sdnrg
Internet-Draft
Intended status: Informational
Expires: January 7, 2017

J. Wang
R. Gu
China Mobile

July 6, 2016

Practice of deploying SDN and VNFs in the data center
draft-wang-sdnrg-deployment-sdn-vnf-01.txt

Abstract

This document introduces a practice of deploying SDN and VNFs in cloud data center and discusses how SDN and VNFs can be combined in the data center.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 7, 2017.

Table of Contents

1. Introduction 2
2. The deployment of SDN and VNFs 2
2.1. SDN controller 4
2.2. VNF/VNFM 4
2.3. Interfaces 5
2.3.1. the interface between SDN controller and OVS 5
2.3.2. the interface between SDN controller and VNFM 5
2.3.3. the interface between SDN controller and VNF 5
3. The operation of the SDN controller and VNFs in the current deployment 5
4. Combinations of SDN controller and VNFs 9
5. Security Considerations 12
6. IANA Considerations 12
7. Normative References 12
Authors' Addresses 12

1. Introduction

SDN (Software Defined Networking) and NFV (Network Functions Virtualization) have placed a critical role in today's data center. SDN decouples the network control from forwarding and makes the network programmable. NFV uses the technologies of IT virtualization to virtualize entire classes of network node functions as the VNF (Virtual Network Function). Deploying SDN and VNFs in the data center can make the network more flexible and increase the network utilization.

This document first introduces a practice of deployment of SDN and VNFs in the data center, and then discusses how SDN and VNFs can be combined in the data center.

2. The deployment of SDN and VNFs

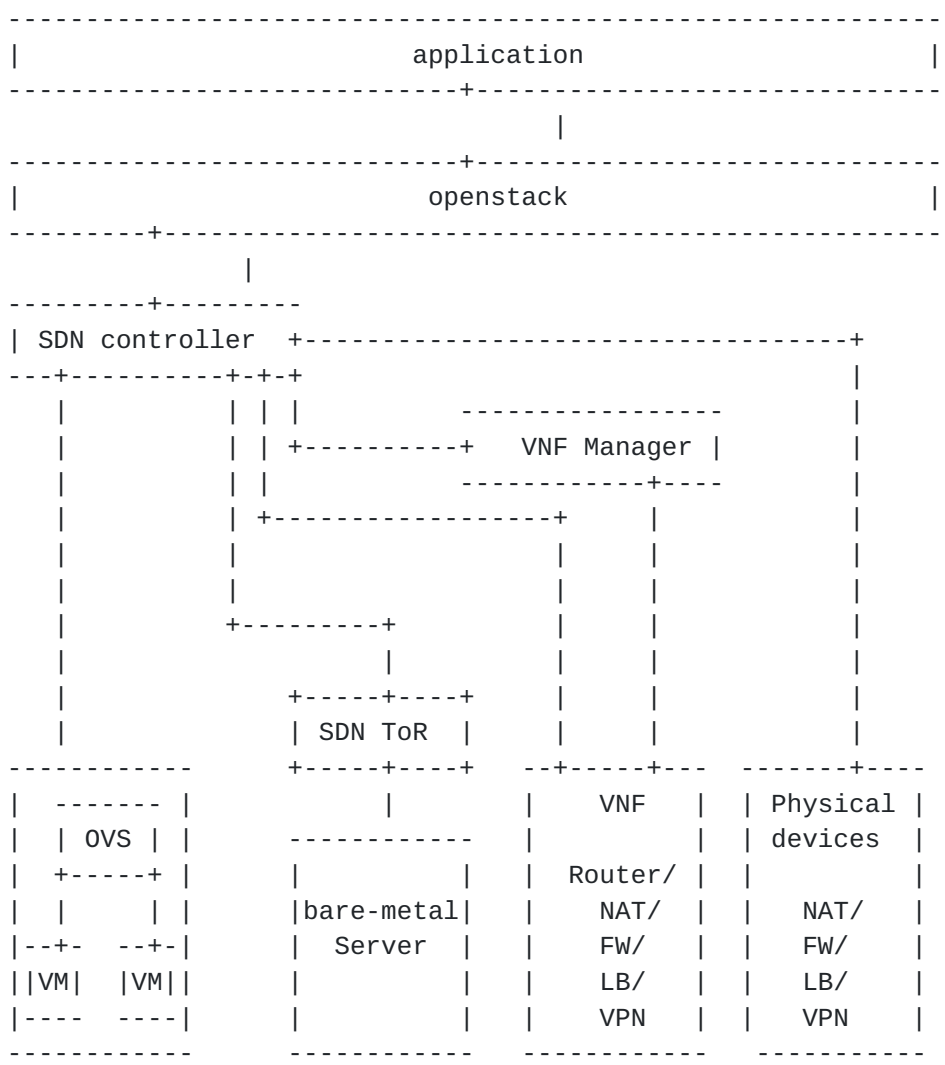


Figure 1: Scenario of SDN and NFV deployed in cloud datacenters

The deployed SDN data center includes OPENSTACK, SDN controller, OVS (Open Virtual Switch), VNFM (Virtualization Network Function Manager) and VNF (Virtualization Network Function) generally. The OPENSTACK manages the virtualization infrastructure include computing, storage and the network. Neutron is the network component of the OPENSTACK which defines the abstract network API to describe the network. SDN controller can be taken as the network solution deployed at the southbound of the Neutron. On the one hand, the SDN controller receives the messages from the neutron which describes the virtual network operation (such as create a network/port/security group), on the other hand, it interprets this operations and maps them into real network operations (such as create a VLAN/ACL or set OPENFLOW tables on the OVS). The OVS is deployed at the server and connects the VMs to the physical network. In the SDN scenarios, the OVS is an OPENFLOW switch. VNF is a VM/VMs which implements the network

functions such as the Router/NAT/Firewall/LoadBalance/VPN. VNFM is a centralized management entity which takes charge of creating/configuring/destroying the VNFs. In the scenario of deployment, baremetal servers and some physical devices of NAT/FW/LB/VPN exist as well. Because in private cloud use cases, some bare-metal servers are needed for specific services such as high performance computing clusters, database hosting which performs better in hypervisor and so on. And in this situation, SDN ToR (Top of Rack) is needed on the top of bare-metal server as the VTEP. Physical NAT/FW/LB/VPN devices are in demand for the reason that physical devices have already existed in datacenter.

2.1. SDN controller

SDN controller is the centralized control unit of the virtual network. On the one hand, it receives the messages from the OPENSTACK NEUTRON which describes the virtual network (such as network/subnet created, VM port attached to the subnet). On the other hand, the SDN controller reads the configuration from the OVS and knows the real deployment (such as VM attaches to which OVS). Then SDN controller takes charge of mapping the virtual network into the real deployment.

SDN controller talks with the OVS via OVSDB and OPENFLOW protocol to handle the VM related network deployment operation. When a new created VM is attached to the OVS, the OVS notifies the SDN controller through OVSDB. Then the SDN controller sends OPENFLOW flow entry to the OVS to implement access policy control and routing. In our deployment, VXLAN is used and thus the SDN controller also configures the OVS to setup VXLAN tunnel with other OVS or VNF.

SDN controller interacts with the VNF/VNFM to handle virtual network function (including vRouter /vFW/vLB/vVPN etc.) related operation, which includes create/delete/update/retrieve the virtual network function. For example when a "Router" is created in OPENSTACK, NEUTRON notifies SDN controller, which then interacts with the VNFM to create a network function VM implementing the Router function. After that, the VNFM sends the feedback to SDN controller, which indicates the information (includes IP address, login username/password for authentication, etc) about the new created virtual Router. Based on this information, SDN controller contacts the created virtual Router and then configures it to route VM's packets.

2.2. VNF/VNFM

VNF is a VM/VMs which implements the network function such as Router/FW/LB/VPN. VNFM is a management entity which takes charge of managing the VNF. First, the VNFM is responsible for creating or

deleting the VNF VM. Second, the VNFM can configure the created VNF to implement specific function via NETCONF/RESTCON/Rest API. For example, the VNFM can configure the routing information for Routing function, configure the security/access policy for Firewall function and configure the virtual IP or policy for load balance function. Note that the VNF can also be configured by SDN controller.

In our current deployment, the VNFM provides the Rest API to SDN controller in order to create/delete the VNF VM. The VNF provides both NETCONF and OPENFLOW interface to SDN controller, where the NETCONF is used to set the network function related configuration such as the Firewall policy, LoadBalance virtual IP etc. The OPENFLOW is used to configure the VM related routing information. For example when forwarding packets to the tenant VM, the OPENFLOW flow entry indicates which VXLAN tunnel to deliver the packets and how to set the VXLAN ID of the forwarded packets.

2.3. Interfaces

2.3.1. the interface between SDN controller and OVS

OVSDB: protocol for configuring the OVS.

OPENFLOW: protocol for setting flow entry to OVS, which takes charge of data forwarding.

Since the OVS is the de-facto standard of the virtual switch, the OVSDB and OPENFLOW which is used by the OVS can be seem as the standard interface between SDN controller and virtual switch.

2.3.2. the interface between SDN controller and VNFM

NETCONF/RESTCON/REST API. Yang model extension is defined by vendor, REST API is vendor specific.

2.3.3. the interface between SDN controller and VNF

NETCONF/RESTCONF/REST API. Yang model extension is defined by vendor, REST API is vendor specific.

3. The operation of the SDN controller and VNFs in the current deployment

Create a Router

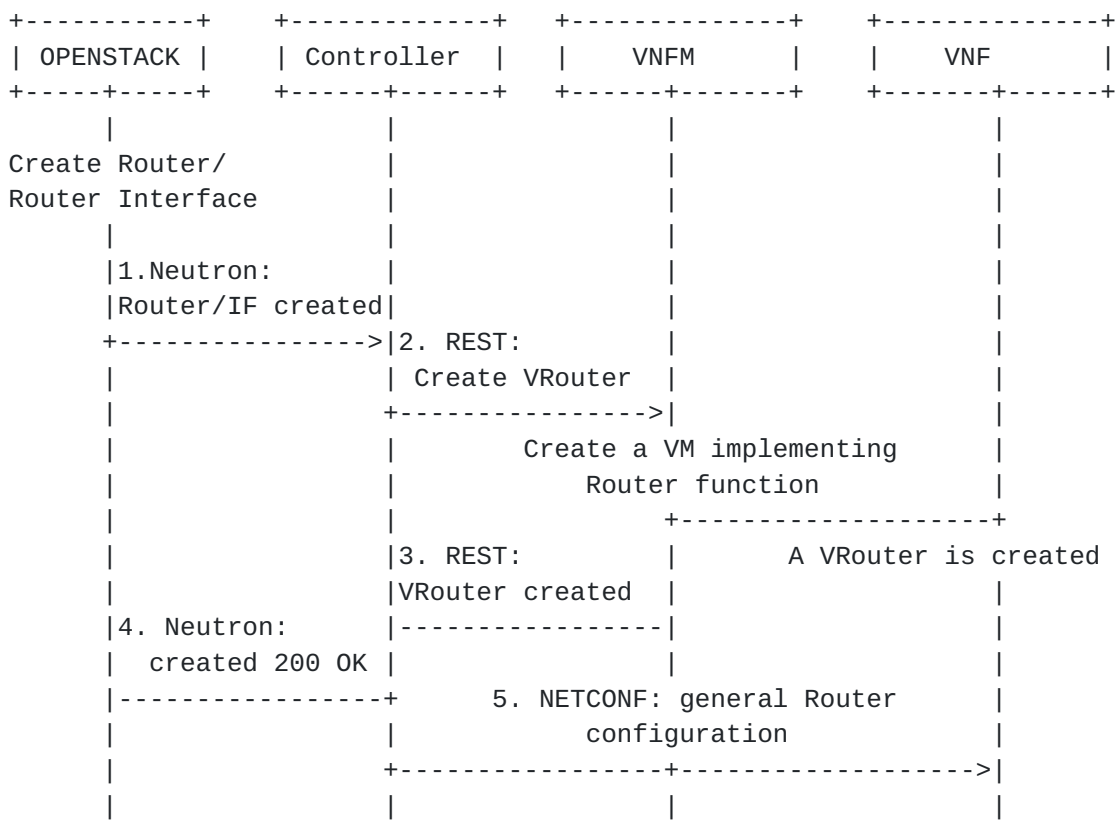


Figure 2: Create a router

- When a "Router/Router Interface" is created in the OPENSTACK, Neutron notifies the SDN controller.
- SDN controller sends the virtual Router created event to the VNFM via the REST API.
- The VNFM creates a VM which implements the Router function.
- The VNFM notifies the SDN controller that a virtual Router is created successfully, at the meanwhile, it sends the information of the virtual Router (including IP address, login username/ password) to SDN controller as the feedback.
- The SDN controller responds Router/Router Interface created successfully to the OPENSTACK.
- The SDN controller connects to the created virtual Router and sets the general configuration to the virtual Router via NETCONF. The configuration includes Router interface configuration, vlan/vxlan configuration, etc.

Create a VM

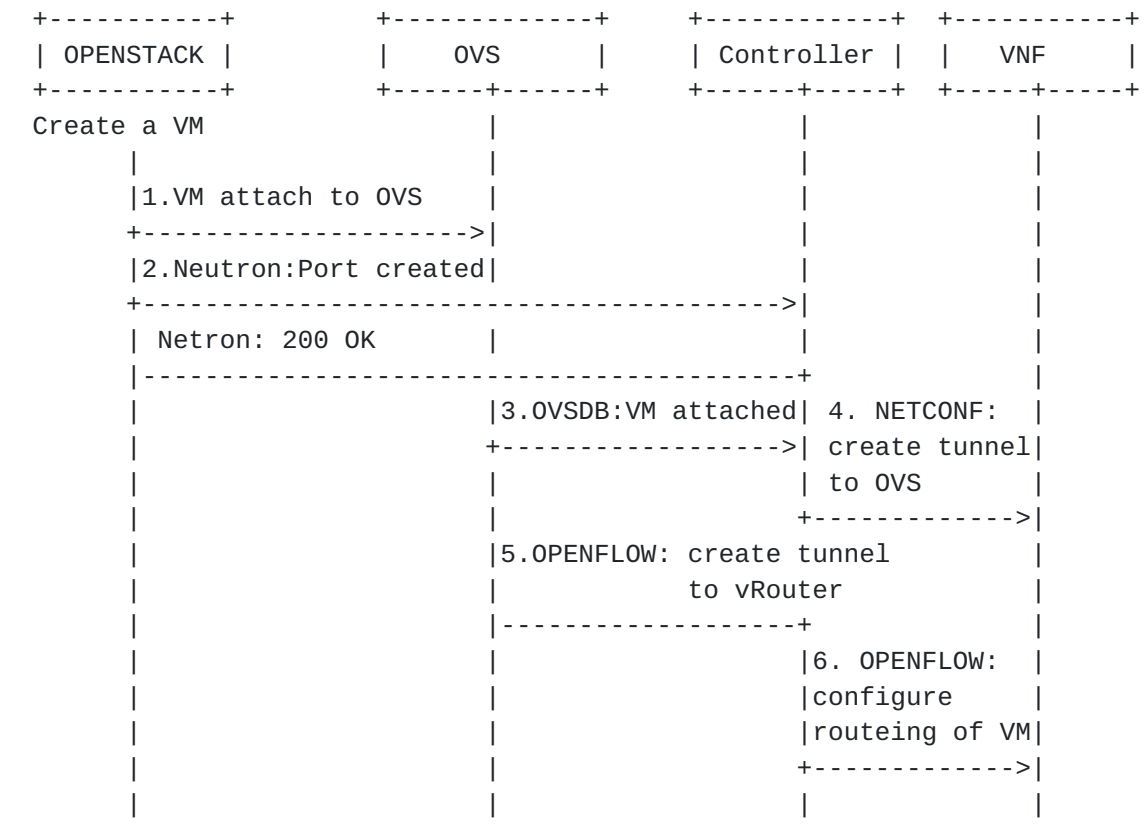
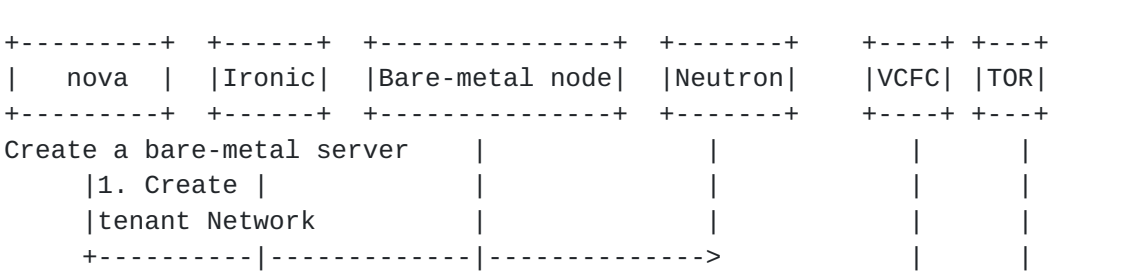
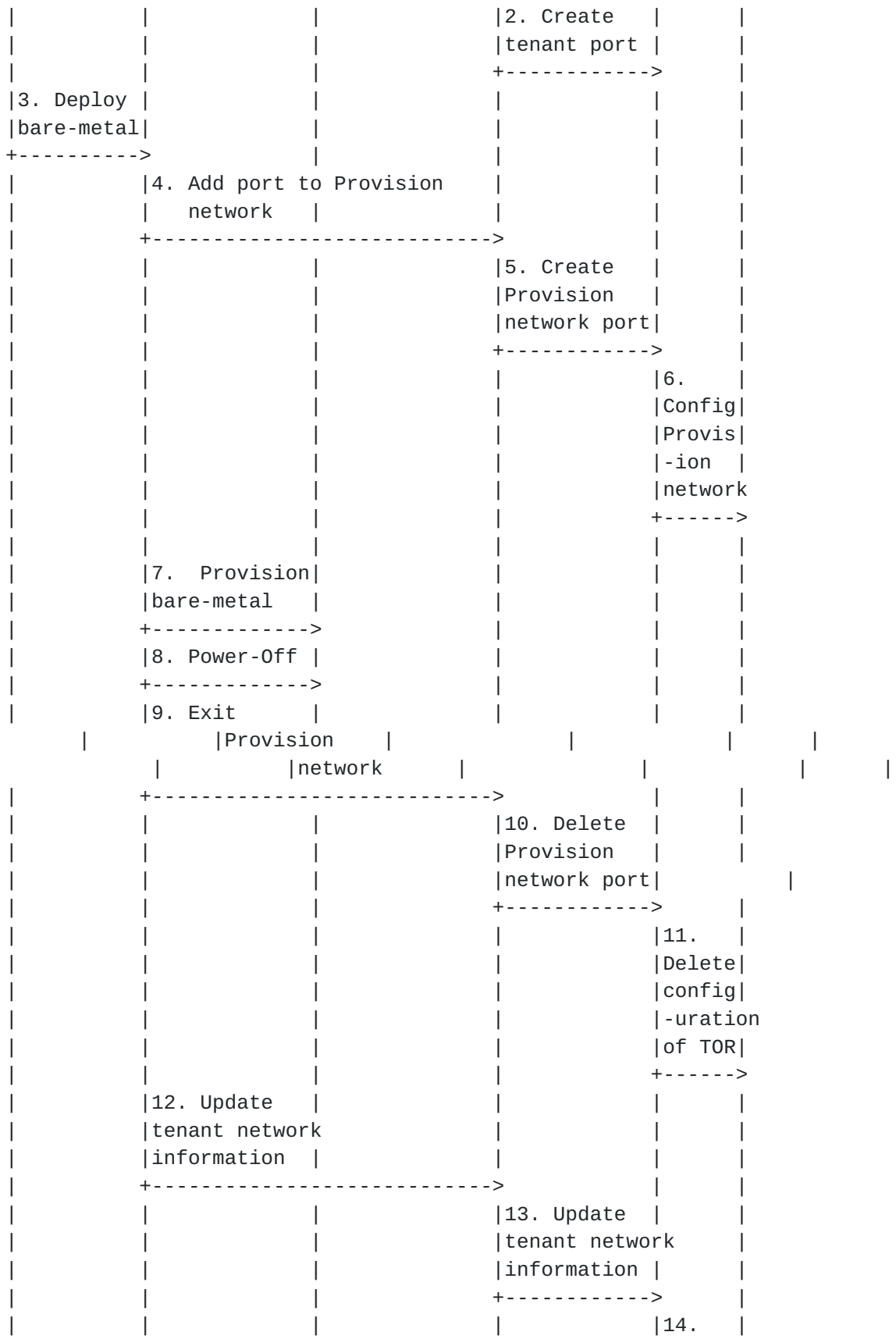


Figure 3: Create a VM

- When a VM is created in the OPENSTACK, the Nova attaches the VM to the OVS via OVSDB. Then Neutron informs the SDN controller that a port related to the created VM is created.
- The OVS notifies SDN controller that a VM is attached. At this moment, SDN controller knows that a tenant VM is created and which OVS the VM attaches to.
- The SDN controller instructs the virtual Router to create a vxlan tunnel towards the OVS via NETCONF.
- The SDN controller instructs the OVS to create a vxlan tunnel towards the virtual Router via OPENFLOW.
- The SDN controller configures the routing of VM in the virtual Router via OPENFLOW.

Create a Baremetal Server




2. Create tenant port
3. Deploy bare-metal
4. Add port to Provision network
5. Create Provision network port
6. Config Provis -ion network
7. Provision bare-metal
8. Power-Off
9. Exit
Provision network
10. Delete Provision network port
11. Delete config -uration of TOR
12. Update tenant network information
13. Update tenant network information
14.

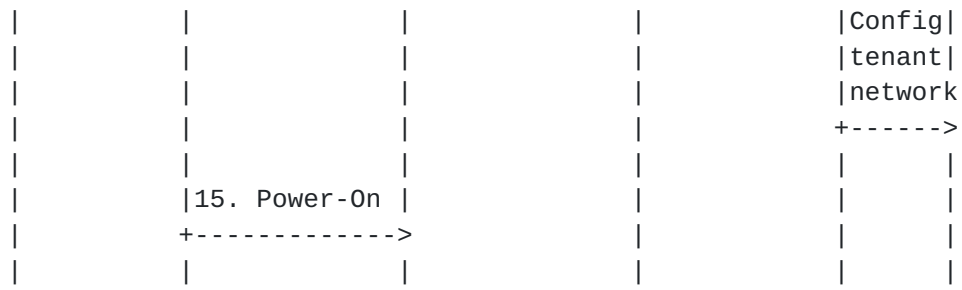


Figure 4: Create a baremetal server

- When a VM is created in the OPENSTACK, the Nova attaches the VM to the OVS via OVSDB. Then Neutron informs the SDN controller that a port related to the created VM is created.
- The OVS notifies SDN controller that a VM is attached. At this moment, SDN controller knows that a tenant VM is created and which OVS the VM attaches to.
- The SDN controller instructs the virtual Router to create a vxlan tunnel towards the OVS via NETCONF.
- The SDN controller instructs the OVS to create a vxlan tunnel towards the virtual Router via OPENFLOW.
- The SDN controller configures the routing of VM in the virtual Router via OPENFLOW.

4. Combinations of SDN controller and VNFs

There are three possible ways of combining SDN controller and VNFs

Option 1: Openstack -> SDN controller -> VNF

OPENSTACK sends all events to the SDN controller via Neutron. If the received events are related to the OVS (such as create security group /create port), the SDN controller sets the flow to OVS via OPENFLOW. On the other hand, if the received events are related to the VNF (such as create firewall/loadbalance), SDN controller interworks with VNFM to handle this events. In this case, the SDN controller only needs to interwork with VNFM. Since SDN controller and VNF/VNFM are often provided by different vendors, this way of combination makes the cooperation of SDN controller and VNFM simple.

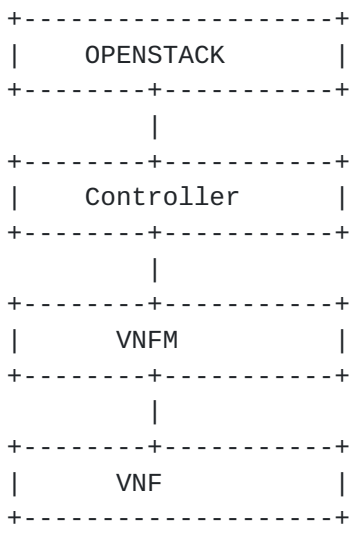


Figure 5: Openstack -> SDN controller -> VNF

Note that in our current deployment, the combination of SDN controller and VNFM/VNF is slightly different from the Option 1. The SDN controller invokes the interface of VNFM to create/terminate the VNF, while talks to the VNF directly to configure it. This way of combination is sub-optimized since the SDN controller not only needs to cooperate with VNFM but also needs to interwork with VNF, which increases the complexity of the entire system and the interface between SDN controller and VNFM/VNF. We will move to the Option 1 in our next stage deployment.

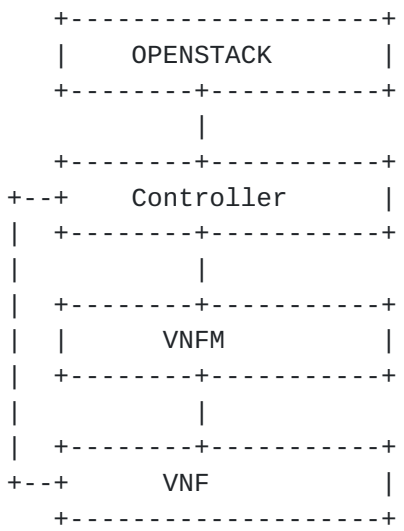


Figure 6: Openstack -> SDN controller -> VNF ->SDN controller

Option 2: Openstack -> VNF -> SDN controller

OPENSTACK sends all events to the VNFM via Neutron. If the received events are related to the VNF (such as create firewall/loadbalance), the VNFM create/configure the VNF through internal interface. At the same time, VNFM notifies the SDN controller of this event. SDN controller then configure the routing in the OVS, such as creating tunnel to the created firewall. On the other hand, if the received events are related to OVS (such as create security group /create port), the VNFM forwards this event to the SDN controller.

This way of combination is more complex than Option1. VNFM in this Option not only needs to handle the VNF related events but also needs to handle the OVS related events, which makes the VNFM more complex and inefficient (since handling the OVS related events makes no sense to VNFM).

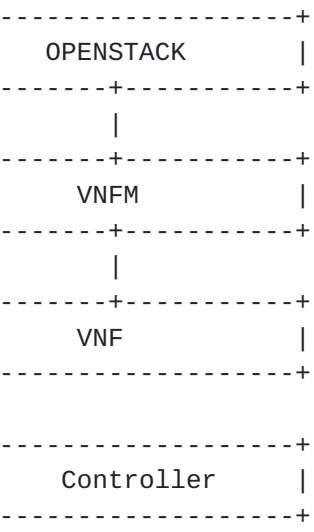


Figure 7: Openstack -> VNF -> SDN controller

Option 3: OPENSTACK -> 1. SDN controller 2. VNFM

OPENSTACK (actually the plugin provided to the OPENSTACK) needs to sends different events to different entities. OPENSATCK sends the OVS related events to the SDN controller while sends the VNF related events to the VNFM. Besides, the SDN controller needs to interact with VNFM. For example, when receiving the Router/Router Interface created event, the VNFM needs to notify the SDN controller, which then configure the routing in the OVS, such as creating tunnel to the created virtual Router.

This way of combination is more complex. OPENSTACK needs to send different events to different entities. Since the SDN controller and VNFM are often provided by different vendors, Cooperation of SDN controller and VNFM in the OPENSTACK is hard to be achieved.

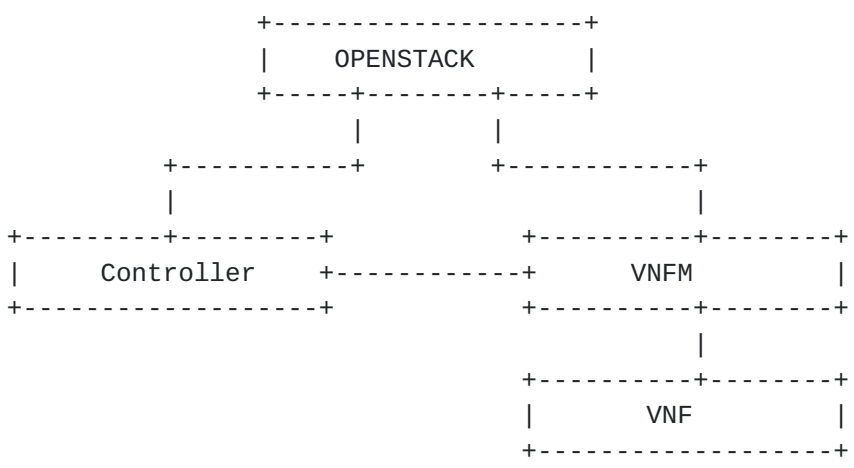


Figure 8: OPENSTACK -> 1. SDN controller 2. VNFM

5. Security Considerations

TBA

6. IANA Considerations

None.

7. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

Authors' Addresses

Jinzhu Wang
China Mobile

Email: wangjinzhu@chinamobile.com

Rong Gu
China Mobile

Email: gurong@chinamobile.com