**Technical white paper**

# NFV solution (VLAN, VXLAN provisioning, Firewall Service) for VMWare ESX networks

## OVSvApp solution for ESX networks

# Table of contents

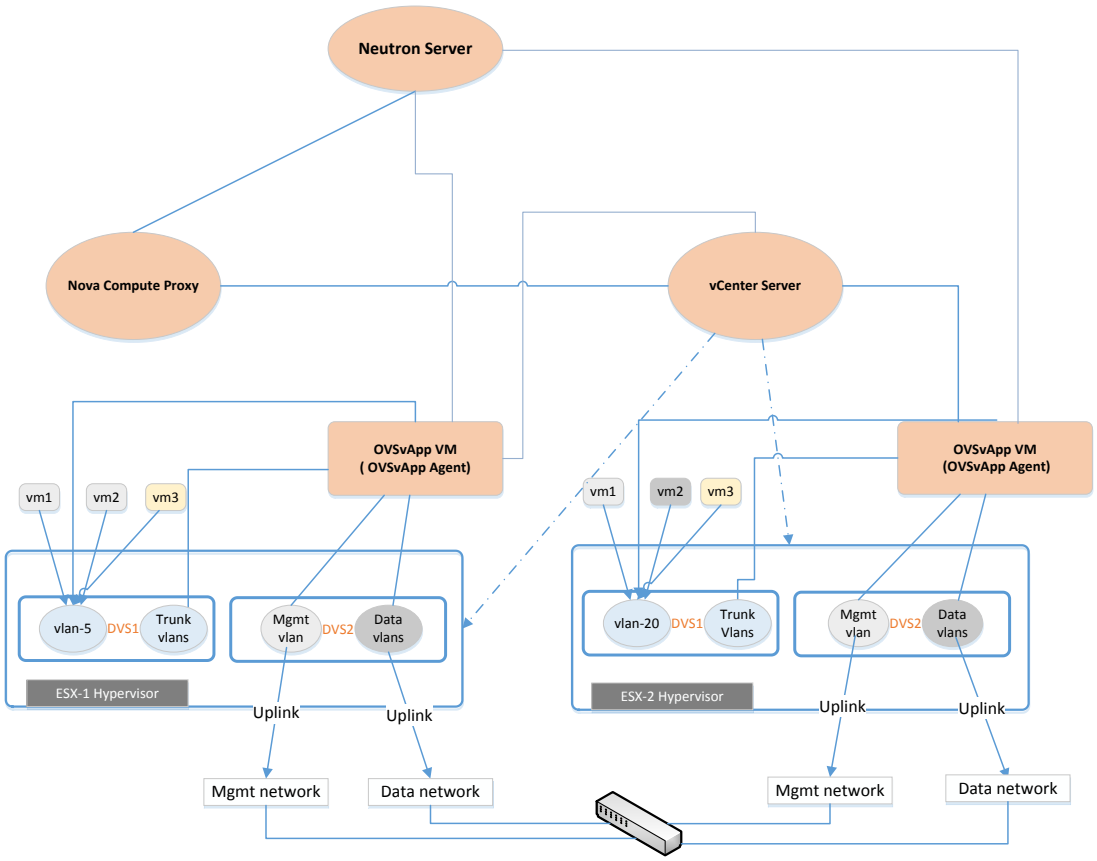
# Introduction

This document outlines Network Function Virtualization (NFV) achieved via a virtual machine mounted on an ESX host in VMWare environments similar to NFV achieved on Openstack KVM compute nodes. It explains how cloud operators are provided with a Neutron supported solution for vSphere deployments in the form of a service VM called OVSvApp VM which steers the ESX tenant VM's traffic through it.

OVSvApp VM renders network connectivity to tenant VMs using network virtualization technologies namely VLAN, VXLAN (Non multicast, $2^{24}$ networks) and firewall services in the form of Security Groups, where traffic filtering rules are applied to tenant VM's networking.

# Architecture

**Figure 1.** OVSvApp Solution – Architecture Diagram



OVSvApp solution comprises of a VM called OVSvApp VM hosted on each ESX hypervisor within a cluster and 2 Distributed Virtual Switches (DVS). OVSvApp VM runs Ubuntu or hLinux as a guest Operating System and has Open vSwitch 2.1.0 installed. It also runs an agent called OVSvApp agent.

For VLAN provisioning, 2 DVS per datacenter is required and for VXLAN, 2 DVS per cluster is required. The 1st DVS do not need any uplinks implying no external network connectivity, but will provide connectivity to tenant VMs and OVSvApp VM. Each tenant VM is associated with a portgroup (VLAN). The tenant VMs' data traffic reaches the OVSvApp VM via their respective portgroup and hence through another portgroup called **TrunkPortgroup** (Portgroup defined with "**VLAN type**" as "**VLAN Trunking**" and "**VLAN trunk range**" set with range of tenant VMs' traffic - VLAN ranges, exclusive of management VLAN as explained below) with which OVSvApp VM is associated.

The 2nd DVS has 1 or 2 uplinks and provides management and data connectivity to OVSvApp VM. The OVSvApp VM is also associated with other 2 portgroups namely **ManagementPortgroup** (Portgroup defined with "**VLAN type**" as "**None**" OR "**VLAN**" with specific VLAN Id in "**VLAN ID**" OR "**VLAN Trunking**" with "**VLAN trunk range**" set with range of management VLANs) and **DataPortgroup** (Portgroup defined with "**VLAN type**" as "**VLAN Trunking**" and "**VLAN trunk range**" set with range of tenant VMs' traffic - VLAN ranges, exclusive of management VLAN). Management VLAN and Data VLANs can share the same uplink or can be on different uplinks and those uplink ports can be a part of the same DVS or can it can be on separate DVS.

**Nova Compute Proxy:** It is a HP customized community's VMWare VC driver required to stitch the network from OVSvApp standpoint. Generally, a tenant VM's vnic is connected to a portgroup on a VSS (Virtual Standard Switch) or DVS (Distributed Virtual Switch).

Normal Flow for nova boot in case of KVM is:

- Allocate the network and allow neutron to asynchronously handle creation of required port for tenant VM based on network-id
- Create tenant VM with required specs
- Power-on VM

Normal Flow for nova boot in case of ESX is:

- Allocate network and have portgroups created before on hand on vCenter server
- Create tenant VM and its vnic gets attached to the pre-exsiting portgroup
- Power-on VM

But, in case of OVSvApp solution, the flow for nova boot is:

- Create tenant VM without vnics
- Allow OVSvApp agent to dynamically create portgroup based on network-id, cluster and DVS details as against the normal flow where a portgroup pre-exists
- Customized nova compute proxy waits until portgroup is created [deviation from normal flow]
- Once portgroup is created, tenant VM is reconfigured with a vnic and that would be attached to the portgroup dynamically created

## OVSvApp Internals

OVSvApp VM which runs OVSvApp agent waits for cluster events like "**VM_CREATE**", "**VM_DELETE**" and "**VM_UPDATE**" from vCenter Server and acts accordingly. OVSvApp agent also communicates with the Neutron Server to get information like port

details per VM and Security Group rules associated with each port of a VM from the Neutron Server to program the Open vSwitch within OVSvApp VM with FLOWs.

Open vSwitch comprises of 3 OVS Bridges namely **Security Bridge, Integration Bridge** and **Physical Connectivity Bridge** in case of VLAN, and **Tunnel Bridge** in case of VXLAN.

**Security Bridge** receives tenant VM traffic where Security group rules are applied at VM port level. It contains Open vSwitch FLOWs based on the tenant's Openstack Security Group rules which will either allow/block the traffic from the tenant VMs. Open vSwitch based Firewall Driver is used to accomplish Security Groups functionality, similar to iptable Firewall Driver used in KVM compute nodes.

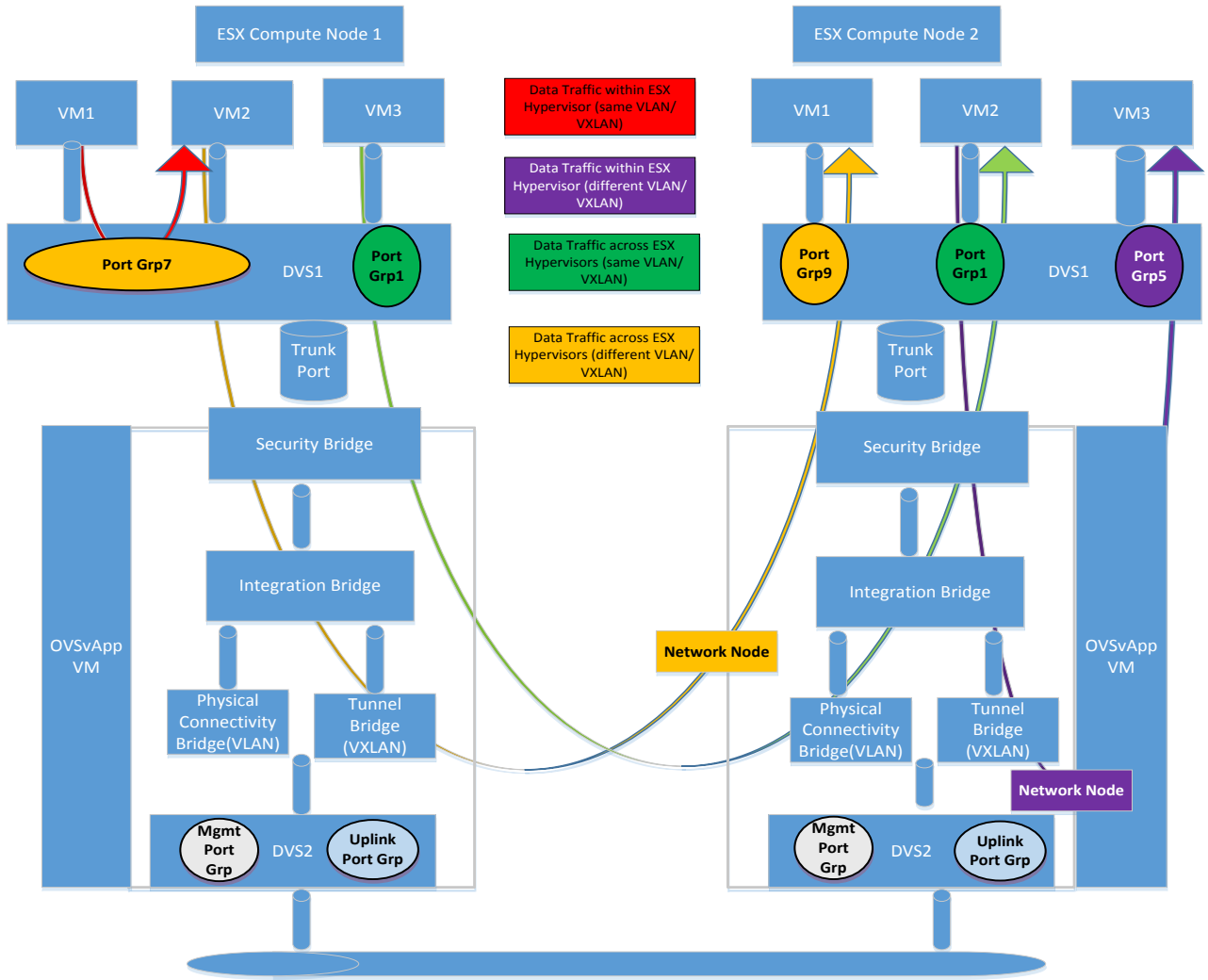
**Integration Bridge** connects Security Bridge and Physical Connectivity OR Tunnel Bridge. The reason to have Integration Bridge is to leverage existing Openstack Open vSwitch L2 agent feature to a maximum.

**Physical Connectivity Bridge** provides connectivity (VLAN provisioning) to the physical network interface cards.

**Tunnel Bridge** is used for establishing VXLAN tunnels to forward tenant traffic on the network.

## Data Flow Use Cases

**Figure 2.** Tenant VM Traffic Flow Scenarios



OVSvApp solution supports Intra VLAN/VXLAN traffic on same / across ESX hypervisors. Network node is used for inter VLAN/VXLAN traffic.

## Sequence Diagram

**Figure 3.** OVSvApp Solution – VLAN provisioning

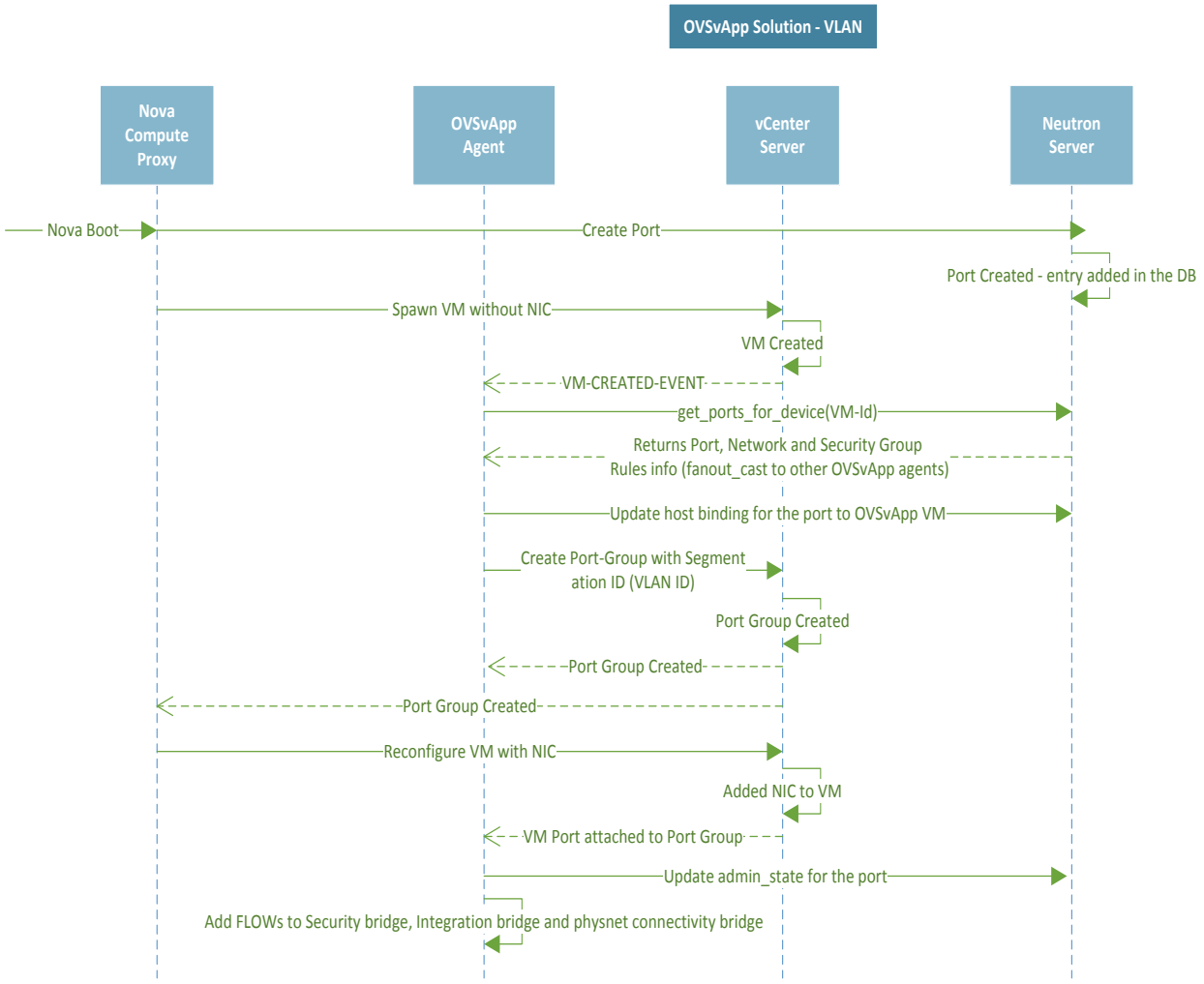
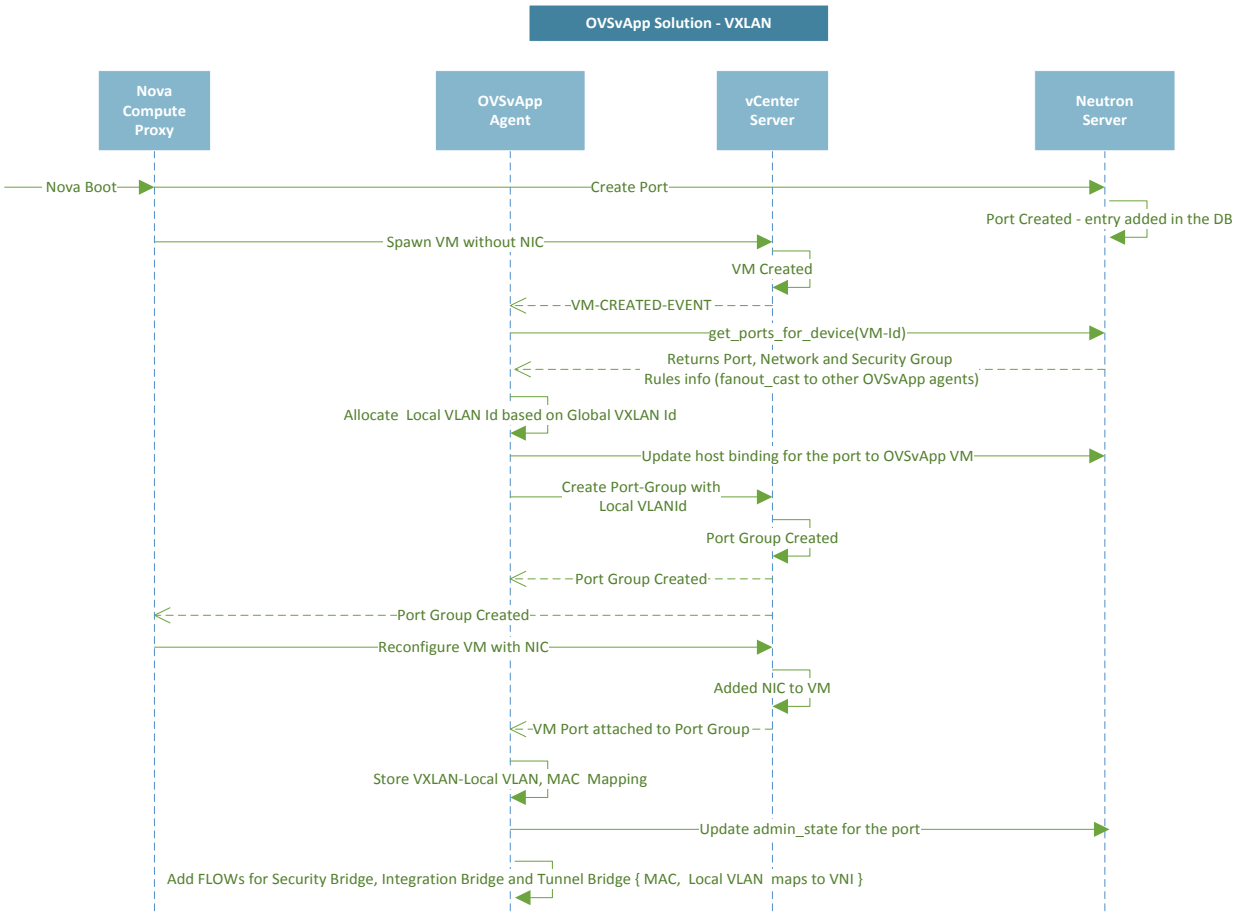


**Figure 4.** OVSvApp Solution – VXLAN provisioning




## VLAN/VXLAN provisioning

In case of VLAN provisioning, direct mapping approach of segmentation-id to vlan id, to be associated with a portgroup, mapping is used and is applicable per datacenter. i.e.; there is no global to local vlan id conversion mechanism. But in case of VXLAN provisioning, to support $2^{24}$ networks (scalable networks), an indirect mapping approach of global VNI (Virtual Network Identifier) to local VLAN Id mapping is used and is applicable within a cluster. So, a global VNI can map to different local VLAN Ids across clusters. So, on the vCenter side, at the portgroup level, a local vlan will be associated with it for a give global VNI.

## Security Groups

Security Groups – traffic filtering rules are applied on tenant VM ports. Open vSwitch based Firewall (OVSFirewallDriver) is used to program Open vSwitch FLOWs on Security Bridge to impose security rules. Based on the security rules of the ports provided by neutron server, the OVSFirewallDriver translates the security rules to Open vSwitch rules and programs the Security Bridge accordingly. A thread within the OVSvApp agent keeps continuously monitoring for any security rules related updates and applies on the Security Bridge accordingly.

## vMotion

VMWare Distributed Resource Scheduler (DRS) uses vMotion for live migration of VM within a cluster. To support vMotion via OVSvApp solution, similar FLOWs are populated on all OVSvApp VMs within a cluster i.e; when a tenant VM is booted on an ESX hypervisor by vCenter server, FLOWs related to the tenant VM are not only added on its hosting OVSvApp VM, but also concurrently on all other OVSvApp VMs within a cluster. So, when a tenant VM is vMotioned from one ESX hypervisor to the other ESX hypervisor, necessary FLOWs are readily available for uninterrupted network connectivity. Necessary VMWare prerequisites for vMotion should be taken care of.

## Fault Management

OVSvApp solution caters to the following failure scenarios:

1. OVSvApp agent is down – upstart mechanism tries to bring up the agent back
2. Open vSwitch process crash - upstart mechanism tries to bring up the open vSwitch process back
3. OVSvApp VM crashes - Agent monitoring mechanism (put ESX host in maintenance mode) and DRS will migrate tenant VMs to other ESX hosts within a cluster safeguards traffic from being black holed

# OVSvApp VM Automated Installer

An automated OVSvApp VM installer is in place, where a VMWare admin/tenant admin inputs required fields [vCenter Server credentials, DVS details, portgroup details etc.] and based on the inputs, OVSvApp VMs are installed on each ESX host within a cluster. The installer supports 2 modes of installation: fully automated/manual where the DVS and portgroups are created automatically on the fly OR the installation uses the existing DVS and portgroups and continues to install OVSvApp VMs on each ESX host within a cluster.

The OVSvApp VM deployment process includes the following:

- Upload the OVSvApp file (.ova) to one of the ESX hosts in a data center
- Input values at ovs_vapp.ini file
- Add settings to the configuration file so that the OVSvApp deployment script can clone the file on each host
- Run the deployment script

Also, OVSvApp VMs can be deleted from the cluster in an automated fashion by running a script called “clean.py”.

# AT&T Collaboration

Industry has been looking for ways to leverage existing investment in virtualization technologies in their open source cloud. HP Helion OpenStack delivers on this with OVSvApp functionality. Using such a solution, cloud operators can extend their NFV initiatives to include resources based on VMWare’s ESX hypervisor without sacrificing the policy definition from the OpenStack control plane or interoperability with other open source hypervisors.

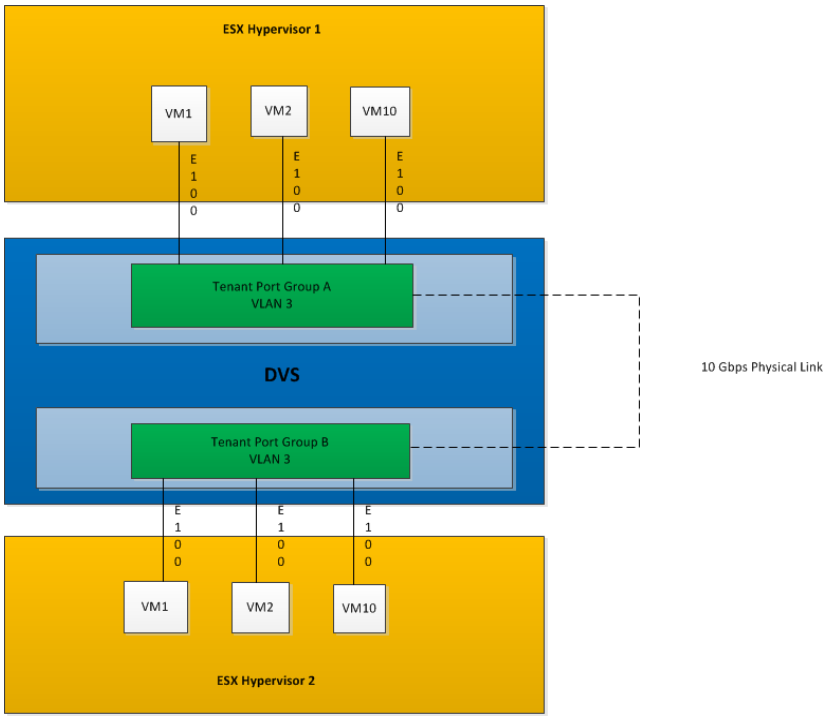
AT&T has reviewed the OVSvApp functionality in this white paper and stands behind HP’s blueprints, which attempt to open source this technology as part of a future OpenStack release.

# OVSvApp Performance

OVSvApp solution's performance is drawn based on 2 metrics namely: **"Throughput"** in "**Gbps**" and **"Round Trip Latency"** in **"µsec/Transaction"** for a count of 10 and 20 tenant VMs. Performance data is drawn using **Netperf** tool. Performance characteristics are carried out based on the following 3 scenarios:

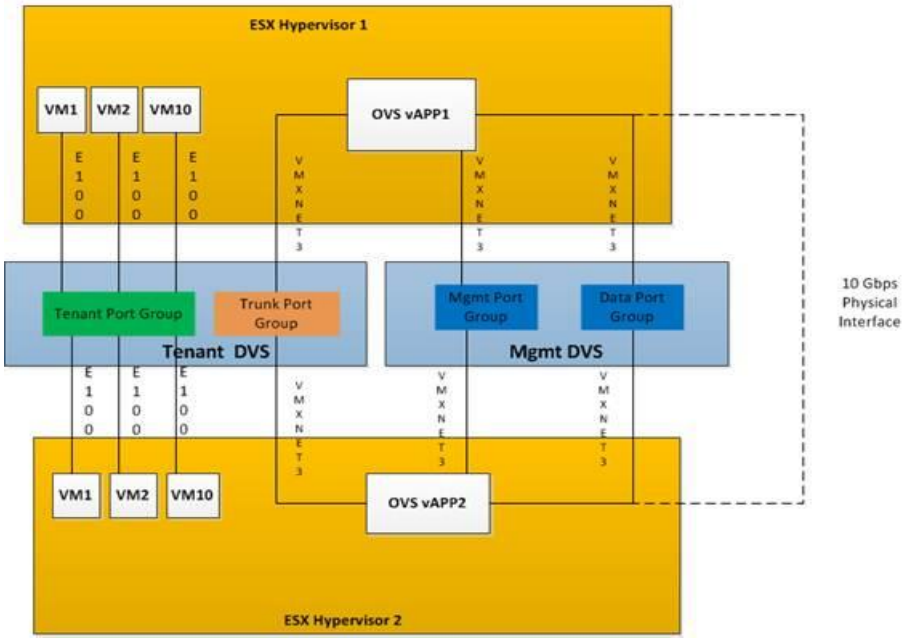
1. **Baseline – Only VMWare Distributed Virtual Switch (DVS)** with no OVSvApp VM to get base reference numbers. ESX hypervisor 1 hosts 10 VMs on each of which Netserver will run. ESX hypervisor 2 hosts 10 VMs on each of which Netclient will run and pumps the TCP traffic to 10 VMs on ESX hypervisor 1.

**Figure 5**. Setup for Scenario 1: Baseline – Only DVS



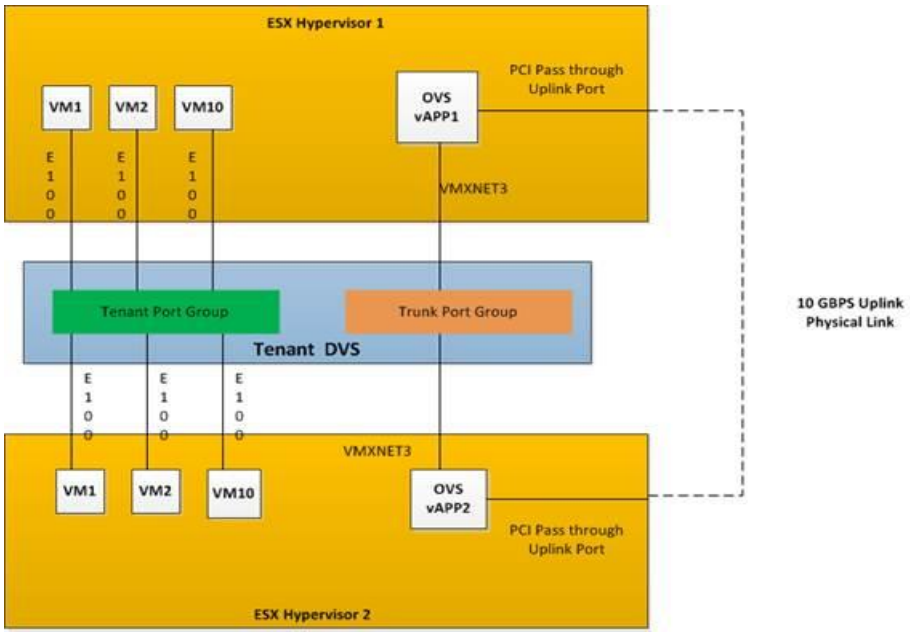
2. **OVSvApp VM – Data uplink interface as VMXNet3** - OVSvApp VM is configured with 3 NICs of type VMXNet3 for management, trunk and data uplink. 10 tenant VMs are spawned in a singke network. 10 tenant VMs from ESX hypervisor 2 will pump traffic to 10 tenant VMs on ESX hypervisor 1.

**Figure 6**. Setup for Scenario 2: Data uplink interface as VMXNet3



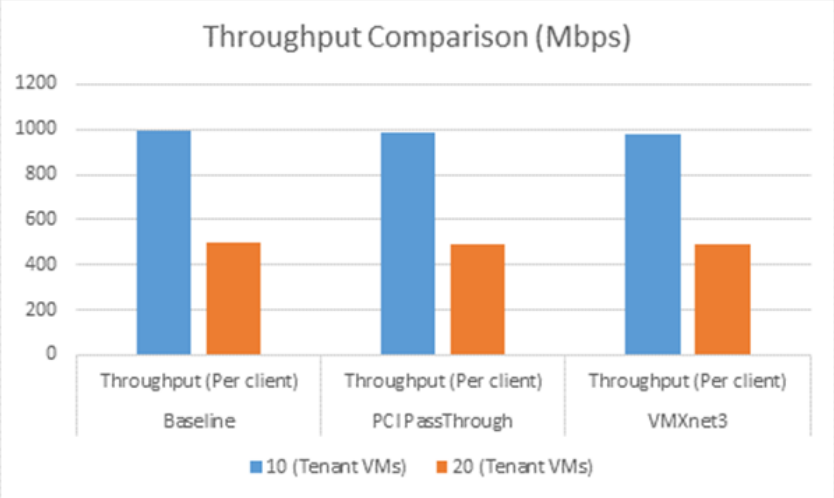

3. **OVSvApp VM – Data uplink interface as PCI Pass through** - Similar setup as in Scenario 2 with a difference that data uplink interface is PCI pass through type.

**Figure 7**. Setup for Scenario 3: Data uplink interface as PCI Pass through

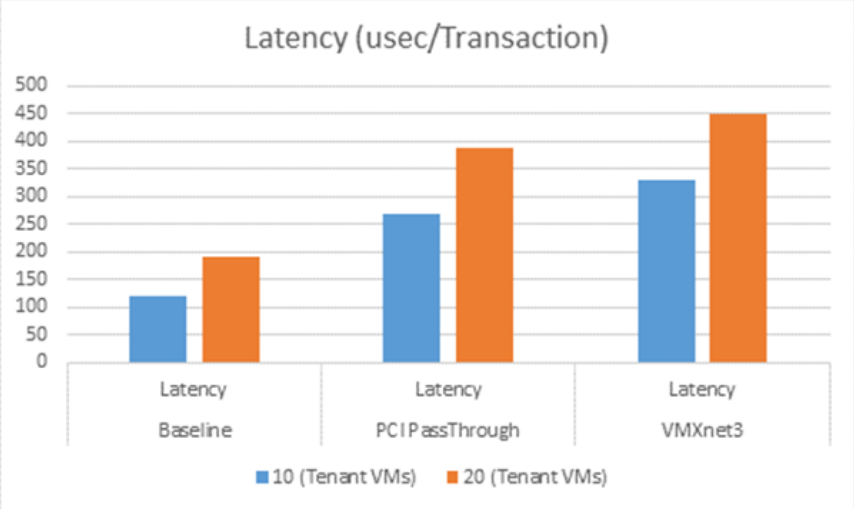


## Throughput and Latency

Graphical Representation of Throughput for 3 Scenarios:




Graphical Representation of Latency for 3 scenarios:




Performance tuning parameters to be set to yield the above stated values:

VMXNet3:

- Uplink interface & Trunk interface MTU to be set to 9000
- Integration bridge, Security Bridge, physical connectivity bridge to be set to 8900
- Guest VM MTU to be set to 8900
- The uplink physical interface link at the ESX host level MTU to be set to 9000

PCI Pass Through:

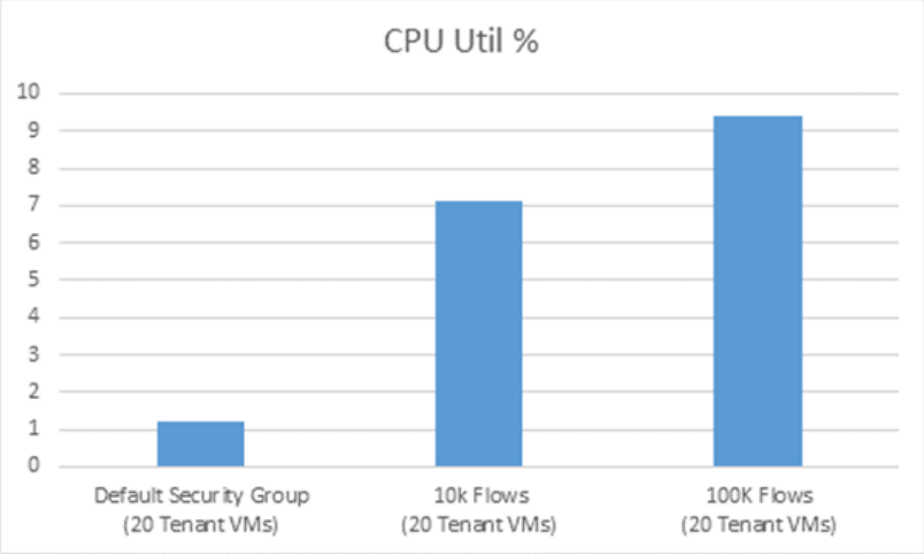
- Uplink interface & Trunk interface MTU to be set to 9000
- Integration bridge, Security Bridge, physical connectivity bridge MTU to be set to 8900

Tenant VM MTU to be set to 8900

## CPU Utilization

With 20 tenant VMs on each ESX host (2 ESX hosts taken for calculation) and netperf pumping in TCP traffic, below is the CPU utilization for 3 different scenarios:

- Deafult Security Groups - ~1% (rounded off)
- 10K FLOWs - ~7% (rounded off)
- 100K FLOWs - ~10%




## Configuration

OVSvApp solution is implemented on ESX hosts which are HP SL230 Gen 8 Servers with 16 CPU, 64 GB RAM, 10G NIC (Emulex) and 1 TB HDD. Further OVSvApp VMs are 4 CPU, 4 GB RAM, 40 GB HDD service VMs.

# References


- https://blueprints.launchpad.net/neutron/+spec/ovsvapp-esxi-vxlan : Blueprint for ESX with VXLAN
- https://review.openstack.org/#/c/104452/ - Blueprint for ESX with VLAN as per Kilo – neutron-specs specification
- https://review.openstack.org/#/c/103728/ - Blueprint for ESX with VXLAN as per Kilo – neutron-specs specification
- https://blueprints.launchpad.net/neutron/+spec/ovs-firewall-driver - Open vSwitch-based Security Groups: Open vSwitch Implementation of FirewallDriver