ARRIS

Virtualized CCAP

September 2017

The ARRIS product roadmaps contained herein are for discussion purposes only to demonstrate our thoughts behind the evolutionary development of the ARRIS product offerings. ARRIS is not obligated to develop the software or hardware with the features and functionality discussed in these materials

Next Generation Cable Access

What’s Coming In The Access Network?

Technological Bifurcation

RPHY/
RMACPHY

Centralized vs Distributed

Exponential Bandwidth Growth

Architectural Bifurcation

Fiber Deeper?

MSO Decisions, Decisions, Decisions

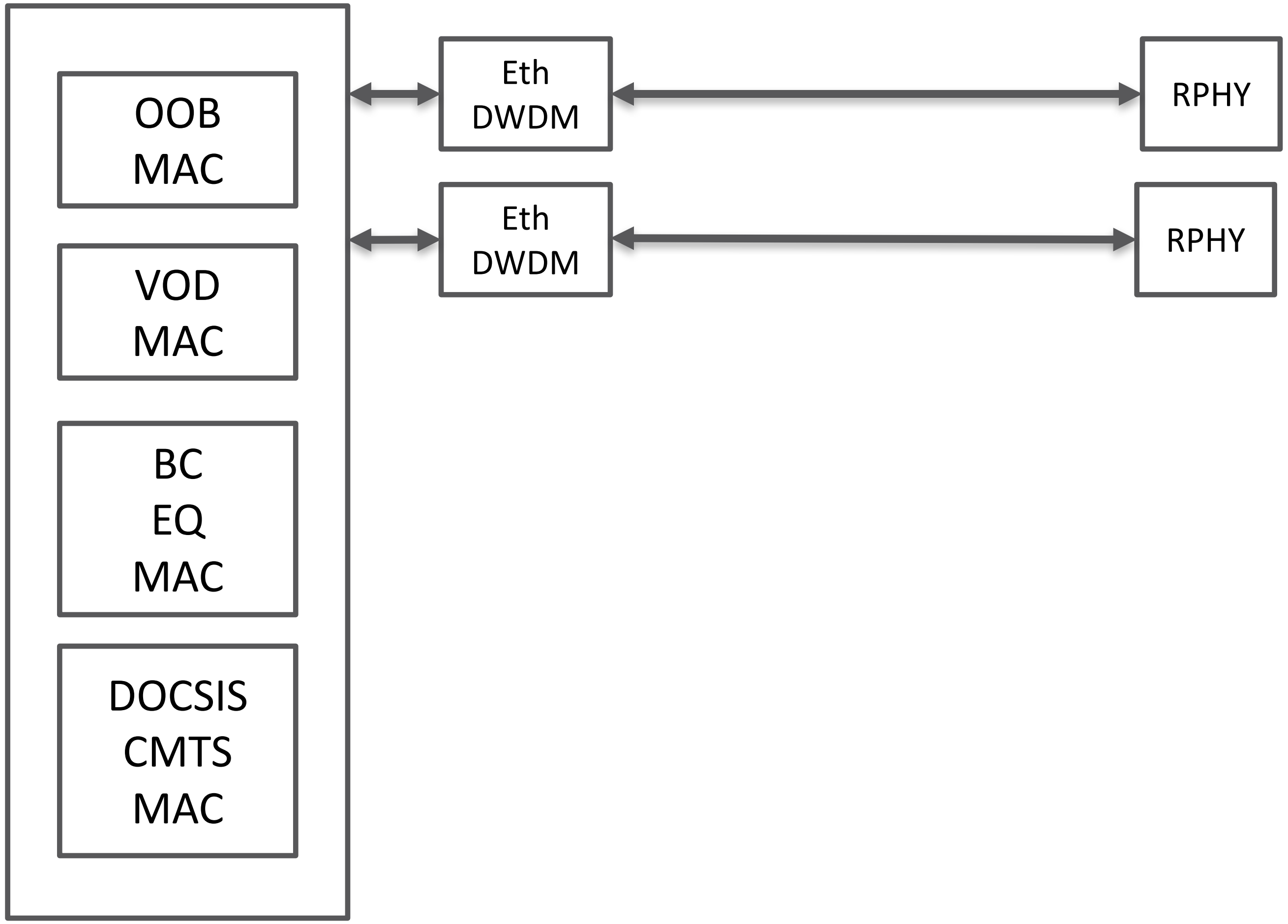
DOCSIS/EPON/GPON/Wireless

Revolution vs Evolution

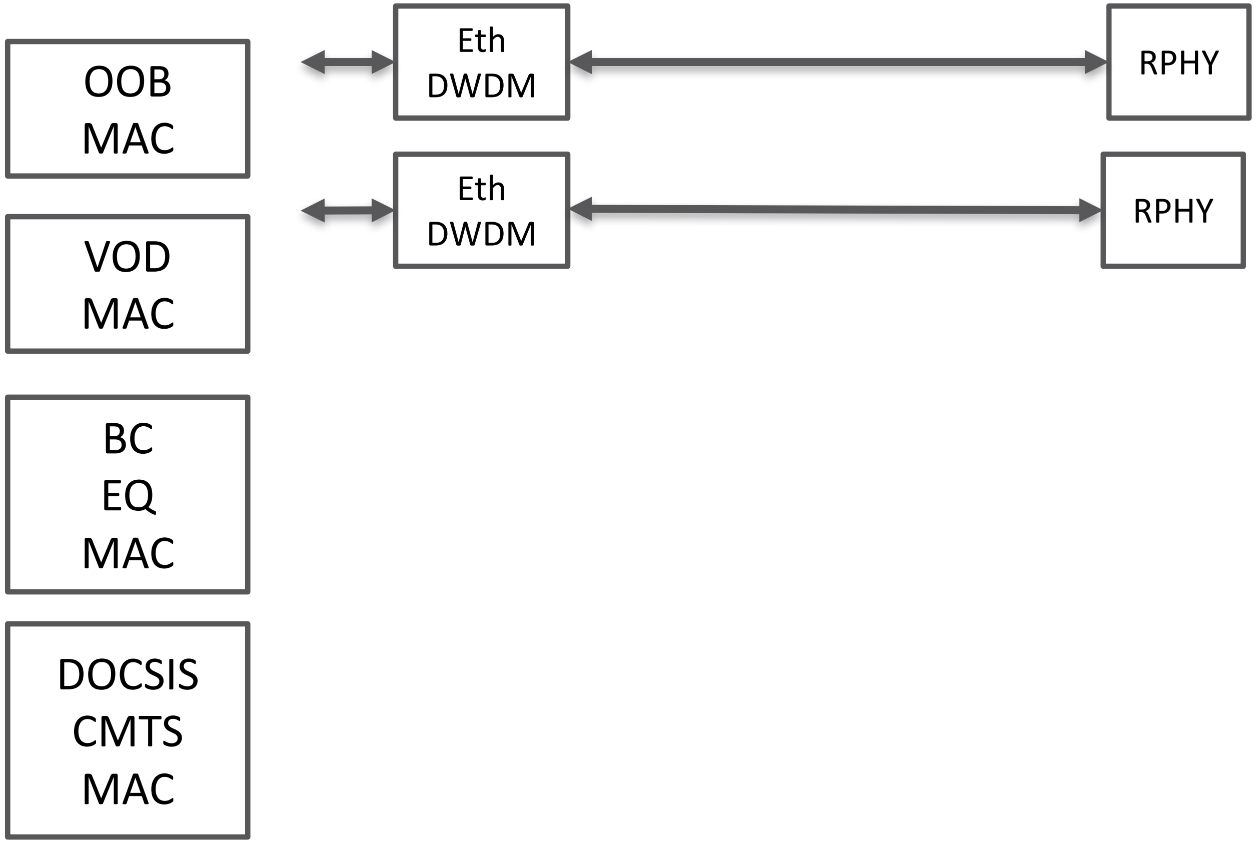
Virtual vs Appliance-based

Increased Competition

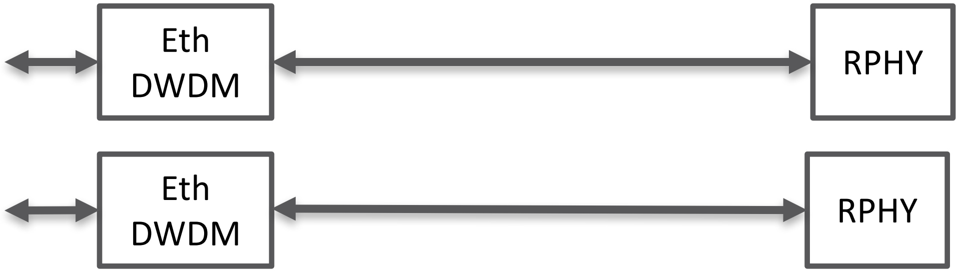
I-Core HFC Access Network Architecture (2nd Evolution) – No Analog Video!



vCMTS is Just the iCORE Implemented in Software!



It Looks Like This and Needs Lots of Help!



- The i-Core does a lot of things that we take for granted like:
  - Handle scaling and failure recovery
  - Some networking and routing functions
  - OOB, encryption, and video

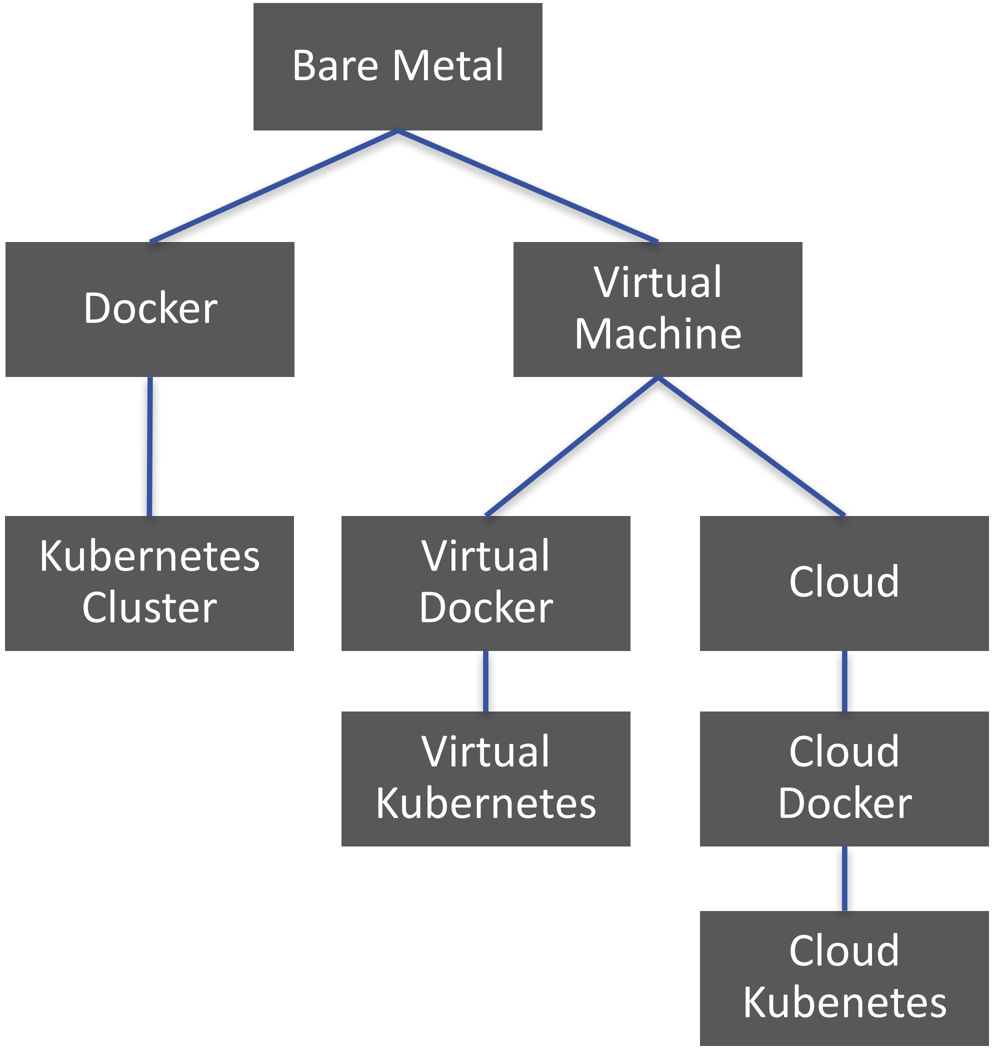
DOCSIS
CMTS
MAC

Virtualization Background

Containers, Clusters, and Clouds

Agenda

- Why?
- Terms
- Computer (bare metal)
- Virtual Machine
- Docker/Containers on Bare Metal
- Docker on a Virtual Machine
- Kubernetes Cluster on Bare Metal
- The Cloud



Why Containers and Clouds?

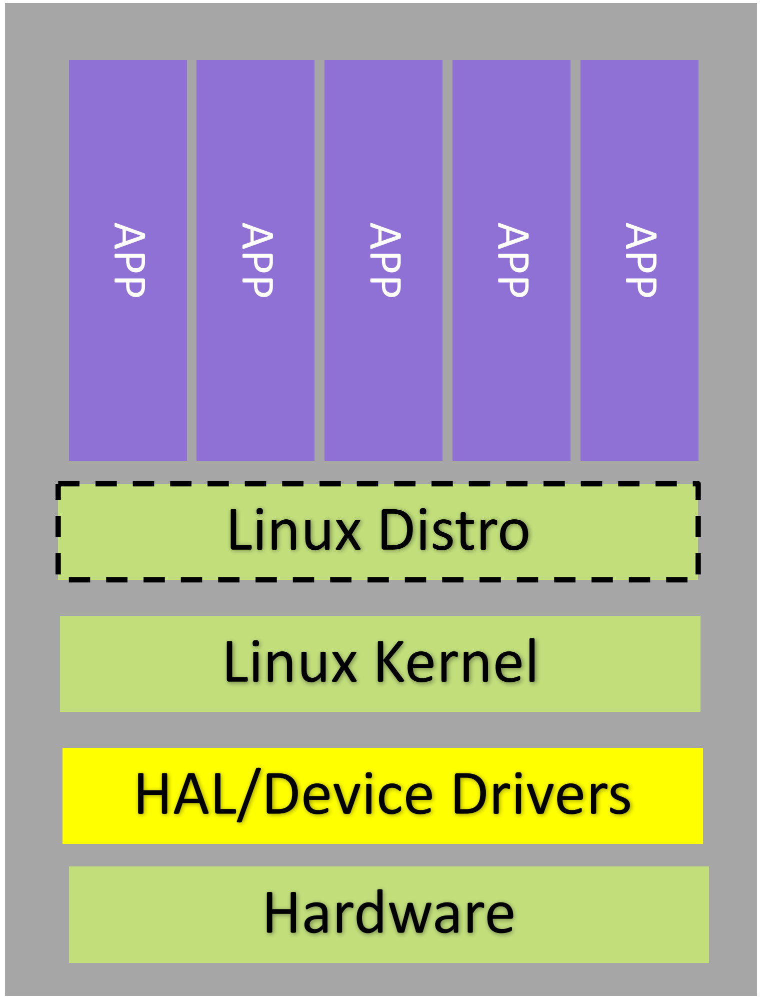
- Bare metal is hard to configure, manage, and maintain

- Clouds and clusters provide scalability and the ability distribute applications geographically
- Virtualization provides hardware independence
- Containers offer clean and simple application and micro service packaging and management.

These technologies help to make software more manageable, scalable, and resilient

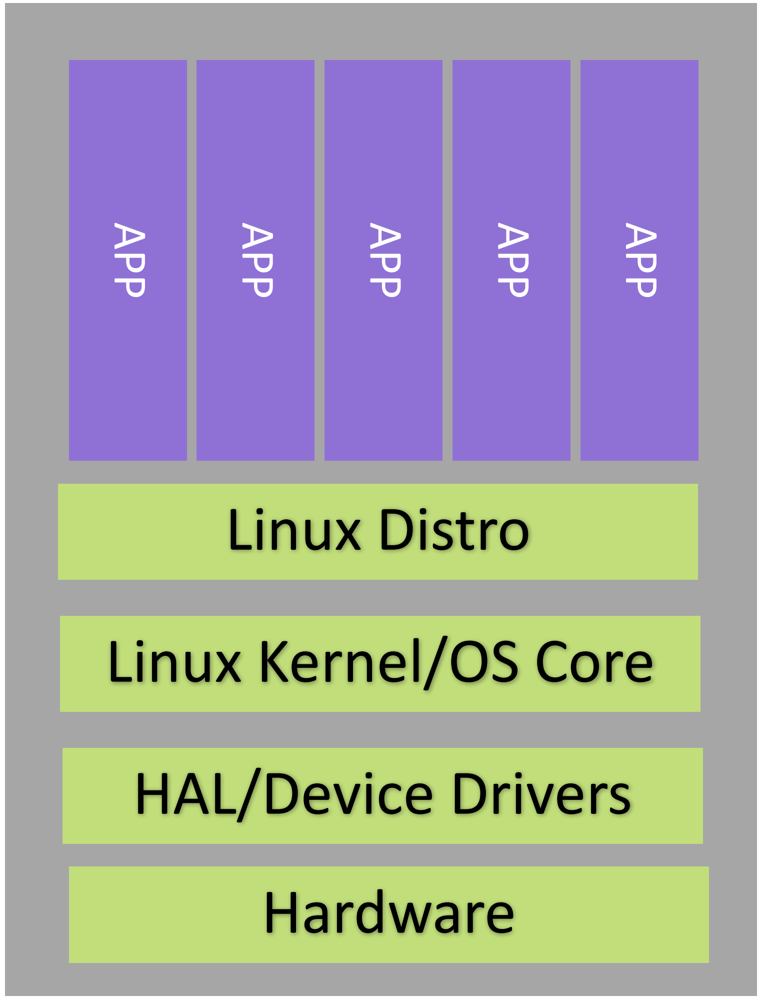
Key

- Underlined terms are new.
- Boxes with a yellow background are new
- Boxes with a dashed border may not be present

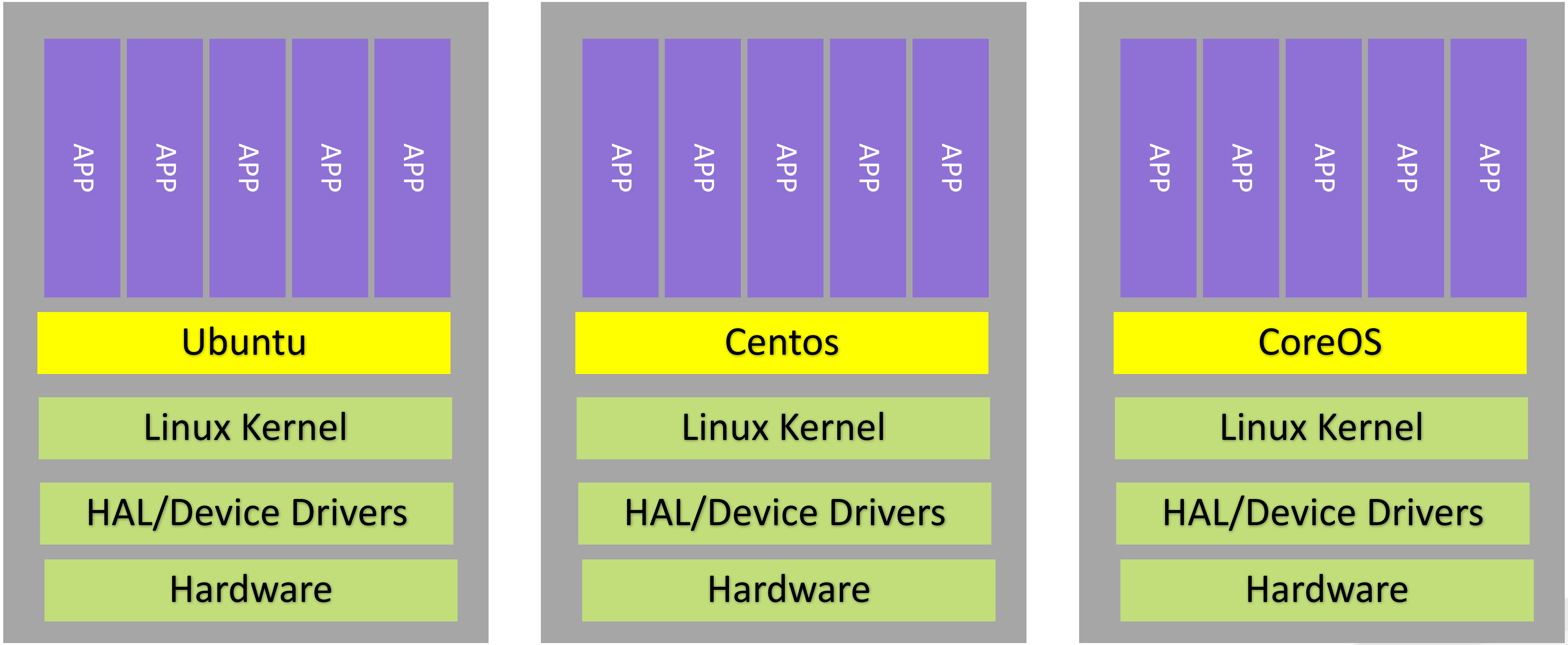


Computer (bare metal)

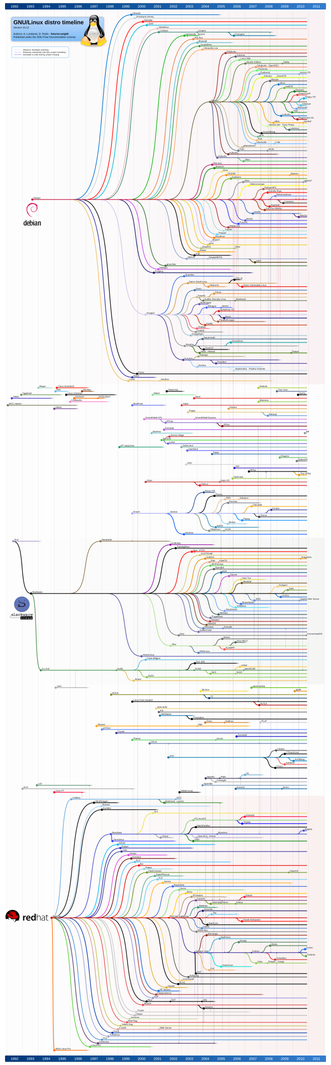
- Bare metal means a regular old computer
- Terms
  - **App:** Any Linux based application or service. Examples include Apache, Nginx, nDVR recorder, VDE, or vCMTS.
  - **Linux Distro:** The Linux distribution. For example, Ubuntu, Centos, Debian, CoreOS, or Yocto
  - **Linux Kernel:** The **Core** of the operating system. Invariant part of Linux. Provides basic O/S services: memory management, CPU/process management, file systems, storage, etc.
  - **HAL/Device Drivers**: Hardware abstraction layer. Allows the kernel to interact hardware in a common way.
  - **Hardware:** Physical devices, for example, the CPU, memory, storage, USB, etc.

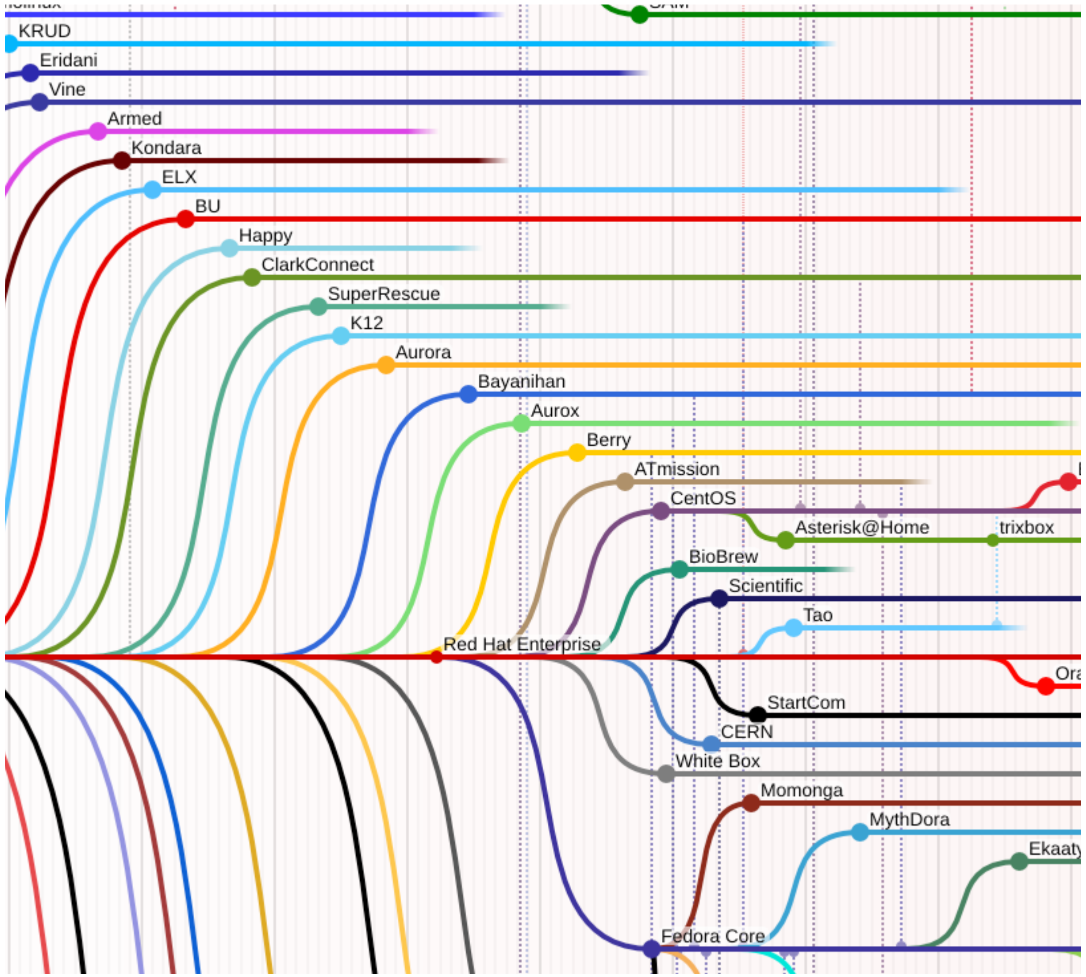


Linux Distros



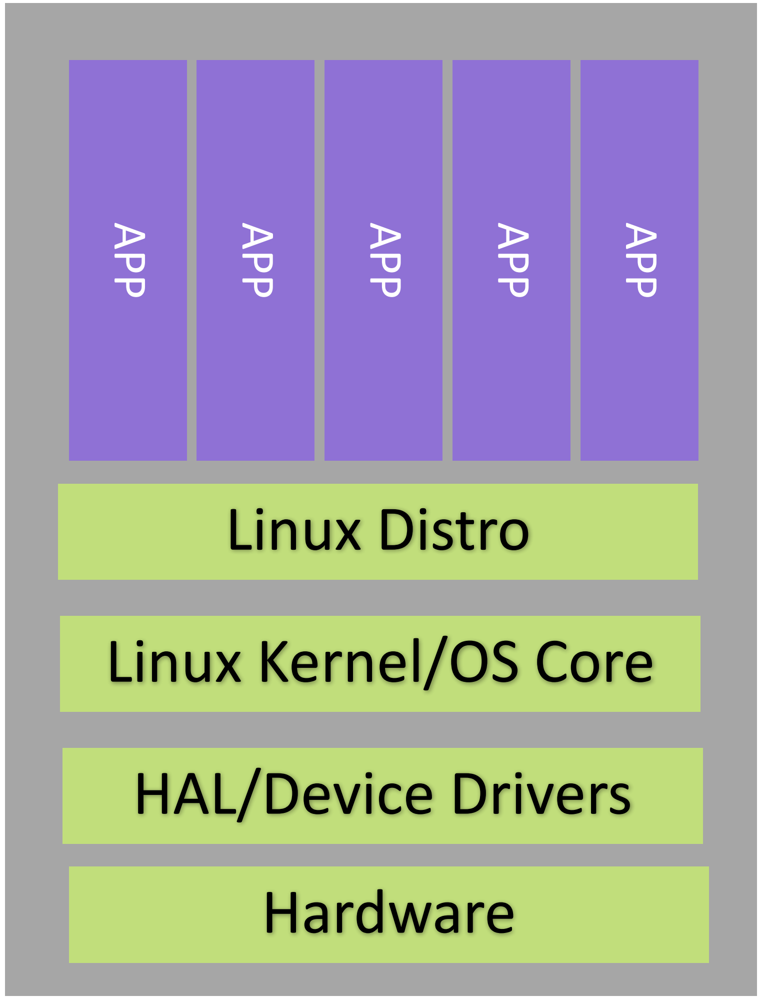
Linux Distros





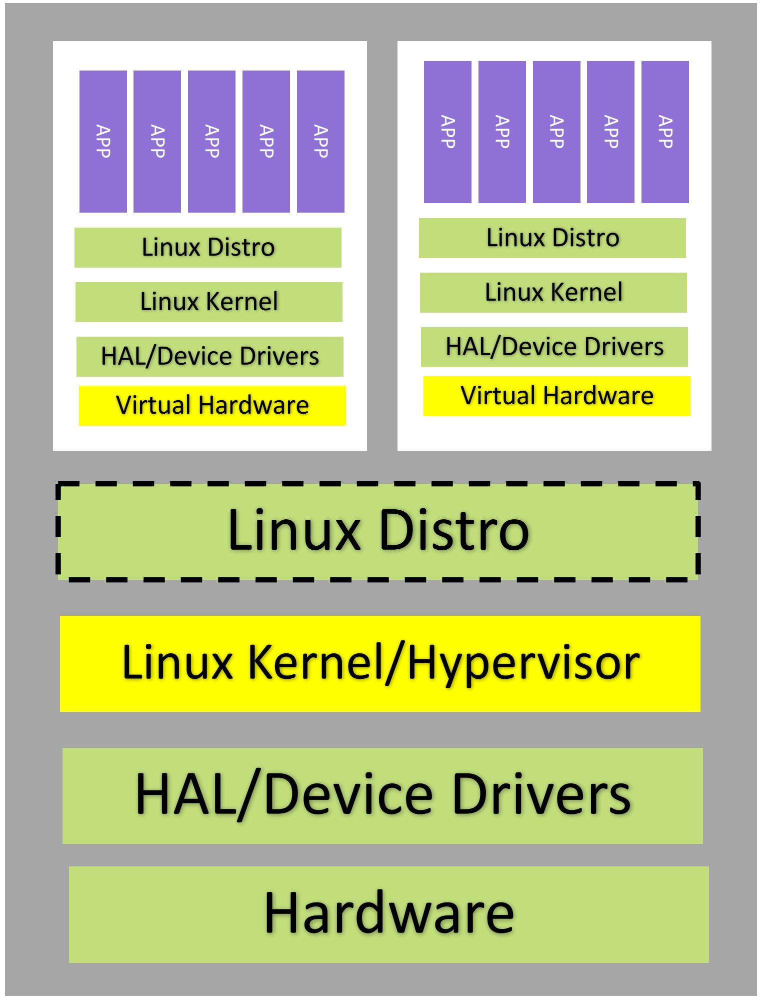
Computer (bare metal)

- Bare metal means a regular old computer
- Terms
  - **App:** Any Linux based application or service. Examples include Apache, Nginx, nDVR recorder, VDE, or vCMTS.
  - **Linux Distro:** The Linux distribution. For example, Ubuntu, Centos, Debian, CoreOS, or Yocto
  - **Linux Kernel:** The **Core** of the operating system. Invariant part of Linux. Provides basic O/S services: memory management, CPU/process management, file systems, storage, etc.
  - **HAL/Device Drivers**: Hardware abstraction layer. Allows the kernel to interact hardware in a common way.
  - **Hardware:** Physical devices, for example, the CPU, memory, storage, USB, etc.



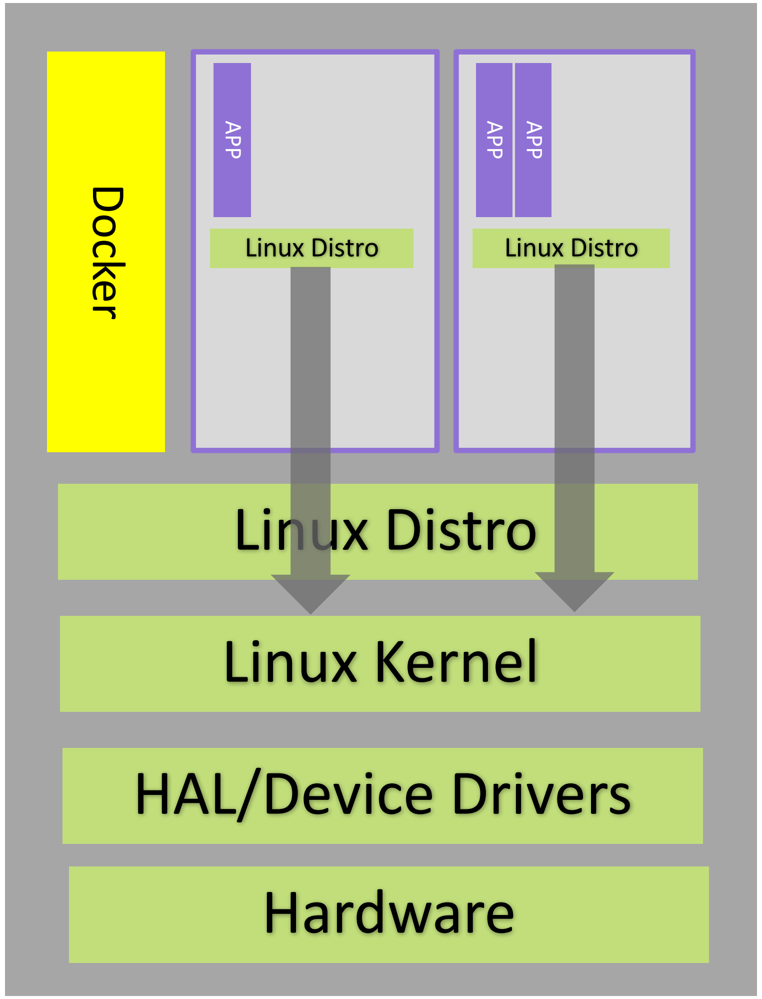
Virtual Machine

- This picture shows **one computer** running virtual machine software
- Brands: KVM, VMware, Xen, etc.
- Terms
  - **App:** Any Linux based application or service. Examples include Apache, Nginx, nDVR recorder, VDE, or vCMTS.
  - **Linux Distro:** The Linux distribution. For example, Ubuntu, Centos, Debian, CoreOS, or Yocto
  - **Linux Kernel:** Invariant part of Linux. Provides basic O/S services: memory management, CPU/process management, file systems, storage, etc.
  - **Hypervisor:** A virtual machine monitor. It may be part of Linux (kvm) or proprietary (VMware)
  - **HAL/Device Drivers**: Hardware abstraction layer. Allows the kernel to interact hardware in a common way.
  - **Virtual Hardware:** Hardware simulated in software.
  - **Hardware:** Physical devices, for example the CPU, memory, storage, USB, etc.
- Linux Distro (with dashed border) may not be present if a non-Linux based hypervisor is used.



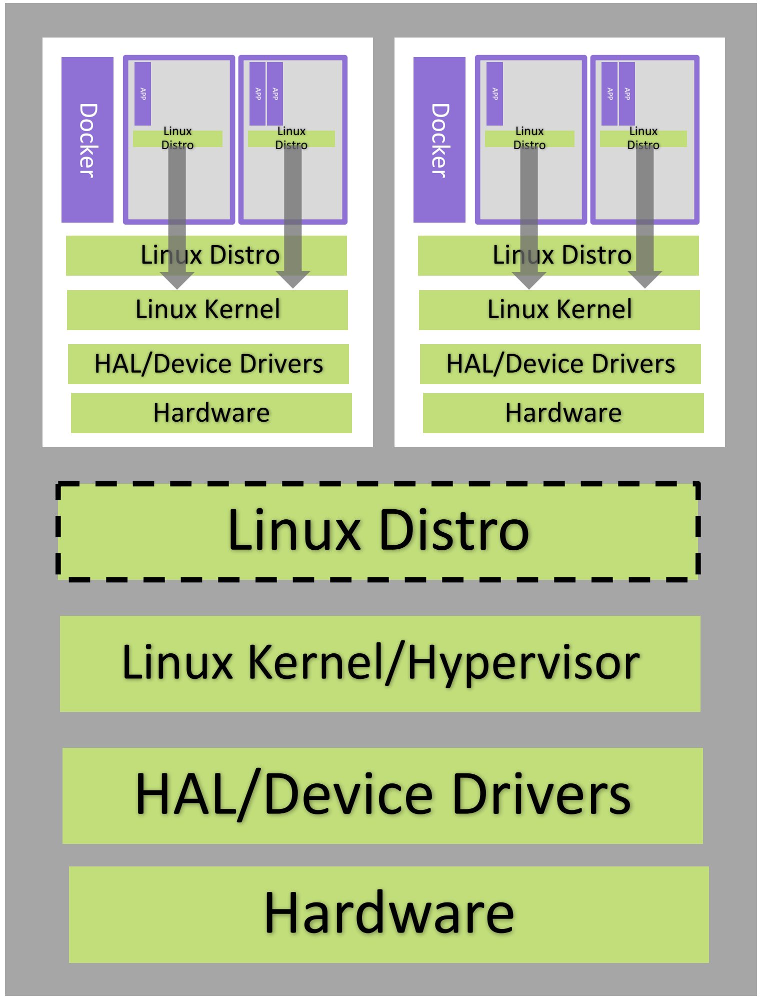
Docker/Containers on Bare Metal

- Docker is software that runs on top of Linux
- Brands: Docker, Rocket
- Terms
  - **Docker:** Container management software.
  - **App:** Any Linux based application or service. **Linux Distro:** The Linux distribution. For example, Ubuntu, Centos, Debian, CoreOS or Yocto
  - **Linux Kernel:** Invariant part of Linux. Provides basic O/S services: memory management, CPU/process management, file systems, storage, etc.
  - **HAL/Device Drivers**: Hardware abstraction layer. Allows the kernel to interact hardware in a common way.
  - **Hardware:** Physical devices, for example the CPU, memory, storage, USB, etc.



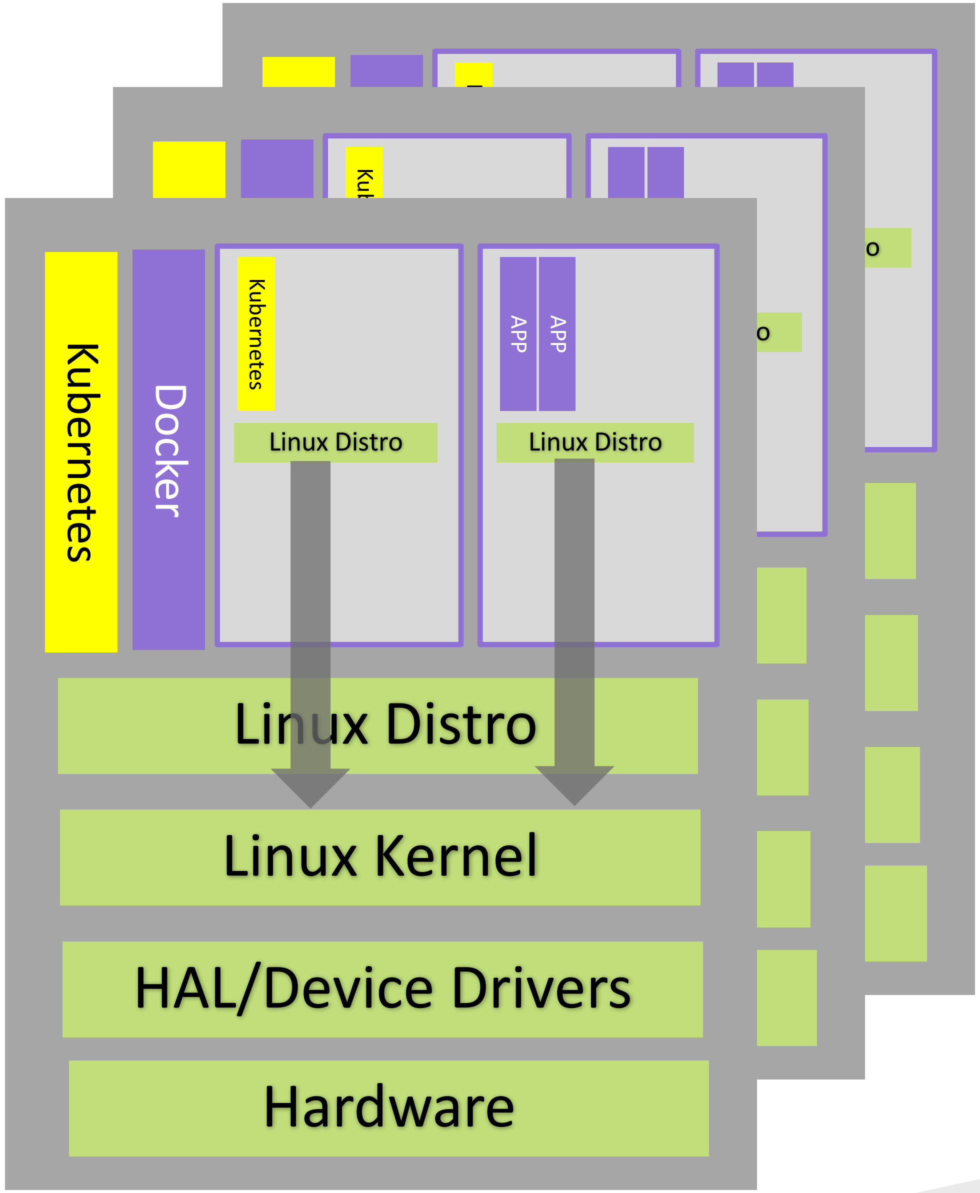
Docker on a Virtual Machine

- This is the Virtual Machine + Docker
- Terms
  - **App:** Any Linux based application or service. Examples include Apache, Nginx, nDVR recorder, VDE, vCMTS.
  - **Docker:** Container management software.
  - **Linux Distro:** The Linux distribution. For example, Ubuntu, Centos, Debian, CoreOS, or Yocto.
  - **Linux Kernel:** Invariant part of Linux. Provides basic O/S services: memory management, CPU/process management, file systems, storage, etc.
  - **Hypervisor:** A virtual machine monitor. It may be part of Linux (kvm) or proprietary (Vmware)
  - **HAL/Device Drivers**: Hardware abstraction layer. Allows the kernel to interact hardware in a common way.
  - **Virtual Hardware:** Hardware simulated in software.
  - **Hardware:** Physical devices, for example the CPU, memory, storage, USB, etc.
- Linux Distro (with dashed border) may not be present if a non-Linux based hypervisor is used.



Kubernetes Cluster on Bare Metal

- Kubernetes allow you to manage a cluster of Docker machines
- Docker is software that runs on top of Linux
- Terms
  - **Kubernetes**: Container cluster management system.
  - **Docker:** Container management software.
  - **App:** Any Linux based application or service. **Linux Distro:** The Linux distribution. For example, Ubuntu, Centos, Debian, CoreOS, or Yocto
  - **Linux Kernel:** Invariant part of Linux. Provides basic O/S services: memory management, CPU/process management, file systems, storage, etc.
  - **HAL/Device Drivers**: Hardware abstraction layer. Allows the kernel to interact hardware in a common way.
  - **Hardware:** Physical devices, for example the CPU, memory, storage, USB, etc.



Cloud (e.g. Open Stack)

- Manages a cluster of virtual machines and computing recourses
- Terms
  - VM – virtual machine
  - **Hypervisor:** A virtual machine monitor. May be part of Linux (kvm) or proprietary (Vmware)
  - **Block Storage:** A storage system that access information in “blocks.” This is basically a traditional file system. A file is made up of a series of blocks. Random access is allowed.
  - **Object Store:** (e.g. Swift or CEPH) A storage system that accesses information at the “object” level. (e.g. the WHOLE file). Random access is not allowed.

Kubernetes in the Cloud

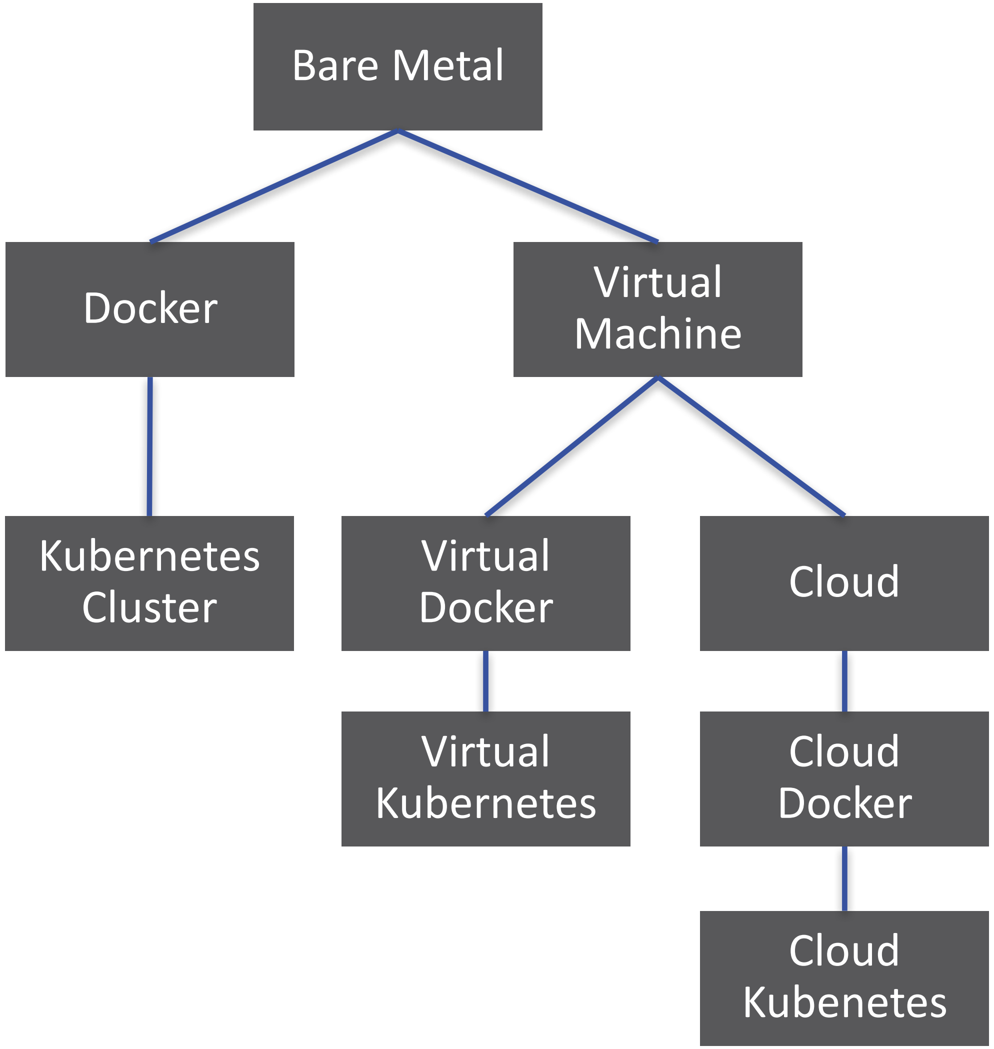
- Left as an exercise for the reader

Commercial Clouds

- Amazon Web Services (AWS)
- Microsoft Azure
- Google Cloud Platform

Take Away Concepts

- Virtual Machine technology allows “any operating system” to run on a computer
  - Examples include KVM, VMware, Xen
- Container technologies is a lightweight alternative to virtual machine software, but is constrained to a single Linux kernel.
  - Examples include Docker and rkt (Rocket).
- A Cluster or Cloud is a collection of computer that a managed as a common resource.
  - For virtual machines examples include OpenStack and VSphere
  - For Containers examples include Kubernetes and Docker Storm



SDN & NFV

What is SDN and NFV?

- **SDN separates data and control and management plane to enable:**
  - A software programmable network
  - A centralized controller with dynamic management and provisioning
  - Reuse of parity control features over multiple access technologies
  - Dynamic creation, modification and deletion of services

- **NFV decouples SW from HW to enable:**
  - Using COTS hardware and open software
  - Dynamic resource and service management
  - Reuse of parity forwarding features over multiple access technologies
  - Defining efficient network and service chains

What are the Drivers behind SDN/NFV? (1 of 2)

- **Elasticity and Scalability Enable by System Modularity (Disaggregation of CCAP & OLT)**
  - Disaggregation: Separation of Hardware and Software Functional Blocks
    - (PHY, MAC/PHY, Traffic Management/Service Gateway, Switch Fabric/Backplane, Control, WAN/NSI Link)
  - Separation of Control and Data Planes
  - Separating the network functions allows placement of capacity where and when needed
    - (IO capacity for access layer like PON, P2P Ethernet, DOCSIS, etc. where and when needed)

- **Agility Enabling Time To Market**
  - Service Creation and Provisioning Automation
  - End-to-end Analytics assists in determining resource service capacity
  - Consistent services and features across vendors and access technologies (DOCSIS, PON, Ethernet, Wireless)

- **Open Platform Ecosystem**
  - No Proprietary System Vendor software or protocols
  - System Vendor, Network Operators, and Community develop features / applications
  - Automate OAM&P
  - Interoperability
  - Innovation

What are the Drivers behind SDN/NFV? (2 of 2)

- **Reduce CAPEX**
  - COTS switches, servers, storage, compute elements
  - Reduced headend estate, power and cooling
  - Ability to scale per demand and integrate only needed functions
  - New pricing options using licenses per subscriber, throughput, features enable (pay-as-you-grow)
  - Open Software platforms from multiple vendors
- **Reduce OPEX**
  - Reduced headend power and cooling
  - Zero touch provisioning and programmable networks and services with reduced complexity
  - End-to-end visibility, analytics and service assurance orchestration
  - Dynamic and efficient resource management (self healing/optimizing networks)
  - Standard APIs, control and management interfaces
- **Agile Services and New Business Models**
  - Abstracted service models and automated networks for shortened service integration
  - Enhanced QoE and customer and business portals
  - Cloud based applications and abstracted service models for business to business services
  - Centralized control of distributed functions directed and composed by service specific requirements

Kubernetes

Kubernetes

- Kubernetes is a robust open source software platform for managing the deployment, scaling and life cycle of application containers

- Kubernetes is a field-proven orchestration solution for very large scale systems comprised of huge numbers of containers

- Kubernetes features
  - Automatically places containers on nodes based on their identified resource requirements and constraints
  - Provides high availability by monitoring nodes to detect failures and re-loading containers onto new nodes
  - Delivers both rolling upgrades and rollback for applications while ensuring continuous up-time operation

- Kubernetes often shortened to be “K8s” in literature – K followed by 8 letters and an s.

Kubernetes Concepts

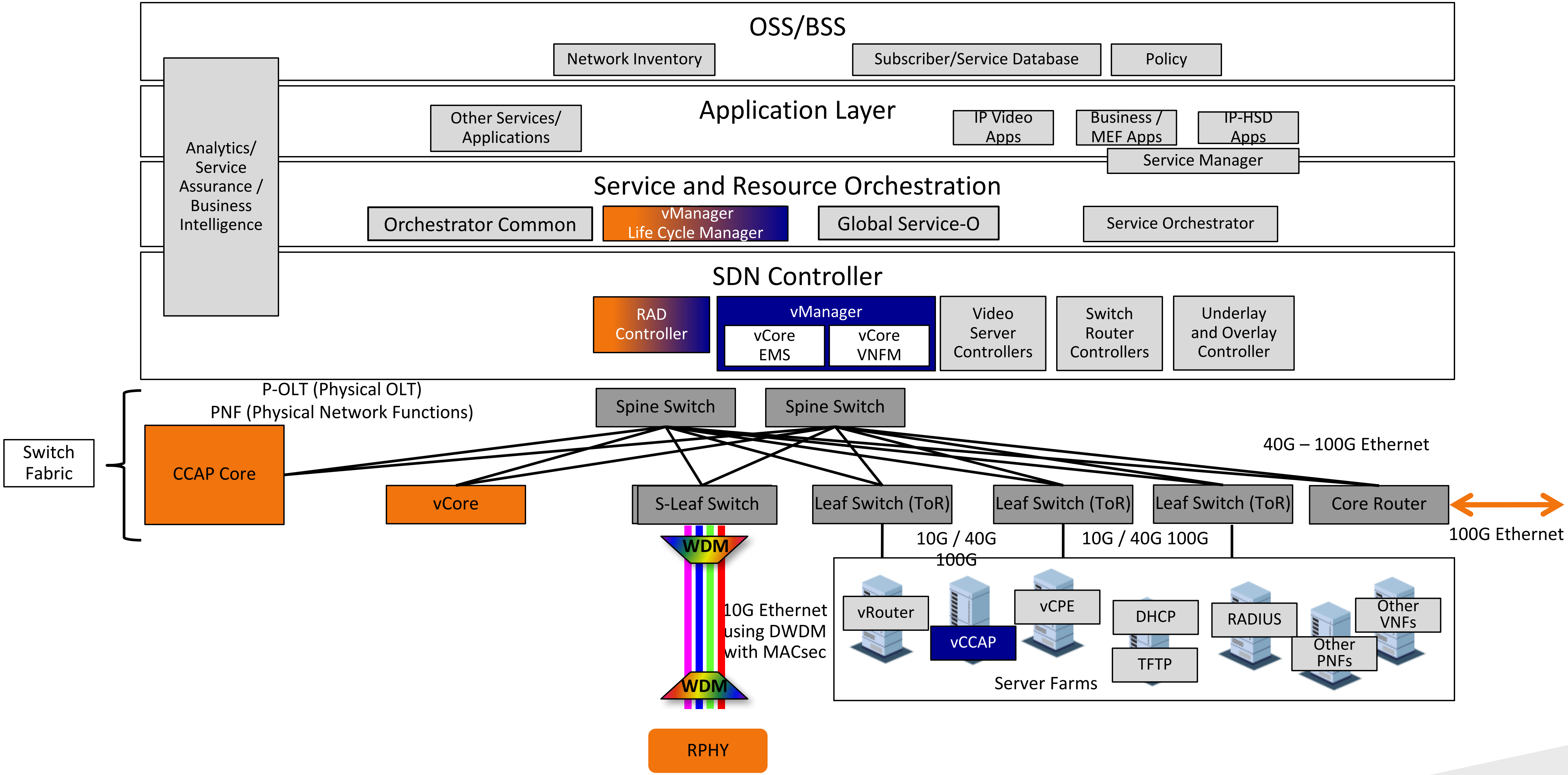
- **Node** = a physical or virtual machine where application containers may be run
- **Pod** = a set of one or more application containers that are managed collectively as a group
- **Kubelet** = the Kubernetes agent that runs on a Node and ensures that the Pods (application containers) that should be present on the Node are running and healthy
- **Label** = a semantic tag used to identify Pod characteristics and control how Pods are mapped onto Nodes
  - May label a Pod with a name and a version number to easily determine its role and specific application versions running in the Pod
  - Can label a Node to identify key characteristics that may be required to run certain Pods and then label the Pod to only run on matching Nodes
- **Service** = a set of one or more Pods that can be reliably accessed externally to provide a function to other Pods
  - By default, as Pods are created and destroyed their IP addresses may change
  - Identifying a Pod as part of a Service ensures that it can be consistently accessed at all times

Pods

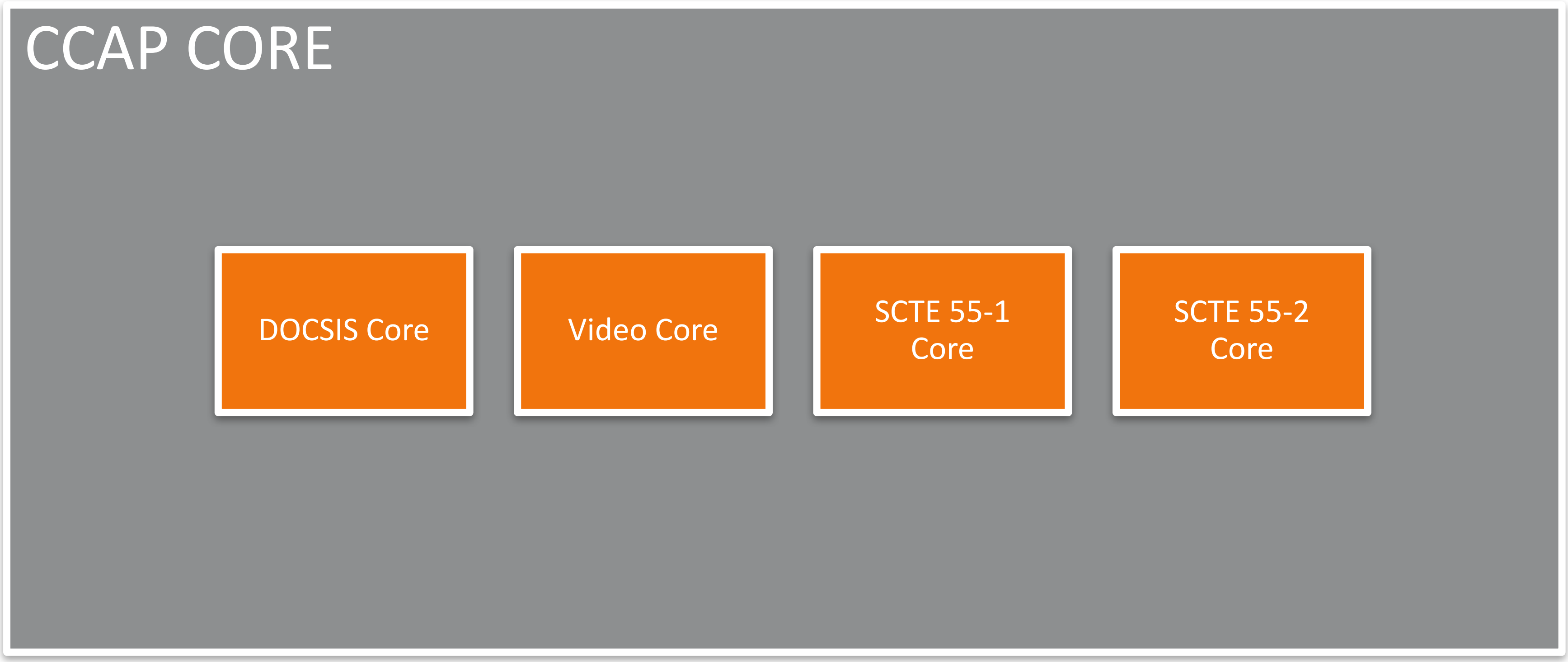
- A Kubernetes pod is a group of one or more containers which are co-located on a group of cores on the same machine, can share resources and have a unique IP address.
- The primary reason to decompose into containers as much as possible is for modularity
  - A container can be swapped out with a different version
  - A container can be reused in a different pod – an independent reusable component
- Separate containers can share resources like memory
- A pod is the atomic unit of scheduling in a Kubernetes cluster.
- A pod has a set of data volumes whose life is longer than that of the containers.
  - Anything in the container is transient.
  - Persistent data must be in external Network Attached Storage.
- A namespace is shared among all the containers in a pod to connect them together via a localhost to make configuration much simpler. Different containers within a pod can all see each other on localhost because they share the same network namespace.

Putting It All Together

High-Level MSO SDN/NFV Architecture for 10G EPON



Logical Functions in a CCAP CORE



Disaggregated Functions in a vCORE

- A vCMTS is only part of the solution

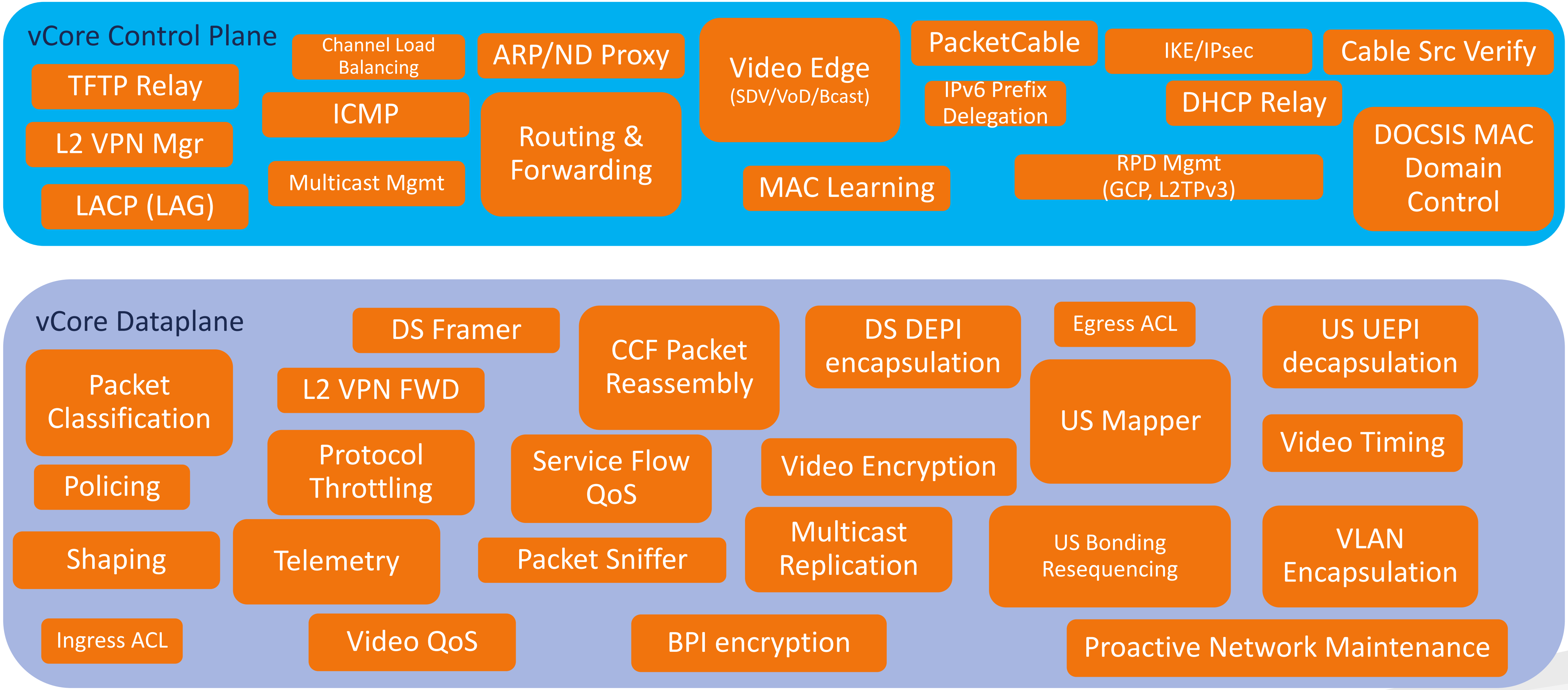
vCore

Soft Video Core

Soft SCTE 55-1 Core

Soft SCTE 55-2 Core

vCore Architecture: Combined data and control planes



vCore Control Plane

TFTP Relay

L2 VPN Mgr

LACP (LAG)

Channel Load Balancing

ICMP

Multicast Mgmt

ARP/ND Proxy

Routing & Forwarding

Video Edge (SDV/VoD/Bcast)

MAC Learning

PacketCable

IPv6 Prefix Delegation

IKE/IPsec

DHCP Relay

RPD Mgmt (GCP, L2TPv3)

Cable Src Verify

DOCSIS MAC Domain Control

vCore Dataplane

Packet Classification

Policing

Shaping

Ingress ACL

DS Framer

L2 VPN FWD

Protocol Throttling

Telemetry

Video QoS

CCF Packet Reassembly

Service Flow QoS

Packet Sniffer

BPI encryption

DS DEPI encapsulation

Video Encryption

Multicast Replication

Egress ACL

US Mapper

US Bonding Resequencing

US UEPI decapsulation

Video Timing

VLAN Encapsulation

Proactive Network Maintenance

SG mobility

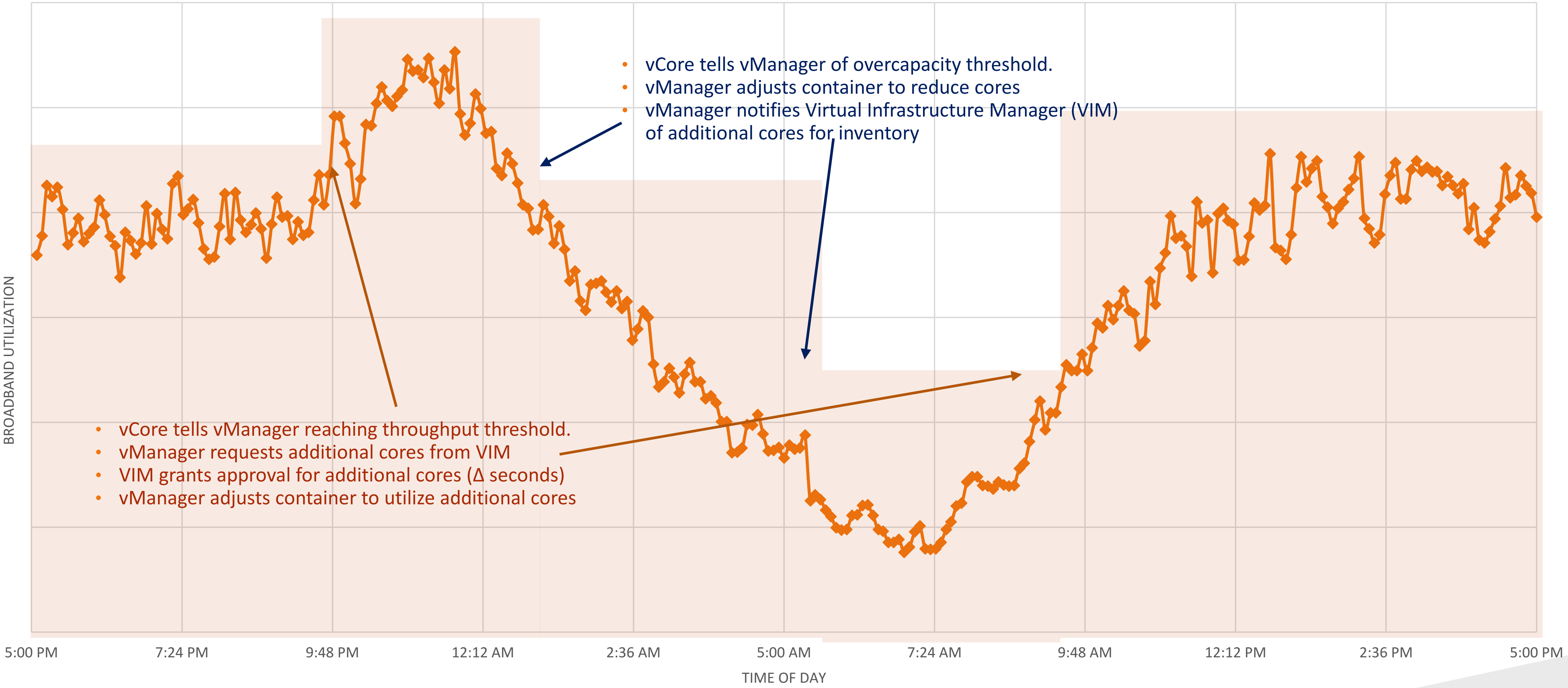
- The number of cores assigned to a container could be reduced significantly during periods of low utilization (e.g. night time).
  - Other workloads could be run on the freed up cores
  - Alternatively, all SGs in a container could be moved to another container on a different server so that the current container (and subsequently the server) could be shut down.
- Moving an SG to a different container is disruptive to in-progress packet flows
  - All SG data needs to be copied into a new container
  - All packet queues associated with the SG need to be drained before the new container takes over
  - The switching over of traffic streams in switches, draining of packet queues for SFs, restarting of scheduling in the new container, etc, is synchronized by vManager
- For redundancy, each SG in a container can be spared by a different POD so the basic mobility unit is an SG

vCore Sparing

- vCore POD sparing consists of a spare vCore Kubernetes POD that is configured to shadow an active vCore Kubernetes POD.
- Each spare vCore Kubernetes POD collects and process checkpoint messages from its active partner.
- This arrangement allows for state and services to be maintained across failures
  - In-transit media packets might be lost for a brief time during an active switch; but all devices remain up and stable
- This arrangement can facilitate:
  - Hitless upgrade: Shut down spare on old release; bring up new spare on new release; allow all checkpoint updates; switch over active vCore from old release to new spare copy on new release; shut down old release and bring up spare of new release in its place; allow all checkpoint updates; optionally, switch active vCore back to original node; shut down formerly active and recreate spare.
  - Load Balancing: Movement of vCores for Load Balancing can be accomplished thru vCore Sparing
  - Energy Management: Movement of vCores for Energy Management can be accomplished thru vCore Sparing
- Spare vCore PODs can be placed on a dedicated node or on nodes shared with active PODs.

The Diurnal Utilization Cycle with vCore scaling

General Trend



- vCore tells vManager of overcapacity threshold.
- vManager adjusts container to reduce cores
- vManager notifies Virtual Infrastructure Manager (VIM) of additional cores for inventory

- vCore tells vManager reaching throughput threshold.
- vManager requests additional cores from VIM
- VIM grants approval for additional cores (Δ seconds)
- vManager adjusts container to utilize additional cores

Thank You!