Future Controller Design and Implementation Trends in Software Defined Networking

Wajid Hassan[1] and Tamer Omar[2]
[1]Indiana State University, Terre Haute, IN, USA
[2]California State Polytechnic University, Pomona, CA, USA
Email: wajidhassan@yahoo.com; tromar@cpp.edu

Abstract**—**The centralization of network control and programmability coupled with network virtualization results in increased efficiency and flexibility with routine deployment activities and even major network design changes. This research attempts to explore the possibilities of Software Defined Network (SDN). For this purpose we have designed a novel testbed which can be used to implement and measure performances and features of many different kinds of SDN controllers. The paper provides a history of the development of SDN including the driving factors and the impact SDN has on today's enterprise networks. This is followed by an overview and discussion of the SDN controller OpenContrail, a cloud based platform that includes the key concepts of network virtualization, programmability, automation, and analytics. Then, an SDN solution leveraging the controller is implemented within the confines of a test environment, after which various OpenContrail use cases are discussed.1

Index Terms—cloud computing, OpenContrail, NFV, SDN controllers, openstack

I. Introduction

Today, the concepts of programmable networks and network function virtualization (NFV) are topics of great interest and are often discussed within the context of Software Defined Networking (SDN). SDN is viewed as the next step in the evolution of computer networks as it promises to finally decouple the control plane from the network nodes moving it to a centrally located controller. Key activities such as network support, administration, and implementation are aggregated through the SDN controller via programmable interfaces and automation rather than the more traditional Command Line Interface. NFV extends upon the initial concept of SDN by proposing that network devices such as routers, switches, firewalls, and load balancers be implemented as virtual machines on commodity hardware rather than proprietary, application-specific hardware.

OpenFlow is the most widely known SDN protocol, but the SDN concept of programmable networks has been around for some time [1]. Also, though the term network function virtualization (NFV) is a recent development, the idea of network virtualization is again not a new concept. Protocols such as X.25, Frame-Relay, and Virtual Local Area Networks (VLANs) are rooted in virtualization and have been in existence for many years [1]. Additionally, more recent virtual overlay protocols such as Transparent Interconnection of Lots of Links (TRILL), Cisco FabricPath, Virtual Extensible Local Area Networks (VXLAN) and Network Virtualization using Generic Routing Encapsulation (NVGRE) have been explored for use in data center environments [2].

This paper presents the design and basic components of SDN solutions. Our contribution in this paper is the design of a new testbed which can be used to implement different controllers in a simplistic manner. The intent of the testbed is to compare the performance and features of different SDN controllers and SDN Solutions. This test bed uses general purpose servers.

The remainder of this paper is organized as follows. Section 2 reviews and shows efforts in standardizing the SDN controller architecture, SDN standards and mainstream SDN protocols. Section 3 discusses the drivers for SDN which are promoting the design of many and various types of SDN solutions. It also introduces the OpenContrail Controller and discusses its architecture. Network Design of the testbed is discussed in this section. Section 3 presents the emulation and the results of the experimentation done with OpenContrail. A conclusion of this paper and future works are presented in Sections 4. Section 5 is an appendix that present the code used for configuring the system.

II. Software Defined Networks

A. SDN Architecture Overview

SDN is the decoupling of the control and data planes of the network, wherein all network policy and logical state information are centralized and the underlying network infrastructure is abstracted from both the network and upper layer applications [3]. The basic building blocks of the SDN model are represented in Fig. 1:

- The SDN controllers,

Manuscript received December 20, 2017; revised May 2, 2018.
The authors would like to acknowledge the work and assistance of Mr. Torrell Griffin and Mr. Jimmy Gibson from East Carolina University whose work and assistance was invaluable for the initial part of the project. We would also like to thank Nexius Insight Inc (B. Yond) whose open compute project lab resources were used in the research of the Open Contrail solution.
doi:10.12720/jcm.13.5.209-217

- Open network forwarding devices (e.g. switches, routers)
- Network operating system providing a set of SDN protocols
- Application programmable interfaces (APIs).[4]

There are several existing SDN controllers developed by academia and industry. A list of open source SDN controllers and commercial SDN solutions are presented in Table I. Many of the commercially available controllers are based on OpenDayLight (ODL). The OpenDaylight project is an open source modular SDN controller that uses open protocols to provide centralized, programmatic control and network device monitoring. ODL possess a well-defined northbound APIs, as well as support for a variety of southbound protocols including OpenFlow and NETCONF. The reason of the variation of available controllers in market is that different SDN solutions exist to cater to various network needs and one size fits all is not a possibility.

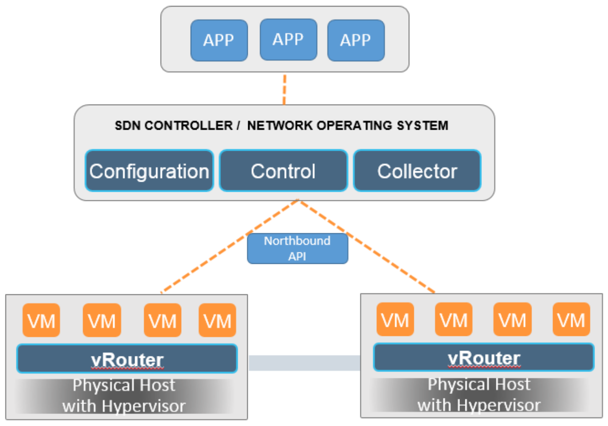
APP
APP
APP
SDN CONTROLLER / NETWORK OPERATING SYSTEM
Configuration
Control
Collector
Northbound API
VM
VM
VM
VM
vRouter
Physical Host with Hypervisor
VM
VM
VM
VM
vRouter
Physical Host with Hypervisor

Fig. 1. SDN basic building blocks

TABLE I: SDN SOLUTIONS

Open Source SDN Controllers	
Flood Light	OpenMUL
LOOM	ONOS
OpenContrail	Ryu (supported by NTT)
OpenDaylight	Trema
POX	Beacon
Commercial SDN Controllers	
Big Switch Big Cloud Fabric	Juniper Contrail
HP Virtual Application Networks (VAN) SDN Controller	Cisco Application Centric Infrastructure (ACI)/Application Policy Infrastructure Controller (APIC).
Brocade Vyatta Controller	Plexxi Big Data Fabric

1) *SDN controllers*

The SDN controller can range from a somewhat simple standard server to a complex network of virtual machines, either of which needs to run one of several available network operating systems [5]. The SDN controller is the conduit through which the network may be programmed by applications allowed to interface with the SDN controller via different APIs. These applications can be user or vendor developed for the purpose of management, deployment and support. Controller APIs can interface with monitoring tools, troubleshooting tools and other network functions, such as probes, sniffers, firewalls, intrusion detection systems, and load balancers in order to build very precise targeted network policies which is in turn distributed to the network forwarding devices. SDN controllers also use APIs in deploying multiple controllers to interface with peer controllers. These APIs are required to maintain network state synchronization between different network forwarding domains and to provide redundancy in both single and multiple domain deployments.

The policy output of the SDN controllers are distributed to the network forwarding devices in the form of a flow [5]. Network flows, manipulation, and distribution of network flows is a fundamental concept in the SDN design model. Network flows are basically the source to destination mapping of packets requiring the same forwarding treatment. The controller must also maintain a database of all active flows in order to ensure consistent network state. The basic functionality of the SDN controller can be summarized as follows.

- Flows are created either manually or dynamically via SDN related protocols and APIs.
- Flows are saved to the controller database.
- Flows are distributed to all applicable network forwarding devices and peer controllers.

2) *Network forwarding device*

The next required element is a SDN capable network forwarding device. SDN's centralizing of the control plane may increase the time required for the network forwarding device to obtain a valid network flow. This, in turn, could also increase the processing demand on the network forwarding device. In the SDN models, the network forwarding device is primarily accountable for only the data plane or packet forwarding, but it still requires substantial CPU and memory resources in order to maintain the data plane and to run SDN related protocols.

3) *Network operating system*

The final required element is a protocol that facilitates communication between the SDN controller(s) and the network forwarding devices. This operating system includes the protocols primarily responsible for the creation and distribution of the network flows. Each time the network forwarding device receives a packet for which it has no related network flow; it will interact with the SDN controller by means of this SDN protocol in order to communicate the need for a new network flow. The controller then responds to the network forwarding device, via the same protocol, with a network "flow decision". It is worth pointing out the phrasing "flow decision" is intentional, as this interaction does not necessarily result in a new network flow.

4) *Application Programmable Interfaces (APIs)*

In a SDN network environment, APIs are used to communicate between the SDN Controller and the diverse set of applications and services running on top of the network. These APIs are meant for efficient orchestration and automation catering to diverse requirements of the applications such as Load balancer, Firewall and IDS/IPS. Since there are variety of applications and their needs are varied, there is not a single API standard and many different interfaces exist at this time. Northbound APIs, which allow the network components to communicate with the upper layer network elements are currently in the process of standardization. SDN Northbound APIs are also used to integrate the SDN Controller with automation stacks, such as Puppet, Chef and Ansible, as well as orchestration platforms, such as OpenStack.

B. *SDN Standards*

OpenFlow is not the only SDN standard in existence today, but it is the most prevalent. ForCES and SoftRouter are two protocols that were developed for similar purposes. Though ForCES and SoftRouter are not as widely known as OpenFlow, they have played a key role in shaping the current SDN landscape.

1) *OpenFlow*

OpenFlow is by far the most widely known SDN protocol standard. It is an open source protocol originally proposed and implemented in a series of research deployments at Stanford University. OpenFlow allows the control plane (controller) to program the data plane (network forwarding device) and to track the overall data plane state. [6]

2) *Forward and Control Element Separation (ForCES)*

ForCES is a SDN standard defined by Internet Engineering Task Force (IETF). It is viewed as more of a predecessor to OpenFlow and it proposed the separation of IP control and data plane without changing the overall network architecture. ForCES has not been widely deployed due to significant gaps in the standard related to the controller-switcher communication rules. [6]

3) *Soft router*

The SoftRouter model demonstrates a distinct separation between the control and forwarding planes categorized as the physical view and the routing view simultaneously. The physical view is composed of a number of interconnected Forwarding Elements (FE) and Control Elements (CE). The forwarding elements are standard routers minus local complex control logic, while the control element is a network server providing the control logic to the forwarding elements. The logical topology is composed of Network Elements (NE), which are made up of logical network interfaces, port groupings, and the associated CE tasked with controlling those groups. [6]

C. *SDN Drivers*

SDN has many applications and drivers, some of the greatest benefits are supporting network management, carrier optical networks, data centers, Internet of Things (IoT), and cloud computing. Fig. 2 depicts the various drivers of Software Defined Networking.

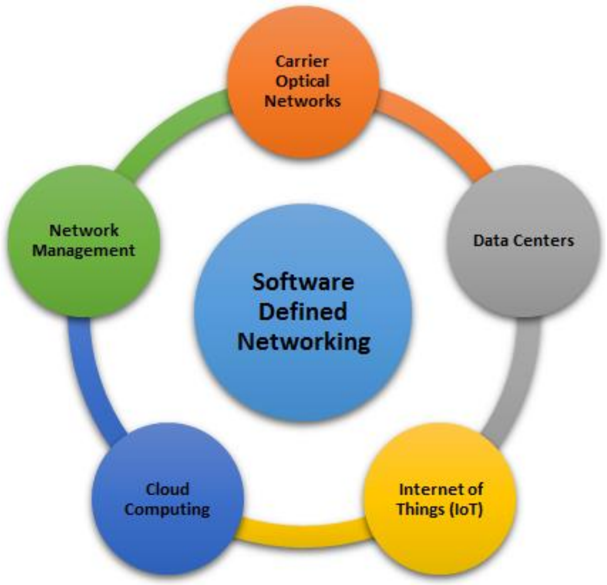

Fig. 2. SDN drivers

1) *Network management*

Network management policies today are often implemented via manual combination of third party management tools supplemented with custom scripting. SDN introduces the possibility of network management frameworks, such as Procera, which use the SDN protocols and APIs to collect events from external sources, such as intrusion detection and prevention systems, network probes, device logs, etc. This information is then sent to a policy engine in order to produce a policy that can then be distributed to network forwarding devices by the SDN controller [7].

2) *Carrier optical networks*

SDN models are of a technology agnostic nature making them ideal for providing a single point of control for carrier networks consisting of both circuit switching and packet switching technologies. These models provide optical carrier networks with more granular management and control systems, while allowing a faster time to market for new services [1]. One of these emerging architectures is transport SDN (T-SDN), which does just this by implementing improvements such as data center bandwidth on demand and virtual transport network services [8]. Together these service offerings provide the customer precise, dynamic control of how much bandwidth they need and how that bandwidth is consumed.

3) *Data centers*

SDN is a key element in the movement toward the software defined data centers (SDDC). The dynamic nature of the software and applications makes them the driving force in the data center. However, hardware profile of the data center is much slower to adapt to change. Virtualization has provided the means to decouple the software from the underlying hardware starting in the compute and storage areas and now, with SDN, in the network area as well. This allows the overall service profile of the data center to change as quickly as the software and applications require. Software as a service (SaaS), compute as a service (CaaS), and, more to the point, network as a service (NaaS). NaaS is the core principle of SDN in its ability to pool the underlying network infrastructure and provision the use of that pool through the SDN controller. [4] [9]

4) Internet of Things (IoT)

The network infrastructure and systems supporting emerging IoT technologies spans from traditional routers, switches, 3G/4G and WiFi to new emerging standards leveraging network function virtualization (NFV) and wireless technologies that consume less power while providing higher security [10]. SDN will be the means by which these network technologies are integrated into the IoT ecosystem, enabling a network to quickly adapt to the ever changing needs and inputs from the growing number of IoT devices.

5) Cloud computing

Cloud computing is the provision of computing as a service instead of providing the same as a product. Shared information, software, and resource are offered to computers as utilities over the internet [11]. It should be note that despite the rise of cloud computing, several issues are slowly making it increasingly difficult for most organizations to consider adopting it to their business processes. The issues include the cost of bandwidth, billing and service delivery, performance, what to migrate, virtualization, reliability and availability, energy consumption, elasticity and scalability, portability and interoperability, and privacy and security [12], [13]. SDN solutions such as OpenContrail resolve some of these issues for example scalability, portability, service delivery achievable by deploying the virtual networks necessary to connect the various segments of the cloud computing resources such as Storage and Computing on a most efficient manner.

D. OpenContrail

Juniper Networks OpenContrail [14] is an open, standards-based software solution that delivers network virtualization and service automation for federated cloud networks. It provides self-service provisioning, improves network troubleshooting and diagnostics, and enables service chaining for dynamic application environments across enterprise virtual private cloud (VPC), managed Infrastructure as a Service (IaaS), and Networks Functions Virtualization use cases. OpenContrail can be used with open cloud orchestration systems such as OpenStack or CloudStack. OpenContrail allows customers to build elastic architectures that leverage the benefits of ` computing — agility, self-service, efficiency, and flexibility — while providing an interoperable, scale-out control plane for network services within and across network domains.

Juniper OpenContrail [14] is a perfect example for not only an SDN platform but also future features extensibility and expansion. Both a free and a commercial option for OpenContrail are available with the main difference being the amount of support the user is entitled to. The two versions of OpenContrail are the same in terms of what it must offer and how it functions. By default, OpenContrail architecture features two primary drivers for networking, the Cloud Networking (CN), and Network Function Virtualization (NFV). Juniper Networks provides details regarding their commercial version of OpenContrail, which is identical to the free open-source version.

Use cases in CN range from private clouds for enterprise, virtual private clouds, to IaaS for cloud service providers. Juniper suggests in each of these use cases multiple tenants in a data center share the same physical resources (physical servers, physical storage, physical network) which results in each tenant having its own logical resources whether they are virtual machines, virtual storage, or virtual networks [15].

NFV involves the administration and configuration of network functions such as firewalls [16], Intrusion Detection, and Prevention Systems (IDPS), and Wide Area Network (WAN) optimization. All of which are taking place in virtual machines instead of on dedicated hardware appliances. Using virtualization to manage WANs instead of physical hardware decreases the time to market and the cost of optimization. OpenContrail can also be used to deploy Linux Containers and Dockers.

1) OpenContrail architecture

SDNs is the concept behind that of OpenContrail. There are two main components that allow the SDN to function as intended, the OpenContrail Controller, as well as the OpenContrail vRouter. These two components allow for the configuration of multiple types of nodes, providing high availability, and horizontal scaling. The OpenContrail controller is a logically centralized but physically distributed SDN controller responsible for providing the management, control, and analytics functions of the virtualized network [7]. This allows the OpenContrail controller to provide a centralized management platform for the SDN and controlling the vRouters. The OpenContrail vRouter is described as a forwarding plane of a distributed router that runs in the hypervisor of a virtualized server. vRouters extend the network from the physical routers and switches in a data center into a virtual overlay network hosted in the virtualized servers [15]. The OpenContrail vRouter is a replication for existing vSwitches but can perform tasks that provide both routing and higher layer services.

The OpenContrail controller consists of three main internal components. These components are referred to as nodes and are broken down into three different types: configuration nodes, control nodes, and analytics nodes.

Configuration nodes are responsible for translating the high-level data model into a lower level to interact with network elements. Control nodes are responsible for reproducing the low-level status to and from both network elements and peer systems in a consistent way. Meanwhile, the analytic nodes capture the real-time data from the network elements, abstract it, and provide data compatible with various applications. [16], [17]. Fig. 3 demonstrates the internal structure of the OpenContrail Controller. Notice that there are altogether six OpenContrail nodes, however, this paper focus on the three internal (control, configuration, analytics) nodes.

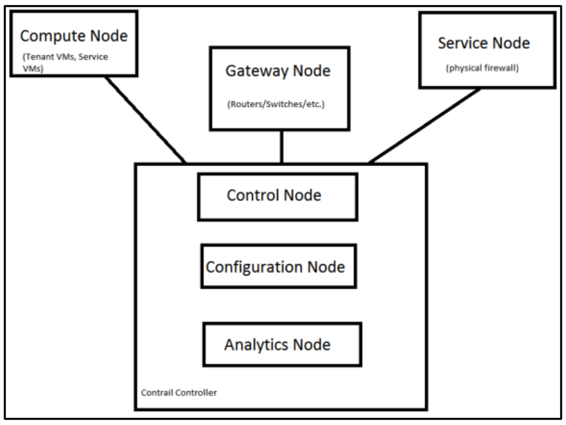

Fig. 3. Contrail nodes

In addition to the three main internal nodes, it is important to introduce the external nodes prior to moving to the OpenContrail vRouter. The three external nodes addressed are the compute node, gateway node, and the service node. The Compute nodes are virtualized servers, used to host VMs (Virtual Machines). Each compute node has a vRouter which is responsible for implementing a forwarding plane, as well as the distributed part of the control plane. Notice the gateway node located in Fig. 3, these nodes represent physical routers and, or switches that connect the virtual networks to physical networks. Examples would be to connect to another Data Center, or even non-virtualized servers. Service nodes represent the remaining physical network elements providing services such as DPI (Deep Packet Inspection), and IPS (Intrusion Detection Systems). This concludes the explanation of node types that make up the OpenContrail Controller, and define it, but the controller was only one of the two main components of OpenContrail.

The second component is the OpenContrail vRouter As mentioned earlier during node explanation that each Compute node has a vRouter responsible for implementing a forwarding plane, as well as the distributed part of the control plane. There are two fundamental blocks in a Compute node that require the implementation of a vRouter. The vRouter Agent, and the vRouter Forwarding Plane.

The vRouter Agent is developed as a user space process running inside the compute node. The agent acts a light-weight, yet logical, control plane that functions in the following ways. The agent exchanges control states such as routes**,** as well as receives low-level configuration routing instances, and forwarding policies from the various Control nodes using Extensible Messaging and Presence Protocol (XMPP) standards. The Agent also report analytics such as state logs, statics, and events to the Analytic nodes, and discover the existence and attributes of VMs with help from the OpenStack Nova. Nova in turn manages the lifecycle of compute instances in an OpenStack environment. Responsibilities include spawning, scheduling and decommissioning of machines on demand. The Agent is responsible for applying forwarding policy for the first packet of each new flow and installing a flow entry in the flow table of the forwarding plane. [15]

The vRouter Forwarding Plane runs as a kernel loadable module in the compute node, and is responsible for tasks such as encapsulating packets to the overlay network, decapsulating packets from the overlay network, and assigning packets to a routing instance. Packets are assigned to a routing instances by performing a lookup of the destination address in the FIB or (Forwarding Information Database). By default, packets that are received from the overlay network are assigned to a routing instance depending on either the MPLS label, VNI (Virtual Network Identifier) and virtual interfaces for local machines are also bound to routing instances. Optionally, the application of Flow Tables towards the forwarding policy is available. In doing so the forwarding policy matches packets against the flow table, applying the appropriate flow actions where necessary. Packets can be punted as well to the vRouter Agent which installs the rule in the flow table, and proxies DHCP, ARP, and DNS packets. Juniper states, the forwarding plane supports MPLS over GRE/UDP and VXLAN encapsulations in the overlay network. The forwarding plane supports layer-3 forwarding by doing a Longest Prefix Match (LPM) of the destination IP address, as well as layer-2 forwarding using the destination MAC address. The vRouter Forwarding Plane currently only supports IPv4. Support for IPv6 will be added in the future. [15]

III. Emulation and Results

The design of a new test bed will be used to test SDN Controllers is shown in Fig. 4. Internet connectivity is provided to remotely access the HP DL 360 Servers. The connectivity in the lab is provided by Juniper Switches.

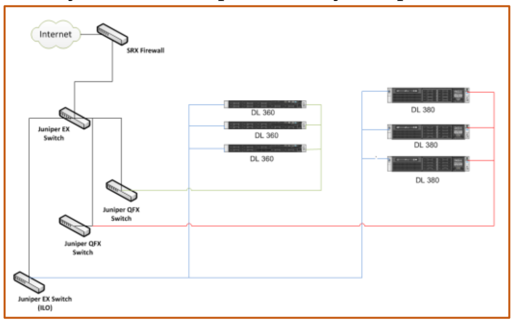

Fig. 4. Network design

This OpenStack and Juniper OpenContrail SDN solution design is intended to allow users to quickly deploy OpenStack/OpenContrail/KVM/ESXi 6.0/EXSi 5.0 software stack which uses overlay networking (MPLSoGRE or VXLAN) for transport between virtual machines [18].

The choice of Juniper Switches and HP Dell Servers is due to their wide availability as well as the fact that OpenContrail is being developed by Juniper Networks.

In this paper we have implemented the OpenContrail using automated scripts and have implemented a usecase where we have demonstrated that many ESXI Servers can be managed using a single instance of OpenStack/OpenContrail environment**.** This highlights the benefits of open source, industry supported SDN controllers. We have used a simplified approach to implement a complex controller. The environment chosen to implement this solution was memory and processor intensive.

This section will outline the testbed environment hardware, software, and operating systems configuration. In general the server requirements for the actual server running contrail are very high especially the memory [14].

Each Contrail running server must have a minimum of:

- 64 GB memory
- 300 GB hard drive
- 4 CPU cores
- At least one Ethernet port

For production environment, each Contrail running server must have a minimum of:

- 256 GB memory
- 500 GB hard drive
- 16 CPU cores

Install Script has been modified to help quickly deploy OpenContrail solutions [19] according to the use cases discussed below.

Python scripts are used to populate the multi-server deployment scenario with networks, network policies, VMs and service chains that will allow the use cases to be demonstrated (the multi-hypervisor images will be pre-configured) [20], [21].

This section describes the system hardware information used for deployment of Juniper OpenContrail and OpenStack.

- Three HP DL 360 Gen 8 for POD C 2x Processor Intel Xeon E502600 v3 Hyper threading available 256GB RAM
- Three HP DL 380 Gen 7 for POD C 2x Processor Intel Xeon E502600 v2 128GB RAM
- Two Juniper QFX 5100 switches
- One Juniper EX 3300 switch
- One Juniper EX 2200 switch
- One Juniper SRX

The following table contains OpenStack and Juniper OpenContrail SDN solution deployment project software and their versions.

TABLE III: SOFTWARE ON HP SERVERS

Software	Version	Physical Hardware
Ubuntu	14.04.2 LTS with kernel version 3.12.02	3 HP DL 360 Gen 8
VMWare EXSi	version 6.0	2 HP DL 380 Gen 7
VMWare EXSi	version 5.0	2 HP DL 380 Gen 7
OpenContrail with OpenStack	OpenContrail version 2.2 and OpenStack Ice house	3 HP DL 360 Gen 8

There are many scenarios where OpenContrail can be useful to enhance the performance of the network. Several ESXI Servers can be managed using a single instance of OpenStack/OpenContrail environment [22]. It is also possible to use OpenContrail to reduce the CapEx and OpEx cost by deploying services in Cloud where the network creation becomes much simplified. The overlay networks support high scaling, together with traffic segmentation, and can be configured to implement service chaining. With the help of overlay networking Virtual Machines on different datacenters or bare metal machines can communicate with each other. Through OpenContrail service chaining feature and Micro Segmentation and employing simple OpenStack GUI network functions such as firewalls and load balancer can be introduced between any two networks.

The code used for the OpenContrail emulation is listed in the appendix.

IV. CONCLUSION

There are many SDN controllers developed by academia and industry such as the Open Source SDN Controllers. In this paper we present the drivers for SDN networking, highlight several SDN use cases and finally implement the Juniper OpenContrail Controller Solution using our newly designed testbed.

OpenContrail creates a standards-based virtualized L2/L3 network fabric over any IP network infrastructure and introduces the vRouter for packet forwarding and distributed routing. Contrail can Segregate network traffic by tenants. It can be used to automate orchestration and deployment of complex, multi-server application topologies. It Supports configuration, provisioning and chaining of virtualized network functions. It facilitates creation of abstract network and security policies and automate distributed enforcement to maintain security, performance, resilience, and meet SLAs however it should be noted that these are some of the same feature implemented by other SDN controllers.

In future we plan to compare the SDN controllers based on their performance and features using the same testbed.

APPENDIX

This appendix includes the code that is used to install Contrail on Ubuntu. It also includes the commands that are used to configure the Ubuntu.

Following configurations were used to configure Ubuntu machines. Restart all three Ubuntu servers. Wait for them to come back up.

To setup network interface card

```
sudo vim /etc/network/interfaces
```

And add the following lines

```
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
address $IP_Address
netmask 255.255.255.0
gateway $Gateway_server
dns-nameservers $DNS_Server
```

Save the file using ":wq" key combination

Restart the network interface

```
sudo ifdown eth0
sudo ifup eth0
```

Ensure all machine in the stack can ping each other

Edit the following file

```
Vim /etc/hosts
```

And add the following lines:

```
$IP_Address          $HostName # machine 1
$IP_Address          $HostName # machine 2
$IP_Address          $HostName # machine 3
```

Turn off firewall

```
Sudo iptables –F
Sudo services iptables off
sudo chkconfig iptables off
```

OpenStack/OpenContrail deployment has been done by uploading the OpenStack/OpenContrail package file [15] [17] to all servers in the stack in

```
/home/wajid/
dpkg –i /home/wajid/OpenContrail-install-packages-
1.xx.xxxx~OpenStack_version_all.deb
cd /opt/OpenContrail /OpenContrail _packages
sudo ./setup.sh
```

Open the file

```
Vim /opt/OpenContrail /utils/fabfile/testbeds/testbed.py
```

Add the following files

```
from fabric.api import env
host1='wajid@10.2.0.10'
host3='wajid@10.2.0.11'
host4='wajid@10.2.0.12'
host2='root@10.2.0.13'
host5='root@10.2.0.14'
host6='root@10.2.0.15'
host7='root@10.2.0.16'
ext_routers = [('mx1', '10.2.0.7')]
router_asn = 64512
host_build = 'wajid@10.2.0.10'
env.roledefs = {
   'all': [host1,host2,host3,host4,host5,host6,host7],
   'cfgm': [host1],
   'OpenStack': [host1],
   'control': [host1],
   'compute': [host1,host2,host3,host4,host5,host6,host7],
   'collector': [host1],
   'webui': [host1],
   'database': [host1],
   'build': [host_build],
   'storage-master': [host1],
   'storage-compute': [host1],
}
env.OpenStack_admin_password = 'wajid123'
env.hostnames = {
   'all': ['WH','OpenContrail VM-24-26']
}
env.passwords = {
   host1: 'rentvm123', host3: 'rentvm123', host4: 'rentvm123',
   host2: 'c0ntrail123', host5: 'c0ntrail123', host6: 'c0ntrail123', host7:
'c0ntrail123',
   host_build: 'rentvm123',
}
env.ostypes = {
host1:'ubuntu',
host2:'ubuntu',
host3:'ubuntu',
host4:'ubuntu',
host5:'ubuntu',
host6:'ubuntu',
host7:'ubuntu',
}
minimum_diskGB = 10
esxi_hosts = {
      'esxi': {
           'ip': '10.2.0.4',
           'username': 'root',
           'password': 'RentVM123',
'datastore': "/vmfs/volumes/55c2f1a5-8dc410dc-943d-e4115bbd72b2/",
                'cluster': "60_cluster",
                'OpenContrail _vm': {
 'name':'OpenContrail VM-24-26',
              'mac': "00:50:56:05:ba:ba",
               'host': host2,
                'vmdk_download_path':"10.2.0.10:10000/ESXi-v5.5-
OpenContrail -host-Ubuntu-trusty-14.04.vmdk",
          }
      'esxi': {
           'ip': '10.2.0.3',
           'username': 'root',
           'password': 'RentVM123',
           'datastore':            "/vmfs/volumes/55c2f1a5-8dc410dc-943d-
e4115bbd72b2/",
           'cluster': "60_cluster",
            'OpenContrail _vm': {
                        'name':'OpenContrail VM-24-26',
                'mac': "00:50:56:05:ba:ba",
                'host': host5,
                'vmdk_download_path':"10.2.0.10:10000/ESXi-
v5.5-OpenContrail -host-Ubuntu-trusty-14.04.vmdk",
            }
      'esxi': {
            'ip': '10.2.0.2',
            'username': 'root',
            'password': 'RentVM123',

'datastore':"/vmfs/volumes/55c2f1a5-8dc410dc-943d-
e4115bbd72b2/",
             'cluster': "60_cluster",
             'OpenContrail _vm': {
                        'name':'OpenContrail VM-24-26',
                'mac': "00:50:56:05:ba:ba",
                'host': host6,
                'vmdk_download_path':"10.2.0.10:10000/ESXi-
v5.5-OpenContrail -host-Ubuntu-trusty-14.04.vmdk",
            }
       'esxi': {
                'ip': '10.2.0.1',
               'username': 'root',
 'password': 'RentVM123',
'datastore':              "/vmfs/volumes/55c2f1a5-8dc410dc-943d-
e4115bbd72b2/",
             'cluster': "60_cluster",
             'OpenContrail _vm': {
                        'name':'OpenContrail VM-24-26',
                'mac': "00:50:56:05:ba:ba",
                'host': host7,
                'vmdk_download_path':"10.2.0.10:10000/ESXi-
v5.5-OpenContrail -host-Ubuntu-trusty-14.04.vmdk",
        }

      }
}
```

Run the following commands

Cd /opt/OpenContrail /utils/
Fab prov_esxi
Manually log into all the ESXI hypervisor machines. Log into the consoles of the newly deployed VMs and ensure there interfaces are up. If not. Manually assisng them IP addresses.
Run the following commands
/opt/OpenContrail/utils/fab install_pkg_all:/tmp/OpenContrail-install-packages-1.xx-xxx~OpenStack_version_all.deb
Cd /opt/OpenContrail /utils/
Fab install_OpenContrail
The system will restart after this command. Wait for the system to come back.
Cd /opt/OpenContrail /utils/
Fab setup_all
OpenContrail is now running.

References

[1] B. A. A. Nunes, M. Mendonca, X. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617-1634, 2014.

[2] R. Jain and S. Paul, "Network virtualization and software defined networking for cloud computing: A survey," *IEEE Communications Magazine*, vol. 51, no. 11, pp. 24-31, 2013.

[3] Open Networking Foundation (2013). Software-Defined Networking: The New Norm for Networks. [Online]. Available: https://www.opennetworking.org

[4] D. L. Beaty, "Is software-defined data center next reality?" *Ashrae Journal*, vol. 56, no. 2, pp. 58-60, 2014.

[5] D. Kreutz, F. M. V. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. (2014). Software-Defined Networking: A Comprehensive Survey. *CoRR*. [Online]. pp. 1-61. Available: http://arxiv.org/abs/1406.0440v3

[6] F. Hu, Q. Hao, and K. Bao, "A survey on software-defined network (SDN) and OpenFlow: From concept to implementation," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, 2014.

[7] K. Hyojoon and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114-119, 2013.

[8] C. Janz, L. Ong, K. Sethuraman, and V. Shukla, "Emerging transport SDN architecture and use cases. *IEEE Communications Magazine*, vol. 54, no. 10, pp. 116-121, 2016.

[9] ONF SDN Evolution. (September 8, 2016). [Online]. Available: www.opennetworking.org

[10] A. L. V. Caraguary, A. B. Peral, L. I. B. Lopez, and L. J. G. Villalba, "SDN: Evolution and opportunities in the development IoT applications," *International Journal of Distributed Sensor Networks*, pp. 1-10, 2014.

[11] K. M. Ahmad, "A survey of security issues for cloud computing," *Journal of Network and Computer Applications*, pp. 11-29, 2016.

[12] D. B. Rawat and S. R. Reddy, "Software defined networking architecture, security and energy efficiency: A survey," *IEEE Communication Surveys & Tutorials*, pp. 1-22. 2016.

[13] V. K. Pachghare, *Cloud Computing*, Delhi: PHI Learning Pvt. Ltd, 2015.

[14] Release Notes for Contrail Release 2.20. [Online]. Available:https://www.juniper.net/techpubs/en_US/contrail2.2/information-products/topic-collections/release-notes/index.html

[15] A. Singla and B. Rijsman, (2013). Understanding OpenContrail Architecture. 1st ed. [Online]. 1. Available: http://www.juniper.net/dayone

[16] Installing the Contrail Packages, Part One (Cen []tOS or Ubuntu). [Online]. Available: http://www.juniper.net/techpubs/en_US/OpenContrail 1.0/topics/task/installation/install-steps-packages-vnc.html

[17] Open Contrail Installation and Provisioning Roles. [Online]. Available: https://techwiki.juniper.net/Documentation/OpenContrail/OpenContrail_Controller_Getting_Started_Guide/30_Installation/20_Installation_and_Provisioning_Roles

[18] Integrating VMware ESXi with OpenStack & OpenContrail. [Online]. Available: http://www.tcpcloud.eu/en/blog/2015/02/08/intergrating-vmware-esxi-openstack-opencontrail/

[19] Install scripts for OpenContrail. [Online]. Available: https://github.com/Juniper/contrail-installer

[20] Release Notes for OpenContrail Release_2.20. [Online]. Available:https://techwiki.juniper.net/Documentation/OpenContrail/OpenContrail_Release_Notes/090_Release_Notes_for_OpenContrail _Release_2.20

[21] Project Name: OpenContrail Quickstart. [Online]. Available: https://wiki.opnfv.org/display/PROJ/Ocq

[22] OpenContrail Blog; Integrating VMware ESXi with OpenStack & OpenContrail, February 12, 2015 Jakub Pavlik. [Online]. Available: http://www.opencontrail.org/integrating-vmware-esxi-with-openstack-opencontrail/

Wajid Hassan is a PhD Fellow in the College of Technology at Indiana State University. He has a Bachelor of Engineering in Electronics and M.S. in Electrical and Computer Engineering from Wichita State University. He has worked for AT&T for 8 years as a Solution Architect and currently consults as Solution Architect for an Ethernet startup and at US Department of Architecture for Network Engineering. He has taught undergraduate and graduate classes and most recently at Indiana State University. His current research interests include IPv6, Network Function Virtualization, Software Defined Networking, System and Network Virtualization, Cloud

Computing and LTE networks. He is a Cisco Certified Internet Expert [CCIE] in Service Provider and Datacenter with active License # 47485. He is also certified VMWare Certified Professional.

Tamer Omar is an Assistant professor at the Electrical and Computer Engineering Department in California State Polytechnic University- Pomona. Dr. Omar earned his Ph.D. from the Electrical Engineering department at Iowa State University, USA. Dr. Omar research interests include broadband wireless networks, cybersecurity in IOT, and big data systems. Dr. Omar has over 15 years of experience in both academia and industry serving in different roles at engineering and technology systems domains.