CUTSE2015 IOP Publishing
IOP Conf. Series: Materials Science and Engineering **121** (2016) 012003 doi:10.1088/1757-899X/121/1/012003

Software Defined Networking challenges and future direction: A case study of implementing SDN features on OpenStack private cloud

Veeramani Shamugam[1*], I Murray[2], J A Leong[3] and Amandeep S Sidhu[3]
[1]Department of Electrical and Computer Engineering, Faculty of Engineering and Sciences, Curtin University Sarawak Campus, Miri, Malaysia.
[2]Department of Electrical and Computer Engineering, Faculty of Science and Engineering, Curtin University, Bentley, Perth, Western Australia, 6102.
[3]Curtin Sarawak Research Institute, Curtin University Sarawak Campus, Miri, Malaysia.
*Corresponding author: s.veeramani@curtin.edu.my

Abstract— Cloud computing provides services on demand instantly, such as access to network infrastructure consisting of computing hardware, operating systems, network storage, database and applications. Network usage and demands are growing at a very ***fast*** rate and to meet the current requirements, there is a need for automatic infrastructure scaling. Traditional networks are difficult to automate because of the distributed nature of their decision making process for switching or routing which are collocated on the same device. Managing complex environments using traditional networks is time-consuming and expensive, especially in the case of generating virtual machines, migration and network configuration. To mitigate the challenges, network operations require efficient, flexible, agile and scalable software defined networks (SDN). This paper discuss various issues in SDN and suggests how to mitigate the network management related issues. A private cloud prototype test bed was setup to implement the SDN on the OpenStack platform to test and evaluate the various network performances provided by the various configurations.

1. Introduction

A cloud is defined as a place, over network infrastructure, where information technology (IT) and computing resources such as computer hardware, operating systems, networks, storage, databases and even entire applications are available instantly, on demand [2-3]. The new cloud architecture provides network infrastructure as a service through a group of layers, which are also provided as services. The cloud framework consists of three main layers: 1. Software as a Service (SaaS) - allows multiple end users to access applications which are running on cloud infrastructure through a web browser. The end users do not manage or control the software in the cloud. 2. Platform-as-a-Service (PaaS) - allows the users to install applications developed through programming languages or packages on the cloud but they do not control or manage the cloud. 3. Infrastructure as a service (IaaS) - provides users with processors, storage, networks and other computing resources as a service and the user do not have to control or manage the infrastructure but they do have control over the operating systems (OS), applications and programming frameworks [1].

2. Software Defined Networking (SDN)

2.1 Background

Software defined networking, referred to as a revolutionary new idea in computer networking, is the new approach that promises to dramatically simplify network-control and the management plane. It is achieved through innovative network programmability. In the cloud environment, monitoring network users and the resources they access, managing the infrastructure, as well as supporting demand becomes complicated [4-5]. Networking devices have control and data movement functions on the same device.

Traditionally, the network admin has to control from the network management plane in order to configure each network's devices separately. This static setup of current network devices does not permit control-plane configuration. The Software-defined networking (SDN) approach allows for open, user-controlled, management of the forwarding hardware in network elements [6]. SDN implements centralized control-plane intelligence while keeping the data plane separate, which in turn enables the administrator to configure network hardware directly from the controller. This approach of centralizing control of the entire network makes the network highly flexible [7-8]. To meet growing demand and cater for network instability, Manar Jammal proposed Software Defined Networking (SDN) combined with network function virtualization (NFV) [6]. SDN isolates the network control logic from the hardware which in turn enables the network administrator to have more control over network functions and a global view of networks.

2.2 Software Defined Networking Architecture

Traditional networks are difficult to automate because of the distributed nature of their decision making process for switching or routing which are collocated on the same device. Managing complex environments using traditional networks is time-consuming and expensive, especially in the case of generating virtual machines, migration and network configuration. The demand for service and network usage is growing rapidly. The current nature of data such as video traffic, big data centers and mobility of network users pose significant challenges to network operators who are facing spectrum congestion. Also, data-center operators are facing tremendous growth in the number of servers and virtual machines and increasing server-to-server communication traffic. To overcome these challenges, network operators require efficient, flexible, agile and scalable software defined networks (SDN). The key role of a SDN is to aggregate and centralize the control plane which is a promising solution for network management and control problems as shown in figure 1.

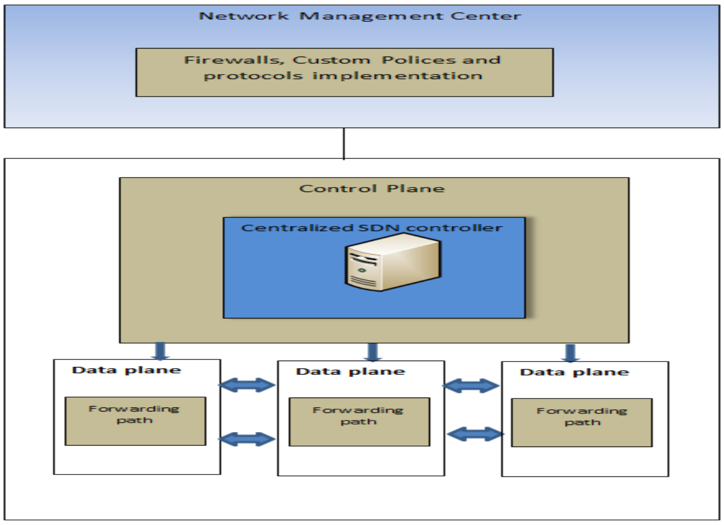


Figure 1: SDN architecture.

An SDN consists of three sections:

1. The Network Management Centre is responsible for implementing various functions such as firewalls, custom policies and protocol implementations.
2. The control plane function centralizes the control plane intelligences (switching and routing) to the controller. This allows administrators to configure the network hardware directly from the controller. This approach makes the network highly flexible.
3. The Data plane represents packet forwarding hardware in the SDN architecture

3. Issues in SDN

Although SDN is a favorable solution for IT and cloud providers and enterprises, SDN faces some challenges that hinders its performance and implementation. The list of SDN challenges consists of: Controller placement, Scalability, Performance, Security, Interoperability and Reliability [6]. SDN controllers must be wisely configured and the SDN's network topology authenticated to prevent manual errors and increase network availability. In a traditional network when one network or many network devices fail, network data flow is routed through another or nearby nodes or devices to continue data flow continuity.

However in the centralized controller architecture of SDN, a single controller is in charge of all the networks, and if there is a failure in the central controller, the whole network collapses since there is no alternate controller. To address this issue the cloud organization needs to focus on how to efficiently utilize main controller functions that can increase network reliability. The SDN controller should have the ability to support multiple-path solutions or fast traffic rerouting to active links if there is a path/link failure. If the main controller fails, the newer architectures support an alternate controller which can handle traffic flow. Controllers also support technologies such as Virtual Router Redundancy Protocol (VRRP) and Multi-Chassis link aggregation groups to increase network availability.

The second SDN challenge is the scalability because in this approach, the data and control planes are decoupled but they can progress independently as long as the API connects them. This centralized view of the network accelerates changes in the control plane. The decoupling process has its own drawbacks, such as having a standard API for both planes and the SDN controller become the bottle neck in a situation where the network scales the number of switches and number of nodes up.

Performance of the network is another important area to look into. SDN is a flow based technique, so the performance is measured on two metrics: flow-setup time and how many flows per second the controller can switch. Flow-setup works in two modes, which are proactive and reactive. These two modes have their respective flow initiation and flow limitation overheads. To overcome the performance limitation, focus is needed on factors affecting flow-setup time and I/O performance of the controller. There are means and ways to increase the performance by considering well-known optimization techniques, such as input/output batching and using the Maestro approach which uses techniques such as input batching threshold (IBT) and pending raw packet threshold (PRT).

3.1 Network Management in SDN

Software Defined Networking and virtualization are a combined solution to overcome the challenges that is faced in traditional, legacy, networks. SDN operates on an accumulated and centralized control plane that provides a likely solution for network management and control problems. The main idea here is that SDN separates the data plane and control plane in addition to providing the flexibility of programmability to the centralized control plane. The ultimate goal of SDN is to provide open, user-controlled, management of the forwarding hardware and network elements. It functions on the idea of a centralized control-plane intelligence by keeping the data plane separate. Thus, the network hardware devices keep their switching fabric (data plane), but hand over their intelligence (switching and routing functionalities) to the controller [6]. This enables the administrator to configure the network hardware directly from the controller. This centralized control of the entire network makes the network highly flexible.

3.2 Prototype test bed to implement SDN features.

In this research project, we implemented the various features of a SDN such as a controller node responsible for collecting routing information and making routing decisions centrally. The SDN data plane will be the hypervisor (compute node) in the prototype model and the functionality of Software Defined Networking in a private cloud built using OpenStack.

4. OpenStack Frameworks

4.1 OpenStack

OpenStack is an open source cloud computing platform that can support all types of cloud environments. OpenStack offers the same open source cloud solution as you would find on Eucalyptus, and also manages to outdo Eucalyptus when it comes to support and troubleshooting. OpenStack also offers PaaS, IaaS and even NaaS in its newer releases. Besides OpenStack offering a cloud computing software solution, it also contains many additional features and a number of tools [10]. These include: Scaling in size depending on demand and user needs; and processing big data and heavy workloads with tools like Hadoop High-performance Computing (HPC) environments. OpenStack has deployed its platform with PaaS and IaaS concepts in mind and supports a wide variety of hardware including the ARM processor architecture. It transcends both services and manages to tie in neatly with the overall package offerings from other clouds, such as Windows Azure [5]. Because of its universal compatibility with most high level programming languages, it can be considered as an IaaS service model. The cloud frame work shown in figure 2 was used to execute various applications and to test our research objective.

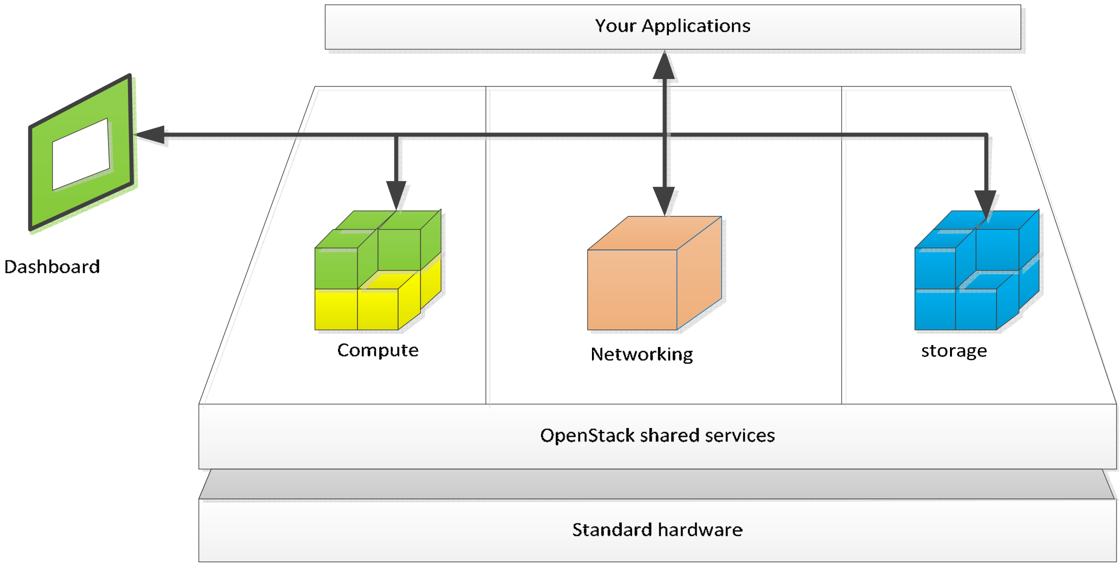


Figure 2: Open source cloud computing framework.

The open source cloud computing frame work consists of three important parts as discussed below:
Compute Node: Offers on-demand computing resources by provisioning and managing large networks of virtual machines. Compute resources are accessible via APIs for developers building cloud applications and via web interfaces for end users
Storage Node: Offers object storage and block storage, with many deployment options depending on use case.

Network Node: Offers pluggable, scalable and an API-driven system for managing networks and IP addresses. It can also be used to increase the value of existing datacenter assets. Lastly, it also ensures the network will not bottleneck a cloud deployment and gives end users real self-service, even over their network configurations [10]

4.2 Network Architecture

In this research, a basic three node architecture was implemented as shown in figure 3. This architecture provides high end computing, networking and storage facilities.

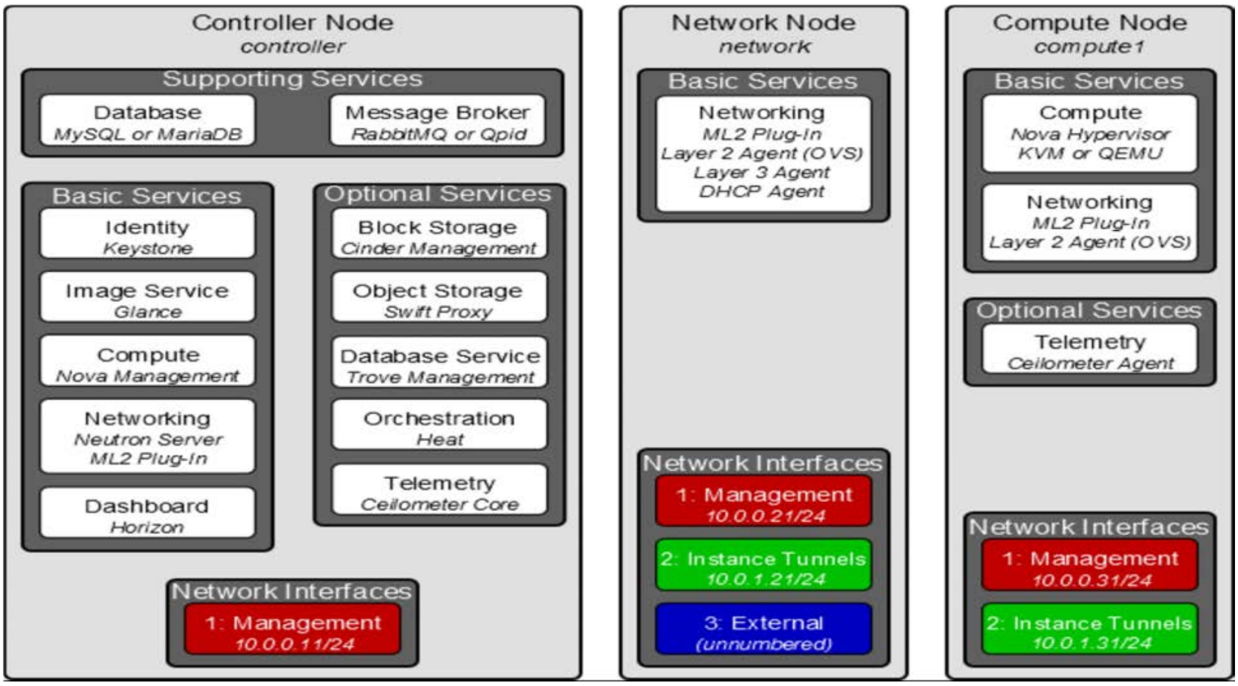


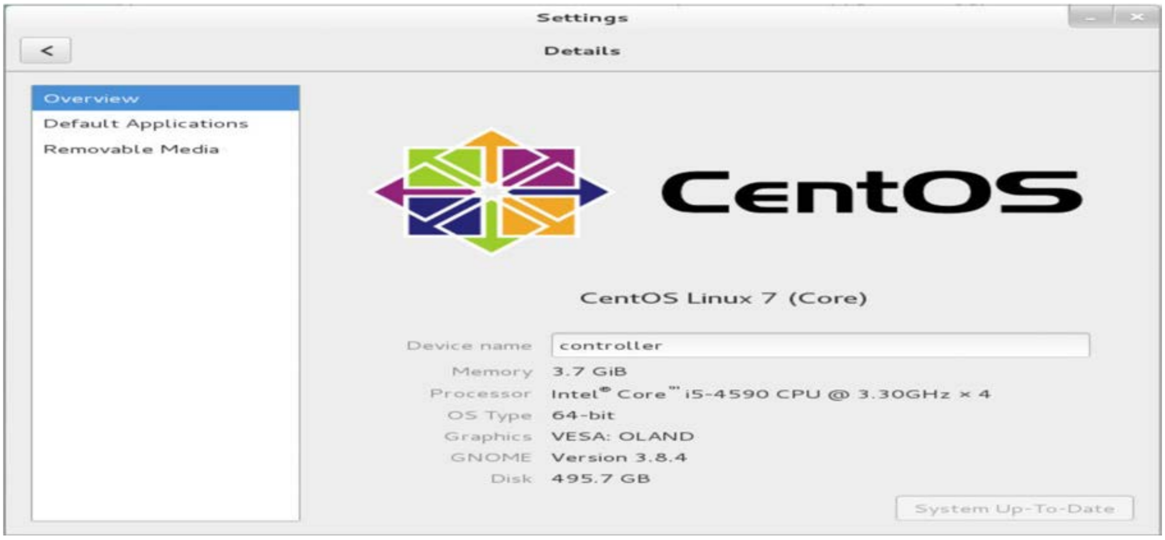
Figure 3: Architecture of three nodes in an open source cloud.

In figure 3 the first block section is the Controller node, the second is the Network node, and the third is for a Compute node.

The controller node is responsible for the following basic services: identity, image, management portion of Compute and Networking, Networking plug-in and the dashboard. It also runs additional supporting features such as a message broker, database in MySQL and Network time Protocol (NTP) [10].

The Network node executes the following services: networking ML2 plugin, layer 2 and layer 3 agents that provide and operate tenant networks. The main role of layer 3 is routing, network address translation and DHCP services. Providing virtual networks and tunnels is taken care of by layer 2.

Compute nodes execute the hypervisor part of compute services that functions as tenant virtual machines or instances. It also runs networking plug-ins and other optional services.



CentOS Linux 7 (Core)

Device name: controller
Memory: 3.7 GiB
Processor: Intel® Core™ i5-4590 CPU @ 3.30GHz × 4
OS Type: 64-bit
Graphics: VESA: OLAND
GNOME: Version 3.8.4
Disk: 495.7 GB

Figure 4: Installation of CentOS.

Figure 4 shows the installation CentOS. OpenStack is subsequently installed on top of CentOS, where it runs a set of software tools, building and managing cloud computing platforms for public and private clouds. The aim is to enable organizations, regardless of size, to create and offer cloud computing services running on standardized hardware. The OpenStack community has collaboratively identified nine key

Components that are part of the “core” of OpenStack, which are distributed as a part of any OpenStack system and maintained by the OpenStack community. By provisioning and managing large networks of virtual machines, these components enable enterprises and service providers to offer on-demand computing resources

4.3 Basic services: Identity service

Keystone provides an authentication and authorization service for other OpenStack services. Keystone integrates with LDAP to provide a central list of all users of the OpenStack cloud and allows administrators to set policies that control which resources various users have access to. It provides multiple means of access, meaning developers can easily map their existing user access methods against Keystone.

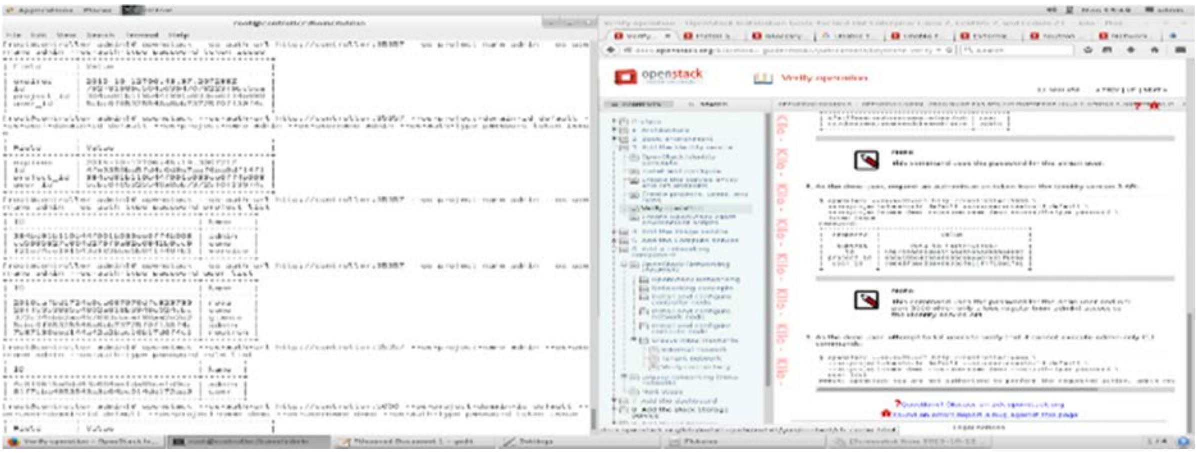


Figure 5: Implementation of the Dashboard service.

4.4 Basic services: Dashboard service

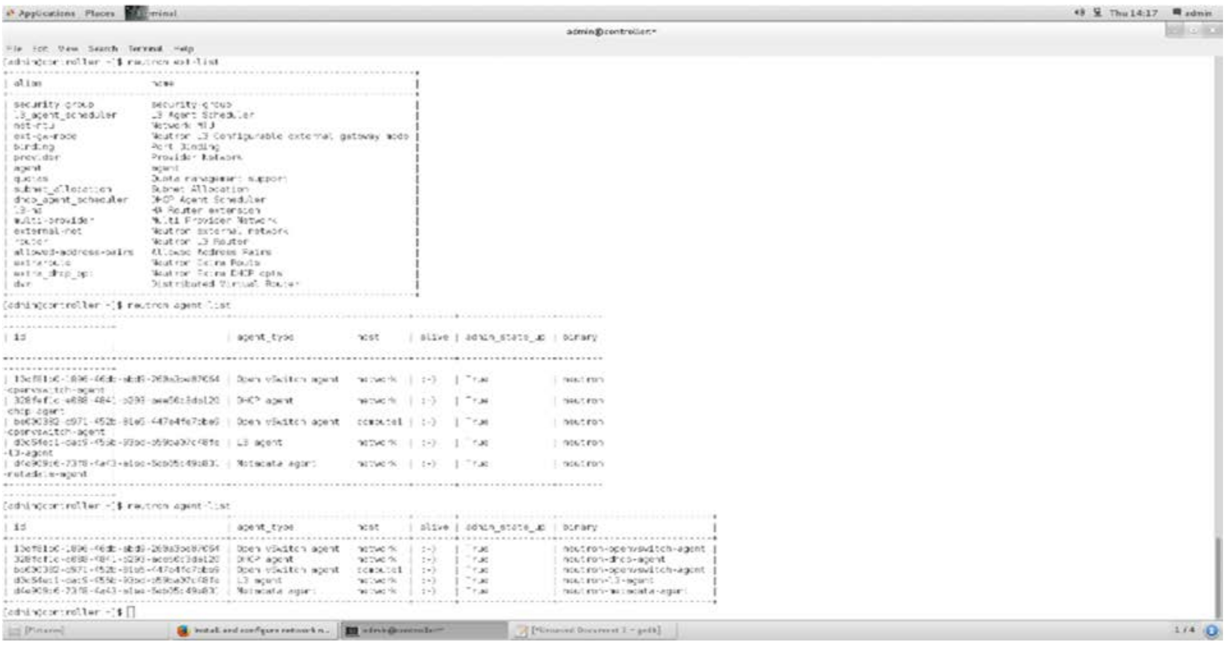
Horizon provides administrators and users a graphical interface to access, provision, and automate cloud-based resources. Developers can access all of the components of OpenStack individually through an application programming interface (API), but the dashboard provides system administrators a look at what is going on in the cloud, and to manage it as needed. It's the primary way for accessing resources if API calls are not used.

4.5 Basic services: Image service

Glance stores and retrieves virtual machine disk images. It allows these images to be used as templates when deploying new virtual machine instances. One of the main benefits to a cloud platform is the ability to spin up virtual machines when users request them. By creating templates for virtual machines, Glance helps to achieve this benefit. Also, it can copy or snapshot a virtual machine image and later on allow it to be recreated. Glance can also be used to back up existing images to save them. Glance integrates with Cinder to store the images. OpenStack Compute makes use of these stored images during instance provisioning.

4.6 Basic services: Networking

Neutron (formerly Quantum) provides the networking capability for OpenStack. It helps to ensure that each of the components of an OpenStack deployment can communicate with one another quickly and efficiently. Neutron manages the networking associated with OpenStack clouds. It is an API-driven system that allows administrators or users to customize network settings. It supports the Open Flow software defined networking protocol and plugins are available for services such as intrusion detection, load balancing and firewalls.



```
[admin@controller ~]$ neutron ext-list
+-----------------------+-----------------------------------------------+
| alias                   name                                          |
+-----------------------+-----------------------------------------------+
| security-group          security-group                                |
| l3_agent_scheduler      L3 Agent Scheduler                            |
| net-mtu                 Network MTU                                   |
| ext-gw-mode             Neutron L3 Configurable external gateway mode |
| binding                 Port Binding                                  |
| provider                Provider Network                              |
| agent                   agent                                         |
| quotas                  Quota management support                      |
| subnet_allocation       Subnet Allocation                             |
| dhcp_agent_scheduler    DHCP Agent Scheduler                          |
| l3-ha                   HA Router extension                           |
| multi-provider          Multi Provider Network                        |
| external-net            Neutron external network                      |
| router                  Neutron L3 Router                             |
| allowed-address-pairs   Allowed Address Pairs                         |
| extraroute              Neutron Extra Route                           |
| extra_dhcp_opt          Neutron Extra DHCP opts                       |
| dvr                     Distributed Virtual Router                    |
+-----------------------+-----------------------------------------------+
[admin@controller ~]$ neutron agent-list
+--------------------------------------+--------------------+----------+-------+----------------+---------
-------------------+
| id                                   | agent_type         | host     | alive | admin_state_up | binary

+--------------------------------------+--------------------+----------+-------+----------------+---------
-------------------+
| [illegible]                          | Open vSwitch agent | network  | :-)   | True           | neutron
-openvswitch-agent
| [illegible]                          | DHCP agent         | network  | :-)   | True           | neutron
-dhcp-agent
| [illegible]                          | Open vSwitch agent | compute1 | :-)   | True           | neutron
-openvswitch-agent
| [illegible]                          | L3 agent           | network  | :-)   | True           | neutron
-l3-agent
| [illegible]                          | Metadata agent     | network  | :-)   | True           | neutron
-metadata-agent
+--------------------------------------+--------------------+----------+-------+----------------+---------
-------------------+
[admin@controller ~]$ neutron agent-list
+--------------------------------------+--------------------+----------+-------+----------------+---------------------------+
| id                                   | agent_type         | host     | alive | admin_state_up | binary                    |
+--------------------------------------+--------------------+----------+-------+----------------+---------------------------+
| [illegible]                          | Open vSwitch agent | network  | :-)   | True           | neutron-openvswitch-agent |
| [illegible]                          | DHCP agent         | network  | :-)   | True           | neutron-dhcp-agent        |
| [illegible]                          | Open vSwitch agent | compute1 | :-)   | True           | neutron-openvswitch-agent |
| [illegible]                          | L3 agent           | network  | :-)   | True           | neutron-l3-agent          |
| [illegible]                          | Metadata agent     | network  | :-)   | True           | neutron-metadata-agent    |
+--------------------------------------+--------------------+----------+-------+----------------+---------------------------+
[admin@controller ~]$ 
```

Figure 6: Implementation of the Networking service (Neutron).

4.7 Basic services: Compute service

Nova is designed to manage and automate the provisioning of compute resources. This is the core of the virtual machine management software, but it is not a hypervisor. Instead, Nova supports virtualization technologies including KVM, Xen, ESX and Hyper-V, and it can run on bare-metal and high performance computing configurations too. Compute resources are available via APIs for developers and through web interfaces for administrators and users. The compute architecture is designed to scale horizontally on standard hardware.

4.8 Basic services: Database service

Trove is the database as a service open source project for OpenStack. Trove is to provide scalable and reliable cloud database as a service, provisioning functionality for relational and non-relational database engines, and to improve its full-featured and extensible open source framework. Trove is designed to run entirely on OpenStack. Cloud users and database administrators can provision and manage multiple database instances as needed. Initially, the service will focus on providing resource isolation at high performance while automating complex administrative tasks including deployment, configuration, patching, backups, restores and monitoring.

5. Conclusion

In this paper, we present an implementation of the various SDN features such as a controller node responsible for collecting routing information and making routing decisions centrally on OpenStack. The SDN data plane will be the hypervisor (compute node) in the prototype model and functionality of the Software Defined Network is deployed in a private cloud on OpenStack.

6. References

[1] Furht B and Escalante A 2010 Handbook of cloud computing (New York: Springer)
[2] Wan Z 2010 *Cloud Computing infrastructure for latency sensitive applications* (IEEE) In Communication Technology (ICCT), 2010 12th IEEE Int. Conf. p 1399-1402
[3] Gathering Clouds of XaaS! http://ibm.com/developer
[4] Costa P, Migliavacca M, Pietzuch P and Wolf A L 2012 *NaaS: Network-as-a-Service in the Cloud* Presented as part of the 2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services 2012.
[5] Dudkowski D, Tauhid B, Nunzi G and Brunner M 2011 *A prototype for in-network management in NaaS-enabled networks* (IEEE) In Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium p 81-88
[6] Jammal M, Singh T, Shami A, Asal R and Li Y 2014 *Software defined networking: State of the art and research challenges* Computer Networks vol 72 p 74-98.
[7] Big Switch Networks 2012 *The open SDN Architecture* https://www.bigswitch.com/sites/default/files/sdn_overview.pdf
[8] Shin M K, Nam K H and Kim H J 2012 *Software-defined networking (SDN): A reference architecture and open APIs* (IEEE) In ICT Convergence (ICTC), 2012 Int. Conf. p 360-361
[9] Khasnabish B, Huang D, Bai X, Bellavista P, Martinez G and Antonopoulos N 2012 *Cloud Computing, Networking, and Services* Journal of Network and Systems Management vol 1 p 1-5.
[10] OpenStack 2013 *OpenStack Installation Guide for Ubuntu 14.04 (LTS)* http://docs.openstack.org/juno/install-guide/install/apt/content/