



# PMCA506L: Cloud Computing

# Module 3 : Virtual Machines

Courtesy : *Ming Lian , Dogules E Comer & Other Sources of Internet*

# Virtualization

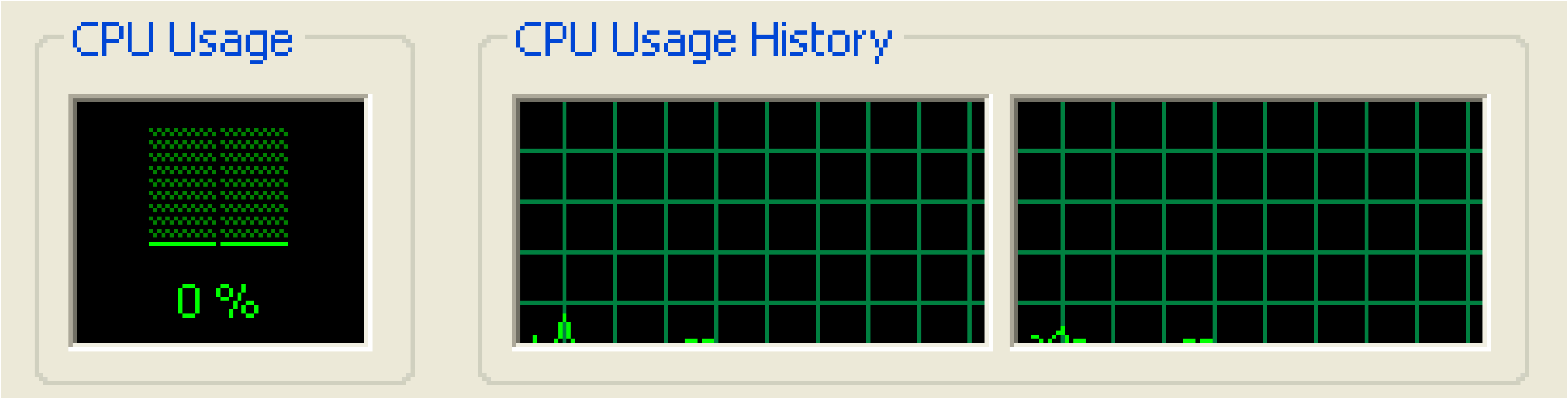
- Virtualization is a broad term that refers to the abstraction of resources across many aspects of computing
- For our purposes - One physical machine to support multiple virtual machines that run in parallel.

# Problem Assessment – Why Virtualization?

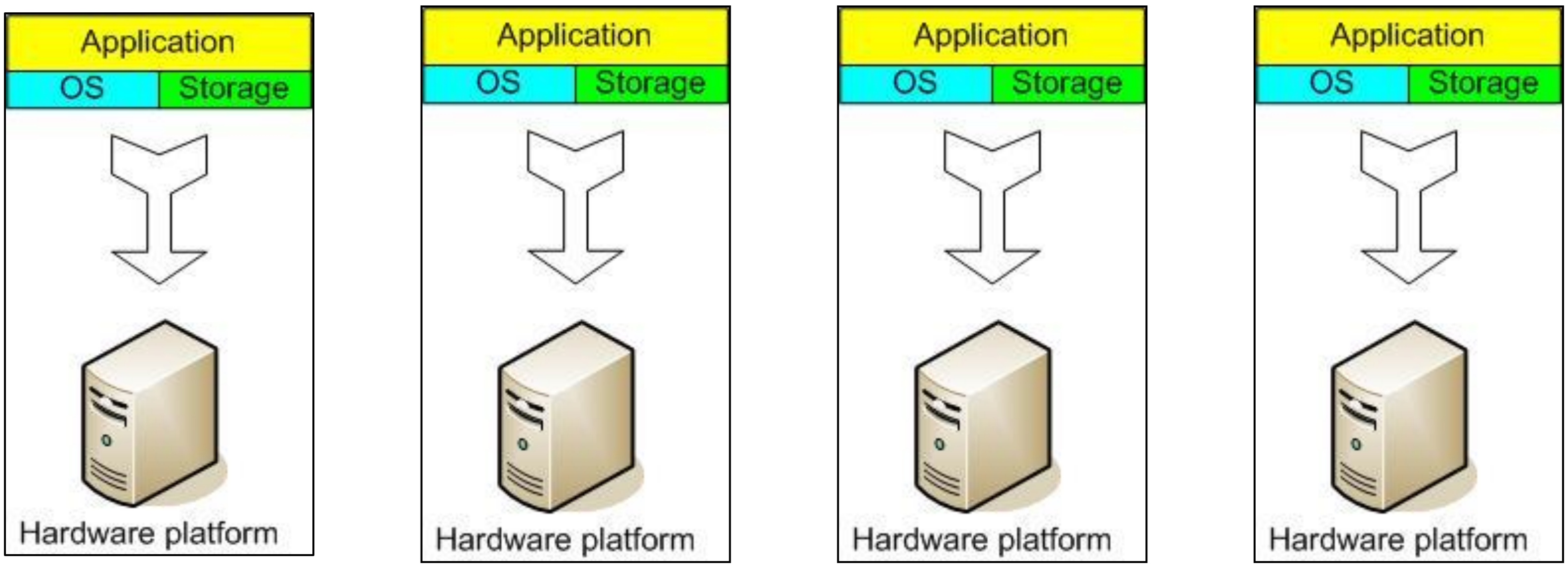
- Too many servers for too little work
- Aging hardware reaching end of usable life
- High infrastructure requirements
- Limited flexibility in shared environments

# Problem Assessment

Low utilization metrics in servers across the organization…

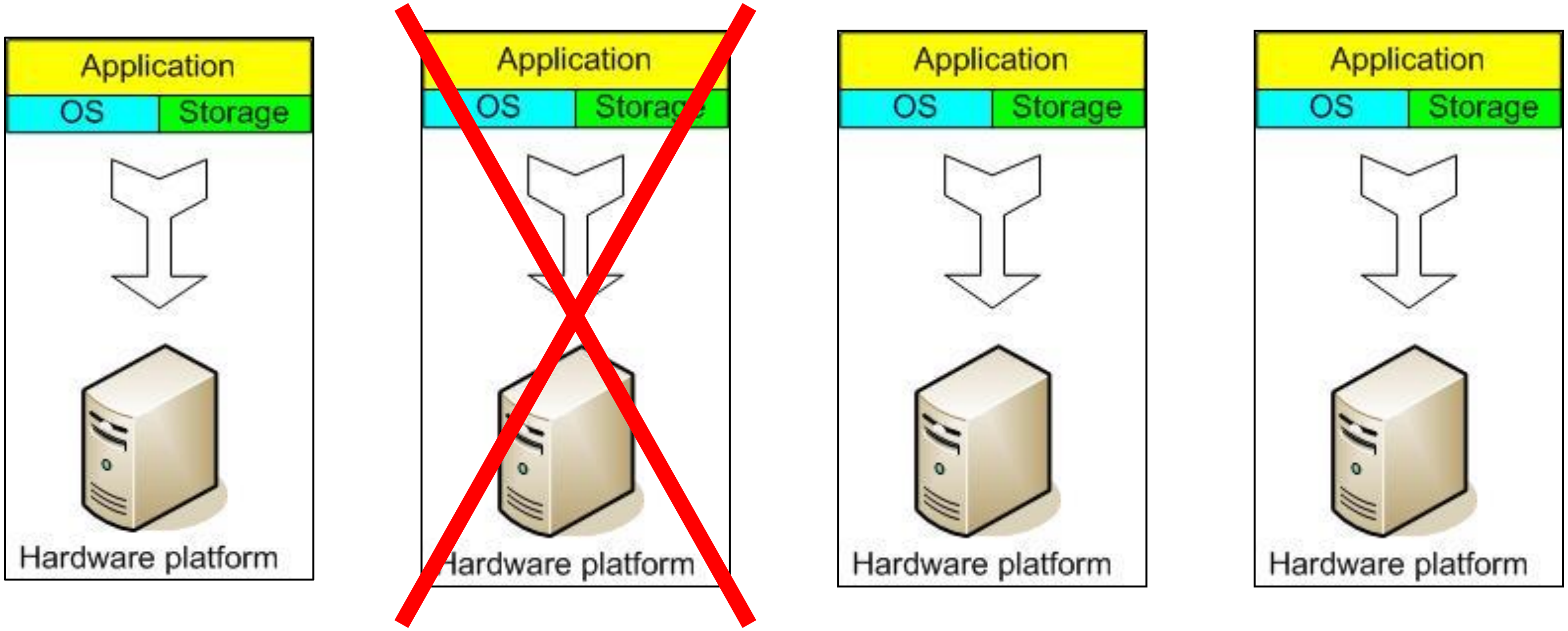


# The Traditional Server Concept




| | | | |
|---|---|---|---|
| **Web Server** | **App Server** | **DB Server** | **EMail** |
| **Windows** | **Linux** | **Linux** | **Windows** |
| **IIS** | **Glassfish** | **MySQL** | **Exchange** |

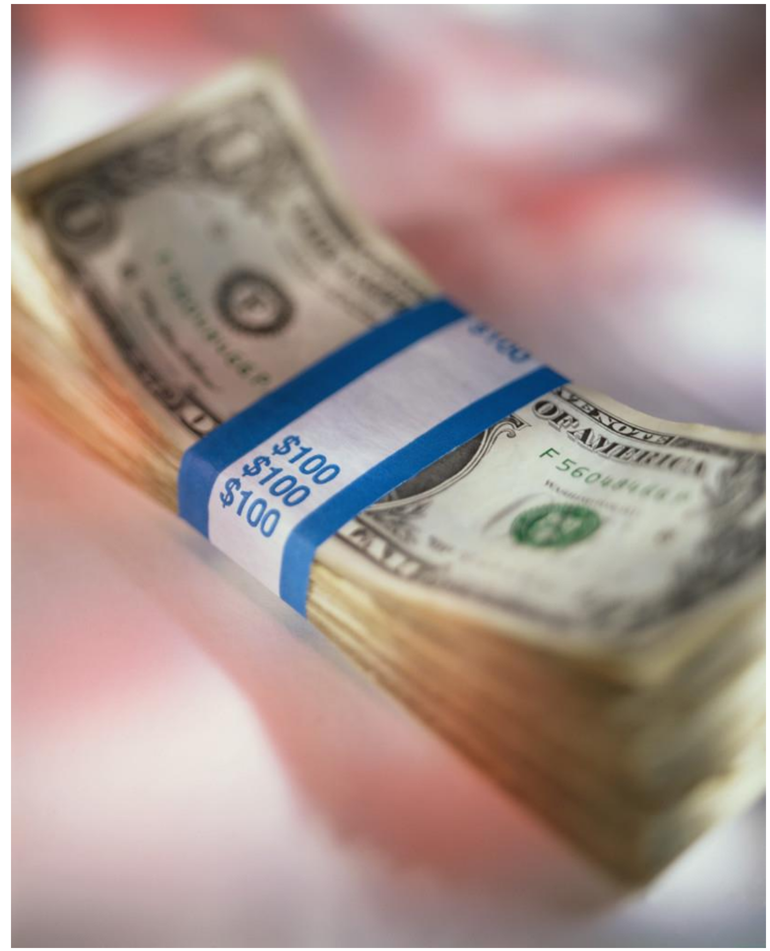
# And if something goes wrong …




| Web Server | App Server | DB Server | EMail |
|---|---|---|---|
| Windows | DOWN! | Linux | Windows |
| IIS | | MySQL | Exchange |

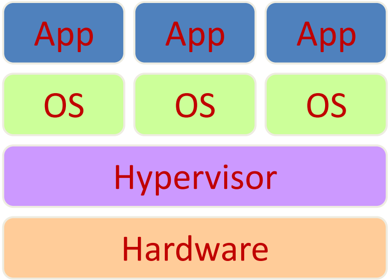
# Problem Assessment

- High costs and infrastructure needs
  - Maintenance
  - Leases
  - Networking
  - Floor space
  - Cooling
  - Power
  - Disaster Recovery



# Virtualization

- Virtual workspaces:
  – An abstraction of an execution environment that can be made dynamically available to authorized clients by using well-defined protocols,
  – Resource quota (e.g. CPU, memory share),
  – Software configuration (e.g. O/S, provided services).
- Implement on Virtual Machines (VMs):
  – Abstraction of a physical host machine,
  – Hypervisor intercepts and emulates instructions from VMs, and allows management of VMs,
  – VMWare, Xen, etc.
- Provide infrastructure API:
  – Plug-ins to hardware/support structures



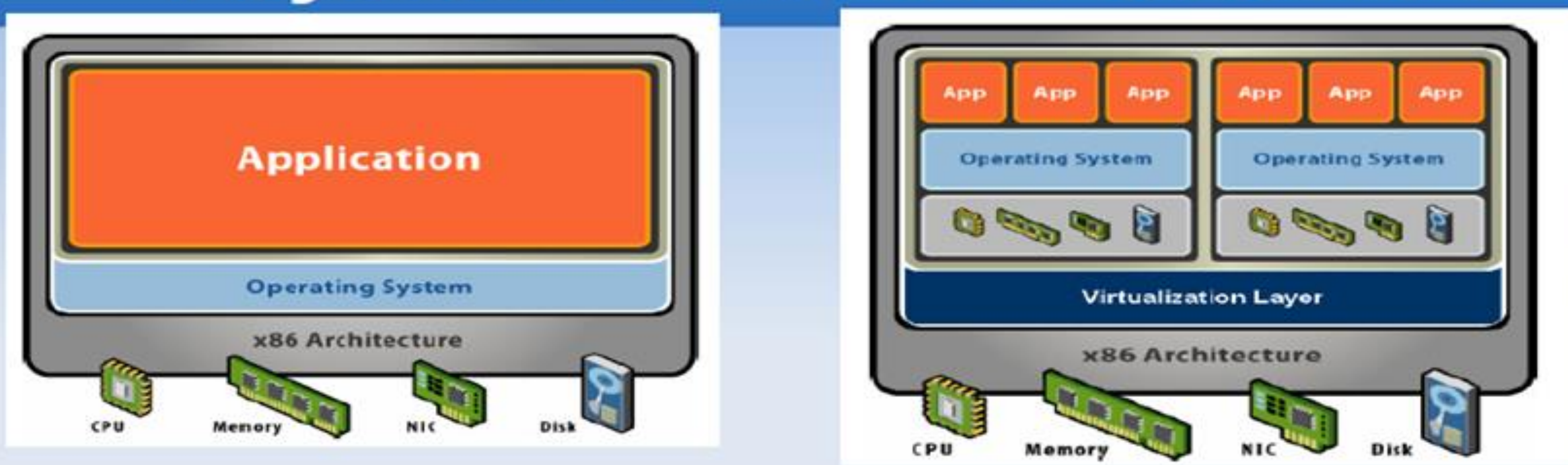

Virtualized Stack

# Core Technology

- The Hypervisor
  - A computing layer which allows multiple operating systems to run on a host computer at the same time.
  - Originally developed in the 1970s as part of the IBM S/360
  - Many modern day variants from different developers

# Comparison

## Physical vs. Virtual Machine




- Single OS
- h/w + s/w tightly coupled
- Application crashes affect all
- Resource under-utilization

- Machine view to OS is independent of hardware
- Multiple OS (isolated apps)
- Safely multiplex resources across VMs

# Uses of Virtualization

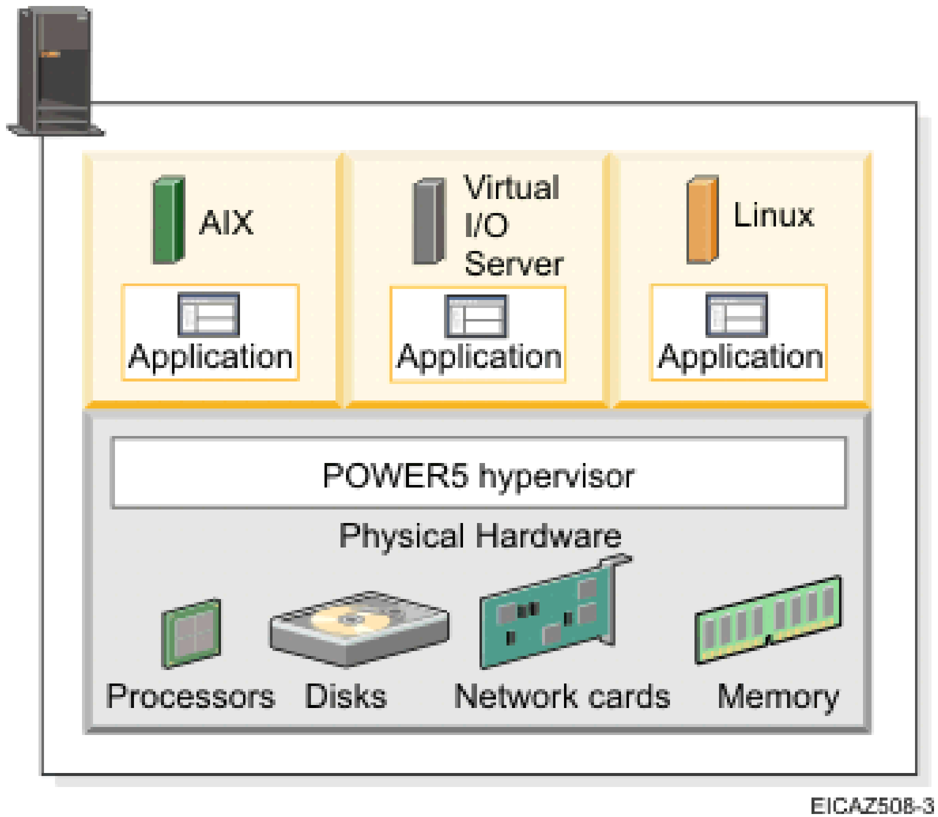
- Server consolidation
  - Run a **web server** and a **mail server** on the **same physical server**
- Easier development
  - Develop critical **operating system components** (file system, disk driver) without affecting **computer stability**
- QA
  - Testing a network product (e.g., a firewall) may require **tens of computers**
  - Try testing thoroughly a product at each pre-release milestone… and have a straight face when your boss shows you the **electricity bill**
- Cloud computing
  - The modern buzz-word
  - Amazon sells computing power
  - You pay for e.g., 2 CPU cores for 3 hours plus 10GB of network traffic

# Virtualization Scenarios

- Hardware Virtualization
- Software Virtualization
  - Full Virtualization
  - Para-Virtualization

# Hardware Virtualization (example)

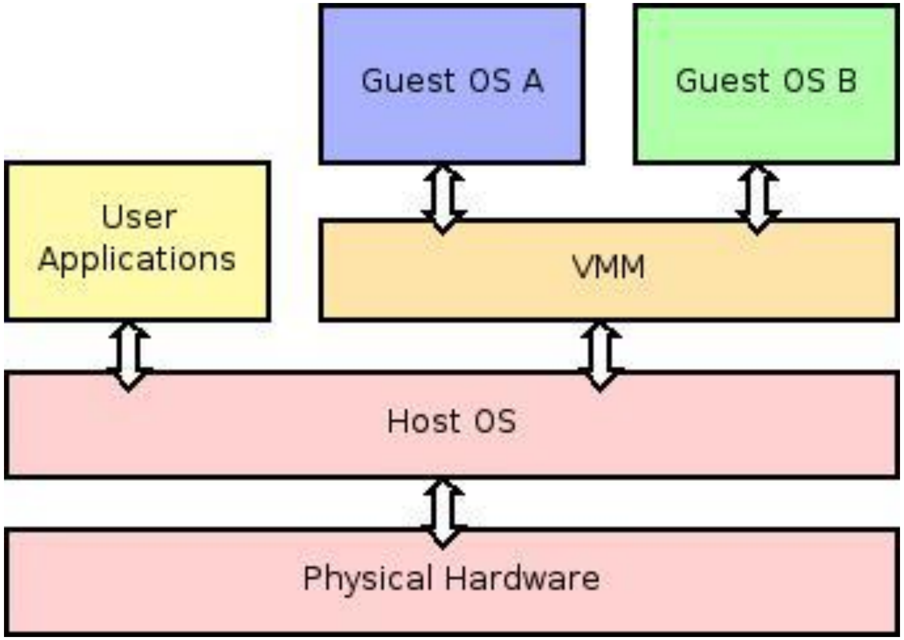
- IBM pSeries Servers




http://publib.boulder.ibm.com/infocenter/eserver/v1r2/topic/eicaz/eicaz508.gif

# Software Virtualization (example)

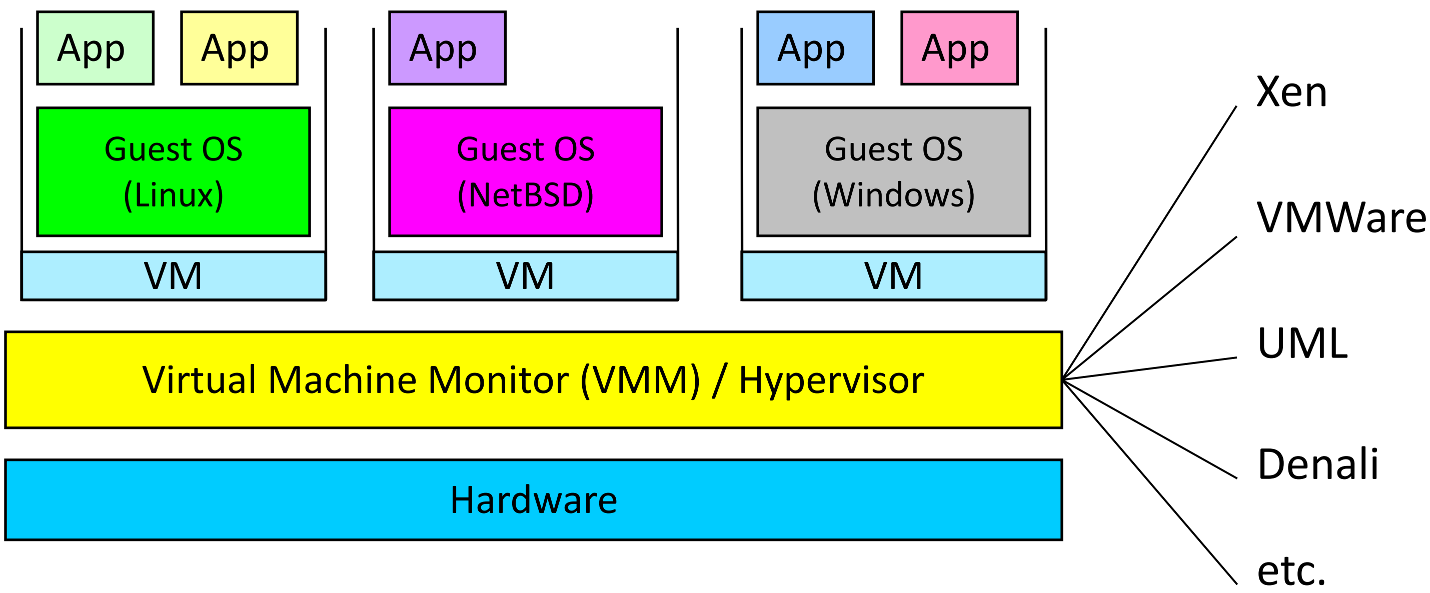
- VMware Server (GSX)



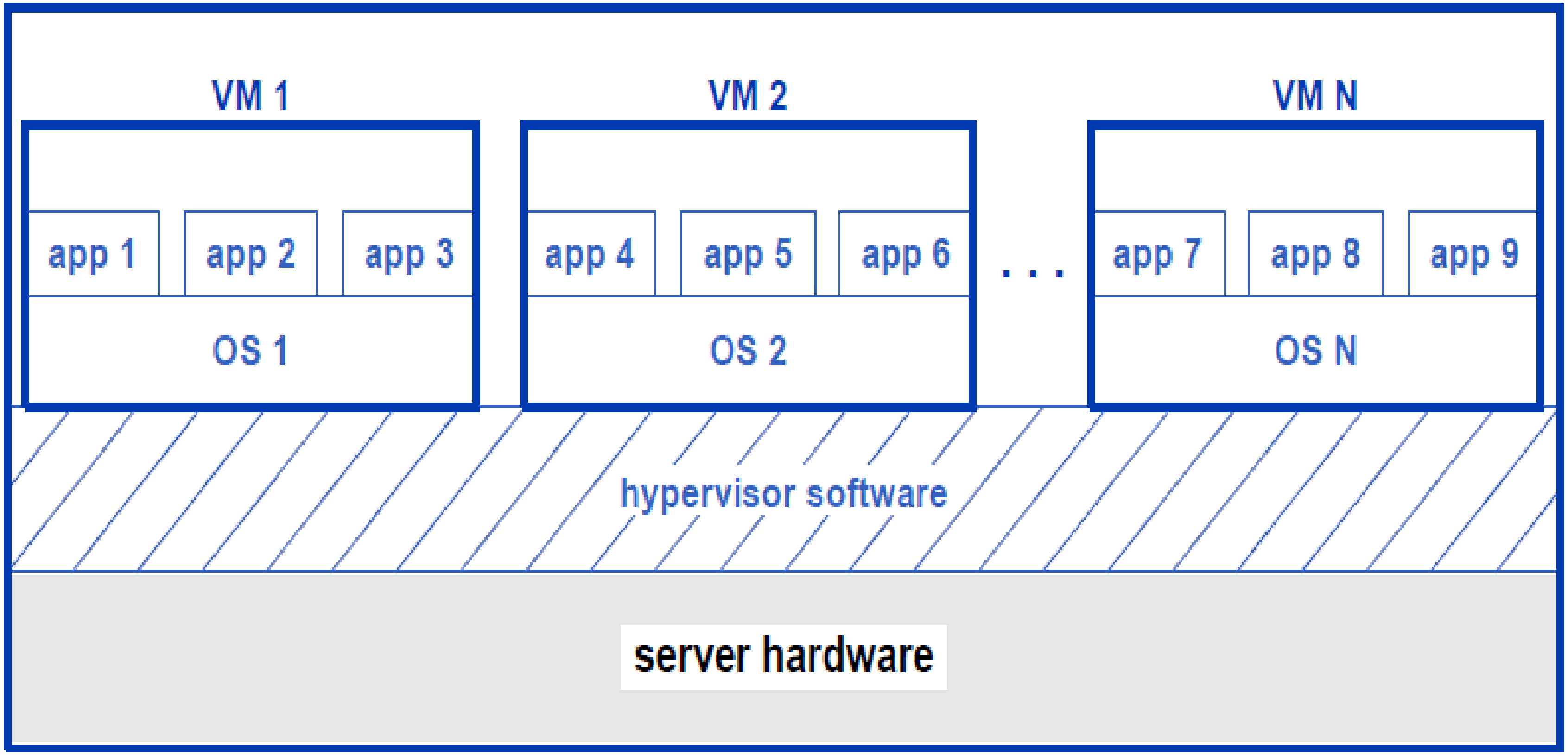

http://openlab-mu-internal.web.cern.ch/openlab-mu-internal/openlab-II_Projects/Platform_Competence_Centre/Virtualization/Virtualization.asp

# Virtual Machines

- VM technology allows multiple virtual machines to run on a single physical machine.



# Conceptual Organization Of VM Systems

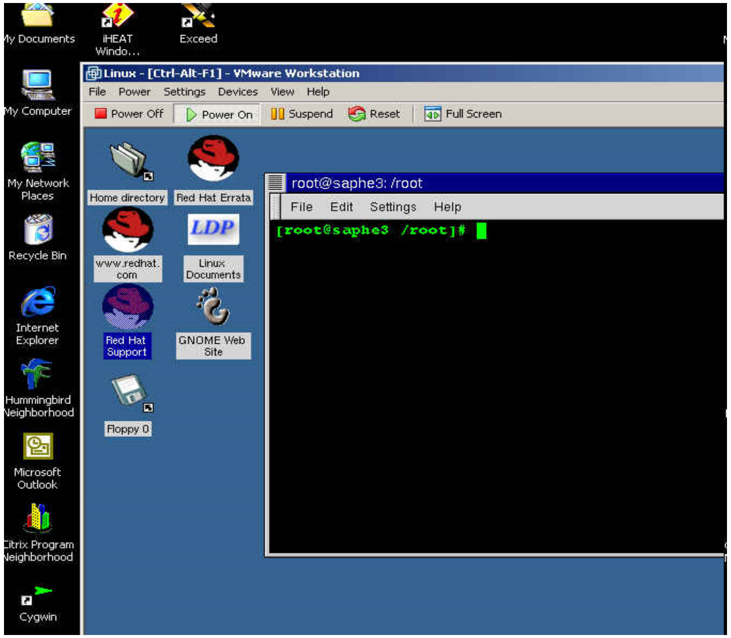


# Advantages of virtual machines

- Run operating systems where the physical hardware is unavailable,
- Easier to create new machines, backup machines, etc.,
- Software testing using “clean” installs of operating systems and software,
- Emulate more machines than are physically available,
- Timeshare lightly loaded systems on one host,
- Debug problems (suspend and resume the problem machine),
- Easy migration of virtual machines (shutdown needed or not).
- Run legacy systems!

# How Virtualization Different from Dual Boot ?

- Virtualization is way to run **multiple operating systems** and **user applications** on the same hardware
  - E.g., run both Windows and Linux on the same laptop
- How is it different from **dual-boot**?
  - Both OSes run **simultaneously**
- The OSes are completely **isolated** from each other

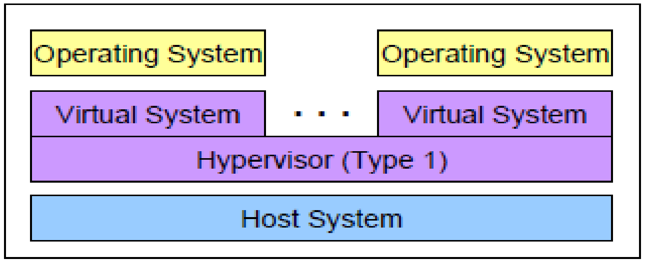



# HyperVisor (Virtual Machine Monitor)

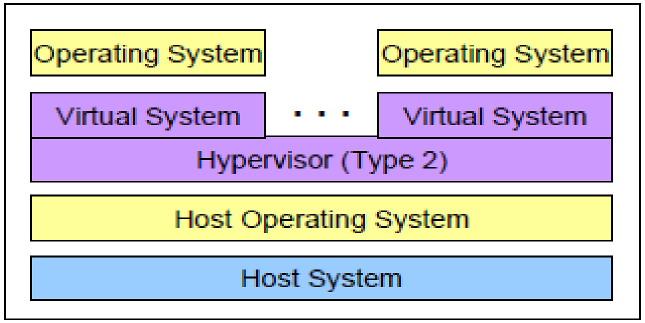
## Hypervisor Types

- **Type-1 hypervisor:**
  - Hypervisor runs directly on underlying host system
  - Alternative terms:
    - "Native hypervisor"
    - "Bare metal hypervisor"
- **Type-2 hypervisor:**
  - A host operating system runs on underlying host system
  - Hypervisor runs on / in host operating system:
    - as one user space process, or
    - as one user space process per virtual system
  - Various degrees of integration of hypervisor into host OS
  - Alternative term:
    - "Hosted hypervisor"
- **Original definition of type 1 & 2 hypervisors:**
  - Goldberg: "Architectural Principles for Virtual Computer Systems" (1973), link

**Type-1 Hypervisor :**




**Type-2 Hypervisor :**



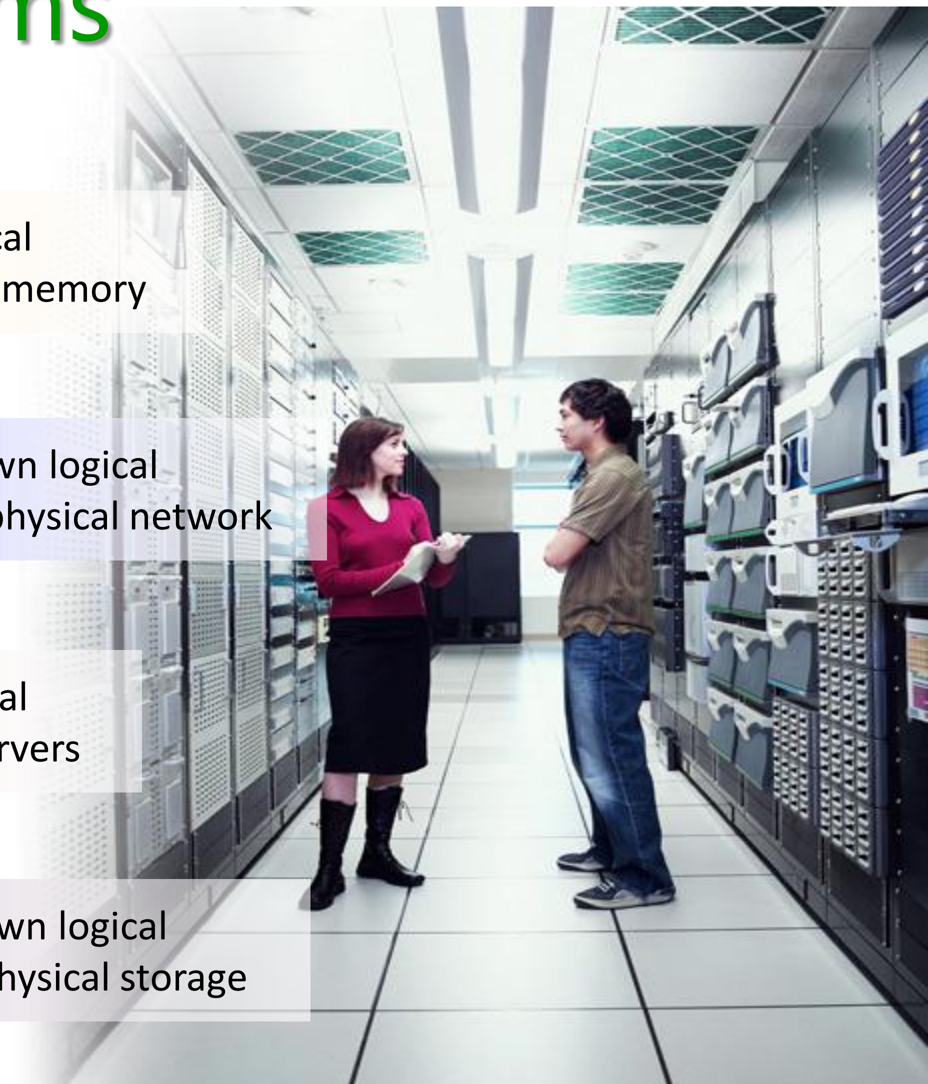
# Virtualization Comes in Many Forms

**Virtual Memory** — Each application sees its own logical **memory,** independent of physical memory

**Virtual Networks** — Each application sees its own logical **network,** independent of physical network

**Virtual Servers** — Each application sees its own logical **server,** independent of physical servers

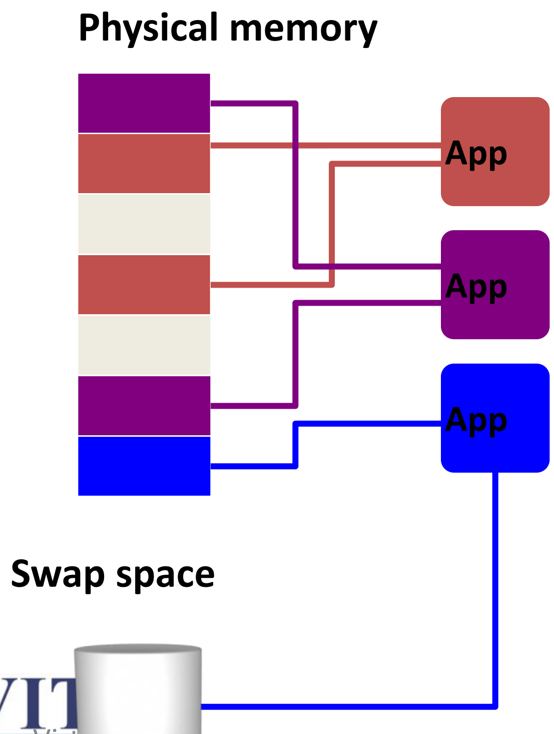
**Virtual Storage** — Each application sees its own logical **storage,** independent of physical storage



# Memory Virtualization

**Virtual Memory**

Each application sees its own logical **memory,** independent of physical memory



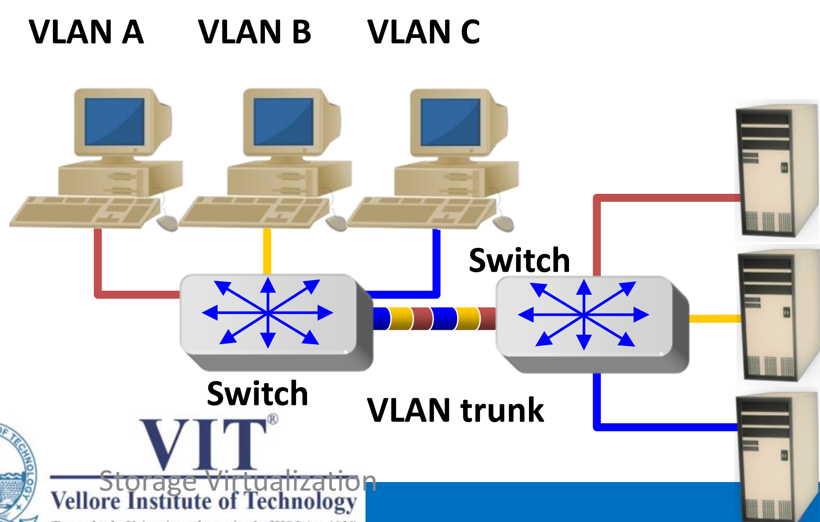

### Benefits of Virtual Memory

- Remove physical-memory limits
- Run multiple applications at once

# Network Virtualization

**Virtual Networks**

Each application sees its own logical **network,** independent of physical network

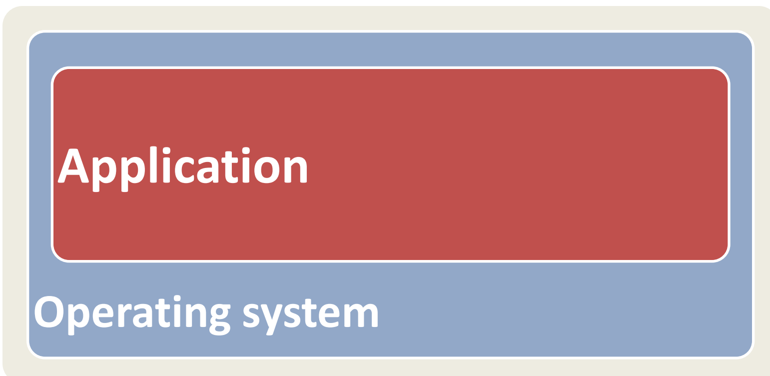



**Benefits of Virtual Networks**

- Common network links with access-control properties of separate links
- Manage logical networks instead of physical networks
- **Virtual SANs** provide similar benefits for storage-area networks

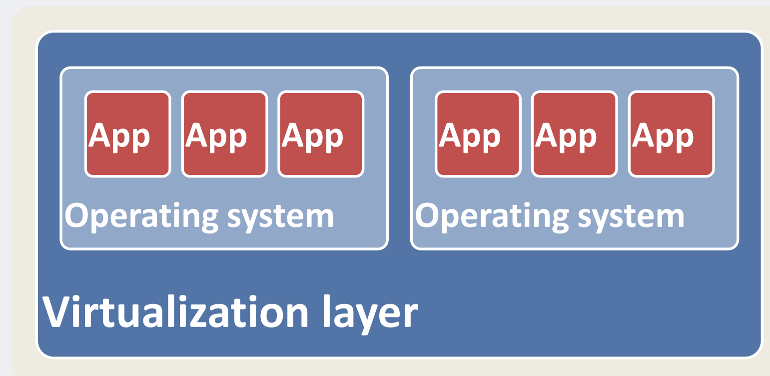
# Server Virtualization

## Before Server Virtualization:




- Single operating system image per machine
- Software and hardware tightly coupled
- Running multiple applications on same machine often creates conflict
- Underutilized resources

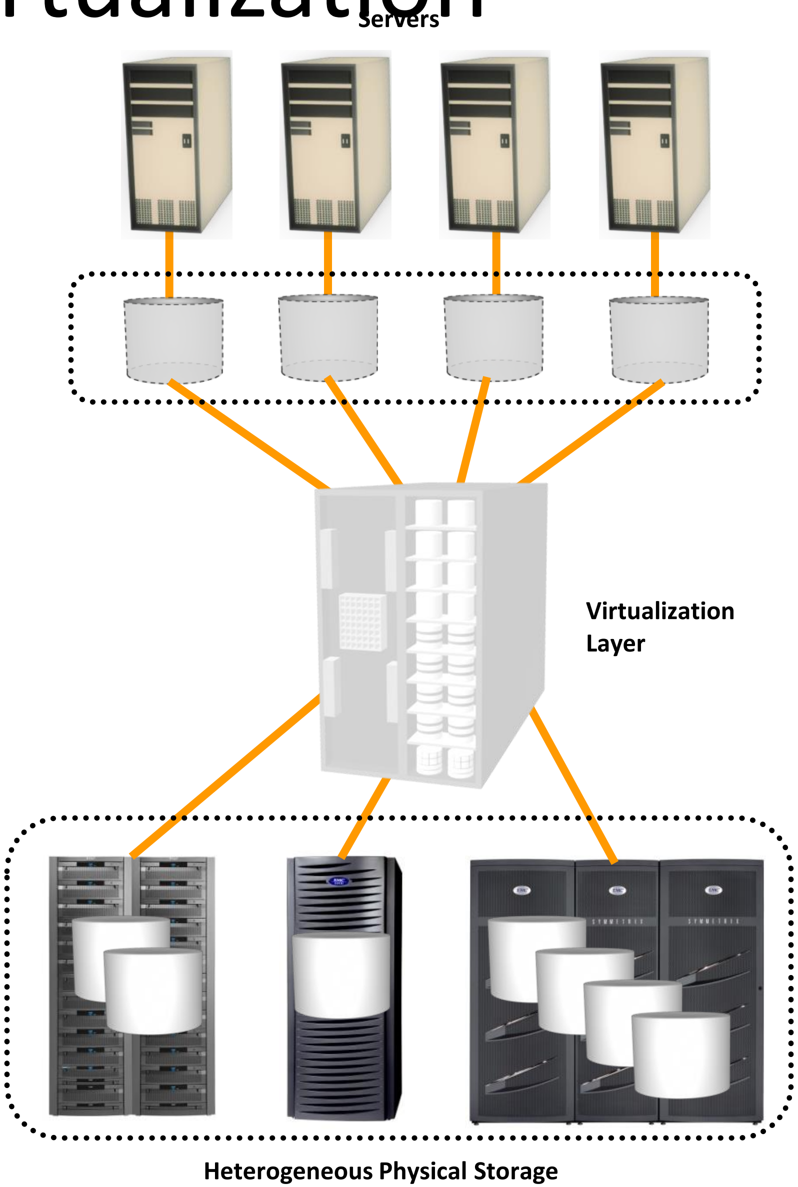
## After Server Virtualization:



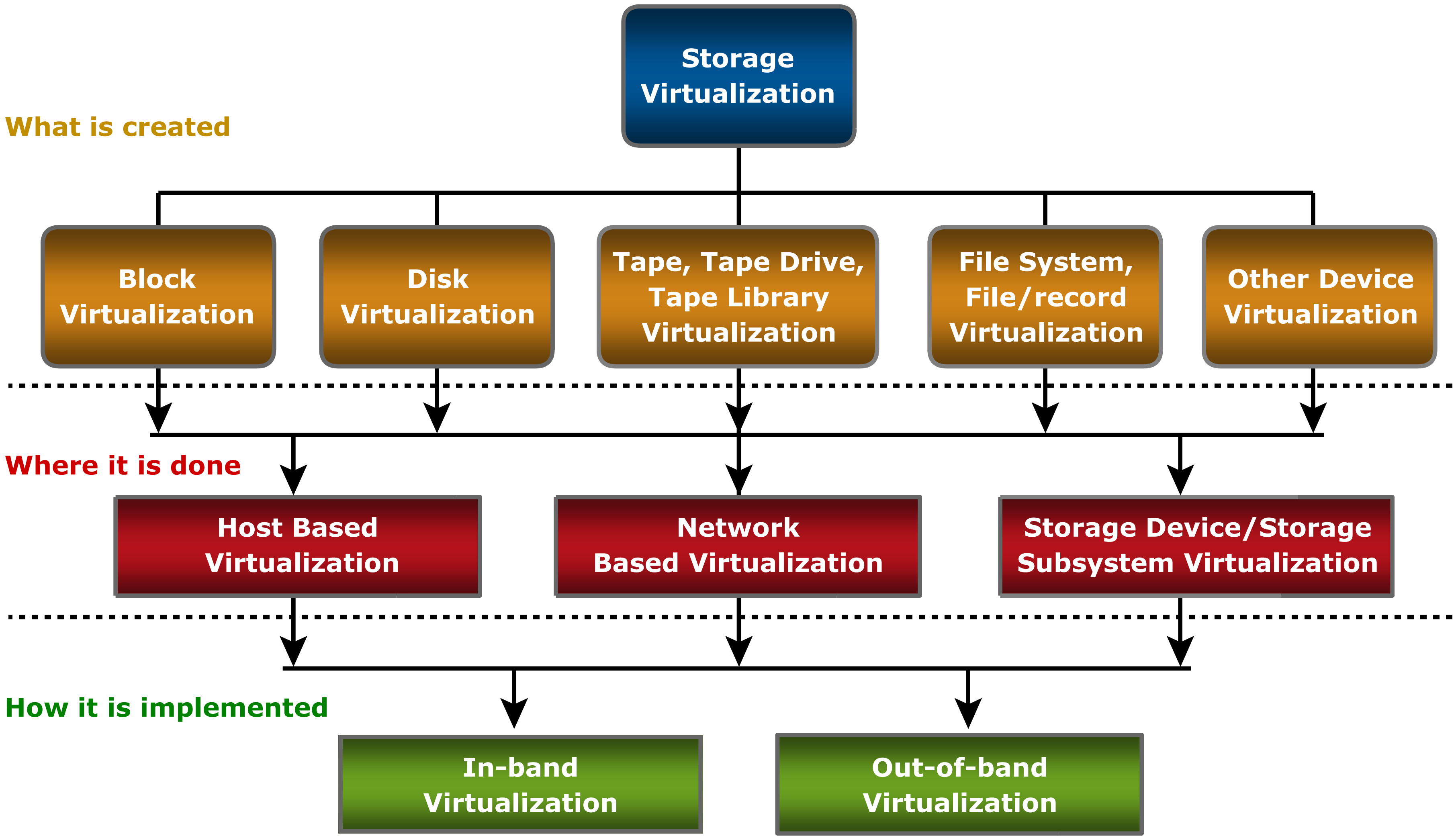

- Virtual Machines (VMs) break dependencies between operating system and hardware
- Manage operating system and application as single unit by encapsulating them into VMs
- Strong fault and security isolation
- Hardware-independent

# Storage Virtualization

- Process of presenting a logical view of physical storage resources to hosts
- Logical storage appears and behaves as physical storage directly connected to host
- Examples of storage virtualization are:
  - Host-based volume management
  - LUN creation
  - Tape virtualization
- Benefits of storage virtualization:
  - Increased storage utilization
  - Adding or deleting storage without affecting application's availability
  - Non-disruptive data migration



# SNIA Storage Virtualization Taxonomy



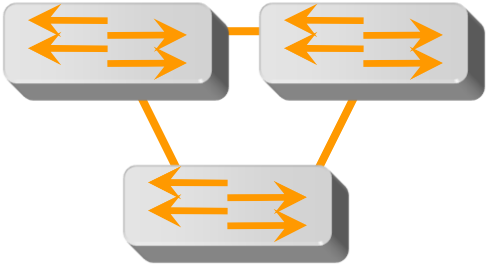
# Storage Virtualization Requires a Multi-Level Approach

**Server**

- Path management
- Volume management
- Replication

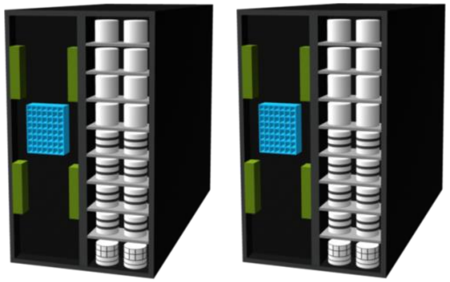
**Storage Network**

- Path redirection
- Load balancing - ISL trucking
- Access control - Zoning



**Storage**

- Volume management - LUNs
- Access control
- Replication
- RAID



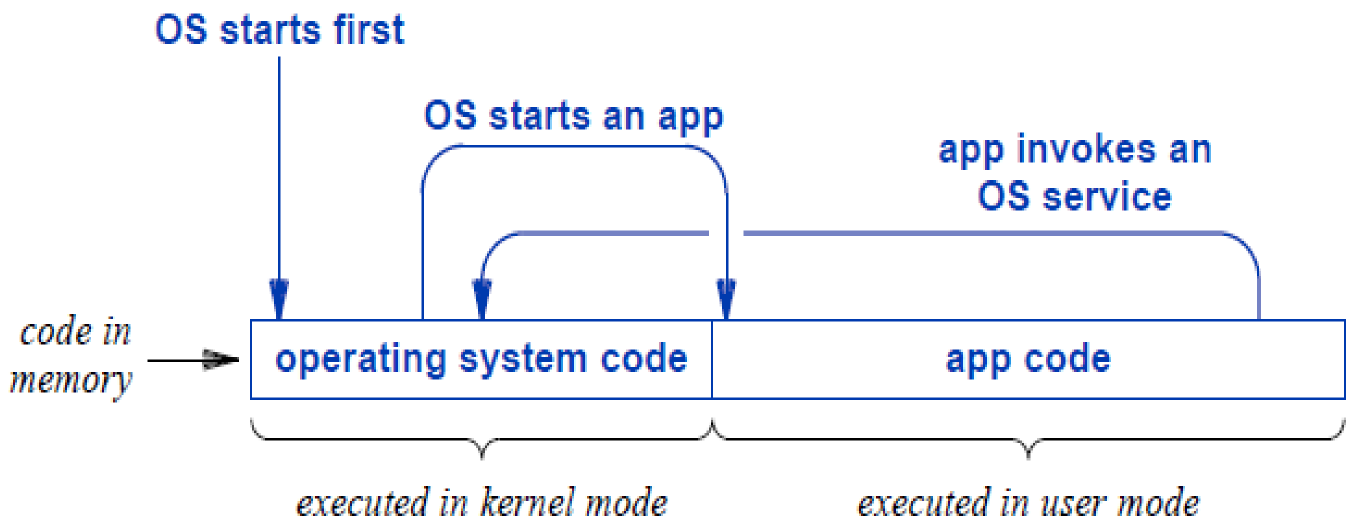
# Efficient Execution And Processor Privilege Levels

- When a user launches an app, the operating system loads the code for the app into the computer's memory.
- The operating system then instructs the processor to start executing the code.
- Execution proceeds at the hardware rate because the processor executes code for the app directly without going "through" the operating system.
- An applicationcannotbeallowedtoexecuteallpossibleinstructionsorthecom- puterwouldbevulnerabletohackerswhomightstealinformationorusethecom puter in a crime.
- To prevent such problems, the processor hardware used in a conventional computer has two ***privilege levels*** **or** ***modes of operation*****.**

## *Kernel mode*, **User Mode**

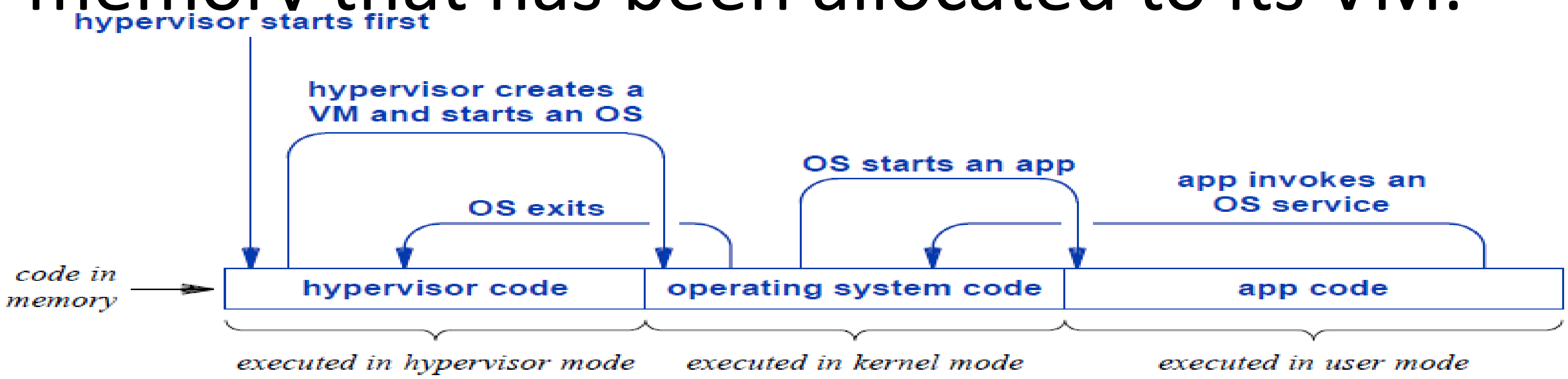
# Illustration of operating system and app code in memory.

- The processor executes each at the same high speed, changing mode when transitioning from one to the other.



# Extending Privilege To A Hypervisor

- Three levels of privilege: one for the hypervisor, a second for an operating system, and a third for apps.
- Only the hypervisor can create a VM and allocate memory to the VM.
- The operating system is restricted to the memory that has been allocated to its VM.



# Hypervisor

- The hypervisor sits directly between the physical hardware and its OS.
- The hypervisor provides ***hyper calls*** for the guest OSes and applications.
- *Microkernel architecture* like the Microsoft Hyper-V
- *Monolithic hypervisor architecture* like the VMware ESX for server virtualization.

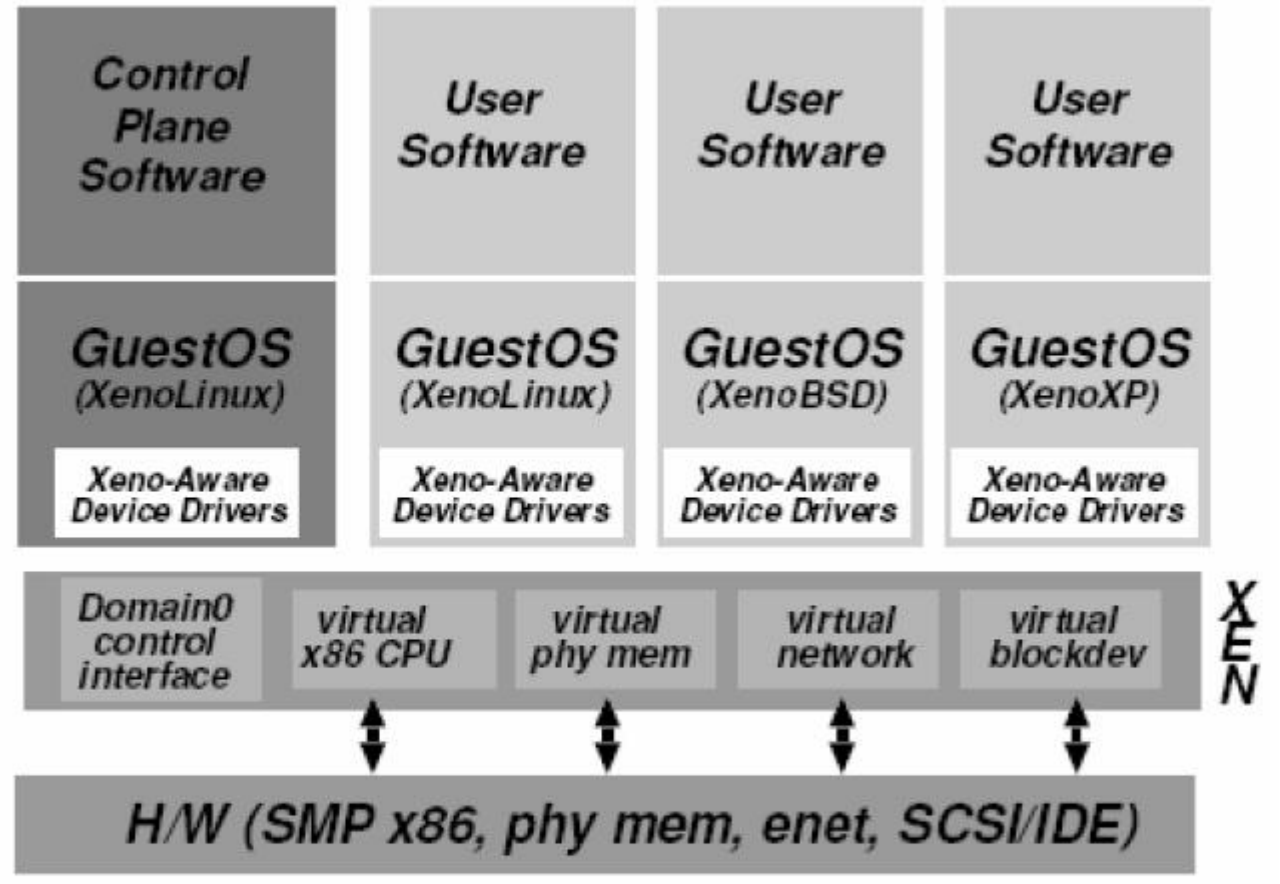
# Hypervisor

- A micro-kernel hypervisor includes only the basic and unchanging functions .

  **(such as physical memory management and processor scheduling).**

- The device drivers and other changeable components are outside the hypervisor.
- A monolithic hypervisor implements all the aforementioned functions, including those of the device drivers.

# Xen Architecture(Micro Kernel)



# Xen( Open Source)

- The core components of a Xen system are the hypervisor, kernel, and applications.
- Many guest OSes can run on top of the hypervisor. However, not all guest OSes are created equal, and one in particular controls the others.
- The guest OS, which has control ability, is called Domain 0, and the others are called Domain U. Domain 0 is a privileged guest OS of Xen.

# CPU Virtualization

- CPU Virtualization emphasizes running programs and instructions through a virtual machine, giving the feeling of working on a physical workstation.

- All the operations are handled by an emulator that controls software to run according to it.

# Software-Based CPU Virtualization

- CPU Virtualization is software-based where with the help of it, application code gets executed on the processor and the privileged code gets translated first, and that translated code gets executed directly on the processor.

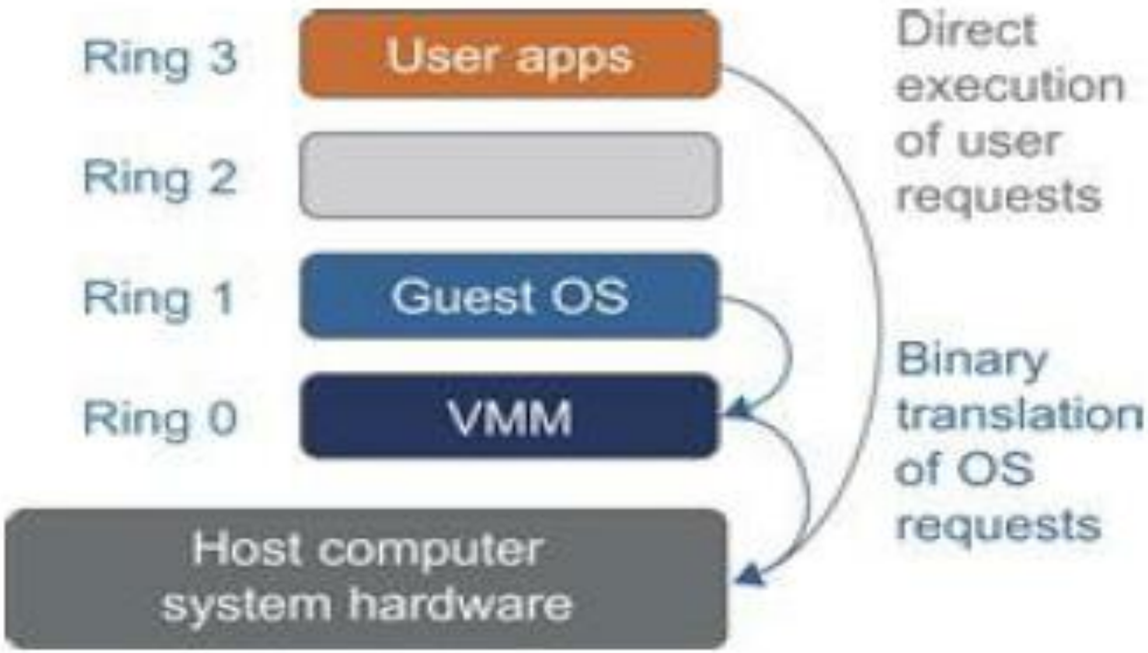
- This translation is purely known as Binary Translation (BT).

- The code that gets translated is very large in size and also slow at the same time on execution.

- The guest programs that are based on privileged coding runs very smooth and fast.

- The code programs or the applications that are based on privileged code components that are significant such as system calls, run at a slower rate in the virtual environment.

# CPU VIRTUALIZATION




**VMM scans the instruction stream and identifies the privileged, control- and behavior sensitive instructions.**
**When these instructions are identified, they are trapped into the VMM, which emulates the behavior of these instructions.**
**The method used in this emulation is called *binary translation*.**
**Therefore, full virtualization combines binary translation and direct execution.**
**The guest OS is completely decoupled from the underlying hardware.**

# Hardware-Assisted CPU Virtualization

- The guest user uses a different version of code and mode of execution known as a guest mode. The guest code mainly runs on guest mode.
- The best part in hardware-assisted CPU Virtualization is that there is no requirement for translation while using it for hardware assistance.
- For this, the system calls runs faster than expected.
- Workloads that require the updation of page tables get a chance of exiting from guest mode to root mode that eventually slows down the program's performance and efficiency.

# Memory Virtualization

- In a traditional execution environment, the operating system maintains mappings of *virtual memory* to *machine memory* using page tables, which is a one-stage mapping from virtual memory to machine memory.
- All modern x86 CPUs include a *memory management unit (MMU)* and a *Translation Lookaside Buffer (TLB)* to optimize virtual memory performance.
- Virtual memory virtualization involves sharing the physical system memory in RAM and dynamically allocating it to the *physical memory* of the VMs.

# Virtual Memory Virtualization

- Two-stage mapping process should be maintained by the guest OS and the VMM, respectively:

- Virtual memory to physical memory

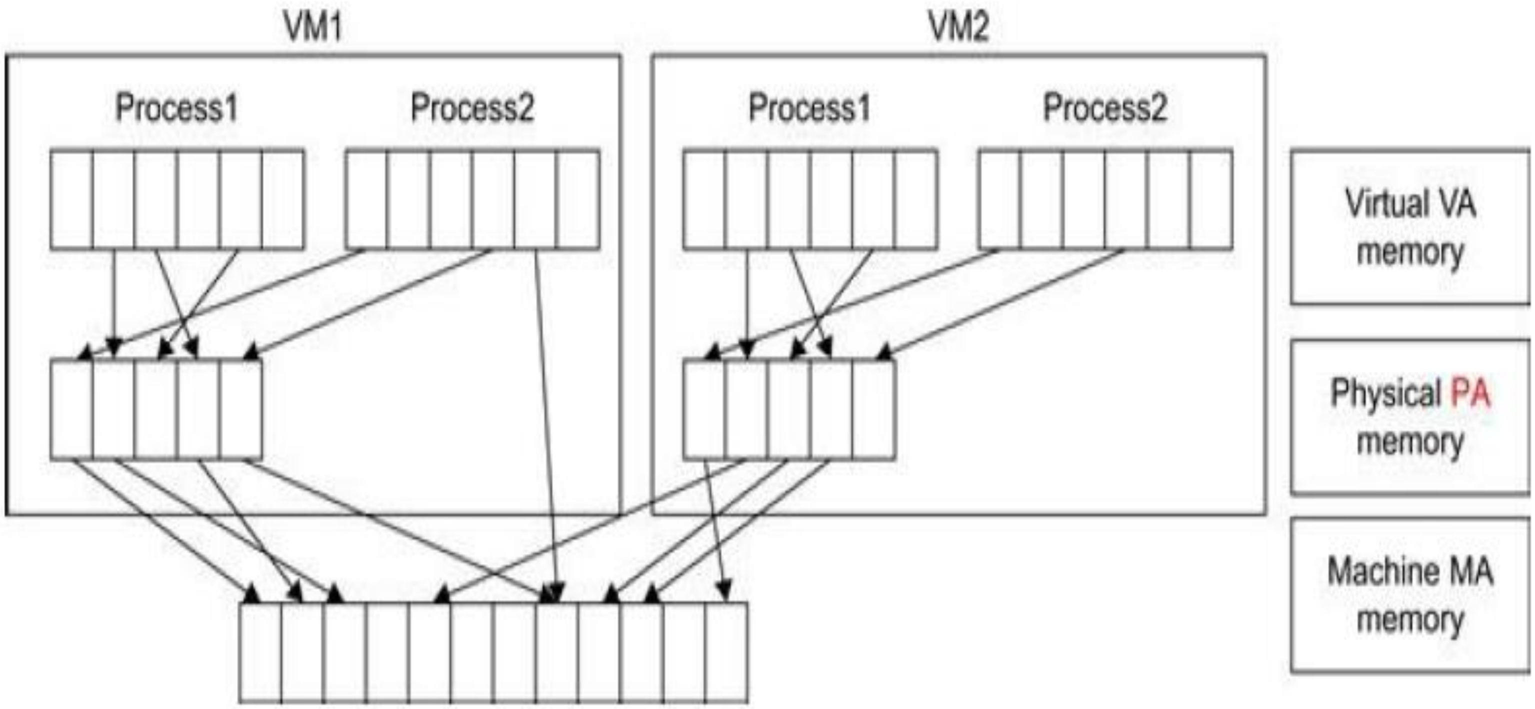
  and

  physical memory to machine memory.

# Virtual Memory Virtualization

- MMU virtualization should be supported, which is transparent to the guest OS.
- The guest OS continues to control the mapping of virtual addresses to the physical memory addresses of VMs. But the guest OS cannot directly access the actual machine memory.
- The VMM is responsible for mapping the guest physical memory to the actual machine memory

# Two-level memory mapping procedure



# Shadow Page Table

- Each page table of the guest OSes has a separate page table in the VMM corresponding to it, the VMM page table is called the shadow page table.
- VMware uses shadow page tables to perform virtual-memory-to-machine-memory address translation.
- Processors use TLB hardware to map the virtual memory directly to the machine memory to avoid the two levels of translation on every access.

# Levels Of Trust And I/O Devices

How an operating system manages I/O devices (e.g., a screen, keyboard, disk, and network interface) on a conventional computer?

- Operating system uses a hardware mechanism known as a *bus* to communicate with I/O devices.
- The first step consists of sending a series of requests across the bus to form a list of all I/O devices that are present.
- The operating system must include *device driver* software for each device.
- The operating system uses the device driver code for a given device to control the device hardware and handle all communication with the device.

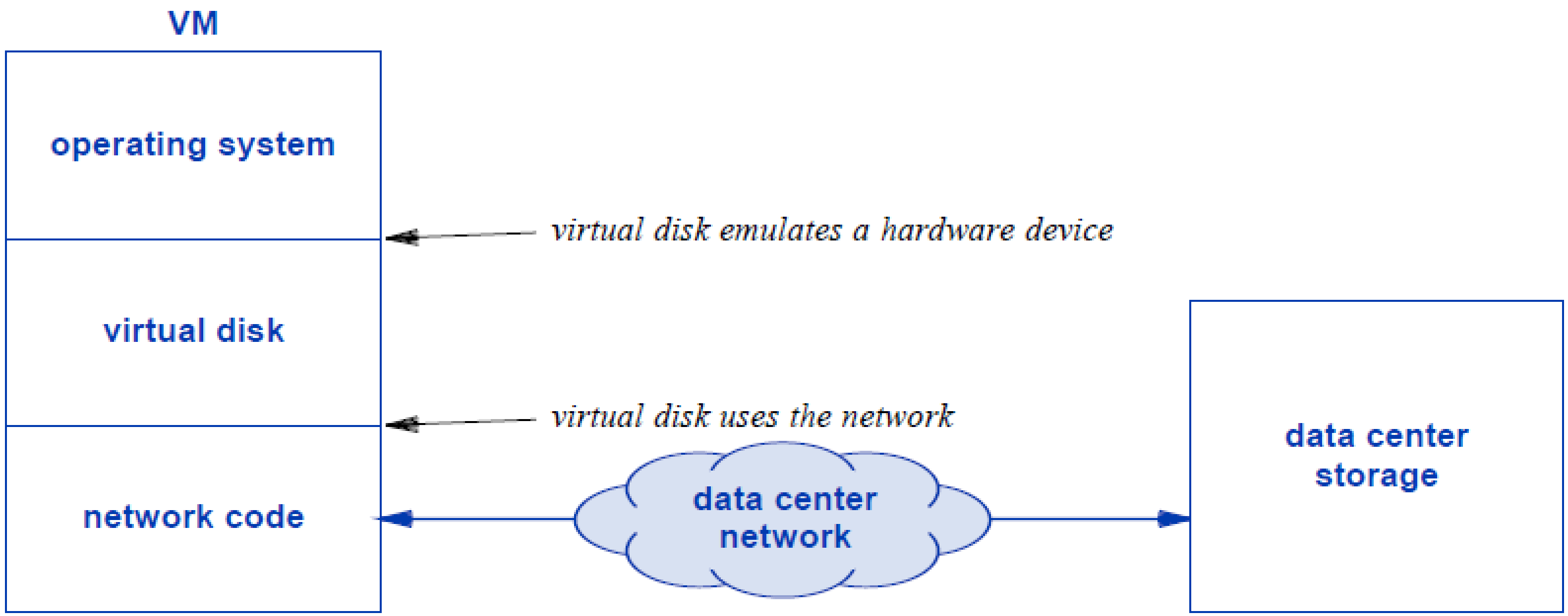
# Virtual I/O Devices

- Hypervisor creates a set of *virtual I/O devices* for the VM to use.
- A Virtual I/O device is implemented by software.
- When an operating system on a VM attempts to use the bus to access an I/O device, the access violates privilege, which means the hypervisor is invoked.
- The hypervisor runs the appropriate virtual device, software, and then arranges to send the response to the operating system as if a physical device responded.

# VM As A Digital Object

- A hypervisor must keep a record of the VM
- Memory that have been allocated to the VM
- The virtual I/O devices that have been created for the VM(including disk space that has been allocated in the data center storage facility).
- Current status of the VM
- VM can be turned into a digital object. That is, the entire VM can be transformed into a set of bytes
- Imagine, for example, that they are placed in a special file.
- ***All the pieces of a VM can be collected together into a digital object.***

# An Example Virtual Device



# VM Migration

- The movement of VMs from one resource to another, such as from one physical host to another physical host, or data store to data store, is known as VM migration.

- There are two types of VM migration:

✓ **Cold**

✓ **Hot (Live)**

# Cold-Hot (Live)

- Cold migration occurs when the VM is shut down.

- Live migration is the process of moving a running virtual machine without stopping the OS and other applications from source host to destination host.

# Cold Migration

- Cold migration is the migration of powered off or suspended virtual machines between hosts across clusters, data centers, and vCenter Server instances.
- By using cold migration, you can also move associated disks from one datastore to another.
- If you attempt to migrate a powered off virtual machine that is configured with a 64-bit operating system to a host that does not support 64-bit operating systems, vCenter Server generates a warning.

# Data Store

- **Datastores** in VMware vSphere are storage containers for files.
- They could be located on a local server hard drive or across the network on a SAN.
- Datastores hide the specifics of each storage device and provide a uniform model for storing virtual machine files.
- Datastores are used to hold virtual machine files, templates, and ISO images.
- They can be formatted with **VMFS** (**Virtual Machine File System**, a clustered file system from VMware)

# VMware vSphere

**VMware vSphere** is a software suite that includes components like ESXi, vCenter Server, vSphere Client, vCenter Orchestrator, vSphere Update Manager, etc.

vSphere components provide virtualization, management, resource optimization and many other features useful for a virtual environment

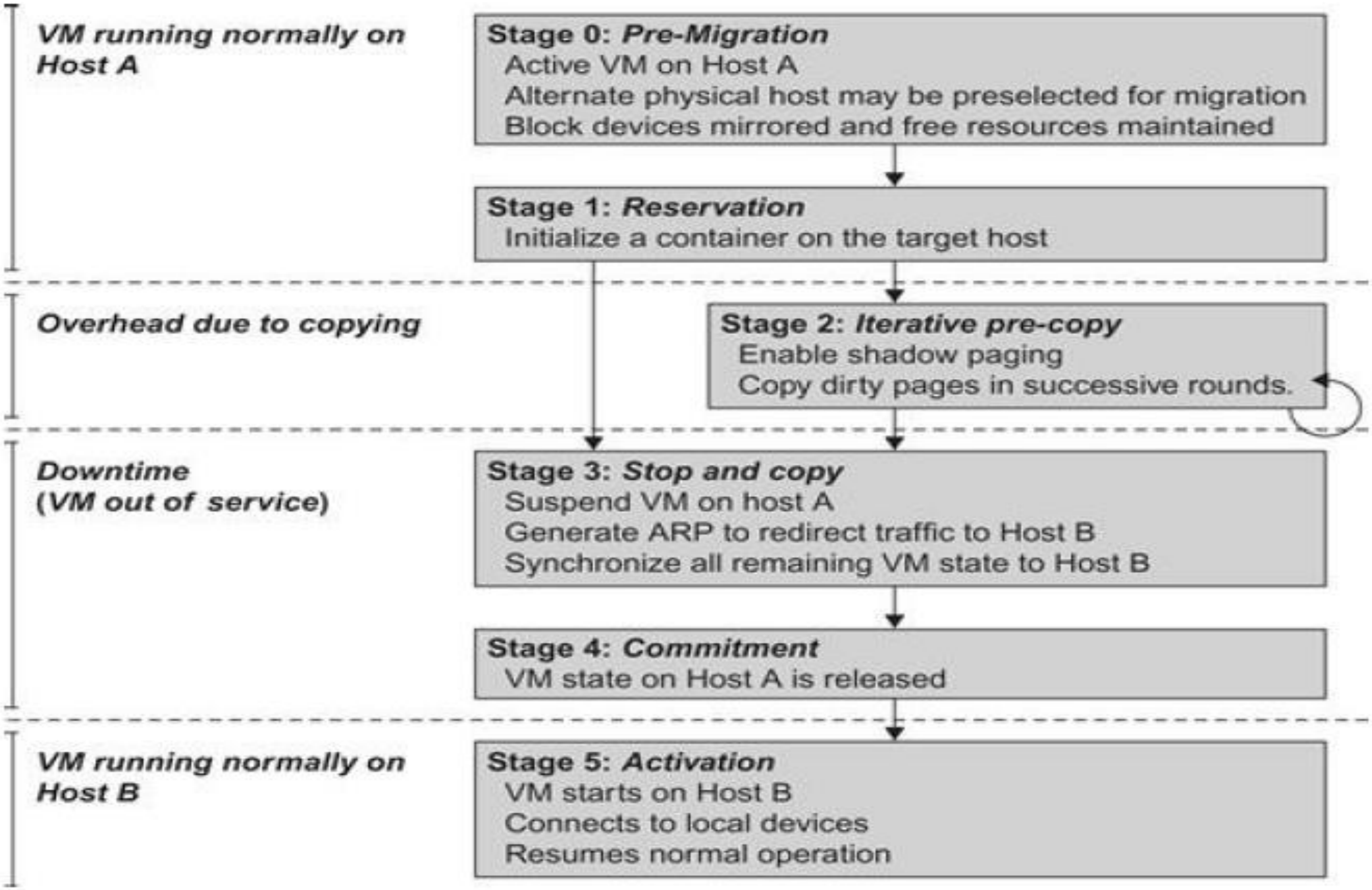
- https://www.vmware.com/products/vsphere.html

# LIVE VM migration

- **Live migration** refers to the process of moving a **running** virtual machine or application between different physical machines without disconnecting the client or application.

- Memory, storage, and network connectivity of the virtual machine are transferred from the original guest machine to the destination.

# Live migration process of a VM from one host to another



# VMware vSphere vMotion

- VMware vSphere vMotion is a zero downtime live migration of workloads from one server to another.
- This capability is possible across vSwitches, Clusters, and even Clouds (depending of the vSphere edition that you have).
- During the workload migration, the application is still running and users continue to have access to the systems they need.

# Virtual Networks

**Conflicting Goals For A Data Center Network**

- ✓ Universal connectivity
- ✓ *Safe, isolated communication*

*On the one hand, to permit VMs and containers to communicate between arbitrary physical servers, a data center network must provide universal connectivity.*

*On the other hand, a given tenant seeks a network architecture that keeps their VMs and containers isolated and safe.*

# Virtual Networks, Overlays, And Underlays

- How can a provider keep each tenant's traffic isolated from other tenants' traffic?
- A cloud service with thousands of tenants makes separate physical networks impractical
- The answer lies in an approach known as ***network virtualization***.

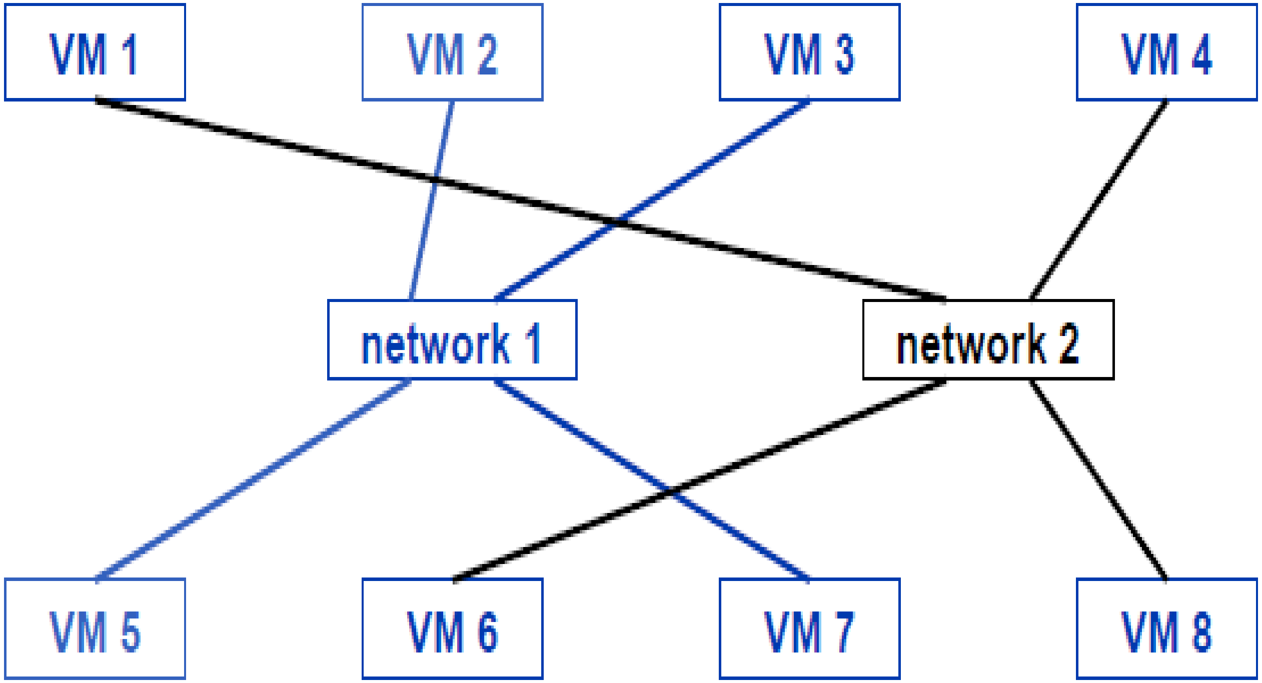
# Virtual Networks, Overlays, And Underlays

- Each virtual network links a tenant's virtual machines and containers.

- Overlay network to refer to a virtual network that does not actually exist but which in effect has been created by configuring switches to restrict communication.

- *Underlay network* to refer to the underlying physical network that provides connections among entities in a virtual network.

# Virtual Networks




**VLAN technology imposes a set of virtual overlay networks on a set of switches, and each computer attached to the switches is assigned to one of the virtual networks.**

# Scaling VLANs To A Data Center With VXLAN

- ***Virtual Extensible LAN* (*VXLAN*)**

Each switch to include VXLAN software, and requires the network administrator to configure special routing protocols

VXLAN system can learn the locations of computers in the data center.

Once it has been configured, VXLAN can provide the equivalent of more than sixteen million virtual networks — enough for even the largest cloud providers.

# Virtual Network Switch Within A Server

- Separate IP address to each virtual machine.
- When multiple VMs run on the same physical server, multiple addresses will be assigned to the server.
- If two VMs in the same server communicate, packets must be forwarded from one to the other
- How can a hypervisor forward packets among the VMs it has created? (Open vSwitch or Virtual Network Switch)
- A piece of software known as a virtual network switch allows a server to assign each VM its own IP address and forward packets among the VMs and the datacenter network; a virtual switch can be configured to follow the same forwarding rules as other datacenter switches.

# Network Address Translation (NAT)

- **A container can clone the host's IP address**
- **A container can receive a new IP address**
- **A container can use address translation**

➢ If a container uses the same IP address as the host OS,we say that the container has *cloned* the address.

➢ Each container can be assigned a unique IP address, and the host operating system can use a virtual switch to provide connectivity .

➢ When used with containers, NAT software runs in the host operating system. When a container that uses NAT begins execution ,the container requests an IP address, and the NAT software responds to the request by assigning an IP address from a set of reserved, *private* IP addresses that cannot be used on the Internet.

# Containers and Virtual Machines

- A container is a software code package containing an application's code, its libraries, and other dependencies.

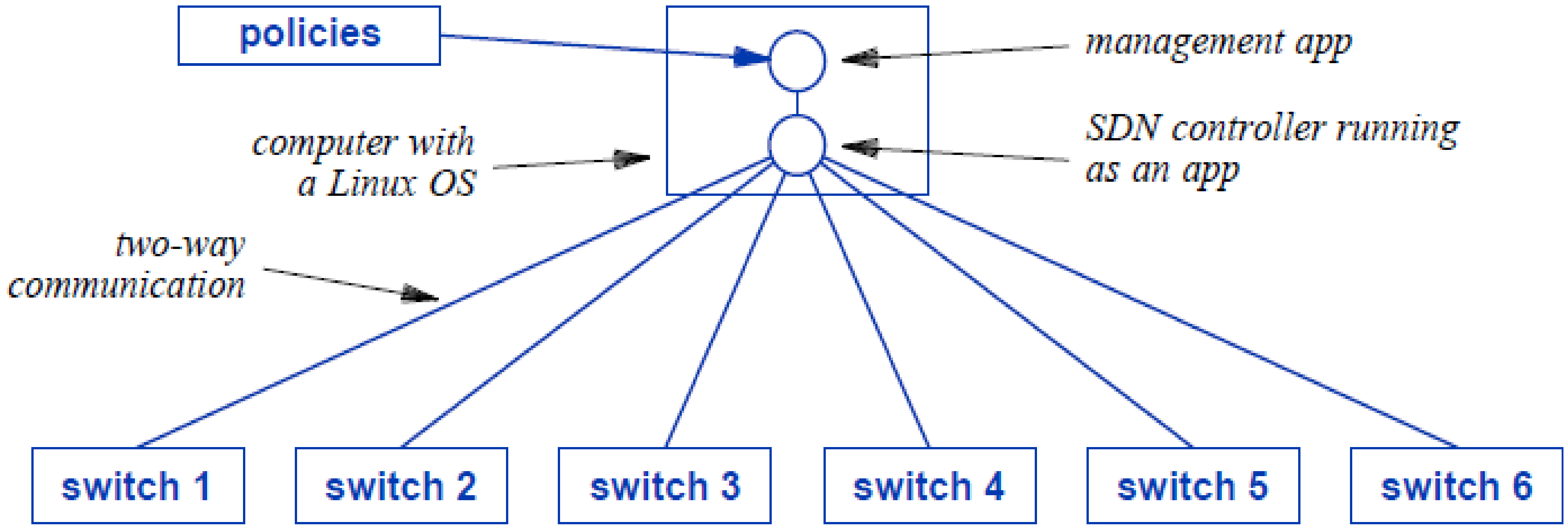
- Containerization makes your applications portable so that the same code can run on any device.

- A virtual machine is a digital copy of a physical machine.

- *The complex interconnections, multiple levels of virtualization, arbitrary placement of addressable entities, and VM migration make configuring and operating a data center network especially difficult.*
- **Automated Network Configuration And Operation**
- *Standard routing protocols and Spanning TreeProtocol* (*STP*) .
- *OSPF* (*Open Shortest Path First*) and *BGP* (*Border Gateway Protocol*)

# Software Defined Networking

- SDN allows a manager to specify high-level policies, and uses a computer program to configure and monitor network switches according to the policies.



# SDN Approach

- The SDN approach uses a dedicated computer to run SDN software on a conventional operating system.
- An ***SDN controller*** **app, and a management app.**
- A management app can use policies to choose how to forward packets.
- The controller forms a logical connection to a set of switches and communicates appropriate forwarding rules to each switch.
- Logical connections between an SDN controller and each switch employ **bi directional communication** that allows data to flow in either direction.
- Controller can monitor the status of the switch itself and the links to other switches. (For example, SDN can configure a switch to inform the controller when events occur, such as link failure or recovery.)

# OpenFlow Protocol

- *OpenFlow* protocol standard defines the communication available to an SDN controller.

- To use SDN, a switch needs an OpenFlow module; the switches used in data centers include such support.

- A controller installs a set of forwarding rules in each switch.

- Each rule describes a particular type of packet and specifies the output port over which the packet should be sent.

- A forwarding rule uses items in a packet header to decide where the packet should be sent.

(World Wide Web traffic one port, database traffic out another port, and all other packets out a third port)

# Header fields that a controller can use in a forwarding rule.

| Field | Meaning |
| --- | --- |
| VLAN tag | The VLAN to which the packet belongs |
| IP source | The sending computer |
| IP destination | The ultimate destination |
| TCP source | The type of the sending application |
| TCP destination | The type of the receiving application |

# Programmable Networks

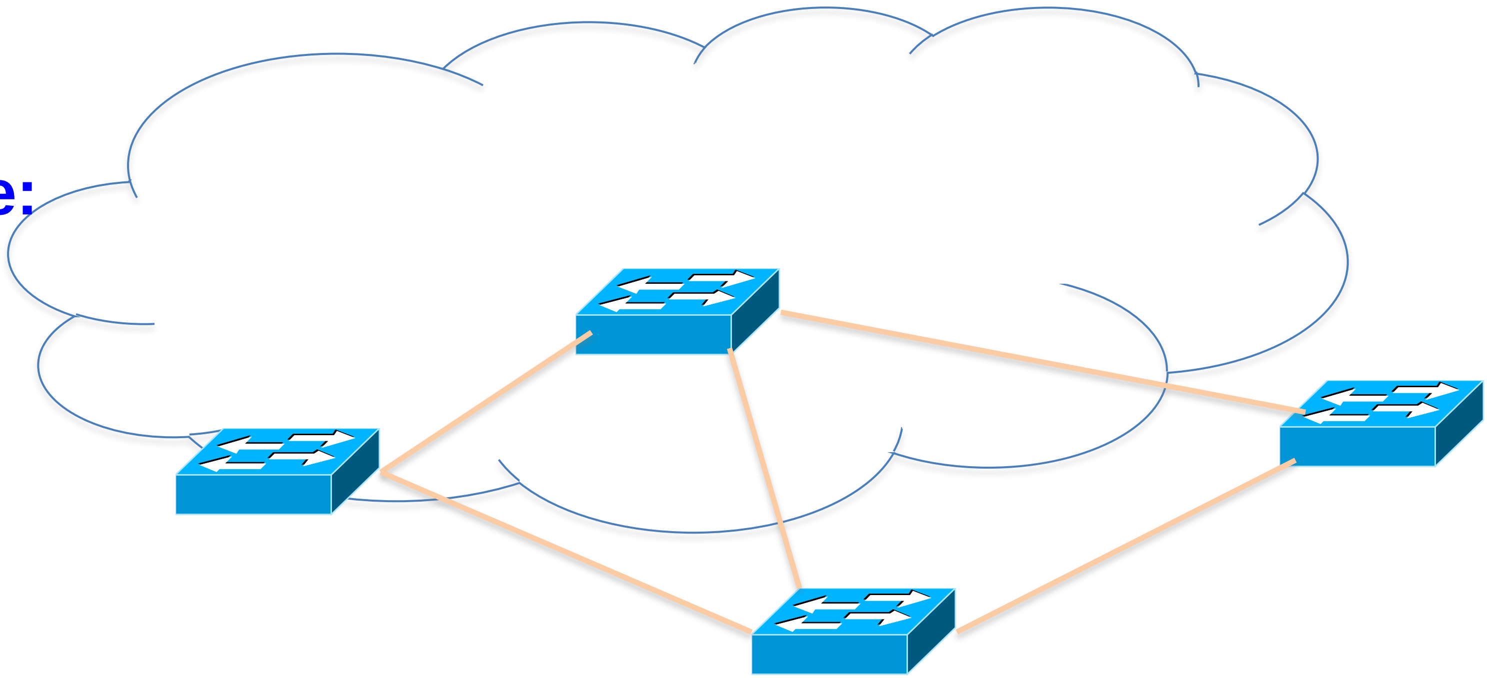
- *The second generation of SDN technology allows an SDN controller to install a computer program written in the* P4 *language in each switch.*

- Programming Protocol-independent Packet Processors (P4) is an open source, domain-specific programming language for network devices, specifying how data plane devices (switches, routers, NICs, filters, etc.) process packets.

- *The use of a program means a switch can handle exceptions locally without sending each exception to the controller.*

# Traditional Computer Networks

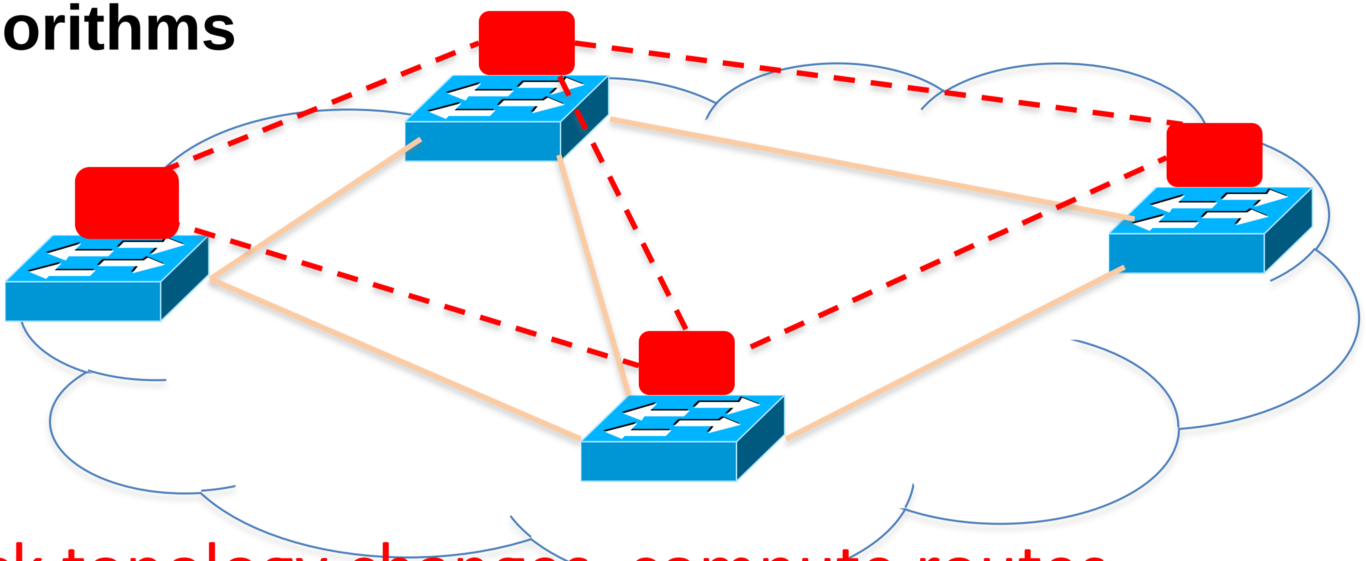
**Data plane:** **Packet streaming**



Forward, filter, buffer, mark, rate-limit, and measure packets

# Traditional Computer Networks

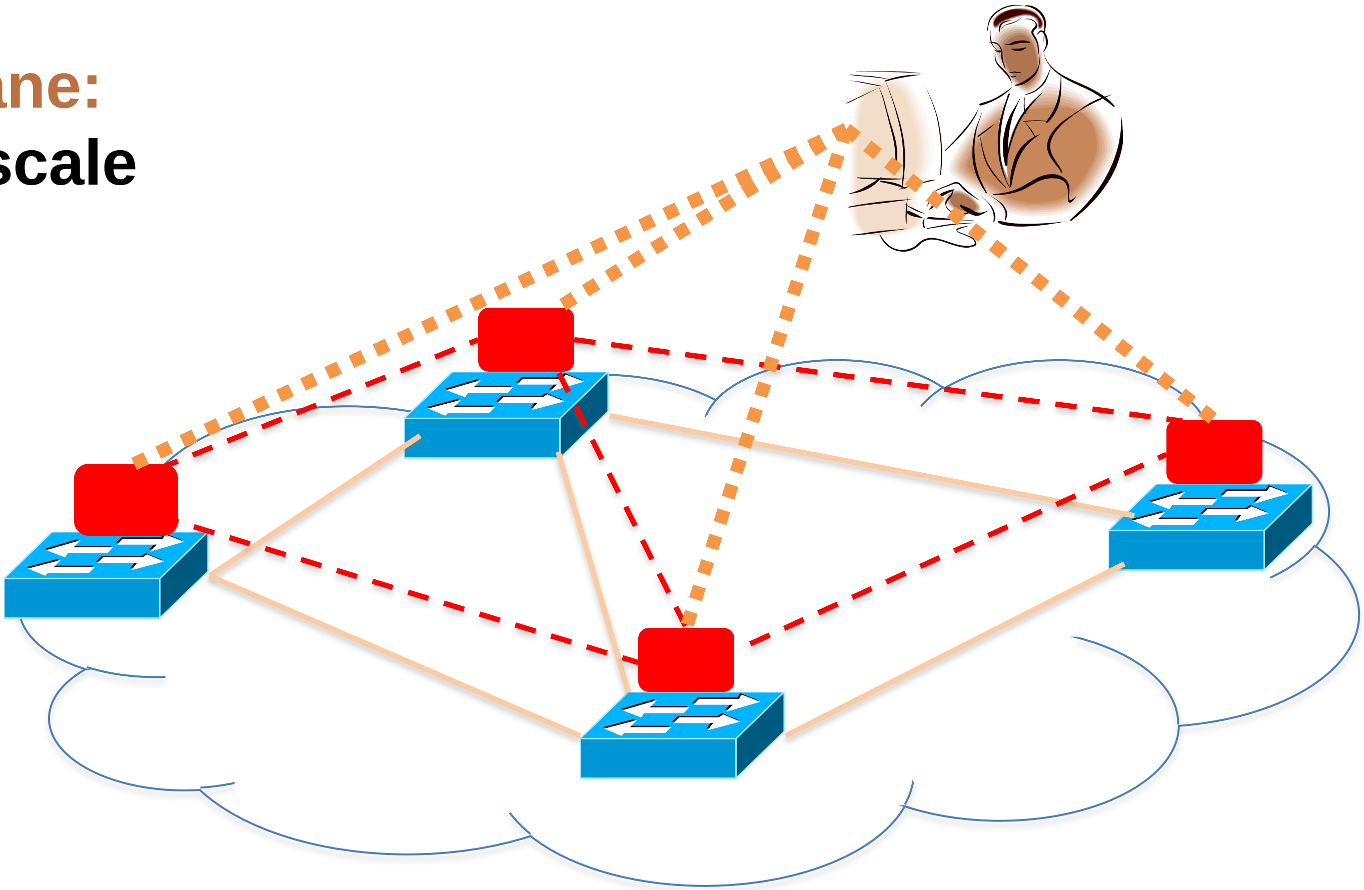
**Control plane:**
**Distributed algorithms**



Track topology changes, compute routes, install forwarding rules

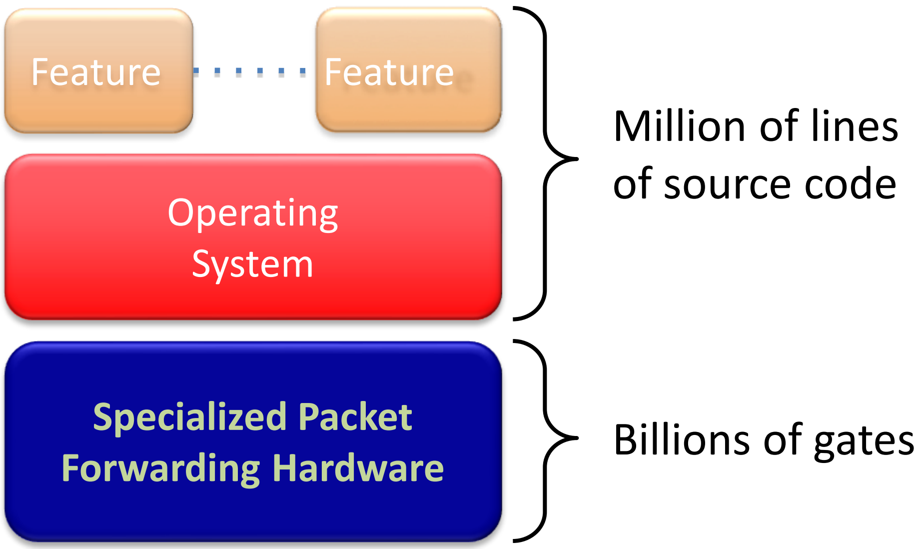
# Traditional Computer Networks

**Management plane:**
**Human time scale**



Collect measurements and configure the equipment

# Limitations of Current Networks



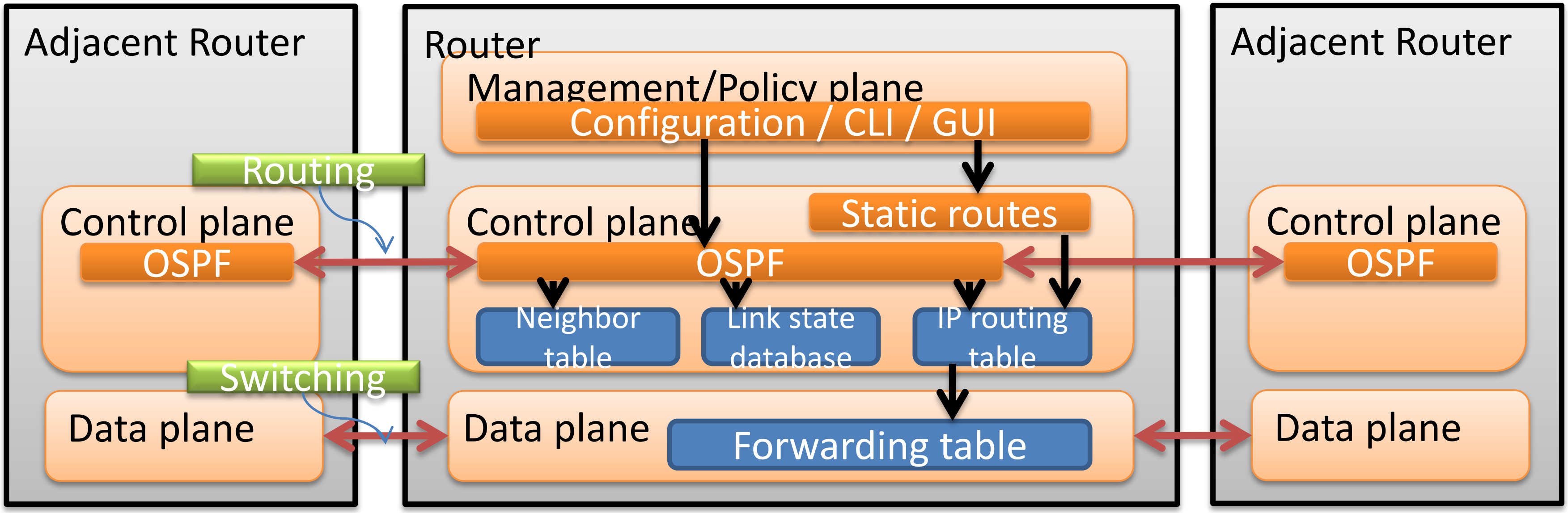

Many complex functions baked into infrastructure

*OSPF, BGP, multicast, differentiated services, Traffic Engineering, NAT, firewalls, …*

Cannot dynamically change according to network conditions

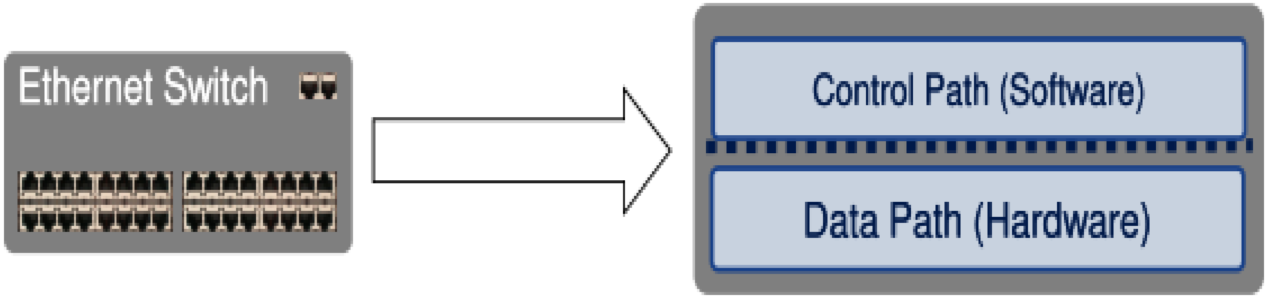
# Traditional Network Router

- Router can be partitioned into control and data plane
  - Management plane/ configuration
  - Control plane / Decision: OSPF (Open Shortest Path First)
  - Data plane / Forwarding



# Traditional network Router In Summary

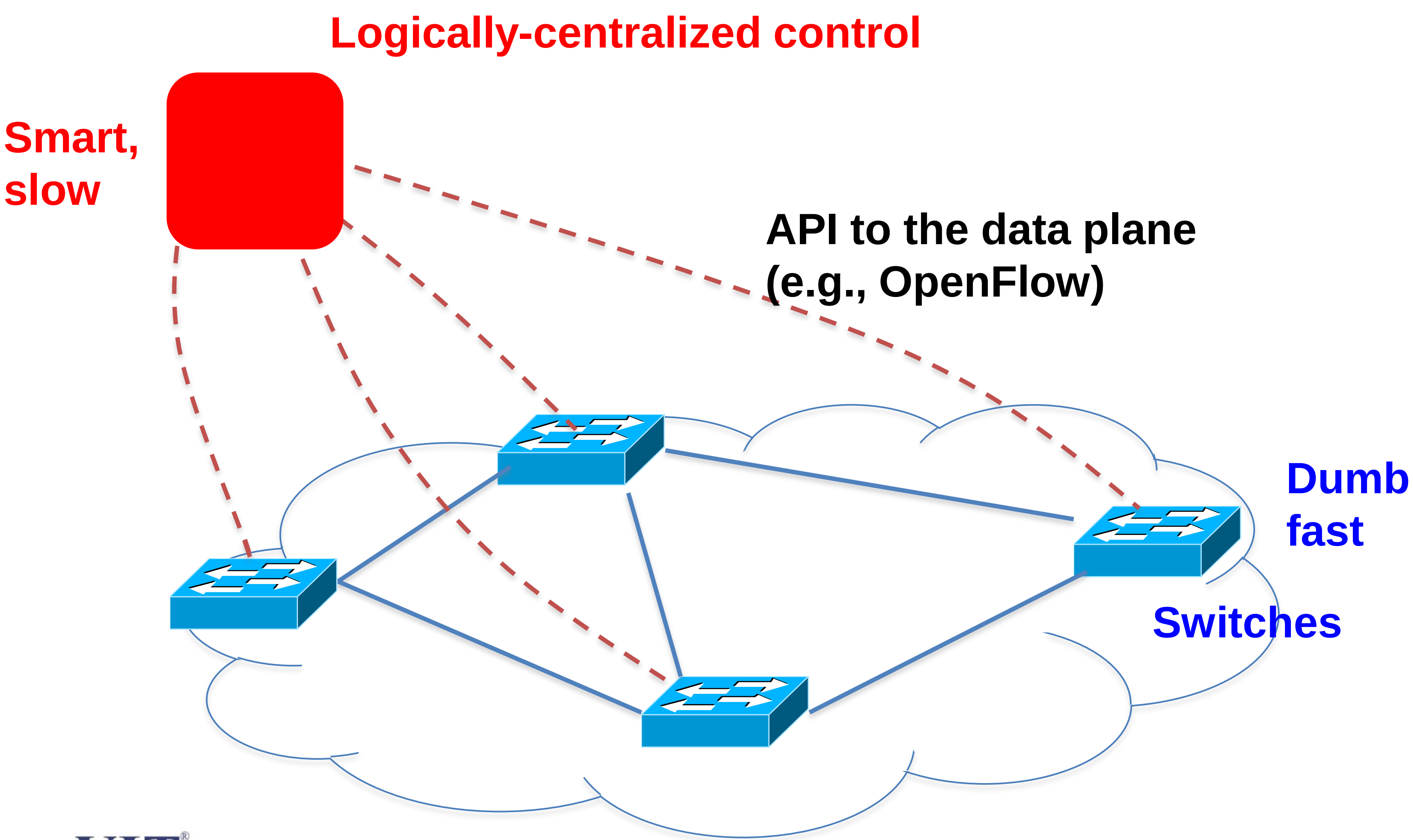
- Typical Networking Software
  - Management plane
  - Control Plane – The brain/decision maker
  - Data Plane – Packet forwarder



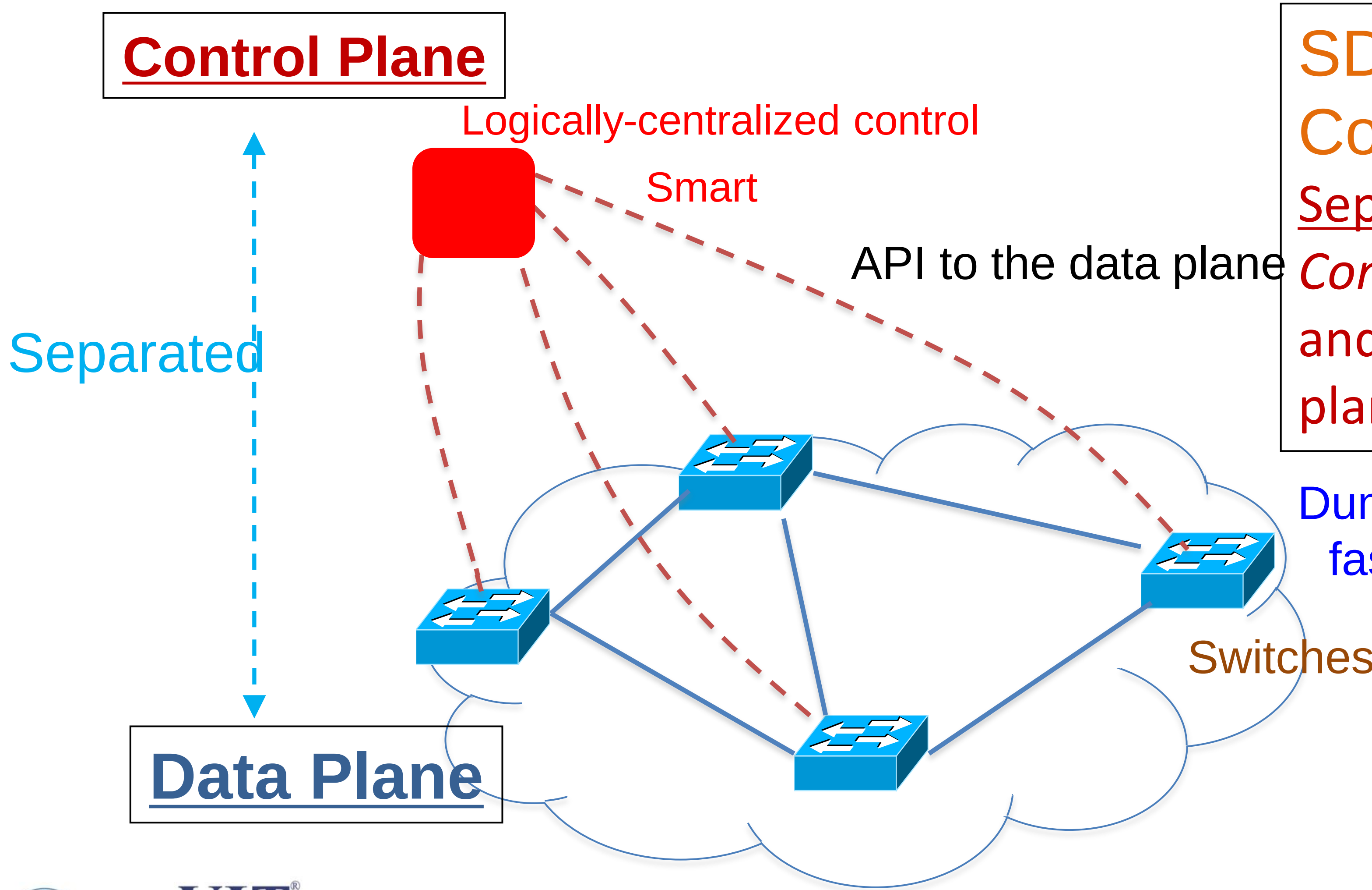
# Limitations

- Data Plane and Control Plane are coupled
- Complex Devices- Static Patterns-Dynamic
- Management Overhead
- Limited Scalability

# Software Defined Networking (SDN)

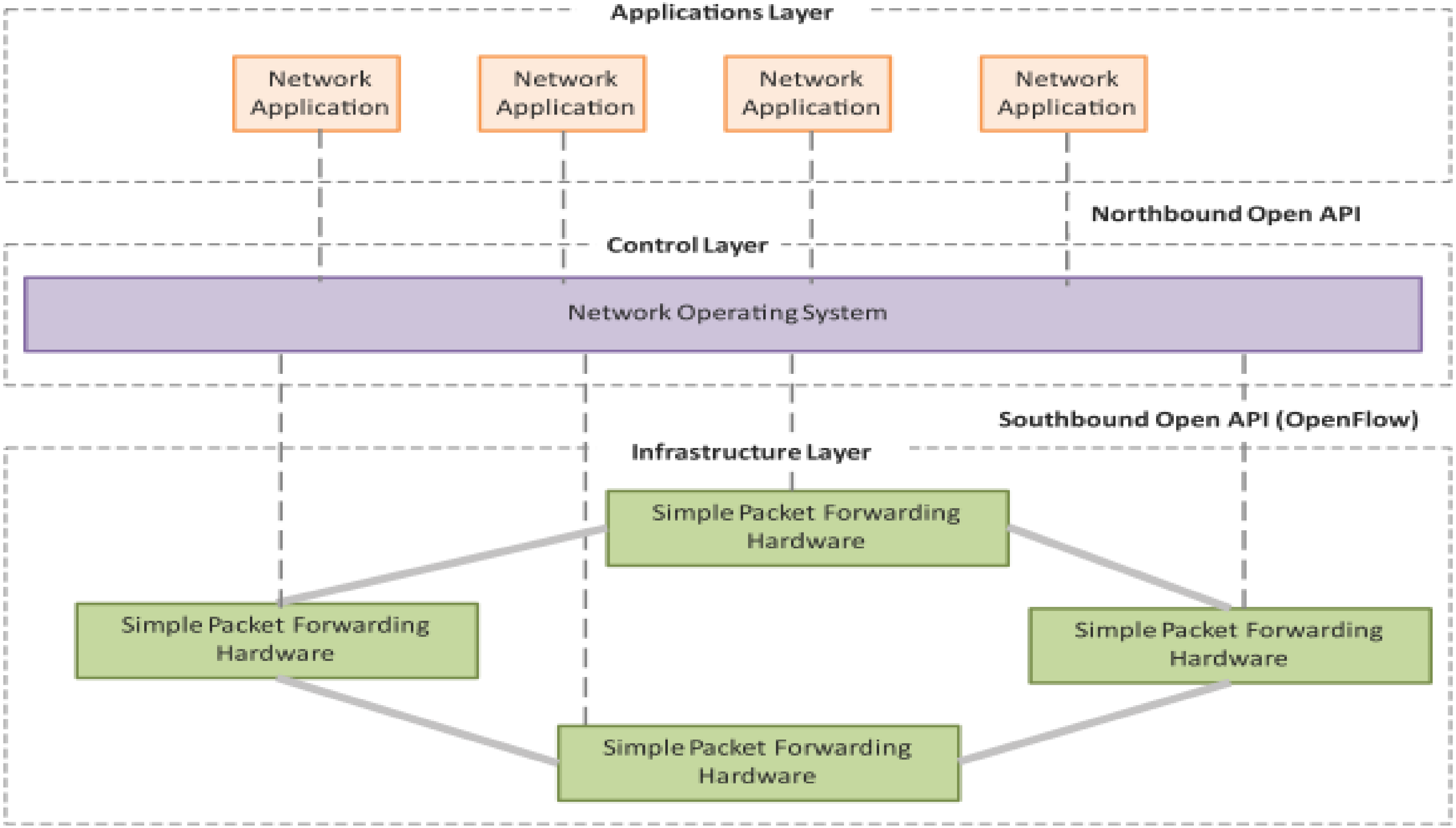


# Imagine **IF** The Network is........!!!

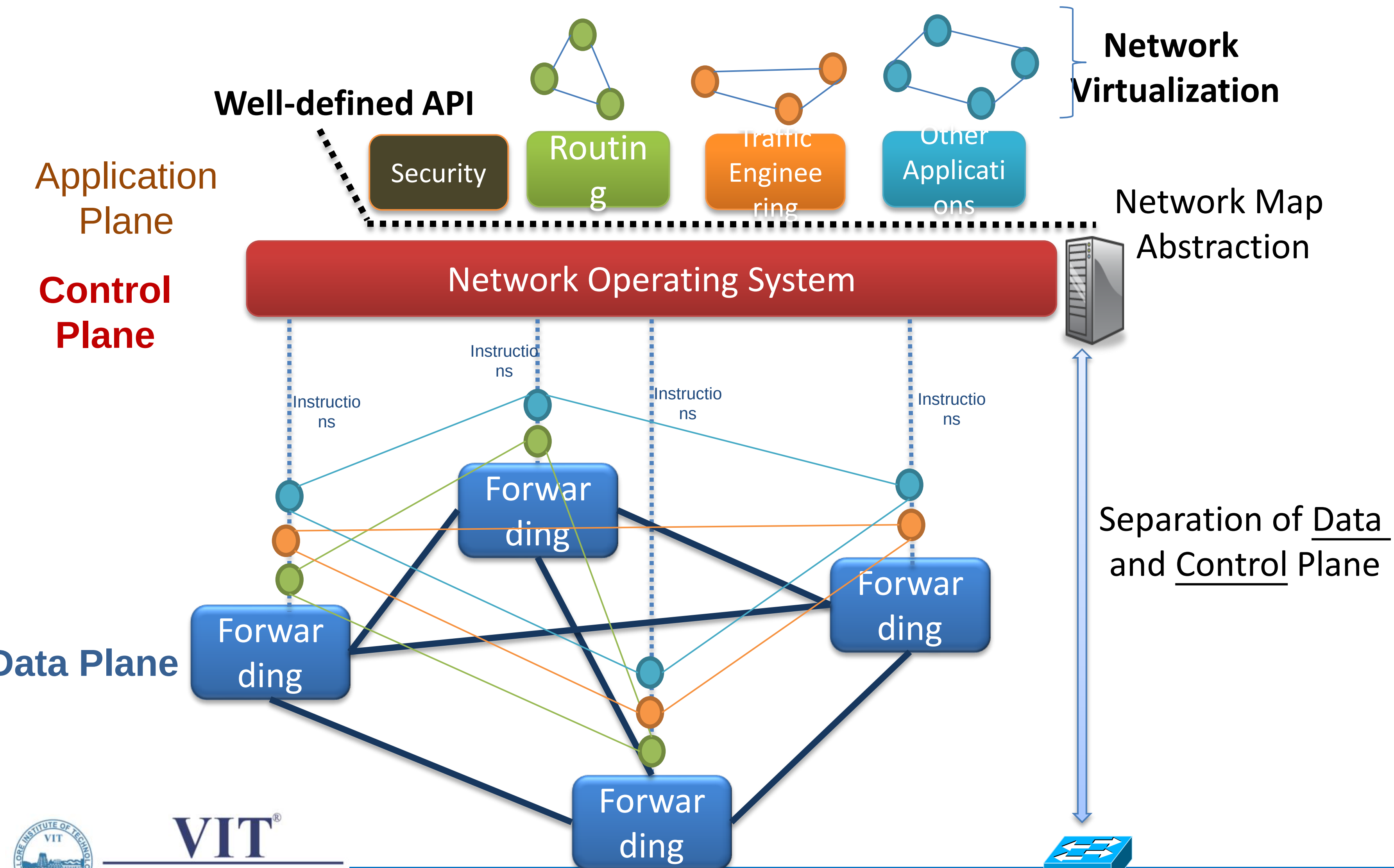



SDN Concept:
Separate *Control* plane and *Data* plane.

# SDN Architecture



# Software-Defined Network with key Abstractions



# Software Defined Networking

- Simpler,Inexpensive,scalable and easy to manage

- Software-Defined Networking (SDN) is a networking architecture that separates the control plane from the data plane and centralizes the network controller.

- Software-based SDN controllers maintain a unified view of the network and make configuration, management and provisioning simpler & uses simple packet forwarding hardware as opposed to specialized hardware in conventional networks.

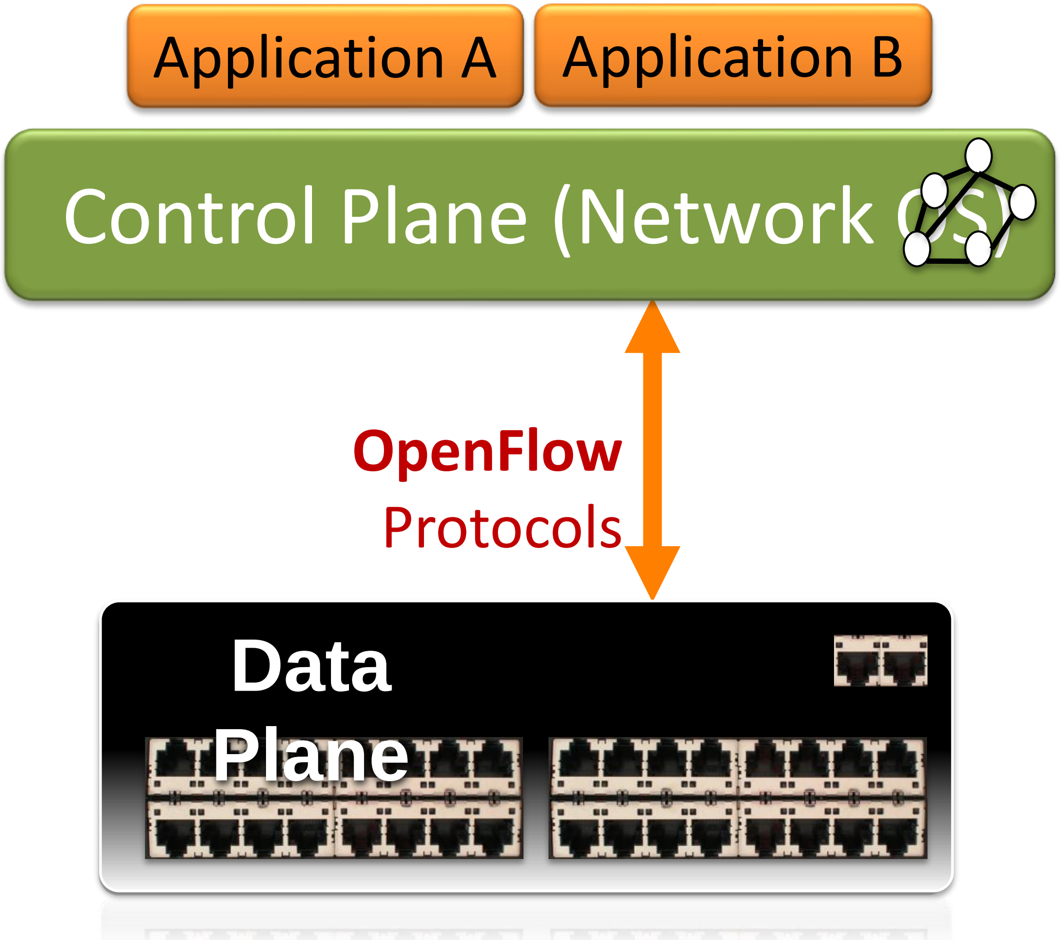
# SDN Basic Concept

- Separate Control plane and Data plane entities.
  - Network intelligence and state are logically centralized.
  - The underlying network infrastructure is abstracted from the applications.
- Execute or run Control plane software on general purpose hardware.
  - Decouple from specific networking hardware.
  - Use commodity servers and switches.
- Have programmable data planes.
  - Maintain, control and program data plane state from a central entity.
- An architecture to control not just a networking device but an entire network.

# SDN in Real World – Google's Story

- The industries were skeptical whether SDN was possible.
- Google had big problems:
  - **High financial cost** managing their datacenters: Hardware and software upgrade, over provisioning (fault tolerant), manage large backup traffic, time to manage individual switch, and  a lot of men power to manage the infrastructure.
  - **Delay** caused by rebuilding connections after link failure.
    - Slow to rebuild the routing tables after link failure.
    - Difficult to predict what the new network may perform.
- Google went a head and implemented SDN.
  - Built their hardware and wrote their own software for their internal datacenters.
  - Surprised the industries when Google announced SDN was possible in production.
- How did they do it?
  - Read "*B4: Experience with a Globally-Deployed Software Defined WAN*", ACM Sigcomm 2013.

# What is OpenFlow?



# What is OpenFlow?

- Allow separation of control and data planes.
- Centralization of control.
- Flow based control.
- Takes advantage routing tables in Ethernet switches and routers.

- SDN is not OpenFlow.
  - **SDN** is a concept of the physical separation of the network control plane from the forwarding plane, and where a control plane controls several devices.
  - ***OpenFlow*** is communication interface between the control and data plane of an *SDN architecture*.
    - Allows direct access to and manipulation of the forwarding plane of network devices such as switches and routers, both physical and virtual.
    - Think of as a protocol used in switching devices and controllers interface.

# How is OpenFlow related to SDN in The Nut Shell?

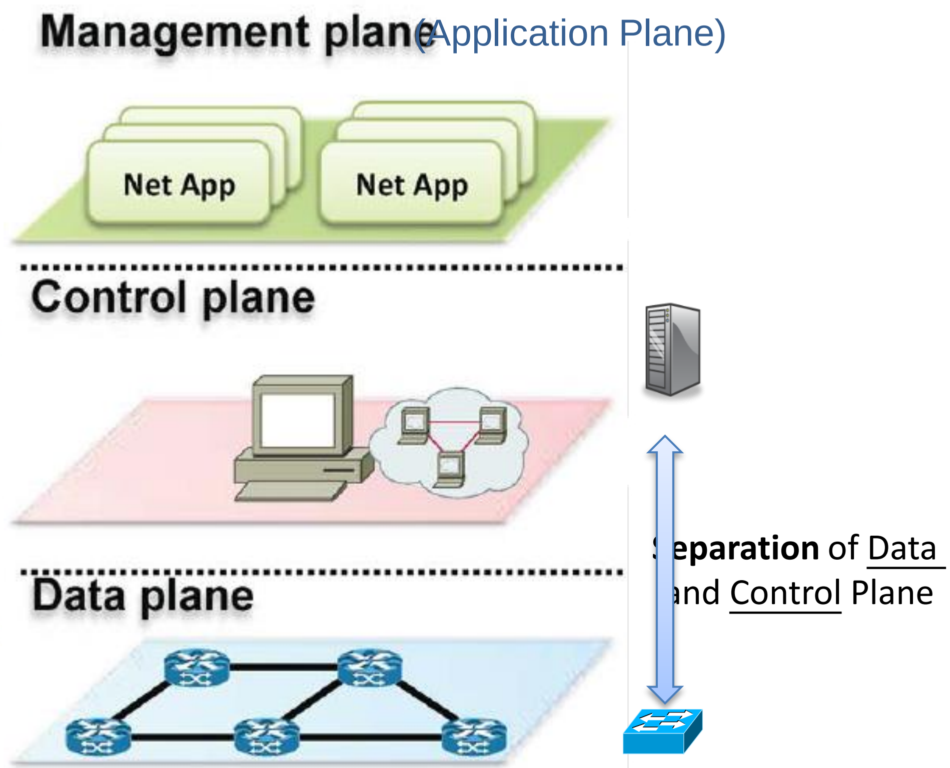
*OpenFlow* allows you to do:

**Programmability**
- Enable innovation/differentiation
- Accelerate new features and services introduction

**Centralized Intelligence**
- Simplify provisioning
- Optimize performance
- Granular policy management

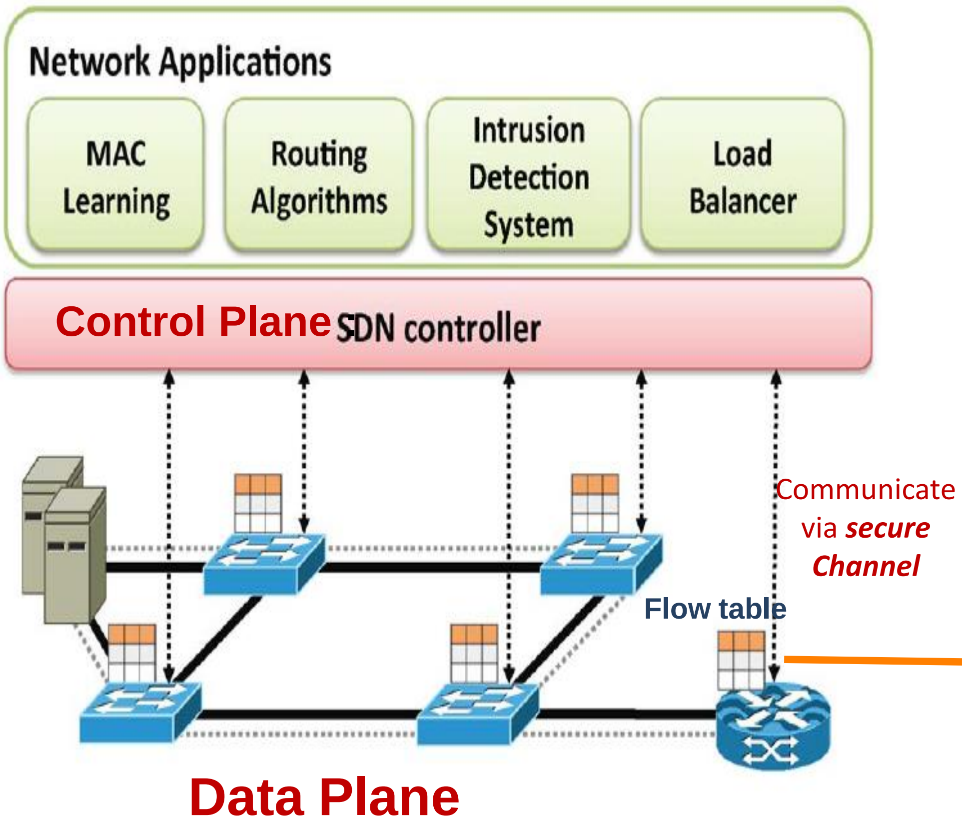
**Abstraction**
- Decouple:
  - Hardware & Software
  - Control plane & forwarding
  - Physical & logical config.

SDN Concept

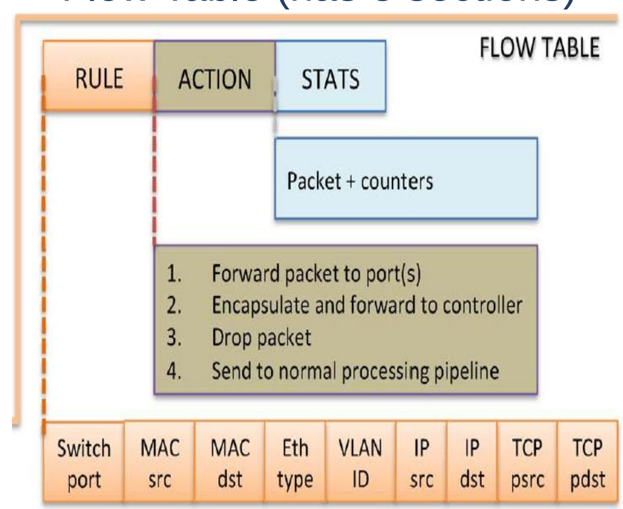


# Basic OpenFlow: How Does it Work?

- Controller manages the traffic (network flows) by manipulating the flow table at switches.
  - Instructions are stored in flow tables.
- When packet arrives at switch, match the header fields with flow entries in a flow table.
- If any entry matches, performs indicated actions and update the counters.
- If Does not match, Switch asks controller by sending a message with the packet header.




Flow Table (has 3 sections)




Match the packet

# Key Elements

- Centralized Network Controller
- Programmable OpenAPIs
- Standard Communication Interface

OPENFLOW – Open Network Foundations

# Why We need SDN?

- **Virtualization:** Use network resource without worrying about where it is physically located, how much it is, how it is organized, etc.

- **Orchestration:** Should be able to control and manage thousands of devices with one command.

- **Programmable:** Should be able to change behavior on the fly.

- **Dynamic Scaling:** Should be able to change size, quantity

- **Automation**: To lower OpEx minimize manual involvement

Troubleshooting

Reduce downtime  Policy enforcement

Provisioning/Re-provisioning/Segmentation of resources

Add new workloads, sites, devices, and resources

# Why We need SDN?

- Visibility: Monitor resources, connectivity
- Performance: Optimize network device utilization

Traffic engineering/Bandwidth management

Capacity optimization

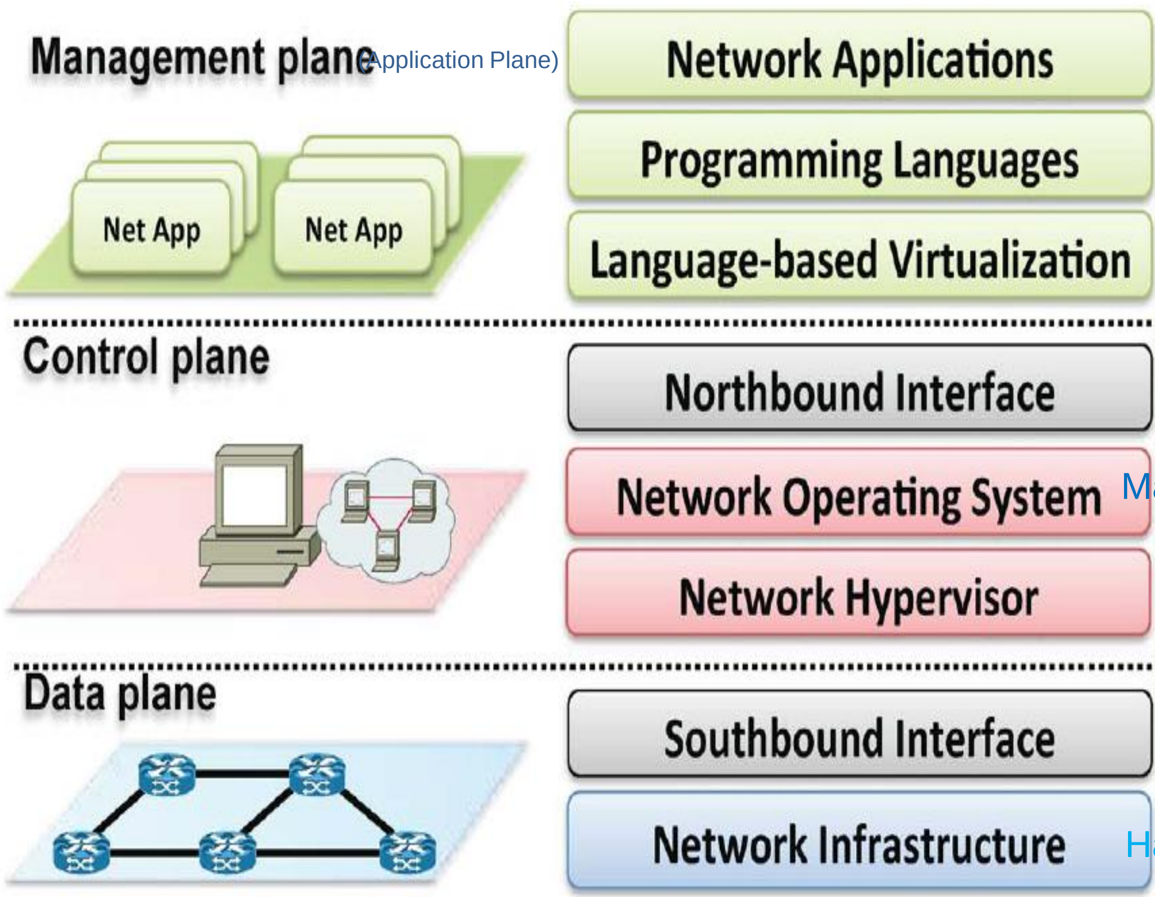
Load balancing  High utilization

Fast failure handling

- Multi-tenancy: Tenants need complete control over their addresses, topology, and routing, security
- Service Integration: Load balancers, firewalls, Intrusion Detection Systems (IDS), provisioned on demand and placed appropriately on the traffic path
- Openness: Full choice of “How” mechanisms

# OpenFlow: More Details

SDN Concept | Different layers in OpenFlow | Discussed




Routing, load balancers, security, et

Make decisions and instructions

Firmware handling instructions from control plane (e.g Open Vswitch) via flow tables.

Hardware (switches)

# Key ideas of SDN

- Dynamic programmability in forwarding packets.
- Decoupling control and data plane.
- Global view network by logical centralization in control plane.
- Applications can be implemented on top of the control plane.
- SDN is a concept to manage network that leverages OpenFlow protocols.

# Summary

- Virtualization
- Types
- VM Migration
- Virtual Networks
- NAT
- SDN