussion of current capabilities and contributions of each.

4.1 Arista

Arista Networks, employs the EOS (Extensible Operating System), a Linux based platform, that provides resiliency and programmability across their network products. The purpose of this extension is to provide uniformity in management, the end user does not define and manage individual network appliances as much as the entire enterprise system as a whole. Modern networks require agility and scalable provisioning to handle changes in deployment and recovery from changes or outages. They support OpenFlow, and DirectFlow. Arista supports the OpenFlow v1.3 API providing the ability to control flows through a centralized OpenFlow controller. In addition, Arista has developed DirectFlow, an Arista proprietary technology, which allows controller-less direction of flows using the capabilities within their EOS platform. Support for OpenFlow is provided through an interface on top of their DirectFlow API. Arista's support for OpenFlow on top of DirectFlow can be categorized as a hybrid SDN technology relying upon functionally independent switching infrastructure that can take configurations from a centralized controller. In addition to OpenFlow, Arista supports application interface plugins for OpenStack and was one of the first vendors to support VXLAN. Arista offers centralized management via zero touch provisioning, and uses Extensible Messaging and Presence Protocol (XMPP) [45] to configure groups of network devices. Arista supports multi chassis link aggregation (MLAG) making active use of all links in the network. Network redundancy and making all paths available is important to the modern data center and relevant to reconfigurable networks [1].

4.2 Brocade

Brocade's entry in SDN is the Brocade VCS (Virtual Control System) Enhancing the existing Linux based OS with embedded OpenFlow API capability such that the inclusion and addition of specific plugins to handle OpenFlow SDN are not necessary. Brocade favors a higher level of support for virtualization as a key feature of its entry. This is accomplished by combining the OpenFlow command control features with additional support for data plane overlay protocols such as VXLAN. Brocade supports Open Stack. In addition to selling Ethernet, and Fiber Channel equipment, they also have SDN and NFV products. They recently purchased the company Vyatta, and now offer a virtualized firewall product, and OpenDaylight based Vyatta SDN controller [6, 7, 8].

4.3 Cisco

Cisco is undoubtedly the largest player in the networking appliance market. They have embraced SDN as the path forward with Cisco ONE (Open Networks Environment). Ironically they have the most to lose from the trend toward SDN as they hold the majority of legacy systems in the marketplace. The SDN environment commoditizes Cisco's enterprise strategy, allowing competitor devices to seamlessly share the network control space. Cisco has SDN support for OpenFlow [12] on their Nexus series, in addition to their Open Network Environment initiative. Cisco also supports VXLAN. Cisco supports multi chassis link aggregation (MLAG) which they call Virtual Port Channel (VPC). Cisco is also championing another form of application driven network configuration called Application Centric Infrastructure (ACI) [11]. ACI appears to be primarily a Cisco backed initiative. For automation Cisco supports standard command-line interface (CLI), API mechanisms, and can operate with configuration management tools such as Puppet [23], or Chef [10].

4.4 Dell

Dell has partnered with BigSwitchNetworks to provide its SDN support. This means Dell will use BigSwitch's Switch Light OS. The offering works on newer switches with all features and partial features on legacy Dell switches. Dell's network lineup consists of their Force10 acquisition, and the Power Connect series. Both product lines can work well as bare metal switches using Cumulous Linux [13], or can interoperate with Vendor specific APIs through a central controller such as the NEC ProgrammableFlow Networking Suite [31]. Both Power Connect and legacy Force10 lines now support Cumulus Networks [13]. They also support a Dell backed managed SDN solution [14].

The SDN offering from HP is the most generic on the market. HP's switch fabric ASICs are the most flexible in their intrinsic control capabilities and therefore can handle virtually any possible combination of routing paths possible on their ports. Utilizing a custom version of Linux, HP has most of the common open source SDM solutions available, and even allows the user to use their own flavor of Linux, such as RedHat, SuSE, or Ubuntu on the switches if they so desire [20].

4.5 Juniper

Juniper Networks has incorporated the Junos OS as their solution to SDN and virtualization support. Their approach is to provide a platform that allows the OS to work with all of their products and to add functionality through extensions and applications running on the OS. This means that the OS has been designed to provide support for these functions rather than the functionality itself. Junos has a modular approach that allows their range of network appliances to be configured to meet specific needs. Different models therefore may have specific versions of modules that share similar features with other models but are written for that device. The underlying OS is common for all [22].

4.6 Mellanox

Mellanox is known for their high speed, low latency InfiniBand appliance. They expanded into Ethernet networking and have one rack unit on top of rack switches. These are non-blocking and have full Layer 2, and Layer 3 functions, as well as SDN support [29]. They have introduced aggregating switches into a fabric using a technology called Virtual Modular switching. This is similar to MLAG and VPC, except, if you lose one of these switches, you don't lose half of your bandwidth [28].

4.7 Vendor Conclusion

All of the major vendors are supporting SDN to various degrees. Some are embracing their own solutions, while others are embracing the open source community and standards, often including industry standards combined with proprietary capability. Cisco, and Arista are looking at making bare metal switches and using Cumulus Linux as their operating systems. The advantage to OS and SDN standards is the ability for network appliance vendors to focus on the ASIC and hardware development. The adoption of open-source software technologies by switch vendors poses challenges for differentiation in a competitive marketplace. It would be reasonable to expect that switch vendors will continue to adopt open source software and open standards while continuing to differentiate by offering more advanced features through proprietary software and interfaces. For the secure enclaves project, we will focus on the use of broadly supported SDN capabilities to alleviate reliance on vendor proprietary technologies [16].

A summary of the vendor compliance with the OpenFlow standard is given by Nunes et al. [33], which is repeated here for easy access in Table 4.1.

Maker	Switch Model	Version
Hewlett-Packard	8200zl, 6600, 6200zl, 5400zl, and 3500/3500yl	v1.0
Brocade	NetIron CES 2000 Series	v1.0
IBM	RackSwitch G8264	v1.0
NEC	PF5240 PF5820	v1.0
Pronto	3290 and 3780	v1.0
Juniper	Junos MX-Series	v1.0
Pica8	P-3290, P-3295, P-3780 and P-3920	v1.2

Table 4.1. Main current available commodity switches by makers (vendors), compliant with the OpenFlow standard (from Table-II of [33]).

Chapter 5

Testbed Description and Evaluation Plan

ORNL has constructed a testbed environment to develop and evaluate the use of HPC and cloud computing technologies. This testbed, illustrated in Figures 5.1 & 5.2, will be used to prototype SDN/NFV for resource isolation in an HPC environment. This environment consists of eight Dell C6220 nodes configured as follows:

- One OpenStack management system
- Three compute systems capable of running bare metal OS images, Virtual Machines, and Linux containers
- One GPFS storage server
- One Lustre storage server

Two DDN 10K storage systems each with dual storage controllers and over $1/2$ petabyte usable capacity. Arista 7150S network switches connecting compute and storage resources. InfiniBand is used for connectivity between storage servers and storage controllers.

To expose SDN capabilities to OpenStack, Arista provides plugins and drivers for OpenStack integration of Layer 2 and Layer 3 functionality. The Layer 2 plugin enables the OpenStack networking service (Neutron) to communicate with Arista's CloudVision eXtension (CVX) through an Arista mechanism driver over the Arista Command API (eAPI) to provision tenant networks. A typical Layer 2 OpenStack integration is shown in Figure 5.3. CVX is a series of open source extensions to Arista switches that enable them to use the open-standard XMPP protocol to establish a single view of the network via an industry-standard CLI. eAPI allows applications and scripts to have complete programmatic control over the switch. Once the API is enabled, commands using Arista's CLI syntax are accepted. Responses are machine-readable output and errors serialized in JSON, served over HTTP.

CVX has visibility of the entire network environment and provisions VLANs on switch interfaces so that the compute instances on the compute nodes have connectivity to the appropriate tenant VLANs. CVX can run in a VM or on an Arista switch itself. The Arista Layer 3 Service Plugin communicates directly with the Arista switches, either TOR or Spine, to provision routing functionality. In response to router create/delete and interface add/remove requests in the OpenStack environment, appropriate SVIs (Switched Virtual Interfaces) are created on respective switches. In future releases the Layer 3 service plugin will communicate through CVX. A typical Layer2/3 OpenStack integrated environment is depicted in Figure 5.4.

This testbed environment will be used to evaluate the ability of the Arista OpenStack integration to establish high performance network enclaves. Enclaves will be created using both the Layer 2 and Layer 3

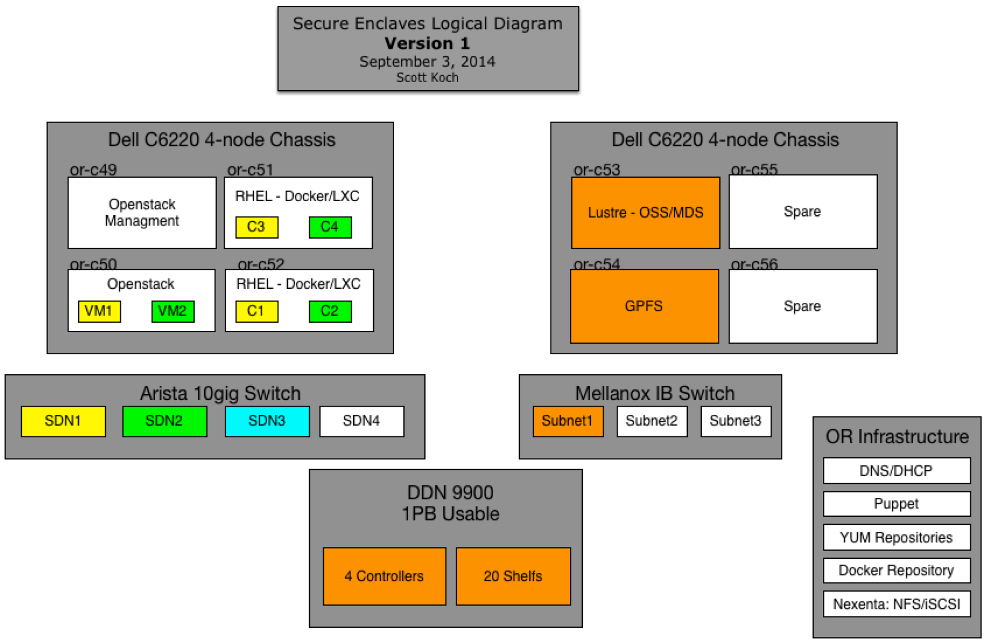

Figure 5.1. Secure Enclaves Testbed Logical Diagram.

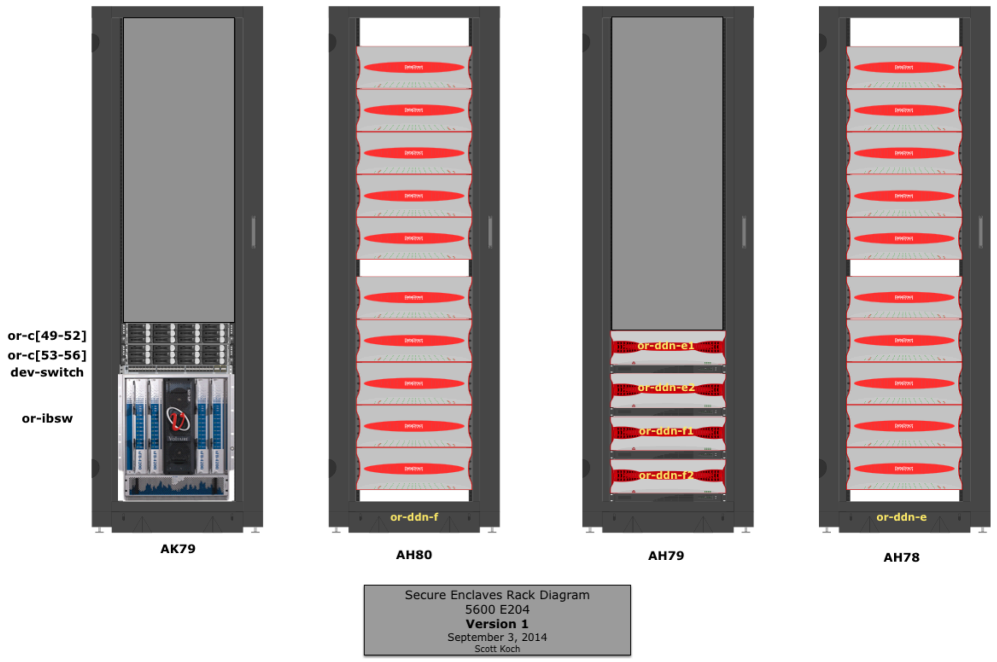

Figure 5.2. Secure Enclaves Testbed Rack Diagram.

capability. This will enable evaluation of the isolation of the tenant networks as well as the performance of the integrated environment in comparison to the virtualized environment.

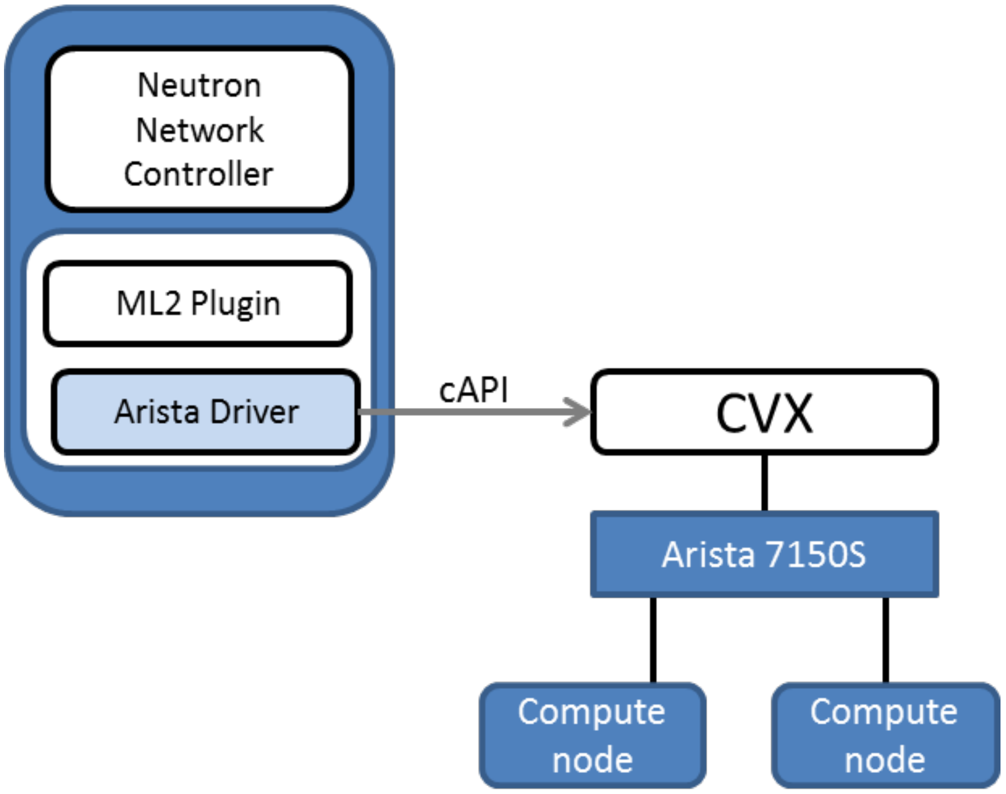

Figure 5.3. OpenStack L2 Deployment.

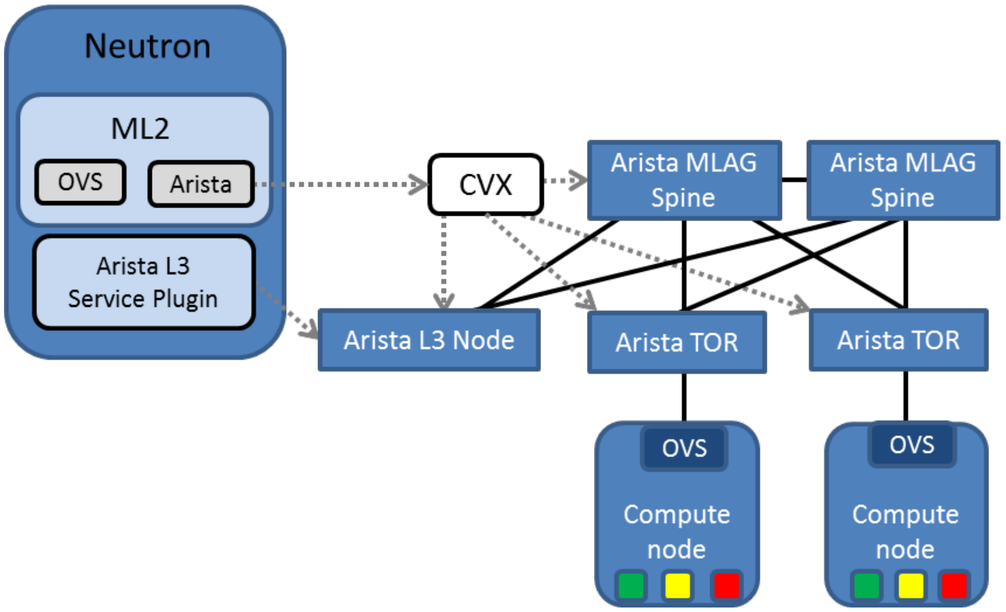

Figure 5.4. ML2 and Layer 3 Service Plugin interactions

5.1 Verification of Tenant Isolation with Respect to Compute and Storage

During evaluation of various reconfigurable network solutions in a test environment, an agent must be allowed to interact with a controller to accomplish certain networking configuration tasks. This includes creating new networks, releasing resources, and making changes to existing network resources. In the case of secure enclaves, a typical task list includes provisioning a new VLAN, attaching resources (compute nodes) and then creating a Layer 3 path from the new VLAN to other external services. Each of these tasks must then be verified to ensure that the networking and host resources are properly configured for the enclave rule set, including Layer 2 path verifications, access control lists verifications, and overlay verifications. Once a VLAN is created, it must be unique to the tenant, allowing connectivity only on the compute resources where the tenant has access. As and example, with VM based compute resources the configuration of the hypervisor must be verified to make sure that the VLAN access is allocated only to the VMs assigned to the tenant. In addition to providing isolated connectivity to compute nodes within the enclave, the tenant will need to be able to access the necessary storage resources. In a secure enclave architecture, this will require that the storage servers are accessible either via a Layer 2 path by using a VLAN specific to the storage servers or via a Layer 3 path between the compute nodes and the storage servers. In a Layer 2 scenario, each compute node resource within the enclave will need to be attached to a dynamically generated storage VLAN specific to the the tenant. Each storage server must then be attached to this VLAN as well. Alternatively, a Layer 3 path routing traffic between the compute node and storage server resources could be employed. In either case, then releasing resources (tear down of an enclave), the network, compute, and storage resources must be verified to ensure that all contiguous resources have been released and that the network configuration has been removed. Similarly, when changing existing configurations, a repeat of the test methodology when creating and releasing resources should be utilized.

Once connectivity within the compute node enclave and the storage servers are established and verified, isolation of the enclave must also be verified. A test of isolation includes an attempt to reach other known tenant enclaves. If overlays are used, all virtual endpoints will need to be verified. Network evaluation tools such as NMAP will be utilized from within an enclave to assess isolation to and from other enclaves.

Once isolation of the enclave is verified, security of the network management services must be assessed. This test will attempt to reach network management resources outside of the network management services API. This test includes using alternate interfaces such as Telnet, HTTP, HTTPS, SNMP, SSH, and attacks against well known API port vulnerabilities. An important test is to request resources that are known to be in use by other tenants. For each of these verification tasks, there will be repeat attempts to verify results and collect useful statistics.

5.2 Performance

Once tenant functionality and isolation has been assessed, performance and scalability of network reconfiguration will be explored. Initially performance measurements will focus on the time that it takes to create networks of various sizes. The measurement will begin when the agent makes the request, and will end once the controller has sent confirmation back to the requesting agent. Additionally any information that the agent requires to utilize the requested resources will be included in the survey. This test will be repeated with various types and sizes of network resources requested.

A similar test will be done for access control list requirements such as allowing certain networks, or all networks, into specific hosts on specific ports. These tests will aim to assess the performance of allocating network resources at different scales.

Scalability of network reconfiguration as a function of concurrent agents interacting with the controller, will also be tested. In this scenario multiple tenants will make simultaneous requests for networking resources, making both changes to and releasing those resources. The main metric is the time that it takes for the controller to respond to individual requests as a function of concurrent agents making these requests.

Network reconfiguration will then be tested under sustained load. In this scenario the data plane will be saturated with known test traffic and will then be verified that provisioning and configuration continues to function as expected under sustained load. This is a direct evaluation of the robustness of the controller under high load, high traffic, and high latency conditions [4].

It is important to test the functionality, and performance of your infrastructure in reconfigurable network environment. Gathering these metrics will help evaluate various solutions. In following sections, we will cover hardening the system, monitoring, scalability, reliability, monitoring and troubleshooting [37, 41, 42].

5.3 Security

SDN systems are not immune to security attacks. There are numerous acknowledged SDN attack vectors. Most of the security vulnerabilities employ exploits within both the OS implementation and the SDN API. The exploits effect the control plane, data plane, hypervisor and storage management interfaces.The use of ACL and namespace control as authentication methods for mitigating these exploits falls under the best use practice control model. As the system is virtualized, snapshots of known secure systems against what is perceived to be deployed can mitigate against unauthorized changes and configurations. Deep packet analysis techniques remain the only method for identifying packets that carry additional data used to attack and control the SDN switch network.

A particular vulnerability in SDN is the assumption that Northbound traffic to the control plane has been previously vetted by another controller on the network before reaching the data plane and therefore is readily accepted. Research is currently ongoing that involves the tagging of data packets within SDN as part of the port control process. This method acts as a port by port firewall authenticating all packets that pass through a port. This method is effective in that it has a much lower latency than deep packet sniffing or NIDS (Network Intrusion Detection System) specific appliances or software. Neutron Routers using OVS can use names space based authenticated security by blocking any and all packet frames that do not match preconfigured IP addresses on VLANs between VMs within a container, this is accomplished by using the name space token and checking it against a localized static IP list. All external or WAN traffic must pass through the Neutron Node before being passed to the OVS component on the Nova Node and therefore is not only passed through a firewall, but has a secondary authentication as to its providence. The Neutron router node using OVS has the advantage of only allowing access to the connected nodes within the specific list, or sharing the VE. Modification to the access list is only possible through the use of the Keystone or similar authentication key server and the VMs at configuration start up time. This is accomplished through the use of the Neutron Firewall API plug in or through internal configuration files within the VM enclave.

5.3.1 Application Programming Interface

The network reconfiguration API represents a well-defined attack vector that must be protected. Changes or upgrades to the API including those for plugins used on vendor devices should be authenticated through a trusted source and checked against a current accepted version list and checksum. Further authentication can be made through the use of access control lists, allowing only authenticated and authorized agents to interact with the API. Without the introduction of additional layers of security, the API's authentication, authorization, and accounting mechanisms will have to be trusted and as such should be validated. All traffic between the agents and the API should be encrypted to protect credentials, and any data about the tenant or the tenant configurations. Additionally, the API is an obvious denial of service attack vector, limiting the number of concurrent connections or connections per agent may prevent this attack vector but could limit the scalability of the service.

5.3.2 Control plane

The next security consideration is the control plane. All network appliances should use MD5 for OSPF, BGP, and other routing protocols. This will help prevent rogue elements from taking over, or corrupting your routing table. Additionally the network appliances configured by the central controller(s) should have access control lists allowing only the controllers IPs as sources, and use strong encrypted passwords. A separate subnet from tenant traffic should be used only for routing control and connect the OVS component of the VE with the Neutron control node. Tenant dynamic use policies for routing within their private domain. The attached OVS plug in should be used in conjunction with the Neutron controller to control traffic to any external network with firewall and packet authentication active. Pre-configured route maps, and other precautions such as separate OSPF keys using MD5 may need to be enforced. This will help prevent an unauthorized controller from configuring network elements. Traffic to the network management should use encryption for all access. The network management subnet should only contain network devices, and not be accessible by tenant networks, or outside networks.

5.3.3 Data Plane

Each tenant should be in their own broadcast domain, and should block broadcast, or unicast traffic including that which originates from other tenants. All tenants still pass through the same data plane, so if there is a potential for Denial of Service vector in protocols such as Multichassis Link Aggregation (MLAG), or Virtual Port Channel (VPC). Hash bugs in protocols such as Link Aggregation Control Protocol (LACP) could exist where traffic will not load balance correctly causing congestion. The best practices for these risks are using up to date code releases and standardizing the code used across the system. Anomaly detection methods such as monitoring graphs automated system trend/alert to identify drastic traffic pattern shifts are also effective.

5.3.4 Hypervisor management

Hypervisor management should only have hypervisors in their subnet. This network should be protected by access control lists, and not be accessible by tenants, or the outside world. This network should only include trusted management hosts, and controller/provisioning systems. All accounts on these systems should use remote authentication, and have strong passwords as a fall back mechanism. All traffic to these systems should use encryption methods.

5.3.5 Storage Management

Storage management should only have storage devices in their subnet. This network should be protected by access control lists, and not be accessible by tenants, or the outside world. This network should only include trusted management hosts, and controller/provisioning systems. All accounts on these systems should use remote authentication, and have strong passwords as a fall back mechanism. All traffic to these systems should use encryption methods.

5.3.6 Security Summary

SDN attack vectors similar in nature to those found in standard wired and wireless protocol networks. Each system and application changes the operating dynamic and therefore both the potential and type of attack that is possible. Even considering the multiple possible exploits, only the ones that are based on specific bugs are the ones that are most difficulty to detect. The majority of exploits are based on performance hits due to DOS, type jamming or blocking of traffic. All of these base exploits are designed to cause both frustration and the causal dropping of security systems in an effort to find the cause system disruption. Methodical analysis and authentication practices should be maintained to ensure that these exploits have little to no effect on the system [19, 40].

5.4 Scalability

There are interesting statements concerning scalability and SDN as noted in the following: The first SDN controller can serve only 30,000 flow initiation requests per second while keeping the response time less than 10ms. It is challenge to serve more flows while keeping the response time within a reasonable small duration.

Thus, understanding and quantifying the scalability of the SDN controller is a critical problem for successful adoption of SDN for large scale networks or networks with many flows. The SDN Control plane is limited in its ability to handle extremely wide data plane domains. As the data plane size increases the reaction time of the control plane exhibits equivalent latency. The scalability of the control plane is therefore dependent on its processing capability [21].

High speed and enterprise backbone switches, such as those from Mellanox, allow custom switch fabric deployment using VMS (Virtual Modular Switch), and 40 Gb/s connections to scale up to 720 ports using only two tiers. Similar networks can be realized by using Arista (MLAG), or Cisco (VPC). In the case of such large build outs a single controller does not have the processing power to handle the entire high speed traffic switching . Scaling this type of system requires the networking elements to have some level of autonomy and still functional independent of the central controller. This is consistent with the movement towards Hybrid-SDN architectures, which applies directly to virtualization and enclaves in HPC environments [38].

There are two accepted approaches to scaling SDN to large data center deployments. One is to build separate SDN instances, and let L3 traffic traverse one instance to another as needed. The majority of the traffic in each instance would be to an externally connected network or WAN, making the migration from one SDN data center instance to another difficult. The other method of scaling data centers, is to have a load balancing mechanism where tenants exist in different data centers. This method can prove problematic if there were performance requirements (latency or bandwidth) between tenants as latency and bandwidth constraints will be far more pronounced for inter data-center workloads [21].

5.5 Reliability and Availability

Under normal conditions SDN is reliable as long as there are no heavy performance demands. Losing the central controller will cause a total loss to all network functionality [3]. Note, Ulema [44] reviews challenges of reliability and availability concerning the various types of reconfigurable networks.

A hybrid SDN allows switches and routers to continue functioning even if the central controller fails. Hybridization allows the controller to handle many more connections through delegation of the lower level intra-tenant traffic switching. When a controller fails in a hybrid SDN environment, you lose the ability to make changes in an easy/automated way [2].

Chapter 6

Conclusion

Supporting multi-tenant environments within HPC systems holds the promise of supporting a diverse set of workloads at significantly higher levels of performance and scalability than a traditional utility compute cloud environment. Traditional cloud computing environments address the security challenge of multi-tenancy through judicious use of full machine virtualization, network virtualization, and per-tenant storage. This approach sacrifices performance, scalability, and usability in favor of secure multi-tenancy. Our work is focused on providing multi-tenant environments, 'secure enclaves', at very low overhead through the use of alternative techniques to traditional cloud environments.

In this report we review the state-of-the-art in SDN and NFV as one technique for providing isolation of enclave resources at very low overhead when compared to full network virtualization. Through the use of hybrid-SDN architectures secure enclaves can be configured on demand by isolating compute and storage resources using layer 2 and layer 3 based approaches. This hybrid-SDN approach allows for the use of high-performance switching technologies that can be configured to isolate resources without the overhead of software based network virtualization. While the promise of hybrid-SDN architecture holds promise, it is not without Manageability, security, and scalability challenges. Orchestration frameworks such as OpenStack help address Manageability but additional work is required to access the security of these systems and their ability to scale to support highly concurrent enclave configuration.

6.1 Synopsis

A brief background of SDN and NFV is discussed in Section 2. This is followed by an overview of relevant terminology and background concepts in Section 2.2. This includes discussion of state-of-the-art technologies and emerging standards such as OpenFlow and OpenDayLight.

Section 3 provides an overview of methods of implementing reconfigurable networks with specific focus on utilizing reconfiguration as a mechanism to support secure enclaves. This overview of methods includes traditional SDN, Hybrid SDN, and the use of OpenStack Neutron. A more detailed treatment of Neutron based approaches is provided in Section 3.8

SDN and NFV vendor technologies are reviewed in Section 4, including Arista, Brocade, Cisco, Dell, Juniper, and Mellanox. Each of these vendors are supporting SDN and NFV to various degrees either by adoption of open source community technologies or through a combination of proprietary technologies and open APIs. A summary of vendor compliance to the OpenFlow standard is presented in Section 4.7.

Section 5 describes the ORNL secure enclaves testbed environment and provides an overview of our evaluation plan for this testbed as it relates to SDN and NFV. The evaluation plan includes:

- Verification of isolation of compute and storage resources within an instantiated enclave
- Performance benchmarking of network reconfiguration to assess how long instantiating an enclave will take within the testbed and how performance is impacted by concurrent requests
- Assessment of the security of the API and network control plane for network reconfiguration in the testbed
- Assessment of security of the data plane within the testbed
- Assessment of security of both hypervisor and storage management within the testbed

Finally, an overview of suggested best practices for architecting for scalability, reliability, and availability is provided in Sections 5.4 and 5.5 .

In Section 6.2 an overview of notable issues and limitations for SDN/NFV is discussed. Although SDN and NVF are becoming the operational standards of large compute resource deployments, the implementation of these standards is still being refined [33]. Our goal in this assessment is to identify secure practices and verify there is proper control over security and management of networking resources within the defined environment. As the available vendor APIs and technologies mature toward a more workable standard, these methods and deployment rules will act as a framework for evaluation and verification of secure networking in the enclave environment.

6.2 Observations

There are several limitations and issues with current networking technologies ranging from vendor dependence, technology complexity & management overheads and scalability.

Monolithic Vendor Dependence The first notable limitation is the monolithic vendor dependence. Large-scale data centers require routers and switches to meet the needs of the core network. The number of vendors providing these large-scale solutions is small and each have developed proprietary SDN technologies alongside open standards such as OpenFlow. The adoption of proprietary SDN interfaces to orchestrate these large-scale resources may result in vendor lock-in. Given this, insulating applications from these proprietary SDN interfaces should be a top priority either by only exposing open standards based APIs or through the development of middleware that insulates the application from the underlying proprietary API. In the context of secure enclaves we will focus on the use open standards based APIs or the use of middleware such as OpenStack Neutron that will then interface with vendor proprietary APIs through Neutron plugins.

Complexity A dynamic reconfigurable network environment will let users, or tenants, request resources to include compute, storage, and networking. This functionality is made possible through standards based APIs that can control the configuration of individual networking components. While an API simplifies the mechanism by which network configuration takes place, the complexity of configuring many individual components to satisfy what might appear to be a simple tenant requirement remains. By using standard templates for common requirements that can be layered upon one another we can manage this complexity while simultaneously ensuring that network security policies can be verified and enforced. Using a template based approach, common low-risk configurations can be configured on-demand by the tenant, while other templates might require approval through a formal change request process. For example one tenant wishing to communicate with another tenant would require both tenants to agree and potentially be approved by a third-party. Once the request is reviewed, the tenant could be authorized to use the specified template.

Scaling Issues Vendors have their own method of managing a large number of devices and different ways of building large non-blocking fabrics. These design paradigms may include:

- Leaf and spine, to host.
- Leaf and spine to top of rack.
- Standard core, distribution, access models.

Each of these designs scale differently and optimal placement of a workload in these fabrics is dependent on a variety of factors [18]. Supporting multiple tenants within these environments while providing optimal data plan performance and scalability to meet tenant requirements will require a thorough understanding of the overall architecture and how compute and storage resources are interconnected within the networking architecture. For the secure enclaves project we will make the simplifying assumption that tenants (enclaves) will be placed on compute resources that are interconnected in a fully non-blocking network. Orchestrating optimal placement of enclaves within alternate networking architectures based on performance and scale requirements, while an interesting challenge, will not be addressed by our initial work [18, 39].

Another important aspect of scalability is the number of concurrent isolated enclaves that can be supported within a single network fabric. One mechanism of implementing isolation of enclaves is to map enclaves to one or more distinct VLANs. Under IEEE 802.1Q the maximum number of VLANs is limited to 4,094 (due to a 12-bit VID field minus reserved values 0x000 and 0xFFF). Using this technique would limit the number of supported enclaves within a single fabric to 4,094. Latest generation switching technologies that provide support for VxLAN scale to supporting up to 16 million logical networks. This is accomplished by encapsulating layer 2 Ethernet frames within layer 4 UDP packets. Many switch vendors and Open vSwitch are now offering VxLAN support [32, 26, 9].

6.3 Future Plans

We conclude with a brief overview of plans moving forward. The current testbed is now being configured to support hybrid-SDN based isolation techniques and will provide a platform on which we can complete our evaluation plan. Virtual network based approaches to resource isolation will be evaluated using Open vSwitch and Neutron plugin applications as discussed in Section 3.8. With the exception of the external physical router all of the virtualization based isolation can be tested in this manner. We will then compare the performance and security of these approaches and provide more detailed recommendations based on our assessment.

6.4 Acknowledgments

This work was supported by the United States Department of Defense (DoD) and used resources of the DoD-HPC Program and the Compute and Data Environment for Science (CADES) at Oak Ridge National Laboratory.

Bibliography

[1] Software driven cloud networking, 2014. Arista Inc. URL: `http://www.arista.com/en/products/software-driven-cloud-networking/articletabs/0` [cited 20-dec-2014].

[2] Charlie Ashton. Demystifying software-defined networking, 2014. Allied Telesis. URL: `http://www.alliedtelesis.com/userfiles/file/WP_Demystifying_SDN_RevA.pdf` [cited 20-dec-2014].

[3] Charlie Ashton. Don't confuse 'high availability' with 'carrier grade', April 2014. SDN Central. URL: `https://www.sdncentral.com/education/dont-confuse-high-availability-carrier-grade/2014/04/` [cited 20-dec-2014].

[4] S. Azodolmolky, P. Wieder, and R. Yahyapour. Scalable Software-Defined Networking Deployment. In *Second European Workshop on Software Defined Networks (EWSDN)*, pages 68–74, Berlin, October 2013. IEEE. URL: `http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&tp=&arnumber=6680561&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel7%2F6679578%2F6680541%2F06680561.pdf%3Farnumber%3D6680561` [cited 20-dec-2014].

[5] Siamak Azodolmolky. *Software Defined Networking with OpenFlow*, volume 1. Packt Publishing Ltd, first edition, October 2013. URL: `https://www.packtpub.com/networking-and-servers/software-defined-networking-openflow` [cited 20-dec-2014].

[6] Brocade Vyatta controller, 2014. Brocade Inc. URL: `http://www.brocade.com/products/all/software-defined-networking/brocade-vyatta-controller/index.page` [cited 20-dec-2014].

[7] Network Functions Virtualization (NFV), 2014. Brocade Inc. URL: `http://www.brocade.com/products/all/network-functions-virtualization/index.page` [cited 20-dec-2014].

[8] OpenStack overview, 2014. Brocade Inc. URL: `http://www.brocade.com/solutions-technology/technology/openstack/index.page` [cited 20-dec-2014].

[9] Jefferey Butt. Cisco CTO warrior software-only SDN has 'limitations'. eWeek Online Magazine, June 2013. URL: `http://www.eweek.com/networking/cisco-cto-warrior-software-only-sdn-has-limitations.html` [cited 20-dec-2014].

[10] Chef: Automation for Web-Scale IT. URL: `https://www.chef.io/` [cited 21-dec-2014].

[11] Cisco application centric infrastructure, 2014. Cisco Inc. URL: http://www.cisco.com/c/en/us/solutions/data-center-virtualization/application-centric-infrastructure/index.html [cited 20-dec-2014].

[12] Cisco plug-in for OpenFlow, 2014. Cisco Inc. URL: http://www.cisco.com/c/en/us/td/docs/switches/datacenter/sdn/configuration/openflow-agent-nxos/cg-nxos-openflow.pdf [cited 20-dec-2014].

[13] Cumulus Linux hardware compatibility list, 2014. Cumulus Networks. URL: http://cumulusnetworks.com/support/linux-hardware-compatibility-list/ [cited 20-dec-2014].

[14] Dell and the software defined network, 2014. Dell Inc. URL: http://en.community.dell.com/techcenter/networking/w/wiki/4904.dell-and-the-software-defined-network [cited 20-dec-2014].

[15] Docker: An open platform for distributed applications for developers and sysadmins. URL: https://www.docker.com [cited 05-dec-2014].

[16] Jim Duffy. Cisco, Arista disaggregating? Network World Online Magazine, 2014. URL: http://www.networkworld.com/article/2844941/cisco-subnet/cisco-arista-disaggregating.html [cited 20-dec-2014].

[17] Open Networking Foundation. Software-defined networking: The new norm for networks, April 2012. URL: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf [cited 20-dec-2014].

[18] Dan Froelich. PCI express 4.0 electrical previews parts i & ii, 2014. PCI SIG. URL: https://www.pcisig.com/developers/main/training_materials/get_document?doc_id=b5e2d4196218ec017ae03a8a596be9809fcd00b5 [cited 20-dec-2014].

[19] Scott Hogg. SDN security attack vectors and SDN hardening, 2014. NetworkWorld Online Magazine. URL: http://www.networkworld.com/article/2840273/sdn/sdn-security-attack-vectors-and-sdn-hardening.html [cited 20-dec-2014].

[20] SDN infrastructure technology, 2014. HP Inc. URL: http://goo.gl/XLErKS [cited 20-dec-2014].

[21] Jie Hu, Chuang Lin, Xiangyang Li, and Jiwei Huang. Scalability of control planes for software defined networks: Modeling and evaluation. In *Proceedings of the IEEE/ACM International Symposium on Quality and Service (IWQoS'14)*. IEEE, 2014. URL: http://www.cs.iit.edu/~xli/paper/Conf/scale-SDN-IWQOS14.pdf [cited 20-dec-2014].

[22] Software Defined Networking, 2014. Juniper Inc. URL: http://www.juniper.net/us/en/products-services/sdn/index.page [cited 20-dec-2014].

[23] Puppet Labs. Puppet Documentation Index. URL: https://docs.puppetlabs.com/puppet/ [cited 02-dec-2014].

[24] LXC - Linux Containers: Userspace tools for the Linux kernel containment features. URL: https://linuxcontainers.org [cited 19-nov-2014].

[25] LXD: The Linux Container Daemon. URL: `http://www.ubuntu.com/cloud/tools/lxd` [cited 30-nov-2014].

[26] Bob Lynch. OpenFlow: Can it scale? SDN Central, June 2013. URL: `https://www.sdncentral.com/technology/OpenFlow-sdn/2013/06/` [cited 20-dec-2014].

[27] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: Enabling innovation in campus networks. Technical report, March 2008. Stanford University, University of Washington, MIT, Princeton University, University of California Berkeley, Washington University in St. Louis. URL: `http://archive.openflow.org/documents/openflow-wp-latest.pdf` [cited 20-dec-2014].

[28] Mellanox OpenStack and SDN/OpenFlow Solution Reference Architecture, 2014. Mellanox Inc. URL: `http://www.mellanox.com/sdn/pdf/Mellanox-OpenStack-OpenFlow-Solution.pdf` [cited 20-dec-2014].

[29] Mellanox's software defined networking (SDN), 2014. Mellanox Inc. URL: `http://www.mellanox.com/sdn/` [cited 20-dec-2014].

[30] Thomas D. Nadeau and Ken Gray. *SDN: Software Defined Networks*. O'Reilly Media, first edition, September 2013.

[31] ProgrammableFlow networking, 2014. NEC Inc. URL: `http://www.necam.com/SDN/` [cited 20-dec-2014].

[32] The scaling implications of SDN, June 2011. NetworkHeresey.com. URL: `http://networkheresy.com/2011/06/08/the-scaling-implications-of-sdn/` [cited 20-dec-2014].

[33] Bruno Astuto A. Nunes, Marc Mendonca, Xuan-Nam Nguyen, Katia Obraczka, and Thierry Turletti. A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys & Tutorials*, 16(3):1617–1634, Third 2014. `doi:10.1109/SURV.2014.012214.00180`.

[34] FAQ: What is OpenDaylight?, 2014. OpenDaylight.org. URL: `http://www.opendaylight.org/project/faq#1` [cited 20-dec-2014].

[35] OpenDaylight technical overview, 2014. OpenDaylight.org. URL: `http://www.opendaylight.org/project/technical-overview` [cited 20-dec-2014].

[36] OpenStack: Online Documentation. URL: `http://docs.openstack.org` [cited 21-dec-2014].

[37] OpenStack: SDN Performance, 2014. URL: `https://www.OpenStack.org/summit/OpenStack-summit-atlanta-2014/session-videos/presentation/software-defined-networking-performance-and-architecture-evaluation` [cited 20-dec-2014].

[38] Ivan Pepelnjak. OpenFlow SDN is not a silver bullet for network scalability, 2014. High Scalability Online Magazine. URL: `http://highscalability.com/blog/2012/6/4/OpenFlowsdn-is-not-a-silver-bullet-for-network-scalability.html` [cited 20-dec-2014].

[39] Arjun Roy, Kenneth Yocum, and Alex C. Snoeren. Challenges in the emulation of large scale software defined networks. University of California, San Diego, 2013. URL: `http://cseweb.ucsd.edu/~snoeren/papers/forgery-apsys13.pdf` [cited 20-dec-2014].

[40] SDN security challenges in SDN environments, 2014. URL: `https://www.sdncentral.com/security-challenges-sdn-software-defined-networks/` [cited 20-dec-2014].

[41] Performance aware software defined networking, 2013. Sflow.com. URL: `http://blog.sflow.com/2013/01/performance-aware-software-defined.html` [cited 20-dec-2014].

[42] Amin Tootoonchian, Sergey Gorbunov, Yashar Ganjali, Martin Casado, and Rob Sherwood. On controller performance in software-defined networks. In *Proceedings of the 2nd USENIX Workshop on Hot Topics in Management of Internet, Clound and Enterprise Networks and Services (Hot-ICE'12)*, April 2012. University of Toronto/ICSI, University of Toronto, Nicira Networks, Big Switch Network. URL: `https://www.usenix.org/conference/hot-ice12/workshop-program/presentation/tootoonchian` [cited 20-dec-2014].

[43] Vivek Twari. SDN and OpenFlow for beginners with hands on labs. First edition. URL: `http://www.amazon.com/SDN-OpenFlow-beginners-hands-labs-ebook/dp/B00EZE46D4/ref=sr_1_1?ie=UTF8&qid=1410613111&sr=8-1&keywords=SDN` [cited 20-dec-2014].

[44] Mehmet Ulema. Vulnerabilities and opportunities in SDN, NFV, and NGSON, 2014. IEEE CQR 2014 International Workshop Emerging Technology Reliability Roundtable. URL: `http://www.ieee-cqr.org/2014/ETR-RT/Ulema_IEEE-ETR-RT-2014_Vulnerabilities%20in%20SDN%20NFV%20NGSON_12May2014.pdf` [cited 20-dec-2014].

[45] RFC-3920: Extensible Messaging and Presence Protocol (XMPP): Core, October 2004. IETF Network Working Gropu, P. Saint-Andre, Ed. URL: `http://www.ietf.org/rfc/rfc3920.txt` [cited 21-dec-2014].

[46] S.H. Yeganeh, A. Tootoonchian, and Y. Ganjali. On scalability of software-defined networking. *IEEE Communications Magazine*, 51(2):136–141, February 2013. `doi:10.1109/MCOM.2013.6461198`.