NetReady: Network Readiness

Release draft (b4f736d)

OPNFV

May 17, 2016

Project NetReady, https://wiki.opnfv.org/display/netready/NetReady

Editors TBD

Authors Georg Kunz (Ericsson), <add your name here>

Abstract OPNFV provides an infrastructure with different SDN controller options to realize NFV functionality on the platform it builds. As OPNFV uses OpenStack as VIM, we need to analyze the capabilities this component offers us. The networking functionality is provided by a single component called Neutron, which hides the controller under it, let it be Neutron itself or any supported SDN controller. As NFV wasn't taken into consideration at the time when Neutron was designed we are already facing several bottlenecks and architectural shortcomings while implementing our use cases.

The NetReady project aims at evolving OpenStack networking step-by-step to find the most efficient way to fulfill the requirements of the identified NFV use cases, taking into account the NFV mindset and the capabilities of SDN controllers.

History

Date	Description
22.03.2016	Project creation
19.04.2016	Initial version of the deliverable uploaded to Gerrit

Definition of terms

Different SDOs and communities use different terminology related to NFV/Cloud/SDN. This list tries to define an OPNFV terminology, mapping/translating the OPNFV terms to terminology used in other contexts.

API Application Programming Interface.

Cloud Computing A model that enables access to a shared pool of configurable computing resources, such as networks, servers, storage, applications, and services, that can be rapidly provisioned and released with minimal management effort or service provider interaction.

Edge Computing Edge computing pushes applications, data and computing power (services) away from centralized points to the logical extremes of a network.

Instance Refers in OpenStack terminology to a running VM, or a VM in a known state such as suspended, that can be used like a hardware server.

NFV Network Function Virtualization.

NFVI Network Function Virtualization Infrastructure. Totality of all hardware and software components which build up the environment in which VNFs are deployed.

SDN Software-Defined Networking. Emerging architecture that decouples the network control and forwarding functions, enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services.

Server Computer that provides explicit services to the client software running on that system, often managing a variety of computer operations. In OpenStack terminology, a server is a VM instance.

VIM Virtualized Infrastructure Manager. Functional block that is responsible for controlling and managing the NFVI compute, storage and network resources, usually within one operator's Infrastructure Domain, e.g. NFVI Point of Presence (NFVI-PoP).

VM Virtual Machine. Virtualized computation environment that behaves like a physical computer/server by (re-)creating the computing architecture of a real or hypothetical computer.

Virtual network Virtual network routes information among the network interfaces of VM instances and physical network interfaces, providing the necessary connectivity.

VNF Virtualized Network Function. Implementation of an Network Function that can be deployed on a Network Function Virtualization Infrastructure (NFVI).

WAN Wide Area Network.

CHAPTER

ONE

INTRODUCTION

This document represents and describes the results of the OPNFV NetReady (Network Readiness) project. Specifically, the document comprises a selection of NFV-related networking use cases and their networking requirements, a corresponding gap analysis of the aforementioned requirements with respect to the current OpenStack networking architecture and a description of potential solutions and improvements.

1.1 Scope

NetReady is a project within the OPNFV initiative. Its focus is on NFV (Network Function Virtualization) related networking use cases and their requirements on the underlying NFVI (Network Function Virtualization Infrastructure).

The NetReady project addresses the OpenStack networking architecture, specifically OpenStack Neutron, from a NFV perspective. Its goal is to identify gaps in the current OpenStack networking architecture with respect to NFV requirements and to propose and evaluate improvements and potential complementary solutions.

1.2 Problem Description

Traditionally, OpenStack networking, represented typically by the OpenStack Neutron project, targets virtualized data center networking. This comprises primarily establishing and managing layer 2 network connectivity among VMs (Virtual Machines). Over the past releases of OpenStack, Neutron has accumulated an extensive feature set, covering L2 and L3 networking features such as <list of examples>.

It is often stated that NFV imposes additional requirements on the networking architecture and feature set of the underlying NFVI beyond those of data center networking. For instance, the NFVI needs to establish and manage connectivity beyond the data center to the WAN (Wide Area Network). Moreover, NFV networking use cases often abstract from L2 connectivity and for example focus on L3-only connectivity. Hence, the NFVI networking architecture needs to be flexible enough to be able to meet the requirements of NFV-related use cases in addition to traditional data center networking.

It is an ongoing debate how well the current OpenStack networking architecure can meet the additional requirements of NFV networking. Hence, a thorough analysis of NFV networking requirements and their relation to the OpenStack networking architecture is needed.

1.3 Goals

The goals of the NetReady project and correspondingly this document are the following:

- This document comprises a collection of relevant NFV networking use cases and clearly describes their requirements on the NFVI. These requirements are stated independently of a particular implementation, for instance

OpenStack Neutron. Instead, requirements are formulated in terms of APIs (Application Programming Interfaces) and data models needed to realize a given NFV use case.

- The list of use cases is not considered to be all-emcompassing but it represents a carefully selected set of use cases that are considered to be relevant at the time of writing. More use cases may be added over time.
- This document contains a thorough analysis of the gaps in the current OpenStack networking architecture with respect to the requirements imposed by the selected NFV use cases.
- This document describes the proposed improvements and complementary solutions needed to enable OpenStack to fulfill the identified NFV requirements.

CHAPTER

TWO

USE CASES

The following sections address networking use cases that have been identified to be relevant in the scope of NFV and NetReady.

2.1 L3VPN Use Cases

Service Providers' virtualized network infrastructure may consist of one or more SDN Controllers from different vendors. Those SDN Controllers may be managed within one cloud or multiple clouds. Jointly, those VIMs (e.g. OpenStack instances) and SDN Controllers work together in an interoperable framework to create L3 services in Service Providers' virtualized network infrastructure.

Three use cases of creating L3VPN service by multiple SDN Controllers are described as follows.

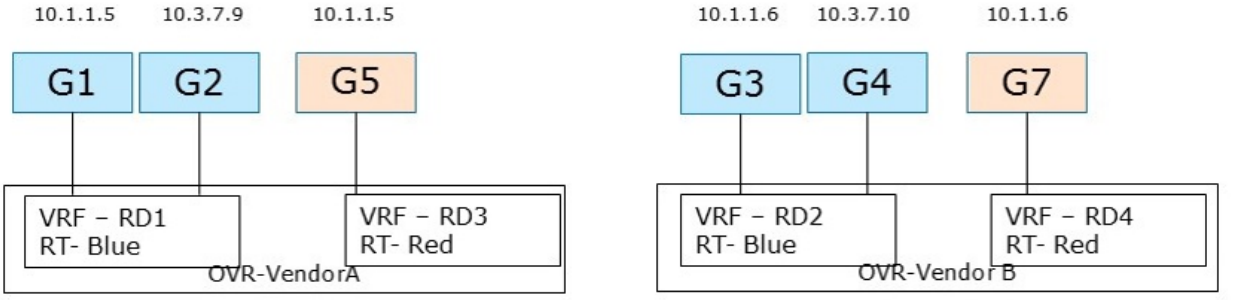
2.1.1 Any-to-Any Base Case

Description

There are 2 hosts (compute nodes). SDN Controller A and vRouter A are provided by Vendor A, and run on host A. SDN Controller B and vRouter B are provided by Vendor B, and run on host B.

There are 2 tenants. Tenant 1 creates L3VPN Blue with 2 subnets: 10.1.1.0/24 and 10.3.7.0/24. Tenant 2 creates L3VPN Red with 1 subnet, overlapping address space: 10.1.1.0/24.

The network topology is shown in Fig. 2.5.1:



In L3VPN Blue, VMs G1 (10.1.1.5) and G2 (10.3.7.9) are spawned on host A, and attached to 2 subnets (10.1.1.0/24 and 10.3.7.0/24) and assigned IP addresses respectively. VMs G3 (10.1.1.6) and G4 (10.3.7.10) are spawned on host B, and attached to 2 subnets (10.1.1.0/24 and 10.3.7.0/24) and assigned IP addresses respectively.

In L3VPN Red, VM G5 (10.1.1.5) is spawned on host A, and attached to subnet 10.1.1.0/24. VM G6 (10.1.1.6) is spawned on host B, and attached to the same subnet 10.1.1.0/24.

Exemplary workflow is described as follows:

1. Create Network

2. Create Network VRF Policy Resource `Any-to-Any` 2.1. This sets up that when this tenant is put on a HOST that: 2.1.1. There will be a RD assigned per VRF 2.1.2. There will be a RT used for the common any-to-any communication 3. Create Subnet 4. Create Port (subnet, network vrf policy resource). This causes controller to: 4.1. Create vrf in vRouter's FIB, or Update vrf if already exists 4.2. Install an entry for Guest's HOST-Route in FIBs of Vrouters serving this tenant Virtual Network 4.3. Announce Guest HOST-Route to WAN-GW via MP-BGP

VRF Lets us do: 1. Overlapping Addresses 2. Segregation of Traffic

Derrived Requirements

- TBD

Northbound API / Workflow

- TBD

Data model objects

- TBD

Orchestration

- TBD

Dependencies on compute services

- TBD

Potential implementation

- TBD

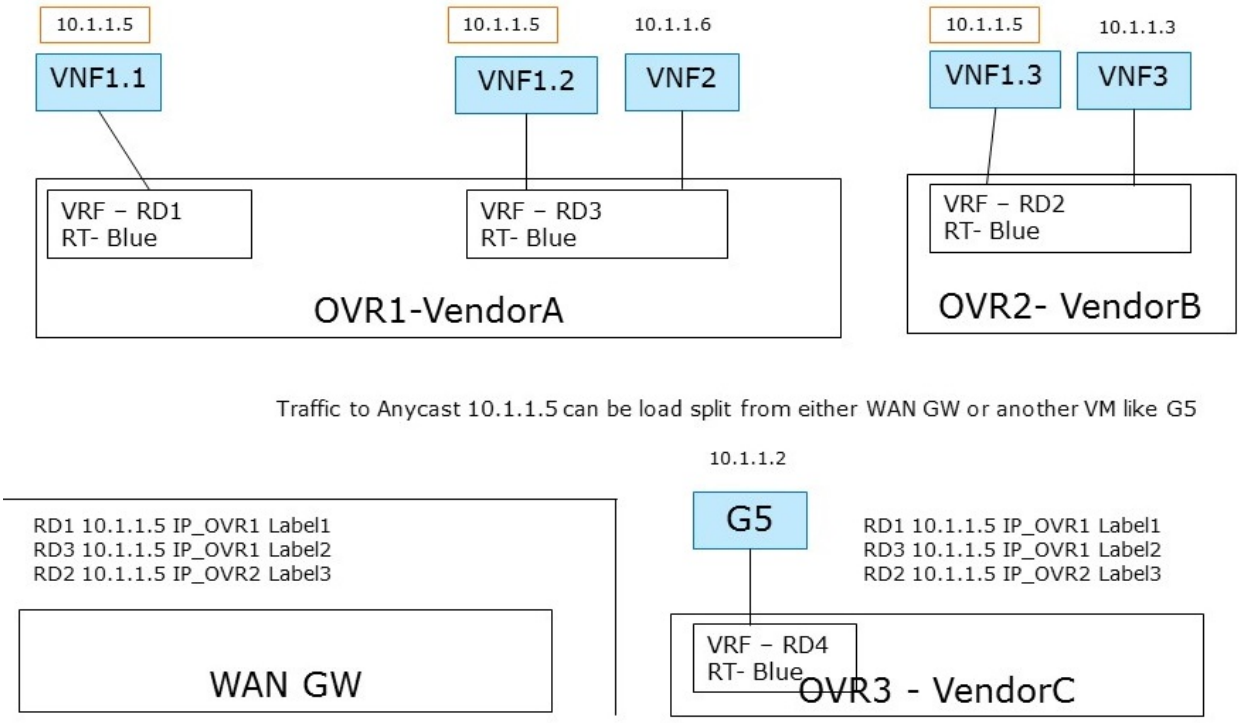
2.1.2 ECMP Load Splitting Case (Anycast)

Description

There are 2 hosts (compute nodes). SDN Controller A and vRouter A are provided by Vendor A, and run on host A. SDN Controller B and vRouter B are provided by Vendor B, and run on host B.

There is 1 tenant. Tenant 1 creates L3VPN Blue with subnet 10.1.1.0/24.

The network topology is shown in Fig. 2.5.2:



In L3VPN Blue, VNF1.1 and VNF1.2 are spawned on host A, attached to subnet 10.1.1.0/24 and assigned the same IP address 10.1.1.5. VNF1.3 is spawned on host B, attached to subnet 10.1.1.0/24 and assigned the same IP addresses 10.1.1.5. VNF 2 and VNF 3 are spawned on host A and B respectively, attached to subnet 10.1.1.0/24, and assigned different IP addresses 10.1.1.6 and 10.1.1.3 respectively.

Here, the Network VRF Policy Resource is `ECMP/AnyCast`. Traffic to **Anycast 10.1.1.5** can be load split from either WAN GW or another VM like G5.

Derrived Requirements

- TBD

Northbound API / Workflow

- TBD

Data model objects

- TBD

Orchestration

- TBD

Dependencies on compute services

- TBD

Potential implementation

- TBD

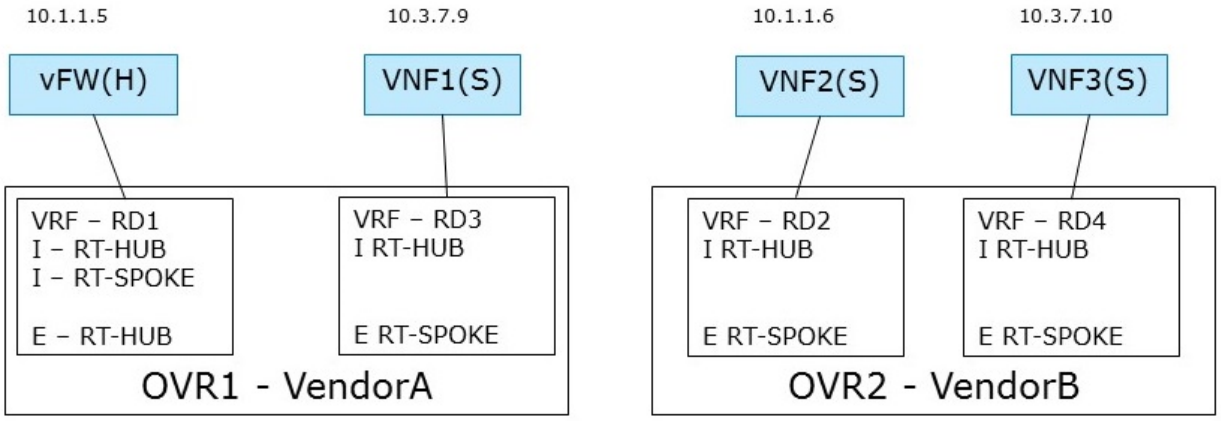
2.1.3 Hub and Spoke Case

Description

There are 2 hosts (compute nodes). SDN Controller A and vRouter A are provided by Vendor A, and run on host A. SDN Controller B and vRouter B are provided by Vendor B, and run on host B.

There is 1 tenant. Tenant 1 creates L3VPN Blue with 2 subnets: 10.1.1.0/24 and 10.3.7.0/24.

The network topology is shown in Fig. 2.5.3:



In L3VPN Blue, vFW(H) is acting the role of `hub` (a virtual firewall). The other 3 VNFsVMs are `spoke`. vFW(H) and VNF1(S) are spawned on host A, and VNF2(S) and VNF3(S) are spawned on host B. vFW(H) (10.1.1.5) and VNF2(S) (10.1.1.6) are attached to subnet 10.1.1.0/24. VNF1(S) (10.3.7.9) and VNF3(S) (10.3.7.10) are attached to subnet 10.3.7.0/24.

Exemplary vFW(H) Hub VRF is as follows:

- RD1 10.1.1.5 IP_OVR1 Label1
- RD1 0/0 IP_OVR1 Label1
- Label 1 Local IF (10.1.1.5)
- RD3 10.3.7.9 IP_OVR1 Label2
- RD2 10.1.1.6 IP_OVR2 Label3
- RD4 10.3.7.10 IP_OVR2 Label3

Exemplary VNF1(S) Spoke VRF is as follows:

- RD1 0/0 IP_OVR1 Label1

- RD3 10.3.7.9 IP_OVR1 Label2

Exemplary workflow is described as follows:

1. Create Network

2. Create VRF Policy Resource 2.1. Hub and Spoke 3. Create Subnet 4. Create Port 4.1. Subnet 4.2. VRF Policy Resource, [H | S]

Derrived Requirements

- TBD

Northbound API / Workflow

- TBD

Data model objects

- TBD

Orchestration

- TBD

Dependencies on compute services

- TBD

Potential implementation

- TBD

2.2 Shared Service Functions

2.2.1 Description

This use case aims at binding multiple networks or network services to a single vNIC (port) of a given VM. There are several specific application scenarios for this use case:

- Shared Service Functions: A service function connects to multiple (tenant) networks by means of a single vNIC.

  Typically, a vNIC is bound to a single (tenant) network. Hence, in order to directly connect a service function to multiple (tenant) networks at the same time, multiple vNICs are needed - each vNIC binding the service function to a separate network. For service functions requiring connectivity to a large number of networks, this approach does not scale as the number of vNICs per VM is limited and additional vNICs occupy additional resources on the hypervisor.

  A more scalable approach is to bind multiple networks to a single vNIC and let the service function, which is now shared among multiple networks, handle the separation of traffic itself.

- Multiple network services: A service function connects to multiple different network types such as a L2 network, a L3(-VPN) network, a SFC domain or services such as DHCP, IPv6 NDP, firewall/security, etc.

In order to achieve a flexible binding of multiple services to vNICs, a logical separation between a vNIC (instance port) - that is, the entity that is used by the compute service as hand-off point between the network and the VM - and a service interface - that is, the interface a service binds to - is needed.

Furthermore, binding network services to service interfaces instead of to the vNIC directly enables a more dynamic management of the network connectivity of network functions as there is no need to add or remove vNICs.

2.2.2 Requirements

Northbound API

The API has to cover the following functionality:

- management of instance ports
- management of service ports
- binding of services to service ports

Data models

TBD

Orchestration

None.

Dependencies on other resources

The compute service needs to be enabled to consume instance ports instead of classic Neutron ports.

2.2.3 Implementation Proposal

TBD

2.3 Programmable Provisioning of Provider networks

2.3.1 Description

In NFV environment the VNFM (consumer of OpenStack IaaS API) have no administrative rights, however in this environment provider networks are used in some cases. When a provider network is ceated administrative rights are needed what in the case of non admin VNFM leeds to manual work. It shall be possible to configure provider networks without administrative rights. It should be possible to assign the capability to create provider networks to any roles.

2.3.2 Derrived Requirements

- Possibility to assign the property of provider networks to any role
- When the provider network is created it should be checked if the role of the user has the permission to create a provider network.

Northbound API / Workflow

- TBD

Data model objects

- TBD

Orchestration

- TBD

Dependencies on compute services

- TBD

Potential implementation

- TBD

2.4 Georedundancy Use cases (Draft)

2.4.1 Connection between different OpenStack cells

Description

There should be an API to manage the infrastructure-s networks between two OpenStack cells. (Note: In the Mitaka release of OpenStack cells v1 are considered as, cells v2 functionaity is under implementation)

Derrived Requirements

- Possibility to define a remote and a local endpoint
- Possiblity to define an overlay/segregation technology
- As in case of cells the nova-api service is shared it should be possible to identify the cell in the API calls

Northbound API / Workflow

- An infrastructure network management API is needed
- When the endpoints are created neutron is configured to use the new network. (Note: Nova networking is not considered as it is deprecated.)

Data model objects

- TBD

Orchestration

- TBD

Dependencies on compute services

- TBD

Potential implementation

- TBD

2.4.2 Connection between different OpenStack regions or cloud instances

Description

There should be an API to manage the infrastructure-s networks between two OpenStack regions or between two OpenStack cloud instances. (The only difference is the shared keystone in case of a region)

Derrived Requirements

- Possibility to define a remote and a local endpoint
- Possiblity to define an overlay/segregation technology

Northbound API / Workflow

- An infrastructure network management API is needed
- When the endpoints are created neutron is configured to use the new network. (Note: Nova networking is not considered as it is deprecated.)

Data model objects

- TBD

Orchestration

- TBD

Dependencies on compute services

- TBD

Potential implementation

- TBD

2.5 L3VPN Use Cases

Service Providers' virtualized network infrastructure may consist of one or more SDN Controllers from different vendors. Those SDN Controllers may be managed within one cloud or multiple clouds. Jointly, those VIMs (e.g. OpenStack instances) and SDN Controllers work together in an interoperable framework to create L3 services in Service Providers' virtualized network infrastructure.

Three use cases of creating L3VPN service by multiple SDN Controllers are described as follows.

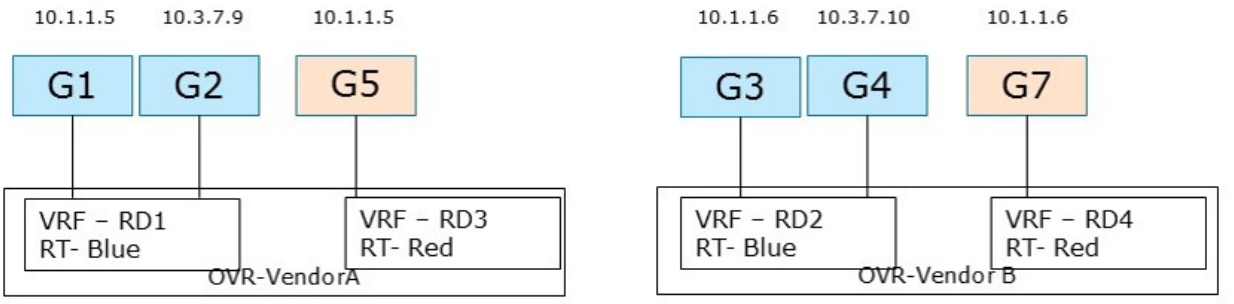
2.5.1 Any-to-Any Base Case

Description

There are 2 hosts (compute nodes). SDN Controller A and vRouter A are provided by Vendor A, and run on host A. SDN Controller B and vRouter B are provided by Vendor B, and run on host B.

There are 2 tenants. Tenant 1 creates L3VPN Blue with 2 subnets: 10.1.1.0/24 and 10.3.7.0/24. Tenant 2 creates L3VPN Red with 1 subnet, overlapping address space: 10.1.1.0/24.

The network topology is shown in Fig. 2.5.1:



In L3VPN Blue, VMs G1 (10.1.1.5) and G2 (10.3.7.9) are spawned on host A, and attached to 2 subnets (10.1.1.0/24 and 10.3.7.0/24) and assigned IP addresses respectively. VMs G3 (10.1.1.6) and G4 (10.3.7.10) are spawned on host B, and attached to 2 subnets (10.1.1.0/24 and 10.3.7.0/24) and assigned IP addresses respectively.

In L3VPN Red, VM G5 (10.1.1.5) is spawned on host A, and attached to subnet 10.1.1.0/24. VM G6 (10.1.1.6) is spawned on host B, and attached to the same subnet 10.1.1.0/24.

Exemplary workflow is described as follows:

1. Create Network

2. Create Network VRF Policy Resource `Any-to-Any` 2.1. This sets up that when this tenant is put on a HOST that: 2.1.1. There will be a RD assigned per VRF 2.1.2. There will be a RT used for the common any-to-any communication 3. Create Subnet 4. Create Port (subnet, network vrf policy resource). This causes controller to: 4.1. Create vrf in vRouter's FIB, or Update vrf if already exists 4.2. Install an entry for Guest's HOST-Route in FIBs of Vrouters serving this tenant Virtual Network 4.3. Announce Guest HOST-Route to WAN-GW via MP-BGP

VRF Lets us do: 1. Overlapping Addresses 2. Segregation of Traffic

Derrived Requirements

- TBD

Northbound API / Workflow

- TBD

Data model objects

- TBD

Orchestration

- TBD

Dependencies on compute services

- TBD

Potential implementation

- TBD

2.5.2 ECMP Load Splitting Case (Anycast)

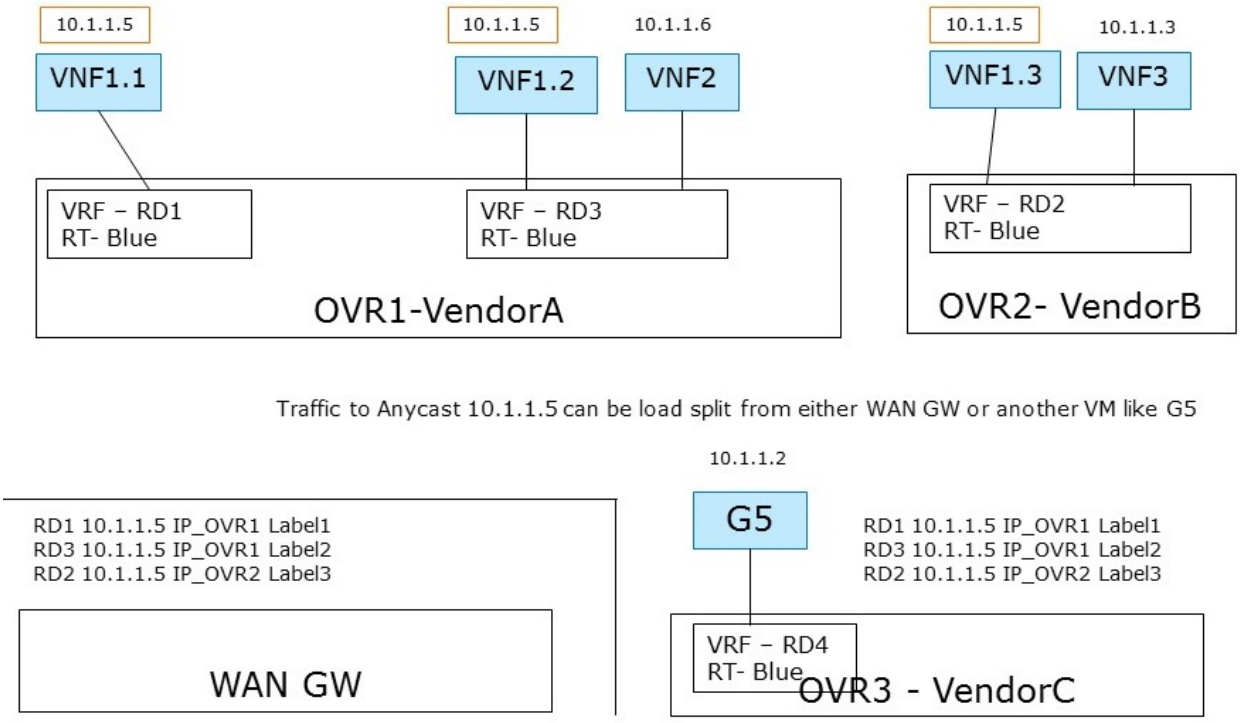
Description

There are 2 hosts (compute nodes). SDN Controller A and vRouter A are provided by Vendor A, and run on host A. SDN Controller B and vRouter B are provided by Vendor B, and run on host B.

There is 1 tenant. Tenant 1 creates L3VPN Blue with subnet 10.1.1.0/24.

The network topology is shown in Fig. 2.5.2:

In L3VPN Blue, VNF1.1 and VNF1.2 are spawned on host A, attached to subnet 10.1.1.0/24 and assigned the same IP address 10.1.1.5. VNF1.3 is spawned on host B, attached to subnet 10.1.1.0/24 and assigned the same IP addresses 10.1.1.5. VNF 2 and VNF 3 are spawned on host A and B respectively, attached to subnet 10.1.1.0/24, and assigned different IP addresses 10.1.1.6 and 10.1.1.3 respectively.



Here, the Network VRF Policy Resource is `ECMP/AnyCast`. Traffic to **Anycast 10.1.1.5** can be load split from either WAN GW or another VM like G5.

Derrived Requirements

- TBD

Northbound API / Workflow

- TBD

Data model objects

- TBD

Orchestration

- TBD

Dependencies on compute services

- TBD

Potential implementation

- TBD

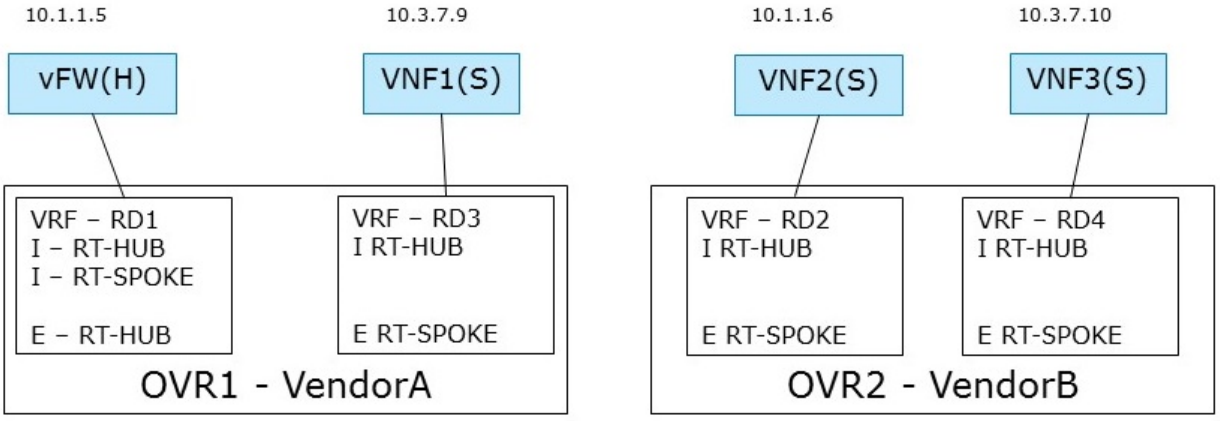
2.5.3 Hub and Spoke Case

Description

There are 2 hosts (compute nodes). SDN Controller A and vRouter A are provided by Vendor A, and run on host A. SDN Controller B and vRouter B are provided by Vendor B, and run on host B.

There is 1 tenant. Tenant 1 creates L3VPN Blue with 2 subnets: 10.1.1.0/24 and 10.3.7.0/24.

The network topology is shown in Fig. 2.5.3:



In L3VPN Blue, vFW(H) is acting the role of `hub` (a virtual firewall). The other 3 VNFsVMs are `spoke`. vFW(H) and VNF1(S) are spawned on host A, and VNF2(S) and VNF3(S) are spawned on host B. vFW(H) (10.1.1.5) and VNF2(S) (10.1.1.6) are attached to subnet 10.1.1.0/24. VNF1(S) (10.3.7.9) and VNF3(S) (10.3.7.10) are attached to subnet 10.3.7.0/24.

Exemplary vFW(H) Hub VRF is as follows:

- RD1 10.1.1.5 IP_OVR1 Label1
- RD1 0/0 IP_OVR1 Label1
- Label 1 Local IF (10.1.1.5)
- RD3 10.3.7.9 IP_OVR1 Label2
- RD2 10.1.1.6 IP_OVR2 Label3
- RD4 10.3.7.10 IP_OVR2 Label3

Exemplary VNF1(S) Spoke VRF is as follows:

- RD1 0/0 IP_OVR1 Label1
- RD3 10.3.7.9 IP_OVR1 Label2

Exemplary workflow is described as follows:

1. Create Network

2. Create VRF Policy Resource 2.1. Hub and Spoke 3. Create Subnet 4. Create Port 4.1. Subnet 4.2. VRF Policy Resource, [H | S]

Derrived Requirements

- TBD

Northbound API / Workflow

- TBD

Data model objects

- TBD

Orchestration

- TBD

Dependencies on compute services

- TBD

Potential implementation

- TBD

CHAPTER

THREE

CURRENT SOLUTIONS

TBD

CHAPTER

FOUR

GAP ANALYSIS

TBD

CHAPTER

FIVE

ARCHITECTURE

TBD

CHAPTER

SIX

IMPLEMENTATION

TBD

CHAPTER

SEVEN

SUMMARY

TBD