VMware® NSX for Multi-Hypervisor (NSX-MH) Network Virtualization Design Guide

DESIGN GUIDE

vmware®

Intended Audience

This document is targeted toward virtualization and network architects interested in deploying the VMware® NSX network virtualization solution in a multi-hypervisor environment.

Overview

IT organizations have gained significant benefits as a direct result of server virtualization. Server consolidation has reduced physical complexity, increased operational efficiency, and introduced the ability to dynamically re-purpose underlying resources to quickly and optimally meet the needs of increasingly dynamic business applications, to name just a handful of the gains already realized.

Now, VMware's Software Defined Data Center (SDDC) architecture is extending virtualization technologies across the entire physical data center infrastructure. VMware NSX, the network virtualization platform, is a key product in the SDDC architecture. With VMware NSX, virtualization now delivers for networking what it has already delivered for compute and storage. In much the same way that server virtualization programmatically creates, snapshots, deletes and restores software-based virtual machines (VMs), VMware NSX network virtualization programmatically creates, snapshots, deletes, and restores software-based virtual networks. The result is a completely transformative approach to networking that not only enables data center managers to achieve orders of magnitude better agility and economics, but also allows for a vastly simplified operational model for the underlying physical network. With the ability to be deployed on any IP network, including both existing traditional networking models and next-generation fabric architectures from any vendor, NSX is a completely non-disruptive solution. In fact, with NSX, the physical network infrastructure you already have is all you need to deploy a software-defined data center.

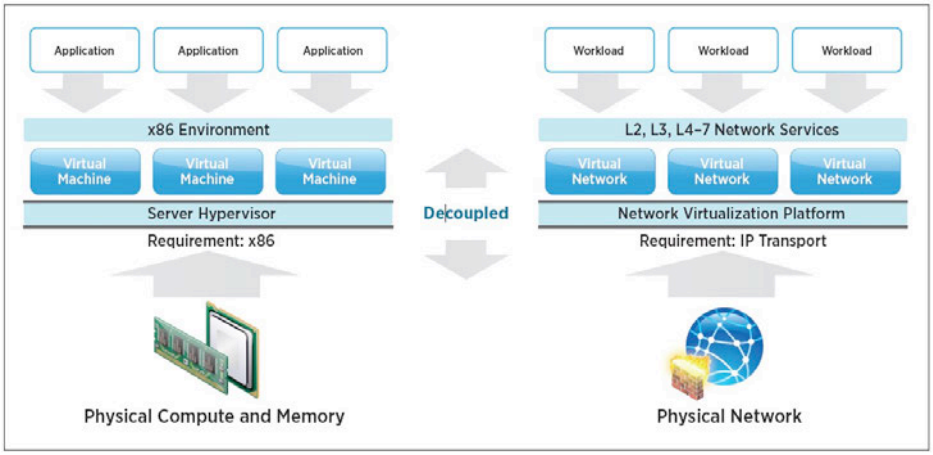


Figure 1: Server and Network Virtualization Analogy

Figure 1 draws an analogy between compute and network virtualization. With server virtualization, a software abstraction layer (server hypervisor) reproduces the familiar attributes of an x86 physical server (e.g., CPU, RAM, disk, NIC) in software, allowing them to be programmatically assembled in any arbitrary combination to produce a unique virtual machine (VM) in a matter of seconds.

With network virtualization, the functional equivalent of a "network hypervisor" reproduces the complete set of layer 2 to layer 7 networking services (e.g., switching, routing, access control, firewalling, QoS, and load balancing) in software. As a result, these services can be programmatically assembled in any arbitrary combination, to produce unique, isolated virtual networks in a matter of seconds.

Not surprisingly, similar benefits are also derived. For example, just as VMs are independent of the underlying x86 platform and allow IT to treat physical hosts as a pool of compute capacity, virtual networks are independent of the underlying IP network hardware and allow IT to treat the physical network as a pool of transport capacity that can be consumed and repurposed on demand. Unlike legacy architectures, virtual networks can be provisioned, changed, stored, deleted and restored programmatically without reconfiguring the underlying physical hardware or topology. By matching the capabilities and benefits derived from familiar server and storage virtualization solutions, this transformative approach to networking unleashes the full potential of the software-defined data center.

With VMware NSX, you already have the network you need to deploy a next-generation software-defined data center. This paper will highlight the various functionalities provided by the VMware NSX for Multi-Hypervisor (NSX-MH) architecture and the design factors you should consider to fully leverage your existing network investment and optimize that investment with VMware NSX-MH.

Note: in the rest of this paper the terms "NSX" and "NSX-MH" are used interchangeably, and they always refer to the deployment of NSX in multi-hypervisor environments.

Introduction to Network Virtualization

Virtualization reaches its full potential when all data center resources -- including networking – are virtualized. Without network virtualization, each VM's (virtual machine's) addressing, connectivity, and associated network services (for example, its NAT configuration, security settings, and QoS policies) are defined in the network hardware, tying that VM to its current location in the data center. Without network virtualization, each VM's IP address and the set of L2-adjacent VMs depend on the physical location of the hypervisor and the physical network to which it is attached.

Network virtualization overcomes this limitation by allowing VMs to connect to logical networks rather than attaching directly to physical networking hardware, removing the restrictions of more traditional in-hardware network configurations that tie a VM to its physical location in the network.

Network virtualization provides:

- Forwarding-plane virtualization

  With NSX-MH network virtualization, each tenant in the data center occupies its own logical network. The logical network appears to each VM as a dedicated switching device and network, providing full L2 and L3 service models. Because the logical network is defined in the data center's virtualization software (hence "software-defined networking" or "SDN") rather than on its networking hardware, it provides address isolation from other tenants' logical networks that share the physical network.

- Management-plane virtualization

  Each VM's network settings (for example, its NAT policies, security settings, QoS policies, and port statistics) automatically follow the VM wherever it is provisioned or migrated, and network management is done through a single management plane, the NSX API. The NSX API enables automated provisioning and monitoring of all logical networks and their underlying transport nodes. Cloud providers can use the NSX API to build tenant-facing management dashboards that give each tenant the ability to create and manage his or her logical network.

Two Network Views: Logical & Transport

In an NSX-MH deployment, there are two distinct network views: the logical network view and the transport network view. The distinction between the logical and transport network views is central to the NSX-MH architecture.

The Logical Network View

The logical network view is the set of connectivity and network services a VM sees in the cloud. The logical view is the view presented to tenants and is independent of the underlying physical devices of the data center. In a multi-tenant cloud, each tenant has his or her own logical network view and cannot see the logical network views of other tenants. The logical network view consists of the logical ports, logical switches, and logical routers that interconnect VMs and connect VMs to the external physical network.

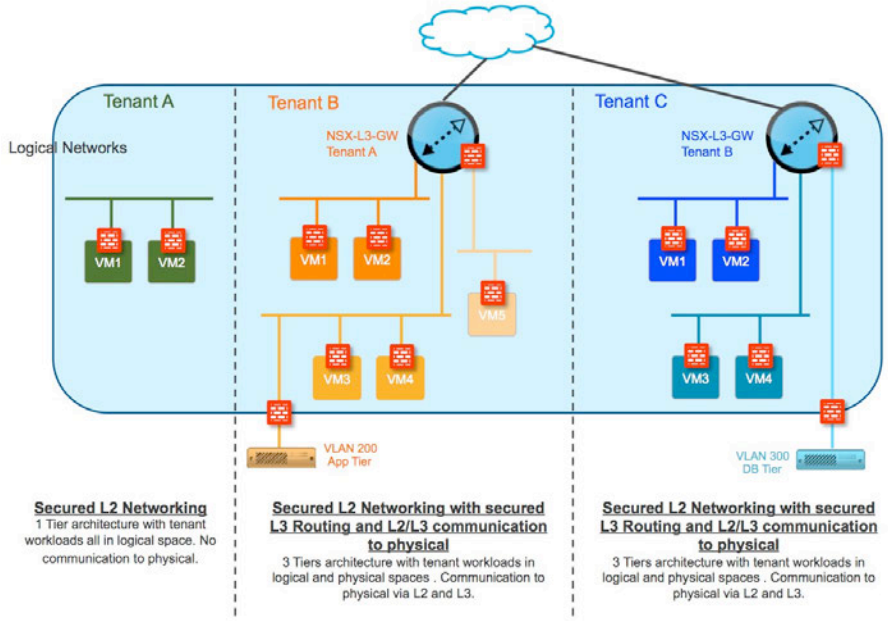


Figure 2: Tenant Logical Network "Logical View"

From the point of view of a VM and its administrators, the logical network is the network. VM administrators perform VM network provisioning, configuration, and monitoring using only the logical network view. The physical network and the placement of VMs on it is not something the VM administrator needs to worry about. The VM administrator just connects VMs to logical switches and logical routers. Once this is done, NSX manages eventual network wiring changes that result from VM migration or from changes in network hardware. The logical network view does not change when a VM migrates.

The Transport Network View

The transport network view is the view presented to cloud / data center administrators (people who deploy hypervisors and their associated network infrastructure). This view describes the physical devices that underlie the logical networks. We refer to these devices as "transport network" and they include the hypervisors that host VMs, NSX transport nodes, and the network hardware that interconnects hypervisors and connects them to external, physical networks.

When a VM migrates, NSX updates transport node devices to ensure unchanged operation of logical networks. The cloud/ data center administrator works in the transport network view, connecting hypervisors to the transport network, deploying other NSX transport nodes such as NSX Gateways, and connecting them to the physical network.

The Network Hypervisor

The logical and transport network views exist simultaneously. NSX acts as a network hypervisor that virtualizes the transport view devices to faithfully underpin one or more logical network views. This task is analogous to how an x86- based hypervisor server provides each VM with a logical view of resources (virtual CPU, memory, and disks) by mapping virtual resources to physical server hardware.

When something changes in the logical network view (for example, when a tenant network administrator re-configures a logical port used by a VM), NSX modifies the forwarding state of all relevant transport node devices in the transport network view to reflect the change. Similarly, when something changes in the transport network view (for example, when a VM starts up or is migrated), NSX updates the forwarding rules and state in the transport network view so the VMs' logical network views stay consistent.

Overview of NSX-MH Network Virtualization Solution

An NSX deployment consists of a data plane, control plane and management plane, as shown in Figure 3.

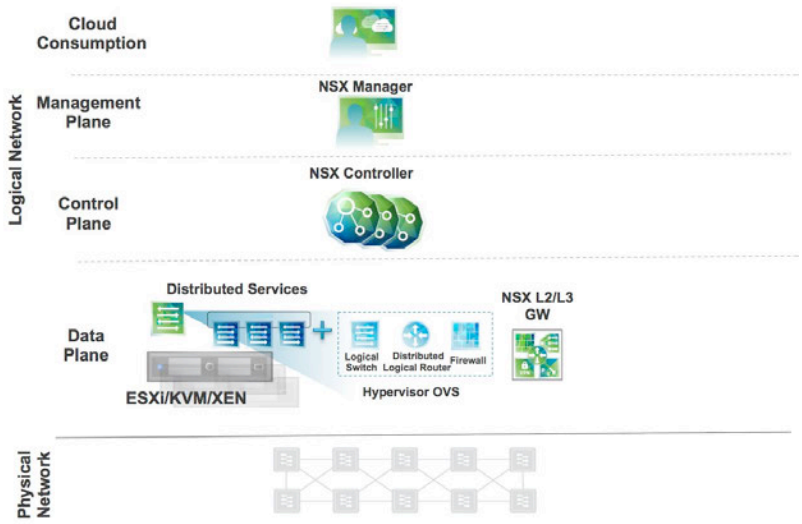


Figure 3: NSX Components

Control Plane

NThe NSX Controller Cluster accepts logical network configuration instructions from tenants and administrators (through the NSX API service). It also learns from hypervisors the locations of the different VM virtual network interfaces.

Using this information, it calculates the required network flow entries, and inserts these network flow entries into Open vSwitch (OVS) instances running on the transport nodes (hypervisor switches and NSX transport nodes) via OpenFlow.

A network flow is a 5-tuple identifier associated with an action. Some fields within the 5-tuple can be wildcarded. For instance:

If all traffic from TenantA-VM1 (hosted on hypervisor1) to TenantA-VM2 (hosted on hypervisor2) is allowed, the NSX Controller Cluster will push a flow on the hypervisor1-OVS "from L2-mac@VM1 to L2-mac@VM2 any IP@ / any protocol àà encapsulate the traffic and send it to hypervisor2")

If only HTTP traffic from TenantA-VM1 (hosted on hypervisor1) to TenantA-VM2 (hosted on hypervisor2) is allowed, the NSX Controller Cluster will push a flow to the hypervisor1-OVS. If we express it in a readable format, the contents of the flow might look like this: "from L2-mac@VM1 / L3-IP@VM1 to L2-mac@VM2 / L3-IP@VM2 on TCP:80 àà Encapsulate the traffic and send it to hypervisor2."

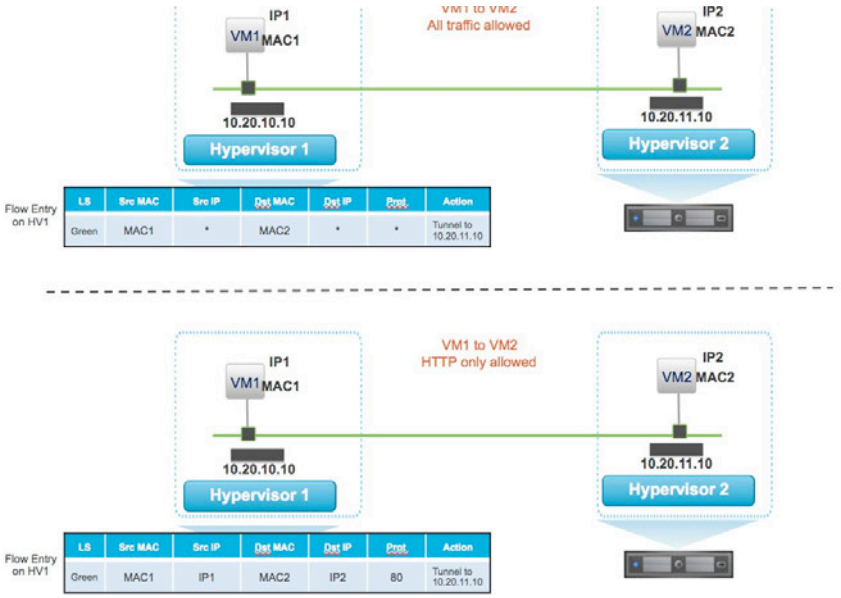


Figure 4: Flow Entry for Unicast Communication

When OVS encounters a BUM frame (broadcast, unknown unicast, or multicast frame) it floods the frame to the logical switch either directly from the source hypervisor to all transport nodes with that logical switch present (if you have chosen the source nodes replication option in NSX) or via an NSX Service Node (if you use the default, recommended Service Node replication option in NSX).

The Controller Cluster is a distributed system that runs on a set of x86 servers. Clustering provides:

- High-availability—The ability to survive failures of individual/multiple simultaneous failures of Controller Nodes.
- Scale-out—The ability to increase the number of endpoints managed by the control plane by adding more servers to the Controller Cluster.

Data Plane

The NSX data plane is implemented by access-layer switching devices (hypervisor virtual switches such as OVS and NSX vSwitch) and NSX appliances (NSX Service Nodes, NSX Gateways, third-party hardware L2 Gateways), all managed by the Controller Cluster. We refer to these access-layer devices as “transport nodes.”

Traditional network hardware (not managed by NSX) provides the underlying fabric that connects the NSX transport nodes. NSX creates network overlays that implement the logical network view atop the physical network.

Transport nodes support the OVS management interface (OVSDB control plane), an open protocol that enables fine- grained control of and visibility into a switch’s packet forwarding behavior. Endpoints are:

- Hypervisor — An OVS-enabled virtual switch is installed on each hypervisor, allowing VM endpoints running on the hypervisor to be attached to a logical network in NSX. Currently supported hypervisors are ESXi, KVM and Xen (Hyper-V support planned for a future release).
- NSX Gateway Service — An NSX Gateway Service consists of one or more NSX Gateway that connects a logical network to a physical network not managed by NSX. Each Gateway Service can operate as an L2 Gateway Service sending traffic to a physical L2 segment, or as an L3 Gateway Service mapped to a physical router port.

In the data plane, logical switches can be implemented using:

- Overlay logical network: Overlay networks encapsulate traffic using L2-in-L3 tunneling, meaning that the logical network topology can be completely decoupled from the transport network topology. Encapsulations available are Stateless Transport Tunneling (STT), Generic Routing Encapsulation (GRE), Virtual eXtensible LAN (VXLAN), Internet Protocol Security STT (IPSEC-STT), and IPSEC-GRE.

  Overlay is the mode selected in most deployments, and STT is the default encapsulation protocol, since it offers the best throughput rate.

- Bridged logical network: Bridged networks do not make use of encapsulation. VM traffic is sent directly on the physical network, similar to traditional hypervisor networking. This mode is rarely selected, as it requires L2 adjacency between hypervisors. As a consequence, the rest of this paper focuses exclusively on the use of overlay logical networks.

NSX Service Nodes are optional transport nodes working in an active-active cluster mode. They can be used:

- to process BUM traffic (broadcast, unknown unicast, or multicast frames) where hypervisors send traffic to NSX Service Nodes, which in turn replicate it to the other transport nodes hosting that logical switch; and
- to change between encapsulation methods, for instance if you have configured NSX to use STT from hypervisors to Service Nodes and STT-IPSEC from Service Nodes to remote NSX-Gateways

Neither OVS nor OpenFlow support is required on the physical network devices that interconnect the access switches. Any network hardware that provides unicast IP connectivity between transport nodes is sufficient.

Management Plane and Consumption Platforms

Clouds are operated and managed by a cloud management platform (CMP), sometimes also referred to as a cloud management system (CMS). A CMP can offer cloud administration services to the end-users / tenants. Services can range from allowing the user to deploy applications from authorized templates to letting them deploy their own applications from scratch.

The CMP (e.g., OpenStack, CloudStack and home-grown cloud platforms) is used to automate the deployment and configuration of applications and their logical networks, providing the agility required in the modern data center.

The CMP pushes logical network configuration requests to the NSX Controller Cluster via the NSX APIs. For troubleshooting and configuration, management tools interact with the NSX installation by making API calls to the NSX Controller Cluster.

A quick way to become acquainted with the NSX APIs is by using the NSX Manager, a web-based GUI offering a user- friendly way to interact with the NSX APIs.

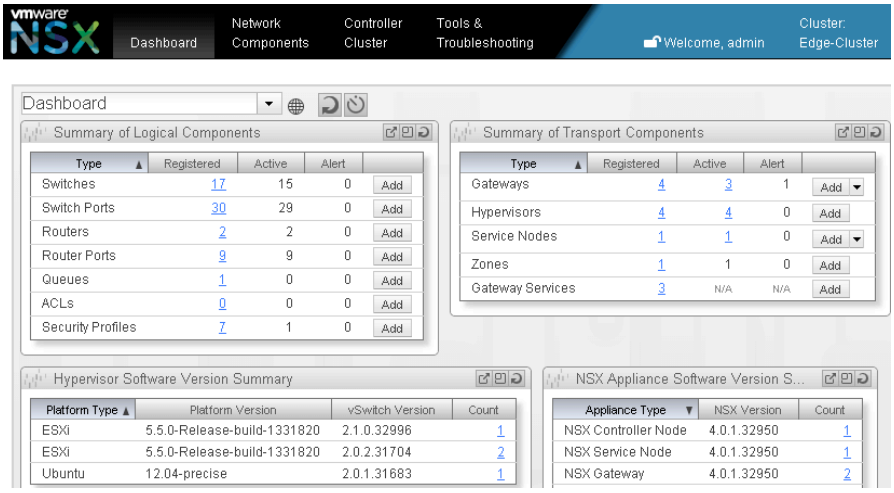


Figure 5: NSX Manager dashboard view

NSX Plugin for OpenStack

NSX-MH deployments very frequently rely on the use of OpenStack as consumption platform. OpenStack is an open- source cloud management platform (CMP) for public and private clouds backed by some of the biggest companies in software development and hosting, including VMware. Refer to https://www.openstack.org/software/ for a list of the different functional components (named "projects") that build the OpenStack architecture.

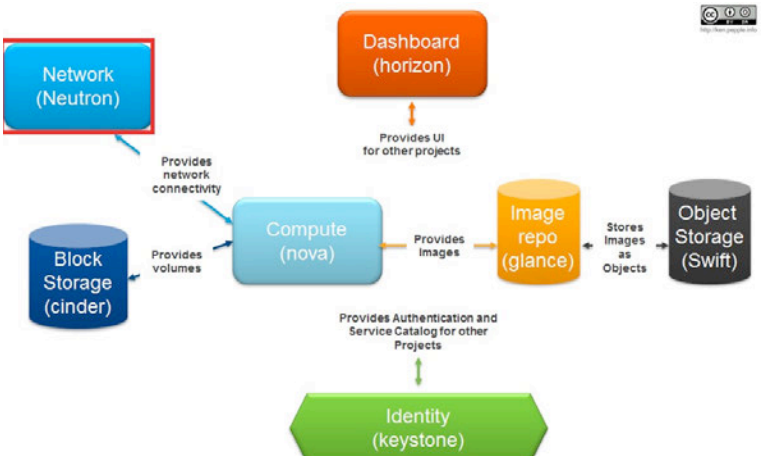


Figure 6: OpenStack Core Projects

The OpenStack network capabilities are offered by its core project called Neutron. Neutron allows third parties to enrich its capabilities, and VMware offers a Neutron Plugin called the "NSX Plugin." This plugin allows OpenStack Neutron to use NSX-MH for all the OpenStack network services.

With its NSX Plugin, all the NSX-MH network services previously described are available in OpenStack:

- L2 overlay networks
- L2 networks that span logical and physical space
- L3 centralized and distributed routing
- Security Profiles
- ACLs
- QoS

OpenStack Example

The figure below highlights an example of logical networks created by a given tenant in an OpenStack-managed environment, as seen in the Openstack user interface, Horizon:

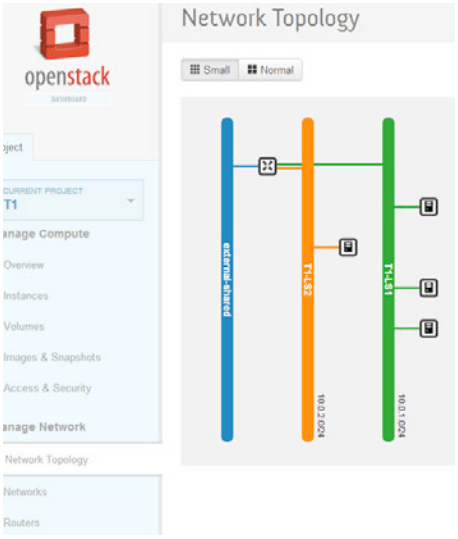


Figure 7: Horizon GUI

NSX Components and Functional Services

Logical Networking Functional Components

In the context of the NSX architecture, the logical networking functionalities (switching, routing, security, etc.) can be thought of as having a packet forwarding pipeline that implements multiple features, including L2/L3 forwarding, security filtering, and packet queuing.

Unlike a physical switch or router, NSX does not rely on a single transport view device to implement the entire forwarding pipeline. Instead, the NSX Controller creates instead packet-processing rules on all relevant transport nodes to mimic how a single logical device would handle data. Those transport nodes can hence be thought of as "line-cards" that implement a portion of the logical network function pipeline. The physical network connecting the transport nodes acts as the "backplane" that carries packets from one "line-card" to the other.

The logical networks created by the NSX platforms leverage two types of access layer entities, the Hypervisor Access Layer and the Gateway Access Layer. The Hypervisor Access Layer represents the point of attachment to the logical networks for virtual endpoints (virtual machines), whereas the Gateway Access Layer provides L2 and L3 connectivity into the logical space to physical devices and endpoints deployed in the traditional physical network infrastructure.

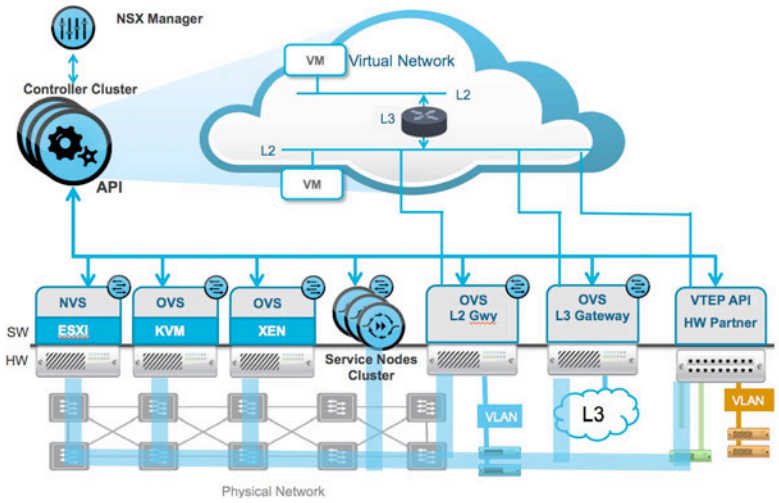


Figure 8: Logical Networking Components

One of the main advantages of network virtualization is the capability of decoupling logical network connectivity from the underlying physical network fabric and providing logical network abstraction and functions. As shown in Figure 8, there are several functional components that enable that and form the NSX-MH architecture. As discussed in the rest of this document, the use of an overlay protocol (STT, GRE, or VXLAN) to encapsulate virtual machine traffic between the different functional components is a key function to achieve logical/physical network decoupling. Let's take a more detailed look at the functions of each component of the NSX-MH platform.

NSX Manager

NSX Manager is the web-based graphical user interface (GUI) that provides a user-friendly method of interacting with the APIs exposed by the NSX Controller, as shown below.

The NSX Manager is hence used to interact with the NSX APIs exposed by the NSX Controller for configuring

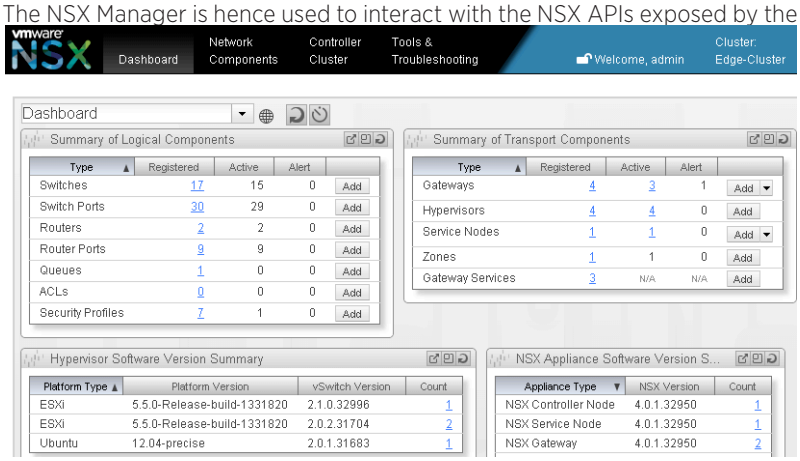


Figure 9: NSX Manager GUI

transport networks and logical switches and connecting virtual machines to these logical switches. In production deployments, it is expected that the deployment and management of logical switches will be automated through a cloud management platform (like OpenStack). In other words, the NSX Manager is not designed for day-to-day provisioning or automated "wizard based" configuration, and as such its user-interface role is not critical in the context of the NSX-MH architecture.

Additionally, the NSX Manager offers basic troubleshooting (Port Connectivity), operational (Syslog) and monitoring (Statistic Collection) capabilities.

Note: The NSX Manager should be deployed as a virtual appliance using the ISO available on the NSX customer support portal.

NSX Controller Cluster

The Controller Cluster in NSX-MH is the control plane component responsible for managing the Open vSwitch (OVS) devices that perform packet forwarding on transport nodes (hypervisors, Service Nodes and Gateways). In doing so, the Controller Cluster enforces consistency between the logical network view and the transport network view. Changes occurring in the transport view, such as a VM start or migration, cause the Controller Cluster to update the forwarding rules and state in the transport layer so that the connectivity seen by a VM is always consistent with the logical view. At the same time, changes in the logical view triggered by API requests (as for example the configuration of a logical port used by a VM), cause the Controller Cluster to modify the forwarding state of all relevant transport nodes to reflect those changes.

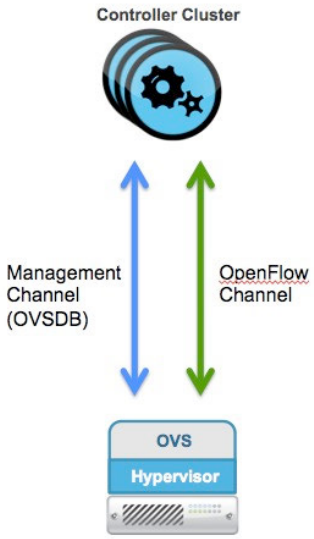


Figure 10: Control Plane Channels between Controller and NSX vSwitch/OVS

As shown in Figure 10, there are two control plane channels established between the Controller Cluster and each transport node (in this example an hypervisor running OVS): the first one is a management channel using the OVSDB protocol (TCP port 6632), the second is an OpenFlow channel (TCP port 6633). The management channel is used by the NSX Controller to communicate to the transport nodes various configuration information, including the creation of logical switches, logical routers, and so on.

At the same time, the hypervisors use the OVSDB management channel to provide the Controller Cluster with the MAC address and the VIF-ID (identifying a logical interface on the internal virtual switch) for each VM locally deployed and connected. This allows the Controller to learn the location of all the VMs and appropriately calculate transport view flows to implement the logical view. The Controller Cluster uses then the OpenFlow Channel to offload to the transport nodes all the required network flow entries required enabling communication between the various virtual workloads connected to the logical networks.

It is important to clarify how each transport node receives from the NSX Controller only the subset of the flow information, the flows relative to logical switches locally active (i.e. with workloads actively connected). In other words, a transport node will never receive flow information for establishing communication from/to a given logical switch, unless it has at least one virtual machine actively connected to that logical segment. This distributed approach is important to ensure high scalability for the overall solution.

The communication channels between each transport node and the Controller Cluster can successfully be

established only after they authenticate each other. SSL certificates are utilized for the mutual authentication purposes:

- Each transport node provides a certificate when registering with the Controller Cluster. The Controller Cluster must have previously registered this certificate to be able to accept OVS management connections from each node.
- At the same time, all the Controller Cluster nodes use the same SSL certificate to establish OVS management connections with the transport nodes, which also must be preconfigured with this certificate or they will trust (by default) the first certificate they receive.

On the north side, the Controller cluster exposes web service APIs (NSX APIs) that can be used for example by a Cloud Management System (CMS) to instruct the NSX Controller to create logical ports and logical network and connect virtual machines to them. The NSX Manager itself (discussed in the next section) is an example of an application that can interact with the APIs exposed by the NSX Controller for the creation of logical network connectivity.

For resiliency and performance, production deployments should deploy a Controller Cluster with multiple nodes. The NSX Controller Cluster represents a scale-out distributed system, where each Controller Node is assigned a set of roles that define the type of tasks the node can implement. The list of roles a given node can perform is highlighted in Figure 11.

Transport Node Management: maintains OVS management connections to the different transport nodes.

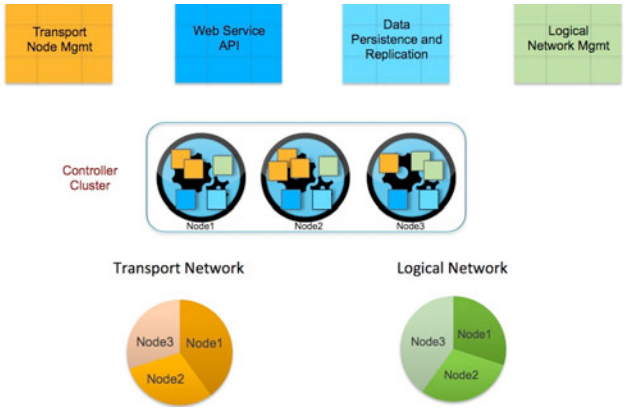


Figure 11: Controller Cluster Node Roles

- Logical Network Management: monitors when workloads arrive or leave a transport node and configures the OVS forwarding states to implement logical connectivity and policies.
- Data Persistence and Replication: stores data from the NSX API and OVS devices that must be persisted across all Controller Nodes in case of node failures or shutdowns.
- Web Service API: handles HTTP web service requests from external clients and initiates processing by other Controller Node tasks.

The figure above also illustrates how the roles responsibilities are also fully distributed between the different cluster nodes (a function named “slicing”). This means, for example, that different logical networks may be managed by different Controller nodes: each node in the Controller Cluster is identified by a unique IP address and when a transport node establishes the OVS management connection with one member of the cluster, a full list of IP addresses for the other members is passed down to the transport node, so to be able to establish communication channels with all the members of the Controller Cluster.

After the failure of a Controller Node, the slices for a given role that were owned by the failed node are

reassigned to the remaining members of the cluster. In order for this mechanism to be resilient and deterministic, one of the Controller Nodes is elected as a leader for each role. The leader for each role is responsible for allocating slices to individual Controller Nodes and determining when a node has failed, so to be able to reallocate the slices to the other nodes. The election of the leader for each role requires a majority vote of all active and inactive nodes in the cluster. This is the main reason why a Controller Cluster must always be deployed leveraging an odd number of nodes.

Number of NSX Nodes in Cluster	2	3	4	5
Majority Number	2	2	3	3
Number of Nodes that can fail	0	1	1	2

Figure 12: Controller Nodes Majority Numbers

Figure 12 highlights the different majority number scenarios depending on the number of Controller Cluster nodes (the recommended number is 3 or 5, depending on the scale of the deployment; in this paper we will use 3 nodes). It is evident how deploying 2 nodes (traditionally considered an example of a redundant system) would increase the scalability of the Controller Cluster (since at steady state two nodes would work in parallel) without providing any additional resiliency. This is because with 2 nodes, the majority number is 2 and that means that if one of the two nodes were to fail, or they lost communication with each other (dual-active scenario), neither of them would be able to keep functioning (accepting API calls, etc.). The same considerations apply to a deployment with 4 nodes that cannot provide more resiliency than a cluster with 3 elements (even if providing better performance).

Note: NSX currently (as of software release 4.1.2) supports clusters with a maximum of 5 nodes. Also, NSX Controllers in production deployments must be run as bare-metal appliances (and not in virtual machine form factor). Please refer to the NSX Release Notes for more specific information about hardware requirements for the NSX Controller appliances.

Service Node with Open vSwitch (OVS)

NSX Service Nodes are physical x86 appliances that run Open vSwitch (OVS) and play an important role in logical switching because they take care of distributing multi-destination traffic (such as L2 broadcast and multicast) belonging to a specific logical switch to all the transport nodes where that logical switch is currently active. More considerations about this replication function can be found in the following “Replication Modes for Multi-Destination Traffic” section.

Note: the Service Nodes can also be used to offload network packet processing from hypervisor vSwitches, such as CPU-intensive cryptographic processing, or to function as Management Rendezvous servers to proxy management traffic between the NSX Controller Cluster and remotely deployed NSX Gateways. Those specific use cases are out of scope for this paper.

Similarly to what was discussed for the NSX Controller, also the Service Nodes can be deployed in a cluster. This is important both from a high-availability and performance point of view.

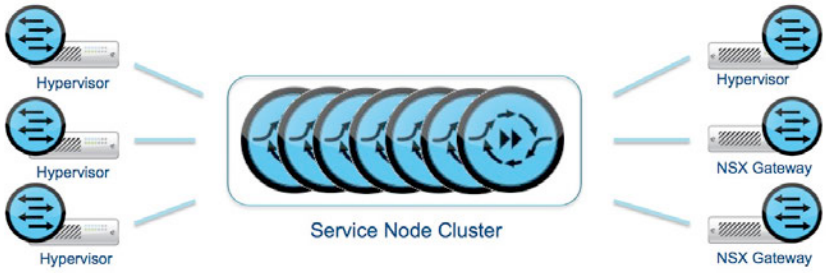


Figure 13: Service Node Cluster

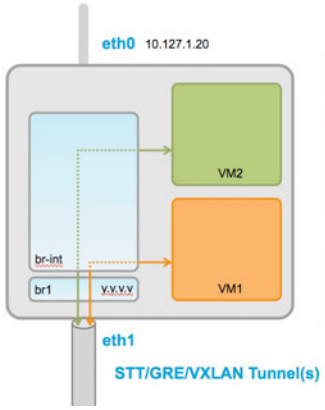
The Service Node cluster can be thought as a pool of OVS forwarding capacity, since all the nodes can simultaneously replicate traffic originated from a transport node. All the transport nodes (hypervisors and NSX Gateways) must hence establish tunnel connections (STT, GRE or VXLAN) to each member of the Service Node cluster. The hypervisors and NSX Gateways are then capable of load-balancing traffic to all the nodes in the Service Node cluster: the selection of which specific node to use is made based on the hashing of the L2/L3/L4 headers of the original packet that needs to be replicated (flow-level hashing).

Given the existence of multiple Service Nodes, it is critical to understand how specific Service Node failures are handled in order to minimize the traffic outage. The NSX Controller Cluster uses the Management connection to properly connect and configure the tunnels across all the transport nodes. The OVS running on each transport node (hypervisor, Service Node or NSX L2/L3 Gateway) is able to react immediately to tunnel failure. If a tunnel going to a specific Service Node fails, the transport node will detect this situation based on the loss of the keepalive probes exchanged through the tunnel and will re-hash the flows into one of the remaining tunnels available to the other Service Nodes, providing resilient Service Node failover.

Note: Up to 7 Service Nodes are supported in the cluster as of release 4.1.2. Also, NSX Service Nodes in production deployments must be run as individual bare-metal appliances using the ISO images from the customer portal (virtual |machine form factor is not supported in production).

Hypervisors with Open vSwitch

The hypervisors represent the transport nodes that implement the data plane for communication among virtual workloads. This is made possible by deploying an Open vSwitch (OVS) virtual switch to which virtual machines are connected.



Term	Brief Description
10.127.1.20	Management Interface (Primary)
10.10.1.10	IP Interface used for overlay traffic
br1	Eth1 is attached to this bridge and used for sourcing/receiving overlay traffic
Integration Bridge (br-int)	The bridge that is programed by the NSX control cluster. All tenant VMs are attached to the integration bridge. The integration bridge is responsible for forwarding logical network traffic to the correct destination hypervisor

Figure 14: Hypervisor with OVS

As shown in Figure 14, a Linux-based hypervisor running OVS has one external bridge (br1) that provides connectivity to the physical network via one (or multiple) uplink (eth1 in the example) used to exchange overlay traffic with the other transport nodes. Additionally, there is an internal bridge (br-int) where all VM interfaces are connected (no physical NIC is attached to this internal bridge). The Controller Cluster sets up flows on the integration bridge to direct packets to and from OVS tunnels (STT, VXLAN or GRE) and allow communication to remote VMs connected to the same logical networks.

OVS is utilized with KVM and Xen hypervisors, whereas a specific type of OVS switch (called NSX vSwitch or NVS) is deployed for VMware ESXi hosts.

Note: NSX vSwitch is supported on vSphere version 5.5 and higher only.

Figure 15 below depicts the Open vSwitch architecture and highlights the distinction between User Space and Kernel Space.

The first data path packet belonging to a flow that is received by Open vSwitch will be sent from the kernel space to the user space (for instance, VM1 on LS1 talking to VM2 on LS2). This is because it is in the user space that all the flow specific information pushed from the Controller Cluster are available. Based on that information, specific flow information will be then installed in the kernel space, so that all the subsequent packets belonging to the same flow will be switched in kernel space, allowing OVS to achieve wire-speed performance.

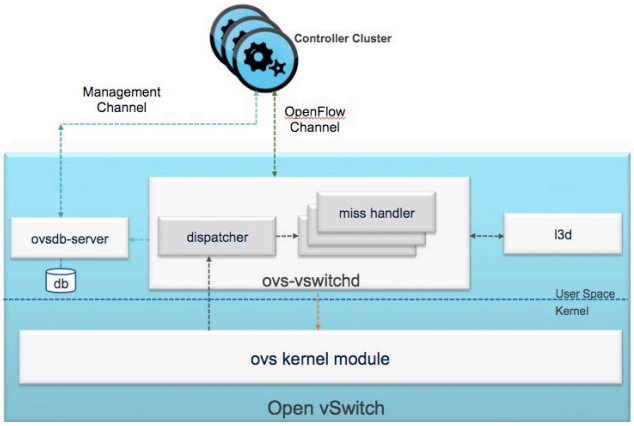


Figure 15: Open vSwitch Architecture

The basic assumption is that the NSX Controller knows where all the virtual machines belonging to a specific logical switch have been deployed, so every hypervisor will always have the right flow level information available to program flows in the kernel space for intra-logical switch communication. This may not be the case when a virtual machine tries to communicate to a physical workload (for example a bare-metal server) deployed as part of the same logical switch and accessible through a L2 Gateway device (software or hardware). More information on this specific scenario can be found in the "Logical Switching" section.

NSX L2 Gateways with Open vSwitch

The deployment of OVS on the different types of hypervisors enables communication between virtual workloads belonging to the same logical switch. However, there are several circumstances where it may be required to establish L2 communication between virtual and physical workloads. Some typical scenarios are (not exhaustive list):

- Deployment of multi-tier applications: in some cases, the web, application and database tiers can be deployed as part of the same IP subnet. Web and application tiers are typically leveraging virtual workloads, but that is not the case for the database tier where bare-metal servers are commonly deployed. As a consequence, it may then be required to establish L2 communication between the application and the database tiers.
- Physical to virtual migration: many customers are virtualizing applications running on bare metal servers and during this P-to-V migration it is required to support a mix of virtual and physical nodes on the same IP subnet.
- Leveraging external physical devices as default gateway: in such scenarios, a physical network device may be deployed to function as default gateway for the virtual workloads connected to a logical switch and a L2 gateway function is required to establish connectivity to that gateway.
- Deployment of physical appliances (firewalls, load balancers, etc).

To fulfill the specific requirements listed above, it is possible to deploy devices performing a "bridging" functionality that enables communication between the "virtual world" (logical switches) and the "physical

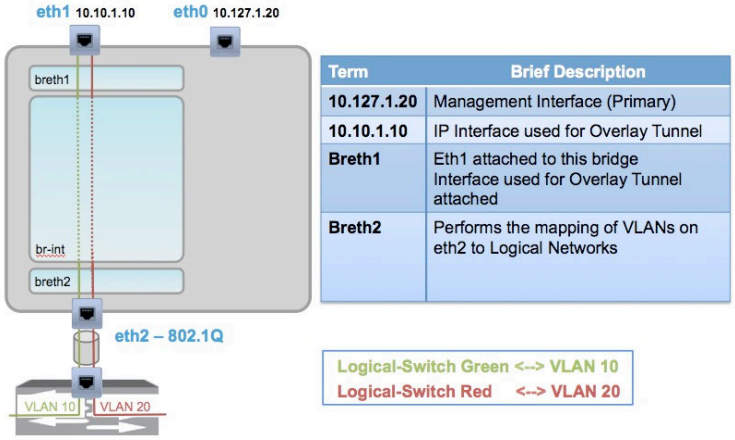
world" (non virtualized workloads and network devices connected to traditional VLANs).

NSX offers this functionality in software through the deployment of an NSX L2 Gateway Service allowing VMs to be connected at layer 2 to an external network (logical switch to VLAN ID mapping), even if the hypervisor running the VM is not physically connected to that L2 network.

Note: NSX Gateways can be deployed in production as physical (x86 bare metal servers) or virtual (virtual machines on ESXi and XEN hypervisors) appliances form factor. For more specific requirements, please refer to the latest NSX-MH Release Note.

Intra-subnet packets originated by VMs connected to the hypervisors and destined to a physical device are tunneled across the DC fabric using an overlay protocol (STT, GRE or VXLAN) and reach the active NSX L2 Gateway. The Gateway unencapsulates the traffic (removing the STT, GRE or VXLAN header) and bridges it to the VLAN where the physical device is connected (and vice versa for physical-to-virtual communication). The specifics of how this bidirectional communication is established will be discussed in the "Unicast Traffic (Virtual to Physical Communication)" section.

Figure 16 shows the high level internal architecture of an NSX L2 Gateway.



Term	Brief Description
10.127.1.20	Management Interface (Primary)
10.10.1.10	IP Interface used for Overlay Tunnel
Breth1	Eth1 attached to this bridge Interface used for Overlay Tunnel attached
Breth2	Performs the mapping of VLANs on eth2 to Logical Networks

Figure 16 NSX L2 Gateway Internal Architecture

In the example shown above, the NSX Gateway leverages 3 physical NICs:

- eth0 (10.127.1.20) is used as management interface to communicate with the NSX Controller (control-plane activity).
- eth1 (10.10.1.10) is used to originate overlay traffic (STT, GRE or VXLAN) and establish connectivity in the logical space with the other NSX transport nodes (hypervisors and Service Nodes).
- eth2 is the interface connecting with the external physical network infrastructure. This is the interface performing the bridging between the virtual and the physical worlds and it is typically configured as an 802.1Q trunk interface to be able to perform bridging for multiple logical switches/VLANs pairs. Notice how the mapping is strictly 1:1, meaning that traffic belonging to each logical switch will be mapped to a unique VLAN on the physical network side.

Note: it is technically possible (and fully supported from a VMware perspective) to consolidate all the traffic on a single NIC (or most likely a NIC bond), leveraging VLANs to provide logical separation between traffic types. More considerations around this can be found in the "Network Virtualization Design Considerations" section.

Resiliency for the L2 Gateway Service is achieved by associating it to a redundant pair of NSX L2 Gateway. For each configured logical switch to VLAN ID mapping, the two NSX Gateways will function in Active/Standby mode, as shown in Figure 17.

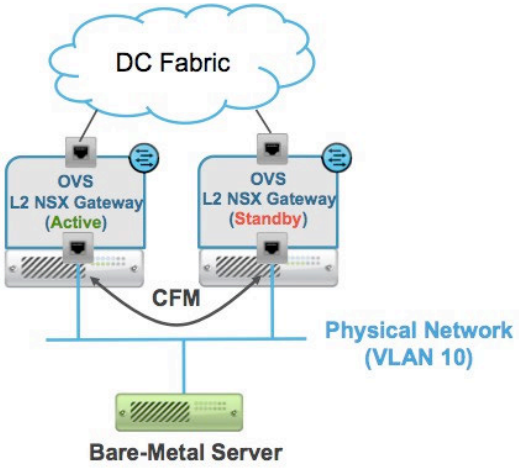


Figure 17: Active/Standby NSX L2 Gateways

This implies that only the active NSX Gateway performs the bridging function between the logical switch and the physical network, whereas the standby remains inactive until it detects the failure of the active unit. This is required to avoid the potential creation of an end-to-end L2 loop.

As shown in Figure 17, the NSX L2 Gateways use a modified version of 802.1ag Connectivity Fault Monitoring (CFM) probes as leader election mechanism. Those probes are exchanged via the interfaces connected to the physical network the L2 gateway is bridging traffic to. CFM packets contain an ID representing the L2 Gateway Service, and if an L2 Gateway hears from another L2 Gateway unit on the same segment with an ID lower than its own, it becomes inactive, breaking any potential loop. If the standby device stops detecting the probes originated by the active NSX Gateway, it takes on the active role and starts bridging traffic into and out of the physical network.

Note: The exchange of CFM probes always happens on the first VLAN that is configured as part of the L2 Gateway service. This implies that even if multiple logical-switch/VLAN mappings are configured for the same L2 Gateway service, the same L2 Gateway transport node (the active one) will always be responsible of performing the bridging functionality for all the VLANs. As a consequence, it makes no sense to associate more than two L2 Gateways with the same L2 Gateway Service. If the desire is to concurrently utilize more than one L2 Gateway node for bridging purposes, the recommendation is to deploy multiple L2 Gateway Services and associate a different pair of Gateway Nodes with each Gateway Service.

The exchange of CFM packets on the physical L2 network represents also an additional level of security against the so- called dual-active scenarios, in which both NSX L2 Gateways would become active at the same time.

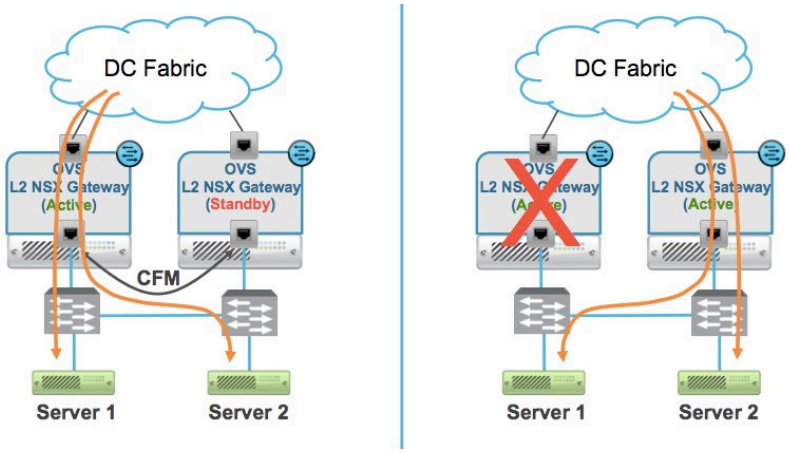


Figure 18: Split Brain Scenario

As shown in Figure 18, when the two NSX Gateways can exchange CFM packets, only one is in active state and

forwards traffic toward the physical servers (Server 1 and Server 2). The failure of the Active NSX Gateway (or any other link specific failure that will impact the capability of the active gateway to perform its functions) would cause the switchover and the previously standby NSX Gateway would become active. As soon as the failure is recovered, the two devices would start exchanging again CFM probes and revert to the initial Active/Standby state.

It is worth mentioning that once the Standby L2 Gateway becomes active, it must ensure that the MAC tables of the physical switches and hosts deployed in the L2 network can be updated (as they would still be pointing to the failed L2 gateway to reach MAC addresses connected in the logical space). In the current implementation this is achieved by generating a Reverse ARP (RARP) frame for each known MAC address connected to logical switches for which a L2 Gateway Service is configured.

Important: currently all logical switch to VLAN ID mappings associated with the L2 Gateway Services are active on the same NSX L2 Gateway appliance. This implies that the bridging function across different logical-switch/VLAN pairs cannot be load-balanced between the two L2 Gateway appliances. If needed, deploy multiple Gateway Services with a dedicated pair of NSX Gateways in each.

L2 Gateways also participate in 802.1D spanning tree protocol (STP) and monitor STP BPDU packets on the network to detect physical network topology changes that might lead to a loop. In addition to that:

- L2 Gateway Services Bridge-IDs are configured so that L2 Gateways are never elected as STP root bridges.
- If a L2 Gateway receives a BPDU with a new spanning tree root-bridge-ID, it assumes a topology change has occurred, and it temporarily stops forwarding traffic between the logical and physical networks in order to avoid potential loops (up to 60 sec).
- Because loop prevention is handled by the L2 Gateway as previously explained, BPDU packets are not forwarded by the L2 Gateway Services to the logical networks.

Hardware Partners L2 Gateway

Physical Top-of-Rack (ToR) switches provided by 3rd party VMware partners Hardware vendors can also be deployed to perform the L2 bridging functionality previously described for software-based NSX L2 Gateways. This essentially allows extending the overlay tunneling capabilities directly to the physical edge of the DC fabric network, building a sort of hybrid overlay connection between the physical switch and the virtual switch in the hypervisor.

The solution leveraging Hardware partners L2 gateways presents the following characteristics:

- The only tunneling protocol supported on those physical switches is VXLAN, which implies that a mix of different tunnels can be established to provide connectivity between virtual and physical workloads. STT is the preferred tunneling options between transport nodes (hypervisors running OVS in this example) supporting this protocol. The main reason to prefer STT to other encapsulation options (GRE or VXLAN) is because of the performance advantages it provides as a result of its ability to leverage TCP segmentation offload (TSO) when using supported and qualified NICs. Figure 19 highlights how those STT tunnels can coexist on a given logical switch with the VXLAN connections established between hypervisors and third-party Hardware L2 Gateways. The same transport connector on the hypervisors in the figure can be utilized to establish STT and VXLAN tunnels.

Note: for more information on STT, GRE and VXLAN Protocols please refer to the RFC Drafts below:

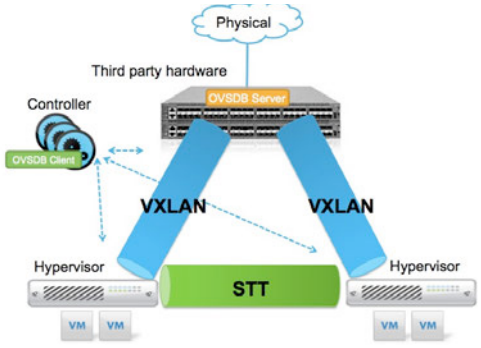


Figure 19: Mix of Tunneling Options between Transport Nodes

http://tools.ietf.org/html/draft-davie-stt-01
http://tools.ietf.org/html/draft-sridharan-virtualization-nvgre-01
http://tools.ietf.org/html/draft-mahalingam-dutt-dcops-vxlan-08

As depicted in figure above, the NSX Controller interacts with the ToR L2 Gateway in a similar fashion as it does with the other NSX transport nodes (hypervisors, for example). This is important because it allows NSX to consistently configure and manage the point of attachment of the endpoints to a given logical switch, independently from the nature (virtual or physical) of the workload itself. While the interaction between the NSX Controller and the hypervisors makes use of a management channel (OVSDB) and an OpenFlow channel (as previously shown in Figure 10), a single control plane channel (using OVSDB as a communication protocol) is used between the NSX Controller and the 3rd party Hardware L2 Gateway. This has two implications:

1. The Hardware switches vendors have to implement OVSDB support to be able to integrate with the NSX platform.

Note: OVSDB is an open standard draft and not a proprietary protocol. For more information please refre to: https://datatracker.ietf.org/doc/rfc7047/

2. The NSX Controller does not push any flow information to the Hardware L2 Gateways. This impacts the way communication can be established between a virtual and physical workload connected to the same logical switch, as it will be later clarified in the "Unicast Traffic (Virtual to Physical Communication)" section.

- The ToR L2 Gateway can offer more connections into the physical network, given the higher interface density characterizing those devices. However, a L2 Gateway Service must be defined for each physical interface (or bundle of physical interfaces), similarly to what was described for the NSX L2 Gateway appliance.
- Hardware L2 Gateways perform data plane learning of the MAC addresses associated with the physical hosts.

Differently from what previously discussed for transport nodes using OVS, the Hardware Gateways do not communicate any MAC information to the NSX Controller using the OVSDB control plane and as a consequence the other transport nodes rely on data-plane learning to map the MAC addresses of the physical hosts to the IP address identifying the L2 Gateway device.

NSX L3 Gateways with Open vSwitch

The deployment of L2 Gateway Services enables communication between virtual and physical endpoints belonging to the same logical switch (intra IP subnet communication). The use of NSX L3 Gateways is introduced to provide connectivity between workloads belonging to different L2 domains (IP subnets). Two common use cases where L3

Gateways are needed are:

- East-west L3 communication between workloads (virtual or physical) connected to different logical switches.
- North-south communication between workloads (virtual or physical) connected to logical switches and physical devices deployed in the L3 physical network infrastructure (WAN, Internet or Intranet).

The internal architecture of the NSX L3 Gateway is highlighted in Figure 20.

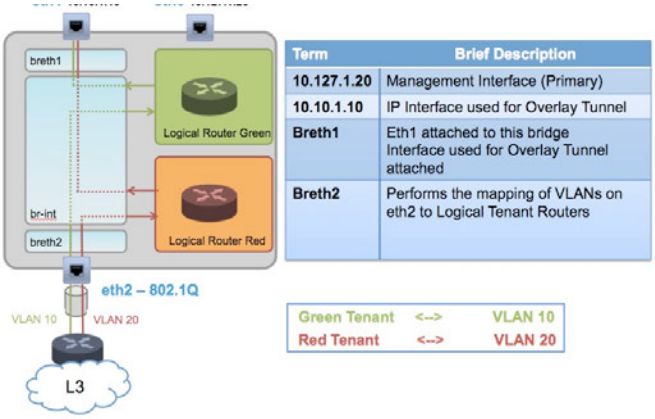


Figure 20: NSX L3 Gateway Internal Architecture

The main difference with the internal architecture of the NSX L2 Gateway (shown in Figure 16) consists in the use of a routing function (usually named Logical Router) to be dedicated to each supported tenant. The Logical Router is responsible for routing traffic between logical switches defined for a given tenant and between the tenant's logical space and the external network infrastructure.

As shown above, each L3 Gateway transport node connects directly to the external network using a physical NIC or bond of physical NICs (a Bridge-ID is exposed for each NIC/bond, for example breth2 for interface eth2). In multi- tenancy deployments, those connections to the external world can be configured as 802.1Q trunks, to be able to route traffic for each tenant toward the next-hop physical router maintaining logical isolation across tenants. The physical router may then implement functionalities like VRF or MPLS VPN to extend this logical isolation end-to-end across the L3 network infrastructure.

The first step to enable routing functionality in the logical space consists in defining a L3 Gateway Service. It is during this definition that a user can assign multiple L3 Gateway transport nodes to the same L3 Gateway Service, specifying also for each node what is the physical interface (or bundle of physical interfaces) connecting to the external L3 network (this connection is usually named "uplink").

Note: both physical (x86 bare metal servers) and virtual (virtual machines) appliances form factors can be deployed in production as NSX Gateways. For more specific requirements, please refer to the latest NSX-MH Release Note.

At this point it is possible to define the different logical routers responsible to perform routing between the tenants logical networks. As shown in Figure 21, for each tenant the NSX Controller chooses 2 Gateways for deploying a pair of Logical Routers in Active/Standby fashion. However, it is possible to spread those Active/Standby pairs of logical routers across the pool of 10 Gateway nodes associated with the same L3 Gateway Service (L3 Gateway Service pool), so that all the Gateway nodes can concurrently forward traffic for multiple tenants.

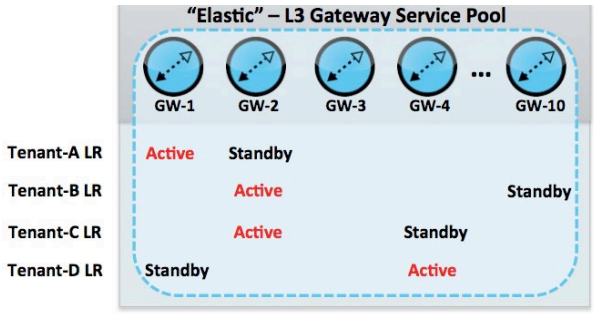


Figure 21: Deployment of a L3 Gateway Service Pool

Some characteristics of this "elastic" L3 Gateways Service pool are:

- The user has no control on the placement of each pair of Active/Standby Logical Routers and it is up to the NSX Controller to select the specific Gateway nodes to deploy them on.
- Currently, the load of Gateway node is not considered as a factor for influencing the placement of the LR instances. Multiple Logical Router Instances (Active or Standby) belonging to separate tenants can hence end up being deployed on the same L3 Gateway appliance. For example, the picture above shows 3 LRs deployed on GW-2 (one Standby and two Actives).
- While the NSX Controller is responsible for deploying a pair of Active/Standby Logical Routers for a given tenant and positioning them on different Gateway nodes, it is instead not at all involved in the mechanism that dictates the role (Active or Standby) of the Logical Routers pair. This decision is taken by the Logical Routers themselves, leveraging the same CFM protocol already introduced for the NSX L2 Gateways. The main difference is that here the exchange of the CFM probes happens across the STT tunnel built between the transport nodes hosting the LRs, and not via the physical infrastructure (Figure 22).

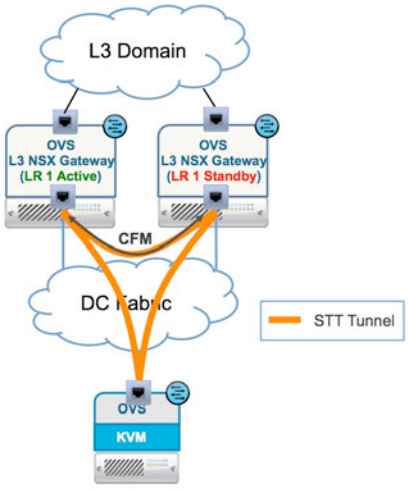


Figure 22: Pair of Active/Standby Logical Router Instances

Every remote transport node (a KVM hypervisor in this example) establishes STT tunnels with both the Active and Standby L3 Gateway nodes at the same time. This allows the transport node to be aware of the status of the remote L3 Gateway nodes.

- When a L3 Gateway node experiences a failure that prevents the local tenant logical routers from actively performing their duties, the Active role is took over by all the previously Standby logical routers deployed on other Gateway nodes. There are several failure scenarios that may cause the failover to happen:
  - L3 Gateway node failure.
  - Failure of the uplink interface toward the physical world.
  - Failure of the interface toward the DC Fabric (used to establish the overlay tunnels with the other transport nodes).
  - The Active L3 Gateway is set to “Admin Down” via configuration.
- After failover, the newly activated logical router has all the relevant routing and NAT state information (synced with the previously active Logical Router) and can start serving traffic received from remote transport nodes. Notice that the L3 Gateway node communicates to the other transport nodes the information about the activation of the Logical Router via control plane messages exchanged via the tunnel connection. Once the failed Gateway node (or its attachment interface) is back online, the local Logical Router preempts its original active role. This is the default behavior and cannot currently be modified by configuration.

- Based on the above considerations, if GW-2 in the example of Figure 21 were to fail, the following se-

quence of actions would happen:

1. Tenant-A Standby LR would be redeployed on another Gateway Node (chosen in the range GW-3 to GW-10).
2. Tenant-B LR would go active on node GW-10 and a new standby LR would be deployed on a different Gateway node.
3. Tenant-C LR would go active on node GW-4 and a new standby LR would be deployed on a different Gateway node.
4. Once the GW-2 node comes back online, the LRs for Tenants A, B and C will be redeployed on it to restore the initial situation shown in Figure 21 (before the failure).

The resiliency of the L3 Gateway Service can be further improved deploying the Gateway transport nodes as part of two separate pools, named "Failure Zones".

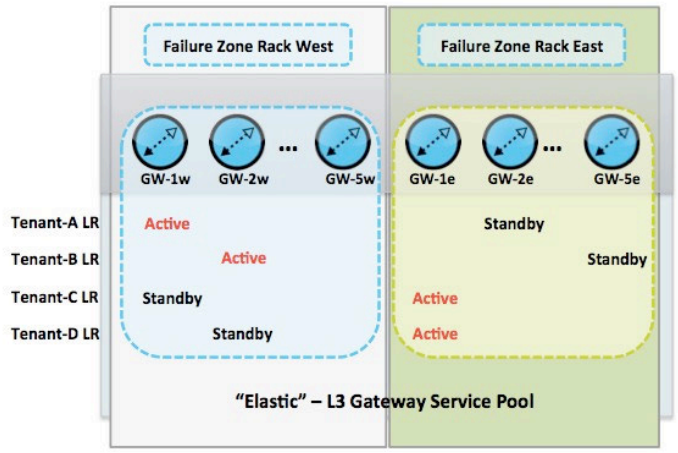


Figure 23: Deployment of L3 Gateway Nodes in Separate Failure Zones

As shown in Figure 23, each Failure Zone could be associated with a different server racks, so that the two Active/Standby LR instances assigned to a given tenant will always be deployed in those separate racks. As a

Transport Zone

In the simplest sense, a Transport Zone defines a collection of OVS devices (transport nodes - hypervisors, Service Nodes, L2 and L3 Gateways) that can communicate with each other across a physical network infrastructure. This communication happens leveraging one (or more) specific interface defined on each transport node and named "transport connector". Each transport connector is always associated with a specific transport zone used to carry overlay traffic sourced and received by that connector.

The definition of a transport zone allows the Controller Cluster to understand what transport connectors can communicate directly when implementing a logical network. For example, with an overlay logical network, the Controller Cluster must know which hypervisor IP addresses can be used to create direct tunnel connections. In other words, the transport zone determines the set of nodes on top of which logical networks can be defined and stretched.

The configuration of a transport connector on a given transport node includes also the specification of the overlay technology to be used. Multiple tunneling technologies (STT, VXLAN, GRE) can be simultaneously associated with a single transport connector, but in order to establish logical connectivity between workloads deployed in separate transport nodes it is mandatory for those transport nodes to utilize a common overlay technology in the transport zone connecting them.

Only a single transport connector can be defined and associated with a given transport zone. However, it is possible to define multiple transport connectors on a transport node and associate them to different transport zones. For example, if hypervisors have one interface for sending public Internet traffic and another interface for sending internal traffic, the deployment can include a public transport zone and a back-end (internal-only) transport zone.

Multi-Tier Application Deployment Example

The deployment of the NSX-MH Components described in the previous section is paramount to the agile and flexible creation of applications with their required network connectivity and services. A typical example is the creation of a multi-tier application, highlighted in Figure 24.

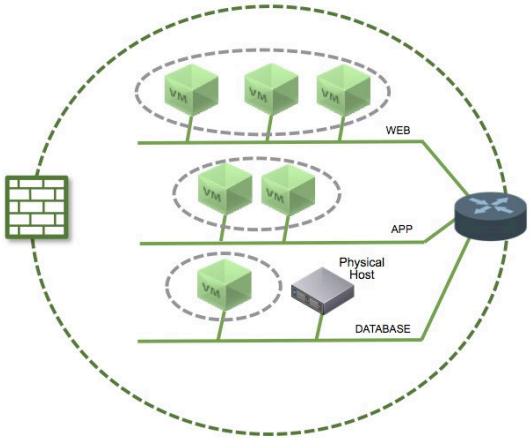


Figure 24: Deployment of a Multi-Tier Application

The following sections will discuss the functionalities (logical switching, routing, security services) required for the dynamic creation of a multi-tier application that are enabled by the deployment of the NSX-MH components.

Logical Switching

The logical switching capability in the NSX-MH platform provides customers the ability to spin up isolated logical L2 networks with the same flexibility and agility, as it is to spin up virtual machines. Endpoints, both virtual and physical, can then connect to those logical segments and establish connectivity independently from the specific location where they are deployed in the Data Center network. This is possible because of the decoupling between network infrastructure and logical networks (i.e. underlay and overlay networks) made possible by NSX network virtualization.

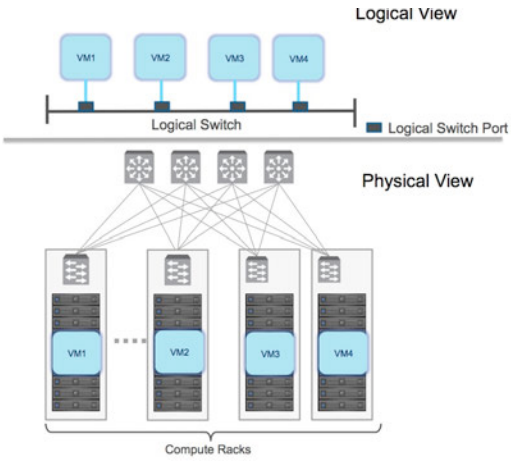


Figure 25: Deployment of a Multi-Tier Application

Figure 25 displays the logical and physical network views when logical switching is deployed leveraging the overlay technologies (STT, GRE, VXLAN) that allow stretching a L2 domain (logical switch) across multiple server racks, independently from the underlay inter-rack connectivity (L2 or L3).

With reference to the example of the deployment of the multi-tier application previously discussed, the logical switching function allows to create the different L2 segments mapped to the different tiers where the various workloads (virtual machines or physical hosts) are connected.

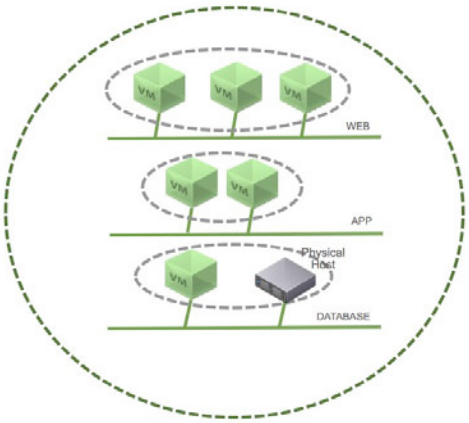


Figure 26: Creation of Logical Switches for the App Tiers

It is worth noticing that logical switching functionality must enable both virtual-to-virtual and virtual-to-physical communication in each segment and that the use of L2 Gateways is also required to allow connectivity to the logical space to physical nodes, as it is often the case for the DB tier.

Unicast Traffic (Virtual to Virtual Communication)

Before being able to establish intra-logical switch data-plane communication, the Virtual Machines need to populate their ARP caches, following the steps highlighted in Figure 27.

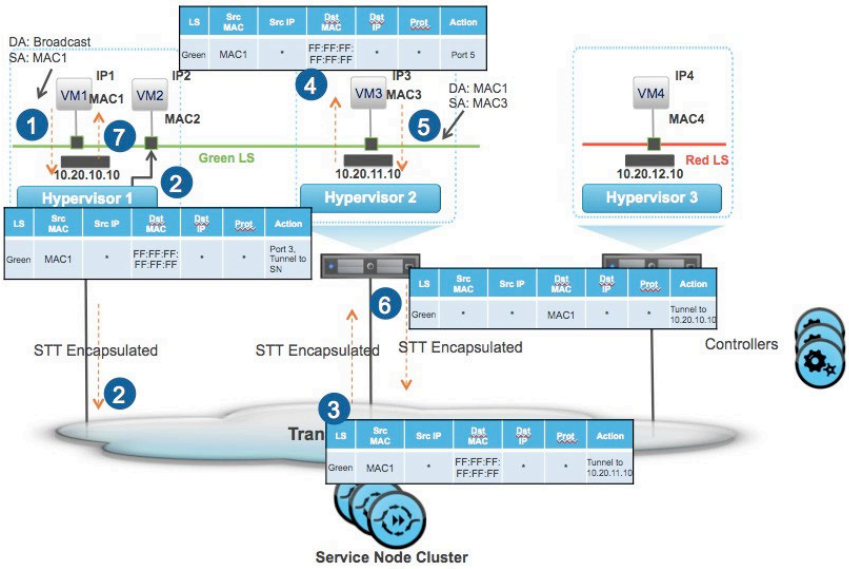


Figure 27: ARP Exchange between Virtual Machines

1. VM1 belonging to the Green Logical Switch (LS) wants to communicate with VM3, part of the same LS but deployed in a different hypervisor. VM1 originates an ARP request to determine the MAC/IP address mapping information for VM3.
2. The ARP request is received by the OVS on Hypervisor 1: since it is a L2 broadcast packet, it must be sent to all the local VMs connected to the Green LS (VM2 in this example, connected to the local port 3) and also encapsulated and sent to the workload part of the same LS and connected to remote transport nodes. In this example it is introduced the use of the Service Node to perform the replication of L2 broadcast traffic. The OVS in Hypervisor 1 must hence encapsulate the L2 broadcast ARP request in the overlay technology of choice and send it to the IP address identifying the transport connector of a specific Service Node, part of the SN cluster. As shown above, a specific flow entry is installed in the kernel space of the OVS of Hypervisor 1 for this purpose.

Note: multi-destination traffic can also be replicated directly by the hypervisor, instead of relying on the Service Node. For more information please refer to the "Replication Modes for Multi-Destination Traffic" section.

3. The Service Node replicates the ARP request to all the hypervisors that have at least one active switch port in the Green LS. The NSX Controller has full visibility about the logical switches that have been defined and the corresponding active logical switch ports and provides this information to the Service Nodes (in terms of specific flow entries). In this specific case, the frame must be replicated to Hypervisor 2 (identified by the IP address 10.20.11.10) but not to Hypervisor 3, since the latter has only a VM connected to a different Red LS. Notice also that the Service Node performs RPF check to avoid sending the packet back to the originating Hypervisor 1.
4. Hypervisor 2 receives the ARP request and forwards it to all local VMs connected to the Green LS (only VM3 in the example, connected to the local port 5).
5. VM3 receives the ARP request, populates its ARP-cache with the MAC1/IP1 mapping information and generates a unicast ARP reply directed to VM1.
6. The ARP reply is STT-encapsulated by the OVS on Hypervisor 2 and sent to the connector IP address identifying Hypervisor 1.
7. ARP reply is de-capsulated and sent to VM1, which is hence able to populate VM3 mapping information in its ARP cache.

Once the ARP cache is populated, data-plane communication between the Virtual Machines can be established, as described in Figure 28.

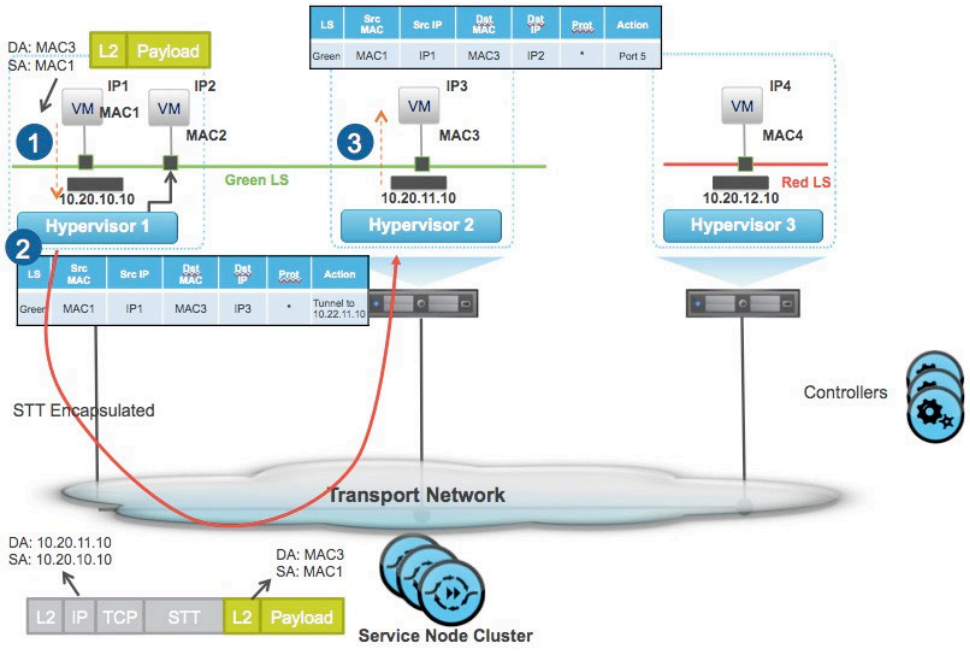


Figure 28: Unicast Data-Plane Communication between Virtual Machines

1. VM1 generates a data packet destined to VM3.

2. The OVS on Hypervisor 1 has a specific flow entry in the kernel instructing to encapsulate the frame and send it to the transport connector IP address identifying Hypervisor 2 (10.20.11.10).

3. OVS on Hypervisor 2 unencapsulates the packet and send it to VM3, connected to the local port 5. The same process happens in the opposite direction, allowing for the establishment of bidirectional unicast-communication between VM1 and VM3.

Unicast Traffic (Virtual to Physical Communication)

As previously discussed, the establishment of unicast communication between virtual and physical workloads part of the same L2 broadcast domain can be achieved by deploying L2 Gateways. Figure 29 shows how physical servers can gain access to a logical network via the bridging services offered by a L2 Gateway.

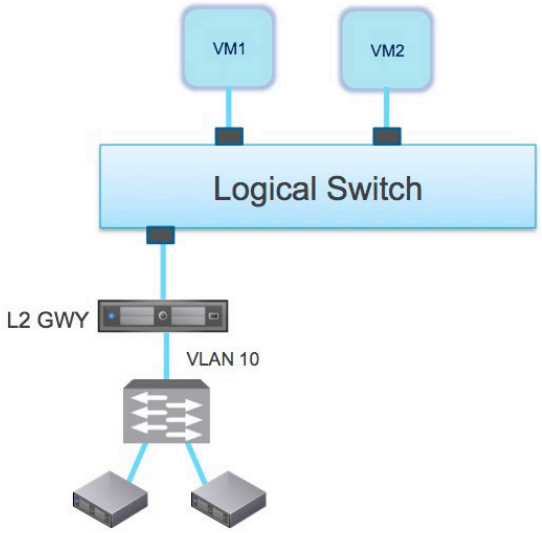


Figure 29: Physical to Virtual Communication

It is worth noticing that multiple L2 Gateway Services can be simultaneously enabled for the same Logical Switch and bridging to different VLAN IDs is supported as well (providing a sort of "VLAN translation" service

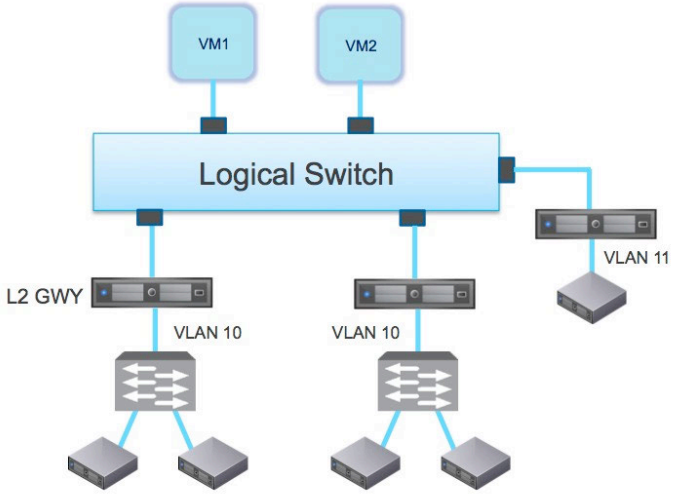


Figure 30: Concurrent Deployment of Multiple L2 Gateway Services

Note: if a L2 backdoor connection was established between the L2 physical switches shown in Figure 30, the two L2 Gateway would discover each other (leveraging the CFM probes) and one of them would become Standby to avoid the possible creation of L2 loops. All the physical servers would still achieve connectivity to the logical switch through the remaining Active L2 Gateway.

When leveraging NSX L2 Gateways, the behavior is similar to the virtual-to-virtual communication discussed in the previous section. This is because also the NSX L2 Gateway transport node leverages an OVS vSwitch, so similarly to hypervisors flow entries are installed in the kernel space to forward traffic between endpoints. The only difference is the fact that the NSX L2 Gateway does not communicate to the Controller any information about the MAC of physical hosts it provides communication to. As a consequence, the remote hypervisors use data plane learning to map the MAC address of those physical hosts with the IP address associated with the transport connector interface of the NSX L2 Gateway.

The behavior is a little bit different when deploying Hardware ToR switches as L2 Gateways. Figure 31 and Figure 32 shows how the ARP exchange is performed in this case.

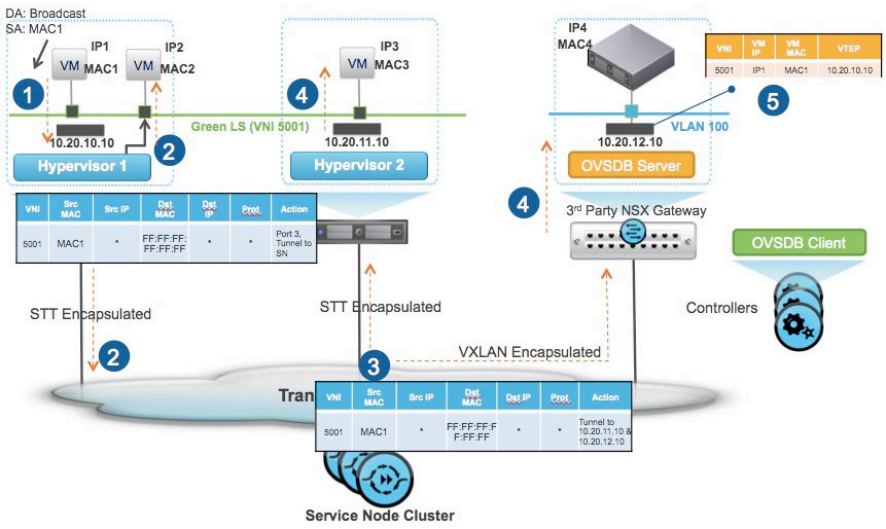


Figure 31: ARP Exchange between Virtual and Physical Endpoints (Part 1)

Virtual Chassis Fabric Switching Architecture

1. VM1 belonging to the Green LS (identified by the VXLAN Network Identifier - VNI 5001) originates an ARP request to determine the MAC/IP address mapping information for the physical workload deployed in VLAN 100 and part of the same L2 domain (IP subnet).
2. The ARP request is received by the OVS on Hypervisor 1: since it is a L2 broadcast packet, it is sent to all the local VMs connected to the same LS (VM2 in this example, connected to local port 3) and also encapsulated and sent to the IP address identifying the transport connector of the specific Service Node responsible for the replication of multi-destination traffic. A flow entry is installed in the kernel space of the OVS for directing L2 broadcast frames to the Service Node.
3. The Service Node replicates the ARP request to Hypervisor 2 (using an STT encapsulation) since VM3 is actively connected to the same Green LS. In addition to that, the Service Node sends a VXLAN encapsulated copy of the ARP request also to the Hardware ToR L2 Gateway, since a L2 Gateway Service is active to provide communication to the physical workload deployed in VLAN 100. As in the previous scenario, the Service Node performs RPF check to avoid sending the packet back to the originating Hypervisor 1.

Note: in order to allow the behavior described above, both STT and VXLAN encapsulations need to be enabled on the transport connector of the Service Node.

4. Hypervisor 2 receives the ARP request and forwards it to all local VMs connected to the Green LS (only VM3 in the example). The Hardware ToR L2 Gateway also receives the ARP request, unencapsulates it and sends it to the physical server.

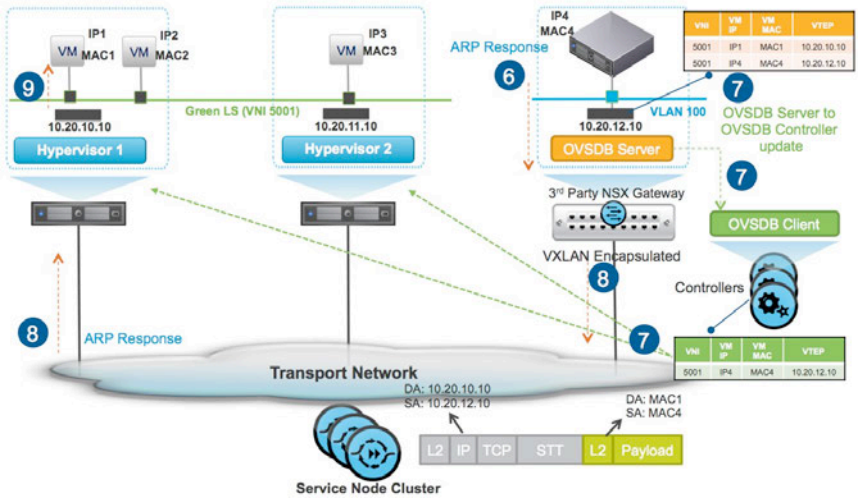


Figure 32: ARP Exchange between Virtual and Physical Endpoints (Part 2)

The Physical server receives the broadcast ARP request, populates its ARP cache with the MAC1/IP1 mapping entry and generates the unicast ARP reply, destined to MAC1.

7. The reception of the ARP Reply by the ToR Switch triggers several control-plane actions (this is true assuming this is the first frame generated by the physical workload):

- The switch adds to its MAC Address Table the information that MAC4 is locally reachable via the ToR physical interface.
- The same information is communicated to the NSX Controller via the OVSDB control plane. This is important, because the Controller is not aware of physical hosts connectivity (the same way it is of virtual machines) and relies on those OVSDB notifications.
- The NSX Controller utilizes the OpenFlow control plane channel to send flow information about the newly discovered physical machine to all the NSX transport nodes.

8. On the data plane, the ToR switch encapsulates the ARP reply into a VXLAN packet destined to the Locator IP address of Hypervisor 1.

Note: as mentioned for the Service Node, it is required that the Hypervisor transport nodes specify both STT and VXLAN encapsulations on their transport connectors.

9. Hypervisor 1 unencapsulates the frame and sends the ARP reply to VM1, which then populates its ARP cache with the received mapping information.

Once the ARP caches on the endpoints are populated, data plane communication can be established between virtual and physical endpoints (Figure 33).

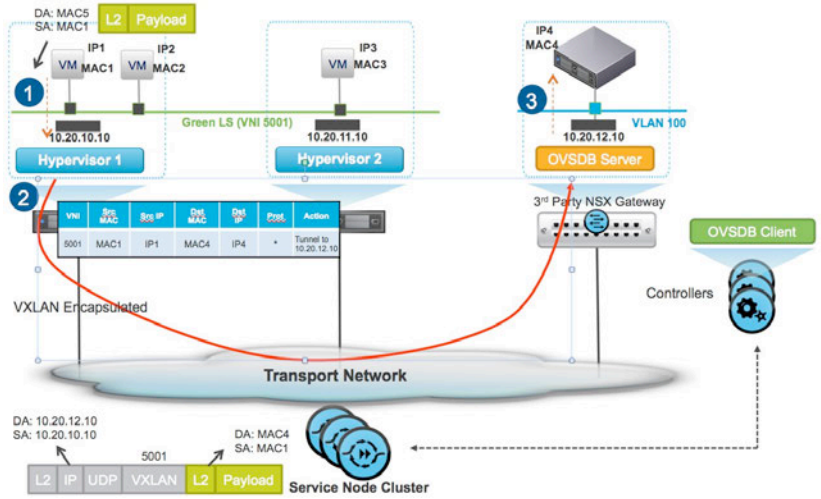


Figure 33: Unicast Data Plane Communication between Virtual and Physical Endpoints

1. VM1 generates a data packet destined to the physical workload.
2. The OVS on Hypervisor 1 installs a specific flow entry instructing to encapsulate the frame and send it to the IP address identifying the transport connector of the Hardware ToR L2 Gateway. VXLAN is the encapsulation utilized between the hypervisor and the physical switch.
3. The ToR switch unencapsulates the packet and sends it to the physical endpoint directly connected in VLAN 100.

The same process happens in the opposite direction, allowing for the establishment of bidirectional unicast communication between the virtual and physical endpoints.

Replication Modes for Multi-Destination Traffic

As discussed in detail in the previous sections, when two endpoints part of the same logical switch and connected to different transport nodes need to communicate directly between them, unicast encapsulated traffic is exchanged between the IP addresses identifying the transport connectors of each transport node. Sometimes, traffic originated by an endpoint needs to be sent to all the other endpoints part of the same L2 domain. This is specifically the case for three types of L2 traffic:

- Broadcast (for example ARP requests, as discussed in the examples in the previous sections)
- Unknown Unicast
- Multicast (currently treated as Broadcast)

Note: usually we refer to these types of multi-destination traffic types using the acronym BUM (Broadcast, Unknown unicast, Multicast). It is also worth noticing that in a NSX deployment, there should never be a need to flood unknown unicast traffic for communication between virtual machines. This is because the NSX Controller is always aware of the connectivity of every virtual machine and provides flow specific information to all the transport nodes. This may be not the case for virtual to physical communication, in the scenario where the physical machine is connected to an Hardware ToR L2 Gateway, since as previously mentioned the NSX Controller is not aware of the physical machine existence until it receives this information via the OVSDB control channel.

In any of the three scenarios mentioned above, traffic originated by an endpoint connected to a given transport node may need to be replicated to multiple remote transport nodes (hosting other endpoints part of the same L2 segment). NSX supports two different replications modes that can be selectively configured on a per logical switch level: Service Node Replication and Source-based Replication.

Figure 34 shows again the already presented behavior when a Service Node is utilized for replicating multi-destination traffic.

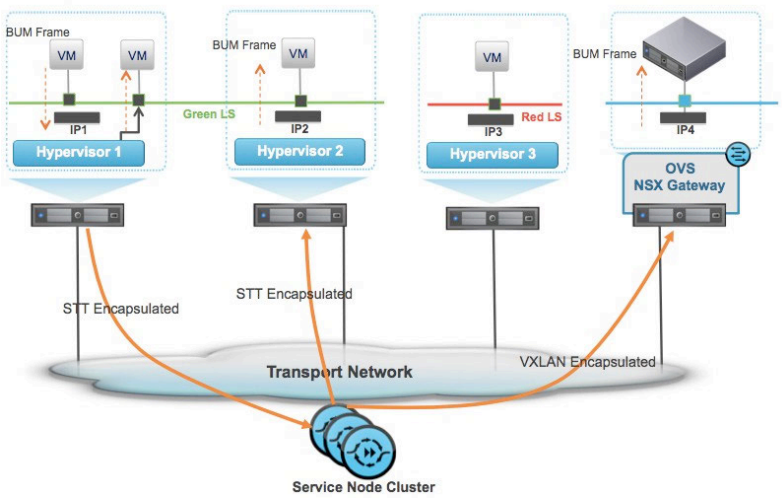


Figure 34: Service Node Replication

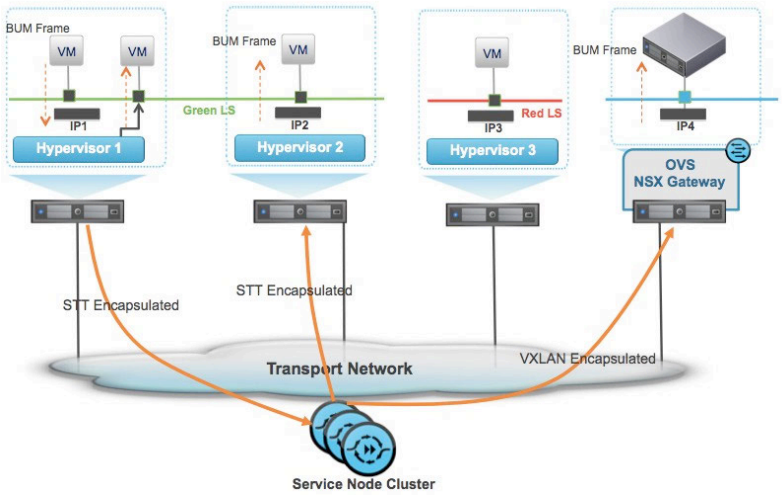


Figure 35: Service Node Replication

In this case, the transport node (hypervisor or NSX L2 Gateway) takes the task of creating multiple copies of the BUM frame and sending them to all the other transport nodes. Each transport node is receiving logical switch information from the Controller so to be able to replicate traffic to all the hypervisors that have virtual workloads actively connected to that given logical switch. In addition, the source transport node must always replicate the frames to the Software and Hardware L2 Gateways, as it may be possible that there are physical hosts connected to the VLAN mapped to the Green LS and not yet discovered.

Logical Routing

The Logical Routing capability in the NSX platform provides customers the ability to interconnect endpoints (virtual and physical) deployed in different logical L2 networks. Once again, this is possible due to the decoupling between network infrastructure and logical networks provided by the deployment of network virtualization.

Figure 36 shows the logical view of the routed topology interconnecting two logical switches, and the corresponding physical view. More considerations around the deployment of servers and appliances as part of compute racks and edge racks can be found in the "Network Virtualization Design Considerations" section.

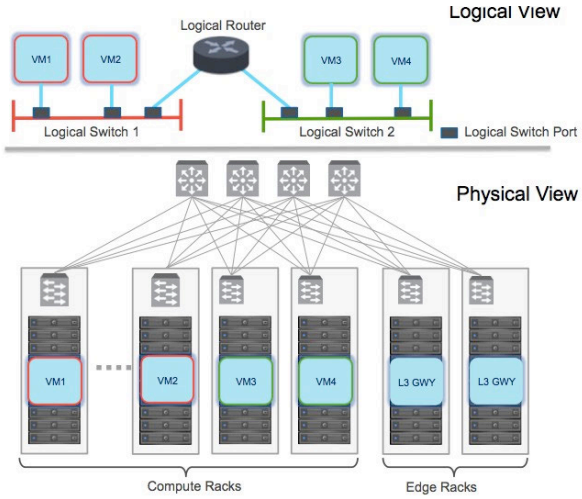


Figure 36: Logical Routing (Logical and Physical Network Views)

Notice that only a single Logical Router can be actively connected to a given logical switch. The redundancy of the routing services is provided by the creation of an Active/Standby pair of Logical Routing instances, as described in the "NSX L3 Gateways with Open vSwitch" section. Also, a single interface (named "uplink interface") is currently supported to connect the Logical Router to the upstream physical next-hop router. This interface can be a physical NIC or a set of physical NICs bundled together in a logical port-channel.

The deployment of Logical Routing can serve two purposes: interconnecting endpoints (logical or physical) belonging to separate logical L2 domains (as shown in figure above) or interconnecting endpoints belonging to logical L2 domains with devices deployed in the external L3 physical infrastructure. The first type of communication, usually confined inside the Data Center, is referred to as "east-west" communication; the second one is called "north-south" communication and provides connectivity into the Data Center from the external physical world (WAN, Internet or Intranet).

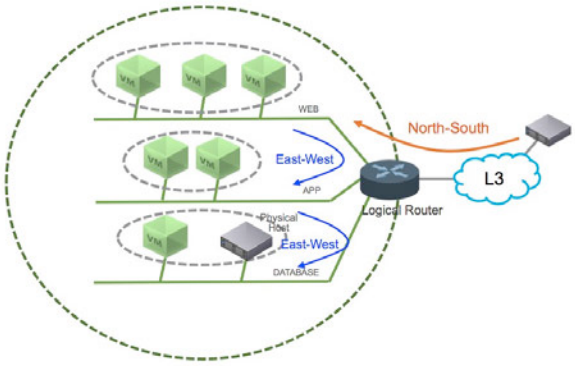


Figure 37: Logical Routing for a Multi-Tier Application

Logical routing services for a given tenant can be deployed in two modes, distributed and centralized. The following two sections will provide more details on those two modes and also describe some common routing topologies that can be built in the customer's environment.

Centralized Routing

The NSX L3 Gateway provides the traditional centralized routing support in the NSX platform. Along with the routing services the NSX L3 Gateway currently also supports NAT network services. A centralized routing deployment can be utilized both for east-west and north-south routed communications, as shown in Figure 38.

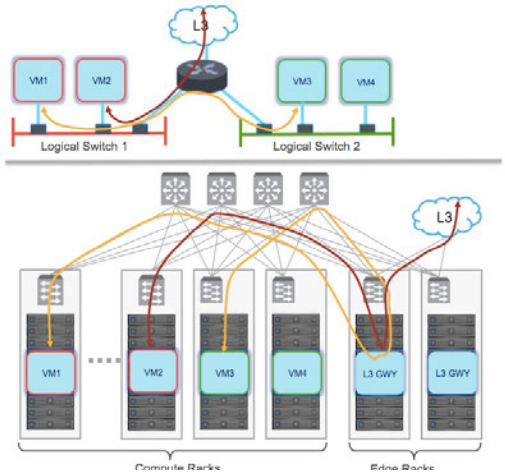


Figure 38: Centralized Routing

East-west routed flows are not optimized in a centralized routing deployment, since traffic is always hair-pinned from the compute racks toward the edge rack where the active Logical Router is deployed. This is the case even when two virtual machines belonging to separate logical switches are deployed inside the same hypervisor.

The ARP exchange with the centralized Logical Router needed for establishing communications between two workloads connected to different logical switches is highlighted in Figure 39.

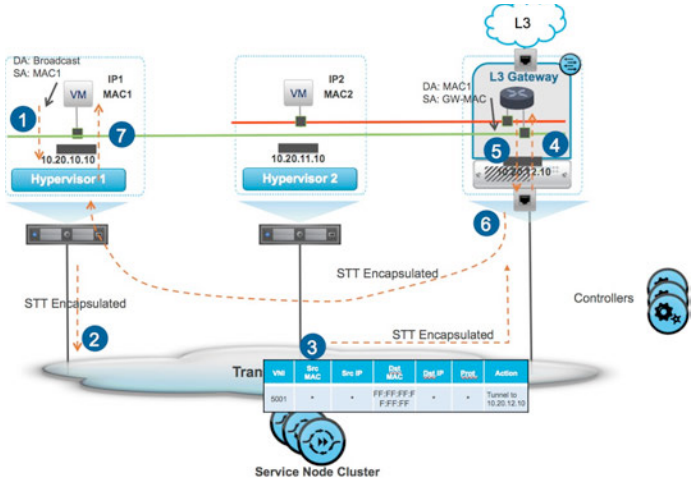


Figure 39: ARP Exchange with the Centralized Logical Router

1. VM1 belonging to the LS wants to communicate with VM2, part of a different Red LS and deployed in a different hypervisor (as previously stated, the behavior would be the same even if the two VMs were both connected to Hypervisor 1). VM1 originates an ARP request to determine the MAC/IP address mapping information for its default gateway, located on the Logical Router interface on the Green LS.
2. The ARP request is sent to the Service Node that is in charge of performing the replication of multi-destination traffic.
3. The Service Node replicates the ARP request to all the hypervisors that have at least one active switch port in the Green LS and to the NSX L3 Gateway node hosting the active Logical Router.
4. The L3 Gateway unencapsulates the frame and passes it to the Logical Router instance, which populates its ARP cache with the IP1/MAC1.
5. The Logical Router generates the unicast ARP reply destined to VM1.
6. The ARP reply is directly tunneled to Hypervisor 1.
7. Hypervisor 1 unencapsulates the frame and send it to VM1.

At this point, VM1 generates a data packet destined to VM2 and the sequence of events shown in Figure 40 happens

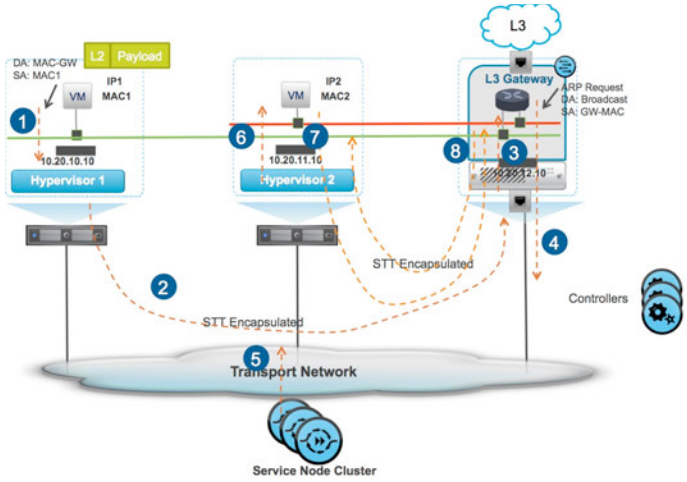


Figure 40: Inter IP Subnet Communication between VM1 and VM2

1. VM1 generates a data packet destined to VM2.
2. The packet is encapsulated and sent to the L3 Gateway node hosting the Logical Router that represents the default gateway for VM1 (using the ARP information previously learned).
3. The L3 Gateway unencapsulates the packet and sends it to the logical router that performs routing between the directly connected IP subnets associated with the Green and Red Logical Switches.
4. Assuming it is the first time the Logical Router tries to communicate with VM2, it must generate an ARP Request to resolve the MAC2/IP2 mapping. The ARP Request is sent to the Service Node.
5. The Service Node replicates the frame to Hypervisor 2 (in this example the only transport node hosting active VMs in the Red LS).
6. Hypervisor 2 unencapsulates the frame and sends it to VM2.
7. VM2 replies with the usual unicast ARP Response destined to the default gateway MAC. The packet is encapsulated by Hypervisor 2 and sent to the L3 Gateway node that receives it and passes it to the Logical Router.
8. After populating MAC2/IP2 information in its ARP cache, the Logical Router is able of sending the original data frame toward VM2, allowing for the completion of the traffic flow between VM1 and VM2.

The same process happens in the opposite direction, allowing for the establishment of bidirectional unicast communication between the virtual machines.

Note: a similar process is required to establish L3 communication between a VM and a physical device in the L3 network domain (north-south L3 communication).

Distributed Routing

The distributed routing capability in the NSX platform provides an optimized and scalable way of handling East - West traffic within a data center, which is all the backend traffic between virtual machines or between virtual and physical machines that makes applications work. The amount of East West traffic in the data center is growing. The new collaborative, distributed, and service oriented application architecture demands higher bandwidth and low-latency for server-to-server communication, hence it makes sense to perform routing in a distributed fashion, as close to the workload as possible. With a simple analogy familiar to the networking people, the deployment of distributed routing on each hypervisor is like configuring and managing one logical router chassis, where each hypervisor host performs like a logical line card.

As highlighted in Figure 41, the deployment of distributed routing prevents the “hair-pinning” for VM-to-VM routed communication by providing hypervisor level routing functionality. Each hypervisor installs in the kernel specific flow information (received from the NSX Controller) ensuring a direct communication path even when the endpoints belong to separate logical switches (IP subnets). Routing between VMs belonging to logical switches and the external L3 network domain happens instead identically to the centralized routing mode.

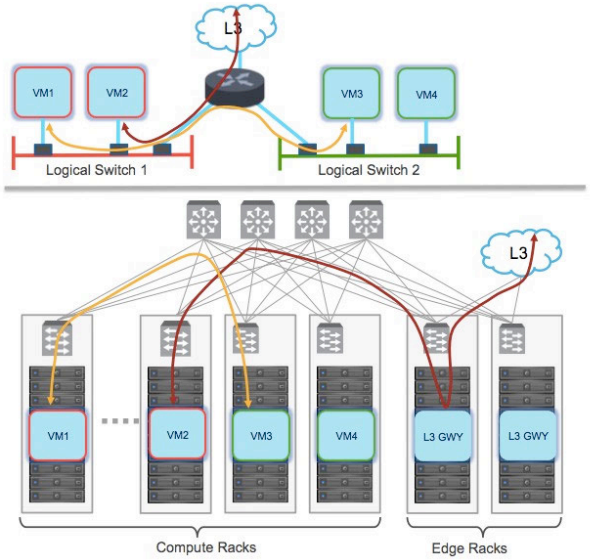


Figure 41: Distributed Routing to Optimize East-West Flows

The behavior shown above is possible by leveraging a dedicated functional element (called "l3d daemon") that implements router functionality on the OVS switch available on the transport nodes (reference previous Figure 15). For the sake of simplification, it is possible to think about this daemon like a "logical router" available on each transport node and able to handle routing flows in a similar manner to the central logical router discussed in the previous section. The NSX Controller installs flows on OVS switch to ensure traffic can be properly directed to this local routing engine.

Below are few additional important considerations when deploying distributed routing:

- It can be deployed concurrently with software NSX L2 Gateways, allowing building the logical topology shown in Figure 42.

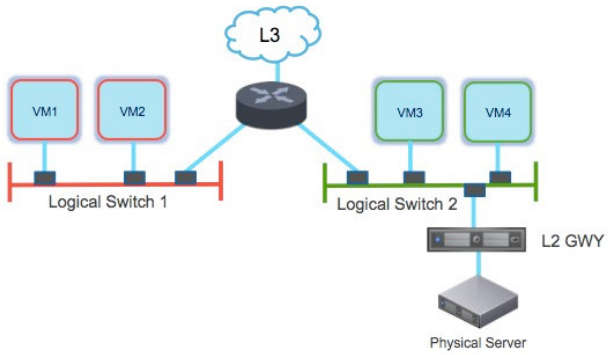


Figure 42: Coexistence of Distributed Routing and L2 Gateway

The NSX L2 Gateway has the same OVS switch found on the hypervisor transport nodes, hence it uses the localized distributed l3d daemon previously discussed to perform local routing directly on the L2 Gateway transport node.

- It is currently not supported when utilizing Hardware L2 Gateways. In that case, the only option is to revert back to a centralized routing deployment.
- A distributed router that spans a large number of transport nodes (hypervisors and Gateways) can impose significant overhead on the Controller Cluster, both in terms of computation and memory. This large span can be a result of either connecting a few large switches to a distributed logical router or connecting a large number of small switches to a distributed logical router. Splitting a large distributed logical router into a few smaller ones (subject to the maximum number of distributed routers supported by NSX) can reduce this pressure on the Controller Cluster.

- NAT cannot be enabled for east-west traffic when deploying routing in distributed mode. However, since NAT is more common for north-south communication and since north-south communication happens in the same fashion than in centralized routing mode, it is still possible to perform NAT for this type of traffic flows.

Security Features

NSX provides several features to secure communications between workloads deployed on the same logical switch, across logical switches and also between virtual and physical workloads. These features are:

- Port Isolation: functionality similar to Private VLANs, allowing controlling communication between virtual workloads deployed as part of the same logical switch (IP subnet).
- Port Security: used to restrict to a set of approved MAC and IP addresses the traffic that can be sent/ received by a workload connected to a logical switch port.
- Access Control Lists (ACLs): stateful and stateless ACLs can be associated with logical router ports, L2 gateway ports and VIF ports (logical switch interfaces where the virtual workloads are connected). This last option is currently delivered with a functionality called "Security Profiles", which is a role-based L3/L4 firewall capability similar to the "Security Group" primitive used by many Cloud Management Systems.

As it will be clarified in the following sections, Port Isolation, Port Security, and Access Control Lists should be thought of as different stages in the logical pipeline, meaning that when concurrently enabled traffic must pass all filters in order to be permitted.

The deployment of security features is the last functional element to be added for our example of deployment of a multi- tier, as represented in Figure 43 below.

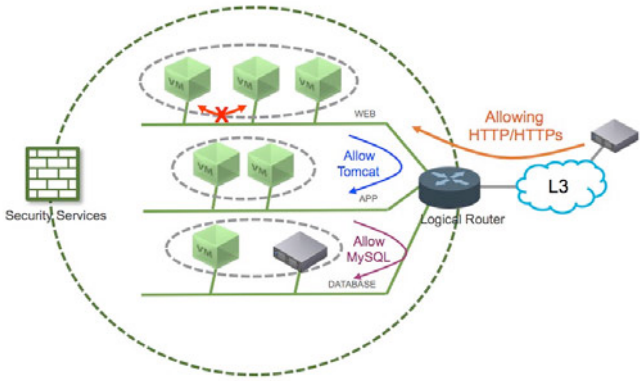


Figure 43: Controlling Communication in a Multi-Tier Application

The example in figure above highlights a typical traffic-filtering schema applied to a multi-tier application deployment. More specifically:

- Limit external communication between the external world and the WEB tier to allow only HTTP/HTTPS connections.
- Control intra-subnet communication on the Web Tier, since there is no requirement for virtual machines in the web tier to communicate intra-subnet.
- Limit communication between the WEB and the APP tiers (only allowing Tomcat).
- Limit communication between the APP and DB tiers (only allowing MySQL).

The following sections will discuss more in details the NSX-MH security services and functionalities that can be utilized to achieve such traffic filtering behavior.

Port Isolation

Port isolation is a feature designed to provide logical isolation at L2 between virtual workloads connected to the same logical switch (usually associated with the same IP subnet). In this, it is very similar to a Private VLANs (PVLANs) deployment and the end result is the provisioning of a "hub-and-spoke" type of connectivity where virtual workloads belonging to a given logical switch can only communicate with virtual machines connected to different logical switches or with physical hosts accessible via a L2 Gateway device (Figure 44).

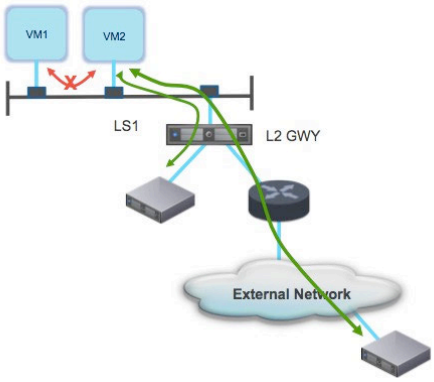


Figure 44: Impact of Port Isolation on Traffic Flows

As a consequence, Port Isolation is a L2 feature that is configured at the logical switch level. When it is enabled, the NSX Controller pushes to the hypervisors all the flow information required to avoid intra logical switch communication. More specifically:

- Traffic originated from or destined to known MAC addresses belonging to that logical switch will always be dropped.
- Traffic originated from or destined to MAC addresses not belonging to that logical switch will be allowed.

Important Note: due to a current issue (as of release 4.1.0), enabling Port isolation on a logical switch prevents any communication between the LS and a Logical Router. This implies that the only allowed communications are with workloads connected via a L2 Gateway logical port, as shown in Figure 44.

Enabling port isolation on logical switches does not come for free and has an impact on the maximum number of logical ports supported on that logical switch. For example, on the current software release 4.1.0, the maximum number of logical ports supported in conjunction with port security goes down to 64 (be sure to check out the latest NSX Release Note for updated information).

Because of its nature, the Port Isolation feature does not protect from malicious users inject traffic into a logical switch from a device leveraging a source MAC address unknown to the system. The NSX Port Security feature comes to the rescue to protect against this specific scenario.

Port Security

Port Security configuration can be applied to a logical switch port to specify a set of MAC and IP addresses pairs whose packets can be sent and received through that port. This is done to prevent MAC spoofing attacks or man-in-the-middle attacks usually carried out by performing ARP/IP spoofing. This is important for cloud network deployments where VMs from multiple tenants are connected to the same layer-2 network and are thus vulnerable to such types of attacks.

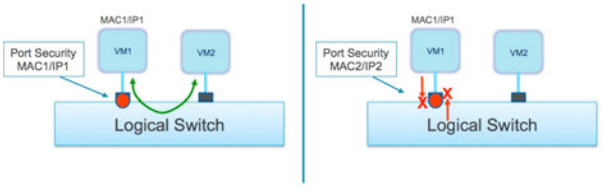


Figure 45: Port Security Configuration

As shown in Figure 45, the application of Port Security to the switch port where VM1 is connected (in this example by specifying the MAC1/IP1 pair) allows achieving the following behavior:

- On logical port ingress (traffic flowing from VM1 into the logical switch), traffic will be dropped unless its source MAC and source IP match the allowed MAC1/IP1 address pair.
- On logical port egress (traffic flowing from the data center network to one or more VMs), a packet will be dropped unless its destination MAC and destination IP match an allowed address pair.

This means that VM1 can successfully communicate with other devices connected to the same logical switch (or other logical switches). On the right part of Figure 45 is shown what would happen if Port Security was configured to allow the MAC2/IP2 pair: in this case, all the traffic sourced by VM1 would be dropped before entering the logical switch and the same would happen for traffic destined to VM1 and originated from other VMs.

Specific considerations apply to the scenario where Port Security is applied to a VIF port and the connected VM leverages DHCP to get assigned an IP address. In this case it is required to modify the Port Security configuration to ensure that the DHCP process happens successfully. This is because the DHCP request originated by the VM consists in an UDP packet with a source IP address of 0.0.0.0; as a consequence, the Port Security policy should specify not only the MAC/IP address pair identifying the workload, but also a MAC/IP address pair using 0.0.0.0 as IP value.

However, a second question comes to mind in DHCP scenarios: since the VM is going to receive an IP address dynamically, how can the MAC/IP address pair be configured to allow the VM to communicate after it has received the IP address? This can't obviously be done before the DHCP exchange is completed, since there is no certainty on what IP address will be assigned to the VM. It's important to keep in mind that in most real life deployment scenarios, a Cloud Management System (CMS) will be deployed to consume the network virtualization functionalities provided by NSX. Taking the example of OpenStack as CMS, it will be responsible to create the compute resources (Virtual Machine) and connect them to the created logical networks. This means that most likely the DHCP services will also be run as part of OpenStack itself (in the Neutron functional component), and this provides OpenStack full visibility to what IP address

will be assigned to each VM, allowing it to interact with NSX (through the REST APIs exposed by the NSX Controller) and configure the proper MAC/IP address pair before the VM gets assigned its IP address.

In deployment scenarios where customers decide to use an external DHCP server in conjunction with their CMS of choice, integration is required between the DHCP service and the CMS (in a similar fashion to what discussed above for OpenStack/Neutron) to allow the following to happen:

- The CMS has to tell the DHCP server about the new MAC address associated with the VM that will be instantiated.
- The DHCP server has to tell the CMS what IP address is being reserved for that VM (MAC to IP mapping)
- The CMS has to call the proper NSX Controller APIs to configure the right MAC/IP address pair for the logical switch port where the VM is going to be connected.

Finally, if the Port Security configuration is applied to the logical switch port where a Virtual Machine is connected (Virtual Interface - VIF), the immediate consequence is that it becomes propriety associated with the Virtual Machine itself. This means that if the Virtual Machine is migrated between two different hypervisors (vMotion), the security policy is also moved with the workload in a consistent fashion.

Access Control Lists (ACLs)

NSX-MH also provides the capability of securing communications between workloads by implementing filtering traffic techniques similar to the Access Control Lists commonly found on networking devices. Depending on where traffic filtering needs to be enforced, a different type of configuration is required:

- Implementation of L3/L4 filtering between virtual workloads connected to logical switch ports (VIFs) is currently possible leveraging a logical construct named Security Profiles.
- L3/L4 traffic filtering can also be applied to logical switch ports where NSX L2/L3 Gateway devices are connect. In this case, the logical construct takes the commonly know name of ACLs.

Figure 46 below highlights the logical places where those traffic-filtering functionalities (discussed in more detail in the following two sections) are applied.

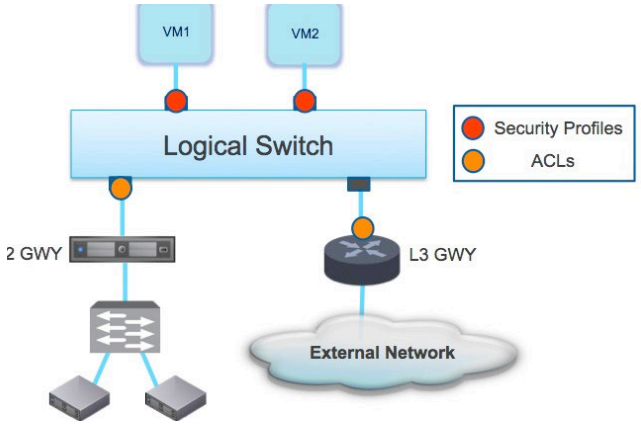


Figure 46: Application of Security Profiles and ACLs

Security Profiles

Security Profiles represent a logical construct very similar to the "Security Groups" created by most Cloud Management Systems (like OpenStack), and provide a simple role-based model for defining a set of filtering rules that are applied on a particular logical switch port.

Each Security Profile defines a set of egress rules (traffic leaving the logical switch port toward the VM) and ingress rules (traffic leaving the VM and arriving at the logical switch). A logical switch port can be associated with multiple Security Profiles, and the resulting security filters allow traffic that matches any of the rules, while denying all other traffic. If a port has no Security Profile associated with it, then no filtering will be applied

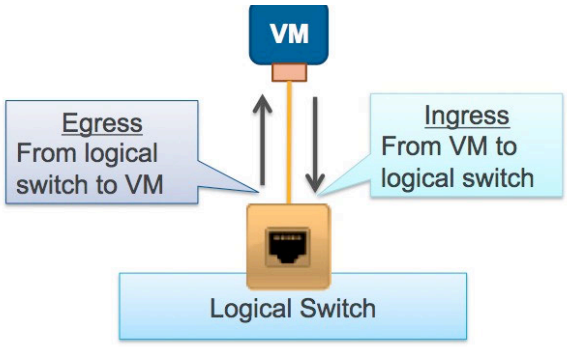


Figure 47: Logical Port Ingress and Egress Filtering

As highlighted in Figure 47 above, when working with Security Profiles, is important to understand that the "ingress" and "egress" traffic filtering directions are expressed from the point of view of the logical switch Port. In other words:

- Logical Port Egress Filtering (connections inbound to the VM)

  The primary goal of Security Profile egress filtering is to protect the VM from untrusted hosts that may attempt to initiate connections to the VM. As soon as the VM (or better the VIF interface where the VM connects) is associated with a Security Profile, a default "deny all" traffic filtering is applied. This implies that the VM is completely isolated and protected from the rest of the network until specific rules are added to the Security Profile to open "holes" allowing traffic toward the VM.

- Logical Port Ingress Filtering (connections outbound from the VM)

  The primary goal of Security Profile ingress filtering is to limit the set of outbound connections a VM can initiate. This is useful when a VM should be prevented from sending a certain type of traffic. Similarly to egress filtering, ingress filters follow a default "deny all" model, meaning that if any Security Profile is applied on a logical switch port, all ingress connections will be denied unless they match a rule explicitly permitting the connection.

The need to "whitelist" traffic flows is an important characteristic of Security Profiles and differentiating them from more traditional Access Control Lists, where a series of "permit" and "deny" statements can be mixed together in defining the traffic filtering behavior.

Also, both egress and ingress rules are "reflexive" by nature. This means that if a rule applied in any of the two directions allows a VM to establish an ingoing (or outgoing) connection, the reply traffic for the same connection bypasses any filtering applied in the opposite direction and is automatically allowed. The NSX Controller implements this by dynamically setting up a flow entry in the transport node reversing the TCP or UDP 5-tuple (IP protocol, source IP address, destination IP address, source TCP/UDP port, destination TCP/UDP port). The same behavior can also be applied to ICMP traffic.

Figure 48 shows a simple example of the "reflexive" nature of Security Profile rules.

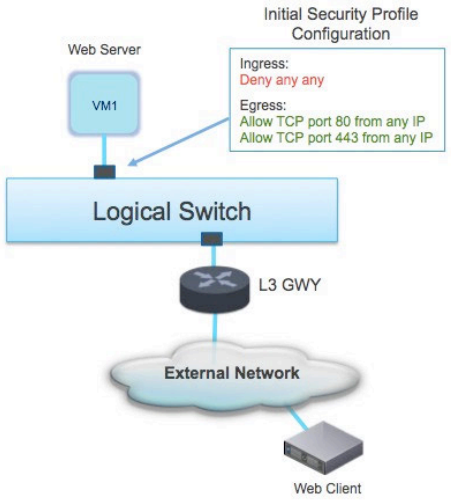


Figure 48: Reflexive Security Profile Rules

A default Security Profile configuration is applied to the VIF where the Web Server VM is connected. The filter rules aim to allow inbound connectivity to the VM (HTTP and HTTPS traffic only), while preventing outbound traffic originated from the VM itself. A dynamic ingress rule is added to the Security Profile when an external Web Client starts communicating to the Web server, ensuring that return traffic can successfully be received by the client (despite the default "deny any any" ingress rule).

Important: if a Security Profile is associated with a VIF and no ingress nor egress rules are specified, the implicit "deny all" statement applied in both directions will completely prevent the VM from communicating to the network.

The OVS switch maintains dynamically established connection entries set up by the NSX Controller. Those entries are removed periodically if no traffic is sent in the direction of the original flow. This is achieved by leveraging the following statically configured timeout:

- SSH traffic (TCP on port 22): 60 seconds
- Other TCP traffic: 300 seconds
- UDP traffic: 5 seconds

The Security Profile configuration associated with the VIF where a given VM is connected would automatically follow the VM when it migrates from one physical host to another. Likewise, if there are any changes to the set of rules within a Security Profile or the set of Security Profiles associated with a port, filters on all affected logical switch ports are updated automatically. However, the connection-tracking state is not migrated with the VM from one hypervisor to another. In the example of ingress filtering configuration, this means that the state (i.e. the reverse flow entry permitting traffic to be sent to the VM) will automatically be refreshed as long as the VM sends at least one packet for that flow after arriving at the new hypervisor. Still referring to the scenario in Figure 48, when VM1 moves to a new hypervisor, the client won't be able to receive data from the Web Server (because of the "deny all" ingress statement) until it tries again to contact it and the reverse ingress flow is installed in the new hypervisor.

It is important to highlight two important prerequisites that need to be satisfied before adding a security profile to a logical switch port:

1. Security Profiles are only supported on logical switch ports where VMs connect (VIF ports).
2. Security Profiles are only supported on logical switch ports with Port Security enabled. This is required to make the NSX Controller aware of all valid IP addresses that can be used by a VM belonging to a specific Security Profile ("Webservers" for example), in order to implement filter rules referring remote hosts part of that Security Profile (e.g., a rule like "allow tcp 3306 from Webservers").

Note: Before you use security profiles, please consult the NSX Release Notes to check the maximum number of security profiles supported, as well as limits on the number of rules in a security profile and the number of rules that can be applied to a logical port. Security profiles vary greatly in complexity, and the memory they consume on the NSX Controller varies in proportion with their complexity.

Access Control Lists for NSX Gateways

While the Security Profiles can only be applied to VIFs in order to filter intra logical switch traffic (intra IP subnet communication between virtual machines), more traditional ACLs can be applied to the L2 and L3 NSX Gateway ports. This is done to be able to prevent inter-subnet communication between virtual and/or physical workloads (ACLs on L3 Gateway) or intra-subnet communication between virtual machines and physical workloads (ACLs on L2 Gateway).

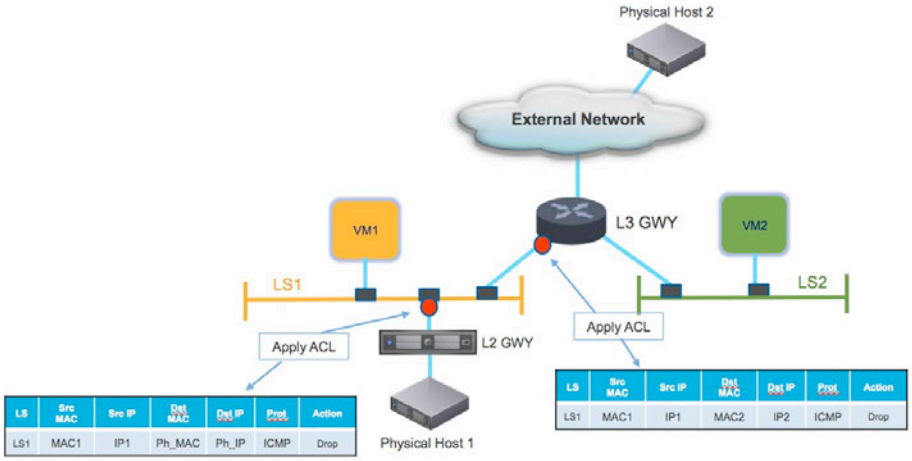


Figure 49: Applying ACLs on NSX Gateways

As shown in Figure 49, the configuration of an ACL on the interface of an NSX Gateway instructs the NSX Controller to push specific flow information to the NSX Gateway to be installed in the OVS Kernel to filter the traffic.

In the example above, the goal is to filter ICMP communication between VM1 and VM2; hence a specific ACL is applied to the logical router port interconnecting the two logical switches. The corresponding "drop" flow entry highlighted in the figure is then applied into the Kernel of the OVS of the transport node to enforce the policy. However, which transport node will have the entry really depends on the specific logical routing deployment:

- In the distributed routing model, the flow will be installed in the kernel of the OVS of the hypervisor where the source VM (VM1) is connected. This means that traffic will be dropped without even leaving the hypervisor. This is obviously required because VM2 could potentially be connected to the same hypervisor.
- In a centralized routing model, the flow entry would instead be applied on the L3 Gateway appliance hosting the active logical router instance. This implies that traffic will be sent from the source hypervisor to the central location and then dropped.

In the case where the ACL was applied to the L2 gateway port (to prevent for example communication between VM1 and the Physical Host 1), the flow will instead always been applied to the transport node hosting the active L2 gateway instance.

Other important considerations are the following:

- Ingress filtering applies to flows originated from workloads connected to a given logical switch and received by the logical router (or by the active L2 gateway instance). Egress filtering happens instead in the opposite direction, from the logical router (or the active L2 gateway instance) toward a given logical switch.
- Differently from what happens when deploying ACLs on traditional networking devices, there is not an implicit "deny all" rule at the end of the ACL. Instead, by default the last rule is a "permit all", which implies it is strongly recommended to explicitly configure a "deny all" last rule to avoid permitting more traffic than originally intended.
- Each ACL rule has three possible actions: "deny", "allow" and "allow reflexive". The first two are intuitive, the third instead permits to achieve a behavior similar to the one described for the Security Profile where allowing a flow in a specific direction (for example from a VM connected to LS1 toward a VM connected to LS2) would dynamically allow the flow on the return path (from VM2 to VM1) independently from any specific policy applied in that direction.

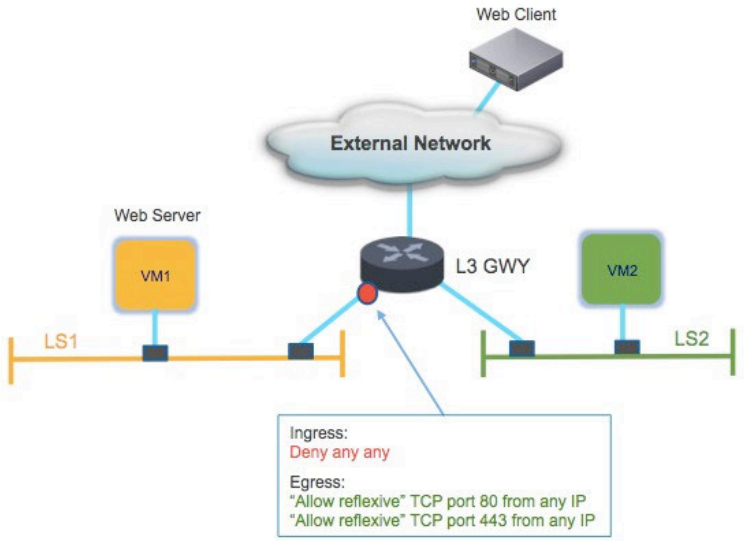


Figure 50: Use of reflexive ACLs

- From release 4.1 it is introduced the capability of configuring ACLs that allows return traffic only for TCP sessions that are already established. This means that a check is performed to ensure that the TCP ACK flag (and reset - RST) is set in the packet before letting it go through. This functionality offers greater scalability when compared to the use of reflexive ACLs even if it applies only to TCP traffic (Figure 51).

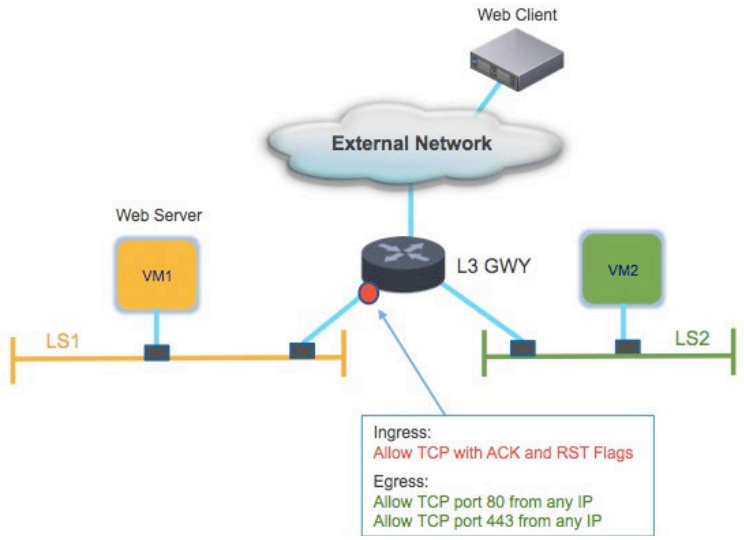


Figure 51: Use of ACLs Matching TCP Flags

QoS

The QoS functionalities supported with NSX-MH can be divided in two categories:

1. Controlling the amount of traffic generated by a virtual machine or by an aggregate of virtual machines connected to the same hypervisor node.
2. Setting the QoS marking (DSCP value) of overlay traffic sent out the physical NIC of a given transport node. Before describing in more details those QoS functionalities, it is important to clarify how their configuration is always associated with the concept of "Logical Queue". A Logical Queue is a new API abstraction implemented on a given transport node using OVS software queues. The traffic shaping and traffic-marking configuration is always applied in the context of a given logical queue. More specifically:

- A logical queue can be associated only to a logical switch port (being a VIF or a L2 Gateway attachment point), as highlighted in Figure 52.

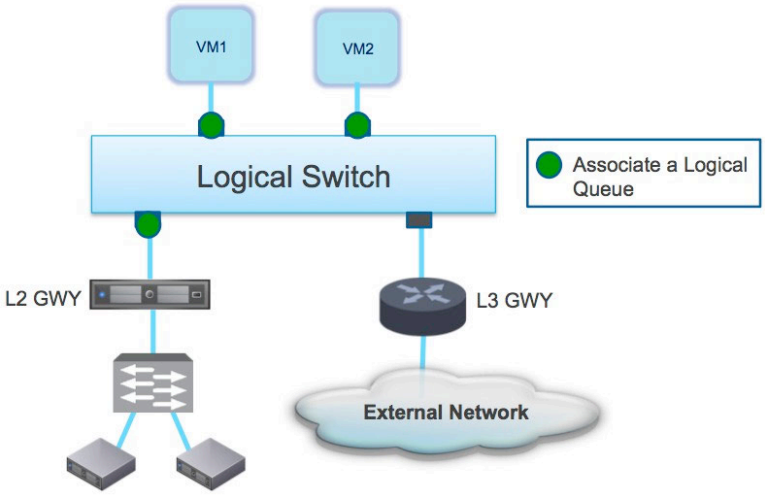


Figure 52: Allowed Mapping of Logical Queues to Logical Ports

- It is possible to define unique Logical Queues and map each of them to a logical switch port in a one-to-one fashion.
- In that case, the configuration associated with each Logical Queue would only apply to the specific traffic generated by the given VM connected to that LS port or to the aggregate of traffic generated by the physical workloads accessible via the NSX L2 gateway (when the Logical Queue is associated with a L2 Gateway attachment point).
- Alternatively, the same Logical Queue can be associated with multiple VIFs. In that case, the configured QoS parameters would apply to the aggregate of traffic generated by the virtual machines deployed in the same hypervisor node and connected to those logical ports.
- Since a Logical Queue is mapped to a VIF (or a set of VIFs), it follows the corresponding VMs when they migrate between hypervisors.
- A Logical Queue is always associated with the specific transport node Physical NIC that forwards traffic for the logical switch port the Logical Queue is mapped to. As a result, the QoS configuration specified in the Logical Queue affects only traffic leaving the hypervisor via the NIC and it does not apply to VM-to-VM traffic locally switched inside the transport node (Figure 53).

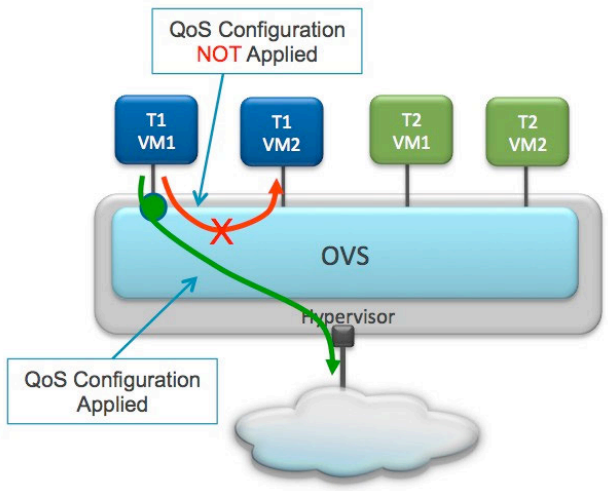


Figure 53: QoS Applied Only to Flows Leaving the Hypervisor

It is worth noticing that if the transport node connects to the physical network with multiple interfaces bonded together in a logical port-channel, the QoS configuration is individually applied to each physical member of the bond. For example, if the QoS policy dictates to allow only a certain amount of traffic originated from a VM, the total amount actually allowed would be a multiple of that depending on the number of bundled physical uplinks.

Traffic Shaping

QoS traffic shaping configuration associated with a Logical Queue allows defining two main parameters:

- Minimum Rate: this QoS policy allocates a guaranteed minimum bandwidth of traffic. As previously mentioned, if a Logical Queue configuration is applied 1:1 to a VIF, then the connected VM is guaranteed at least the configured amount of bandwidth even in times of congestion. If the Logical Queue configuration is applied to multiple VIFs, then the minimum bandwidth is guaranteed for the aggregate of traffic originated by all the connected VMs. Despite the configured minimum rate parameter, a VM will always be able to leverage more bandwidth if there is no congestion (up to the total amount of bandwidth available on the physical uplink). In order for this QoS policy to be efficient, it is key that the total minimum bandwidth configured for all the VMs belonging to a given hypervisor is less or equal of the total bandwidth available via the physical uplinks (else traffic will obviously be dropped even if should theoretically be guaranteed).
- Maximum Rate: this QoS policy caps the maximum rate at which a logical switch port (or an aggregate of logical switch ports for workloads connected to the same hypervisor node) can send traffic. This is functionality equivalent to the rate limiting usually offered by physical networking devices, and ensures that traffic total amount is kept below the configured threshold even in absence of congestion when sufficient capacity would exist on the transport node physical uplinks.

Traffic Marking

Marking of traffic is usually enforced at the edge of a network for being able to properly classify and prioritize the traffic while it traverses it. In the context of NSX-MH, it is possible to mark traffic that is generated by virtual and physical workloads and propagate that marking to the outer header of the overlay traffic (STT, VXLAN or GRE) so that this tunneled traffic can then be prioritized while traversing the physical fabric.

NSX-MH acts on DSCP marking found in the IP header (and not COS marking) because DSCP is widely supported in datacenter Hardware and supports fine-grained prioritization. Figure 54 shows the propagation of DSCP marking from the IP header of the original packet generated by the VM into the outer IP header (in the example STT is the used encapsulation).

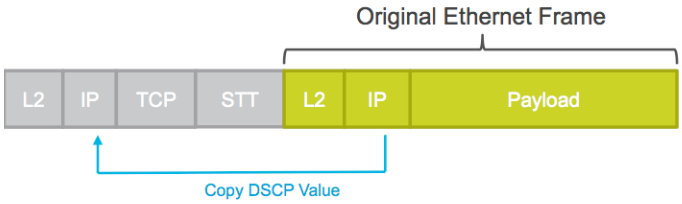


Figure 54: Propagating DCSP Marking into the Outer IP Header

It is worth noticing that the original inner DSCP value is always preserved (after unencapsulation), even in cases where the outer one may be modified while traversing the physical network infrastructure.

When it comes to the marking behavior specified in a Logical Queue configuration, the following scenarios are possible:

- Default Behavior: if no logical queue is associated with the LS port, NSX preserves the DSCP markings set by the VM connected to that port and propagate it to the outer IP header.
- Trusted Mode: with this Logical Queue configuration, the behavior is identical to the default one described above (DSCP marking set by the VM is preserved).
- Untrusted Mode: in this case, the original marking set by the VM is ignored and replaced with a different value (in the range 0-64). This value can be specified as part of the Logical Queue configuration; if it is not explicitly configured, the DSCP field will be set to a value of 0.

Based on the behavior described above, in deployments where the desire is to keep tight control on the marking of traffic generated by each VM, it is strongly recommended to always apply a Logical Queue configuration on the VIFs where the VMs are connected. Once again, this is because in absence of that configuration (default behavior), the original DSCP marking is always trusted and propagated to the outer header.

Network Virtualization Design Considerations

In this section, we discuss how the logical networks implemented using overlay technologies (STT, GRE or VX-LAN) can be deployed over common data center network topologies. We first address requirements for the physical network and then look at the network designs that are optimal for network virtualization. Finally, the logical networks and related services and scale considerations are explained.

Evolution of Data Center Network Designs

Before discussing what attributes a physical network should have in order to provide a robust IP transport for deploying network virtualization, it may be worth it to quickly describe the current trends and design evolutions of DC networks. Until few years ago, most of the DC networks were exclusively built in a modular fashion, as shown in Figure 55.

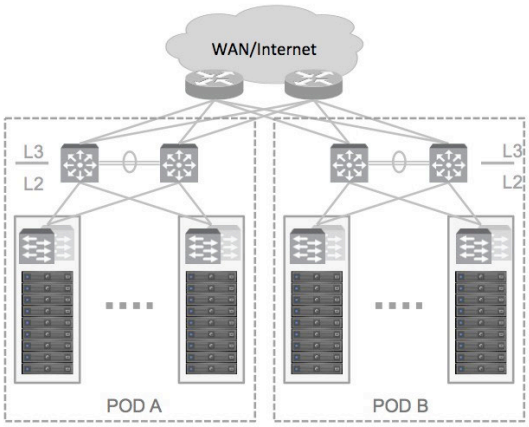


Figure 55: Classical Access/Aggregation/Core DC Network

The modular design shown above represents a scalable approach, where multiple PODs can be interconnected via a L3 DC Core. Each POD represents an aggregation block that limits the extension of VLANs inside the DC fabric. The aggregation layer devices of each POD are in fact the demarcation line between L2 and L3 network domains and this is done to limit the extension of L2 broadcast and failure domains. The immediate consequence of this approach is that all the applications requiring L2 adjacency (Application clusters, Virtual Machines live migration, etc.) had to be deployed inside a given POD. Also, this classical design is mostly effective if the majority of traffic flows are exchanged in the north-south direction and only limited communication happens inter-PODs.

The classical modular approach just described has evolved in the last few years into a leaf-spine design (Figure 56).

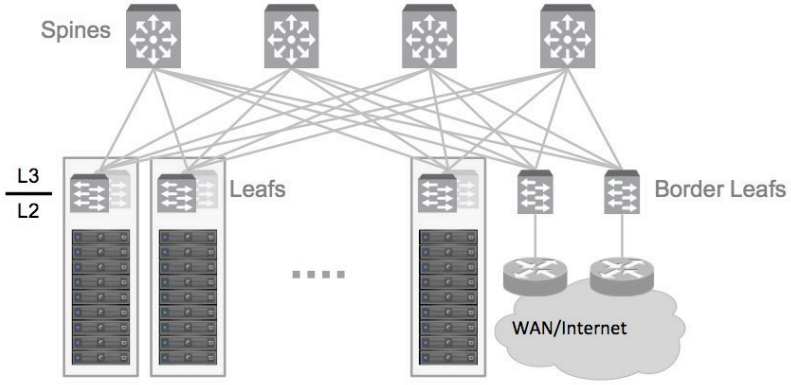


Figure 56: Leaf-Spine Fabric Design

The network shown above typically takes the name “folded CLOS” design, and is usually deployed by large customers who request a DC fabric providing high scalability characteristics with a reduced number of functionalities. Every device in the CLOS fabric takes the name of leaf or spine, depending on the tier where it is deployed:

Leaf Layer: the Leaf layer is what determines oversubscription ratios, and thus the required size of the Spine. Devices in this layer usually employ simpler designs that have fewer “moving parts” to effectively forward packets while learning the network graph accurately.

Spine Layer: the Spine layer is responsible for interconnecting all Leafs. Individual nodes within a Spine are usually not connected to one another nor form any routing protocol adjacencies among themselves. Rather, Spine devices typically are characterized by a simple and minimal configuration, as they are responsible for performing fast traffic switching and essentially function as the fabric “backplane”.

Border Leaf Layer: in some scenarios the Spine devices may be used to directly connect to the outside world, but more often they forward externally destined traffic to a pair of specialized leaf nodes acting as Border Leafs. Border Leafs may inject into the fabric default routes to attract traffic intended for external destinations.

This design evolution is driven by two main requirements: the ever increasing amount of east-west communication and the desire to be able to deploy applications across the DC fabric without being restricted by the traditional limited extension of L2 domain inside each POD.

The first requirement is addressed by increasing the number of spine devices each leaf connects to: as shown in figure above, each leaf is offered multiple equal cost paths in order to augment the bandwidth available for east-west communication and make predictable and deterministic the latency experienced for those flows.

The second requirement can be tackled by creating a larger L2 Domain that extends across the entire DC Fabric. Different switch vendors have proposed alternatives solutions to build those large L2 Fabric networks by eliminating the use of Spanning-Tree as control plane protocol, like for example Cisco’s proprietary FabricPath technology or the similar Transparent Interconnection of Lots of Links (TRILL) that is currently an IETF Draft.

However, more and more customers are steering toward the deployment of leaf-spine Routed Fabric networks, where the boundary between L2 and L3 is brought down to the leaf (or Top-of-Rack) device layer (see previous Figure 56).

The routed fabric design offers the same characteristics of low oversubscription, high east-west bandwidth availability, and predictability and configuration uniformity. In addition, enhances device interoperability (since switches from multiple vendors can be interconnected together) and overall resiliency. The main challenge of a routed design is the incapability of supporting applications that require L2 adjacency between different nodes, given that the extension of a L2 domain is limited behind each leaf switch. This is where Network Virtualization comes to the rescue: the decoupling between logical and physical networks allows providing any type of connectivity in the logical space, independently from how the underlay network is configured.

Since the expectation is the deployment of routed fabric networks will become the preferred option for customers, the focus on the rest of this document is on Data Center routed access designs where the leaf/access nodes perform full layer 3 functionalities.

However, it is also expected that customers will have to slowly migrate from the original multi-pod deployment toward pure leaf/spine architecture. NSX can ease this migration process thanks to the decoupling of virtual networks and physical network that allows it to seamlessly create a common pool of IT resources and consistent networking service model across multiple disparate types of networks (Figure 57).

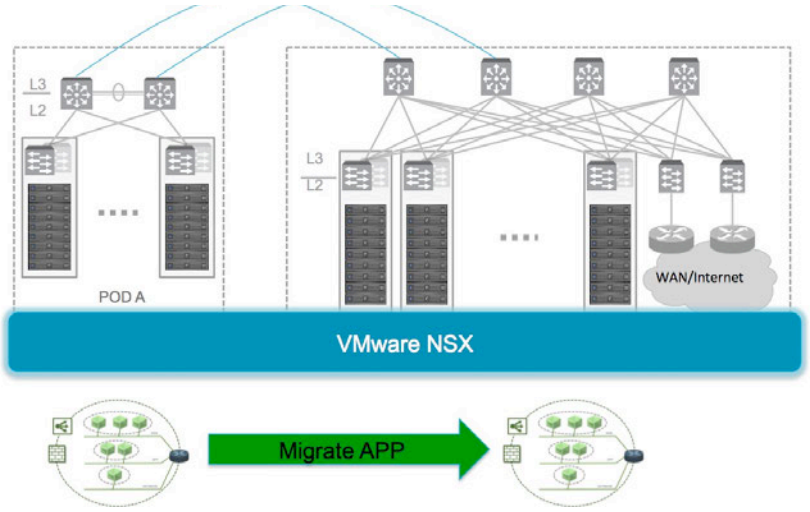


Figure 57: Application Migration across DC PODs

Physical Network Requirements

One of the key goals of network virtualization is to provide virtual-to-physical network abstraction and decoupling. In order to have a successful deployment, it is however important that the physical fabric provides a robust IP transport with the following attributes:

- Simplicity
- Scalability
- High-bandwidth
- Fault-tolerance
- QoS-providing

The following sections offer some detail for each of these parameters.

Note: in the following discussion, the terms access layer switch, top-of-rack (ToR) switch and leaf switch are used interchangeably. A leaf switch (or better, a redundant pair of leaf switches) is typically located inside a rack and provides network access to the servers inside that rack. The terms aggregation and spine layer—which effectively provide connectivity between racks—refer to the location in the network that aggregates all the access switches.

Simplicity

Configuration of the switches that compose the overall fabric inside a data center must be simple. General or global configuration such as AAA, SNMP, SYSLOG, NTP, and so on, should be replicated almost line-by-line, independent of the position of the switches. The following are the primary examples of data center fabric design connectivity:

Leaf Switches

Ports facing the servers inside a rack should have a minimal configuration. Assuming the server has multiple interfaces of the same speed, link aggregation (also called link bonding) can be used. Figure 58 shows a transport node dual connected to a pair of ToR switches.

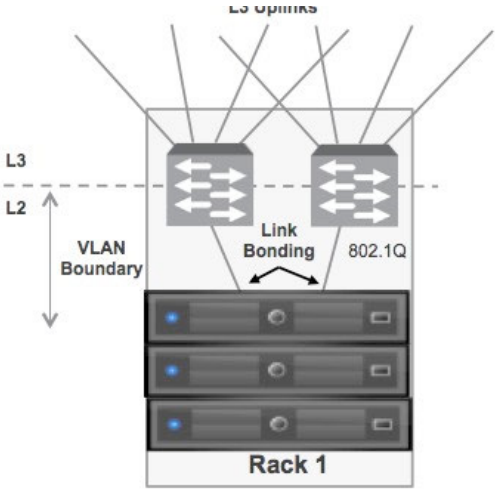


Figure 58: Hypervisor Dual-Attached to a Pair of Leaf Devices

Link bonding is valuable for two main reasons: improving availability of the connections between the transport node and the physical network; additionally, increasing bandwidth availability and traffic forwarding capacity (depending on the specific bonding options deployed).

OVS vSwitch currently supports three link-bonding options, each one providing different levels of traffic load balancing. The choice of a specific link bonding solution may have an impact on the ToR side, as clarified in the points below.

1. Active/Standby: this mode provides only fault tolerance, given that all the traffic flows originated from the transport node are sent and received on the single NIC interface in active state. If the active interface fails, the standby link is activated and traffic start flowing via the alternate path. The physical ToR devices do not need of being aware of this configuration on the transport node side.
2. Active/Active Balance SLB: outgoing flows are balanced across all active uplinks based on a hash of the source MAC address. Also in this case there is no need for the physical switch to be aware of the link bonding.
3. Active/Active Balance TCP: this is the recommended link bonding option, where outgoing flows are balanced across all active uplinks based on a hash of the L2/L3/L4 header data. This is the option that is usually referred to as “port-channeling”, where multiple physical links are bundled together into a single logical Port-Channel. This bundling could be dynamic (leveraging the 802.3ad LACP protocol) or static. The end result is a more fine-grained load balancing (flow based), but it requires that the physical switches are available of supporting a distributed port- channeling mechanism where a port-channel can be created between a server and two separate ToR devices. Hardware vendors offer different solutions to meet this requirement, as for example vPC (virtual Port-Channel, offered by Cisco) or M-LAG (MultiChassis Link Aggregation, offered by Arista).

Additional considerations are required for the load balancing of overlay traffic. Given the fact that all the overlay traffic generated from a transport node is always sent and received from the single IP address identifying the overlay interface (usually referred to as transport connector), the use of the first two link bonding options described above will always result in using a single physical uplink for sending and receiving traffic. This is not true when leveraging a Port-Channel as the load bonding solution, since in this case traffic load sharing on the different uplinks also depends on the specific overlay encapsulation in use.

More specifically:

• STT: this overlay option leverages a L2inTCP encapsulation, where the original Ethernet frame sourced by a virtual machine is encapsulated with external L2, L3 and L4 (TCP) headers (Figure 59).

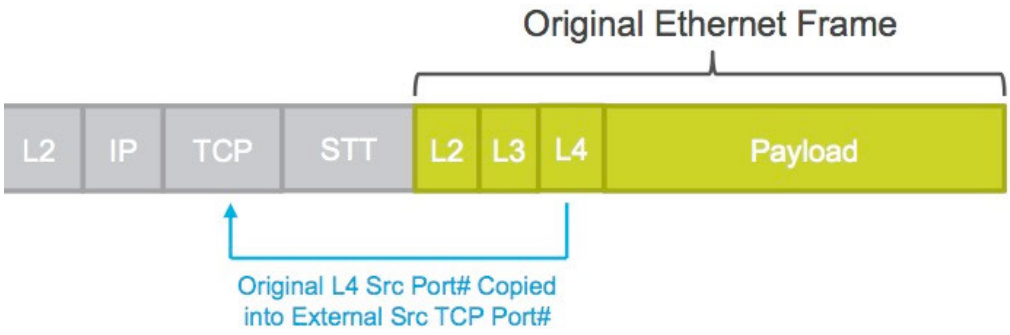


Figure 59: STT Encapsulation

As a consequence, traffic will be distributed across all the physical uplinks bundled in the Port-Channel, given that the external L3 Destination address will vary depending on the remote transport node the traffic is destined to, and on top of that the Source TCP port used will be the same of the Src L4 port used in the original Ethernet frame (if present). Notice how the physical link used in the northbound direction (from the server to the network) may be different from the one used in the southbound direction (from the network toward the server), depending on the load-balancing scheme configured on the physical switches.

- VXLAN: this last one is a L2inUDP encapsulation option, where the Source Port value in the external UDP header is calculated by performing an hashing of the L2/L3/L4 headers present in the original Ethernet frame (Figure 60).

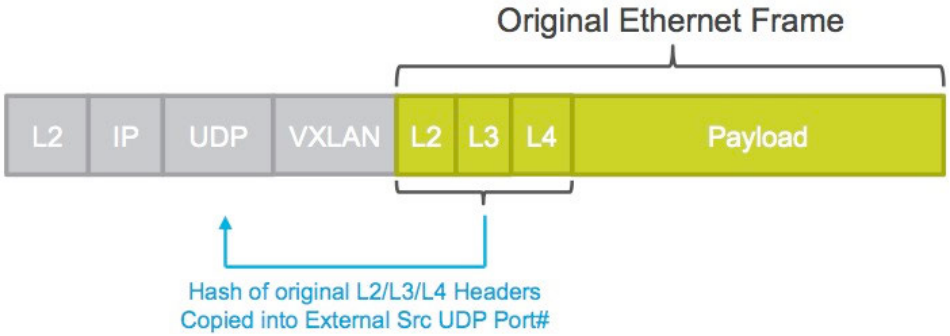


Figure 60: VXLAN Encapsulation

This implies that traffic will be load-balanced across the different active uplinks on a per-flow basis.

- GRE: this is a simpler L2inIP encapsulation (Figure 61), where the original Ethernet frame is encapsulated with L2 and L3 external headers (no L4 header is present).

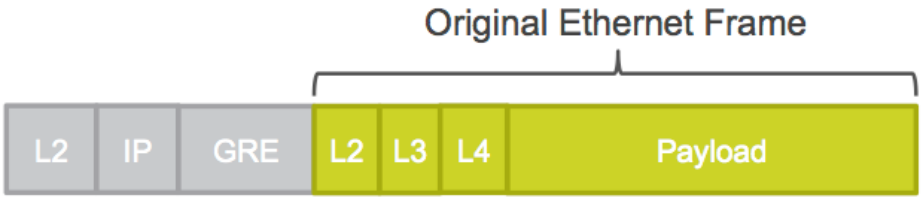


Figure 61: GRE Encapsulation

This means that even when using a Port-Channel, the only variable in the external headers is represented by the Destination IP address, for overlay traffic flows destined to different remote transport nodes. All the traffic flows exchanged with a given remote transport nodes are instead always leveraging the same physical uplink.

Note: when leveraging encapsulation technologies it is important to increase the MTU supported on the transport connector interface and an all the interface of the devices deployed in the physical network. STT adds 54 Bytes to each original frame, whereas GRE adds 42 Bytes and VXLAN 50 Bytes. Hence, the typical recommendation is to increase the MTU size at least to a 1600 Bytes value.

Typically, 802.1Q trunks are configured on the connections between a transport node and the leaf switches. In an NSX deployment, those trunks are used for carrying a small number of VLANs, independently from the number of logical networks that can be created in logical space. Once again, this is the beauty of decoupling logical and physical network connectivity. In the specific case of an ESXi hypervisor, no more than 4 VLANs are required (assuming all those traffic types are carried over the same physical bond of uplink interfaces): Management, vMotion, IP Storage and Transport (for carrying the overlay traffic).

The leaf switch terminates and provides default gateway functionality, respectively, for each VLAN; that is, it has a switch virtual interface (SVI) for each VLAN. An FHRP protocol (VRRP, HSRP, etc.) is run between the redundant pair of leaf deployed in each rack to provide a resilient default gateway functionality.

Uplinks from the ToR switch, or leaf switch, to the aggregation or spine layer are routed point-to-point links. VLAN trunking on the leaf uplinks is not allowed—not even for a single VLAN. A dynamic routing protocol—OSPF, ISIS, BGP, for example—is configured between the leaf and spine switches (Figure 62).

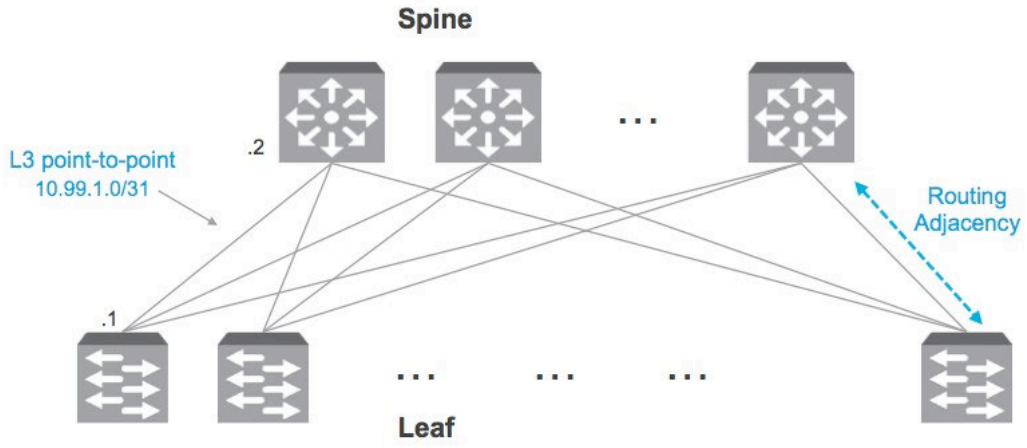


Figure 62: L3 Connectivity between Leaf and Spine Switches

Each ToR switch in the rack will advertise a small set of prefixes, typically one per VLAN or subnet it has present. In turn, it will calculate equal cost paths to the prefixes received from other ToR switches.

Spine Switches

The spine switch has only interfaces that connect to leaf switches; all interfaces are configured as routed point-to-point links, effectively acting as the "other end" of the leaf switch's point-to-point uplinks.

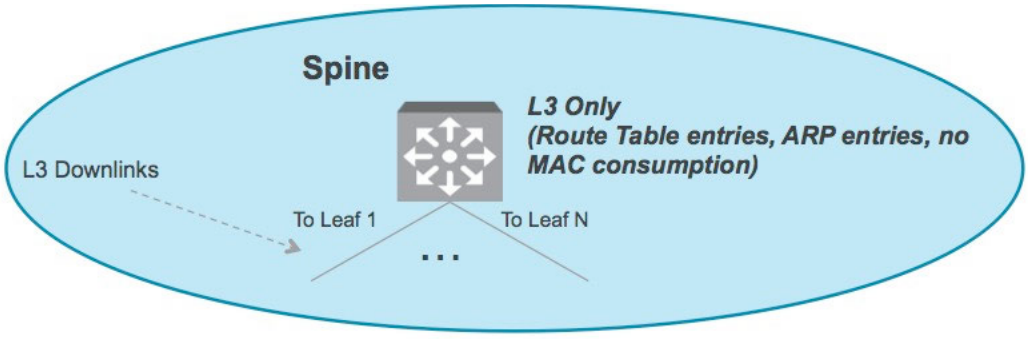


Figure 63: Spine Switch Interfaces

Links between spine switches typically are not required. If there is a link failure between a spine switch and a leaf switch, the routing protocol will ensure that no traffic for the affected rack is attracted to the spine switch that has lost connectivity to that rack.

Scalability

Factors concerning scale include the number of racks supported in a fabric, the amount of bandwidth existing between any two racks in a data center, the number of paths a leaf switch can select from when communicating with another rack, and so on.

The total number of ports available across all the spine switches usually dictates the number of racks supported in a fabric and the oversubscription that is acceptable. More details are provided in the "High Bandwidth" section.

Different racks might be hosting different types of infrastructure. As it will be discussed in more detail in the "NSX-MH Network Design Considerations" section, it is possible to identify three types of racks functionalities: compute racks, edge racks and infrastructure racks.

Keeping the design modular ensures flexibility in the network connectivity provided to the various racks. This is important because different type of racks might have different bandwidth requirements (storage filers for example drive more traffic in and out the infrastructure racks). Link speed as well as number of links can hence be varied to satisfy different bandwidth demands. This can be done for each rack without sacrificing any of the architecture of the spine switch or leaf switch.

The number of links to the spine switches dictates how many paths that traffic from this rack to another rack can be chosen from. Because the number of hops between any two racks is consistent, equal-cost multipathing (ECMP) can be utilized. Assuming traffic sourced by the servers carry a TCP or UDP header, traffic spray can occur on a per-flow basis. As previously shown, overlay encapsulated traffic is usually characterized by the presence of external L2/L3/L4 headers (with the exception of GRE that presents only external L2/L3 headers). Each leaf and spine device can perform a hashing of the L2/L3/L4 information available in those external headers, ensuring a statistically optimal distribution of overlay traffic across all the ECMP links available between the leaf and spine nodes.

High Bandwidth

In spine–leaf switch topologies, oversubscription typically occurs—if at all—at one point: the leaf switch. The calculation is simple: total amount of bandwidth available to all servers connected to a given leaf switch divided by the aggregate amount of uplink bandwidth provides the oversubscription.

For example, 20 servers with one 10 Gigabit Ethernet (10GbE) port each create up to 200Gbps of bandwidth. Assuming eight 10GbE uplinks to the spine—a total of 80 Gbps— this results in a 2.5:1 oversubscription factor.

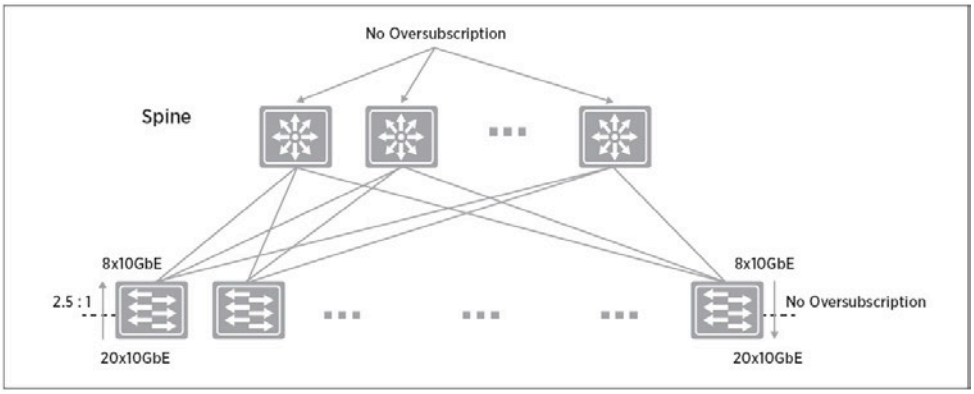


Figure 64: Oversubscription Example for Leaf-Spine Topologies

As discussed in the previous section, depending on the rack's function, more or less bandwidth can be made available to a rack by virtue of provisioning more or fewer uplinks. In other words, the level of oversubscription can vary on a per- rack basis.

From an architecture standpoint, one rule must be obeyed: the number of uplinks from a leaf switch to each spine switch must be the same. This means that having two uplinks to spine switch A and only one uplink to spine switches B, C and D would be a poor design because "more" traffic would be sent to the leaf switch via spine switch A, potentially creating a hot spot.

Fault Tolerance

The larger the environment, the more switches that make up the overall fabric and the greater the possibility for one component of the data center switching fabric to fail. The reason for building a resilient fabric is that it can sustain individual link or box failures without a widespread impact.

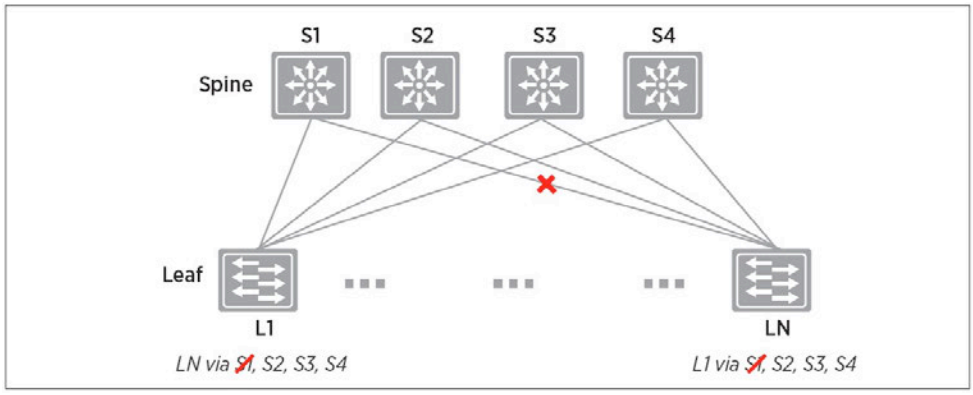


Figure 65: Link Failure Scenario in Leaf-Spine Topologies

For example, if one link or one of the spine switches were to fail, traffic between racks would continue to be routed in an L3 fabric across the remaining spine switches. For L3 fabrics, the routing protocol ensures that only remaining paths would be chosen. And because more than two spine switches can be installed, the impact of a spine switch failure is reduced.

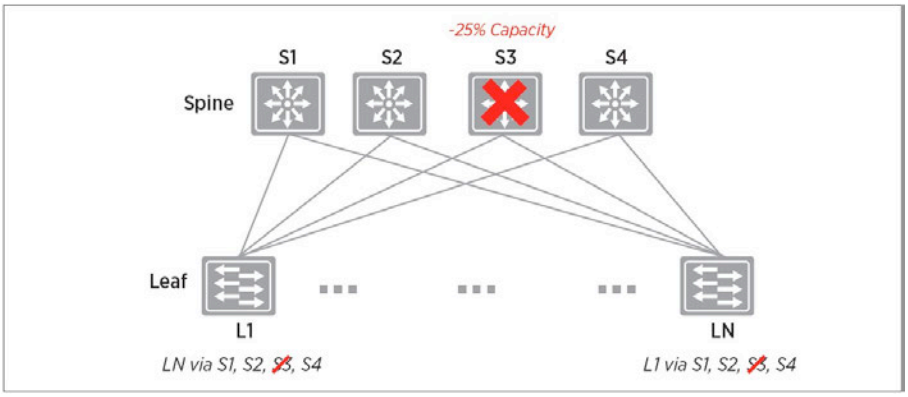


Figure 65: Spine Switch Failure Scenario and Impact on Bandwidth

Multipathing-capable fabrics handle box or link failures, reducing the need for manual network maintenance and operations. If a software upgrade must be performed on a fabric switch, the node can be taken out of service gracefully by changing routing protocol metrics; quickly, the traffic through that switch will be drained, freeing it up for maintenance. Depending on the width of the spine—that is, how many switches are in the aggregation or spine layer—the additional load the remaining switches must carry is not as significant as if there were only two switches in the aggregation layer.

Differentiated Services –Quality of Service

Virtualized environments must carry various types of traffic—including tenant, storage and management—across the switching infrastructure. Each traffic type has different characteristics and applies different demands on the physical switching infrastructure. Although management traffic typically is low in volume, it can be critical for controlling physical and virtual network state. IP storage traffic typically is high in volume and generally stays within a data center. The cloud operator might be offering various levels of service for tenants. Different tenants' traffic carries different quality of service (QoS) values across the fabric.

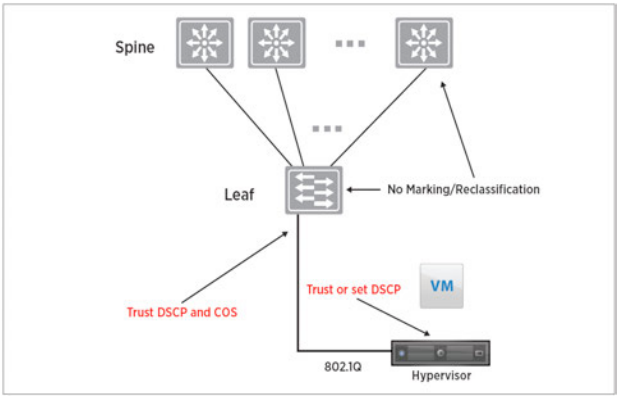


Figure 67: Quality of Service (QoS) Tagging

For virtualized environments, the hypervisor presents the trusted boundary, meaning it sets the respective QoS values for the different traffic types. In this case, the physical switching infrastructure is expected to "trust" these values. No reclassification is necessary at the server-facing port of a leaf switch. If there were a congestion point in the physical switching infrastructure, the QoS values would be looked at to determine how traffic should be sequenced—and potentially dropped—or prioritized.

There are two types of QoS configuration supported in the physical switching infrastructure; one is handled at L2 and the other at L3 or IP layer. The L2 QoS is sometimes referred to as "Class of Service" (CoS) and the L3 QoS as "DSCP marking".

As discussed in the "QoS" section, NSX-MH allows trusting the DSCP marking originally applied by a virtual machine or to explicitly modify and set the DSCP value at the logical switch level. In both cases, the DSCP value is then propagated to the outer IP header (independently from the encapsulation of choice – STT, VXLAN or GRE). This enables the external physical network to prioritize the traffic based on the DSCP setting on the external header.

NSX-MH Network Design Considerations

This section discusses how network virtualization offered by VMware NSX-MH can be placed on top of the scalable L3 network fabric presented in the previous sections. Network virtualization consists of three major aspects: decouple, reproduce and automate. All three aspects are vital in achieving the desired efficiencies. This section focuses on decoupling, which is key to simplifying and scaling the physical infrastructure. Being able to provide L2 connectivity at the logical network level, independently from the characteristics of the underline network infrastructure is the fundamental property that enables the decoupling effect. In the already introduced example of a L3 DC fabric where the L2/L3 boundary is positioned at the leaf layer, VLANs cannot span beyond a single rack inside the switching infrastructure but this does not prevent in providing L2 adjacency between workloads connected to the logical networks.

When building a new environment, it is essential to choose an architecture that allows for future growth. The approach discussed here works for deployments that begin small but grow to large-scale ones while keeping the same overall architecture.

The guiding principle for such deployments is that the network virtualization solution does not imply any spanning of VLANs beyond a single rack. Although this appears to be a simple requirement, it has widespread impact on how a physical switching infrastructure can be built and on how it scales.

We differentiate between the following three types of racks within the infrastructure (Figure 68):

- Compute
- Edge
- Infrastructure

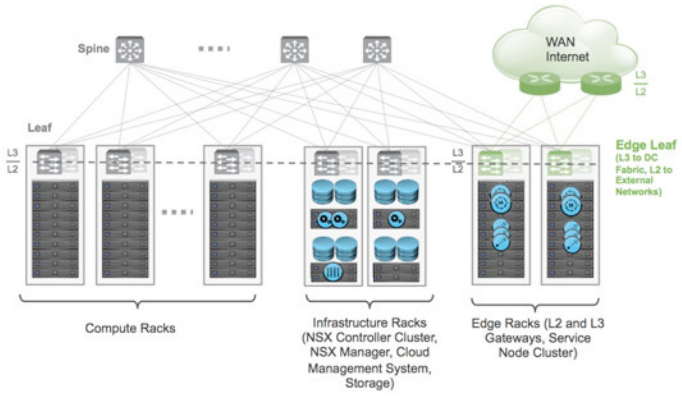


Figure 68: Data Center Design - L3 in Access Layer

Note: the separation of functions for each type of racks can be logical and not necessarily physical, For example, it may be common in smaller deployments to consolidate edge and infrastructure functions together in the same physical racks.

Before discussing more in detail the functionalities performed by the different types of racks, it is worth bringing back the application multi-tier example discussed in the previous chapter of this paper to highlight how it maps to the physical layout of the Data Center.

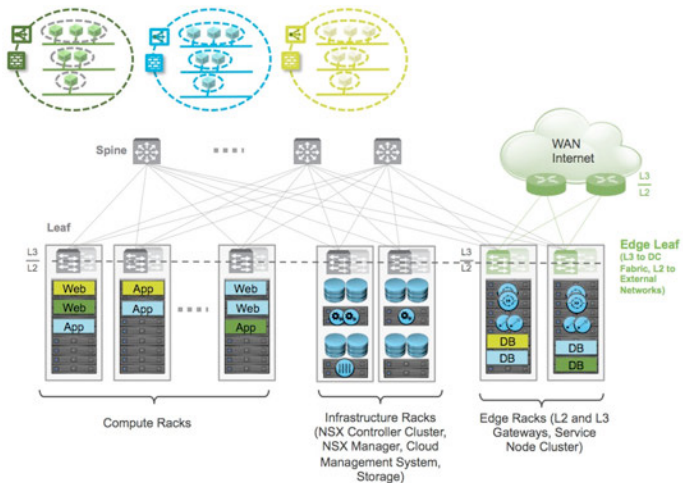


Figure 69: Deployment of Multi-Tier Applications

As shown above for a multi-tenant design, the virtual machines belonging to the WEB and APP tiers are usually deployed on the hypervisor nodes part of the Compute Racks, whereas the physical machines functioning as DB nodes are part of the Edge Racks. As it will be discussed in a following section, this is done to limit the extension of the VLANs the DB nodes belong to only to the Edge Racks and not requiring extension to the other racks (which would be challenging when deploying a routed fabric design). The implication is that traffic between the APP and DB tiers will have to be tunneled between the compute and the edge racks, and as previously mentioned it is the need of supporting this increase in east-west communication that drives the evolution of Data Center networks toward “flatter” leaf-spine designs.

Compute Racks

Compute racks make up the part of the infrastructure where tenant virtual machines are hosted. In an NSX-MH solution, those are the racks where the supported hypervisors (ESXi, KVM, XEN) running OVS vSwitch would be deployed. They should have the following design characteristics:

- Interoperate with an existing network
- For new deployments or redesigns:
    - Should not require VLANs for virtual machines
    - Should not require VLANs to extend beyond a compute rack
- Provide a repeatable-rack design

A hypervisor typically sources different types of traffic. In the following, we look at overlay, management, vSphere vMotion and storage traffic. The overlay traffic is a new traffic type that carries all the virtual machine communication and encapsulates it in TCP (STT), UDP (VXLAN) or simply IP (GRE) frames. The following section will discuss how the hypervisors connect to the external network and how these different traffic types are commonly configured.

Connecting Hypervisors

In the previous "Physical Network Requirements" section they already have been introduced different options to connect a transport node to the networks (link bounding options), so in the rest of this section we'll discuss how those server network connections are used to carry different types of traffic for the different hypervisor currently supported (ESXi, KVM and Xen).

ESXi

Generally speaking, there are three types of virtual switch available for ESXi 5.5: NSX vSwitch (the OVS flavor utilized with NSX-MH deployments), vSphere Standard Switches (VSS) and vSphere Distributed Switches (VDS). When deploying ESXi in NSX-MH, each ESXi host may run NSX vSwitch and VSS uplinks simultaneously, or just make use of NSX vSwitch uplinks. What is not supported instead is running NSX vSwitch on an ESXi host where vDS is configured.

Figure 70 shows the scenario where an ESXi hosts is equipped with 4 physical NICs interfaces, two of which connected to the NSX vSwitch while the other two connected to the native VSS switch.

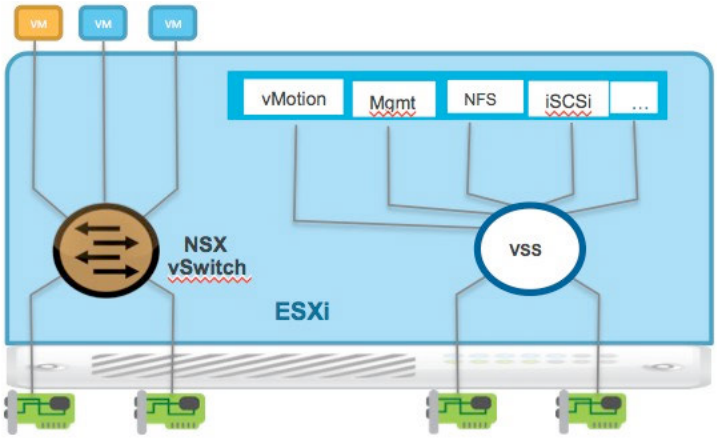


Figure 70: ESXi Uplinks Connected to NSX vSwitch and VSS

As highlighted in the diagram above, in the recommended deployment model the uplinks connected to the VSS are utilized for the various types of traffic originated from the hypervisor itself, as Management, vMotion or storage related (NFS, iSCSI). Overlay traffic originated from the virtual machines and destined to remote transport nodes is instead sent and received on the uplinks connected to the NSX vSwitch. Deploying dedicated uplinks on the VSS allows supporting all the vSphere specific features, as FT, HA, iSCSI, NFS, vMotion, SRM, VSAN. Also, a dual TCP/IP stack is available with this deployment model:

- The default IP stack is associated with the VSS, which implies a default route can be utilized for the management traffic, while static routes can be deployed for the other types of traffic.
- A separate IP stack is associated with the NSX vSwitch, which allows defining a default route to be exclusively used for the overlay traffic originated from the virtual machines and destine to remote transport nodes. Deploying network virtualization on top of a routed DC network usually implies that transport connectors IP addresses for hypervisors deployed in separate racks are usually part of different IP subnets.

The ESXi alternative deployment model is show in Figure 71

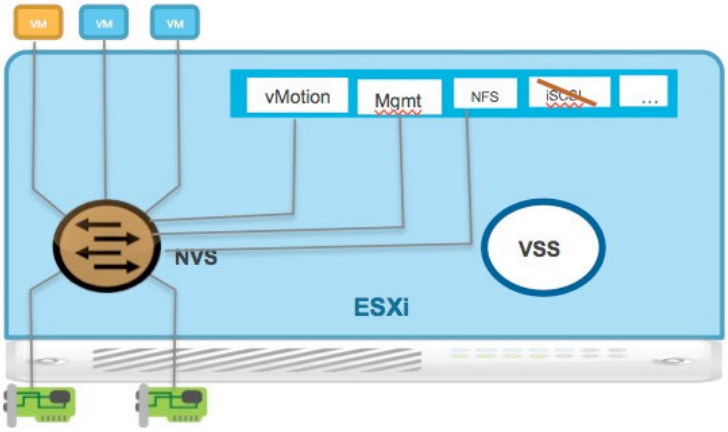


Figure 71: ESXi Uplinks Connected Only to the NSX vSwitch

In this case, the uplinks are only connected to the NSX vSwitch and used to carry all the different types of traffic. The two main differences compared to the previous scenario are:

- Only a subset of the vSphere features are currently supported in this case, specifically vMotion, NFS and Management.
- A single TCP/IP stack is available, which implies that if a default-route is used for the management traffic, it is then required to specify static routing information for the other types of traffic, including overlay.

When connecting uplinks to the NSX vSwitch, there are a couple of possible deployment alternative models:

1. Use dedicated interfaces for the different types of traffic: this implies having a dedicated interface for management traffic (usually a built-in not redundant GE interface), a pair of redundant links for vMotion and storage traffic and finally a dedicated redundant pair of links for the VM originated traffic (encapsulated with the overlay technology of choice). This approach allows reserving dedicated bandwidth to the overlay traffic, ensuring the other types of communication do not disrupt it, and still remains the recommended design option.
2. With the speeded adoption of 10GE interfaces (and upcoming 40GE interfaces) as server NICs, it becomes more applicable the alternative option of consolidating the different types of traffic on the same pair of physical uplinks. In this case, the use of VLANs is mandatory to maintain traffic logical isolation in addition of the deployment of QoS and traffic shaping technique to ensure a fair and deterministic access to the available bandwidth to those different types of communications.

KVM and Xen

The case of KVM and Xen hypervisors is different than ESXi, since a only the OVS vSwitch is available, so all the physical uplinks can only be associated with that one. Still, the same recommendation holds true: when possible, dedicate specific uplinks for management, storage and live migration traffic and leave a pair of redundant uplinks to be used for the overlay traffic generated by the virtual machines.

Edge Racks

Tighter interaction with the physical infrastructure occurs while bridging between the overlay world and the physical infrastructure. The following are the main functions provided by the servers and appliances deployed as part of the edge racks:

- Provide on-ramp and off-ramp connectivity to the external layer 3 physical network (WAN, Internet, Intranet): this is the functionality provided by NSX L3 Gateways.
- Allow bare-metal servers connected to traditional VLANs in the physical world to communicate in the logical space with virtual workloads: NSX L2 Gateways or 3rd party Hardware ToR devices are deployed for this purpose.
- Provide specific services in the logical space, as for example the handling of multi-destination traffic (Broadcast, Unknown Unicast and Multicast). As previously described, this is the task performed by Service Node appliances.

A redundant pair of ToR switches normally provides connectivity toward the Data Center network for the services deployed in the edge racks. As shown in Figure 72, those ToR devices normally play a dual role:

- On one side they function as layer 3 leaf nodes connecting with L3 point-to-point links to the fabric spine devices. In doing so, the ToR switches also represent the L2/L3 boundary and default-gateway for all the local VLANs used by the servers, as for example the management VLAN or the Transport VLAN used for overlay traffic. Because of that, the span of those VLANs is limited to each local Edge Rack.
- On the other side, they perform as a pure layer 2 device connecting to the switched physical infrastructure (green) represented in figure above. As it will be clarified below, the switched infrastructure is used to provide both L2 and L3 Gateway services and it is completely decoupled from the data center internal network.

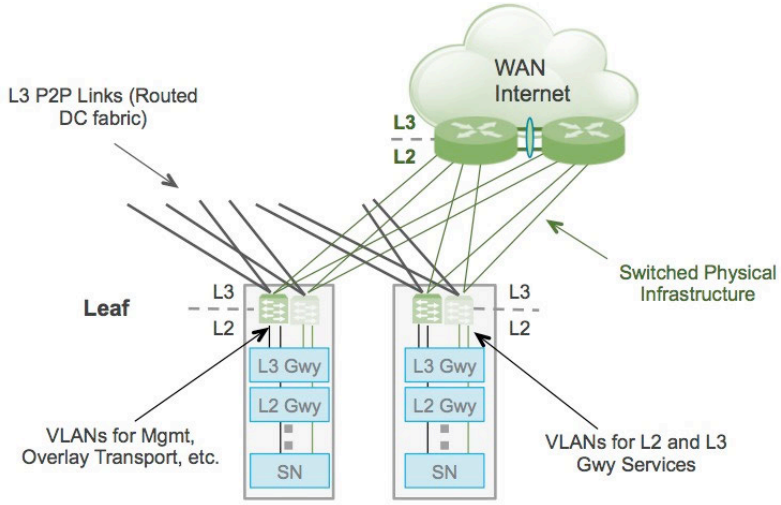


Figure 72: Connectivity of Edge Racks toward the Physical Network

For what concerns the physical connectivity of the L2/L3 Gateway and Service Node appliances to the pair of ToR switches, all the binding options already discussed in the "Physical Network Requirements" section apply here. As usual, it is recommended to leverage separate physical NIC adapters to be used for different purposes. In the case of L2/L3 Gateways, Figure 72 shows how a pair of uplinks (black) could be used to carry all the traffic destined to the DC Fabric (Mgmt, Overlay transport, Storage, etc.). Separate VLANs (localized to each Edge Rack) could be used on those connections to keep these traffic types logically isolated from each other, or alternatively separate physical NICs could also be used. A second pair of uplinks (green) could instead be used for bridging/routing traffic from the logical space toward the physical network. A separate VLAN (or set of VLANs) would be used for each deployed tenant.

NSX Layer 3 Gateway Deployment Considerations

The NSX L3 Gateway appliances host the logical routing instances assigned to each tenant functioning as default gateways for all the logical tenant networks. As previously discussed in the "Logical Routing " section, those logical routers are always utilized for north-south communication happening between virtual and physical workloads connected to logical networks and the devices deployed in the external layer 3 network domain. Depending on the use case, users can decide to employ either NAT or the static routing option to provide connectivity to the external network.

The logical routing deployment supported with NSX-MH is a single-tier model leveraging a physical router (or a redundant pair of physical routers) as Next-Hop, where a single logical interface can be utilized to connect a Logical Router to this next hop device.

In the example shown in Figure 73, an organization hosting multiple applications wants to provide connectivity among the different tiers of the applications as well as connectivity to the external network. In this topology separate logical switches provide L2 network connectivity for the VMs in each particular tier. The distributed logical routing configuration allows the VMs on different tiers to communicate with each other in an optimized fashion. At the time of writing of this document, dynamic routing between the logical router instance and the physical network is not supported with NSX-MH. Hence static routing can be used between the centralized L3 Gateway and the Next-Hop physical router, in order to allow external users to access the applications connected to the logical switches in the data center.

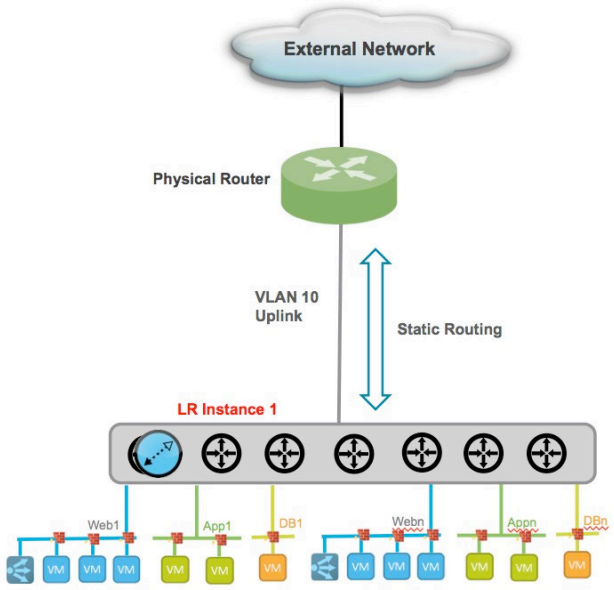


Figure 73: Physical Router as Next Hop

Once again, in this topology the east-west routing decision happens at the hypervisor level in a distributed fashion, whereas the north-south communication is centralized on the x86 appliances performing L3 Gateway functionalities.

In a service provider environment (but this is becoming more frequent also in the enterprise space) there is the need of supporting multiple tenants, with each tenant having different requirements in terms of number of isolated logical networks and other network services such as NAT, ACLs, etc. In such deployments, the different logical router instances assigned to each tenant may need to communicate to the same next-hop L3 gateway. Figure 74 shows the logical view of a typical multi-tenant deployment model.

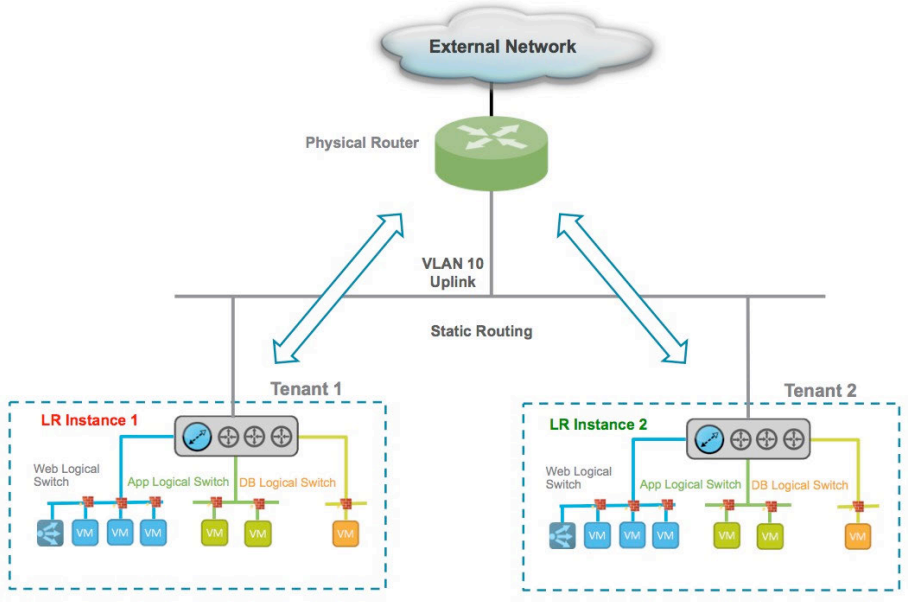


Figure 74: Multi-Tenant Deployment Model #1

In the example above, the same VLAN is used for each tenant uplinks, which implies that the tenant routers and the next-hop physical router are deployed on the same IP subnet. The advantage of this model is that the addition of new tenants requires only minimal configuration on the physical router (basically simply the addition of static routes to enable communication with the tenant subnets). On the other side, all the routing information for the different tenants are merged together on the physical router, which implies that it is not possible to support overlapping IP addresses across different tenants. To handle this case, it is required to enable NAT at the tenant logical router level. Deploying NAT brings also a second benefit: the addition of new tenants can now be handled without requiring any modification at the physical router level (i.e. no requirement to add static routing information).

An alternative deployment model is shown in Figure 75

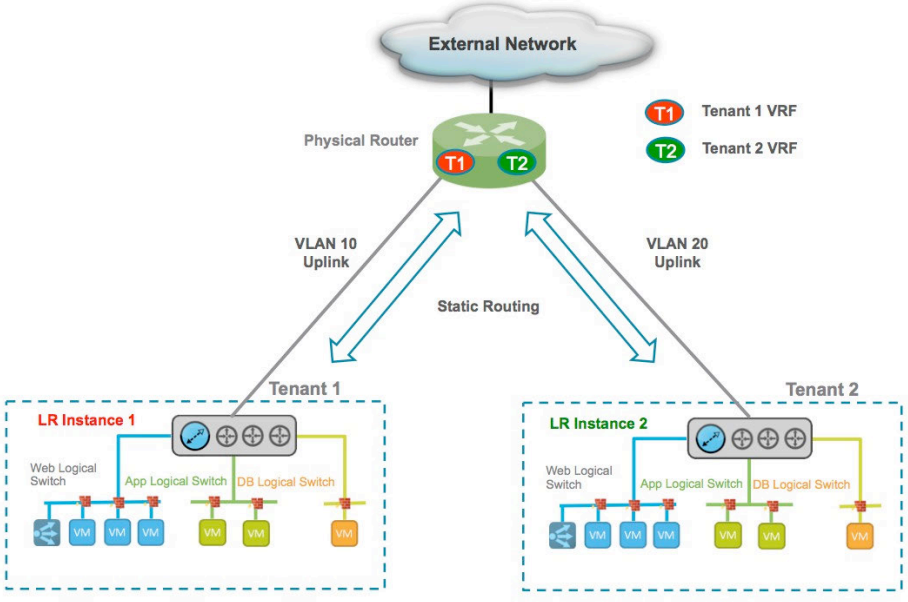


Figure 75: Multi-Tenant Deployment Model #2

The example above suggests the use of different VLAN tags to be used on the uplink of each defined tenant. The main advantage of this deployment model is the fact that logical isolation for each tenant can be maintained also in the physical infrastructure by leveraging network virtualization technologies like VRF (each VLAN is mapped to a different

VRF at the physical router layer). This also allows supporting overlapping IP addresses across tenants without requiring the use of NAT at the tenant logical router level. On the other side, a good amount of configuration needs to be applied to the physical router every time a new tenant is added.

Moving from the logical to the physical view, Figure 73 highlights the scenario leveraging static routing between logical routers deployed on L3 Gateway nodes part of the Edge Racks and a pair of redundant next-hop physical routers.

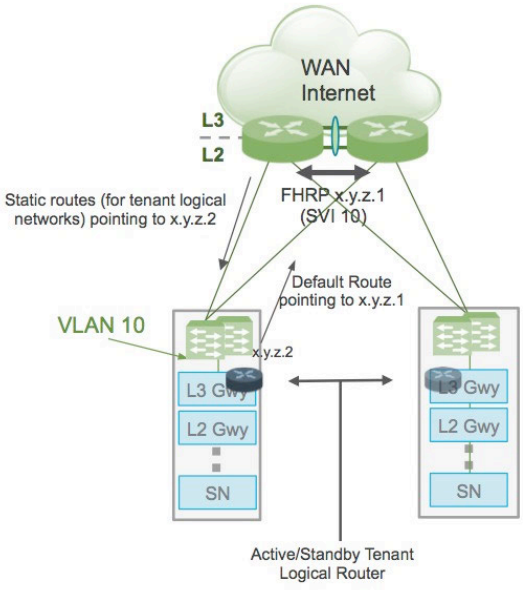


Figure 76: Active/Standby NSX L3 Gateway Topology

Since currently the logical router can leverage a single uplink interface to connect to the upstream physical router, the use of the green physical infrastructure is important for allowing the deployment of redundant physical next-hop devices. In this example, the logical router sends traffic into the physical infrastructure in VLAN 10 that is extended from the ToR switches to the pair of redundant physical routers. Those devices expose a single next-hop IP address (x.y.z.1) on

VLAN Interface 10 by implementing an FHRP protocol (HSRP, VRRP, etc.) and the logical router usually leverages a default route pointing to this active FHRP address for northbound communication with the physical network.

In order to provide redundancy to the NSX L3 Gateway function, an Active/Standby pair of logical routers is deployed for each tenant. As highlighted in figure above, it is recommended to deploy those Active/Standby instances in separate Edge racks, so that a failure scenario (for example a power outage) affecting one entire rack would cause the Standby logical router to take over and assume the outside IP address of the previously active one (x.y.z.2 in this example). As previously discussed in the “NSX L3 Gateways with Open vSwitch” section, this behavior can be easily achieved by deploying the Gateway appliances as part of two separate “Failure Zones”. The physical next-hop routers deploy static routes pointing to the IP address of the active logical router (x.y.z.2) I order to establish southbound connectivity with the logical network space.

Figure 77 shows the specific rack failure scenario, which causes a failover between the active and standby logical router instances (note that the same would happen in case of failure of the L3 Gateway appliance hosting the active logical router).

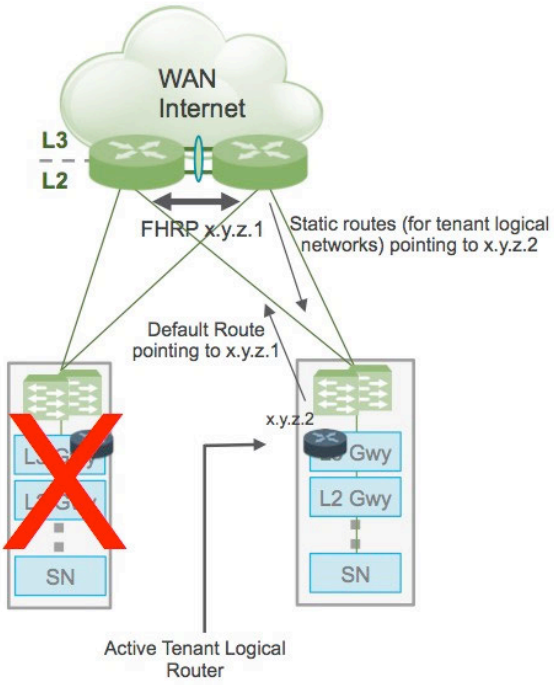


Figure 77: Failure of Active Logical Router

To notify the upstream infrastructure—that is, L2 switches that potentially interconnect the NSX L3 Gateways and the first physical router—an RARP message is sent out by the L3 Gateway appliance where the Logical Router is newly activated. For this mechanism to work, the VLAN interconnecting the logical router to the physical routers must be extended between the edge racks.

Few additional considerations:

- Those VLANs are only extended between the edge racks and not to the compute racks. That is why it is important to deploy the NSX L3 Gateways only in those racks and not to spread them across the entire fabric infrastructure.
- No VLAN extension across the Edge Racks is needed for exchanging keepalives between the active and standby logical router instances, as the STT tunnels built between the L3 Gateway appliances hosting them are used for this purpose. Before the failover, overlay traffic originated from hypervisors in the Compute Racks is directed toward the NSX L3 Gateway appliance hosting the active Logical Router. After failover, that traffic is switched toward the L3 Gateway that hosts the newly active Logical Router leveraging a separate STT tunnel.

NSX Layer 2 Gateway Deployment Considerations

The NSX L2 Gateways are used to provide access to logical networking to bare-metal servers that are deployed in VLANs defined in the physical infrastructure. An Active L2 Gateway service must then be able to terminate and unencapsulate overlay traffic (belonging to a given logical switch) and to bridge it on a specific VLAN. It is important to clarify how for each defined L2 Gateway service there must always be a 1:1 mapping between a logical switch and a VLAN ID value.

A L2 Gateway Service functions in an Active/Standby fashion, hence similarly to what previously discussed for L3 Gateways the resiliency of the service can be increased by deploying those Active/Standby instance in different Edge Racks, as shown in Figure 78.

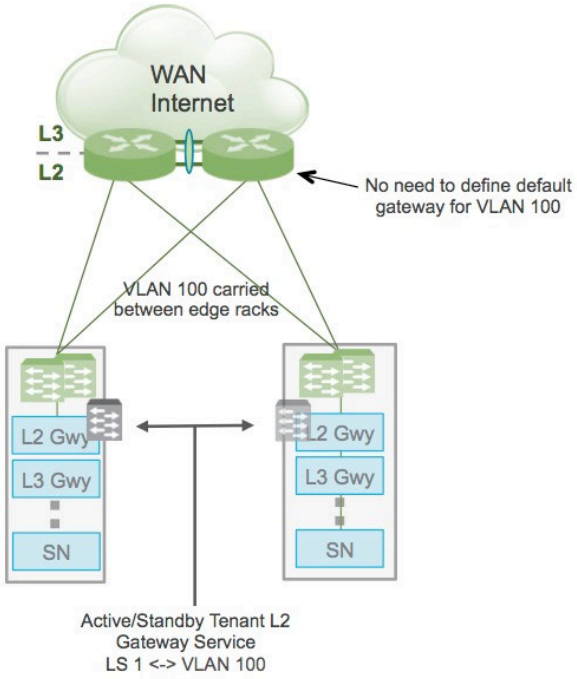


Figure 78: Active/Standby NSX L2 Gateway Topology

Note: the physical servers that get access to the logical networks via the L2 Gateway service could be deployed in the same edge racks, or alternatively a dedicated set of racks could be used, with their own dedicated pair of ToR switches.

Some important deployment considerations are below:

- It is recommended to keep the bare metal servers requiring connectivity to the logical networks into the Edge Racks. This is important to limit the extension of the VLANs they belong to only to those racks and not to the other compute racks connected to different ToR switches.
- The VLAN where traffic is bridged to (VLAN 100 in the example above) is carried on the same switching infrastructure previously introduced and utilized for the L3 north-south communication. However, there is no requirement to define SVIs or L3 interfaces for that VLAN on the physical devices. The default gateway for the bare metal servers could be deployed in two ways, shown in figure xxx.

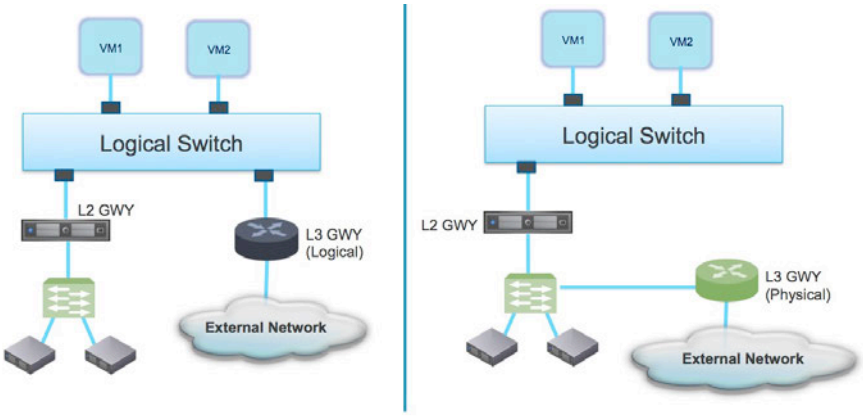


Figure 79: Default Gateway Deployment Options for Physical Servers

1. The default gateway is deployed in the logical space and the physical servers access it through the L2 Gateway service.
2. The default gateway is deployed in the physical space and the virtual machines connected to the LS access it via the L2 Gateway service.

- The keepalives messages between the active and standby L2 Gateway services are not exchanged through the STT tunnel but on VLAN 100 across the physical infrastructure. If the Active L2 gateway fails (because of a rack or appliance failure, for example), the standby will take over after few seconds and will generate a Spanning-Tree TCN notification into VLAN 100 to clear the MAC address tables of all the physical L2 devices.

NSX Service Nodes Deployment Considerations

When deploying the Service Nodes for performing replication for multi-destination traffic, they only require connectivity to the DC Fabric network to exchange overlay traffic with the other transport nodes (plus additional connectivity with the management network). Because of that, there are not VLAN extension requirements, as for the L2 and L3 Gateways, which implies the Service Nodes could be deployed in different racks (compute and edge) at the edge of the DC fabric.

However, it is a best practice to stick with the recommendation of keeping the Service Nodes as part of the Edge Racks, just in case they needed to be used also as Rendezvous points for remote gateway connectivity. In that case, having them positioned closed to the point of entrance into the physical network infrastructure would definitely represent an advantage.

Infrastructure Racks

Infrastructure racks host all the appliances and virtual machines required for management and control plane purposes (NSX Controllers, NSX Managers, Cloud Management Systems, etc.) and the storage components (filers, disk arrays, etc.). It is key that this portion of the infrastructure does not have any tenant-specific addressing and usually no requirement for sending/receiving overlay traffic. If bandwidth-intense infrastructure services are placed in these racks—IP-based storage, for example—bandwidth of these racks can be dynamically scaled, as discussed in the “High Bandwidth” subsection of the “Data Center Fabric Attributes” section.

Conclusion

The VMware NSX-MH network virtualization solution permits to overcome the current challenges with physical network infrastructure, bringing flexibility, agility and scale through the creation of overlay logical networks.

NSX-MH reproduces in the logical space typical network services traditionally delivered by the physical infrastructure, as switching, routing, security (Figure 80).

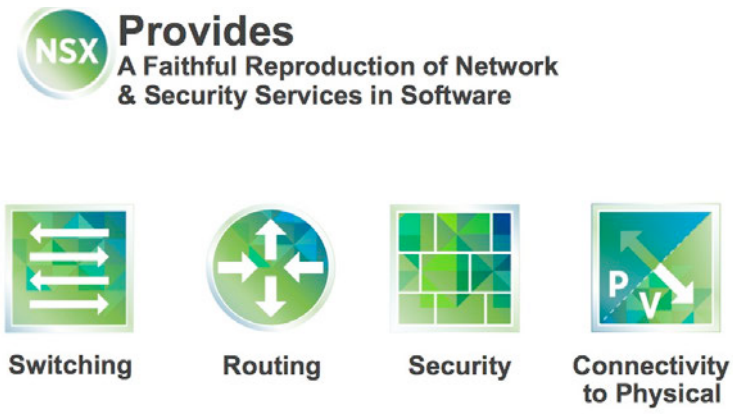


Figure 80: Logical Network Services Provided by NSX-MH

The provisioning of those logical services ensures the rapid and agile deployment of applications, completely decoupling the connectivity in the logical space from the physical network infrastructure, which consequently becomes much simpler to configure and operate. All this, can be achieved on top of any network infrastructure and deploying any type of hypervisor (ESXi, KVM, Xen).

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com