Network Virtualization Concepts

Self-Paced Microcourse

At the end of the course, you should be able to

- articulate NSX capabilities and benefits
- describe the major VMware NSX® components in the data, management, and control planes and their interactions
- apply relevant NSX features to use cases
- explain NSX network virtualization components and services
- explain how network virtualization is utilized in an SDDC environment

Module 2: Introduction to Network Virtualization

NDG - Network Development Group

VMware is a cloud computing and virtualization software provider for X86 or IBM compatible computers and servers. ... With **VMware** server virtualization, a hypervisor is installed on the physical server to allow for multiple virtual machines (VMs) to run on the same physical server.

A 5G world has virtualization at its core. As the number of out connected devices mushrooms from the hundreds of millions to the tens of billions, data centers are relying more and more on virtualized infrastructure to handle the tsunami of data that we're producing and consuming. And it's not just data centers: the fact that 100% of the Fortune 100 companies use virtualization (and VMware virtualization technology at that), tells its own story.

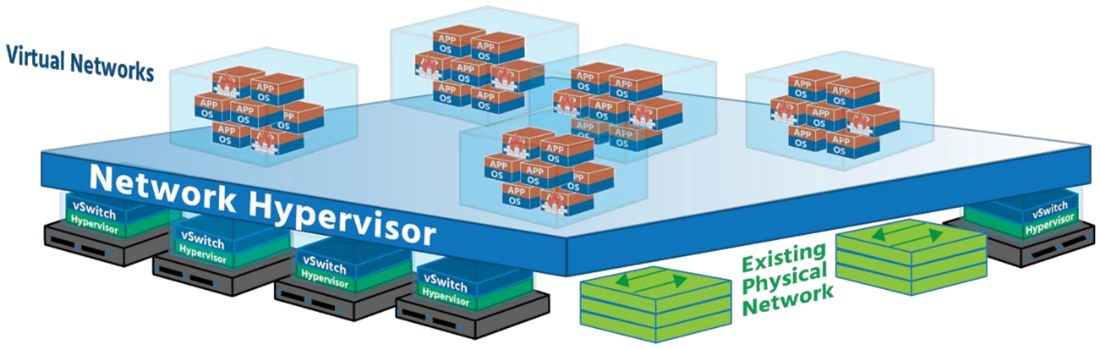
In the Software-Defined Data Center (SDDC), compute, networking, and storage infrastructure is virtualized so that resources can be pooled and used more efficiently, less expensively, and faster. Real strides have been made in server (compute) virtualization and are increasingly being seen with storage virtualization.

The efficiencies gained from them, however, have been limited to a certain extent by legacy, (i.e, traditional, non-virtual) network infrastructure that's still reliant on physical hardware and mainly manual processes. While an organization's virtualized compute and storage may be dynamic, agile, and flexible, its legacy networking just can't keep up. And an infrastructure or organization that can't keep up often gets left behind.

Network virtualization enables the speed, mobility, and security, needed in a 5G world. Infrastructure can be made ready for new applications or be changed in minutes, rather than days or weeks. Apps and workloads are no longer restricted to individual physical subnets, neither are switches, routers, firewalls, etc. The security focus moves from simply protecting the perimeter (the *outside* surface) of a data center's infrastructure to providing the ability to give each virtual machine and virtual network its own firewall, shifting the focus to the *inside* perimeter of the data center and reducing the attack surface. In addition, virtual networks are isolated. and (as we will learn later in this course) segmented from each other and from the underlying physical infrastructure so that threats cannot be spread if they do get in.

Network virtualization extends these features and many others to the cloud as well, a critical factor to the 81% of enterprises that now use multiple cloud deployment models.

What is Network Virtualization?



Network virtualization totally separates network resources from physical hardware by recreating those networking resources in software- by virtualizing them. Physical routers (which forwards data across multiple networks,) switches (which forwards data on a single *Local Area Network* or *LAN*) and load balancers (which even out workloads to prevent servers from being overwhelmed) are virtualized in the hypervisor layer using off-the-shelf, industry-standard servers (server/compute hosts). This virtualized pool can then be used as needed, on-demand. The underlying physical hardware remains important (it's still used for forwarding) but no longer needs to be reconfigured every time a new VM or container is added or updated, or every time a device on the network is moved to a different part of the network. The whole network can now be run in software.

This hypervisor-based networking software (which will include security services as well) uses a **controller** to send network services to virtual switches and *attaches* the services to individual *virtual machines (VMs)* and containers. The result is a virtual network (The exact services will be determined by the policies already assigned to the VM/containers.). In this virtual network, whenever new VMs and containers are created, the appropriate policies are automatically applied to them, and when VMs and containers move, their networking and security move with them. (You'll sometimes hear VMs and containers referred to as *workloads*, so we'll use that term here to familiarize you with it.)

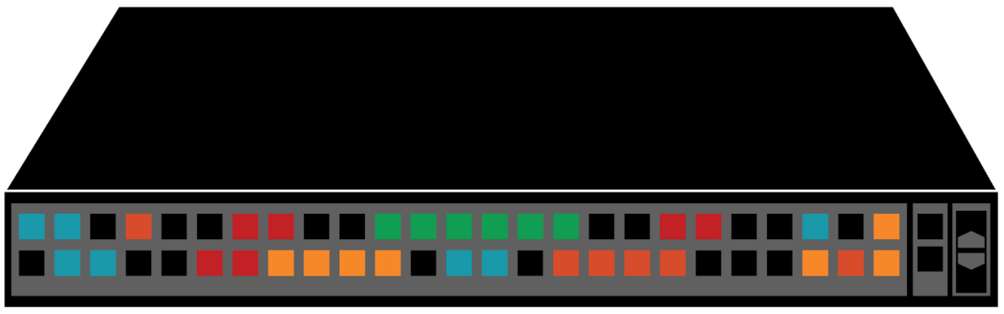
Creating a virtual network on top of a physical network is known as **overlay networking**. Imagine two devices (or *endpoints*) on an organization's network – *Finance Department Laptop* and *Sales Department Laptop*, for example, both connected to physical network ports. (VMs and containers can be *endpoints*, too.) Both laptops are given a virtual network ID (*VNID*) and assigned to a virtual network. Virtual switches then connect *Finance Department Laptop* to *Sales Department Laptop* via virtual links (software representations of physical links) that form a tunnel across the network. Each virtual link corresponds to a path in the underlying physical network.

Network virtualization works just as well in the cloud and can be managed by using a Cloud Management Platform (*CMP*) such as *VMware's vRealize Automation* or an open-source option such as *Apache CloudStack* and *OpenStack*. Hypervisor-based network virtualization can be set up and run using a Graphical User Interface (*GUI*) and a Command-Line Interface (*CLI*) or using Application Programming Interfaces (*APIs*).

Network virtualization provides administrators with tremendous flexibility. It can be used for networks as small as two connected devices, or as large as networks spanning multiple sites of major enterprises. In addition, it is flexible enough to work with any cloud or cluster (or pod if you're using a new application framework like *Kubernetes*) while having different virtual networks that can be associated with different workloads. **VMware's NSX for vSphere** (NSX-V) virtual network and security product works with ESXi (a Type 1 hypervisor - i.e., one that runs directly on the host's hardware, independent of the host's operation system), while their **NSX-T Data Center** works with ESXi and with KVM (a Type 2 hypervisor running within the host's operating system).

Virtual networks are not to be confused with *virtual local area networks* – VLANs. A VLAN takes the ports of a physical switch and groups (or isolates) them to fit a specific purpose. An organization might have its Finance Department on the first floor, HR on the second and third, Production on the fourth, Sales on the fifth, and printers scattered around the building. On one physical switch, five VLANs could be created for these separate functions, each with its own *broadcast domain*.

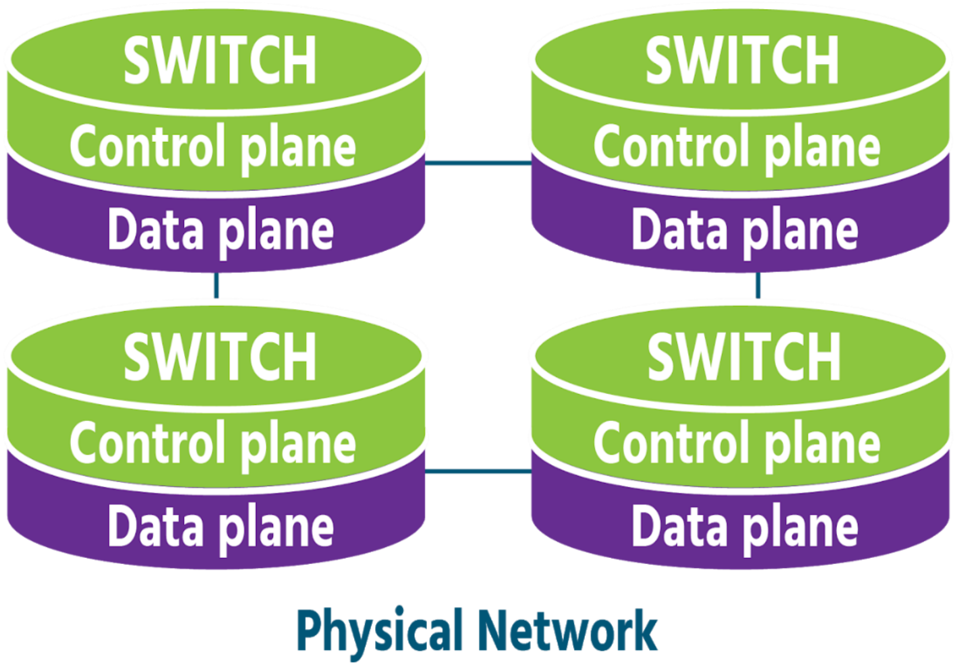
.



However, only a maximum of 4096 VLANS can be created on a Layer 2 (*Data Link Layer)*network. (This is a reference to the **OSI networking model**, which has 7 layers that together describe the different communication functions of a networking system.) This may seem like a lot, but imagine being a company that gave each of its customers 5 VLANS: after customer 819 you would have no more VLANS – which would mean no new customers. Security is an issue since VLANs are separated from a logical perspective but are actually running through the same connection, and a security breach on one can affect them all. And every time a VLAN is extended, a time-consuming physical configuration is needed.

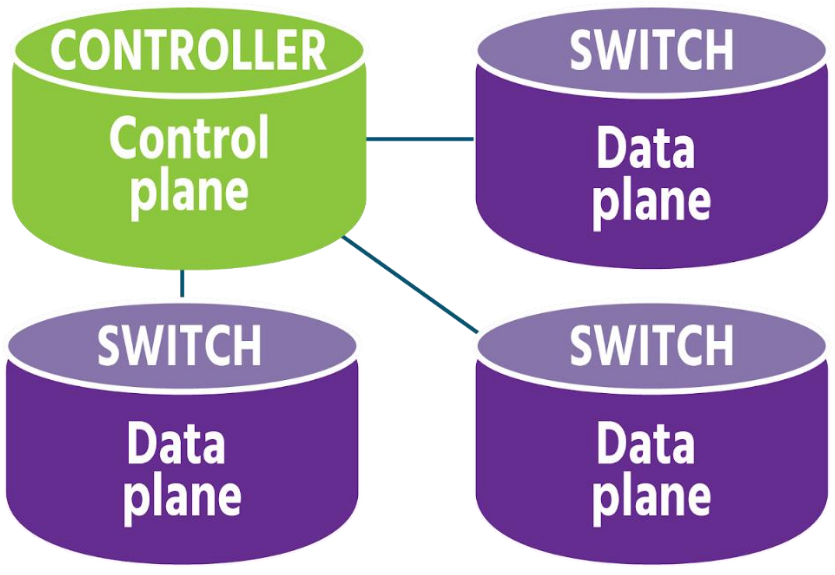
With network virtualization, on the other hand, network services beyond data transfer are also available – switching, routing, firewalling, and load balancing (Layer 3 to Layer 7 functions). The network in its entirety (Finance, HR, Production, Sales, and printers) can be recreated in software in seconds, and cloned or moved if needed; or *snapshots* representing the exact state of a network at a particular point in time can be created, saved, and used to recreate the network if required. Every networking and security service is virtualized (handled in software) and attached directly to individual workloads, reducing the need for physical configuration.

What is Software-Defined Networking?



Network virtualization and *Software-Defined Networking (SDN*) stem from a common pursuit- the goal of greater network agility- and still share certain traits:

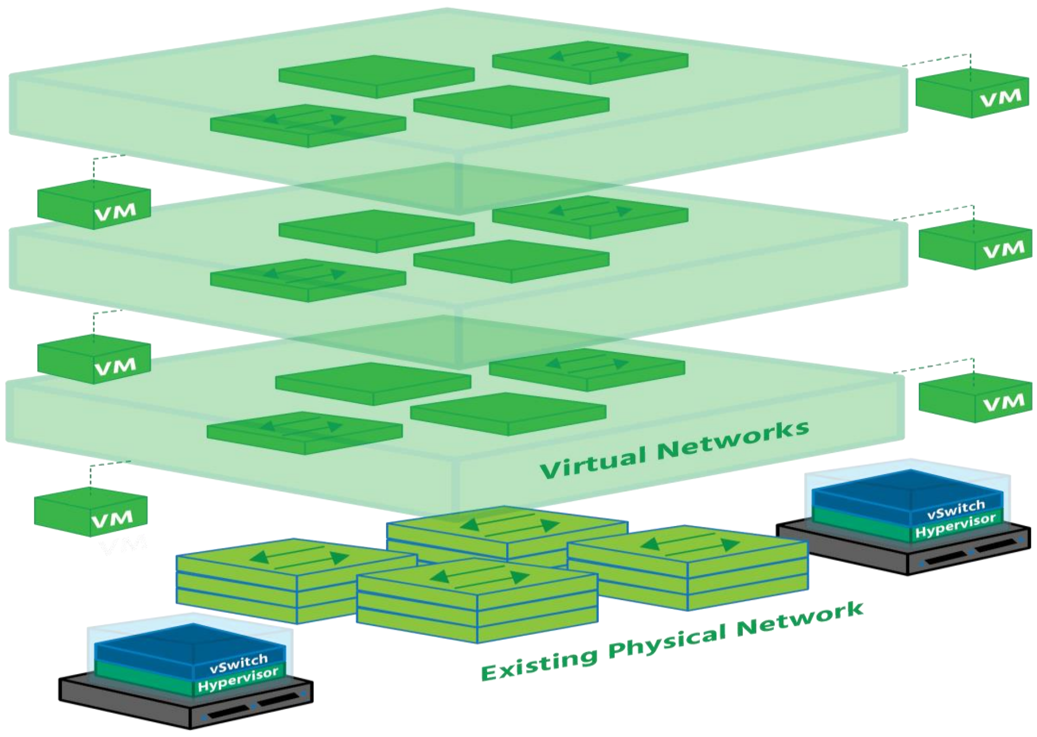
- Both use software to recreate key components of networking infrastructure
- Both separate the **control plane** (the part of the network that manages and controls it- a network's *brains*) from the **data plane** (the part where data traffic flows- a network's *muscles*) This means that with both network virtualization and SDN, network control can be programmed directly for applications and network services, without the need for manual configuration.
- Both use a **controller** that runs specialized software to centralize network management.
- Both (to varying degrees) fulfill the primary goal of increased agility by allowing administrators to quickly and precisely adjust the flow of data traffic across a network.



Software-Defined Network

Over time, however, SDN has become more broadly-defined than network virtualization, meaning different things depending on who you speak to and how they are using SDN. The thread that links these different definitions is *SDN's use of software to control networks and their physical devices.* With SDN, software controls network switches and routers, but the network is not fully virtualized, as it is with network virtualization (components, configurations, functions, and all). Hardware often still plays a major role in an SDN network.

Virtual Networks in Physical Networks

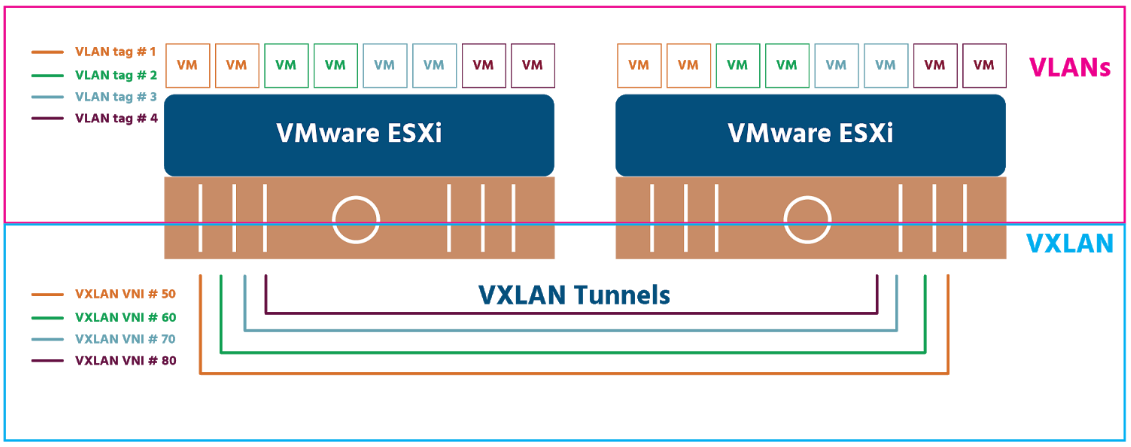


Network virtualization allows you to look at your current physical network from a fresh perspective, one in which you're no longer restricted to the capabilities of your hardware. Software vastly extends your range of possibilities, recreating your hardware as a virtualized pool that can then be used as needed, on-demand.

Imagine your physical network doubled in scope: your physical network is there doing the job it's always done, but in parallel with it you now have an identical virtual version running independently, alongside or *on top* of it. Once that virtual network has been created, it can be saved, closed, and restored later, possibly at another site. Or it can be deleted altogether.

Now imagine the scope of your physical network tripled, quadrupled, or more, because virtualization separates networking from the underlying hardware, you can create as many virtual networks (copies of the physical network) as you need. Each of these virtual networks runs in splendid isolation, unaffected by events in other virtual networks or in the data centre.

Bridging Between Virtualized Networks and Traditional VLANs



As we mentioned in section 2.1, creating a virtual network on top of a physical network is known as **overlay networking**. The underlying infrastructure becomes the *underlay* - also known as the physical (Layer 3) network. Several overlay methodologies exist. Two of the most widely-used are **Virtual Extensible Local Area Network** (VXLAN) and **Generic Network Virtualization Encapsulation** (GENEVE). It is important to note that VXLAN is vendor-neutral and has been recognized as RFC (Request for Comments) 7348, which is a formal document from the Internet Engineering Task Force (IETF).

VXLAN works on hardware (e.g., on routers or switches), on software (e.g., on a hypervisor) or on both (hybrid). It uses 24-bit binary identifiers (from 000000000000000000000 to 111111111111111111111111 and everything in between) meaning that a maximum 16,777,215 VXLANs are possible. Compare that to the maximum 4096 VLANs permitted by their 12-bit identifiers! A VXLAN ID is called a **VXLAN Network Identifier** (VNI). Each VNI is a separate virtual network that runs in the overlay network which are also known as **bridge domains**.

VXLAN Tunnel Endpoints (*VTEPs*) connect the physical network to the overlay network. Every VTEP has an IP address in the physical network and one or more VNIs in the overlay network. Encapsulated traffic (traffic that's had certain information added to it at key stages of its journey - see section 4) is transferred between hosts via a stateless tunnel that is created between a source VTEP and a destination VTEP. By the time data from a host reaches a VXLAN switch, it's in the form of frames, specifically "inner MAC frames" which include MAC (i.e., hardware) address information and data. The switches add a "VXLAN header" containing the 24-bit VNI.

The source VTEP then adds the *IP address* of the destination VTEP in an IP header, as well as its own IP address. It adds a **User Data Protocol** (UDP) header (UDP being the transport protocol that VXLAN uses). The MAC address of the next physical device that the frame will be delivered to on its journey is added in an *Ethernet header*. The physical network (the underlay) forwards the encapsulated frame on to the destination VTEP, which removes the headers in a process called **decapsulation** (mentioned again in section 4). The frame is then delivered to the destination host.

GENEVE is a relatively new entrant to the tunnel protocol field. It was jointly developed and drafted as an IETF proposal by Intel, Microsoft, Red Hat, and VMware and released in 2014. At the time this micro-course was written, GENEVE is currently going through the IETF process to become an RFC itself and so GENEVE is equally vendor-neutral. It works almost identically to VXLAN but is more flexible because it offers control plane independence between tunnel endpoints. And there's a slight difference in terminology: GENEVE does not have VTEPs (VXLAN tunnel endpoints), just tunnel endpoints (TEPs).

It is important to note that NSX-V utilizes VXLAN overlay encapsulation, and NSX-T uses GENEVE.