Volume 57, Number 1, 2016

ACTA TECHNICA NAPOCENSIS
Electronics and Telecommunications

MIGRATION OF AN SDN-BASED TESTBED INTO A PRIVATE CLOUD: AN OPENSTACK APPROACH

Eduard LUCHIAN Iustin A. IVANCIU Andrei B. RUS Gabriel LAZAR Virgil DOBROTA
Technical University of Cluj-Napoca, Communications Department, 400027 Cluj-Napoca, Romania,
Phone: +40-264-401226, Fax: +40-264-401916,
Emails: {Eduard.Luchian, Iustin.Ivanciu, Bogdan.Rus, Gabriel.Lazar, Virgil.Dobrota}@com.utcluj.ro

Abstract: **The paper discusses an example for the migration of a Software-Defined Networking-based testbed into a private cloud. The software solution is entirely open-source and starts from an existing 4-node real implementation in OpenFlow. The OpenStack approach employs the same topology, while running OpenvSwitch on each virtual node. The Beacon controller which governs the SDN implementation is also virtualized. Preliminary results show that the migration is feasible and also it is very promising for the need of composing systems of systems in the Future Internet.**

Keywords: *OpenFlow; OpenStack; private cloud; Software-Defined Networking.*

I. INTRODUCTION

Software-defined networking combined with cloud computing fascinates the research communities because of the huge potential for applications in the Future Internet. An OpenFlow-based technology is an innovative example of how a solution could be rapidly prototyped and deployed. Paper [6] investigates the major open source projects focused on this technology, following several criteria such as: (1) type of soft switching (e.g. Indigo from Big Switch Networks, LINC from infoBlox, OFSS from Ericsson, openvSwitch from Nicira/VMWare), (2) OpenFlow controllers (e.g. Beacon from Stanford University, Floodlight from Big Switch Networks, NodeFlow from Cisco, NOX and POX from ICSI), and (3) routing (e.g. FlowScale from Indiana University, Quagga from Quagga Routing project, Resonance from Georgia Tech, RouteFlow from CPqD). Orchestration and slicing are key parameters in designing any SDN infrastructure. FlowVisor from On.Lab, Maestro from Rice University, NDDI OE SS from Internet 2, or Neutron from OpenStack Foundation are among the most notable candidates, according to [6]. Security, simulation and testing tools, software libraries and SDN applications are also important when considering a solution.

This paper starts from our previous work described in [1] which is based on a real OpenFlow testbed. A four-node topology employed OpenvSwitches with a Beacon controller for Infrastructure Providers (IPs) and a light controller for the Service Providers (SPs). The main goal was that of providing a proof of concept for the theoretical idea of GRAS (Gearbox-like Routing Algorithms Selection) in runtime, initially proposed in [2].

We managed to demonstrate the increase of performance regarding end-to-end available transfer rates and latencies, when the single path routing involved Floyd-Warshall for IPs and Modified Dijkstra (proposed in [1]) for SPs. Whenever the needs cannot be fulfilled, the Beacon controller may decide to replace the single path approach by a multiple path Ford-Fulkerson scheme (for both types of provider). Although we demonstrated the benefits of this implementation, it is very difficult to expand it to a larger scalable testbed (having a lot more than four nodes).

Due to the exceptional evolution of orchestration and slicing, we are trying herein to discuss the principles needed to migrate the solution towards a private cloud. This concept refers to a computing model for the resources dedicated to a single organization. For instance Microsoft Private Cloud has the same characteristics as the public one (resource pooling, self-service, elasticity and eventually pay-by-use scheme), but with some new features: (1) virtualization, and (2) Infrastructure as a Service (IaaS). The integration of these features should be done as a whole, thus making the management even more critical than in the public cloud.

We investigated an open-source solution, i.e. OpenStack, employing the same simple four-node topology, with OpenvSwitch running on each virtual node. If we manage to demonstrate its feasibility, then a larger virtualized testbed can be implemented much easier than a larger real testbed with manual configuration. Such work is requested for instance in the CHIST-ERA "DIONASYS" project, where there is a need for composing systems of systems in the Future Internet.

The rest of the paper is organized as follows. Section II describes the design principles of a private cloud based on the OpenStack cloud solution. The experimental results in Section III refer to an SDN-based testbed running on single-node OpenStack, whilst in Section IV we discuss the multi-node topology. Conclusions and future work end the paper.

II. PRIVATE CLOUD BASED ON OPENSTACK

The initiative for using a cloud based testbed was attained from the fact that cloud computing brings important benefits, such as rapid elasticity of computing and networking resources. A cloud infrastructure is a suite of hardware and software that enables the five essential

Manuscript received December 18, 2015; revised February 10, 2016

characteristics of cloud computing. According to [4], these five essential characteristics for cloud systems are: on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service. In a cloud infrastructure both the physical layer and an abstraction layer are considered for inclusion. The physical layer consists of the hardware resources necessary to support the cloud services being provided. The abstraction layer consists of the software deployed across the physical layer, which materialize the essential cloud characteristics. When referring to cloud computing, it is also helpful to define whether a cloud is deployed within an organization or more broadly. From this perspective four main deployment models can be distinguished: (1) private clouds, (2) public clouds, (3) community clouds, and (4) hybrid clouds [5].

OpenStack [12] is an open source software suite that provides scalability for building public and private clouds. It provides both large and small organizations an alternative to closed cloud environments and allows for the deployment and management of an IaaS cloud platform. It also offers flexibility and choice through a highly engaged global community consisting of innovators, developers, software and hardware vendors, service providers, and more than 17,000 individuals and 500 companies including Rackspace, Dell, HP, IBM, and Red Hat (as of 2014). OpenStack lets users deploy virtual machines and other instances which handle different tasks for managing a cloud environment at the snap of a finger. Such service is provided by the numerous different moving parts. Because of its open nature, anyone can add additional components. However six key components have been defined and officially maintained (see their code names between brackets): (1) Identity and access management (KEYSTONE): responsible for the authentication and authorization service; (2) Image service (GLANCE) which stores and retrieves virtual machine disk images; (3) Compute (NOVA) manages the lifecycle of compute instances; (4) Networking (NEUTRON) enables network connectivity as a service for other OpenStack services (such as Compute); (5) Dashboard (HORIZON) provides a web-based self-service portal; (6) Usage data and orchestration: (HEAT) the orchestrator of the cloud, and (CEILOMETER) which monitors and meters services within the cloud. Note that there are several other components (e.g. IRONIC) which are not needed in this deployment [11].

As described before, a private cloud comprises of more subsystems that offer the desired concept functionality. In order to achieve the desired implementation several approaches were tested. OpenStack is highly configurable meeting different needs by providing numerous storage and networking options. The first step in designing the architecture was to choose whether to use single- or multi-node configuration.

III. EXPERIMENTAL RESULTS FOR SINGLE NODE OPENSTACK-BASED TESTBED

The initial implementation of a private cloud solution was based on a single node OpenStack Juno release, VMware Workstation 11 (trial version) and MobaXterm Personal Edition 7.7. The configuration for the virtual machine running OpenStack is presented in Table 1.

Operating System	Processor	RAM	HDD	Network
Ubuntu 14.04	2 cores	28 GB	Dynamic allocation < 250 GB	NAT connected 10.8.8.0/24

Table 1. Configuration of the cloud hosting machine

During the installation of the operating system, basic Ubuntu server and OpenSSH server were marked for installation. After the installation, a port forwarding rule was created in the NAT network in order to allow a SSH connection from the hosting machine using MobaXterm. The subsystems for the installation were selected by configuring the *local.conf* file from the home directory of the cloud installation. A description of the functionality of these subsystems is presented in Figure 1.

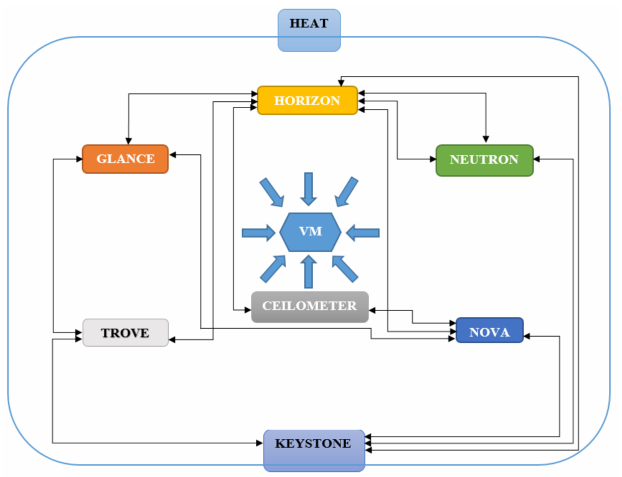


Figure 1.Subsystems of the single node OpenStack

Apart from the six core components mentioned before we have also installed TROVE, providing a scalable and reliable database. Once the installation was complete, a private cloud system was available for configuration. This was accomplished through either a graphical interface (i.e. HORIZON), or a terminal (i.e. CLI using MobaXterm) in seven steps (see Table 2).

Steps for creating a testbed	*HORIZON*	*CLI*
Image upload	✓	✓
Network creation	✓	✓
Network configuration	✓	✓
Security configuration	✓	✓
Instance creation	✓	✓
Instance access		✓
Instance configuration		✓

Table 2. Steps for creating the testbed

When choosing an operating system image, there is always a dilemma when referring to Linux, as different distributions are suited for different purposes. Unfortunately we could not rely on a homogeneous approach as several parts of this system were designed and implemented at different moments in time, with uncorrelated approaches. Summarizing, we had to use Ubuntu for OpenStack (due to a better support, provided in [9]), and CentOS for the SDN-based machines (our previous implementations were written on top of it). The testbed was based on [1], which proposes a four-node topology (SW1, SW2, SW3 and SW4), in this case four CentOS7 virtual machines that belong to an Infrastructure Provider: OpenvSwitch 2.3.1 LTS was installed on each of them. The same Linux distribution was installed on the Source and Destination machines. Fedora Core 18 has been used for Beacon. The traffic from Infrastructure Providers (IPs) and Service Providers (SPs) was generated by the Source connected to SW1, as illustrated in Figure 2. The Destination node connected to SW4 receives traffic from IPs, whilst the virtual machine VM4 (CirrOS) is needed to receive the traffic sent by SPs. All the nodes are directly connected through dedicated links to the centralized Beacon controller 1.0.

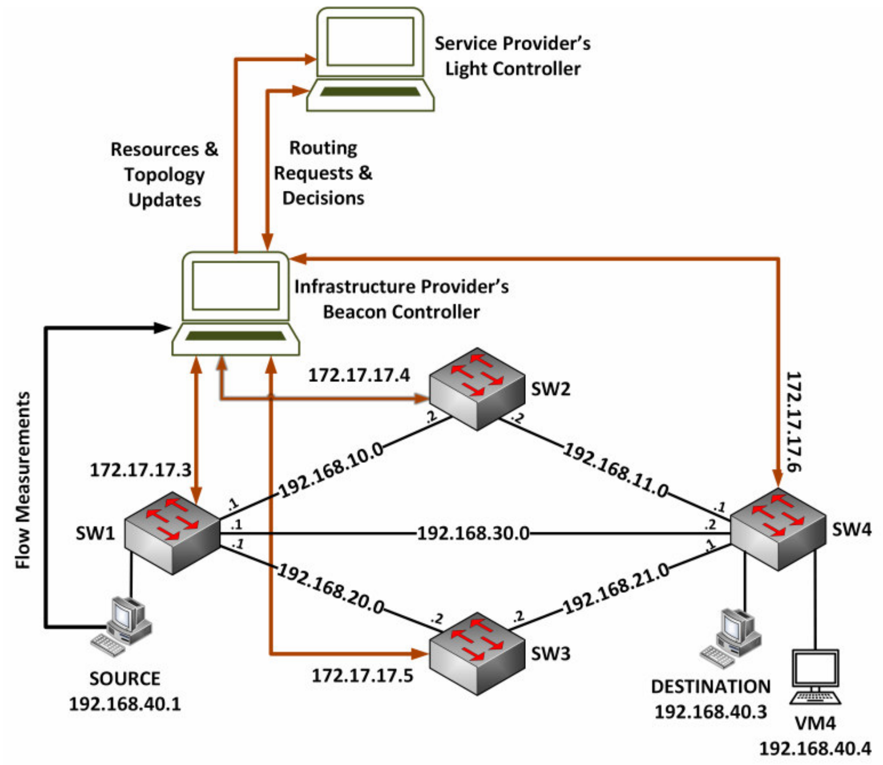


Figure 2. Real implementation testbed proposed in [1] that has to migrate towards an OpenStack-based private cloud

The Service Provider uses a light controller in CirrOS, in order to receive via sockets the resources and the topology information, as well as the routing requests. SP will send back its routing decisions to be applied by the Beacon controller of IP (on behalf of SP) in the real network. The number of nodes could be easily expanded, due to the scalability of the OpenFlow architecture. OpenvSwitch is configured to run in the forwarding mode: every time a packet of a flow arrives in a switch, the software tries to identify the corresponding flow. The switch forwards the packet to the proper port whenever there is an action defined for the respective flow; otherwise the software encapsulates the packet and sends it to the controller, which returns the action to be applied to it. For more information about the implementation details see [3].

The network configuration includes all of the necessary modifications to the NEUTRON settings and files in order to provide SSH access and Internet access to/from the instances. Security configuration provides the SSH keys and the firewall rules in the attached security group for all the instances. These instances are created through the CLI even if the option of GUI (HORIZON) is available, because the CLI provides the possibility of specific characteristics configuration, such as the static IP and specific MAC addresses. Access to the instances is obtained through the CLI using the predefined SSH key, with all the necessary configuration and software installation being performed there.

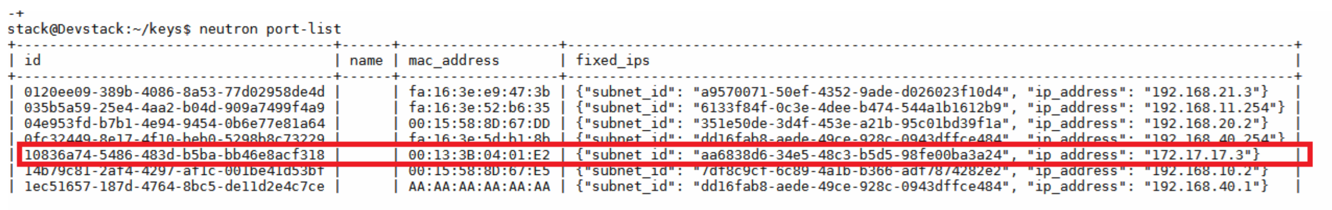
Although the installation of the OpenStack subsystems following the explanations provided in [9],[11] is not a trivial task, it is out of the scope of this paper to include a description of all the steps. The MAC and IP addresses are configured as follows:

```
$ neutron port-create <name of the network> --mac-
address <MAC> --fixed-ip ip_address=<IP>
```

Figure 3 demonstrates that the configuration was successful. The most important information herein is within column "id", representing the port ID used when we created the virtual machines corresponding to the previously mentioned entities. For instance 10836a74-5486-483d-b5ba-bb46e8acf318 is the identifier for the interface eth0 of the SW1 with the IP address 172.17.17.3.

The virtual machines images inside the cloud will only have one network interface for each instance enabled by default, so we have to manually power up all the remaining interfaces. We must connect with SSH to each machine and configure every interface. The successful migration is proved by gaining the connectivity between all machines running under the control of OpenStack. Furthermore, we were now able to reproduce in a private cloud the experiments described in [1].

We started the query agent on each switch in order to measure the available transfer rate and latency on the data plane for each link. The principle of calculating the Available Transfer Rate (ATR) takes as a reference the maximum transfer rate at top of MAC-Sublayer in the most pessimistic case (e.g. 82.051 Mbps for FastEthernet [2] with no background traffic). The total link flow on that link, measured once every second, is subtracted from the reference and provides the first result. The values for ATR obtained in the cloud are comparable to those obtained in the real implementation [1]. While the latency is still estimated as half of the Round-Trip Time, the accuracy of the measurements is better in a private cloud, because the VMs are running on the same physical host. This partially compensates the lack of synchronization between nodes.



```
-+
stack@Devstack:~/keys$ neutron port-list
+--------------------------------------+------+-------------------+-----------------------------------------------------------------------------------------+
| id                                   | name | mac_address       | fixed_ips                                                                               |
+--------------------------------------+------+-------------------+-----------------------------------------------------------------------------------------+
| 0120ee09-389b-4086-8a53-77d02958de4d |      | fa:16:3e:e9:47:3b | {"subnet_id": "a9570071-50ef-4352-9ade-d026023f10d4", "ip_address": "192.168.21.3"}   |
| 035b5a59-25e4-4aa2-b04d-909a7499f4a9 |      | fa:16:3e:52:b6:35 | {"subnet_id": "6133f84f-0c3e-4dee-b474-544a1b1612b9", "ip_address": "192.168.11.254"} |
| 04e953fd-b7b1-4e94-9454-0b6e77e81a64 |      | 00:15:58:8D:67:DD | {"subnet_id": "351e50de-3d4f-453e-a21b-95c01bd39f1a", "ip_address": "192.168.20.2"}   |
| 0fc32449-8e17-4f10-beb0-5298b8c73229 |      | fa:16:3e:5d:b1:8b | {"subnet_id": "dd16fab8-aede-49ce-928c-0943dffce484", "ip_address": "192.168.40.254"} |
| 10836a74-5486-483d-b5ba-bb46e8acf318 |      | 00:13:3B:04:01:E2 | {"subnet_id": "aa6838d6-34e5-48c3-b5d5-98fe00ba3a24", "ip_address": "172.17.17.3"}    |
| 14b79c81-2af4-4297-af1c-001be41d53bf |      | 00:15:58:8D:67:E5 | {"subnet_id": "7df8c9cf-6c89-4a1b-b366-adf874282e2", "ip_address": "192.168.10.2"}    |
| 1ec51657-187d-4764-8bc5-de11d2e4c7ce |      | AA:AA:AA:AA:AA:AA | {"subnet_id": "dd16fab8-aede-49ce-928c-0943dffce484", "ip_address": "192.168.40.1"}   |
```

Figure 3. Port-list (interfaces list) of NEUTRON

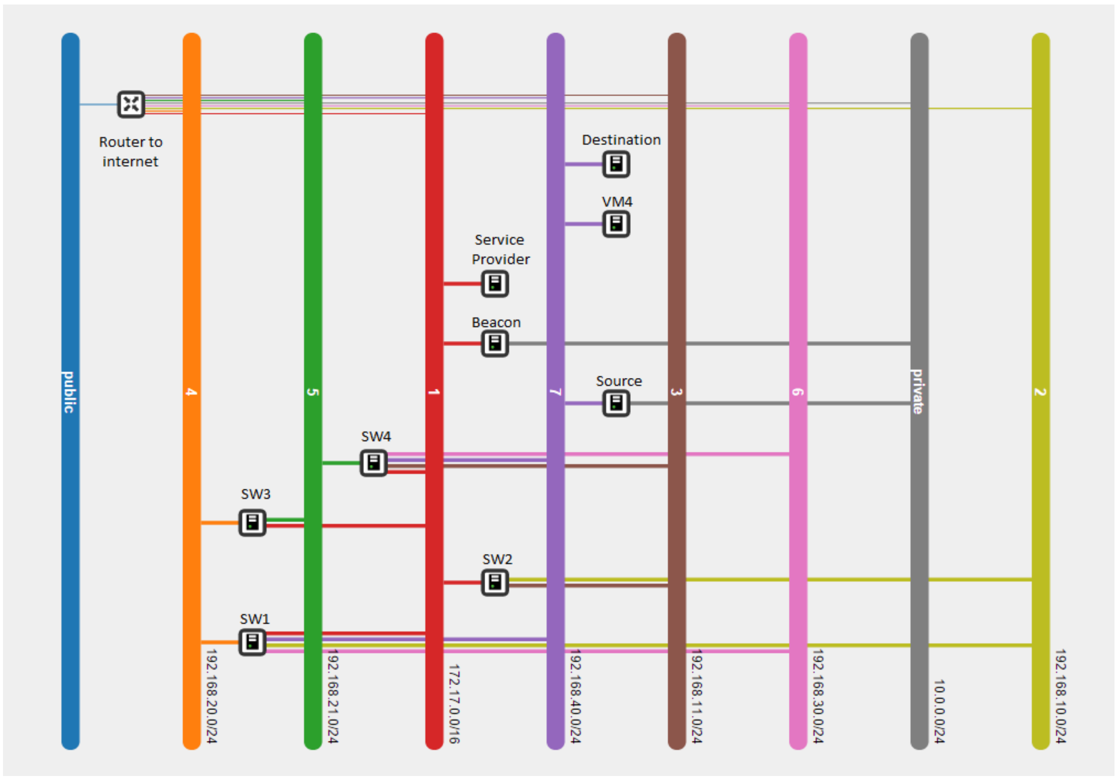


Figure 4. SDN-based testbed topology in OpenStack

Test 1: No background traffic, no OpenFlow:
In the first step, we did not start OpenFlow within the testbed described in Figure 4. This means that none of the routing algorithms were enabled in Beacon and the flow tables were not operational. Figure 5 shows an example of results, where S is the source OpenvSwitch, D is the destination; OWD represents the latency in [s] and ATR is the Available Transfer Rate in [bps]. The values are stored in a file, which is updated once every second. This step is important for calibrating the measurements. For instance, the active measurement of the OWD by sending probes between neighboring nodes contributes with about 82,051,000-82,049,725=1275 bps, whilst the ATR measurement is passive.

```
S D OWD      ATR
1 2 0.000451 82049723.1
4 2 0.000556 82049721.0
1 3 0.000926 82049728.7
4 3 0.000991 82049723.1
4 1 0.001458 82049725.3
2 1 0.000760 82049721.9
3 1 0.002436 82049727.8
2 4 0.000725 82049725.6
3 4 0.002230 82049723.2
1 4 0.004908 82049724.6
```

Figure 5. Snapshot of the measurements in the private cloud without OpenFlow and without background traffic

An initial 16 GB-RAM single-node testbed (8 GB RAM for physical machine, 4,416 MB RAM for the OpenStack, seven VMs with 512 MB each and 128/64MB for SP/VM4) showed an average round-trip time/link up to 7 ms. Thus we decided to increase the resources as follows: each machine used one virtual CPU, 10 GB on disk, 4 GB RAM (except SP and VM4 which ran cirros0.3.2-x86_64-uec with less RAM, i.e. 512 MB and no disk resources needed). Also the virtual machines for the Source and Destination used 1 GB RAM each. Note that the physical machine running Windows and hosting the cloud had 32 GB RAM in total (4 GB RAM for physical, 5 GB RAM for the OpenStack and 23 GB for the virtual machines).

The RTT for each link varied between 1…2 ms (in average ten times more than the expected value for a real Fast Ethernet point-to-point link).

Test 2: Background traffic, no OpenFlow
UDP traffic of about 30 Mbps was injected into the link 1-4 using *iperf*. The response did not follow the request, i.e. the average rate injected being 9.17 Mbps, as in Figure 6.

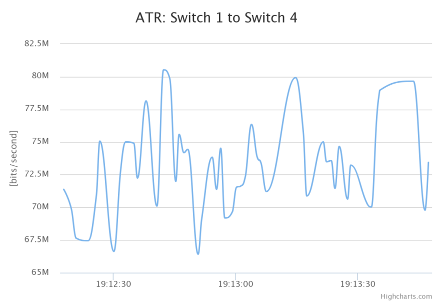


Figure 6. ATR for the link 1-4 in the private cloud with desired background traffic of 30 Mbps

Although the migration into the single-node OpenStack-based testbed was apparently successful, we concluded that for the given resources (CPU, memory etc.) the performance of the virtualized solution is unacceptable compared to the physical testbed. Thus no more experiments involving SDN part were carried out. We estimated that the following technical configurations will make the results for OWD become more competitive in the cloud: (1) allocation of at least 8 GB RAM for the OpenStack controller; (2) replacing the SATA HDD technology for the hard disk with SATA SSD; (3) splitting the OpenStack deployment by allocating a dedicated physical machine to the compute node; (4) removing components that are not used (CEILOMETER, TROVE, HEAT). Furthermore, we investigated an additional level of splitting, the Network Node (Neutron) being placed on a different virtual machine. Note that a data center with larger resources would have allowed this single-node OpenStack testbed to be fully functional. However our approach is to evaluate this migration process with low cost equipment (motherboards with maximum 32 GB DDR3 RAM), i.e. affordable for a larger community of researchers. Note that a 64 GB (or higher) DDR4 RAM-based motherboard was not available by the time we performed the experiments and its current price is still a lot more than the price of two 32 GB RAM-based consumer PCs [16]. Therefore, the multi-node approach was investigated within the next section.

IV. EXPERIMENTAL RESULTS FOR A MULTI-NODE OPENSTACK-BASED TESTBED

The manual multi-node installation of OpenStack was carried out following the instructions provided in [11]. One of the immediate advantages referred to the possibility of restarting the OpenStack environment that was deployed in this manual mode. Note that other installations had to be recreated from scratch whenever a hardware upgrade was intended or whenever the power was cut off. The other two deployment methods experimented herein are the following: a) RDO (Rebuilt Daily, Regularly Delivered, OpenStack) [13]; b) DevStack which is not and has never been intended to be a general installer, although it evolved to support a large number of configuration options and alternative platforms and services [14]. For the power cut off VMware offered a solution through the snapshot feature, but the hardware upgrade of the OpenStack environment host still remained.

The Juno version of OpenStack was selected to run under Ubuntu 14.04. The discussed architecture comprises three nodes: Controller that runs services as in Figure 10, Network (see Figure 11) and Compute (presented in Figure 12).

While in the previous deployments the environment was only virtual, for this one a merger of physical and virtual entities was selected. The description of the testbed topology presented in Figure 8 includes only the amount of RAM, which is the key performance indicator. According to Figure 9, for better RTT between two virtual machines compared to the case of two physical ones, the minimum RAM for Compute Node was proved to be 32 GB.

First we configured the network topology on all machines, updated the system and imported the cloud repositories. There was an enclosed and synchronized environment where the Controller Node offered NTP server with the others as clients. In this manual deployment the level of customizing the services of the cloud is even more detailed and demanding than in DevStack. For the purpose of the testbed only basic services were selected. The configured services on the Controller node are presented in Figure 10.

Installing all the services presented above is done through command line coding and file editing. In this deployment some additional services were installed in order to offer support and functionality to the core ones. MySQL provides a database with the basic services and their credentials, RabbitMQ [15] is the messaging system between the OpenStack sevices and NTP provides synchronization.

The Network Node runs the Networking plug-in and different agents of the services presented in Figure 11. The following were installed and configured: ML2 (a plugin for NEUTRON that enables different network virtualizing technologies), OpenvSwitch Agent (the network virtualization technology used), DHCP Agent (provides dynamic IP addresses for the instances), Layer 3 Agent (offers Internet connectivity to the testbed) and Metadata Agent (injects different configurations into the instances when they boot like the SSH key).

The single Compute Node in the current testbed had the services presented in Figure 12. KVM is the key one on this node, providing the hypervisor functionality for NOVA. After that the functionality of the services can be verified from the terminal of the Controller Node, presented in Figure 13.

Once the installation is finished the OpenStack dashboard is available and ready to use. In the present case the IP where the dashboard could be accessed was http://192.168.100.11/horizon/auth/login.

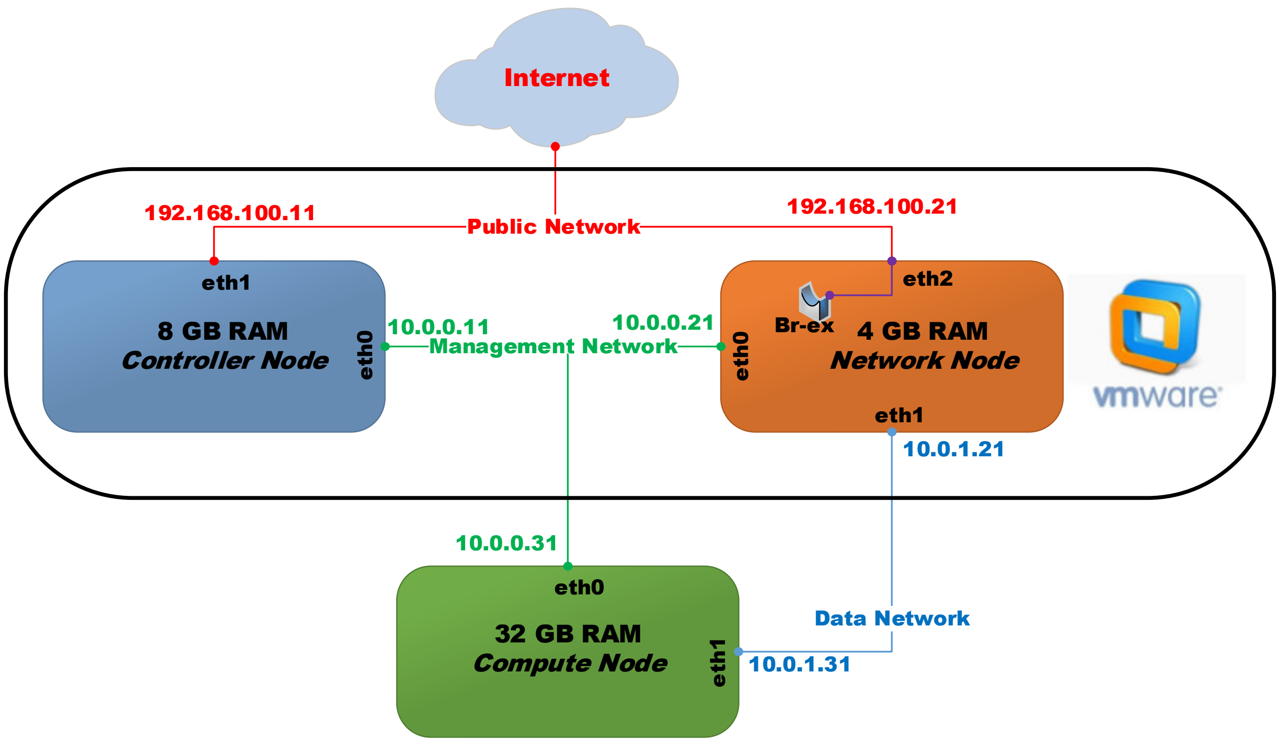
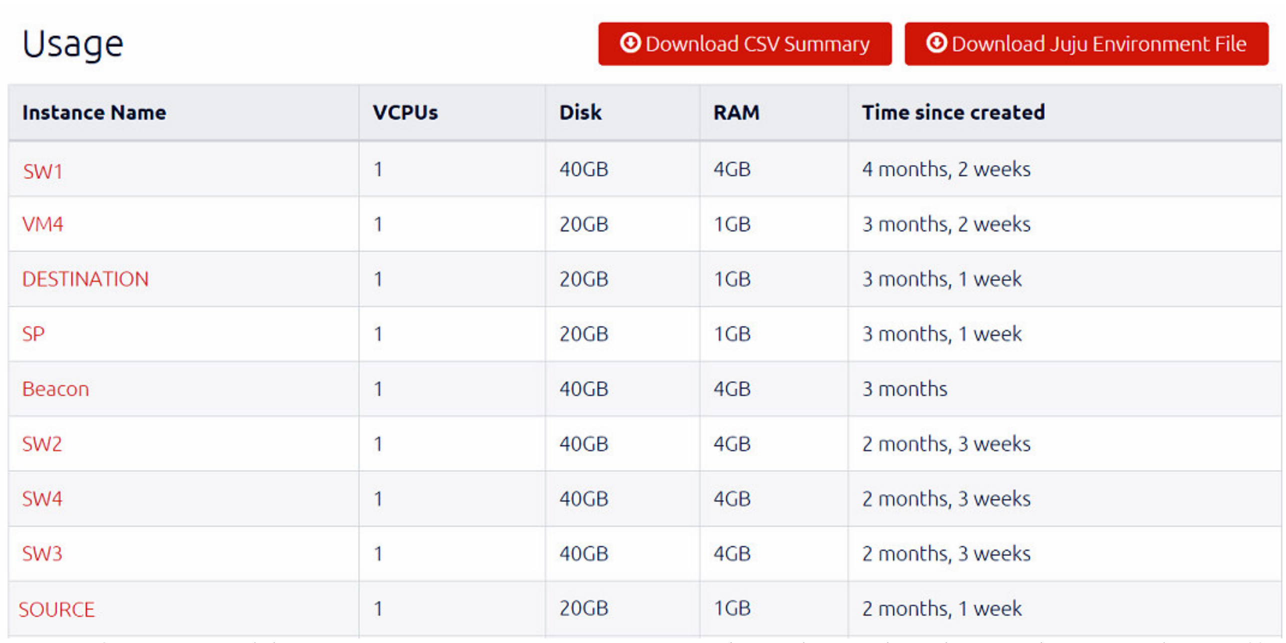


Figure 7. Multi-node OpenStack-based testbed



Instance Name	VCPUs	Disk	RAM	Time since created
SW1	1	40GB	4GB	4 months, 2 weeks
VM4	1	20GB	1GB	3 months, 2 weeks
DESTINATION	1	20GB	1GB	3 months, 1 week
SP	1	20GB	1GB	3 months, 1 week
Beacon	1	40GB	4GB	3 months
SW2	1	40GB	4GB	2 months, 3 weeks
SW4	1	40GB	4GB	2 months, 3 weeks
SW3	1	40GB	4GB	2 months, 3 weeks
SOURCE	1	20GB	1GB	2 months, 1 week

Figure 8. Overview of the nine instances running on Compute Node: each virtual machine used one virtual CPU, 40 GB on disk, 4 GB RAM (except Source, Destination, SP and VM4 which needed 20 GB on disk, 1 GB RAM).

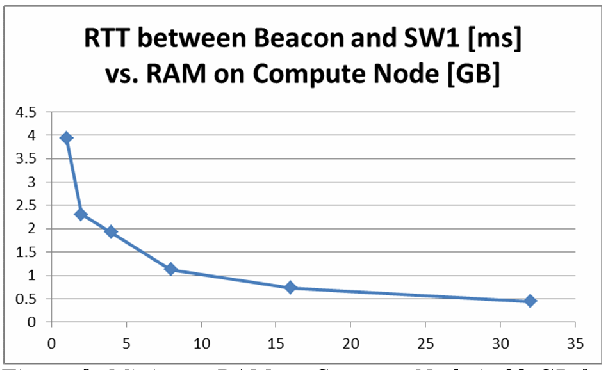


Figure 9. Minimum RAM on Compute Node is 32 GB for better average RTT =0.445 ms between two virtual machines (Beacon and SW1) compared to average RTT=0.552 ms between two physical machines

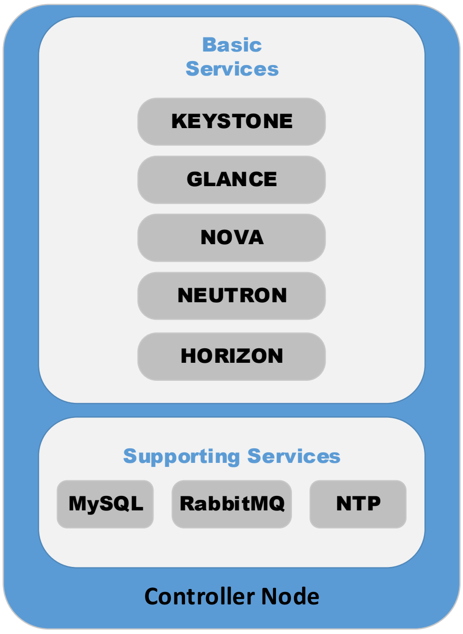


Figure 10. Configured services on the Controller Node

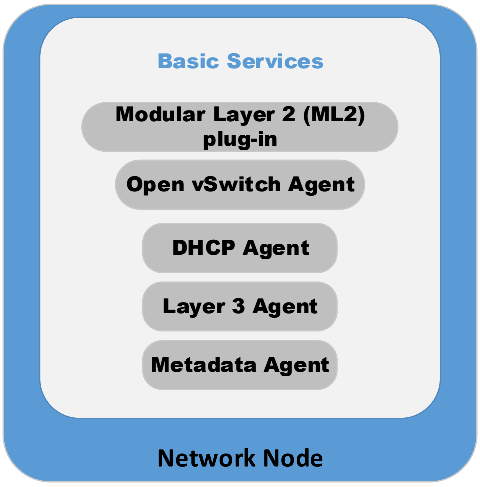


Figure 11. Configured services on the Network Node

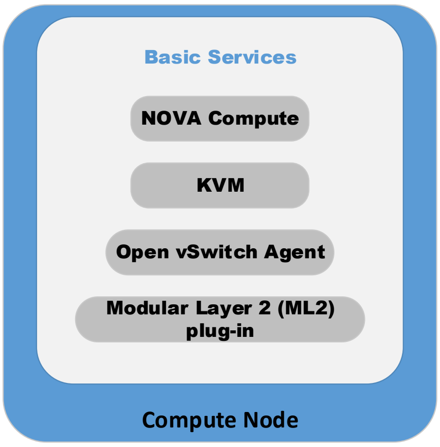


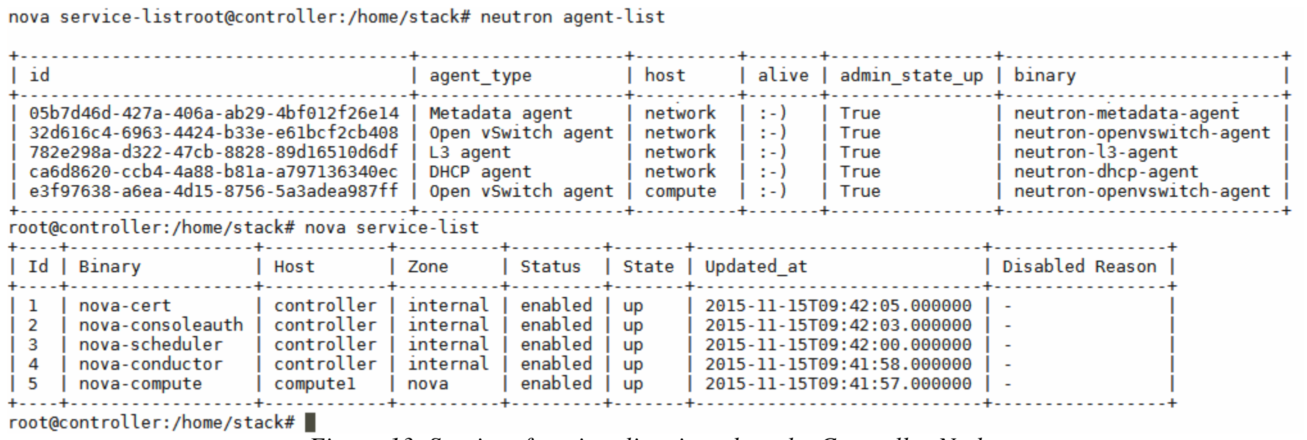
Figure 12. Configured services on the Compute Node

GRE tunneling was selected for the networking because the network component (Neutron) with VLAN topology requires a predefined maximum number of tenant's value and underlying network equipment configuration while Neutron with GRE topology allows more flexibility. The downside is that this encapsulation decreases the speed of communication between the VMs because of the increased CPU, RAM and HDD utilization. That is why several deployments with various hardware configurations were tried out in order to determine if the performance of the virtual testbed comes close or is even better than the one of the real testbed.

Table 3 presents a comparison between the performance of the network services on the VMs inside the OpenStack and the physical testbed. After pinging 100 times between Beacon (local machine instance in all the deployments) and the virtual machine SW1, the average RTT obtained was highly dependent on the amount of RAM memory installed.

Throughout the experimenting with different hardware capacity deployments the characteristics of the machines inside of the OpenStack environment were modified as follows: from about 512 MB or 1 GB RAM per virtual machine on the single-node experiment up to 1 GB or 4 GB per virtual machine on multi-node solution (see Figure 8).

The average RTT for single-node OpenStack was worse than that for physical testbed (0.685 ms. vs. 0.049 ms for loopback, respectively 1.214 ms. vs. 0.552 ms for Beacon-SW1). The three-node virtualized topology was better than the real implementation with respect to this parameter (i.e. 0.036 ms. for loopback and 0.445 ms. for Beacon-SW1) which is very promising. This encourages us to deploy faster and easier in the future larger scalable networks.



```
nova service-listroot@controller:/home/stack# neutron agent-list
+--------------------------------------+--------------------+---------+-------+----------------+---------------------------+
| id                                   | agent_type         | host    | alive | admin_state_up | binary                    |
+--------------------------------------+--------------------+---------+-------+----------------+---------------------------+
| 05b7d46d-427a-406a-ab29-4bf012f26e14 | Metadata agent     | network | :-)   | True           | neutron-metadata-agent    |
| 32d616c4-6963-4424-b33e-e61bcf2cb408 | Open vSwitch agent | network | :-)   | True           | neutron-openvswitch-agent |
| 782e298a-d322-47cb-8828-89d16510d6df | L3 agent           | network | :-)   | True           | neutron-l3-agent          |
| ca6d8620-ccb4-4a88-b81a-a797136340ec | DHCP agent         | network | :-)   | True           | neutron-dhcp-agent        |
| e3f97638-a6ea-4d15-8756-5a3adea987ff | Open vSwitch agent | compute | :-)   | True           | neutron-openvswitch-agent |
+--------------------------------------+--------------------+---------+-------+----------------+---------------------------+
root@controller:/home/stack# nova service-list
+----+------------------+------------+----------+---------+-------+----------------------------+-----------------+
| Id | Binary           | Host       | Zone     | Status  | State | Updated_at                 | Disabled Reason |
+----+------------------+------------+----------+---------+-------+----------------------------+-----------------+
| 1  | nova-cert        | controller | internal | enabled | up    | 2015-11-15T09:42:05.000000 | -               |
| 2  | nova-consoleauth | controller | internal | enabled | up    | 2015-11-15T09:42:03.000000 | -               |
| 3  | nova-scheduler   | controller | internal | enabled | up    | 2015-11-15T09:42:00.000000 | -               |
| 4  | nova-conductor   | controller | internal | enabled | up    | 2015-11-15T09:41:58.000000 | -               |
| 5  | nova-compute     | compute1   | nova     | enabled | up    | 2015-11-15T09:41:57.000000 | -               |
+----+------------------+------------+----------+---------+-------+----------------------------+-----------------+
root@controller:/home/stack# █
```

Figure 13. Services functionality viewed on the Controller Node

OpenStack Deployment	Single-node 8 GB RAM VM		Single-node 28 GB RAM VM		Three-node VMs		Physical Testbed [1]	
	Loopback	Beacon-SW1	Loopback	Beacon-SW1	Loopback	Beacon-SW1	Loopback	Beacon-SW1
RDO	0.994	2.757	-	-	-	-	-	-
DevStack	0.943	2.643	0.685	1.214	-	-	-	-
Multi-Node	-	-	-	-	0.036	0.445	-	-
Physical Testbed [1]	-	-	-	-	-	-	0.049	0.552

Table 3. Comparison regarding RTT in milliseconds between Beacon and SW1

A network performance anomaly was encountered in the first two single-node deployments (RDO, DevStack) when using *iperf*. We tested the communication between two instances, but no matter how high the throughput of the probe was, the maximum transfer rate always remained around 27 Mbps. Thus the network traffic flows could not be controlled and the real testbed could not be replicated. On the multi-node deployment a maximum transfer rate of 27.1 Gbps between virtual machines was recorded. Another drawback for the first two single-node deployments refers to the instances that responded very slowly when connected via SSH. This is due to the fact that the host operating system ran on a HDD disk which was used at full capacity, as illustrated in Figure 14.

The countermeasure for this was to move the operating system on a SSD disk. The results were improved significantly with respect to the machine response speed and the usage, as can be observed in Figure 15. Note that the maximum usage was about 18% of the disk activity.

Subsequently to the tweaking of the OpenStack deployment the SDN-based testbed was verified regarding topology recognition. The SDN links (topology) discovered by the Beacon are illustrated in Figure 16.

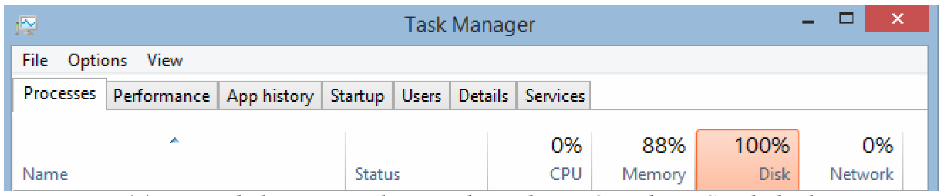


Figure 14. HDD disk usage within single-node RDO and DevStack deployments

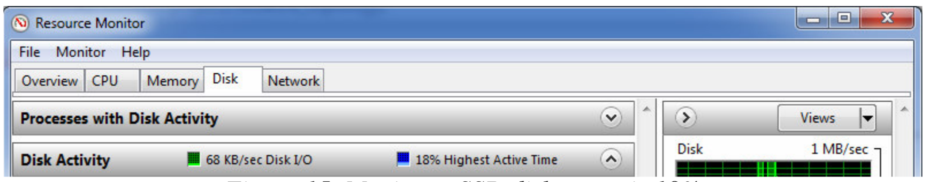
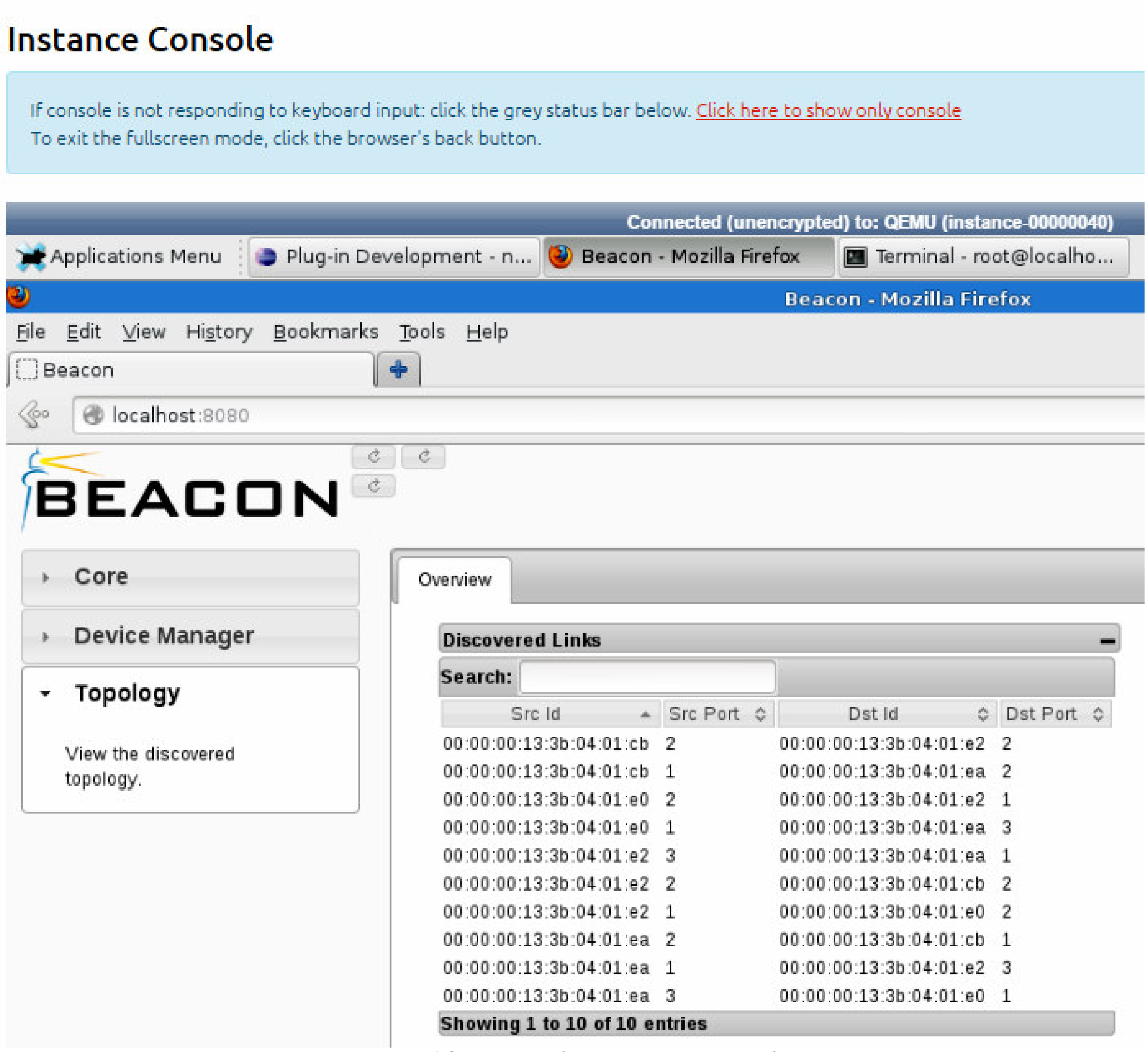


Figure 15. Maximum SSD disk usage is 18%


Instance Console
If console is not responding to keyboard input: click the grey status bar below. Click here to show only console
To exit the fullscreen mode, click the browser's back button.
Connected (unencrypted) to: QEMU (instance-00000040)
Applications Menu
Plug-in Development - n...
Beacon - Mozilla Firefox
Terminal - root@localho...
Beacon - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Beacon
localhost:8080
BEACON
Core
Device Manager
Topology
View the discovered topology.
Overview
Discovered Links
Search:

Src Id	Src Port	Dst Id	Dst Port
00:00:00:13:3b:04:01:cb	2	00:00:00:13:3b:04:01:e2	2
00:00:00:13:3b:04:01:cb	1	00:00:00:13:3b:04:01:ea	2
00:00:00:13:3b:04:01:e0	2	00:00:00:13:3b:04:01:e2	1
00:00:00:13:3b:04:01:e0	1	00:00:00:13:3b:04:01:ea	3
00:00:00:13:3b:04:01:e2	3	00:00:00:13:3b:04:01:ea	1
00:00:00:13:3b:04:01:e2	2	00:00:00:13:3b:04:01:cb	2
00:00:00:13:3b:04:01:e2	1	00:00:00:13:3b:04:01:e0	2
00:00:00:13:3b:04:01:ea	2	00:00:00:13:3b:04:01:cb	1
00:00:00:13:3b:04:01:ea	1	00:00:00:13:3b:04:01:e2	3
00:00:00:13:3b:04:01:ea	3	00:00:00:13:3b:04:01:e0	1

Showing 1 to 10 of 10 entries

Figure 16. SDN topology on Beacon console

This proved that the migration into the multi-node OpenStack-based testbed was successful and the performances of the virtualized solution are comparable to the physical testbed ones. It is out of the scope of this paper to investigate other key performance indicators, as the next steps of evaluation should be conducted with respect to SDN-based routing schemes [1], [2], [3] and not with respect to OpenStack components.

V. CONCLUSIONS AND FUTURE WORK

An SDN-based testbed running on low cost equipment can successfully run on an OpenStack-based private cloud, allowing the deployment of larger scalable networks. The Future Internet requires systems of systems, and the herein solution could fulfil such complex requirements. It is for further work to optimize the resources (number of cores, number of GB for hard disk/ RAM etc.). Next, we will continue to monitor the performance with respect to the QoS parameters (by active/passive measurements), routing schemes (how efficient are the simple/multiple path mechanisms in the virtual world), management (already simplified by OpenStack) and security (still an open issue).

ACKNOWLEDGMENTS

This work was partially supported by the CHIST-ERA “DIONASYS” project. However the views expressed in this paper are solely those of the authors and do not necessarily represent the views of the entire project. Thanks go to Dr. Yehia Elkhatib from the University of Lancaster for his valuable comments.

REFERENCES

[1] M.V.Ulinic, A.B.Rus, and V.Dobrota, "OpenFlow-Based Implementation of a Gearbox-Like Routing Algorithm Selection in Runtime", *ACTA TECHNICA NAPOCENSIS, Electronics and Telecommunications*, ISSN 1221-6542, Vol.55, No.2, 2014, pp.23-32

[2] A.B.Rus and V.Dobrota, "Case Study of a Gearbox-Like Routing Algorithm Selection in Runtime", *18th IEEE LANMAN 2011*, Chapel Hill, North Carolina, pp. 1-6, DOI: 10.1109/LANMAN.2011.6076938

[3] A.G.Furculita, M.V.Ulinic, A.B.Rus, and V.Dobrota,"Implementation Issues for Modified Dijkstra's and Floyd-Warshall Algorithms in OpenFlow", *12th RoEduNet International Conference*, Constanta, Romania, 2013, pp.50-55, DOI:10.1109/RoEduNet.2013.6714208

[4] The NIST Definition of Cloud Computing, 2011, Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf, [Accessed May 2015]

[5] "Cloud Computing: The Concept, Impacts and the Role of Government Policy", OECD Digital Economy Papers, No. 240, *OECD Publishing*, 2014, Available: http://dx.doi.org/10.1787/5jxzf4lcc7f5-en, [Accessed December 2015]

[6] S.Azodolmolky, P.Wieder, and R.Yahyapour, "Cloud computing networking: challenges and opportunities for innovations", *IEEE Communications Magazine,* 51(7), 2013, pp.54-62

[7] A.Lara, A.Kolasani, and B.Ramamurthy, "Network innovation using openflow: A survey". *IEEE Communications Surveys & Tutorials*, 16(1), 2014, pp. 493-512.

[8] B.Nunes, M.Mendonca, X.Nguyen, K.Obraczka, and T.Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks", HAL Id: hal-00825087, 2014. Available: https://hal.inria.fr/hal-00825087v2, [Accessed May 2015]

[9] Canonical "OpenStack deployment starts here..", Available: http://assets.ubuntu.com/sites/ubuntu/latest/u/files/section/cloud/openstack-deployment.pdf

[10] I. Pepelnjak, SDN and OpenFlow - The Hype and the Harsh Reality, *ipSpace.net AG*, 2014

[11] OpenStack official installation guide, Available: http://docs.openstack.org/juno/install-guide/install/apt/content/, [Accessed December 2015]

[12] F.Lespinasse, A quick overview of OpenStack technology, 2014, Available: http://www.thoughtsoncloud.com/2014/08/quick-overview-openstack-technology/, [Accessed December 2015]

[13] RDO OpenStack installation guide, Available: https://www.rdoproject.org/rdo/faq/, [Accessed November 2015]

[14] DevStack OpenStack installation guide, Available: https://wiki.openstack.org/wiki/DevStack, [Accessed November 2015]

[15] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures", Dissertation, *University of California Irvine*, 2000

[16] A. Wu, "Average DRAM Contract Price Sank Nearly 10% in October Due to Ongoing Supply Glut", *TrendForce Reports*, November 2015, Available: http://press.trendforce.com/node/create_pdf/2145, [Accessed November 2015]