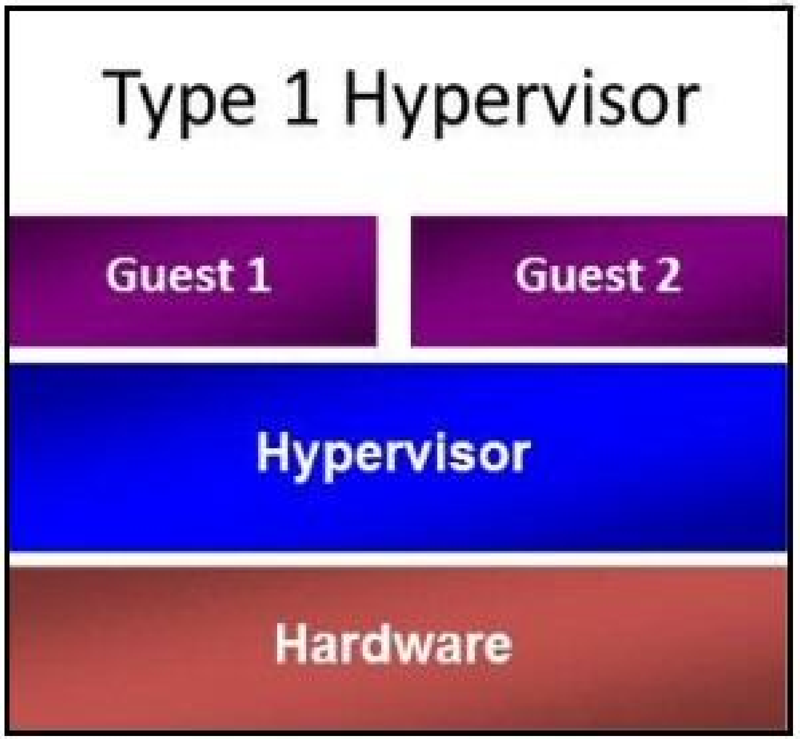
Module 2: Network Virtualization

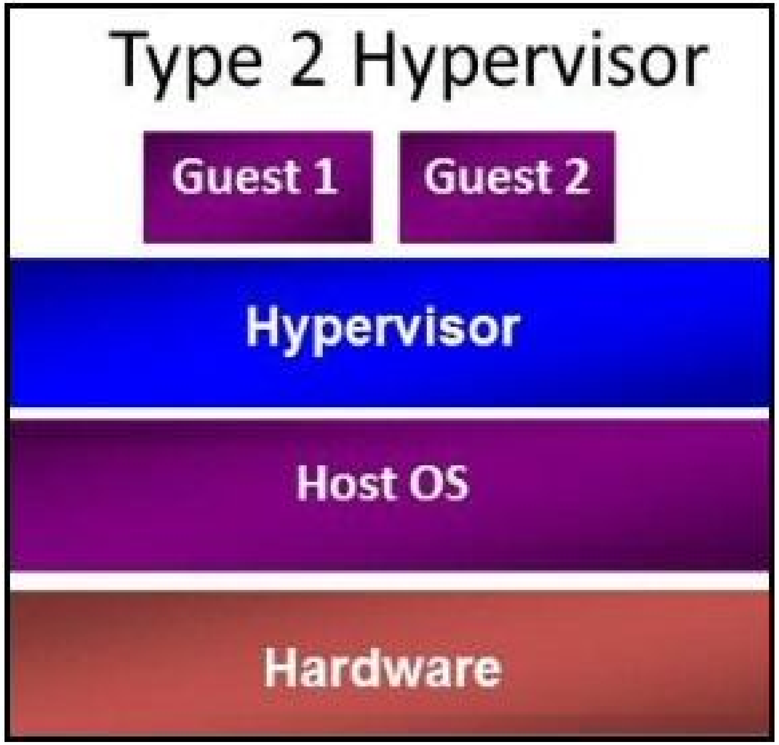
VIRTUALIZATION

- Virtualization is the creation of a virtual - rather than actual - version of something, such as an operating system, a server, a storage device or network resources.
- Virtualization is technology that allows you to create multiple simulated environments or dedicated resources from a single, physical hardware system.
- Software called a **hypervisor** connects directly to that hardware and allows you to split 1 system into separate, distinct, and secure environments known as Virtual Machines (VMs).
- Hypervisor Also called as Virtual Machine Monitor (VMM) and can be a piece of software, firmware or hardware that gives an impression to the guest machines (VM) i.e., virtual machines as if they were performing on a physical hardware.

- There are two types of Hypervisor

1. Type I Hypervisor
2. Type II Hypervisor

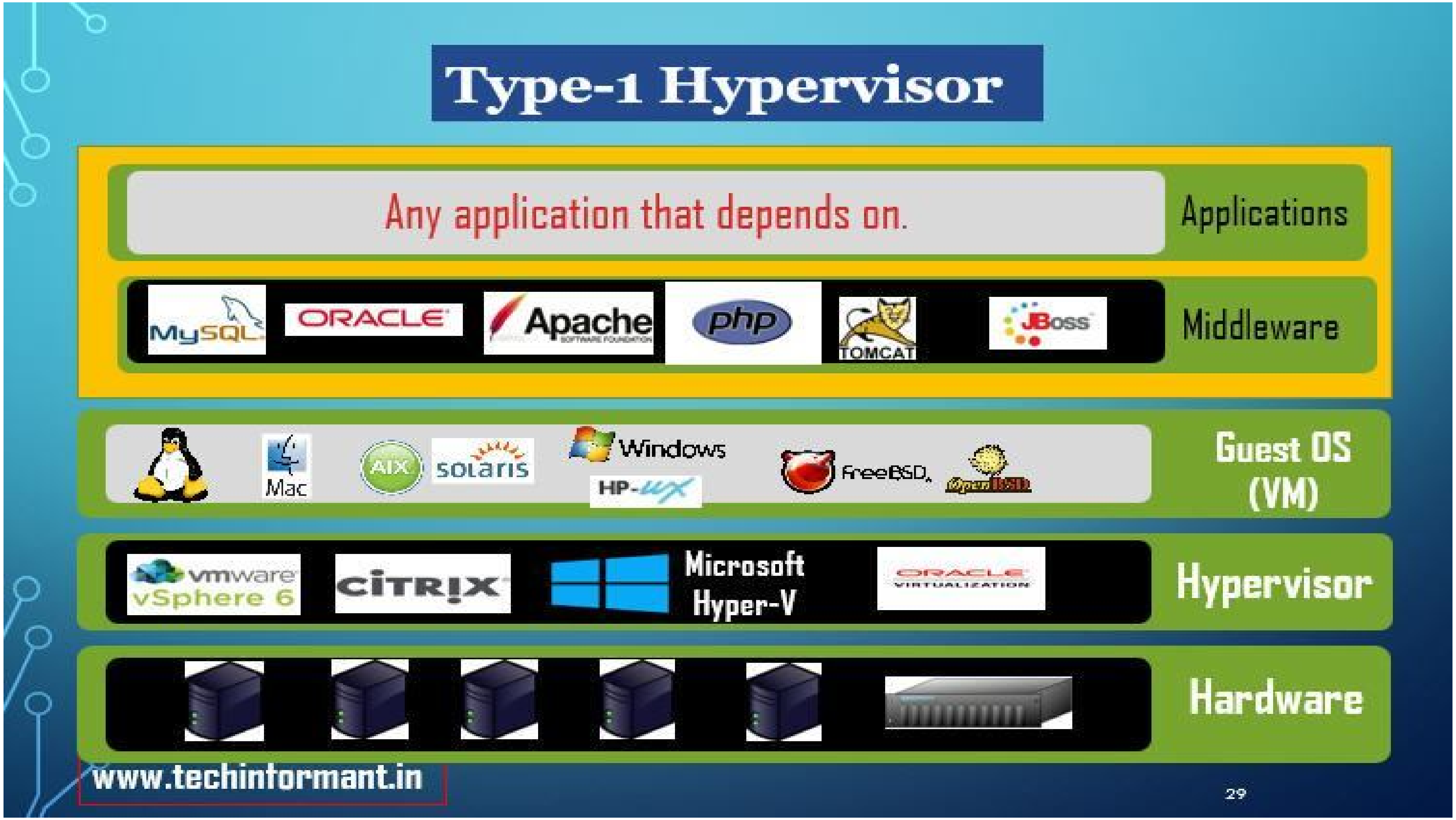




1. Type I Hypervisor:

- Considered as a bare-metal hypervisor and runs directly on top of hardware.
- Often referred as a hardware virtualization engine.
- A Type 1 hypervisor provides better performance and greater flexibility because it operates as a thin layer designed to expose hardware resources to virtual machines (VMs), reducing the overhead required to run the hypervisor itself.
- Servers that run Type 1 hypervisors are often single-purpose servers that offer no other function.
- It requires management console to manage the hypervisor & instances of OS installed on it
- We can move instances of OS from one hypervisor to another using management console (Software).
- E.g VMWare ESX/ESXi (Hypervisor) & Vsphere (Management S/W) , Oracle virtual box

Type-1 Hypervisor



Any application that depends on.

Applications

Middleware

Guest OS (VM)

Hypervisor

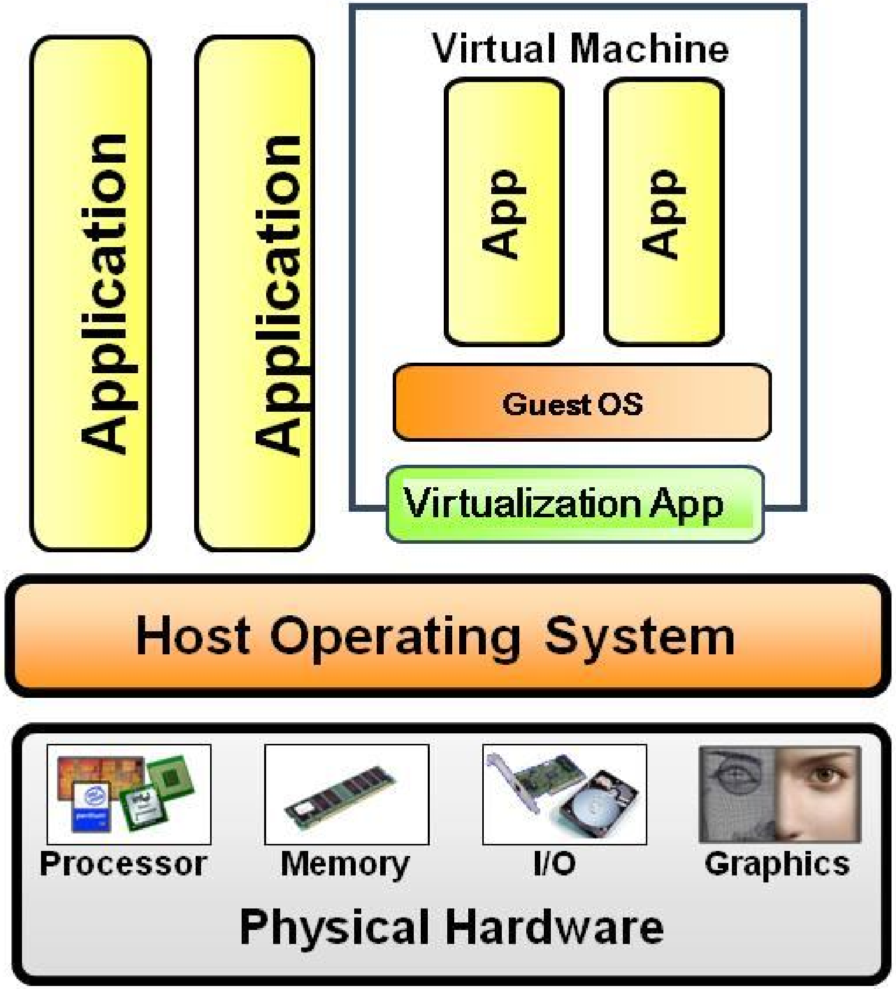
Hardware

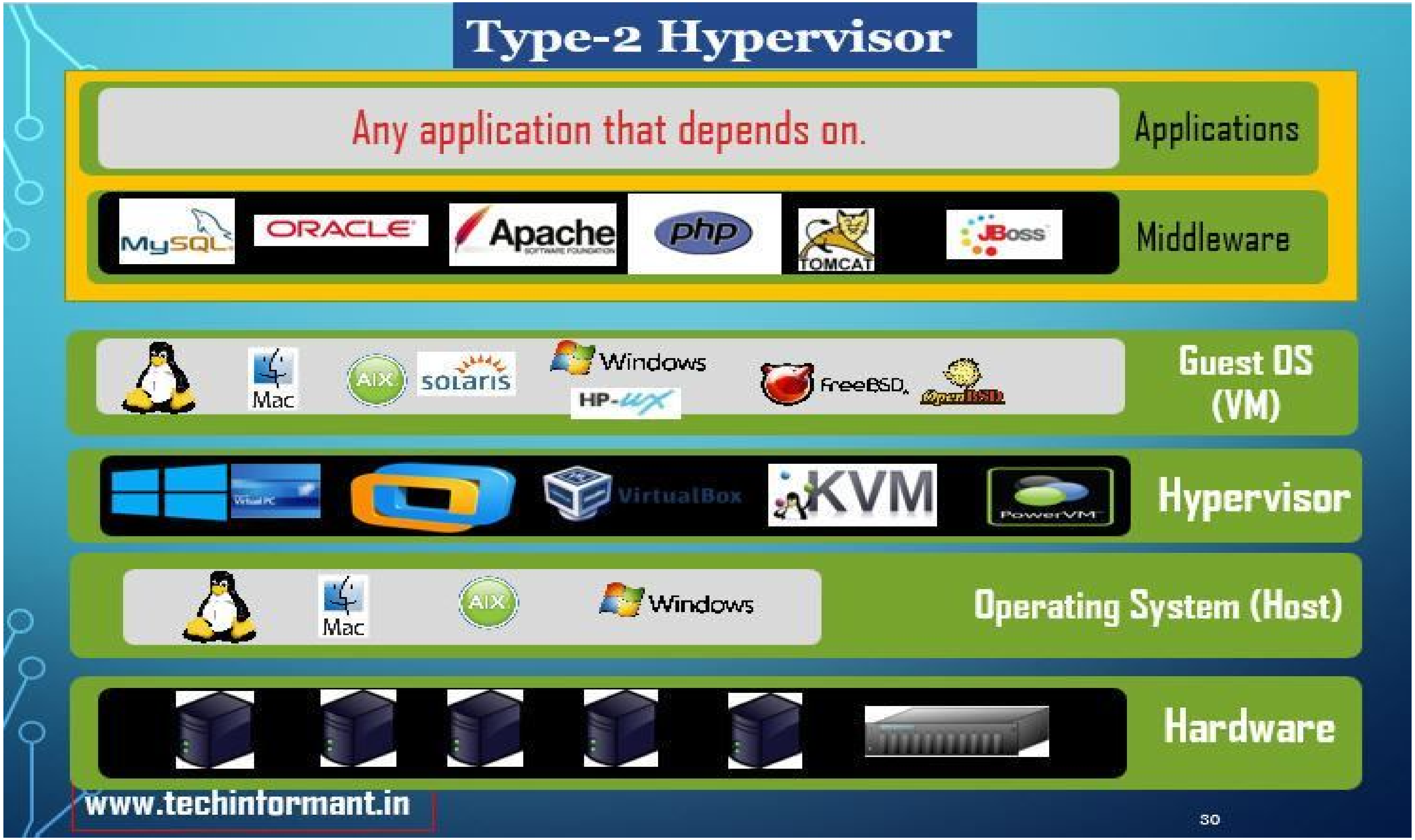
www.techinformant.in

2. Type II Hypervisor:

- ➢ Operates as an application on top of an existing (host) operating system that provides virtualization service.
- ➢ Not directly installed on hardware, so management console is not required.
- ➢ It is installed on Existing OS i.e Host OS so called as <u>Hosted Hypervisor</u>.
- ➢Mostly used for end-user virtualization.
- ➢ Resource allocation is very important.
  - If you allocate 4GB of RAM to any one instance then it will take all the RAM allocated to it even if it is not using it.
  - E.g. Suppose you are having 5GB of RAM, out of which 2GB is required for host OS to run.
  - again you allocated 2GB to 1 instance, 1 GB to another instance, 1GB to one more instance then total 4GB of RAM is allocated, so only 1 GB is remaining so host computer will crash in this case.
- ➢Examples are, VMware Workstation, VMware Player, VirtualBox, Parallels Desktop for Mac, Microsoft Virtual Server

- The original, physical machine equipped with the hypervisor is called the host,
- While the many VMs that use its resources are called guests.
- These guests treat computing resources—like CPU, memory, and storage—as a store (Hangar) of resources that can easily be relocated.





Type-2 Hypervisor

Any application that depends on.

Applications

Middleware

Guest OS (VM)

Hypervisor

Operating System (Host)

Hardware

www.techinformant.in

- Benefits:
  - Cost Saving
  - Agility and Speed
  - Lowers the downtime

Network Virtualization: Concept

- In computing, **network virtualization** is the process of combining hardware and software network resources and **network** functionality into a single, software-based administrative entity, i.e. a virtual **network**.
- The network relevant to the virtual machines is sometimes more specifically referred to as the *virtual network.*

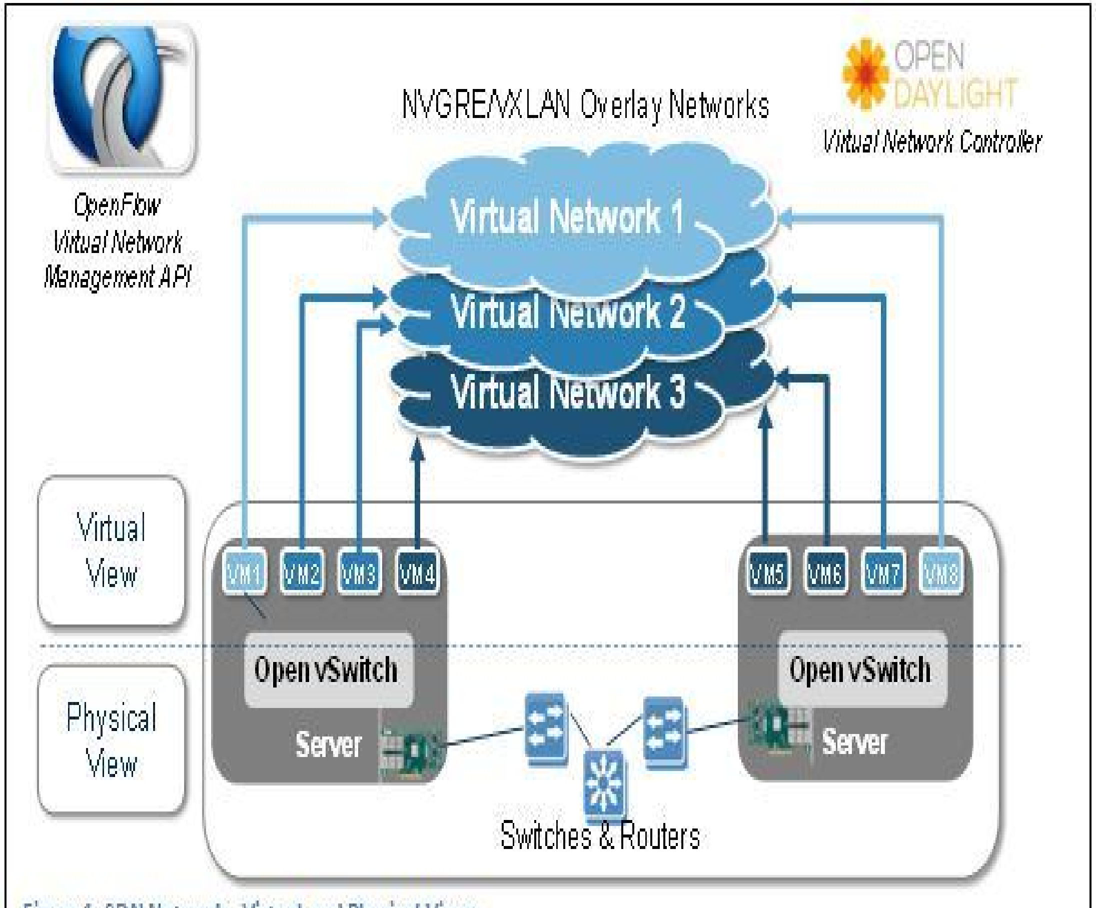
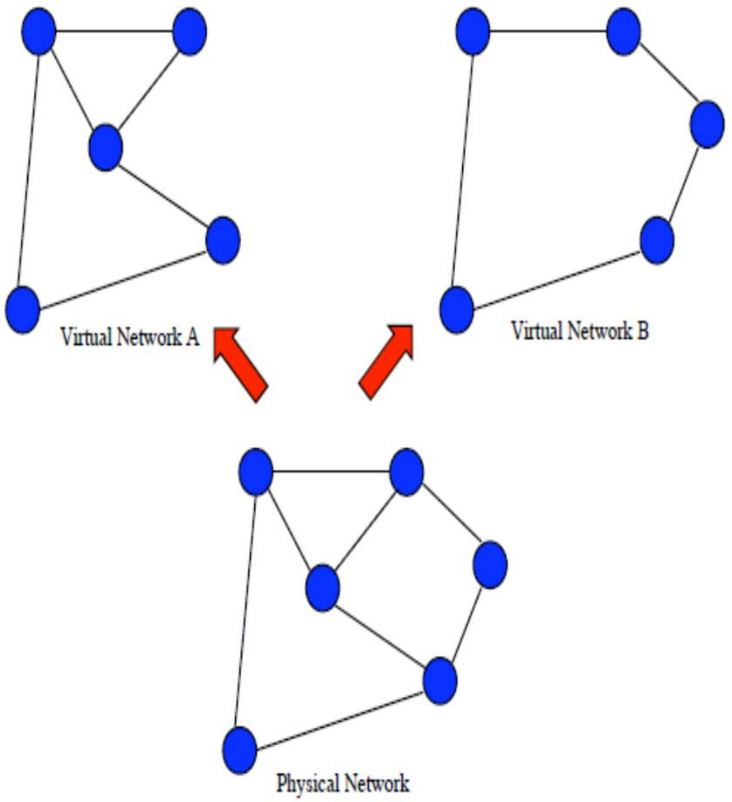


Figure 1: SDN Network - Virtual and Physical Views

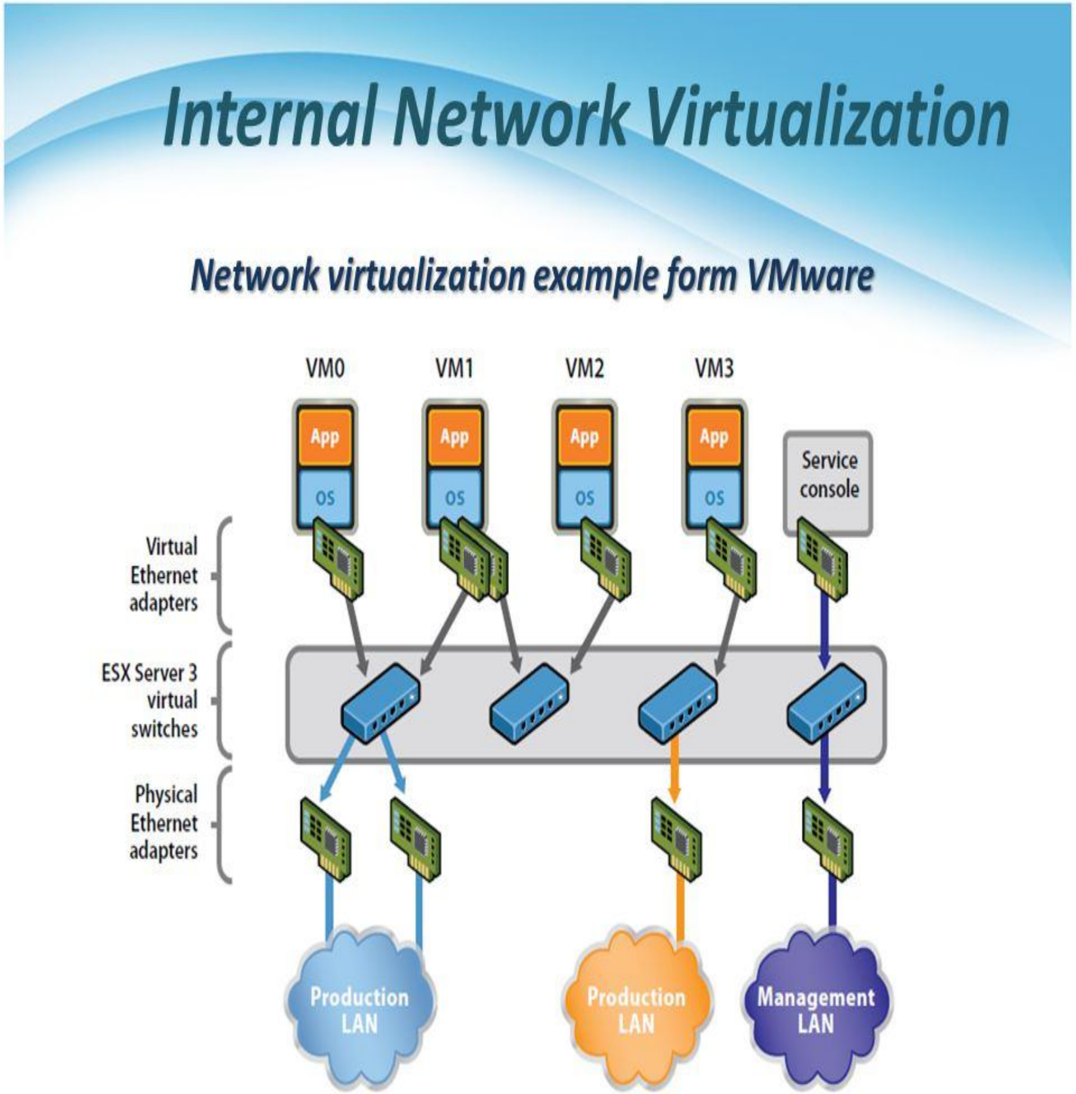
Network Virtualization: Concept

- Allows the creation of multiple virtual networks on same substrate
- Each virtual network:
  - is a collection of virtual nodes and virtual links
  - is a subset of underlying physical network resources
  - co-exists with, but is isolated from, other virtual networks.
- https://youtu.be/HFQdbOY8Ams
- https://youtu.be/QE5y5PLqZZI
- Network virtualization is categorized as either **external virtualization & internal virtualization.**



- **External Virtualization:**
  - Combines or subdivides one or more local area networks (LANs) into virtual networks to improve a large network's or data center's efficiency.
  - A virtual local area network (VLAN) and network switch comprise the key components.
  - A system administrator can configure systems physically attached to the same local network into separate virtual networks.
  - Conversely, can combine systems on separate local area networks (LANs) into a single VLAN spanning segments of a large network.
  - Acts on systems outside of a single server

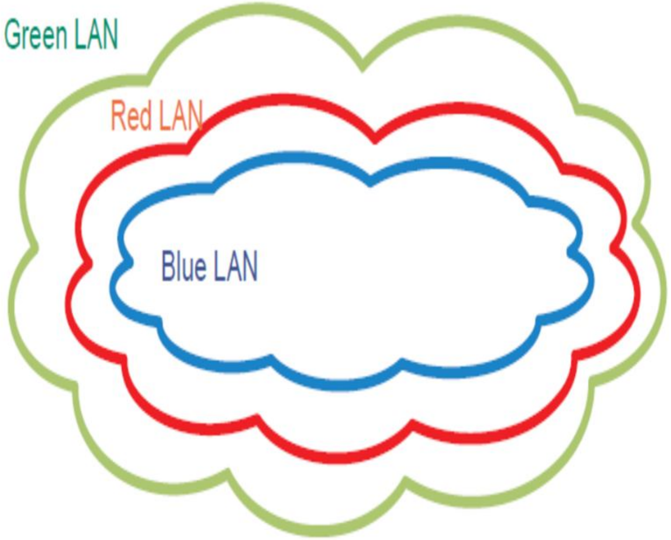
- **Internal Virtualization:**
  - Acts within 1 server to emulate a physical network.
  - Configures a single system with software containers, such as Xen hypervisor control programs, or pseudo-interfaces, such as a VNIC, to emulate a physical network with software.
  - Can improve a single system's efficiency by isolating applications to separate containers or pseudo-interfaces
  - With containers, individual applications can be isolated or different operating systems can be run on the same server.

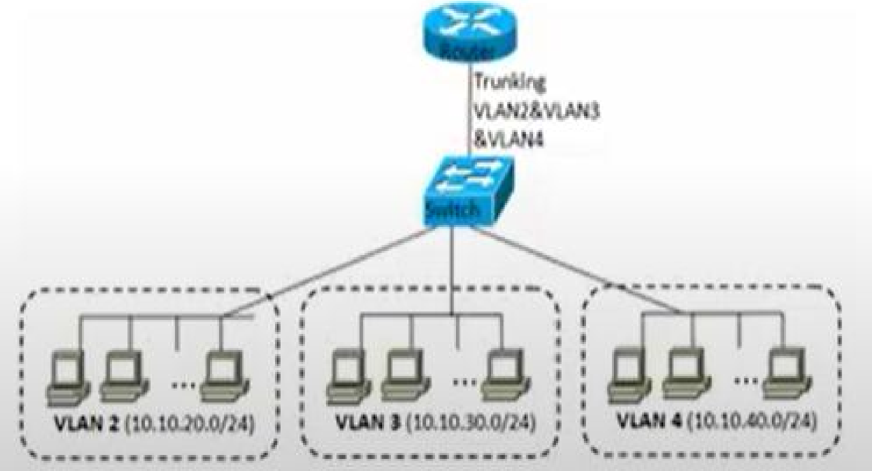

Internal Network Virtualization
Network virtualization example form VMware
VM0
VM1
VM2
VM3
App
OS
Service console
Virtual Ethernet adapters
ESX Server 3 virtual switches
Physical Ethernet adapters
Production LAN
Production LAN
Management LAN

- **Historical Perspective:**
  - Virtual LAN (VLAN)
  - Virtual service network (VSN)
  - Virtual private network (VPN)
  - Active and programmable networks
  - Overlay networks

Virtual LANs

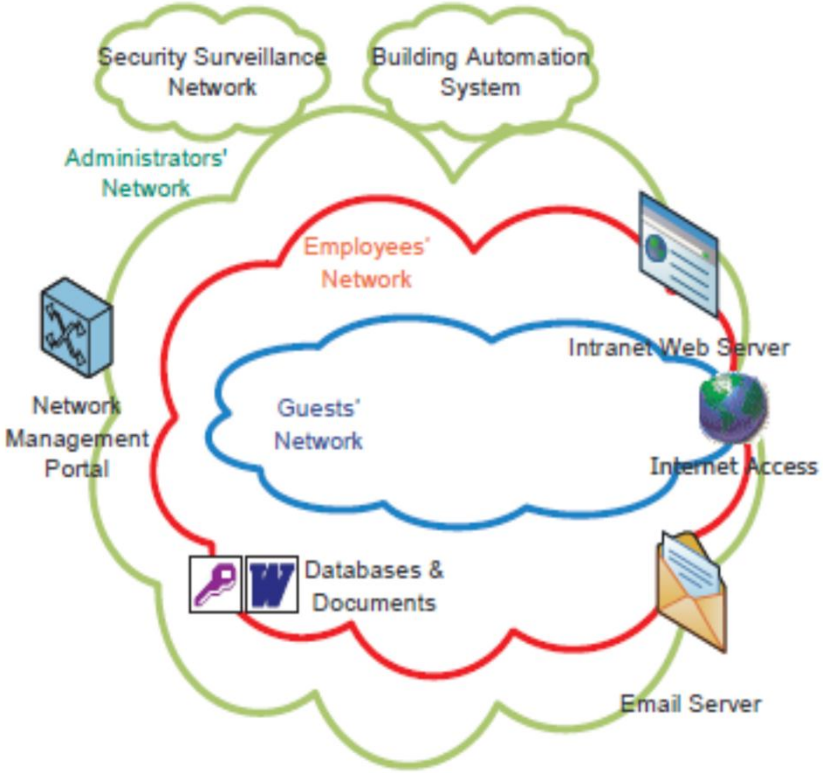
- L2 constructs, share same physical LAN infrastructure
- Groups of hosts with common interests, regardless of connectivity
- Segmented within boundary of broadcast domains-frame colouring
- High levels of trust, security, isolation, easy to configure
- Provide a basic form of topology management





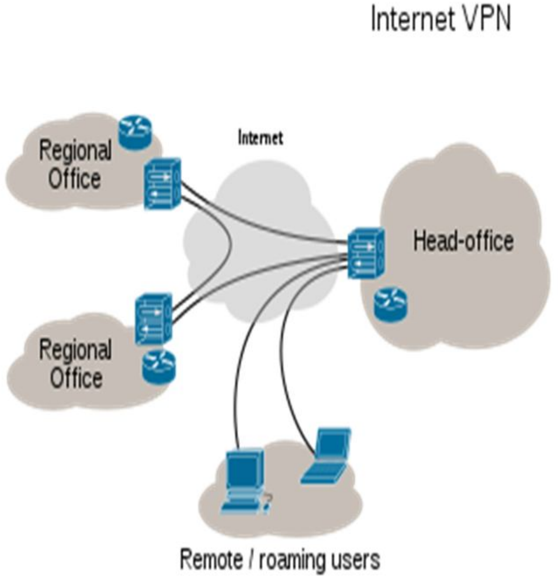
Virtual Service Networks

- Generalization of VLAN concept to broader network
- Define multiple network instances that
  - share the physical resources
  - are properly segmented (functionality, access permissions, etc.)



Virtual Private Networks

- Extends a private network across a public network, and enables users to send and receive data across shared or public networks as if their computing devices were directly connected to the private network.
- Connect multiple sites using tunnels over public network
- VPN technology was developed to allow remote users and branch offices to access corporate applications and resources.
- To ensure security, the private network connection is established using an encrypted layered tunnelling protocol and VPN users use authentication methods, including passwords or certificates, to gain access to the VPN.
- Only provide traffic isolation



Active and Programmable Networks

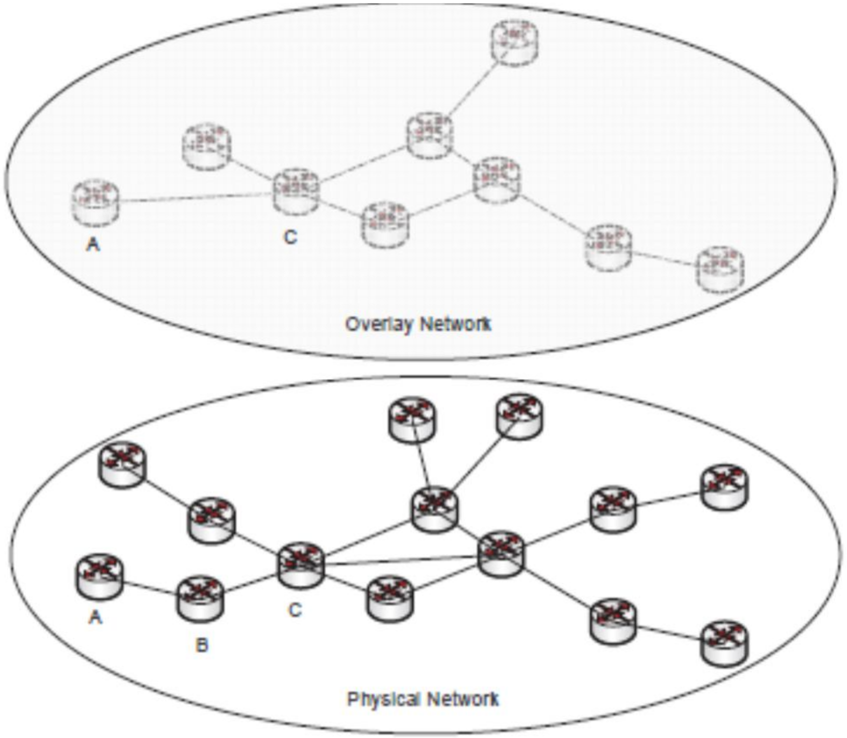
- Support co-existing networks through programmability
- Customized network functionality through
    - programmable interfaces to enable access to switches/routers
    - active code
        - call functions already installed
        - allow user to execute own code at switches/routers

ACTIVE NETWORKS

Traditional Network	Active Network
Fossilized: Resistant to Change	Built for Change
Layers of Complexity *O(4000)* Request For Comments (RFC)s	Reduced Complexity
Inability to Customize Quickly or Efficiently	Rapid, Efficient Customization
Lack of Security Paradigm	Security Paradigm Built-in
Downward Side of the Innovation Curve	Upward Innovation Path

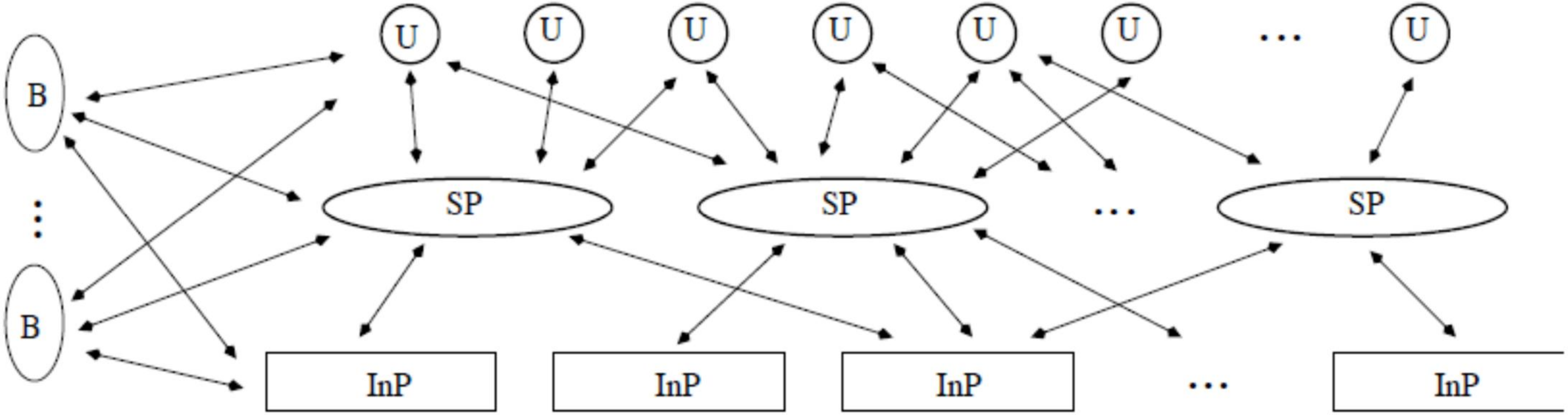
Overlay Networks

- Application layer virtual networks
- Built upon existing network using tunnelling/encapsulation
- Implement new services without changes in infrastructure
- Application-specific, not flexible enough
- It is a telecommunications network

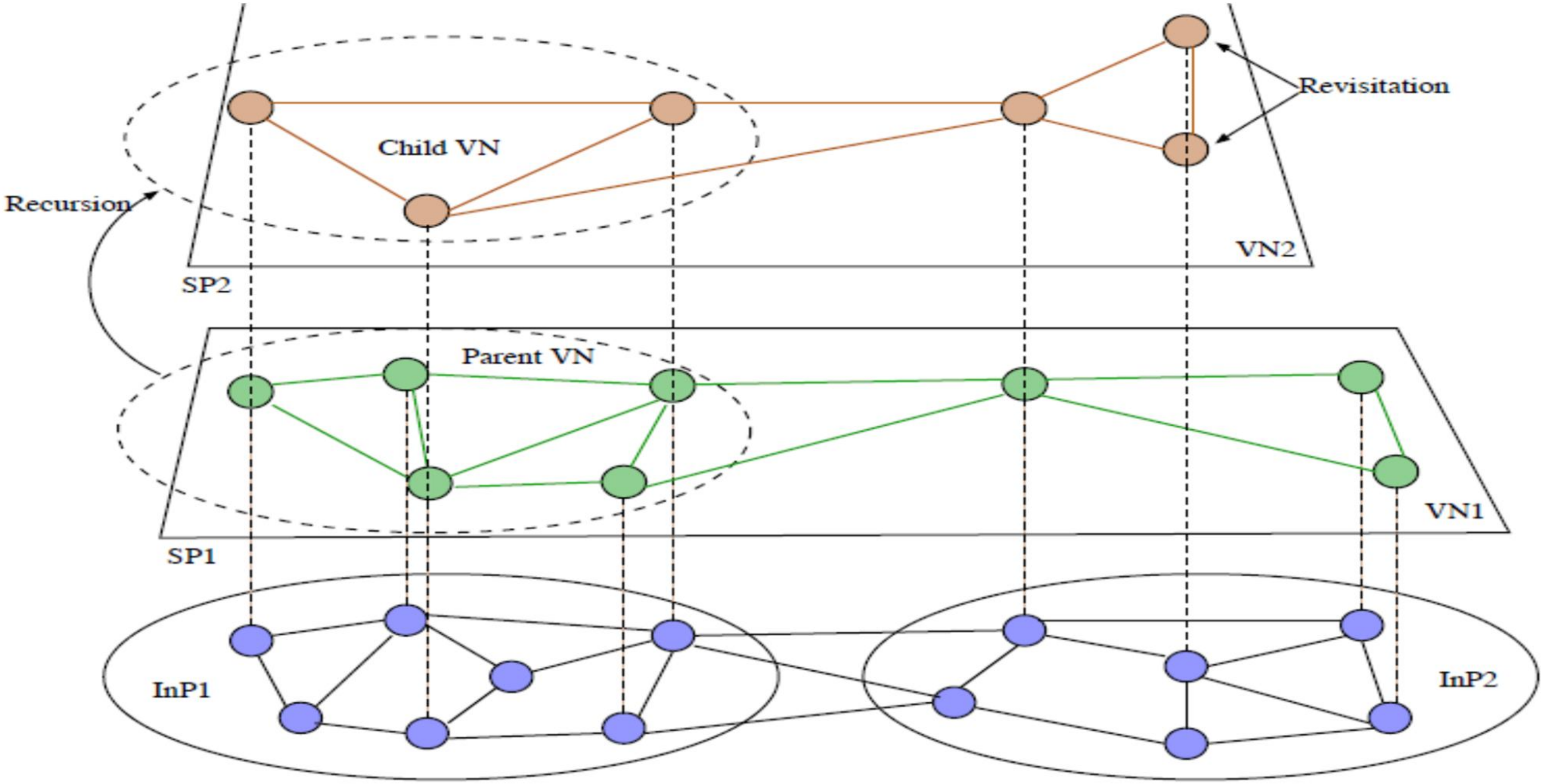


Reference Model

- **Infrastructure providers:** manage physical networks
- **Service providers:**
  - create and manage virtual networks
  - design and deploy end-to-end services
- **Users:** select from competing services offered by providers
- **Brokers:** aggregate offers, match requirements to services/resources



Architectural Principles



- Concurrency:
  - multiple service providers compete
  - multiple virtual networks coexist! diversity
- Nesting:
  - virtual network hierarchy
  - child VNs derived from parent VNs
- Inheritance:
  - child VN inherits architectural attributes
  - creation of value-added services
  - VN economics
- Revisitation:
  - single physical node hosts multiple virtual nodes
  - deploy diverse functionalities
  - simplify operation and management

- Because the virtual resources are software defined, the manager or administrator of a virtual network potentially has a great deal of flexibility
  - In altering the topologies,
  - Moving resources and
  - Changing the properties and service of various resources
- In addition virtual network users can include not only users of service or applications but also service providers
- For example, a cloud service provider can quickly add new services or expanded coverage by leasing virtual networks as needed
- https://youtu.be/uLfPSRg5QT0

- To get some feel for the concepts involved in network virtualization, we begin with a simplified example.
- Figure 9.9, shows a network consisting of three servers and five switches.
- One server is a trusted platform with a secure operating system that hosts firewall software.
- All the servers run a hypervisor (virtual machine monitor) enabling them to support multiple VMs.
- The resources for one enterprise (Enterprise 1) are hosted across the servers and consist of three VMs (VM1a, VM1b, and VM1c) on physical server 1, two VMs (VM1d and VM1e) on physical server 2, and firewall 1 on physical server 3.
- The virtual switches are used to set up any desired connectivity between the VMs across the servers through the physical switches.
- The physical switches provide the connectivity between the physical servers.

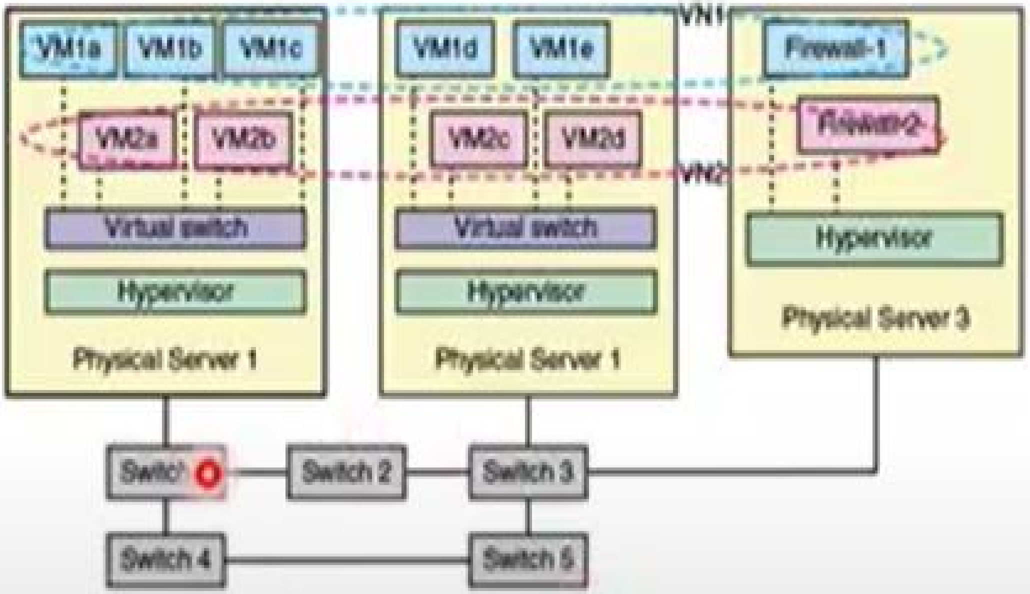
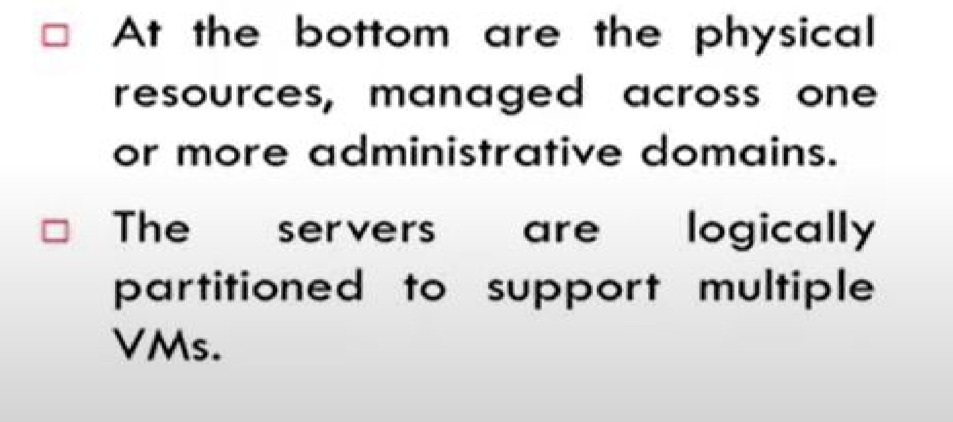


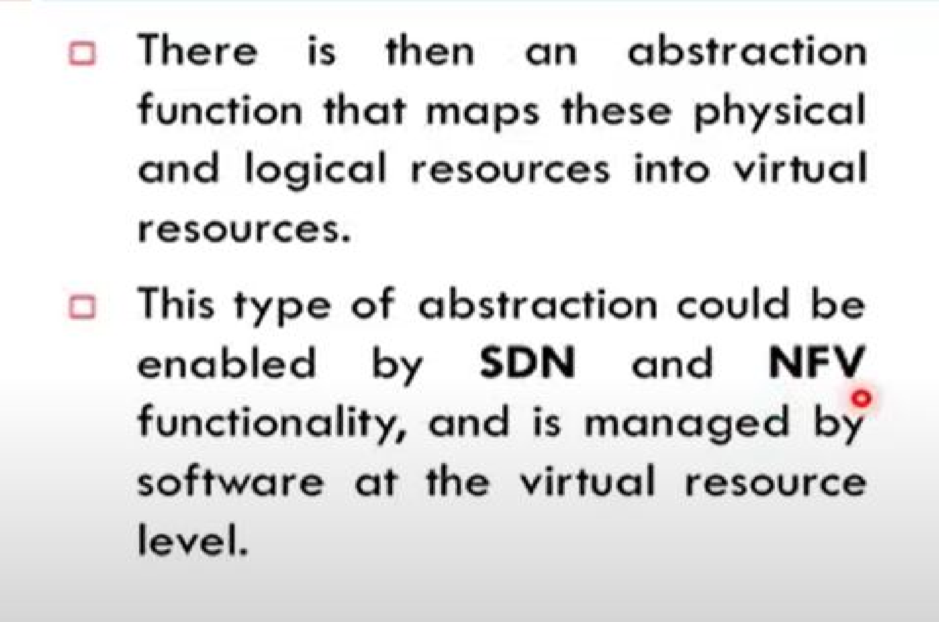
FIGURE 9.9 Simple Network with Virtual Machines Assigned to Different Administrative Groups

- Because resources are defined in software, network virtualization provides a great deal of flexibility, as this example suggests.
- The manager of virtual network 1 may specify certain QoS requirements for traffic between VMs attached to switch 1 and VMs attached to switch 2, and may specify firewall rules for traffic external to the virtual network.

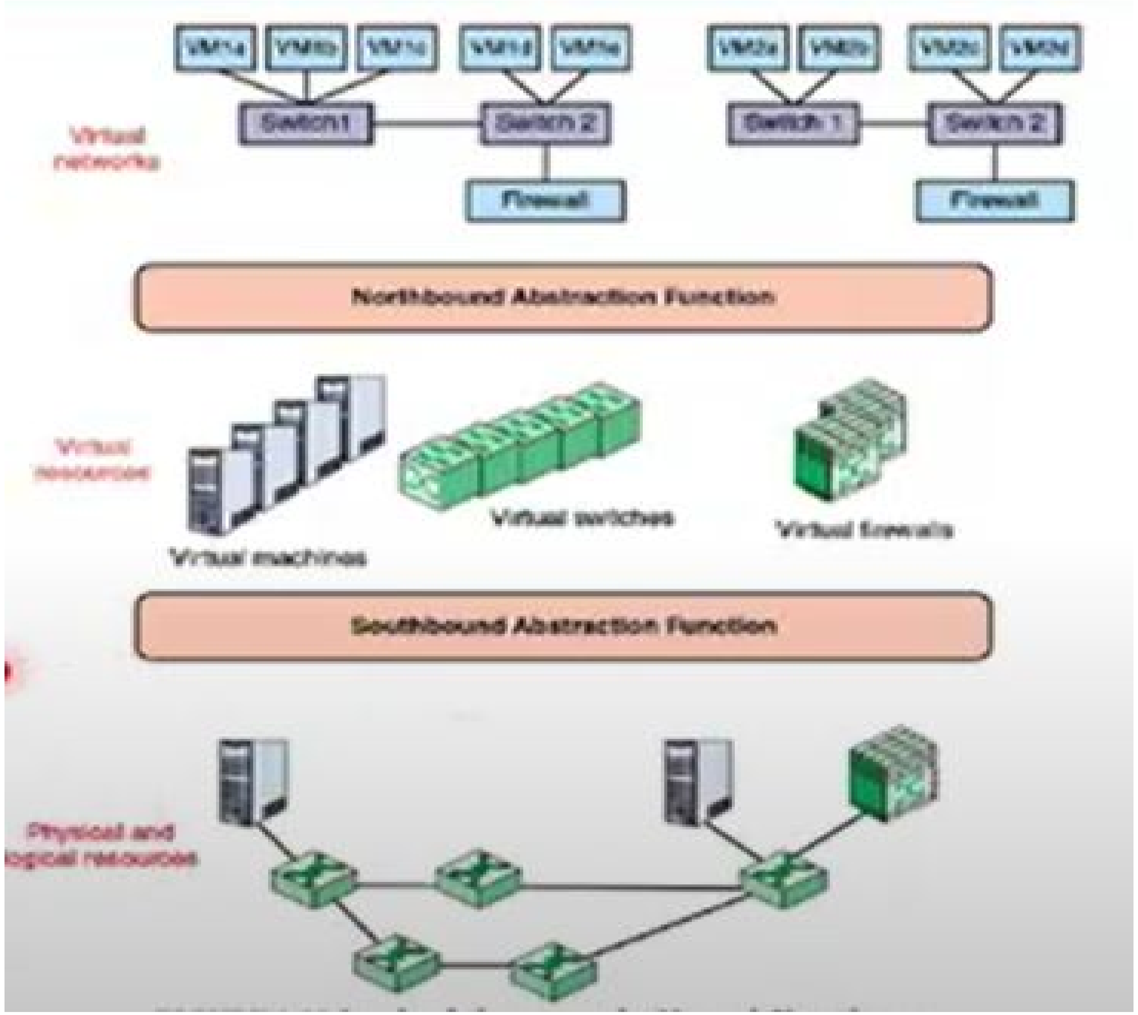
Levels of abstraction for Virtual Networks



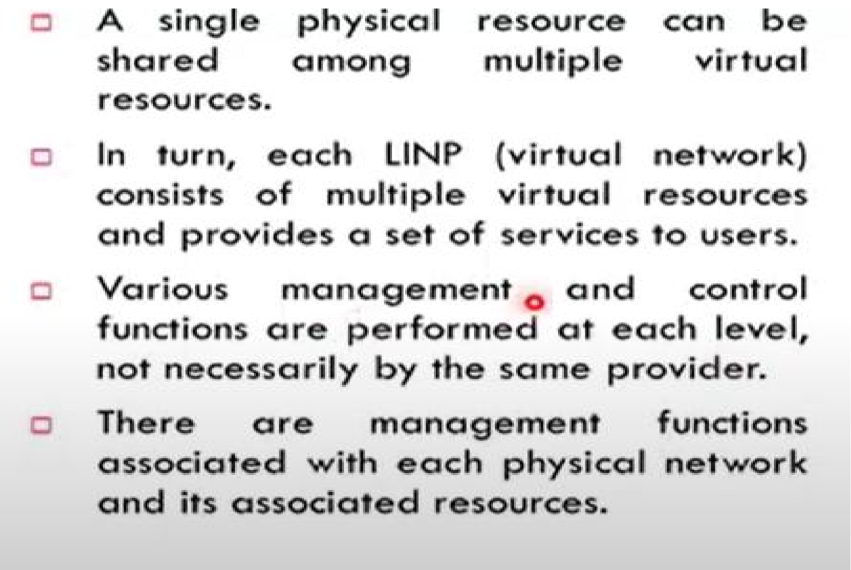
- □ At the bottom are the physical resources, managed across one or more administrative domains.
- □ The servers are logically partitioned to support multiple VMs.



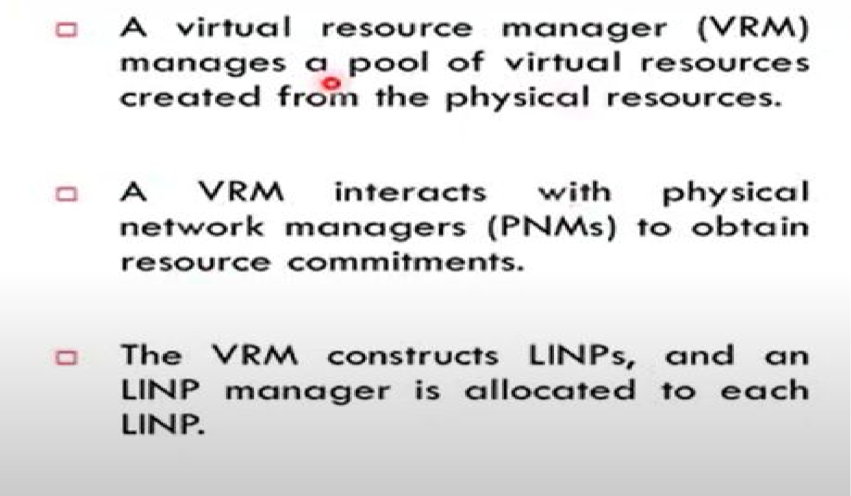
- □ There is then an abstraction function that maps these physical and logical resources into virtual resources.
- □ This type of abstraction could be enabled by **SDN** and **NFV** functionality, and is managed by software at the virtual resource level.



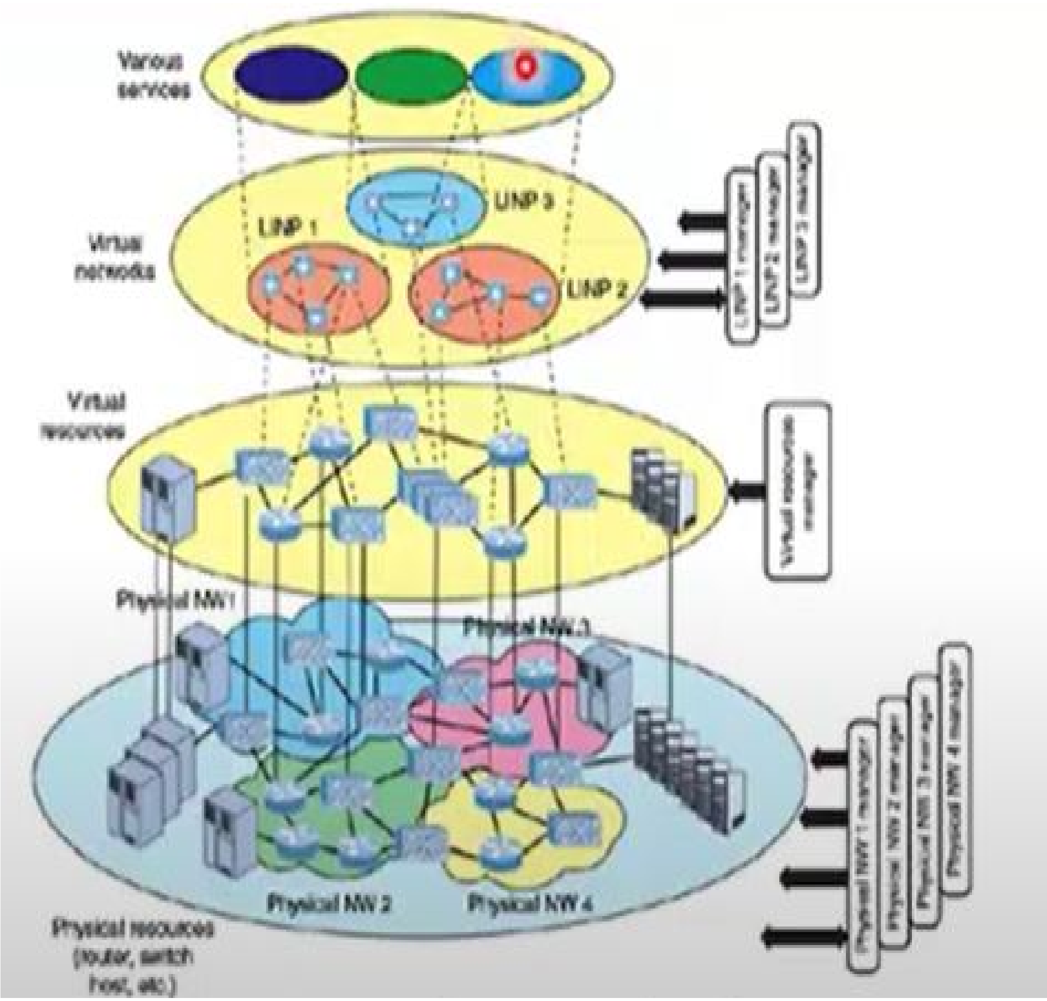
Levels of abstraction for Virtual Networks



- □ A single physical resource can be shared among multiple virtual resources.
- □ In turn, each LINP (virtual network) consists of multiple virtual resources and provides a set of services to users.
- □ Various management and control functions are performed at each level, not necessarily by the same provider.
- □ There are management functions associated with each physical network and its associated resources.



- □ A virtual resource manager (VRM) manages a pool of virtual resources created from the physical resources.
- □ A VRM interacts with physical network managers (PNMs) to obtain resource commitments.
- □ The VRM constructs LINPs, and an LINP manager is allocated to each LINP.



Design Objectives

- Isolation
  - in terms of logical view, resources, operation
  - attacks, failures, bugs, misconfigurations do not affect other VNs.
- Flexibility
- service providers may assemble VNs with customized
  - network topology
  - control and data plane functionality
  - without the need to coordinate with each other
- Scalability
  - support multiple VNs
  - utilize resources efficiently

Design Objectives

- Programmability
  - deploy customized protocols/services in network elements
  - at packet level or below (e.g., optical devices?)
  - how to expose to service providers/users
  - opens up vulnerabilities (active networks redux?)
- Heterogeneity
  - support a diverse set of
    - physical network technologies (wireless, sensor, optical, · · ·)
    - virtual network capabilities
- Manageability
  - mechanism"policy
  - “know what happened”!SPs and InPs held accountable

Design Objectives

- Dual use
  - experimental and deployment facility
  - design, evaluate, and deploy new services
  - plausible pathway for adopting radically new network architectures
  - PlanetLab, GENI, VINI
- Legacy support
  - existing Internet: another instance of a VN
  - deploy IPv6 as VN!compete with IPv4

Components of NV

- Various equipment and software vendors offer network virtualization by combining any of the following:
  - Network hardware, such as switches and network adapters, also known as network interface cards (NICs)/ VNIC
  - Network elements, such as firewalls and load balancers
  - Networks, such as virtual LANs (VLANs) and containers such as virtual machines (VMs)
  - Network storage devices
  - Network machine-to-machine elements, such as telecommunications devices
  - Network mobile elements, such as laptop computers, tablet computers, and smart phones
  - Network media, such as Ethernet and Fibre Channel

7 Properties of Network Virtualization:

1. **Independence from network hardware.**
   - A network virtualization platform must be able to operate on top of any network hardware, much like x86 server hypervisors.
   - This independence means the physical network can be supplied by any combination of hardware vendors.
2. **Faithful reproduction of the physical network service model**
   - The vast bulk of enterprise applications have not been written as web applications.
   - The cost/payback ratio of rewriting tens of billions of dollars of application development is neither realistic nor even possible.
   - Therefore, a network virtualization platform must be able to support any workload that runs within a physical environment today.
   - In order to do so, it must recreate Layer 2 and Layer 3 semantics fully, including support for broadcast and multicast.

- Commonly, virtual networks are migrated from or integrated with physical environments where it is not possible to change the current addresses of the VMs.
- Therefore, it is important that a virtual network environment not dictate or limit the addresses that can be used within the virtual networks, and that it allows overlapping IP and MAC addresses between virtual networks.

3. **Following an operational model of compute virtualization.**

- A key property of compute virtualization is the ability to treat a VM as soft state, meaning it can be moved, paused, resumed, snapshotted, and rewound to a previous configuration.
- In order to integrate seamlessly in a virtualized environment, a network virtualization solution must support the same control and flexibility for virtual networks.

4. **Compatibility with any hypervisor platform:**
   - Network virtualization platforms must also be able to work with the full range of server hypervisors, including Xen, XenServer, KVM, ESX, and HyperV, providing the ability to control virtualized network connectivity across any network substrate as well as between hypervisor environments.
   - This “any-to-any” paradigm shift provides for:
     i. More effective utilization of existing network investments,
     ii. Cost and management reduction of new, Layer 3 fabric innovations,
     iii. Workload portability from enterprise to cloud service provider environments

5. **Secure isolation among virtual networks, the physical networks, and the control plane:**
    - The promise of multi-tenancy requires maximum utilization of compute, storage and network assets through sharing of the physical infrastructure.
    - It is important that a network virtualization platform maintain this consolidation while still providing the isolation needed by regulatory compliance standards

6. **Cloud performance and scalable:**
    - Cloud drives a significant increase in the scale of tenants, servers, and applications supported in a single data center.
    - However, current networks are still bound by the physical limitations of networks, especially VLANs.
    - Network virtualization must support considerably larger scale deployments with tens thousands, or even hundreds of thousands of virtual networks.
    - This not only enables a larger number of tenants, but also support critical services like disaster recovery, data center utilization, etc., which outstrip current limitations.

- Virtual network solution should also not introduce any chokepoints or single points of failure into the network.
- This roughly entails that to all components for the solution must be fully distributed, and all network paths should support multi-pathing and failover.

7. **Programmatic network provisioning and control:**

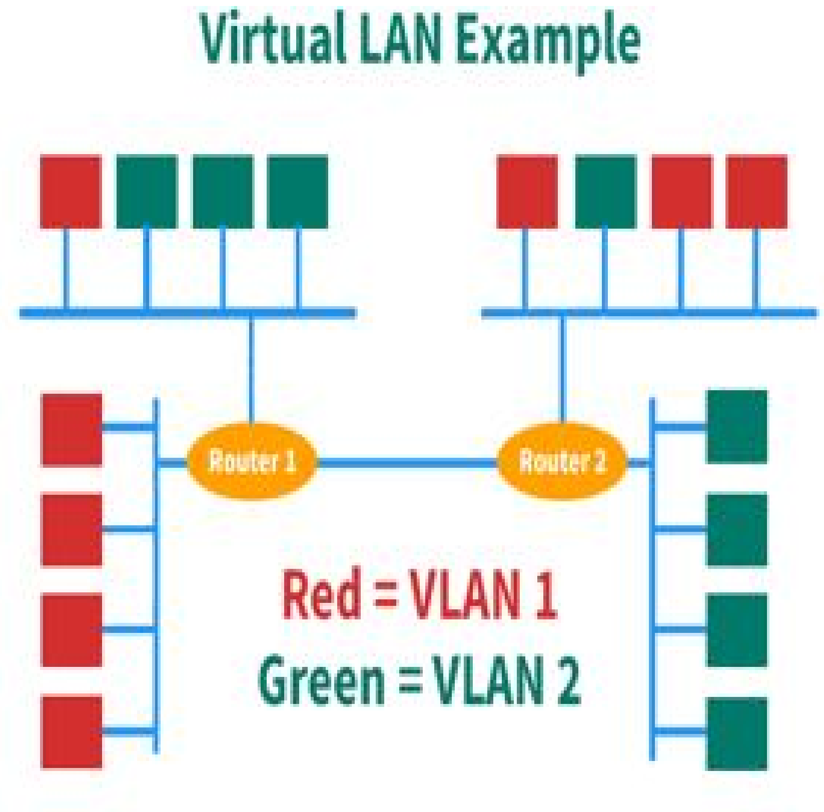
- Manual configuration make network configuration slow, error prone and open to security holes through a mistaken keystroke.
- In a large-scale cloud environment, this introduces a level of fragility and manual configuration costs that hurt service velocity and/or profitability.
- Network virtualization solution should provide full control over all virtual network resources and allow for these resources to be managed programmatically.
- The programmatic API should provide full access to management and configuration of a virtual network to not only support dynamic provisioning at cloud time scales, but also the ability to introduce and configure services on the fly.

Network Virtualization

- Mostly defined by decoupling the roles of Internet Service Provider (ISPs)
  1. Infrastructure Provider (InPs) : Manages the physical infrastructure
  2. Service Provider (SPs) : Creates virtual network by aggregating resources from multiple InPs and offer end-to-end services
- Concept of coexisting multiple logical networks can be categorized into:
  1. Virtual LAN (VLAN)
  2. Virtual Private Networks
  3. Active and programmable networks
  4. Overlay Networks
- Network Virtualization permits distributed participants to create almost instantly their own network with application specific naming, routing, and resource management mechanism
  - E.g. server virtualization enables users to use even a whole computing center arbitrarily as their own personal computer

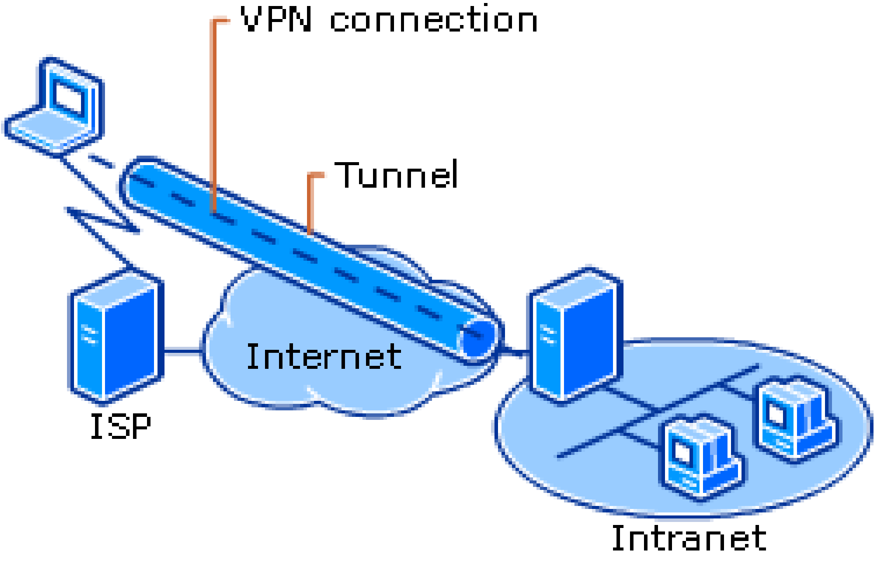
1. Virtual Local Area Network:

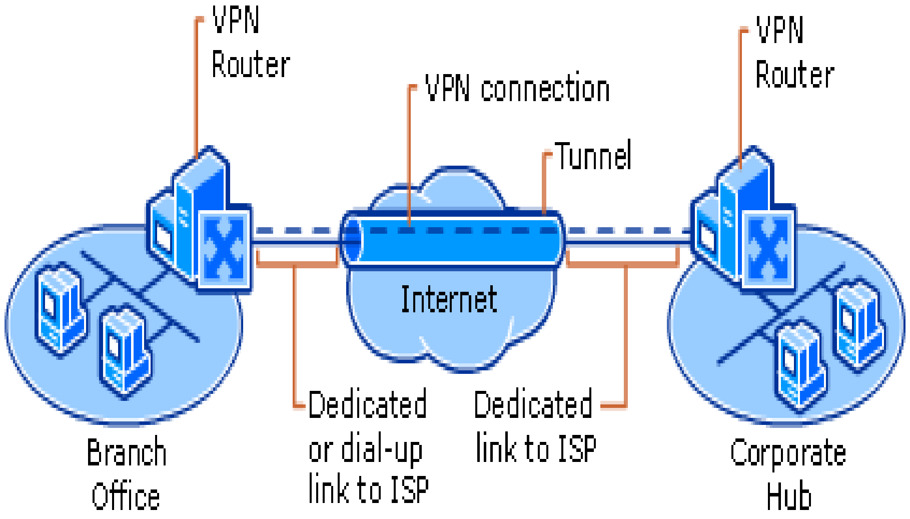
- It is a custom network created from one or more existing LANs.
- It enables groups of devices from multiple networks (both wired & wireless) to be combined into a single logical network.
- Routers & switches used must be VLAN configured.
- VN created using VLAN can be customized by network administrator using a software admin tool which configures the hardware.
- The admin software can be used to assign individual ports or groups of ports on a switch to a specific VLAN. For example, ports 1-12 on switch #1 and ports 13-24 on switch #2 could be assigned to the same VLAN.



2. Virtual Private Network (VPN):

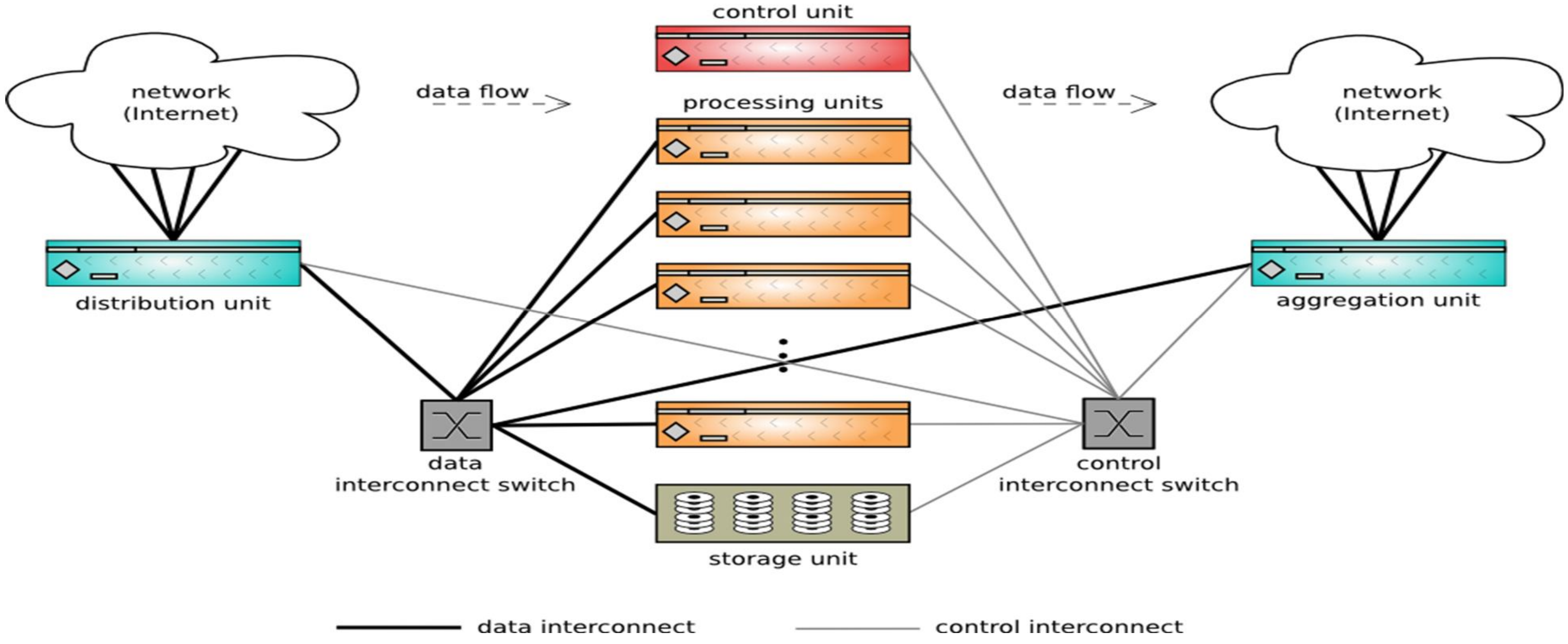
- It is a dedicated network connecting multiple sites using private & secure tunnels.
- VPN connects geographically distributed sites of a single corporate enterprise.
- Each VPN site contains one or more Customer Edge (CE) devices that are attached to one or more Provider Edge (PE) routers.
- To ensure security, data would travel through secure tunnels and VPN users would use authentication methods – including passwords, tokens and other unique identification methods – to gain access to the VPN
- VPNs help enable users working at home, on the road, or at a branch office to connect in a secure fashion to a remote corporate server using the Internet





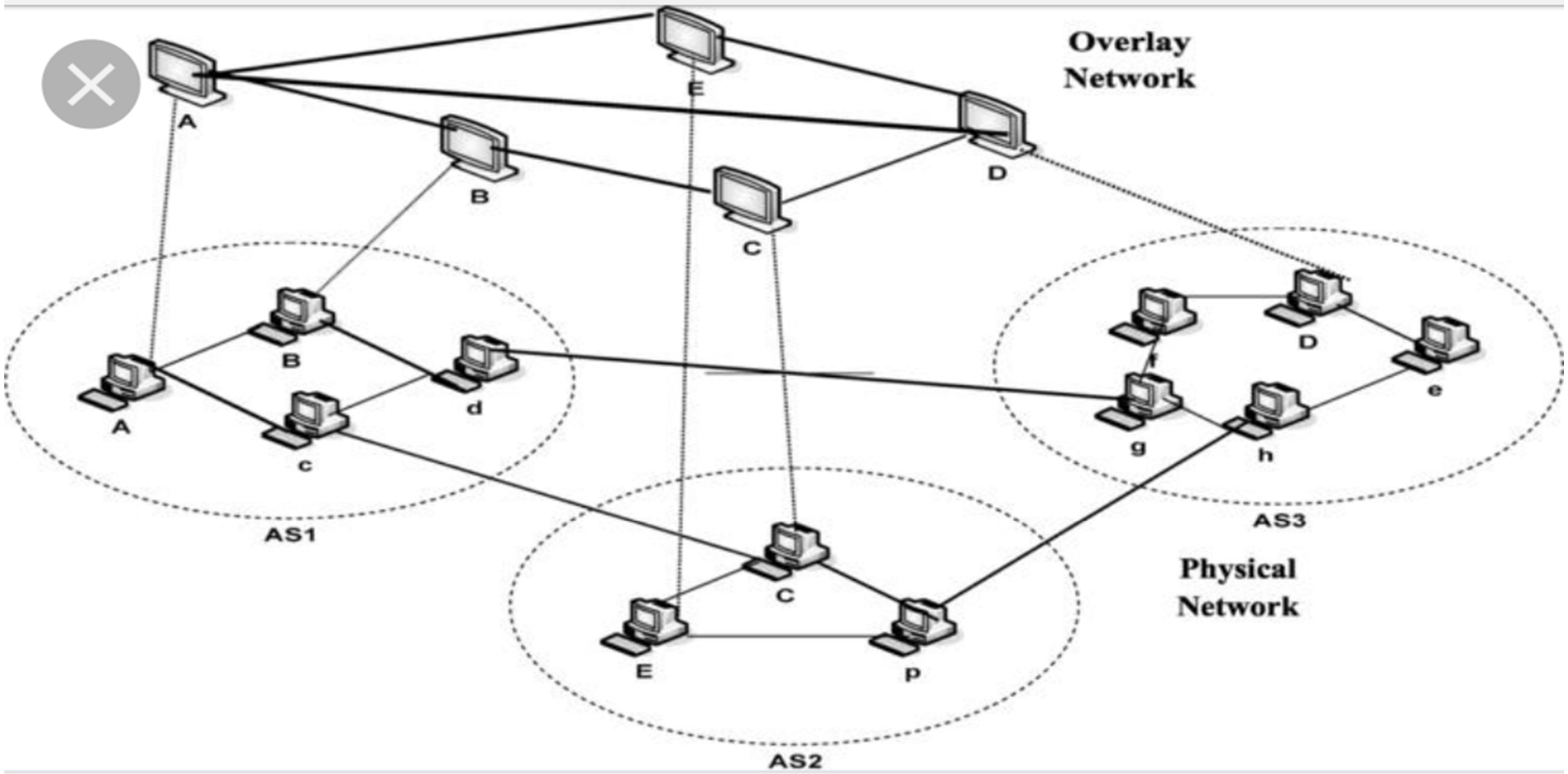
3. Active & Programmable Networks:

- Motivated by the need to create, deploy, and manage novel services on the fly.
- In addition, they also promote concept of isolated environment to allow multiple parties to run possibly conflicting codes on the same network elements without causing network instability.
- Two approaches to implement active network:
  1. Open Signaling Approach
  2. The Active Network Approach

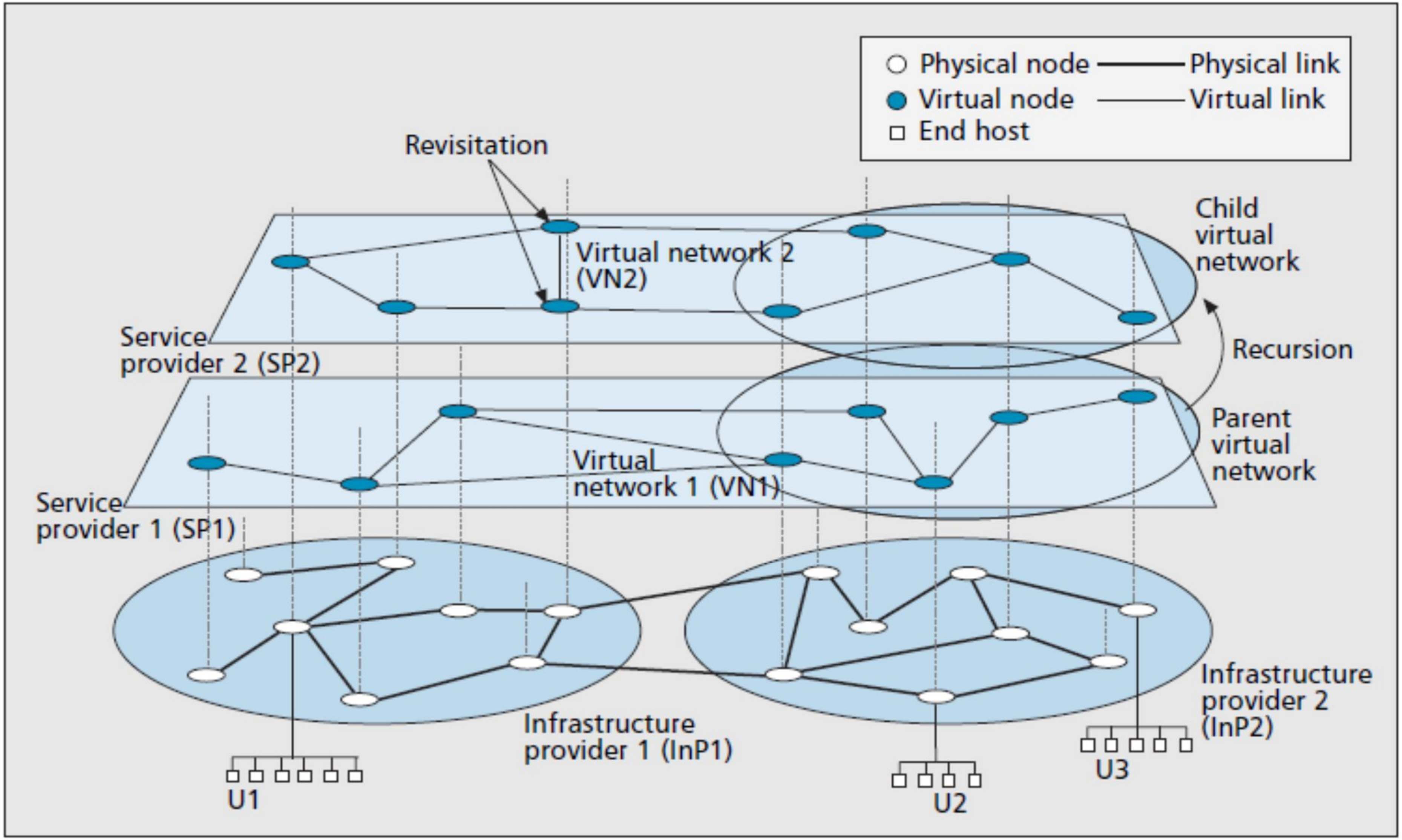


4. Overlay Networks:

- Logical network built on top of one or more existing physical network.
- Internet itself is an overlay network on top of telecommunication network.
- It don’t require or cause any changes to the existing network.



- A virtualized networking environment is collection of multiple heterogeneous network architectures from different service providers (SPs).
- Then each SP leases resources from one/more InPs to create virtual network
- **Infrastructure Provider :**
    - InPs deploy and actually manage the physical resources.
    - They offer their resources through programmable interfaces to different SPs.
    - InPs distinguish themselves through the quality of resources they provide, freedom they delegate to their customers, and the tools they provide to exploit that freedom.


○ Physical node
● Virtual node
□ End host
Physical link
Virtual link
Revisitation
Virtual network 2 (VN2)
Child virtual network
Recursion
Parent virtual network
Service provider 2 (SP2)
Virtual network 1 (VN1)
Service provider 1 (SP1)
Infrastructure provider 1 (InP1)
Infrastructure provider 2 (InP2)
U1
U2
U3

- **Service Provider :**
  - Lease resources from multiple InPs to create and deploy VN by programming allocated network resources to offer end-to-end to end users.
  - SP can also provide network services to other SPs.
  - It can also create child VNs by partitioning its resources and act as a virtual InP.
- **End User :**
  - NV end users are similar to those of existing Internet, except that the existence of multiple VNs from competing SPs provide them a wider range of choice.
  - Any end user can connect to multiple VNs from different service providers for different services.

Architectural Principles of VN

- **Coexistence :**
  - It refers to the fact that multiple VNs from different SPs can coexist together , spanning over part or full of the underlying physical networks provided by one/more InPs.
  - VN1 & VN2 coexist together.
- **Recursion :**
  - When one/more VNs creating a VN hierarchy with parent-child relationships, it is known as recursion as well as nesting of VNs.
- **Inheritance :**
  - Child VNs in an VNE (virtual network environment) can inherit architectural attributes from their parents.
  - Constraints on the parent VN automatically translate to similar constraints on its children

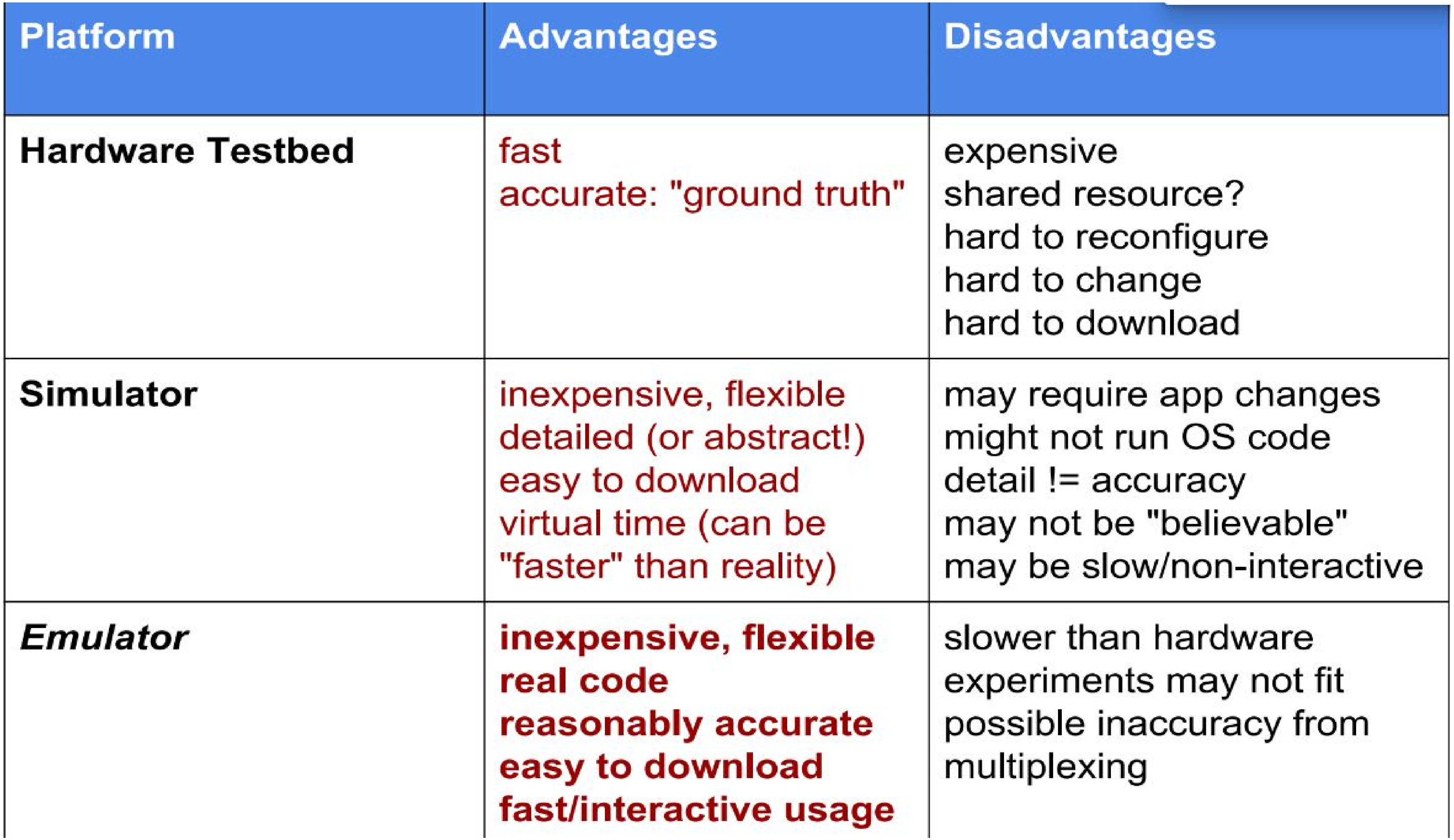
  - Inheritance allows an SP to add value to the spawned child VNs before reselling them to other SPs.
- **Revisitation :**
  - Allows physical node to host multiple virtual nodes of a single VN.
  - Use of multiple logical routers to handle diverse functionalities in a large complex network allows an SP to logically rearrange its network structure and to simplify the management of VN.

Design Goals of VN

1. **Flexibility :**
   - VN must provide freedom in every aspect of networking.
   - Each SP should be free to implement arbitrary network topology, routing and forwarding functionalities, customized control protocols independent of underlying physical network.
2. **Manageability :**
   - By separating SPs from InPs, network virtualization will modularize network management tasks.
   - It must provide complete end-to-end control of the VN to the SPs.
3. **Scalability :**
   - It is indispensable part of coexistence property of VN.
   - InPs in an virtual network must scale to support an increasing number of coexisting VNs without affecting their performance.

4. **Isolation :**
    - Network virtualization must insure isolation between coexisting VNs to improve fault tolerance, security, and privacy.
    - Network protocols are prone to misconfiguration and implementation errors.
    - Misconfiguration in one VN don’t affect other coexisting VNs.
5. **Stability & Convergence :**
    - Network errors and misconfigurations in underlying physical network can destabilize the VN.
    - Instability in the InPs can lead to instability of all hosted VNs.
    - Virtualization must ensure the stability and in case of any instability the affected VNs must be able to successfully converge to their stable states.

6. **Programmability :**
   - It is required to ensure flexibility and manageability.
   - Only thorough programmability SPs can implement customized protocols and deploy diverse services.
7. **Heterogeneity :**
   - It can be of two types: i. heterogeneity of underlying networking technologies (e.g optical, wireless, sensor)
   - ii. Each end-to-end VN, created on top of that heterogeneous combination of underlying network, can also be heterogeneous.



Platform	Advantages	Disadvantages
Hardware Testbed	fast accurate: "ground truth"	expensive shared resource? hard to reconfigure hard to change hard to download
Simulator	inexpensive, flexible detailed (or abstract!) easy to download virtual time (can be "faster" than reality)	may require app changes might not run OS code detail != accuracy may not be "believable" may be slow/non-interactive
Emulator	**inexpensive, flexible** **real code** **reasonably accurate** **easy to download** **fast/interactive usage**	slower than hardware experiments may not fit possible inaccuracy from multiplexing

MININET

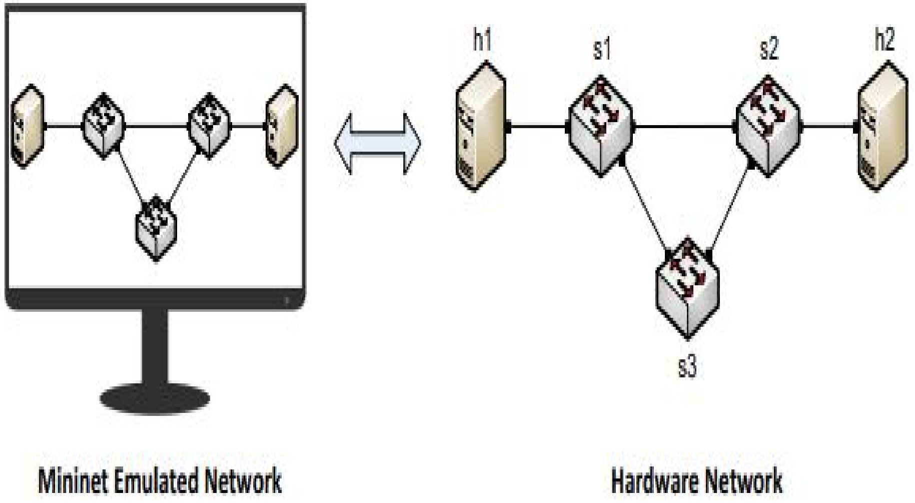
- It is a *network emulator* which creates a network of virtual hosts, switches, controllers, and links.
- Mininet hosts run standard Linux network software, and its switches support OpenFlow for highly flexible custom routing and Software-Defined Networking.
- Supports research, development, learning, prototyping, testing, debugging, and any other tasks that could benefit from having a complete experimental network on a laptop or other PC.
- Mininet networks run *real code* including standard Unix/Linux network applications as well as the real Linux kernel and network stack

- **Advantages of Using Mininet:**

1. Provides a simple and inexpensive **network testbed** for developing OpenFlow applications.
2. Enables **multiple concurrent developers** to work independently on the same topology
3. Supports **system-level regression tests**, which are repeatable and easily packaged
4. Enables **complex topology testing**, without the need to wire up a physical network
5. Includes a **CLI** that is topology-aware and OpenFlow-aware, for debugging or running network-wide tests
6. Supports **arbitrary custom topologies**, and includes a basic set of **parametrized topologies**
7. is **usable out of the box** without programming, but
8. also Provides a straightforward and extensible **Python API** for network creation and experimentation

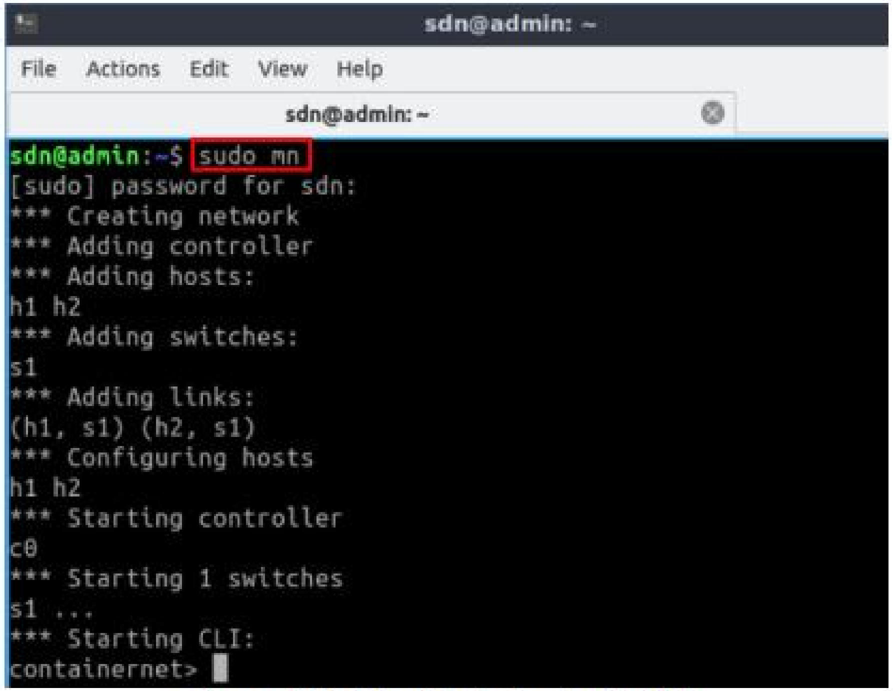
- **Limitations :**

1. Mininet-based networks cannot (currently) exceed the CPU or bandwidth available on a single server.
2. Mininet cannot (currently) run non-Linux-compatible OpenFlow switches or applications; this has not been a major issue in practice.



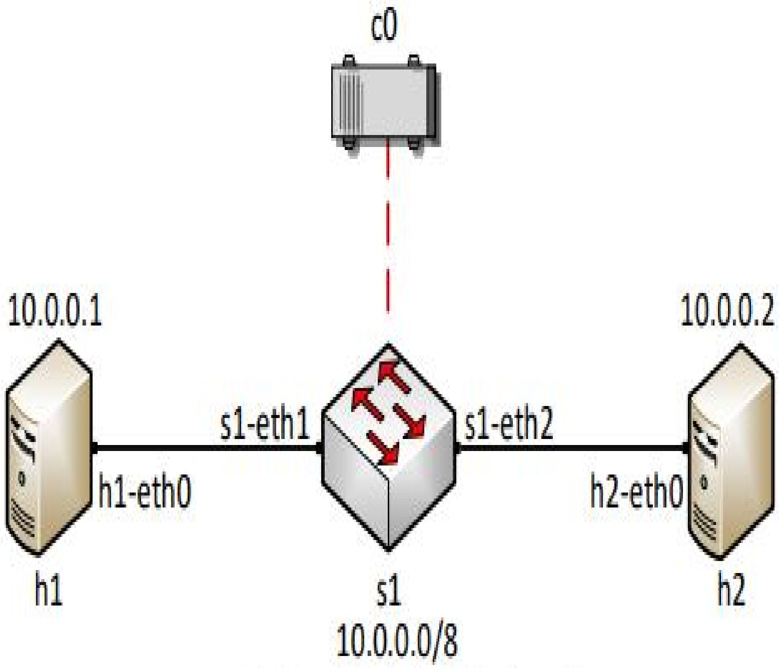
MININET Topologies

- Default Topology:



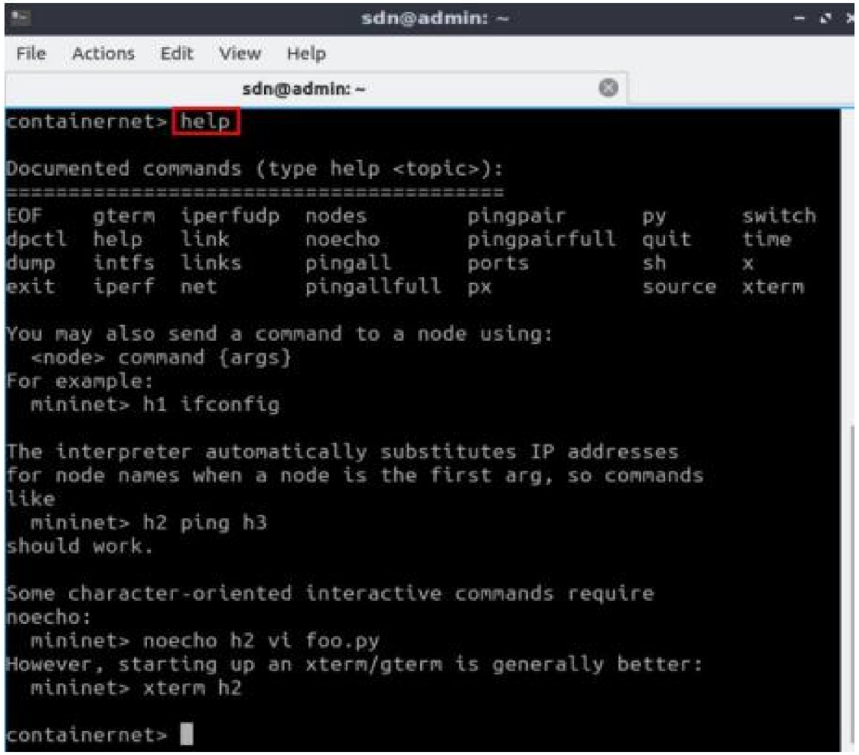
```
sdn@admin:~$ sudo mn
[sudo] password for sdn:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
containernet> █
```

Figure 4. Starting Mininet using the CLI.

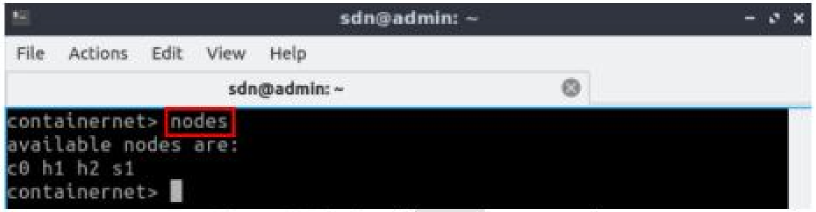


MININET Topologies

- Commands:

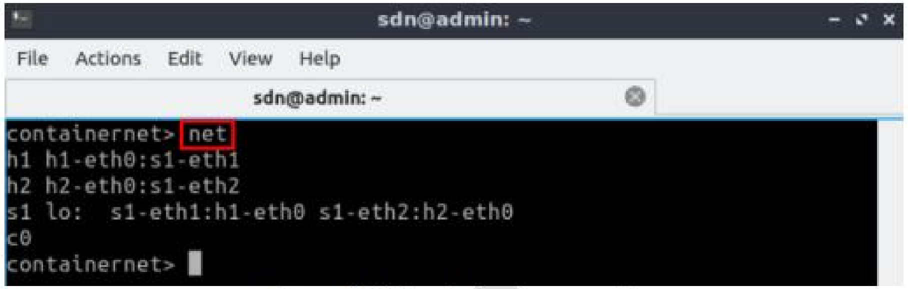


```
containernet> help

Documented commands (type help <topic>):
========================================
EOF    gterm  iperfudp  nodes        pingpair      py      switch
dpctl  help   link      noecho       pingpairfull  quit    time
dump   intfs  links     pingall      ports         sh      x
exit   iperf  net       pingallfull  px            source  xterm

You may also send a command to a node using:
  <node> command {args}
For example:
  mininet> h1 ifconfig

The interpreter automatically substitutes IP addresses
for node names when a node is the first arg, so commands
like
  mininet> h2 ping h3
should work.

Some character-oriented interactive commands require
noecho:
  mininet> noecho h2 vi foo.py
However, starting up an xterm/gterm is generally better:
  mininet> xterm h2

containernet> █
```



```
containernet> nodes
available nodes are:
c0 h1 h2 s1
containernet> █
```

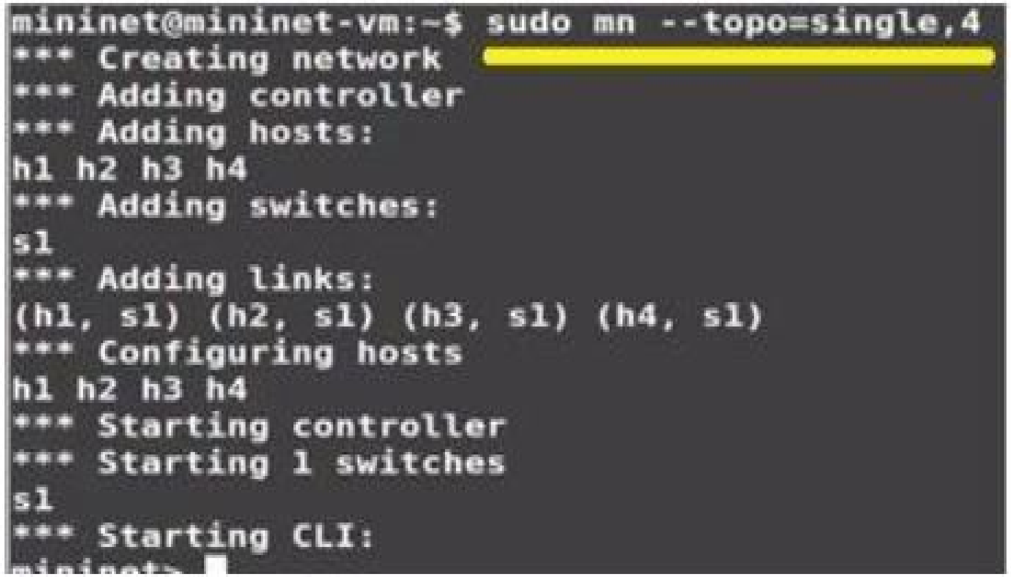
Figure 7. Mininet's nodes command.



```
containernet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
containernet> █
```

MININET Topologies

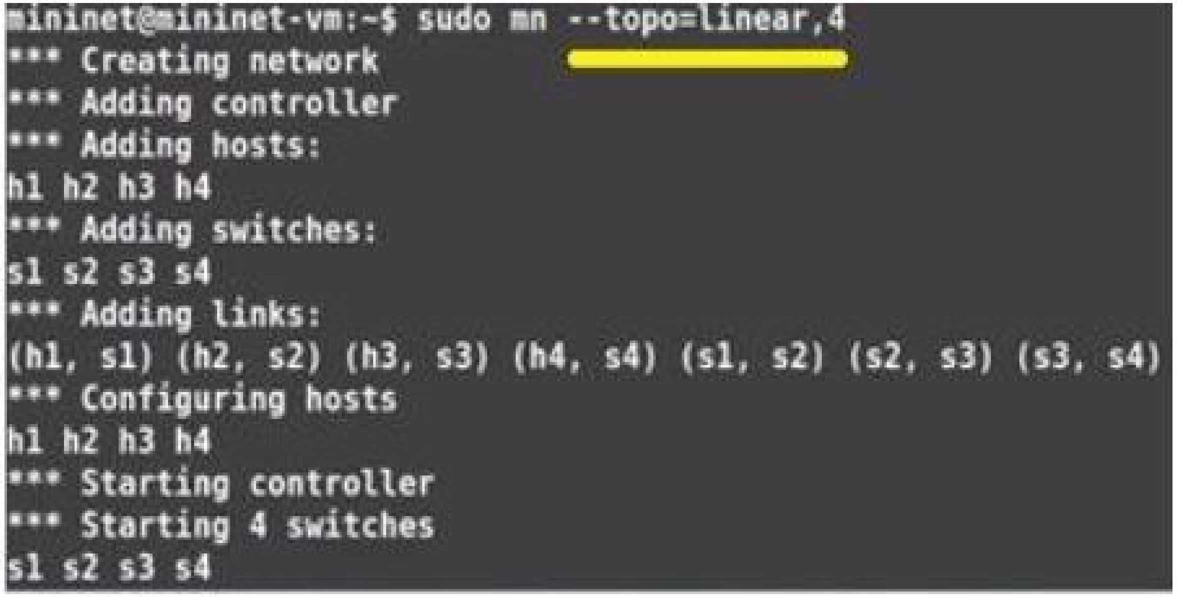
- Single Topology:



```
mininet@mininet-vm:~$ sudo mn --topo=single,4
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
*** Starting 1 switches
s1
*** Starting CLI:
mininet>
```

MININET Topologies

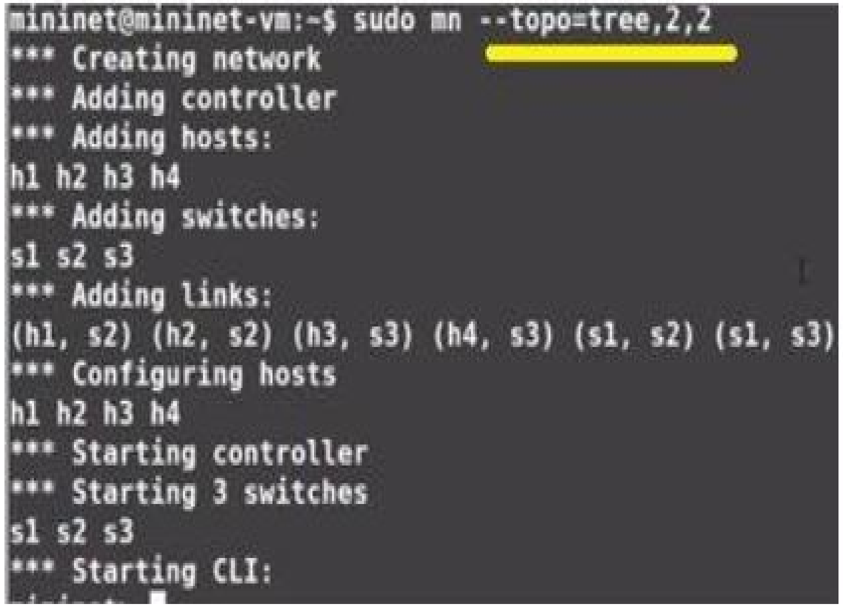
- Linear Topology:



```
mininet@mininet-vm:~$ sudo mn --topo=linear,4
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (s1, s2) (s2, s3) (s3, s4)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
*** Starting 4 switches
s1 s2 s3 s4
```

MININET Topologies

- Tree Topology:



```
mininet@mininet-vm:~$ sudo mn --topo=tree,2,2
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s2) (h2, s2) (h3, s3) (h4, s3) (s1, s2) (s1, s3)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
*** Starting 3 switches
s1 s2 s3
*** Starting CLI:
```

Working with MININET

- Mininet enables you to quickly create, interact with, customize and share a software defined network prototype, and provides a smooth path to running on hardware.
- **Creating a Network:**

  sudo mn --switch ovs --controller ref --topo tree,depth=2,fanout=8 --test pingall

- Starts a network with a tree topology of depth 2 and fanout 8 (i.e. 64 hosts connected to 9 switches), using Open vSwitch switches under the control of the OpenFlow/Stanford reference controller, and runs the pingall test to check connectivity between every pair of nodes.

- **Interacting with a Network**

  ```
  mininet> h2 ping h3
  ```
- tells host h2 to ping host h3’s IP address.
- *Any available Linux command or program can be run on any virtual host*. You can easily start a web server on one host and make an HTTP request from another

-
  ```
  mininet> h2 python -m SimpleHTTPServer 80 >& /tmp/http.log
  ```

-
  ```
  mininet> h3 wget -O - h2
  ```

- **Customizing a Network :**

Mininet’s API allows you to create custom networks with a few lines of Python. For example, the following script, creates a small network (4 hosts, 3 switches), and pings one host from another

```
from mininet.net import Mininet
from mininet.topolib import TreeTopo
tree4 = TreeTopo(depth=2,fanout=2)
net = Mininet(topo=tree4)
net.start()
h1, h4 = net.hosts[0], net.hosts[3]
print h1.cmd('ping -c1 %s' % h4.IP())
net.stop()
```