LOGTEL
It's more than knowledge

Introduction to NFV

Samuel Dratwa
Samuel.dratwa@gmail.com

About the Copyright

Covering Variety of Advanced Technology areas

Software

Telecom

Hardware

Currently:
- Automotive
- LBS
- Smart city

Internet of Things
& related industries

Data Science:
Big data & AI

LOGTEL

Knowledge By All Means

Courses

Nextgen Seminars

Web info

Conferences

Consultation & projects

LOGTEL

Agenda

- Cloud computing
- NFV
  - NFVI (VIM & OpenStack)
  - SDN vs. NFV
  - MANO
  - VNF and (EMS vs. VNFM)
- ETSI vs. at&t ECOMP vs. ONAP vs. ...
- Tacker / TOSCA
- RFP/RFQ analyze
- Thoughts and conclusions

What are we selling ?

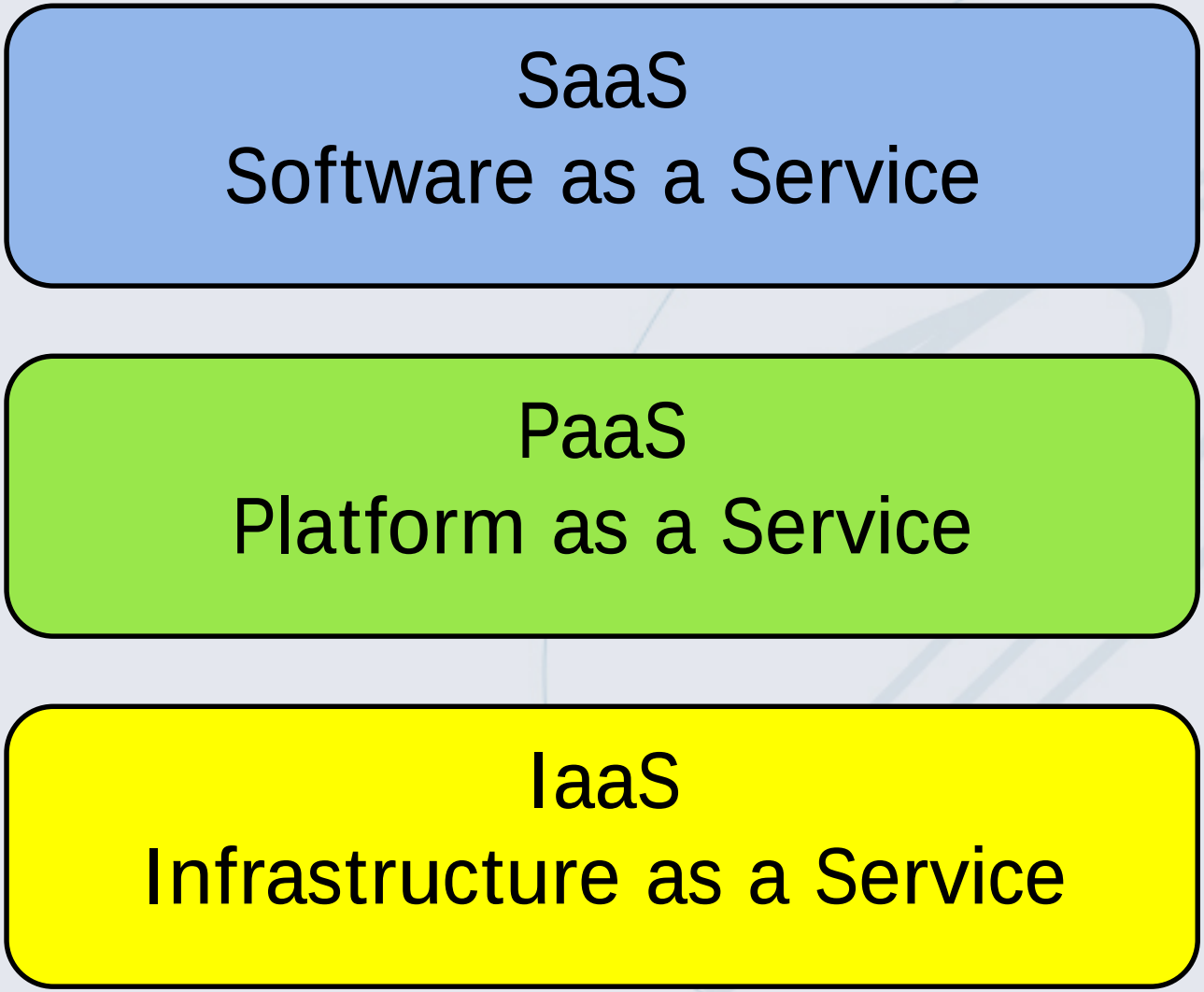
customer satisfaction !

It's all about customer satisfaction

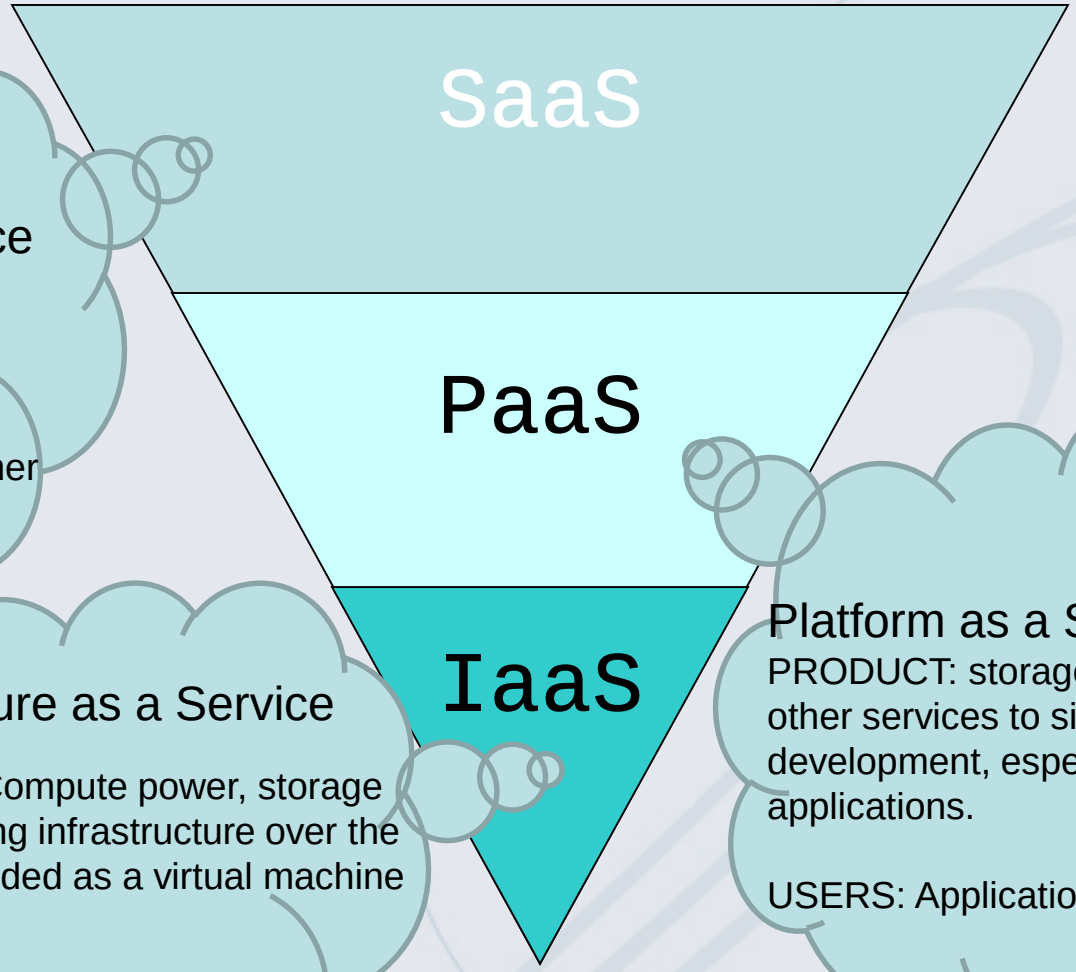
Definition

- *Wikipedia* - a type of Internet-based computing that provides shared computer processing resources and data to computers and other devices on demand. It is a model for enabling ubiquitous, on-demand access to a shared pool of configurable computing resources (computer networks, servers, storage, applications and services),which can be rapidly provisioned and released with minimal management effort.
- *Academic (simple)* - provide on-demand resources or services over a network, often the Internet, with the scale and reliability of a data center.
- *Mine* – distributed mainframe, reachable from any device connected to the internet

Three layers of services



IaaS, PaaS and SaaS Point of View



Software as a Service
PRODUCT: Finished application available on demand to end user

USERS: Software consumer

Infrastructure as a Service

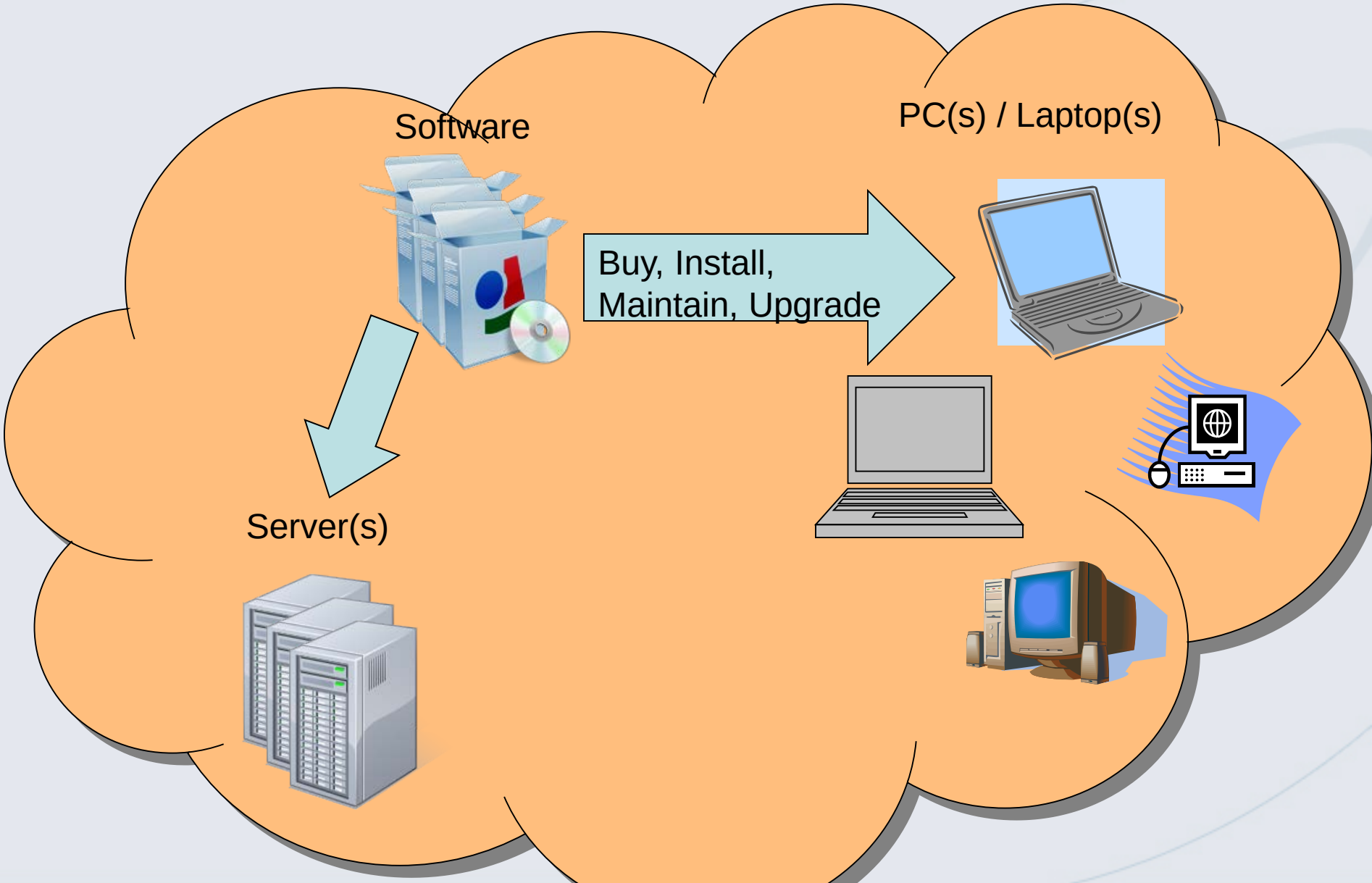
PRODUCT: Compute power, storage and networking infrastructure over the internet, provided as a virtual machine image

USERS: Developers

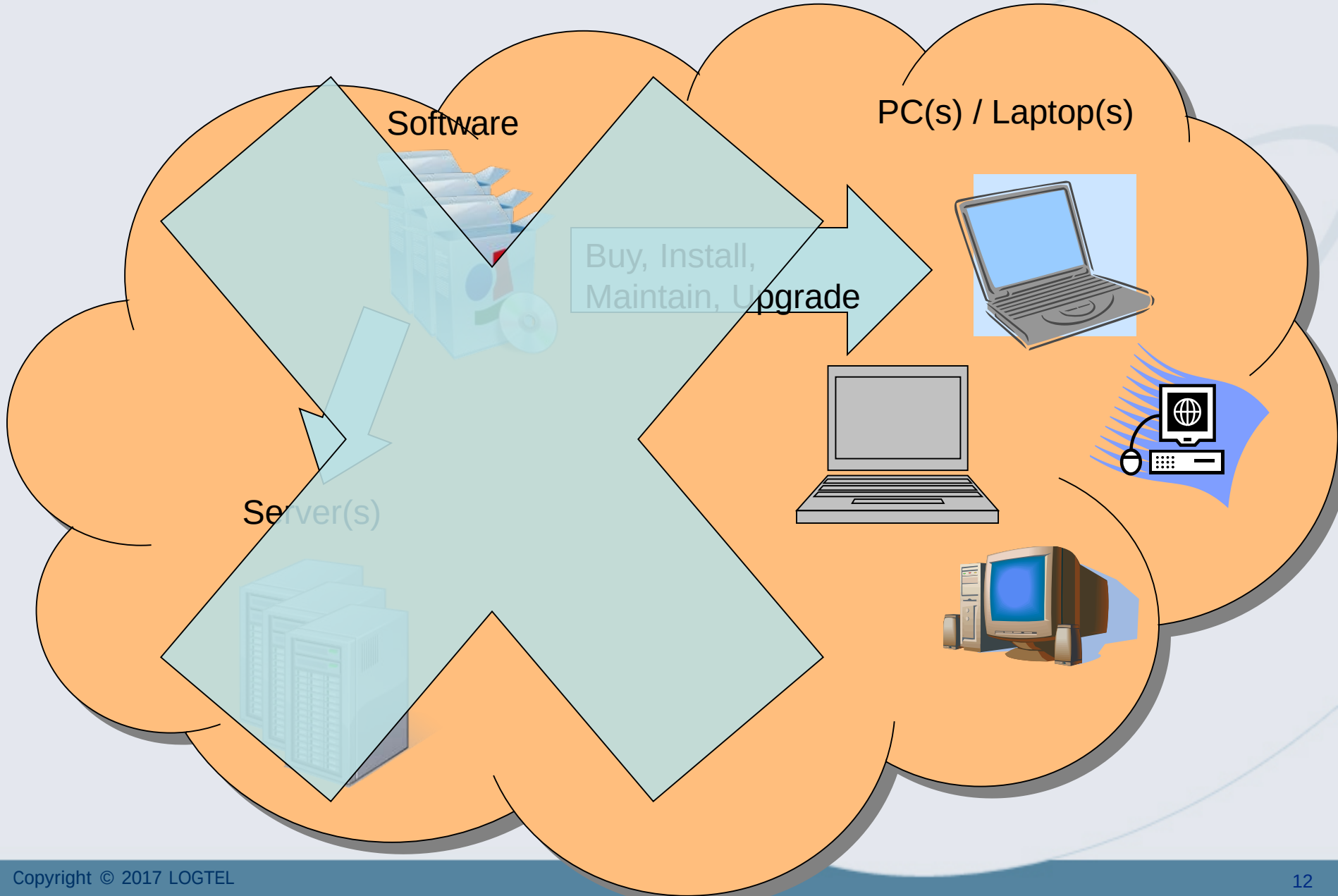
Platform as a Service
PRODUCT: storage, compute and other services to simplify application development, especially of web applications.

USERS: Application Developers

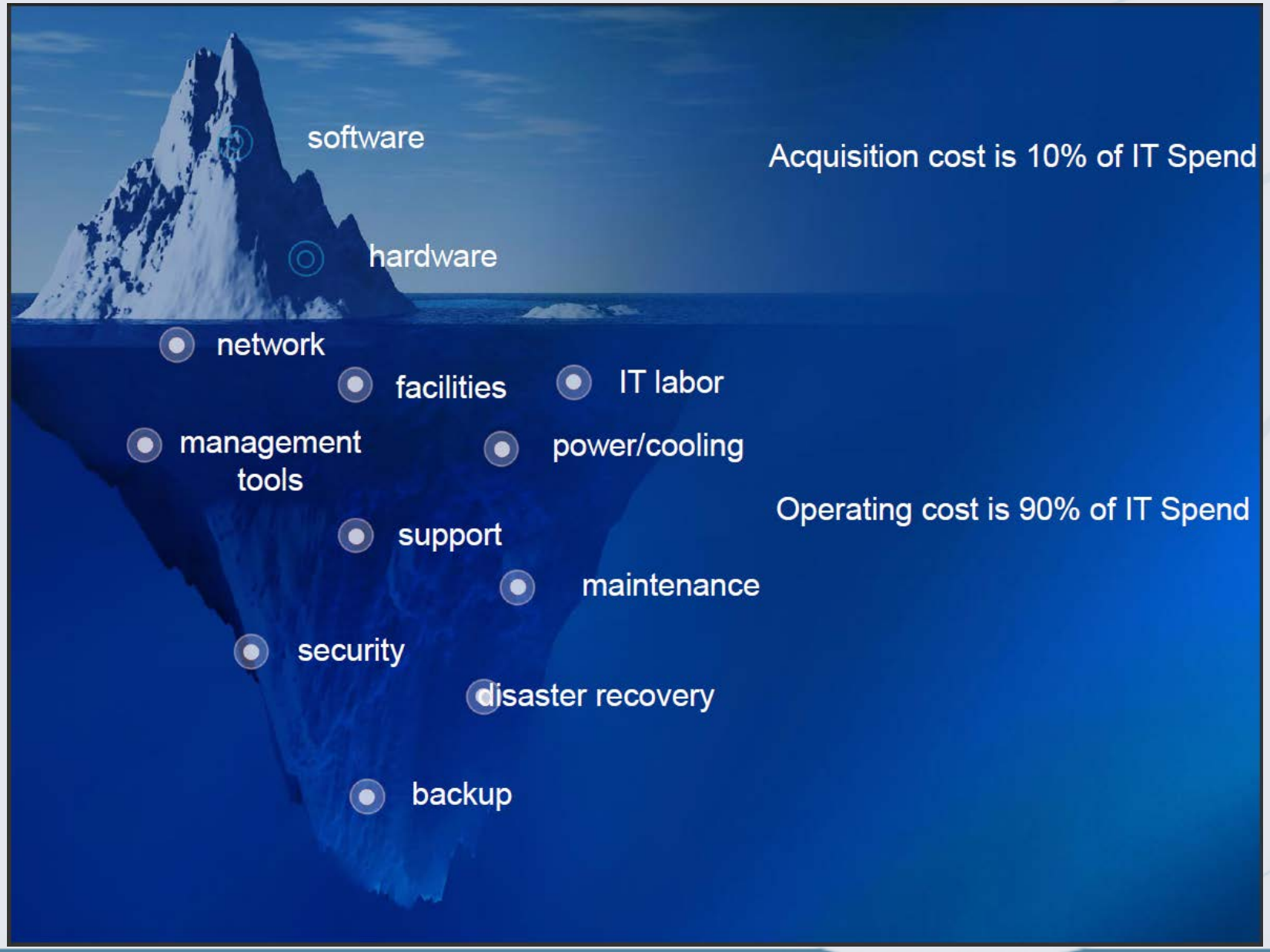
The “old way”



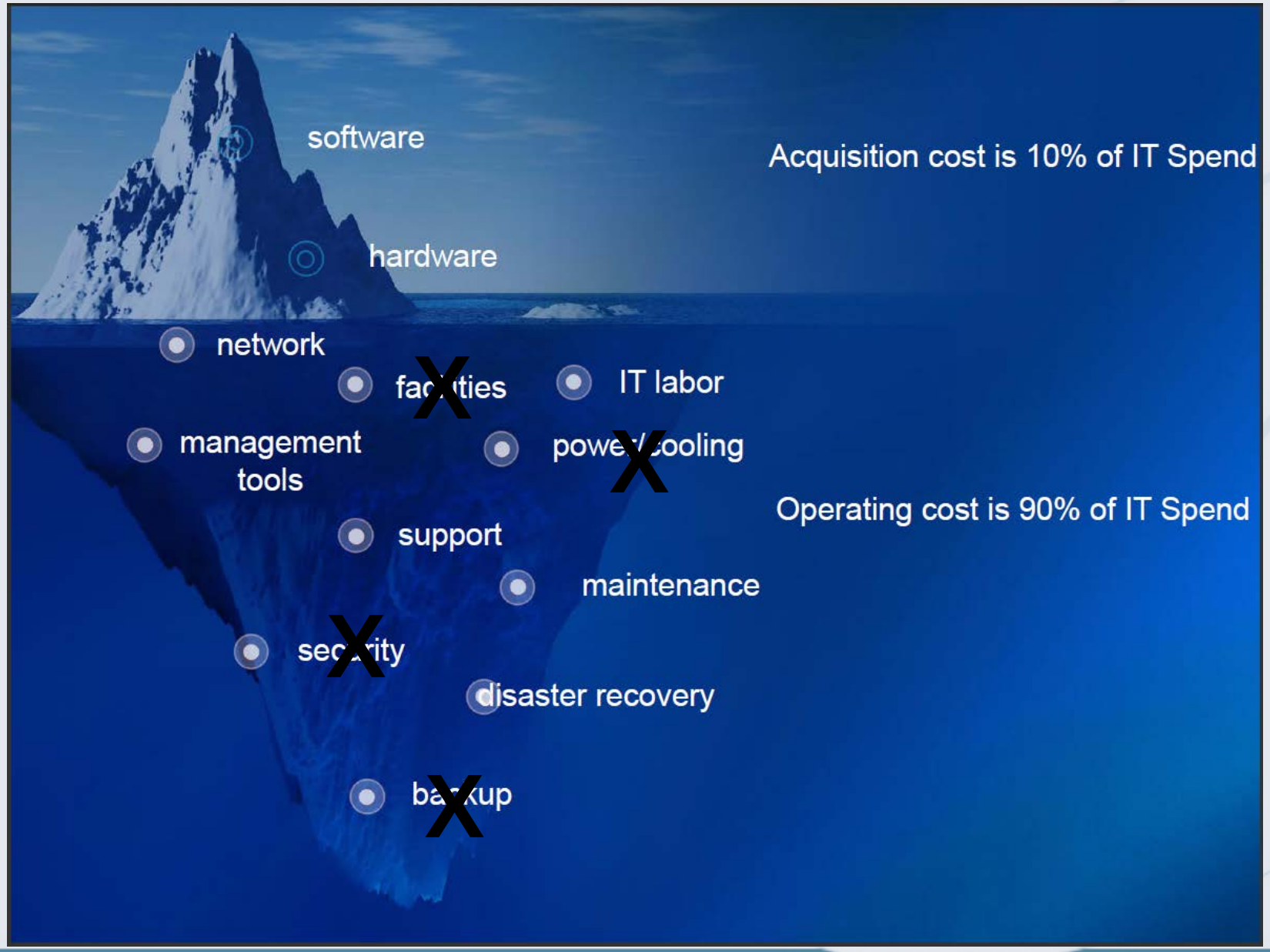
The “new way”



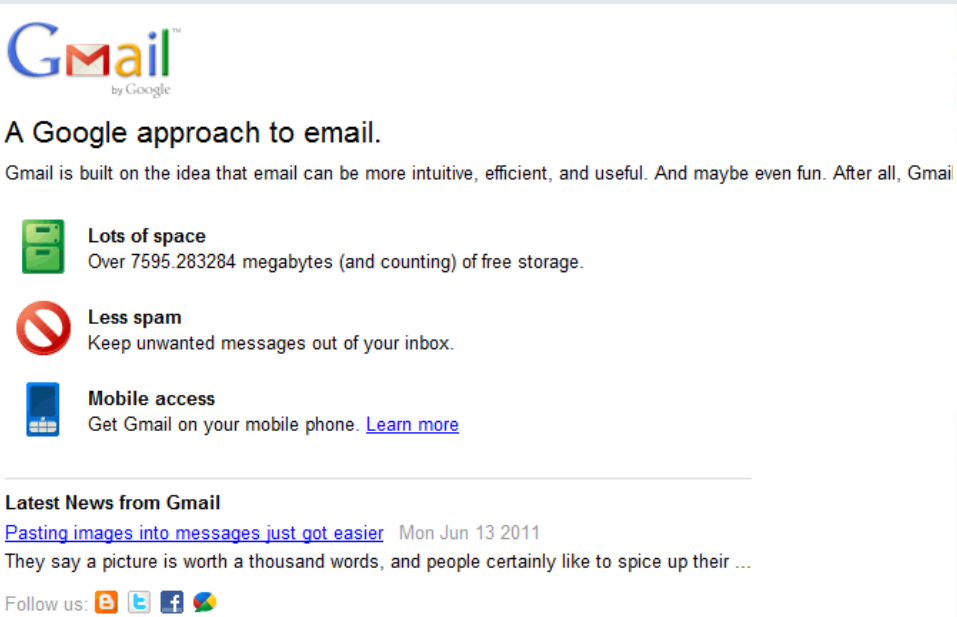
It’s not just above the line

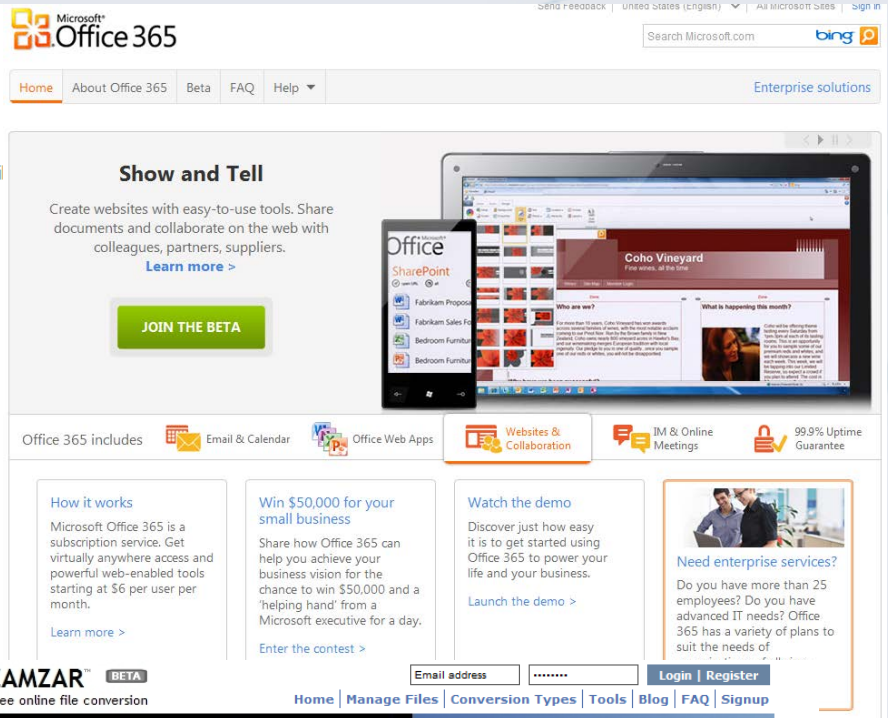


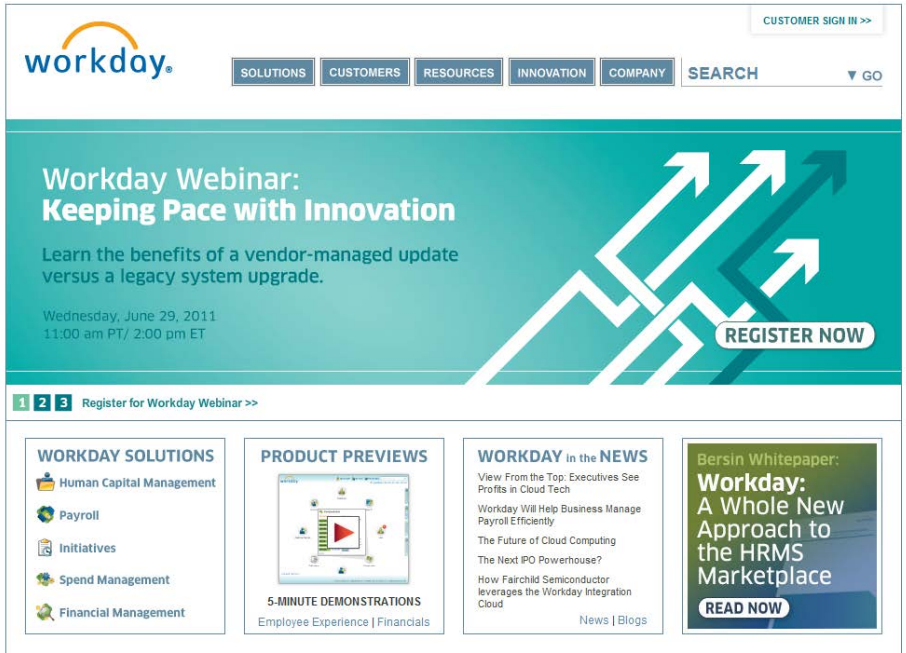
Also (almost) all the rest

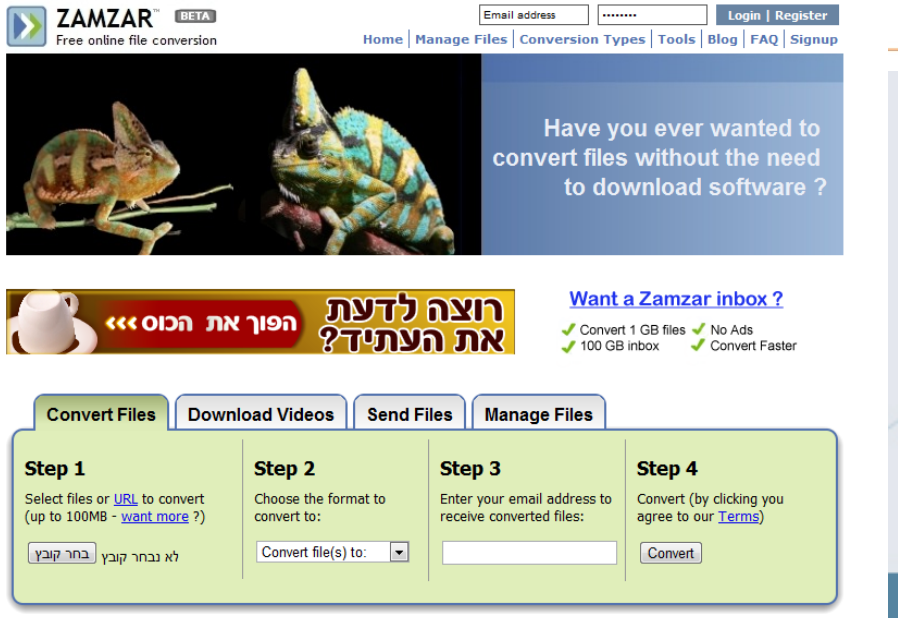


SaaS examples



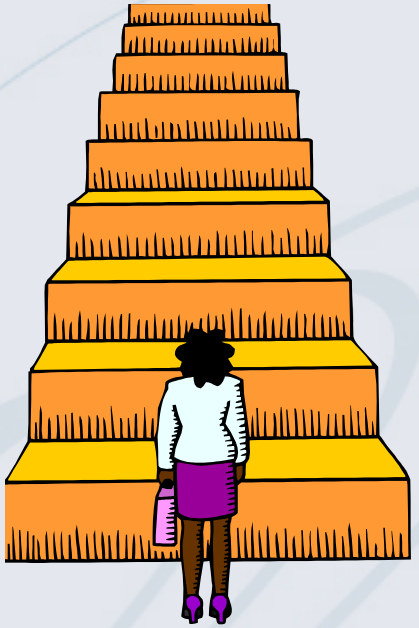






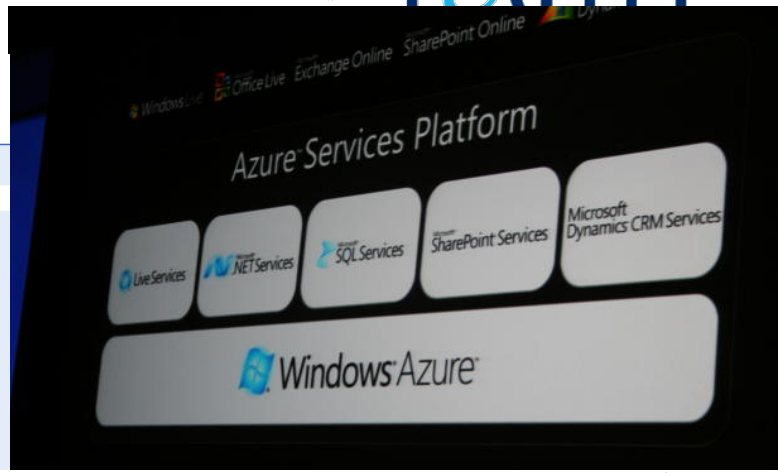
SaaS - Advantages

- Pay per use
- Instant Scalability
- Security
- Reliability
  - Disaster recovery
- APIs



SaaS

PaaS examples



Google code Search
e.g. "templates" or "datastore"

Google App Engine Home Docs FAQ Articles

Run your web apps on Google's infrastructure
Easy to build, easy to maintain, easy to scale

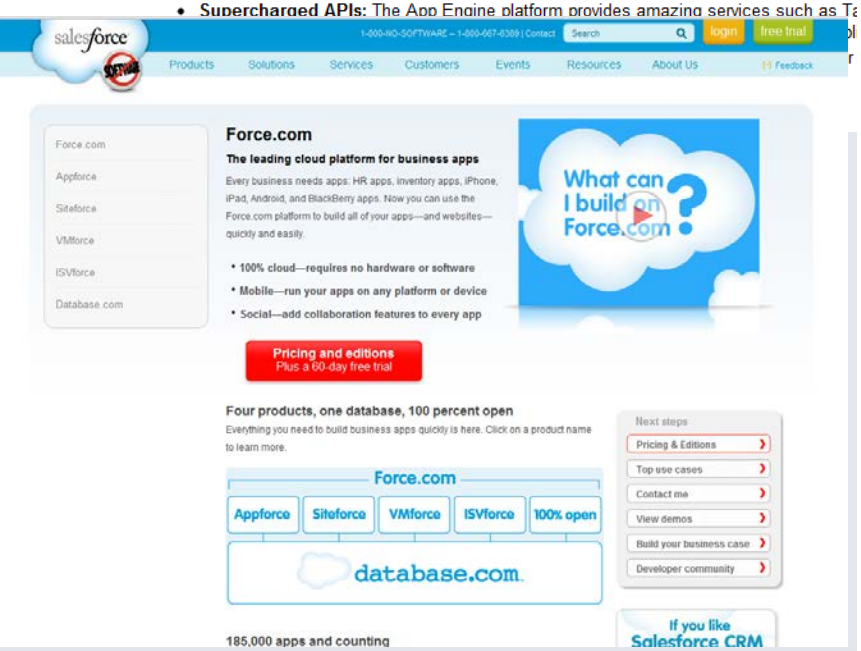
Google App Engine enables you to build and host web apps on the same systems that power Google applications. App Engine offers fast development and deployment; simple administration, with no need to worry about hardware, patches or backups; and effortless scalability.

Discover why developers are choosing App Engine.

Focus on your app, leave the rest to us

All the power of Google in one, simple platform.

- **Zero to sixty:** App Engine enables your application to scale automatically without wo machines.
- **Supercharged APIs:** The App Engine platform provides amazing services such as Ta



amazon web services

Sign in to the AWS Management Console | Create an AWS Account | English

Search: AWS Product Information

AWS | Products | Developers | Community | Support | Account

Products & Services

Amazon EC2 Details
- EC2 Overview
- EC2 FAQs
- EC2 Pricing
- Amazon EC2 SLA
- EC2 Instance Types
- EC2 Instance Purchasing Options
- Reserved Instances
- Spot Instances
- Windows Instances

Amazon EC2 Features
- Elastic Block Store
- Amazon CloudWatch
- Auto Scaling
- Elastic Load Balancing
- High Performance Computing
- VM Import

Amazon Elastic Compute Cloud (Amazon EC2)

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers.

Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change. Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use. Amazon EC2 provides developers the tools to build failure resilient applications and isolate themselves from common failure scenarios.

Easy to sign up, pay only for what you use
Sign Up Now

This page contains the following categories of information. Click to jump down:

- Amazon EC2 Functionality
- Service Highlights
- Features
- Instance Types
- Operating Systems and Software
- Pricing
- Resources
- Detailed Description
- Intended Usage and Restrictions

Amazon EC2 Functionality

Amazon EC2 presents a true virtual computing environment, allowing you to use web service interfaces to launch instances with a variety of operating systems, load them with your custom application environment, manage your network's access permissions, and run your image using as many or few systems as you desire.

Amazon EC2

Amazon pricing

Pricing

Pay only for what you use. There is no minimum fee. Estimate your monthly bill using AWS Simple Monthly Calculator. The prices listed are based on the Region in which your instance is running. For a detailed comparison between On-Demand Instances, Reserved Instances and Spot Instances, see Amazon EC2 Instance Purchasing Options.

Free Tier*

As part of AWS's Free Usage Tier, new AWS customers can get started with Amazon EC2 for free. Upon sign-up, new AWS customers receive the following EC2 services each month for one year:

- 750 hours of EC2 running Linux/Unix Micro instance usage
- 750 hours of Elastic Load Balancing plus 15 GB data processing
- 10 GB of Amazon Elastic Block Storage (EBS) plus 1 million IOs, 1 GB snapshot storage, 10,000 snapshot Get Requests and 1,000 snapshot Put Requests
- 15 GB of bandwidth in and 15 GB of bandwidth out aggregated across all AWS services

On-Demand Instances

On-Demand Instances let you pay for compute capacity by the hour with no long-term commitments. This frees you from the costs and complexities of planning, purchasing, and maintaining hardware and transforms what are commonly large fixed costs into much smaller variable costs.

The pricing below includes the cost to run private and public AMIs on the specified operating system ("Windows Usage" prices apply to Windows Server® 2003 R2, 2008 and 2008 R2). Amazon also provides you with additional instances for Amazon EC2 running Microsoft Windows with SQL Server, Amazon EC2 running SUSE Linux Enterprise Server and Amazon EC2 running IBM that are priced differently.

Region: EU (Ireland)

	Linux/UNIX Usage	Windows Usage
Standard On-Demand Instances		
Small (Default)	$0.095 per hour	$0.12 per hour
Large	$0.38 per hour	$0.48 per hour
Extra Large	$0.76 per hour	$0.96 per hour
Micro On-Demand Instances		
Micro	$0.025 per hour	$0.035 per hour

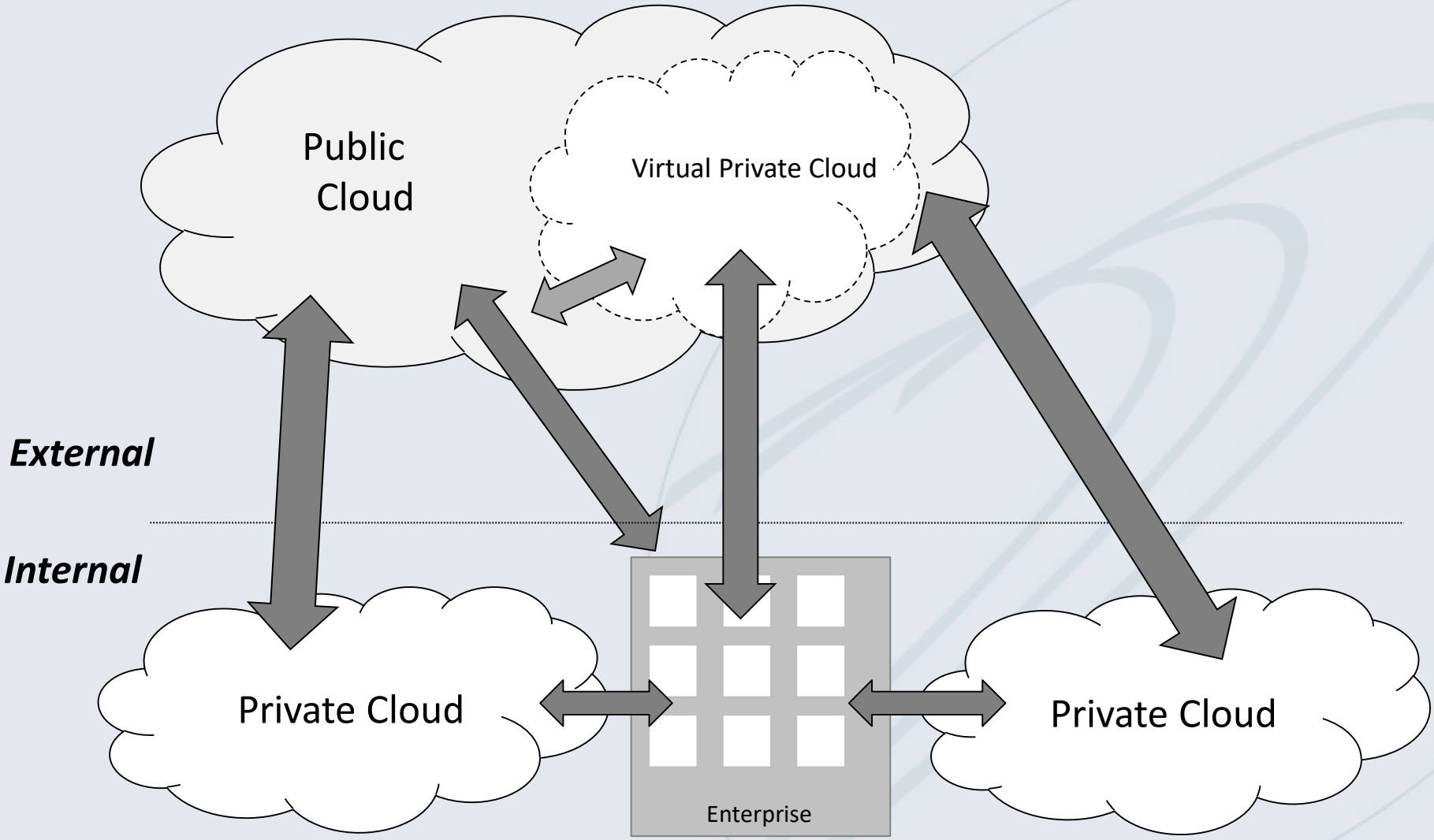
Common Factors

- Pay per use
- Instant Scalability
- Security
- Reliability
- APIs

Aren’t we missing something ?

- NaaS – Network (connectivity) as a Service
  - Manage the network itself
  - Having on demand bandwidth
  - Network abstraction

Cloud Ecosystem



What operator learned ?

- Software can be bought as SaaS
- They can build cloud
  - For their on need
  - For customers
  - NaaS is required

- Need to decuple software from hardware
- It’s all about IT (engineering is declining fast)
- Reuse...

LOGTEL
It's more than knowledge

e
ce innovation
ion with

scale and utilize
sources more

ating costs
mogenized
rastructure

NFV

The require

1. Maintain/ex
   service SLA
   performanc
2. Operate acr
   traditional a
   based infra
3. Operate in a
   environmer

Definition

- ***Network functions virtualization (NFV) is***
- a network architecture concept that uses the technologies of IT virtualization to virtualize entire classes of network node functions into building blocks that may connect, or chain together, to create communication services.
- NFV relies upon, but differs from, traditional server-virtualization techniques, such as those used in enterprise IT. A virtualized network function, or VNF, may consist of one or more virtual machines running different software and processes, on top of standard high-volume servers, switches and storage devices, or even cloud computing infrastructure, instead of having custom hardware appliances for each network function.

Source: https://en.wikipedia.org/wiki/Network_function_virtualization

NFV: IT'S NOT JUST THE NETWORK—IT'S ALL ABOUT THE SERVICE!

OCTOBER 2016

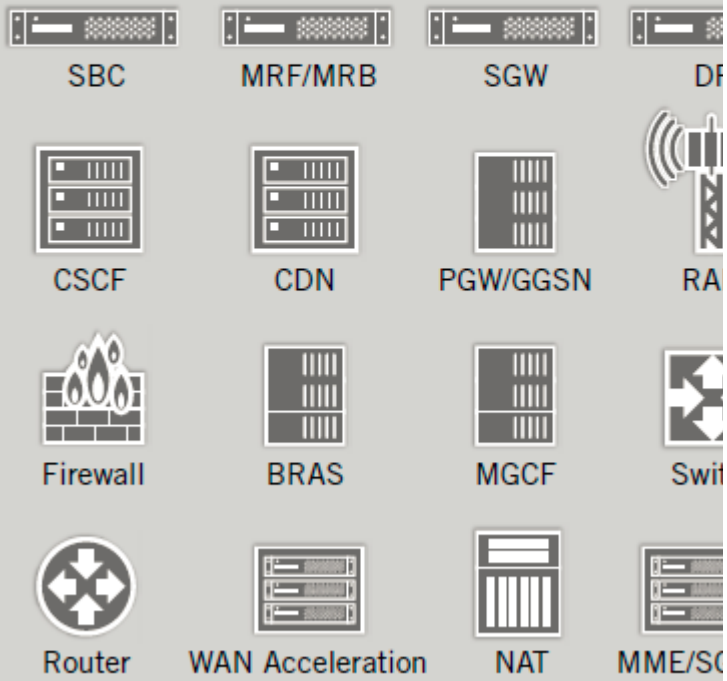
amdocs

embrace challenge experience success®

No more dedicated hardware

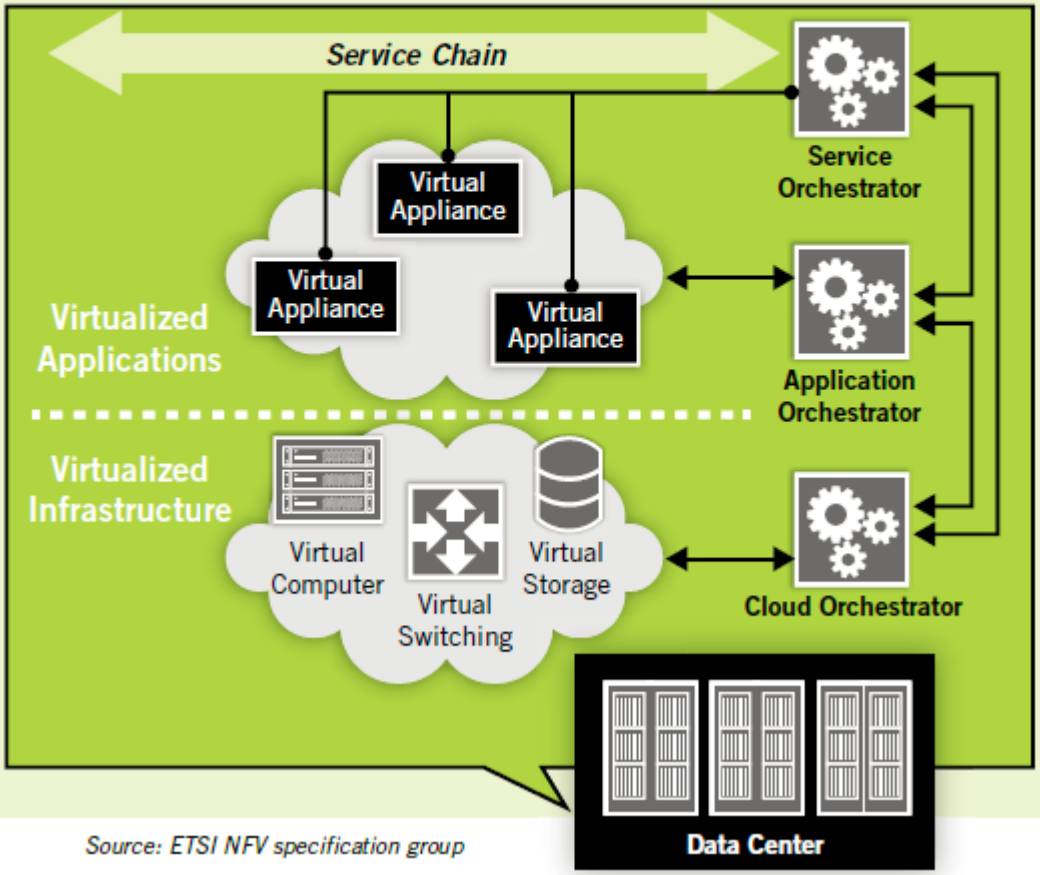
Classical Hardware Appliance Approach

- Fragmented non-commodity hardware
- Physical install for each appliance
- Hardware development = barrier to entry to new ve
  - *Constrains innovation*
  - *Restricts competition*



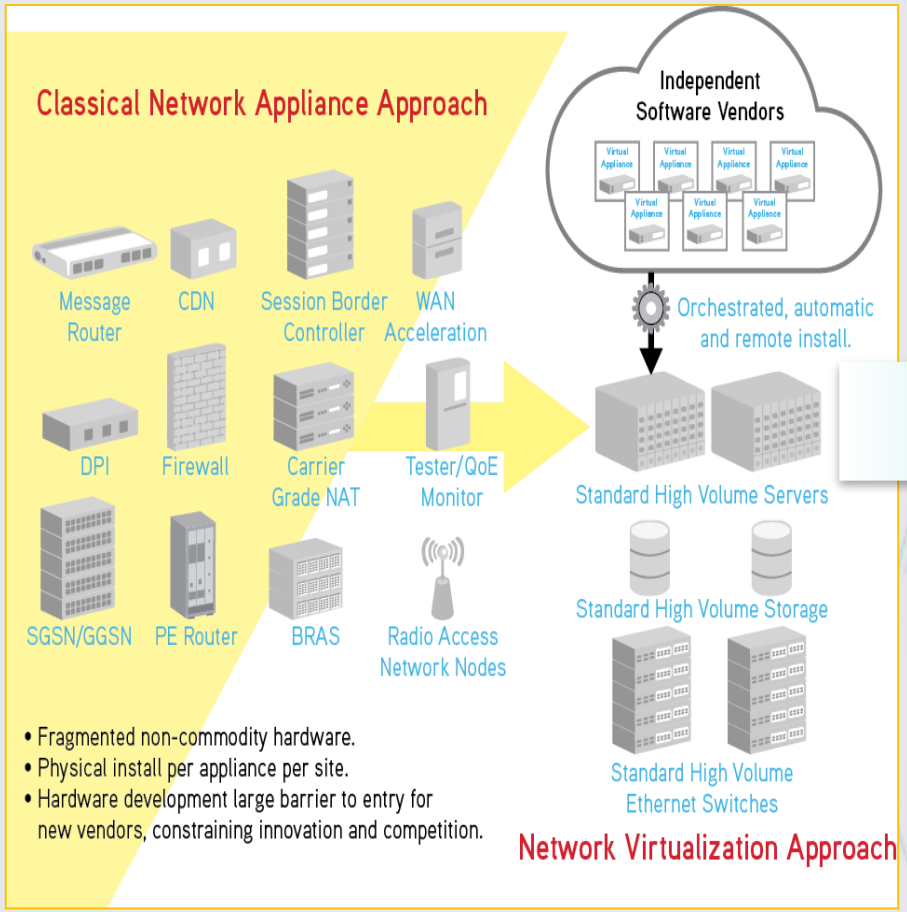
Network Functions Virtualization Approach

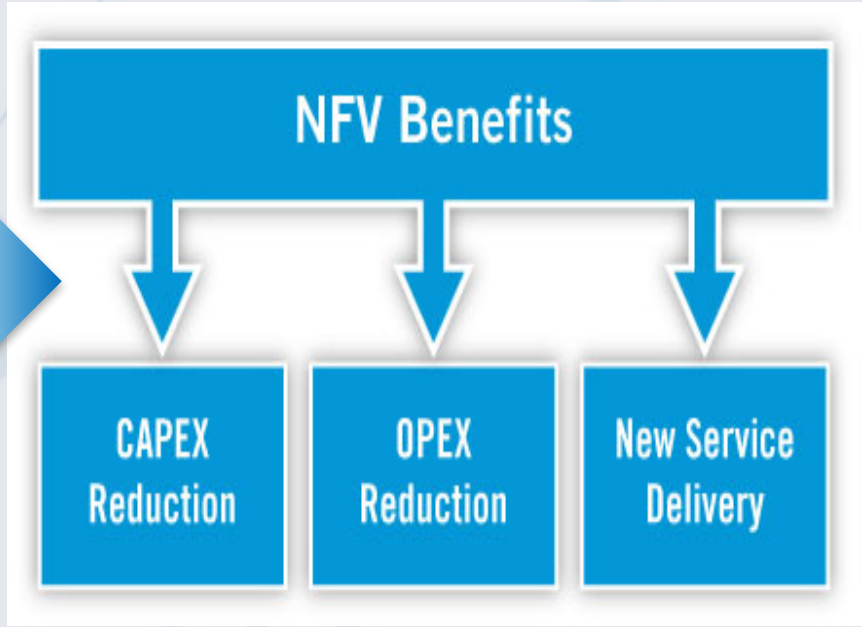
- Diverse ecosystem of ISV application providers
- Orchestrated, automatic and remote install
- COTS servers, storage and networking
- Cloud-based benefits of agility and elasticity



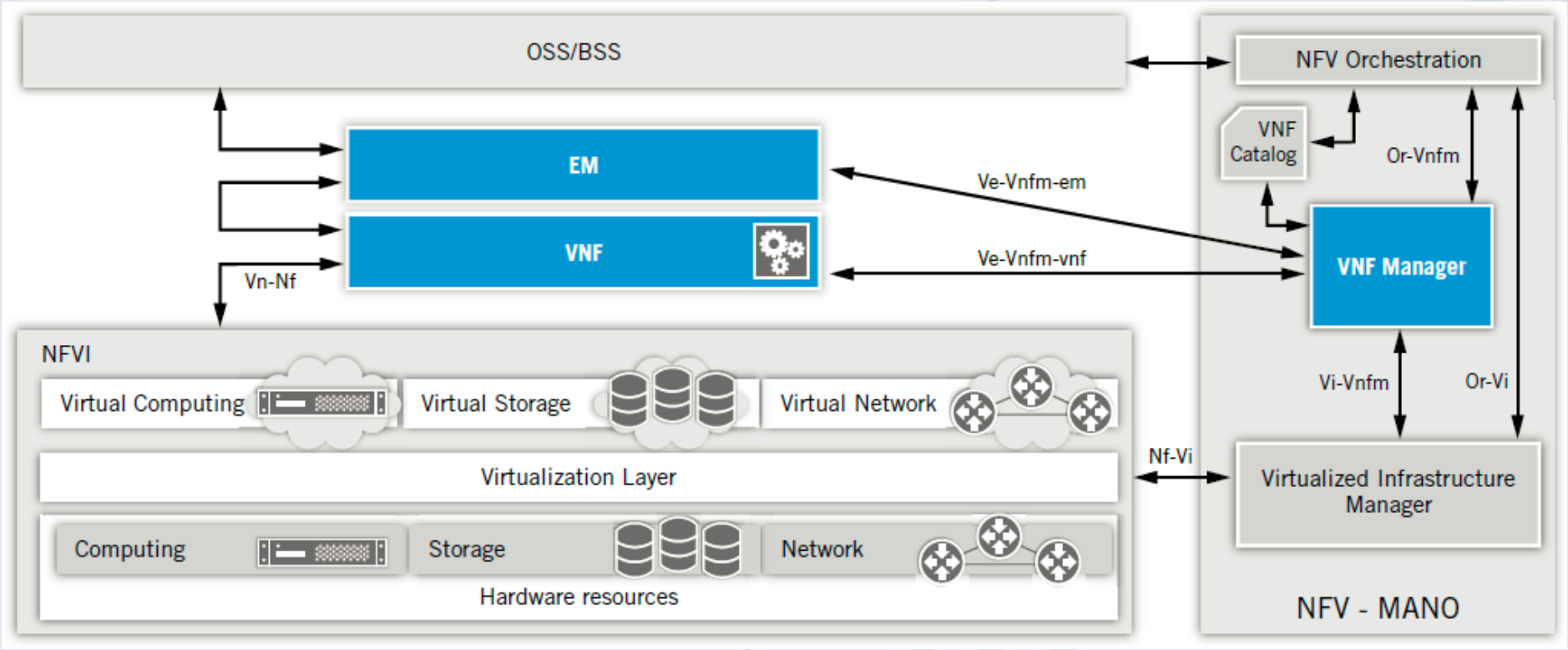
Source: ETSI NFV specification group

NFV Benefits

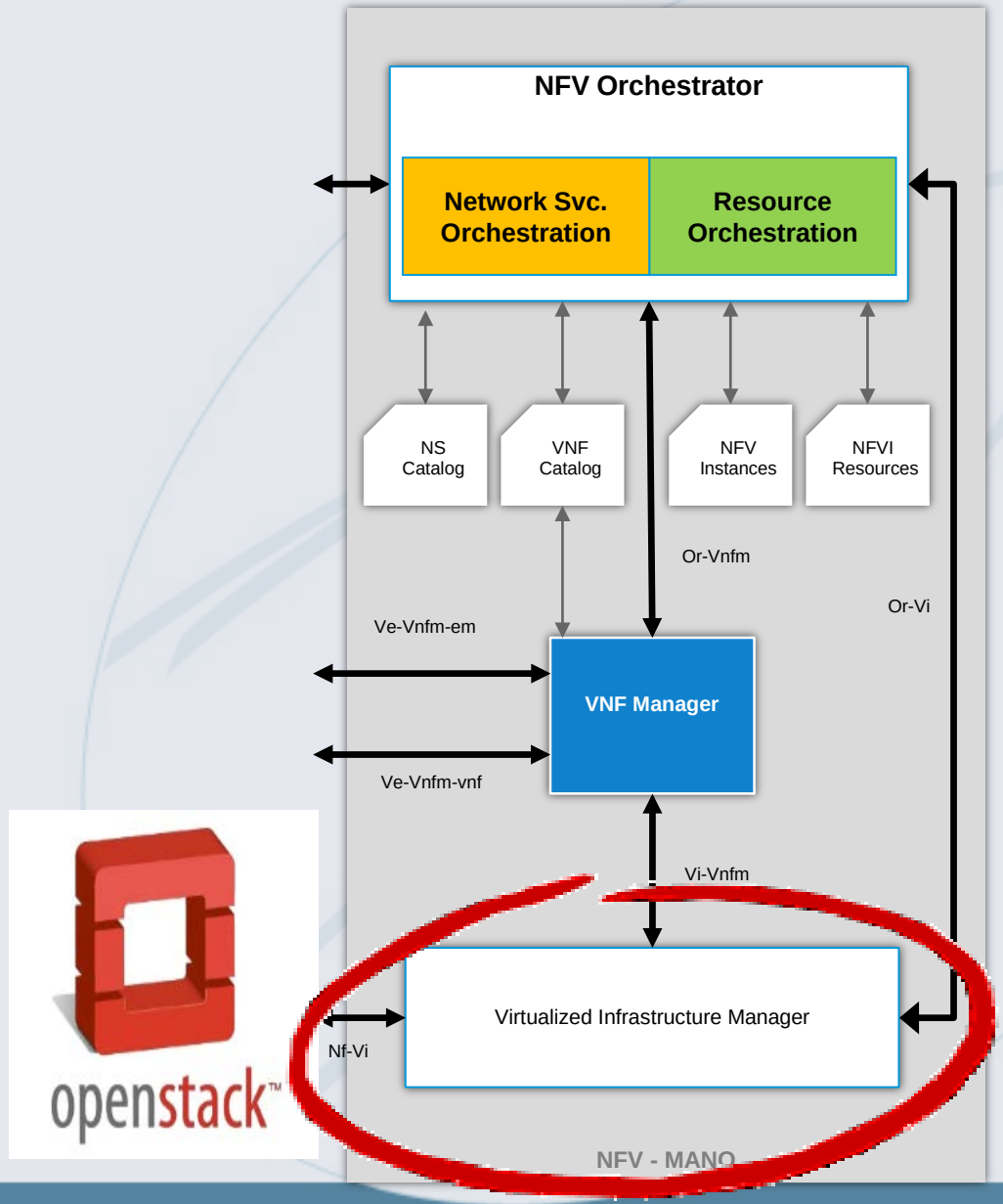




NFV HL architecture



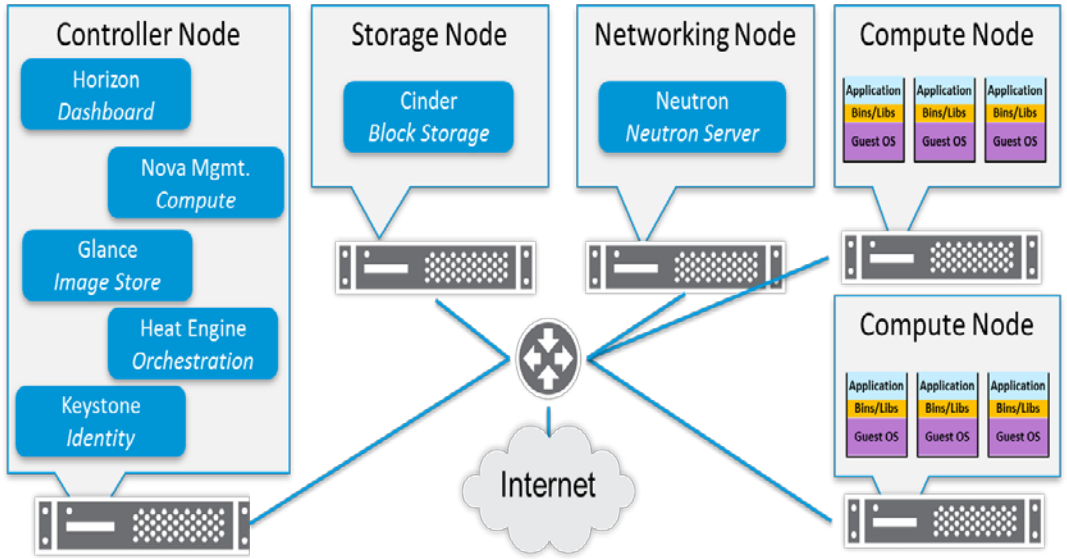
Focus on OpenStack – Open Source VIM



Open Stack – A Cloud Operating System

OpenStack APIs

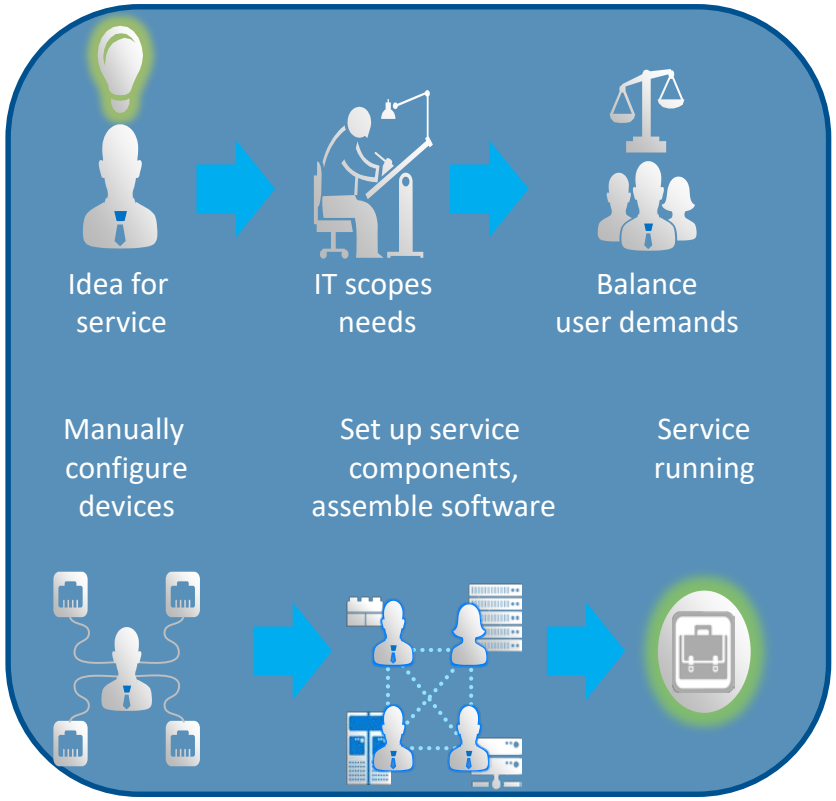
- **Nova**: Open Stack Compute automates provisioning of virtual machines
- **Cinder**: Block storage system accessible by VMs
- **Neutron**: OpenStack service for establishing connectivity between VMs
- **Swift**: Large scale redundant object (i.e. VM) store
- **Glance**: Image service that catalogs and manages (stores, retrieves, registers and delivers) virtual machine images
- **Keystone**: provides system-wide authentication
- **Ironic**: Bare metal management
- **Horizon**: Web-based dashboard access and management for users



The Intel SDI Vision

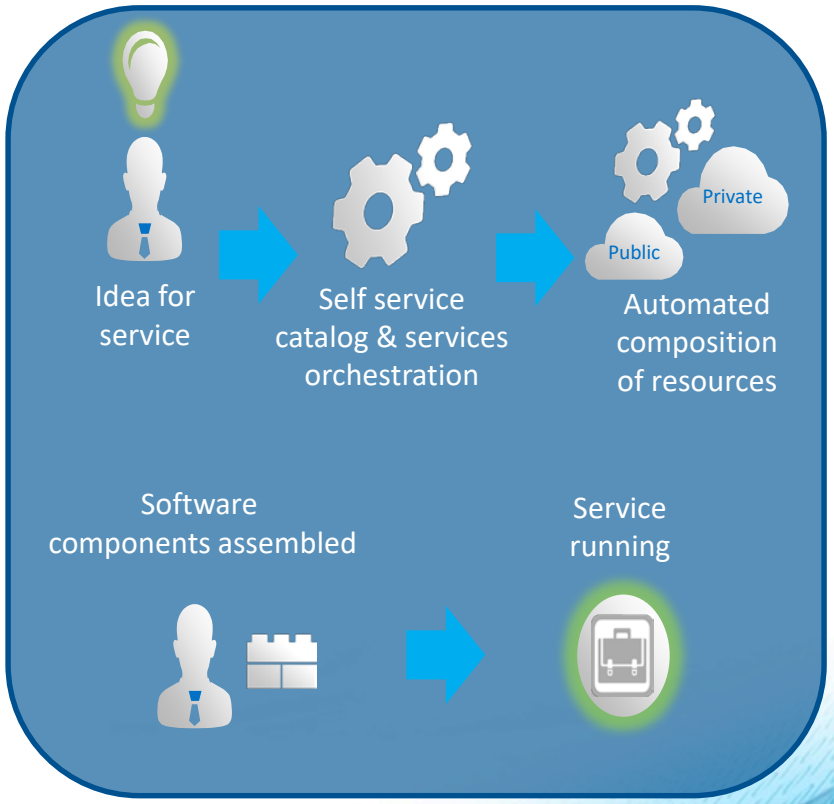
Self-provisioning, automated orchestration, composable resource pools

Datacenter Today



Time to Provision New Service: Months[1]

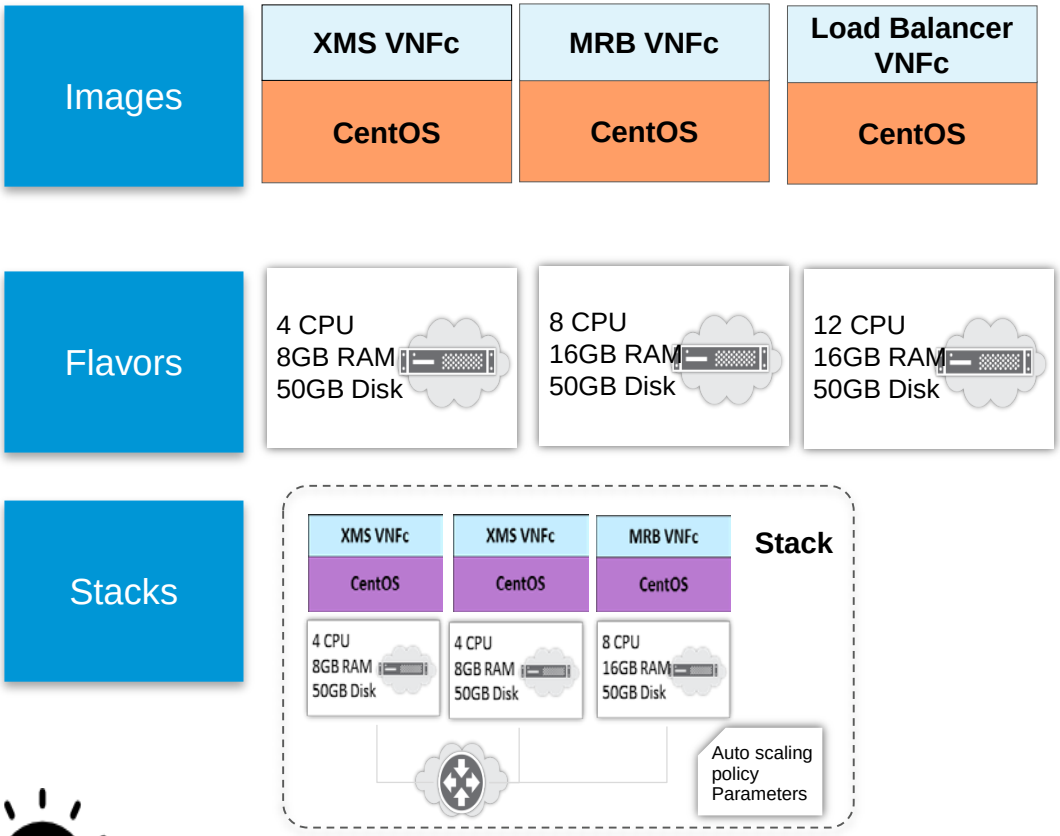
Software-defined Infrastructure



Time to Provision New Service: Minutes[1]

1: Source: Intel IT internal estimate

Open Stack – A Cloud Operating System



Heat Templates contain info to create stacks

Stack: Collection of resources

› Virtual machines, networks, auto scaling rules

Template: Definition of resources that make up the Stack

› Four sections

- **Resources** – Objects that will be created - like a server
- **Properties** – image, flavor
- **Parameters** – Property values
- **Output** – Information passed back to user or dashboard

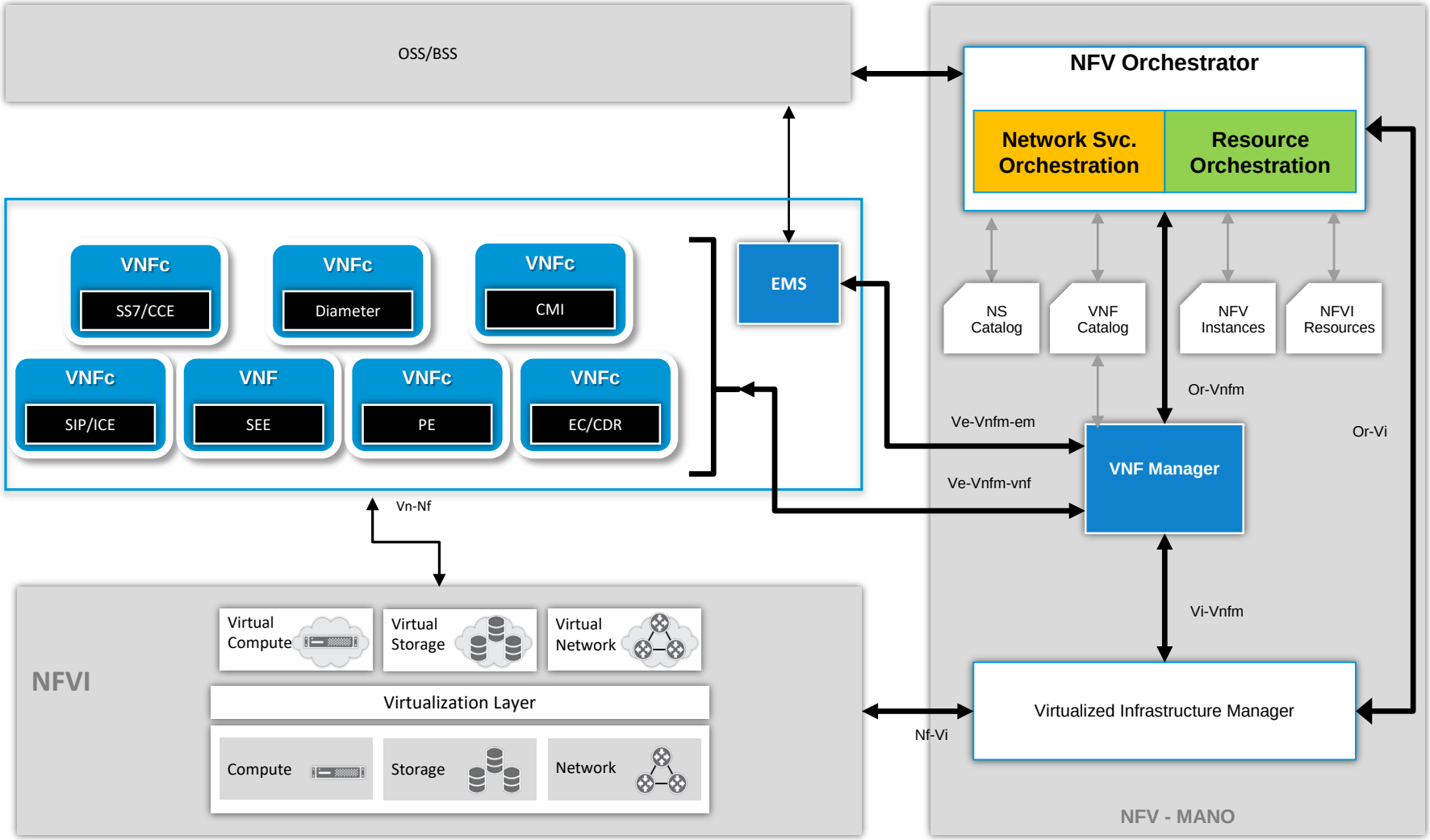
Platform Description

- Supports the private cloud infrastructure for Verizon virtualized network applications
- Increases speed and agility of service delivery
- Reduces capital and operating expenses
- Provides compute, network, and storage infrastructure to support both traditional applications and cloud-enabled or cloud native applications
- Open source and third party software, including OpenStack, Linux, and KVM, used for key software components

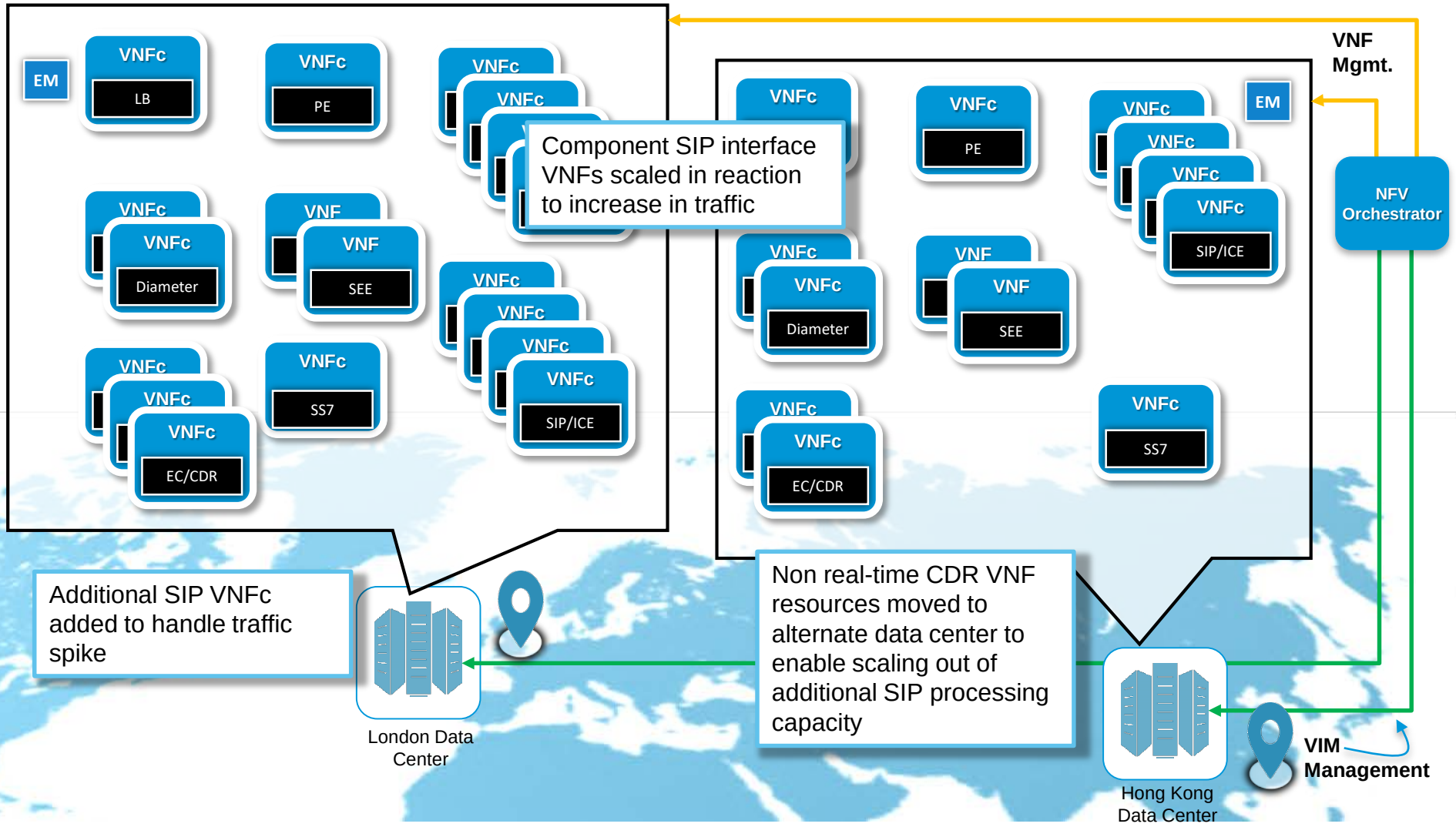
Compute

- Initial release supports hosted applications in virtual machines (VMs), using the KVM hypervisor
- KVM hypervisor supports a wide range of Linux, Windows, FreeBSD, and other operating systems; hosted applications can use any of the guest OSs in VMs that host their applications
- Initial release uses RHEL 7.1 release of the KVM hypervisor
- Uses Red Hat OpenStack Platform 7 software for cloud management
- Nova component of OpenStack cloud management software used to automate configuration of servers on platform
- Each VM provisioned with specific processor, memory, and storage resources
    - These resource combinations are called flavors. Platform supports a range of VM flavors for hosted applications
- Specifications of maximum size VM supported on initial release are:
    - vCPU: 24
    - Memory: 96 GB
    - Storage in VM: 500 GB

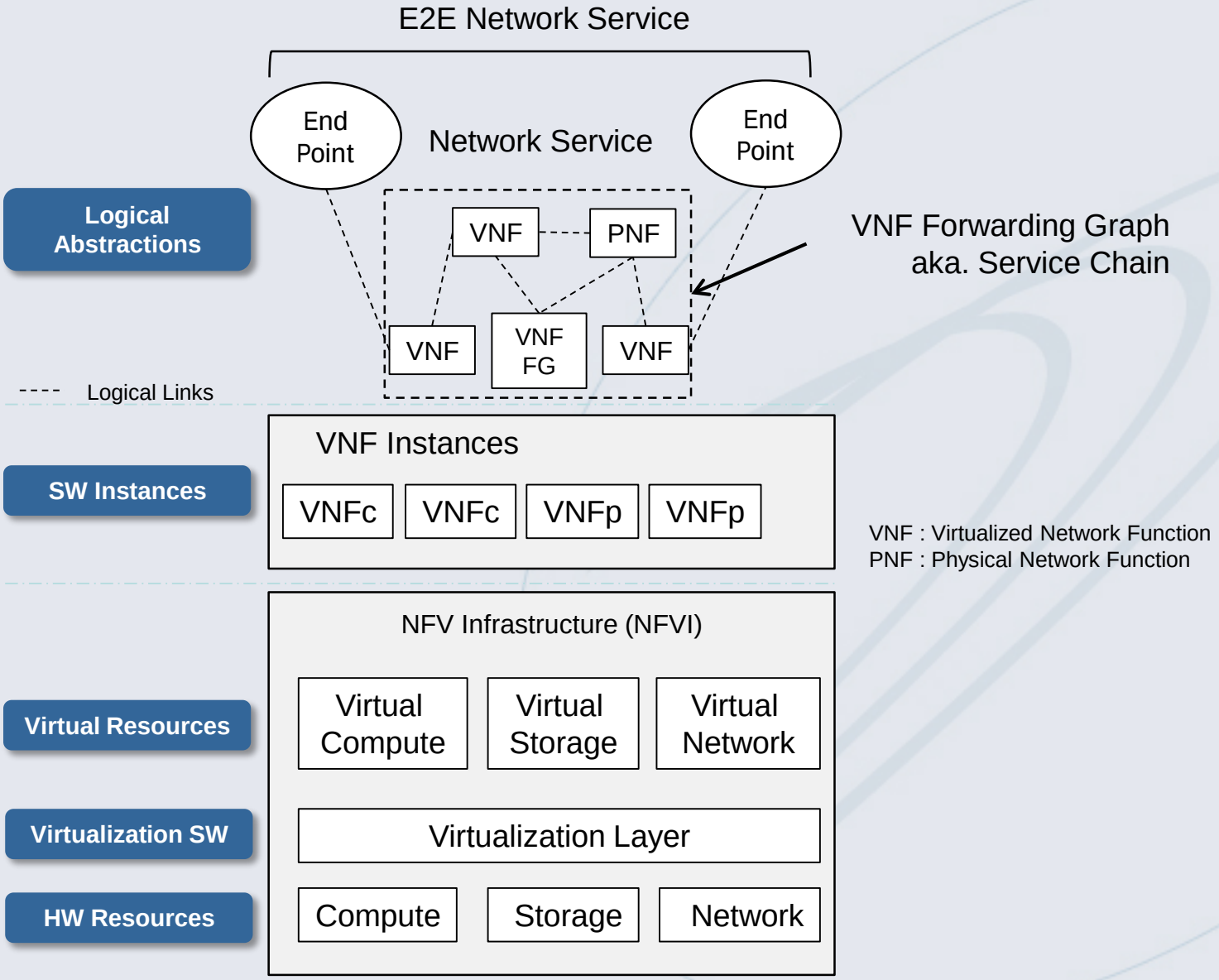
Dialogic® ControlSwitch™ System
Intelligent Routing, Call and Session Control in the Cloud



The Benefits of Decomposition and Virtualization



ETSI NFV - Layered view



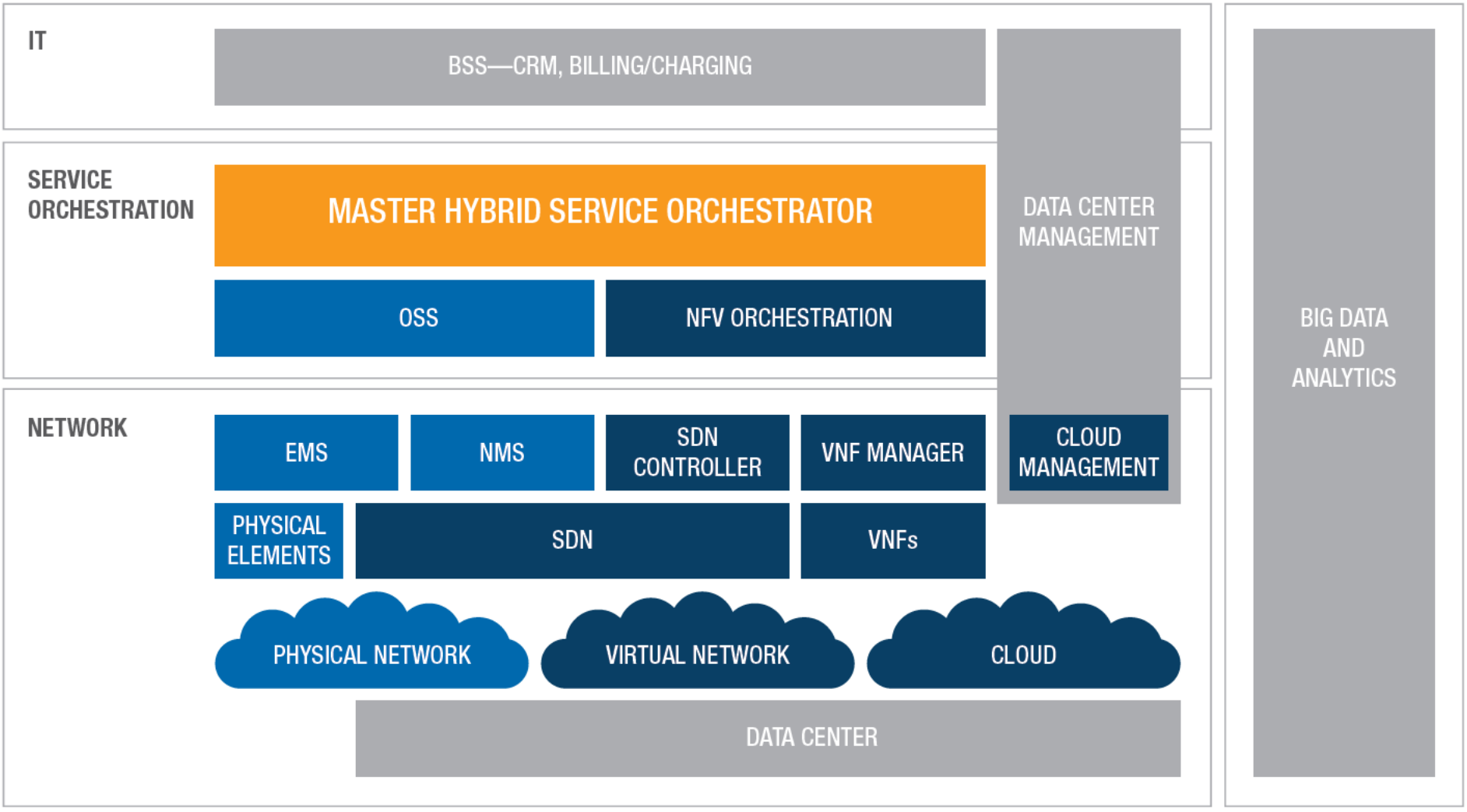
So NFV is about

- Decupling
- (auto) Scaling
- Service chain
- Network agility
- Reduce over engineering
- Resource reuse

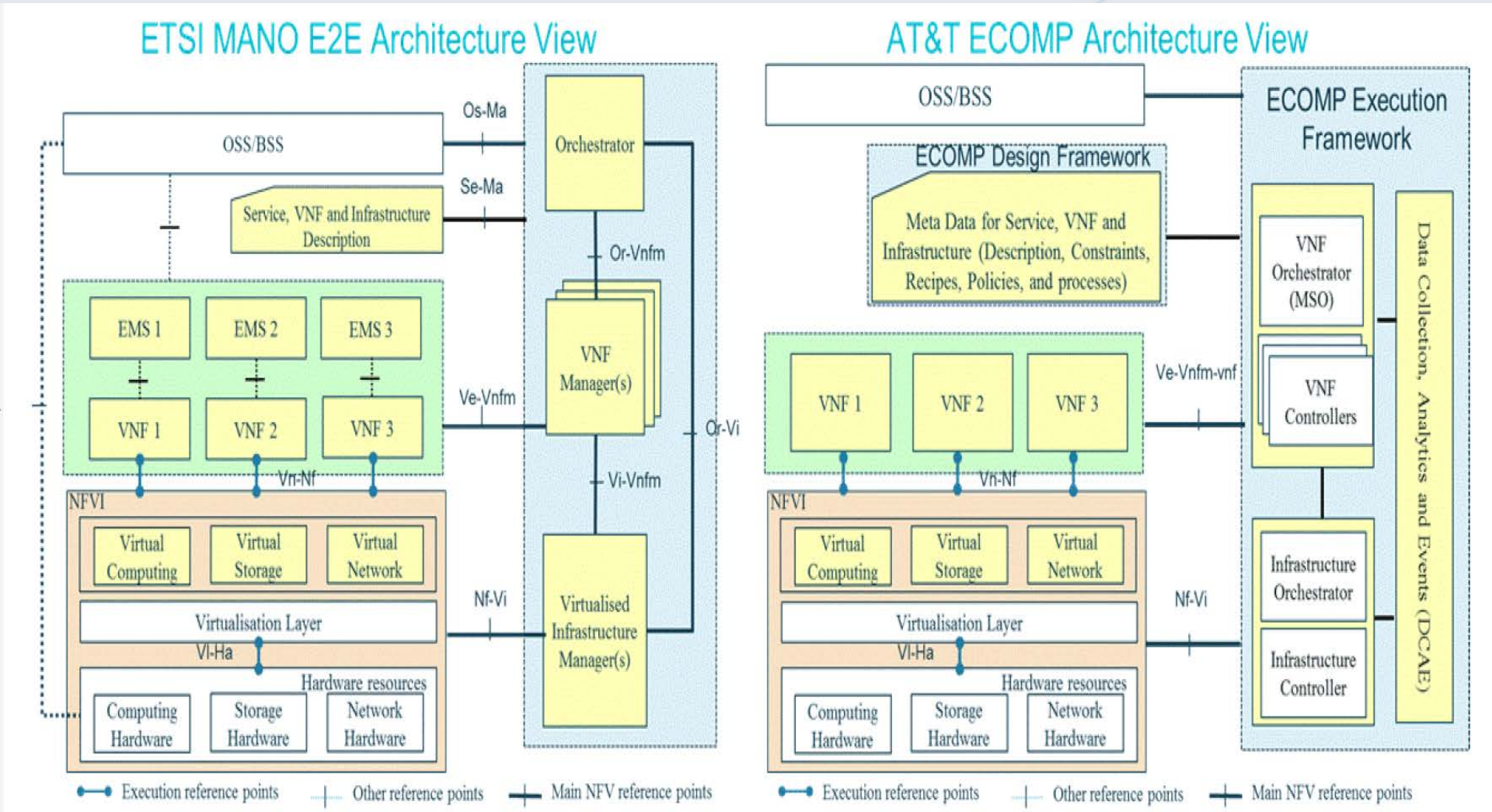
- Virtualization
- Open source
- Micro services

amdocs view

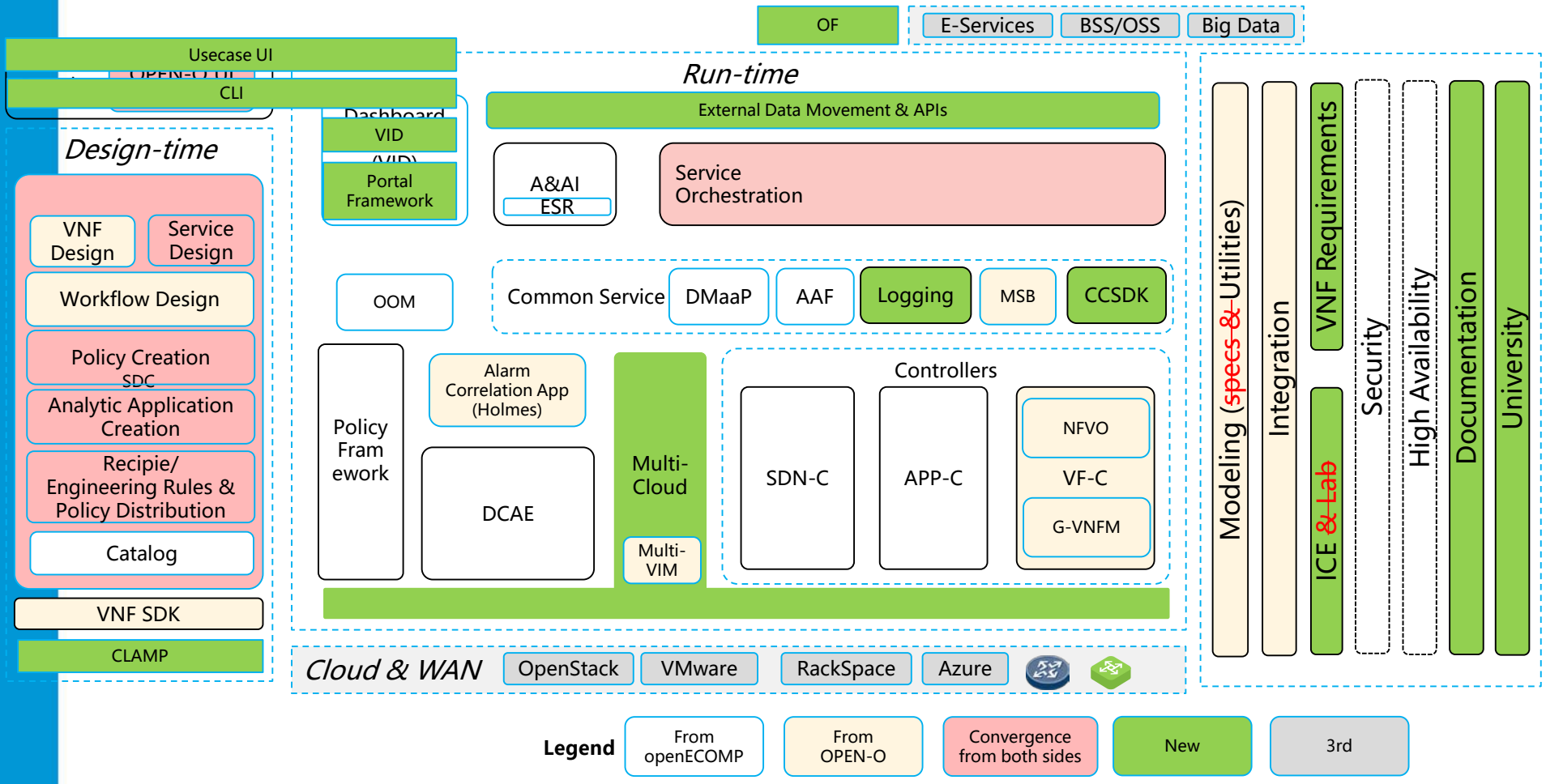
service-driven network transformation



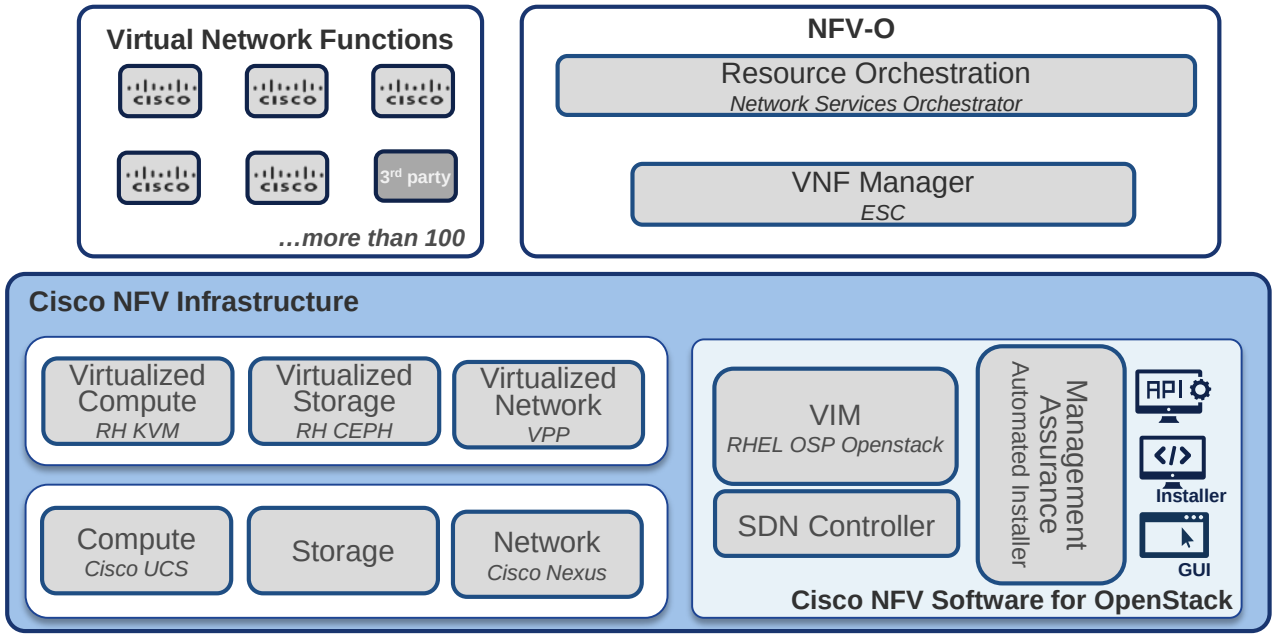
ETSI MANO vs. AT&T ECOMP Architectures



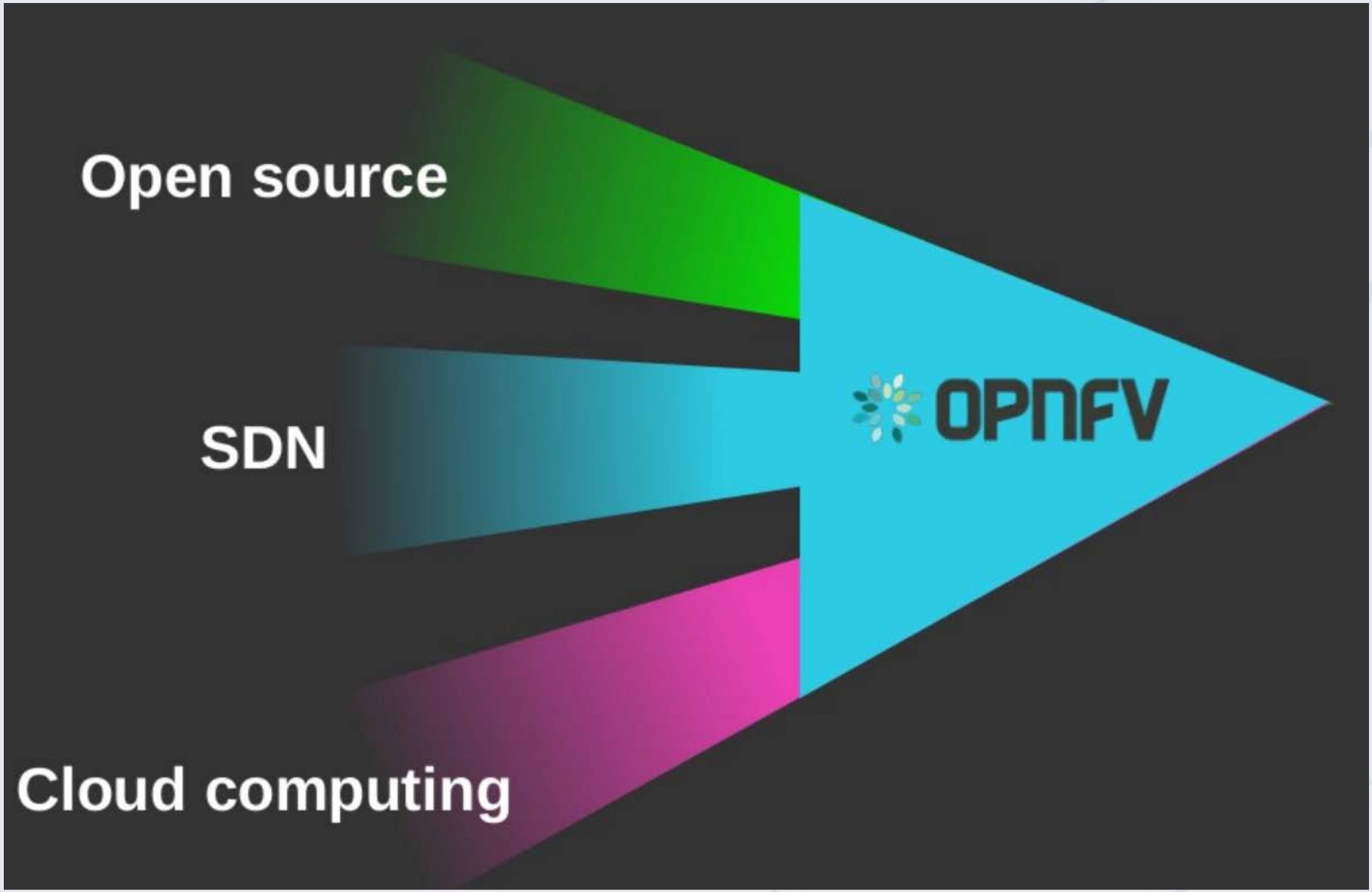
ONAP Projects R1 (Lingli)



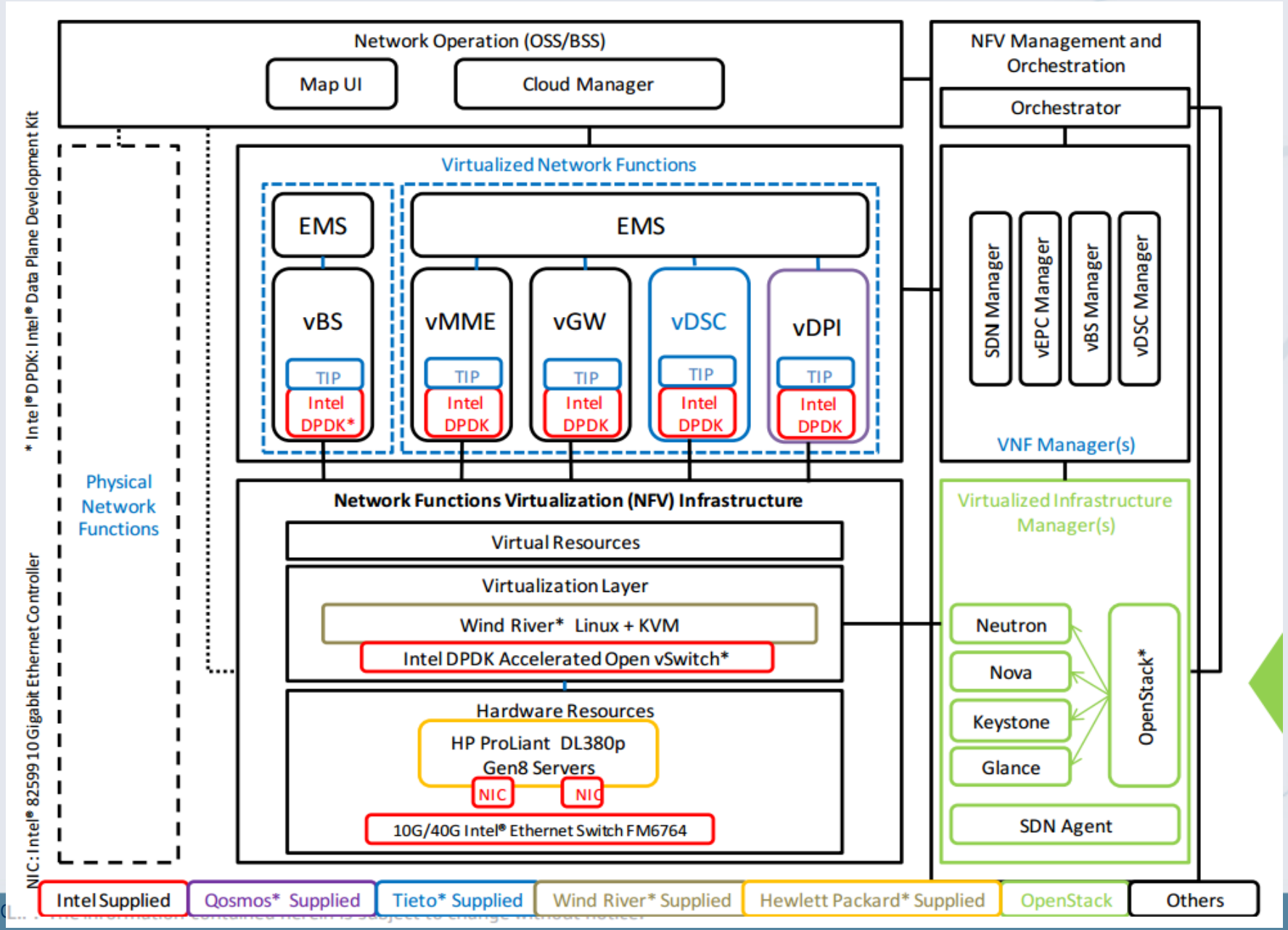
Comprehensive Cisco NFV Architecture



Why now ?

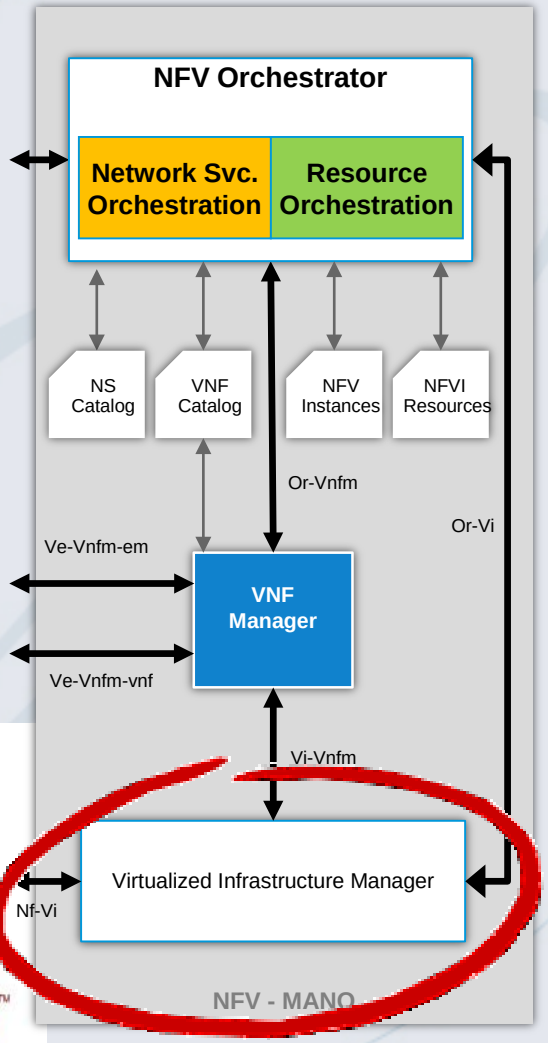


Full view


Network Operation (OSS/BSS)
Map UI
Cloud Manager
NFV Management and Orchestration
Orchestrator
Virtualized Network Functions
EMS
EMS
vBS
vMME
vGW
vDSC
vDPI
TIP
Intel DPDK*
Intel DPDK
SDN Manager
vEPC Manager
vBS Manager
vDSC Manager
VNF Manager(s)
Physical Network Functions
Network Functions Virtualization (NFV) Infrastructure
Virtual Resources
Virtualization Layer
Wind River* Linux + KVM
Intel DPDK Accelerated Open vSwitch*
Hardware Resources
HP ProLiant DL380p Gen8 Servers
NIC
10G/40G Intel® Ethernet Switch FM6764
Virtualized Infrastructure Manager(s)
Neutron
Nova
Keystone
Glance
OpenStack*
SDN Agent
* Intel® DPDK: Intel® Data Plane Development Kit
NIC: Intel® 82599 10 Gigabit Ethernet Controller
Intel Supplied
Qosmos* Supplied
Tieto* Supplied
Wind River* Supplied
Hewlett Packard* Supplied
OpenStack
Others

Focus on OpenStack – Open Source VIM

- Cloud operating system at the data center level
  - Controls physical compute, networking, storage & virtualization technology resources
- Creates pools of resources and automates the network
  - Turns hypervisors within a data center or across multiple data centers into pools of resources.
- Common platform across enterprise and public clouds – cloud federation
  - Move data and applications between clouds
- OpenStack Compute service
  - Hypervisor and hardware agnostic
  - Rest-based API
- OpenStack Image service
  - Stores and retrieves virtual machine images
  - Rest-based APIs
- OpenStack APIs enable configuration of
  - Compute, storage and memory - Flavor
  - Application and Guest OS - Image
  - Networking



What is SDN?

- Software-defined networking (SDN) is the separation of the **Network Control Plane** from the **Forwarding Plane**, and where a control plane can control multiple devices.

  Open Network Foundation

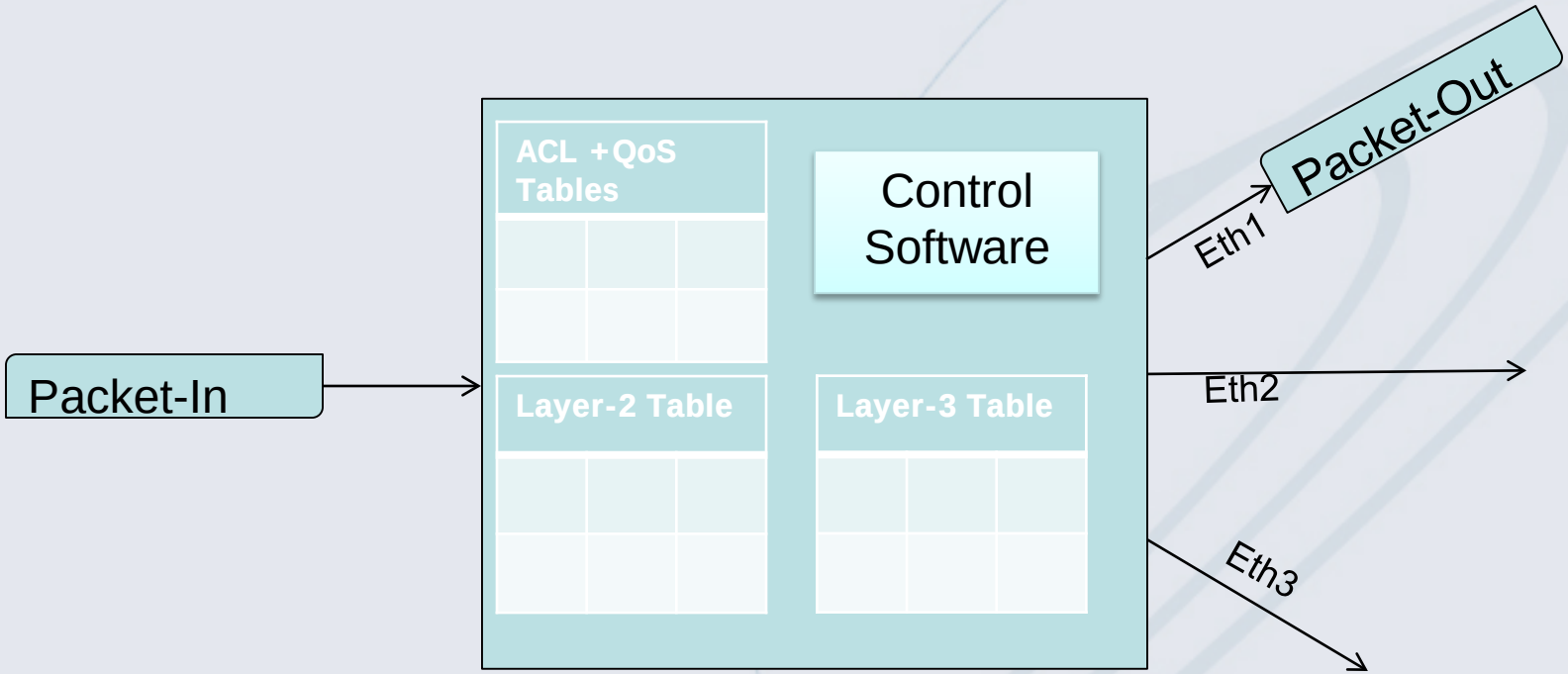
- SDN is whatever we can ship today

  Vendor X

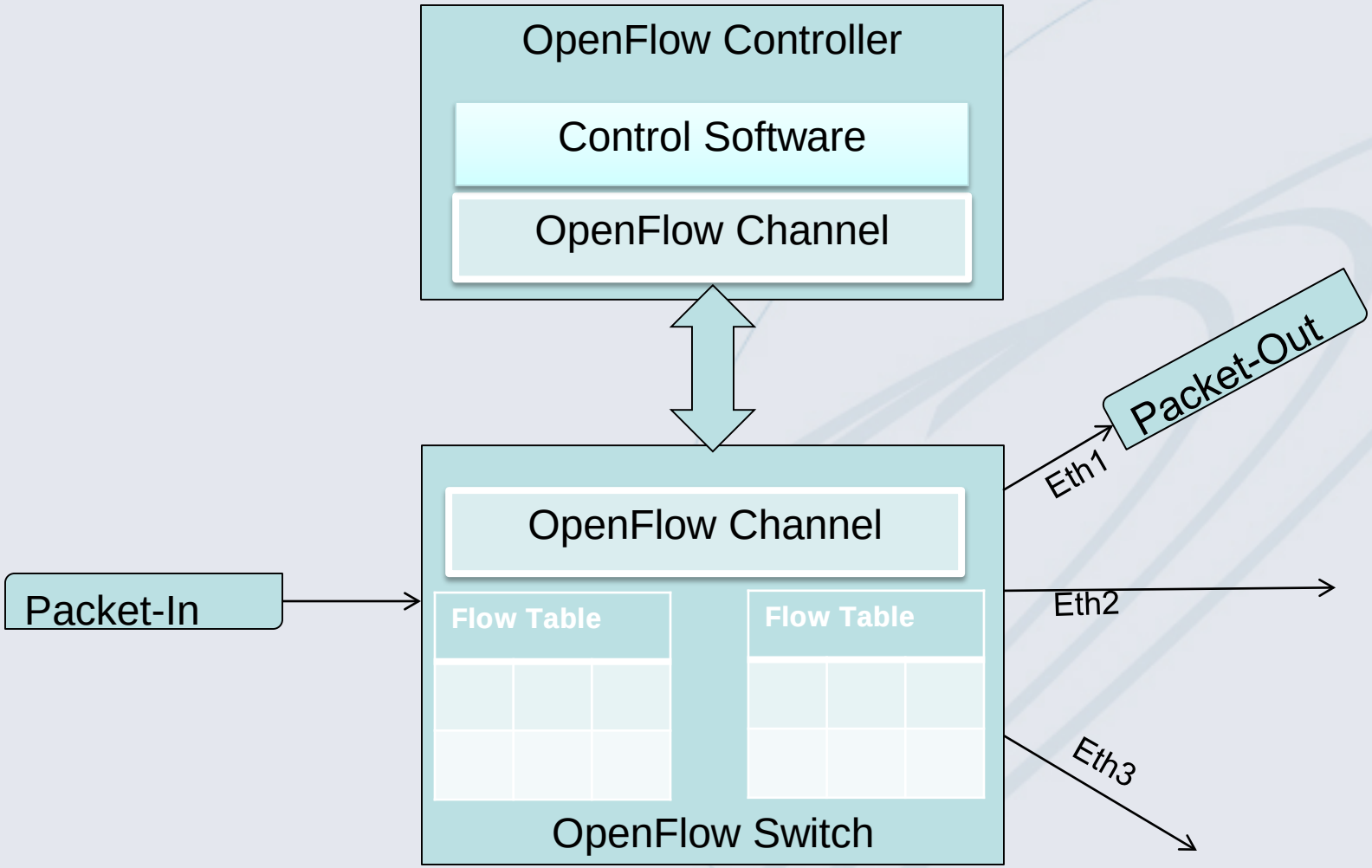
- SDN is the magic buzzword that will bring us VC funding

  Startup Y

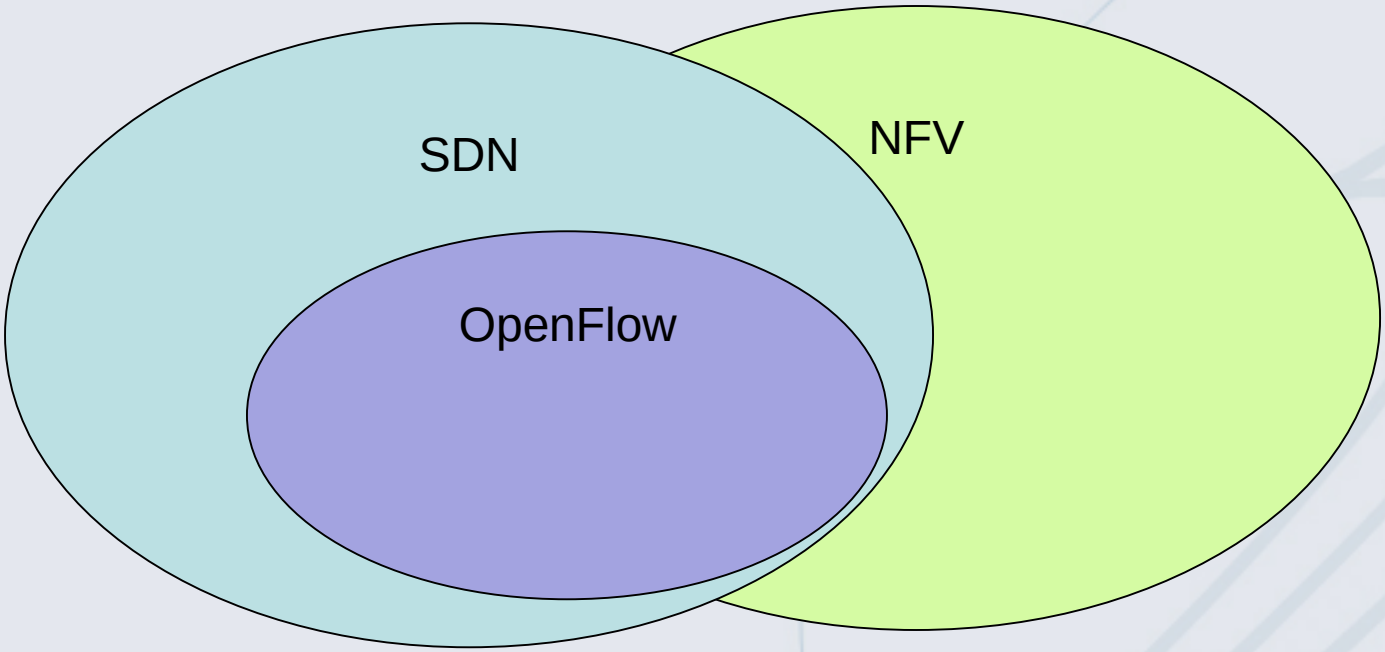
Traditional Switch Forwarding



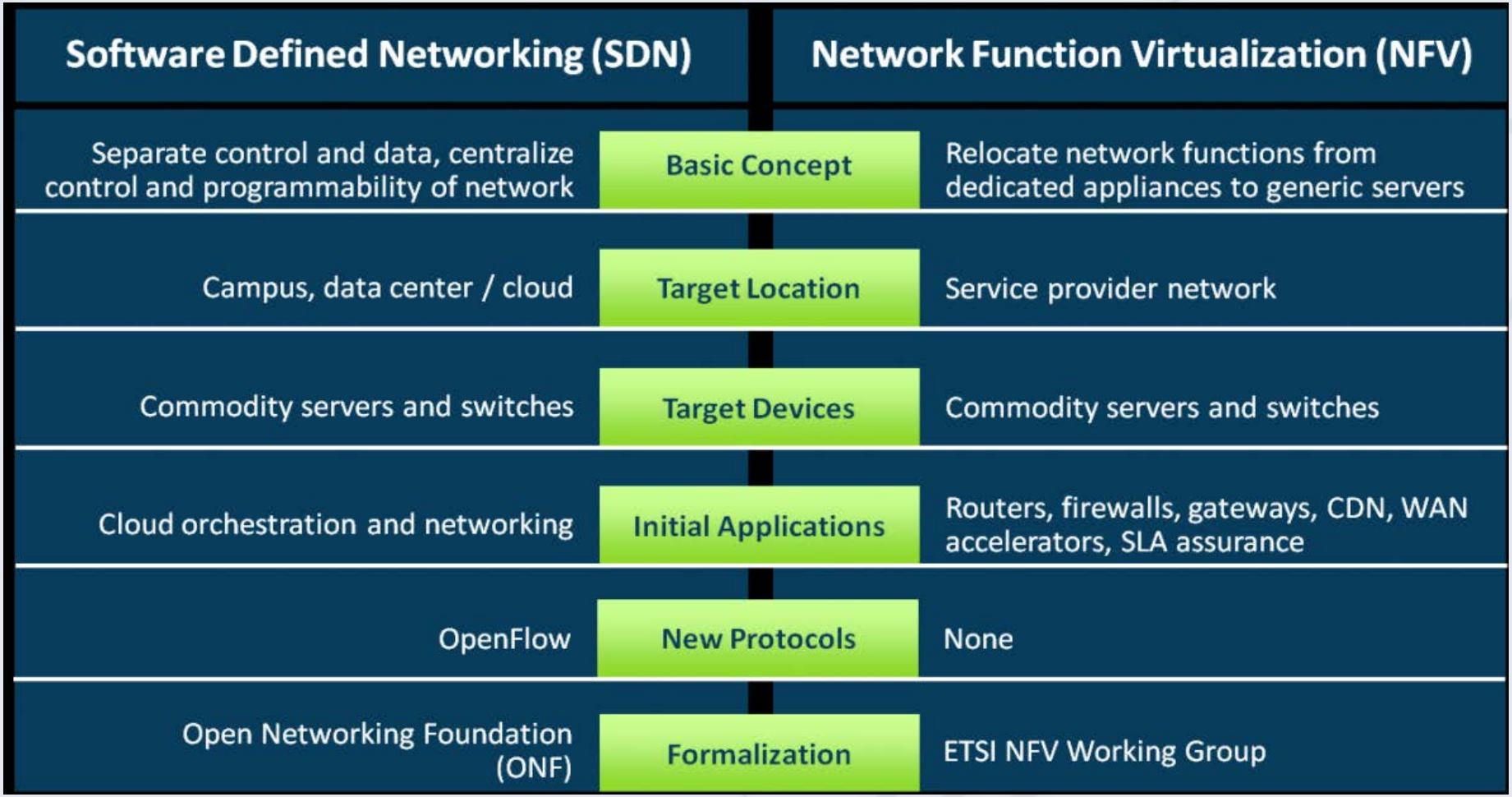
OpenFlow Switch Forwarding



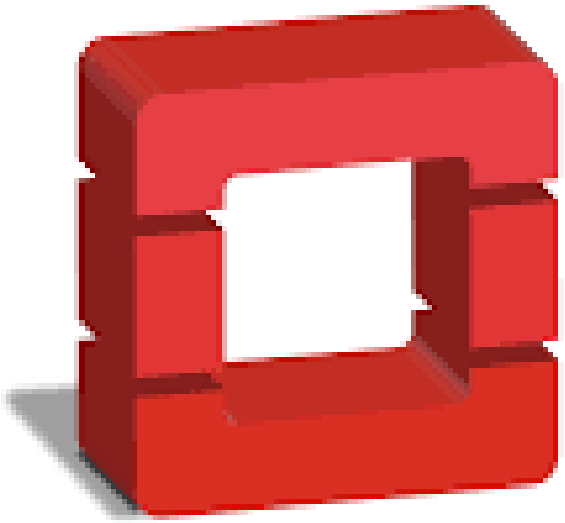
SDN / OpenFlow / NFV



NFV vs SDN



Source: Adopted from http://www.overturenetworks.com/blog/2013/04/12/network-function-virtualization-and-software-defined-networking-whats-difference

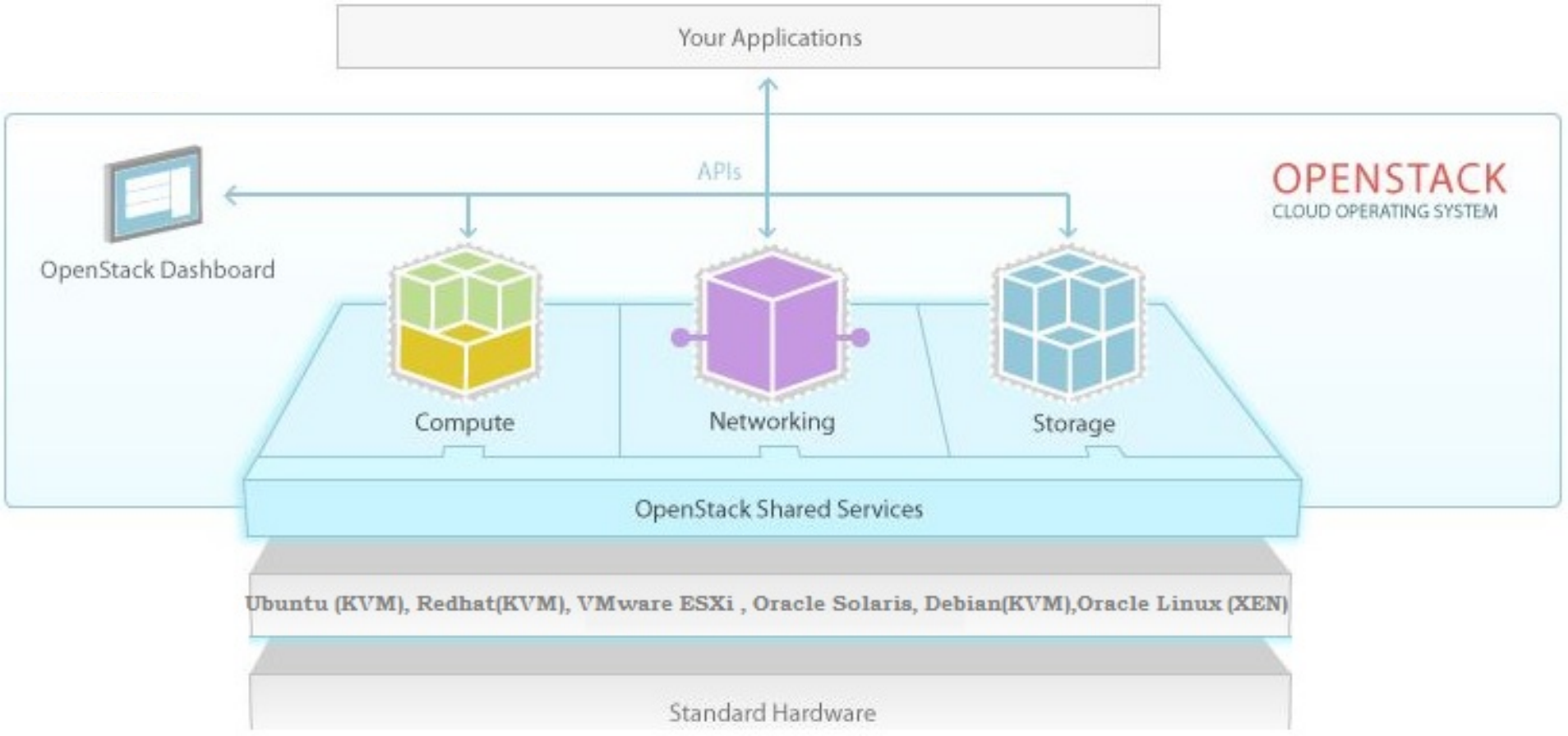


openstack®

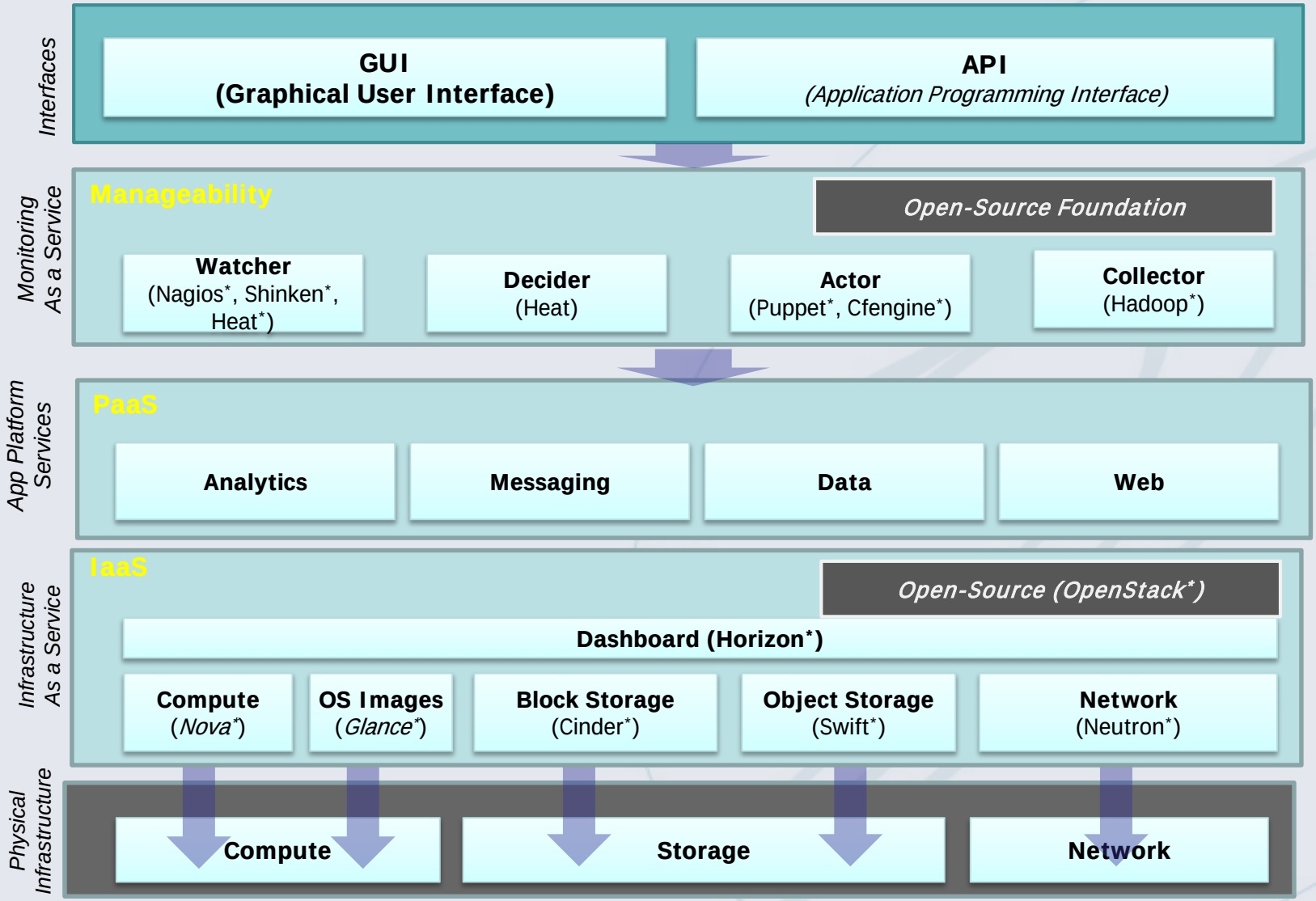
CLOUD SOFTWARE

OpenStack versions

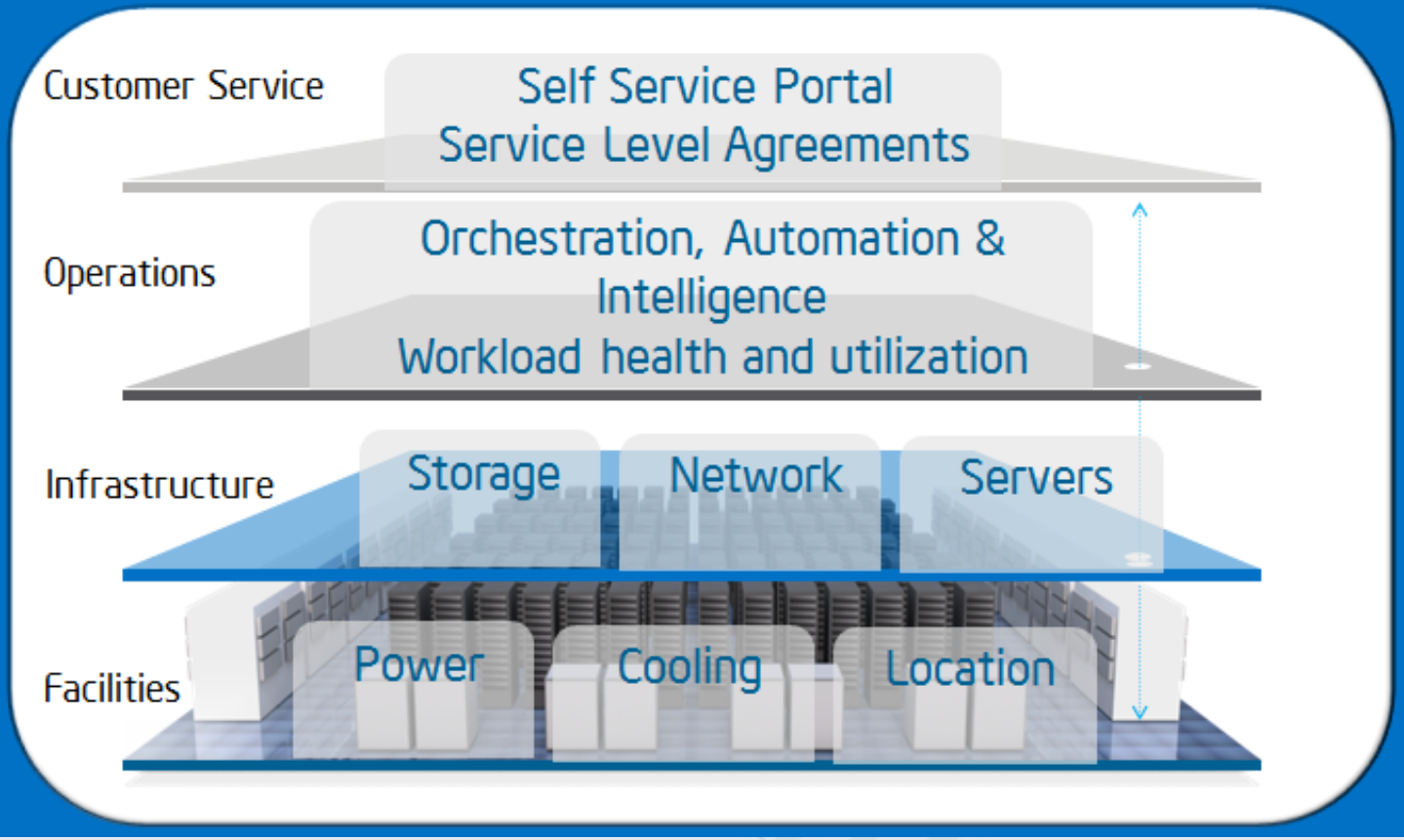
Release name	Release date	Included Component code names[45]
Austin	21 October 2010	Nova, Swift
Bexar	3 February 2011	Nova, Glance, Swift
Cactus	15 April 2011	Nova, Glance, Swift
Diablo	22 September 2011	Nova, Glance, Swift
Essex	5 April 2012	Nova, Glance, Swift, Horizon, Keystone
Folsom	27 September 2012	Nova, Glance, Swift, Horizon, Keystone, Quantum, Cinder
Grizzly	4 April 2013	Nova, Glance, Swift, Horizon, Keystone, Quantum, Cinder
Havana	17 October 2013	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer
Icehouse	17 April 2014	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer, Trove
Juno	16 October 2014	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer, Trove, Sahara
Kilo	30 April 2015	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer, Trove, Sahara, Ironic
Liberty	16 October 2015	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer, Trove, Sahara, Ironic, Zaqar, Manila, Designate, Barbican, Searchlight
Mitaka	7 April 2016	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer, Trove, Sahara, Ironic, Zaqar, Manila, Designate, Barbican, Searchlight, Magnum
Newton	6 October 2016	



Open Cloud Components



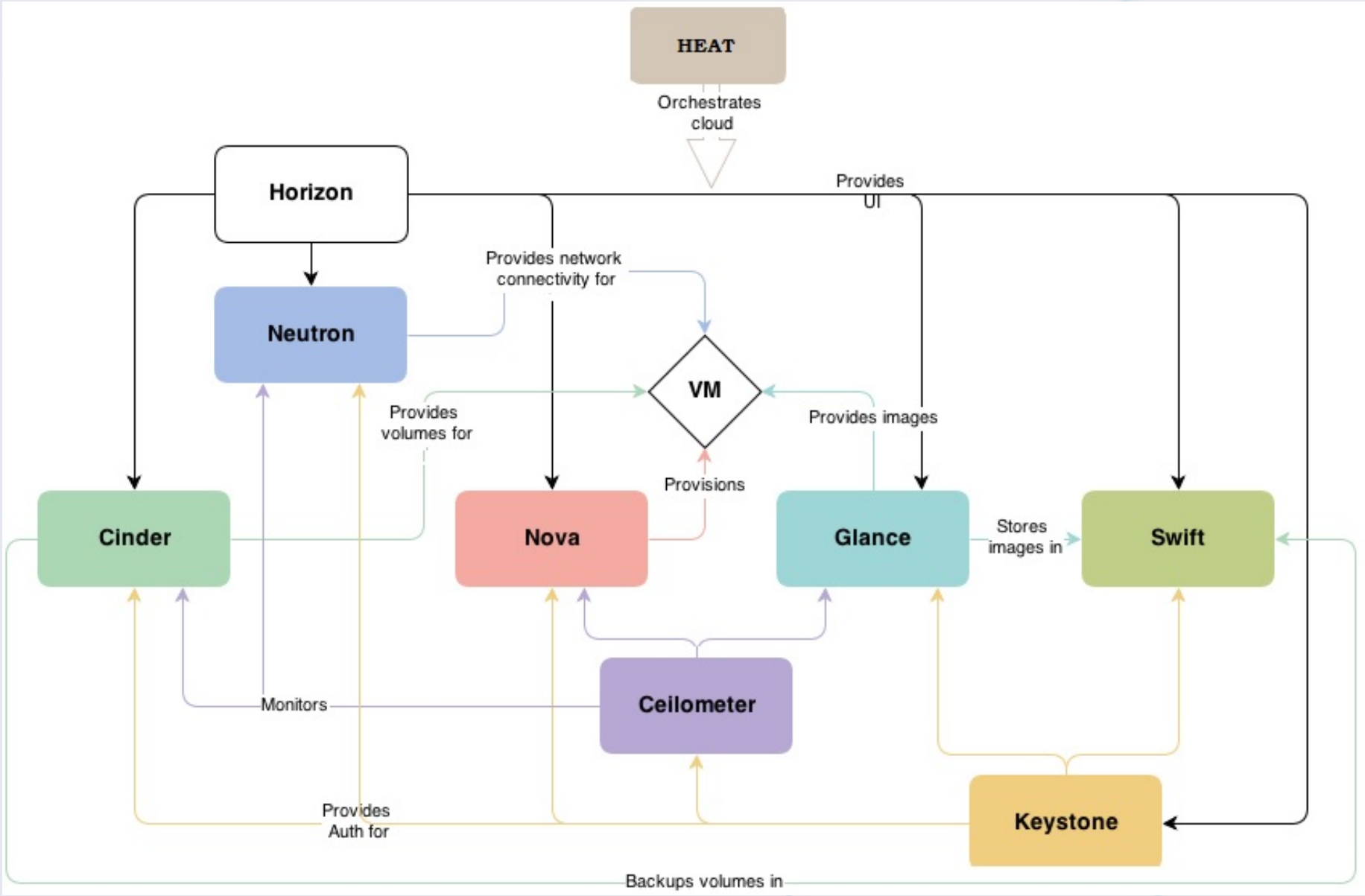
Intel Software Defined Infrastructure



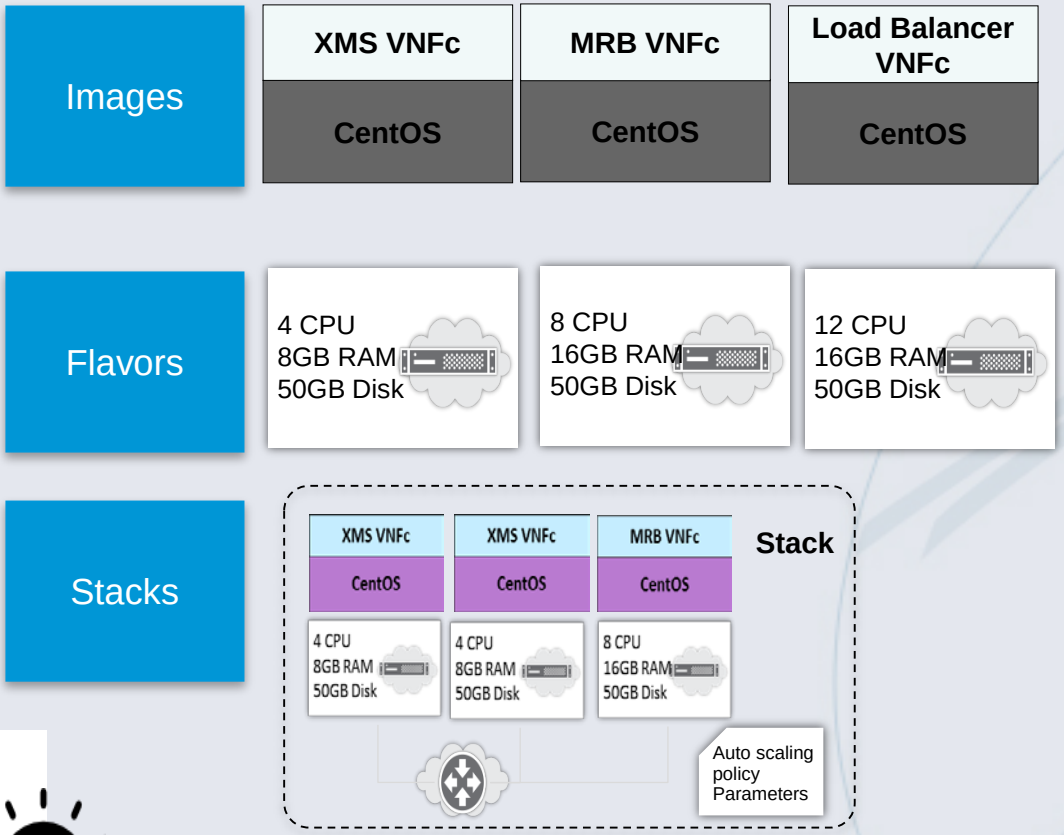
OpenStack components

Service	Project name	Description	Requirement
Dashboard	Horizon	Web-Based Dashboard	Mandatory
Compute	Nova	Create virtual Machine & manage VM	Mandatory
Networking	Neutron	Software defined networking (Advanced Networking)	Optional
Object Storage	Swift	Store files & Directories	Optional
Block Storage	Cinder	Volume & Snapshot Management	Mandatory
Identity service	Keystone	Creating Projects/User/Roles/Token Management/Authentication	Mandatory
Image Service	Glance	To Manage OS Images	Optional
Telemetry	Ceilometer	Monitoring & Billing purpose	Optional
Orchestration	Heat	HOT(Heat Orchestration Template) based on YAML	Optional
Database Service	Trove	Database as a Service	Optional
Hadoop as Service	sahara	Hadoop as Service	Optional
Messaging	RabbitMQ	Messaging	Mandatory

Conceptual architecture



Open Stack – A Cloud Operating System



Heat Templates contain info to create stacks

- Stack: Collection of resources
  - Virtual machines, networks, auto scaling rules
- Template: Definition of resources that make up the Stack
  - Four sections
    - **Resources** – Objects that will be created - like a server
    - **Properties** – image, flavor
    - **Parameters** – Property values
    - **Output** – Information passed back to user or dashboard

Resources

```
resources:
  <resource ID>:
    type: <resource type>
    properties:
      <property name>: <property value>
    metadata:
      <resource specific metadata>
    depends_on: <resource ID or list of ID>
    update_policy: <update policy>
    deletion_policy: <deletion policy>
    external_id: <external resource ID>
    condition: <condition name or expression or boolean>
```

```
resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      flavor: m1.small
      image: F18-x86_64-cfntools
```

```
resources:
  server1:
    type: OS::Nova::Server
    depends_on: server2

  server2:
    type: OS::Nova::Server
```

Parameters

```
parameters:
  <param name>:
    type: <string | number | json | comma_delimited_list | boolean>
    label: <human-readable name of the parameter>
    description: <description of the parameter>
    default: <default value for parameter>
    hidden: <true | false>
    constraints:
      <parameter constraints>
    immutable: <true | false>
```

```
parameters:
  user_name:
    type: string
    label: User Name
    description: User name to be configured for the application
  port_number:
    type: number
    label: Port Number
    description: Port number to be configured for the web server
```

OpenStack Community – 60+ companies

Demo

SDN vs. NFV

What is SDN?

- Software-defined networking (SDN) is the separation of the **Network Control Plane** from the **Forwarding Plane**, and where a control plane can control multiple devices.

  Open Network Foundation

- SDN is whatever we can ship today

  Vendor X

- SDN is the magic buzzword that will bring us VC funding

  Startup Y

What is SDN?

SDN Definition

Centralization of control of the network via the

Separation of **control** logic to off-device compute, that

Enables **automation** and **orchestration** of network services via

Open **programmatic** interfaces

SDN Benefits

Efficiency: optimize existing applications, services, and infrastructure

Scale: rapidly grow existing applications and services

Innovation: create and deliver new types of applications and services and business models

Source: Adopted from SDN Central (Software-Defined Networking (SDN) Use Cases)

Need for SDN

- Network Virtualization (Data Center & Cloud)– Use network resource without worrying about where it is physically located, how much it is, how it is organized, etc.
- Orchestration (Cloud) - Automated arrangement, coordination, and management of complex computer systems, middleware, and services.

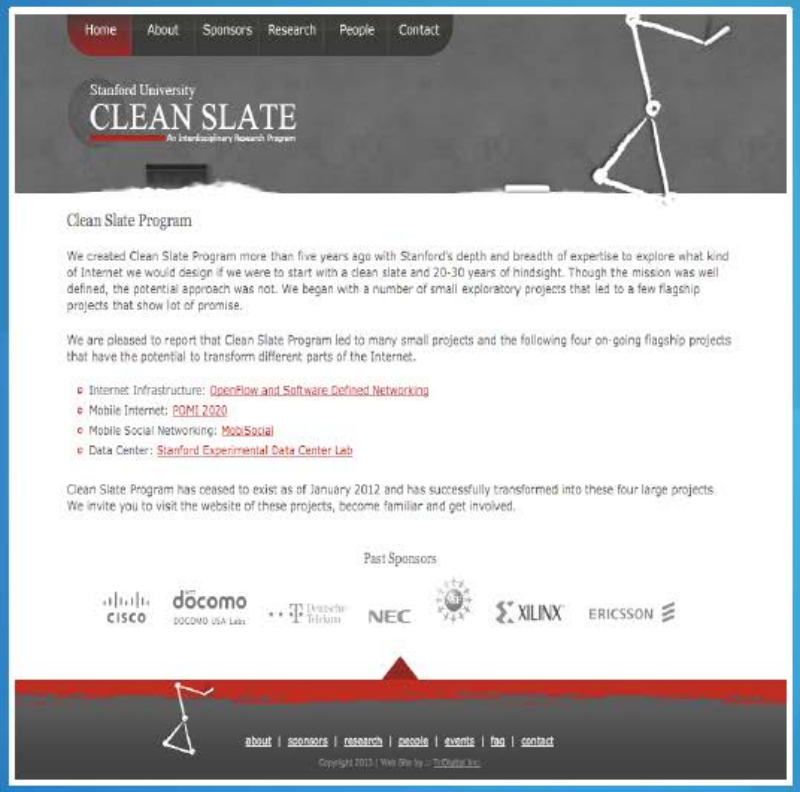
Source: Adopted from Introduction to Software Defined Software Defined Networking (SDN) Networking (SDN) by Prof. Raj Jain

Need for SDN (Contd..)

- Programmable (Enterprise) - Should be able to change behavior on the fly
- Dynamic Scaling (Cloud) - 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

Source: Adopted from Introduction to Software Defined Software Defined Networking (SDN) Networking (SDN) by Prof. Raj Jain

Where did SDN comes from

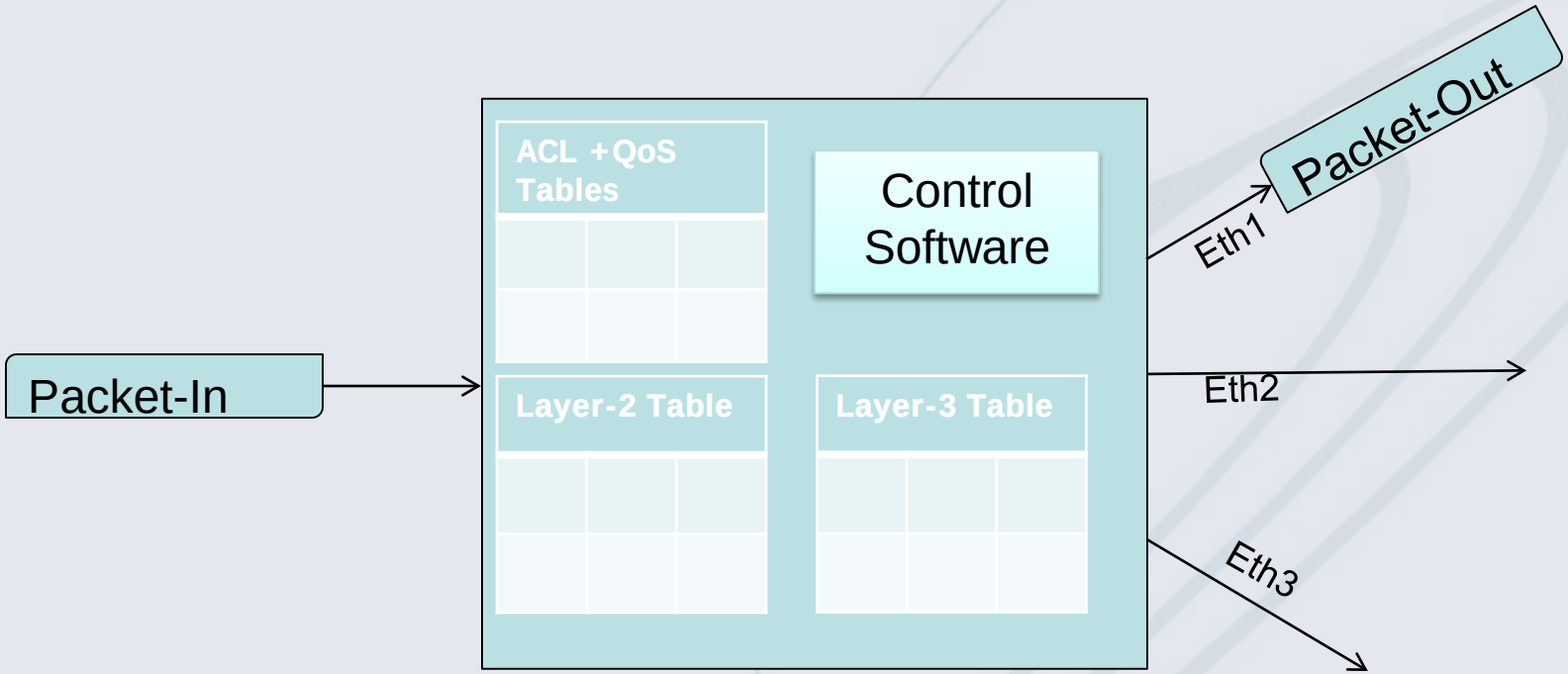


Stanford University – Clean Slate Project

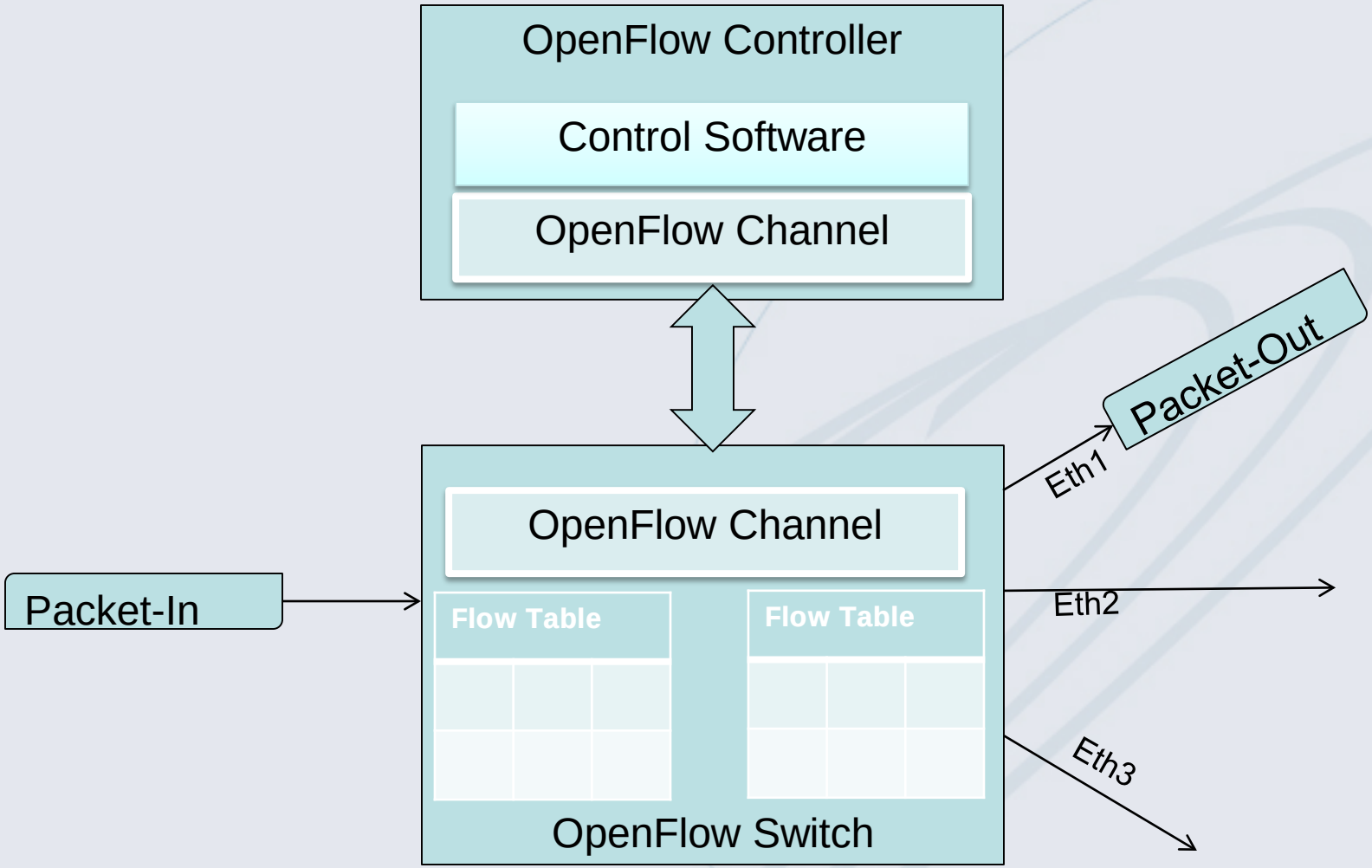
"…explore what kind of Internet we would design if we were to start with a clean slate and 20-30 years of hindsight."

http://cleanslate.stanford.edu/

Traditional Switch Forwarding

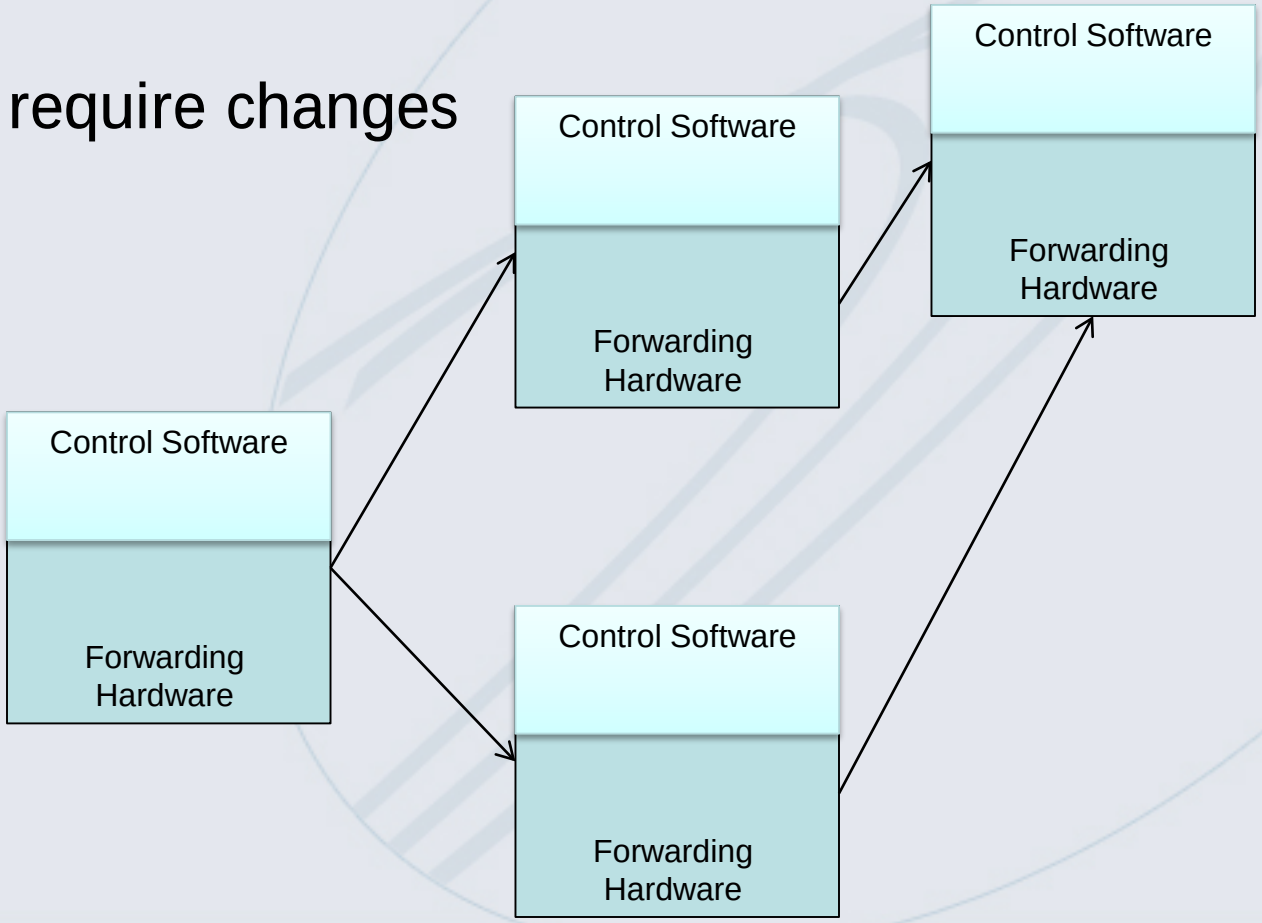


OpenFlow Switch Forwarding



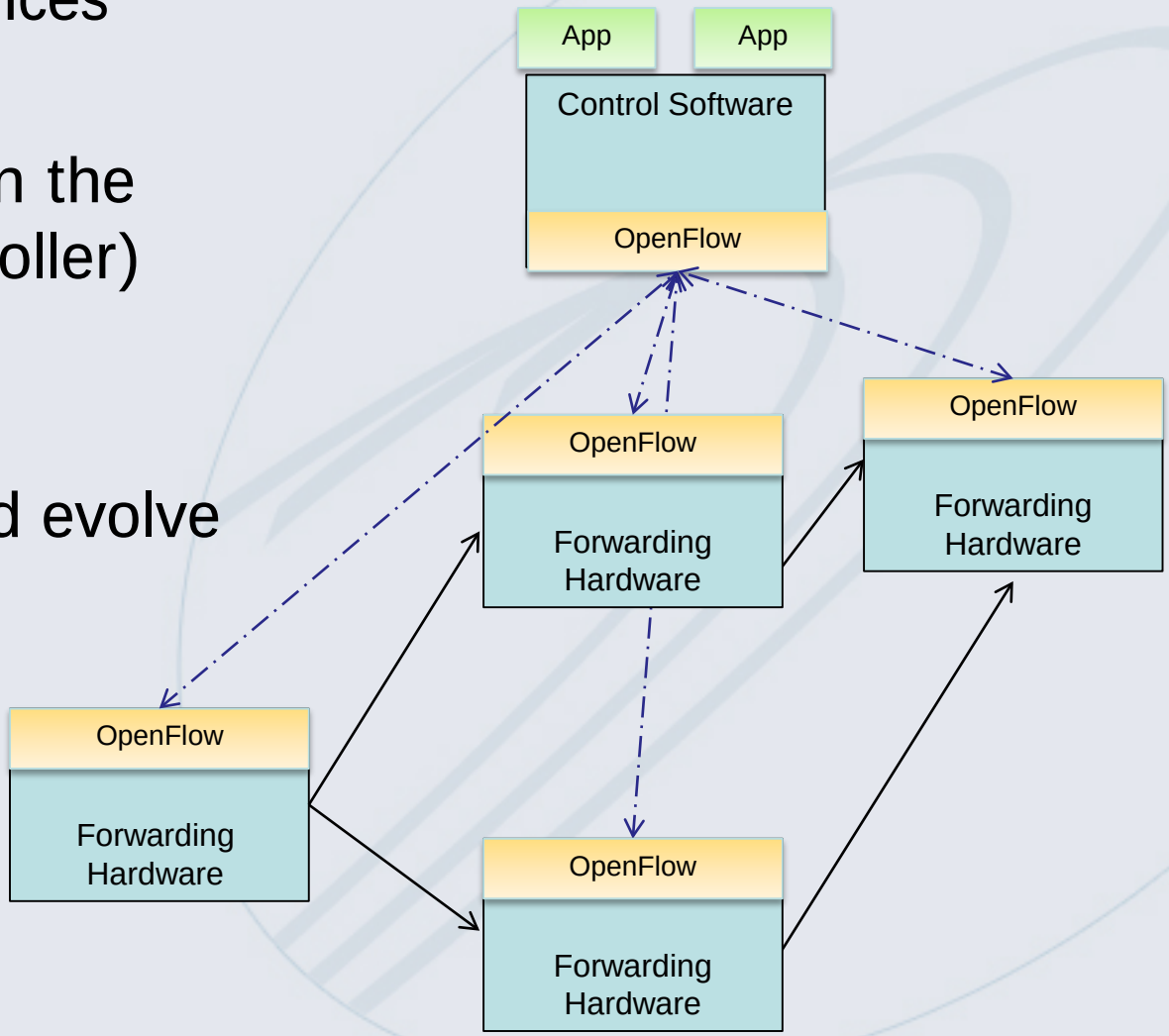
Traditional Network

- Each device in the network has a Control part
- No centralized control in the network
- New Applications require changes in all devices



OpenFlow Network

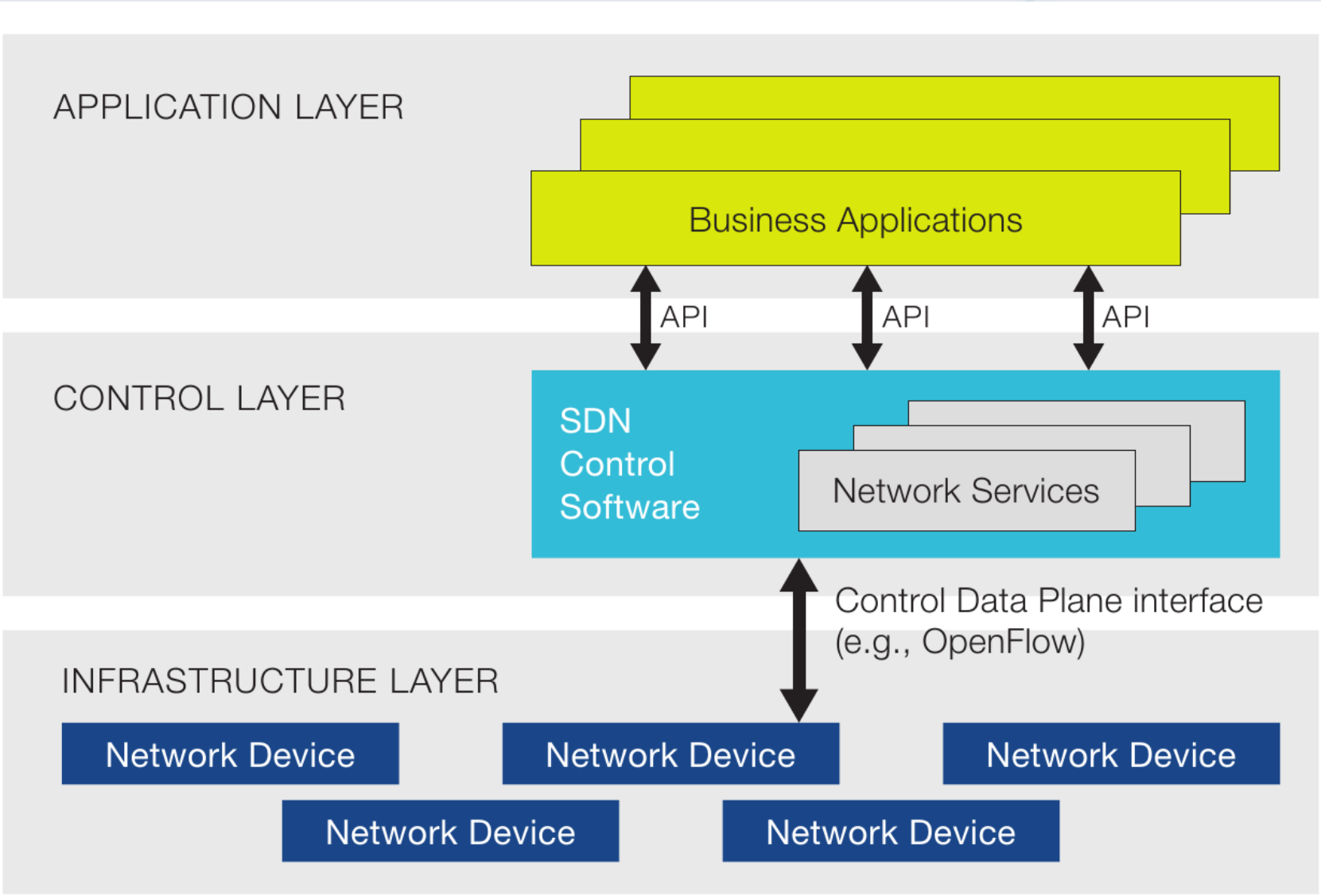
- Removing Control Software from forwarding devices
- Simplified devices
- Centralized control in the network (SDN Controller)
- Fully distributed enforcement
- Easy to innovate and evolve



SDN Advantages

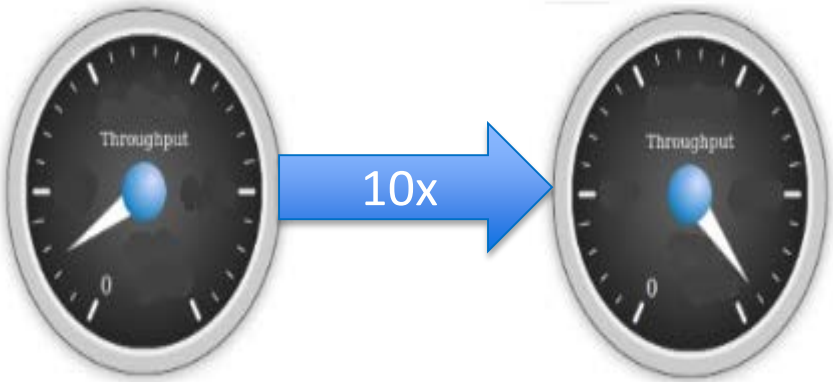
- Cheaper Hardware and cost saving
- Central Control
- Network Managed Dynamically
- Resource optimization
- Fault Tolerance handling
- Programming the network instead of configuring it

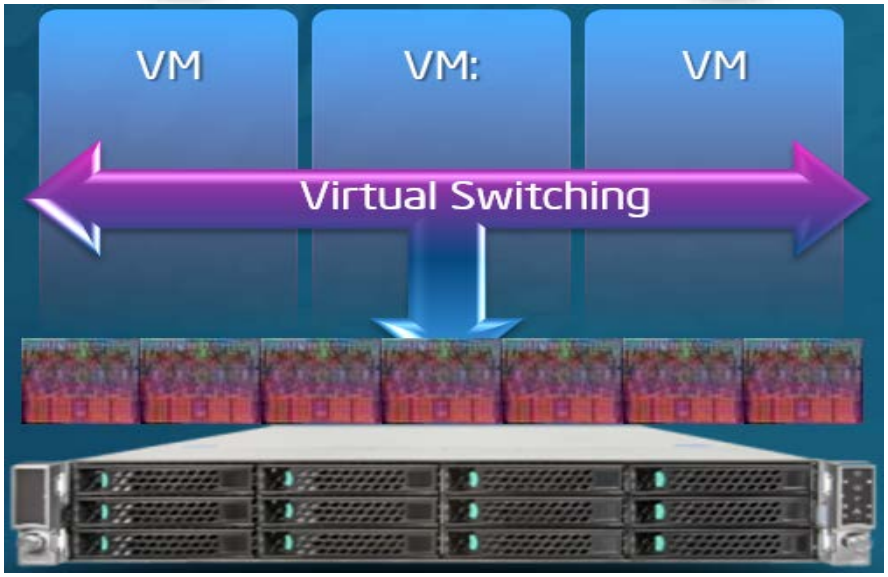
SDN Architecture



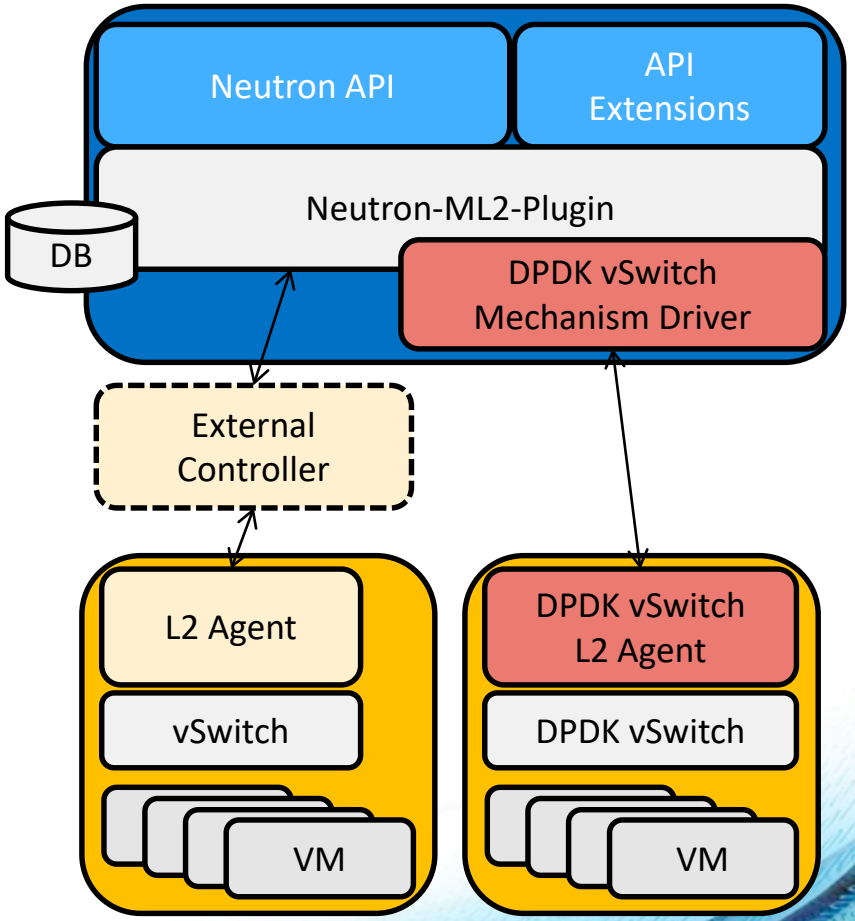
Intel® DPDK Accelerated Open vSwitch In Neutron

Open vSwitch Intel DPDK vSwitch





ML2 Driver/Agent in Development

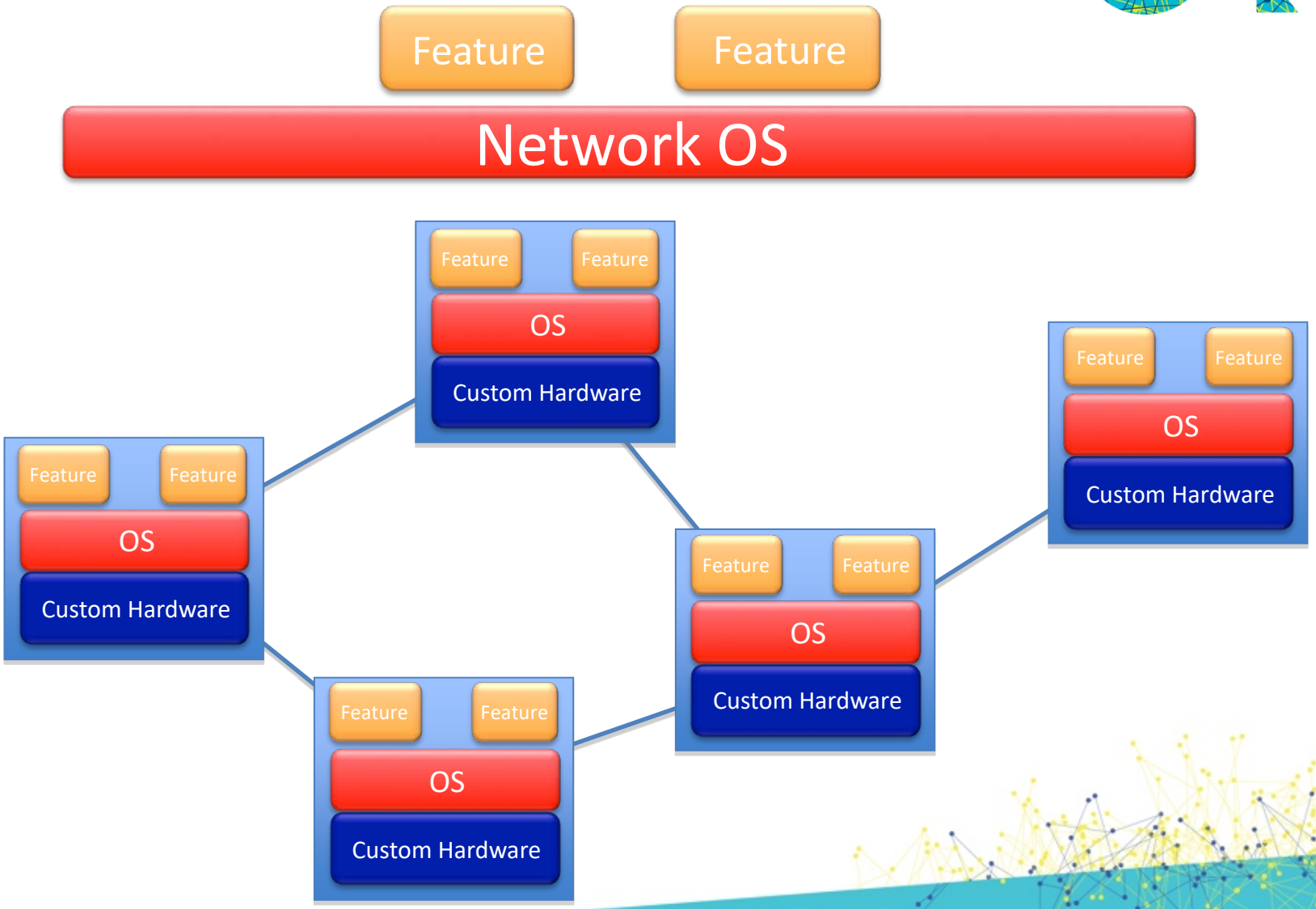


Unleashing Intel® DPDK vSwitch Performance in Neutron

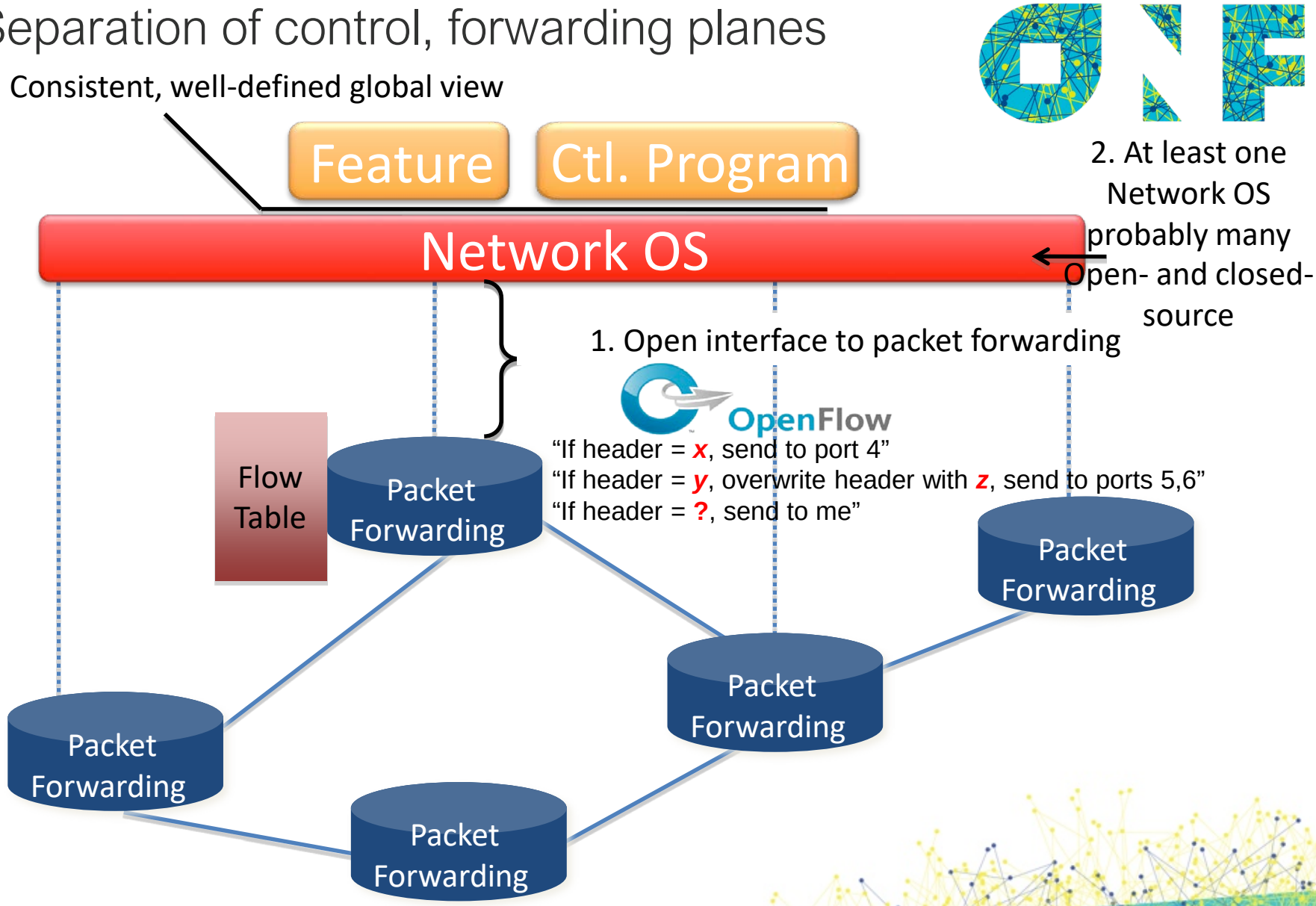
Need a standard - SDN

- Future Networks: a Programmable Network ?
  - Standardization in ONF
  - Standardisation in IETF
  - Standardisation in ITU-T
  - Standardisation in ETSI
  - Standardisation in 3GPP
  - Standardisation in ISO IEC JTC1

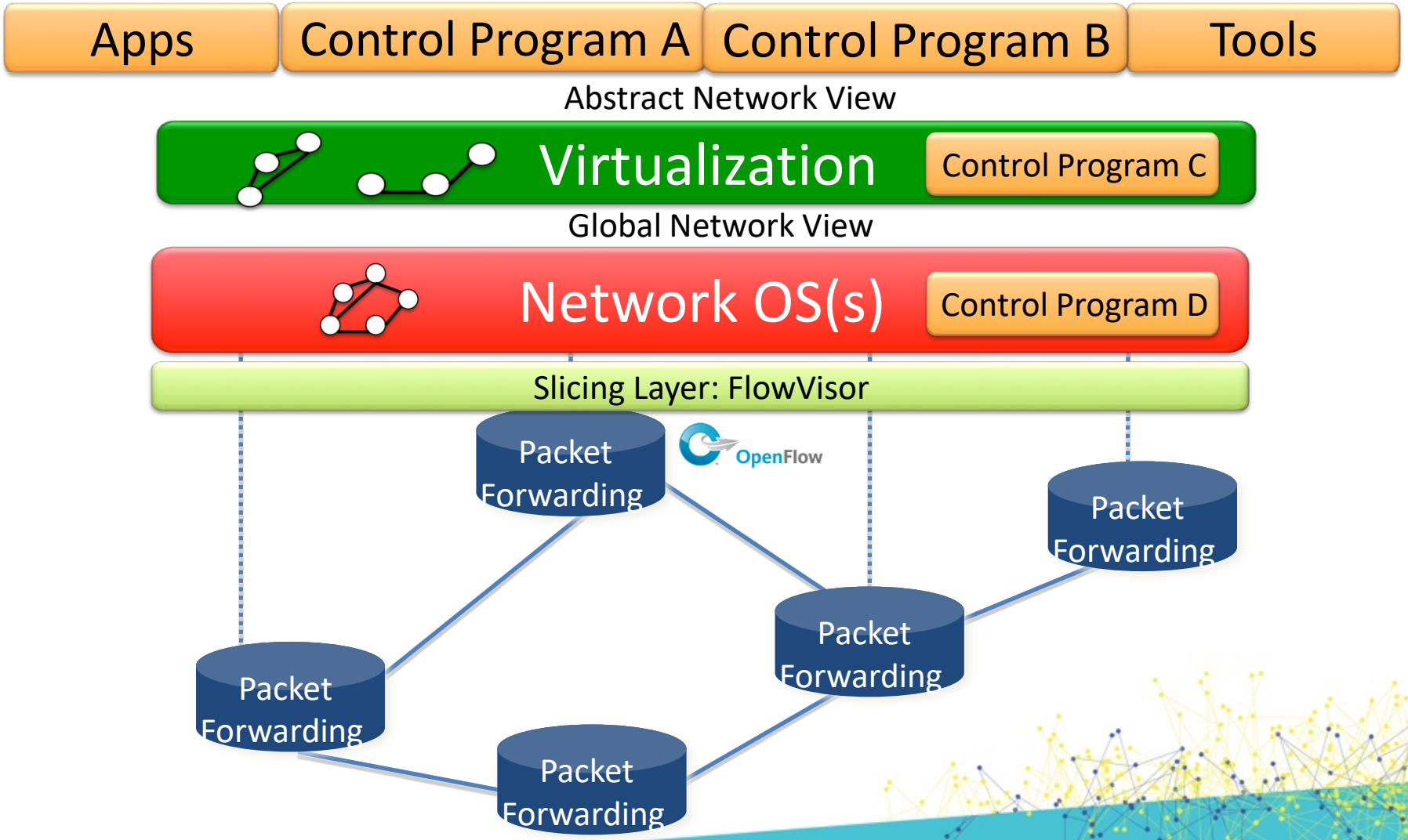
The transition



Separation of control, forwarding planes



Rich environment above OpenFlow



Future Networks: a Programmable Network ?

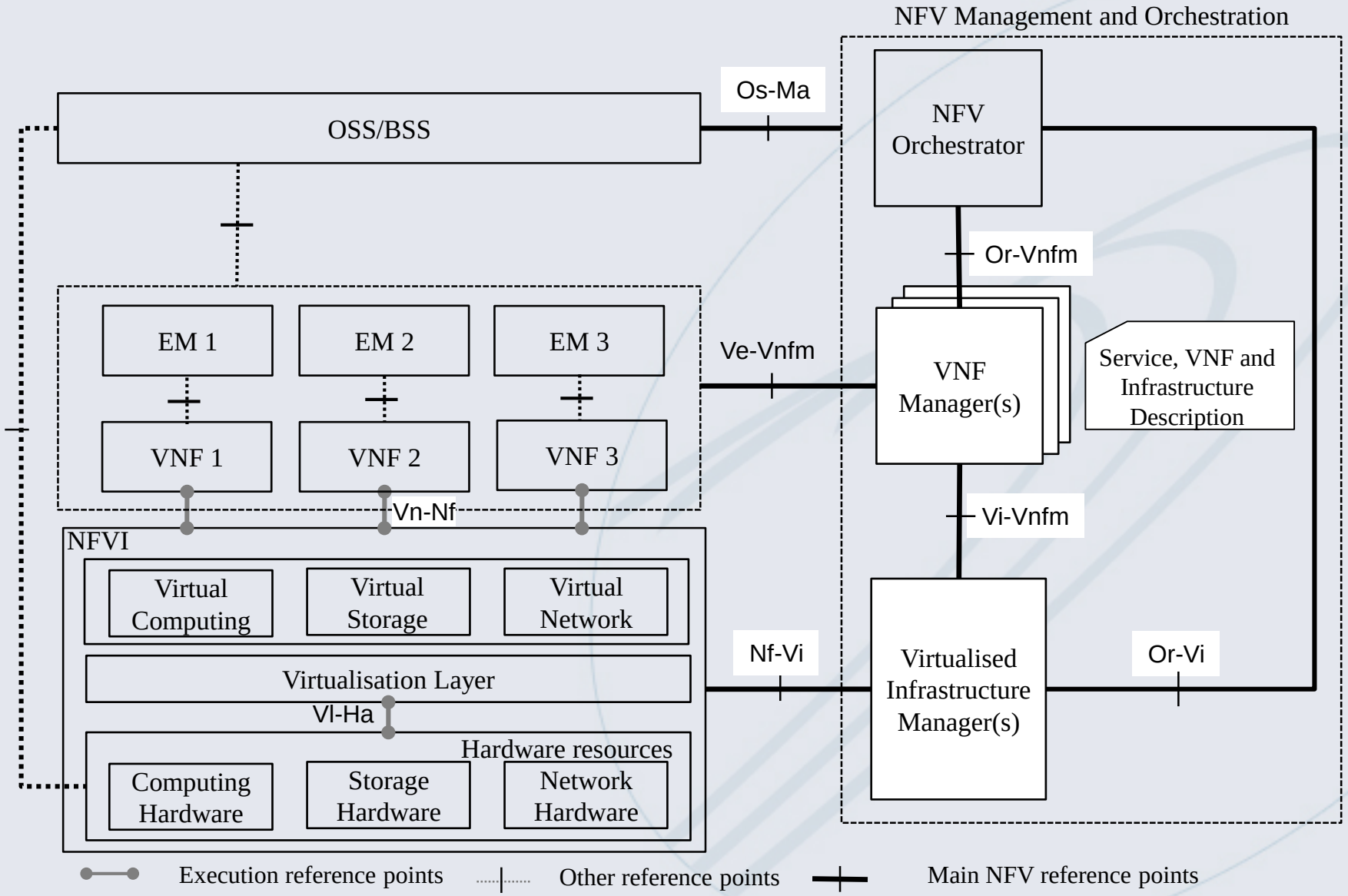
- **Several solutions and Terminologies**
  - SDN: Software-Defined Networking
    - Introduced by the New initiative ONF and recently by ITU-T SG 13 Future Networks
    - *Under discussion at* IETF as Network Programmability (or *Software-Driven Networks)*
  - Self Organizing & Autonomic Networks
  - Network resources and policy controller
  - Network Virtualization & slicing
  - Cloud Network & Network as a Services
  - …. Smart Ubiquitous & Distributed Services, Information-Centric Networks….
- **Opportunity for telecom operators:**
  - To hide network complexity by abstraction layer
  - Improve "Dynamically" network Management 'Programmability' & performance
  - Ability to deliver "On demand" network resources
- …And some use cases: Bandwidth On Demand, Network virtualization , Policy control, Chained Business services , Cloud Network, NaaS, Traffic Offload…

Practical Considerations

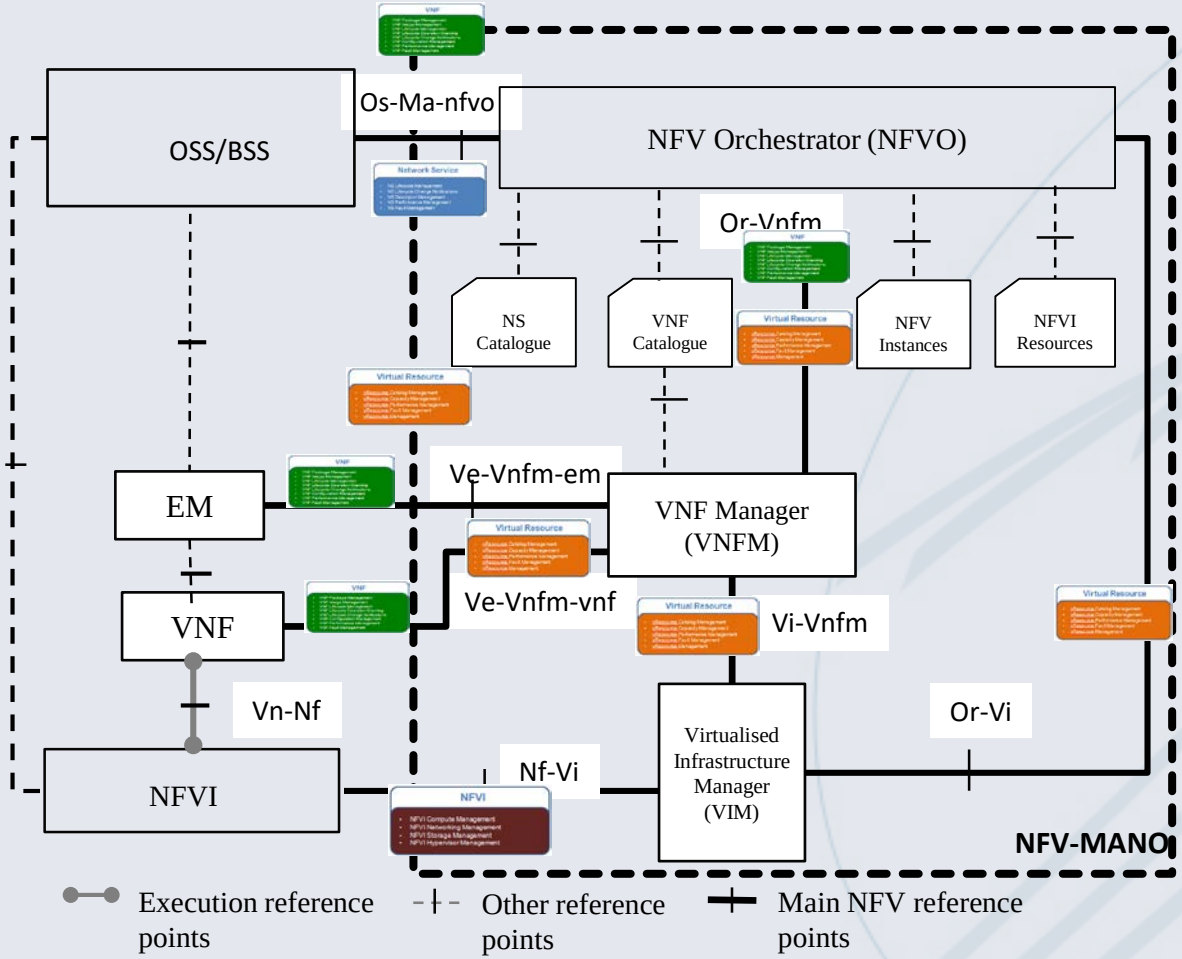
- Cloud Development is slightly different from traditional in house model.
- Everything is virtualized (most of the time)
- Everything is distributed
- Per instance reliability is much lower
- Overall reliability is much higher

MANO

ETSI NFV002 Architectural Framework



MANO Architectural Framework - Reference points and Interfaces



Network Service

- NS Lifecycle Management
- NS Lifecycle Change Notifications
- NS Descriptor Management
- NS Performance Management
- NS Fault Management

VNF

- VNF Package Management
- VNF Image Management
- VNF Lifecycle Management
- VNF Lifecycle Operation Granting
- VNF Lifecycle Change Notifications
- VNF Configuration Management
- VNF Performance Management
- VNF Fault Management

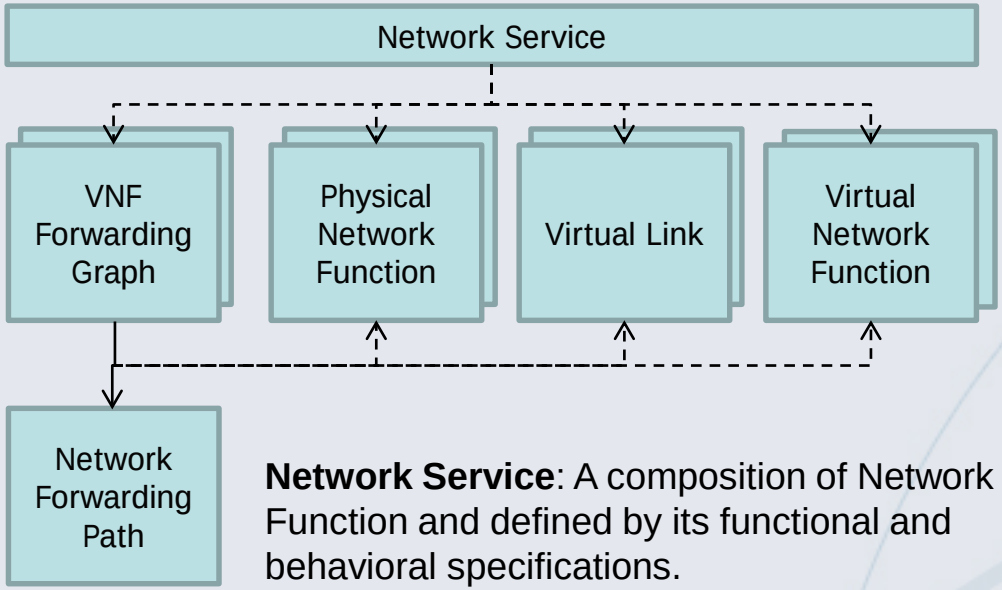
NFVI

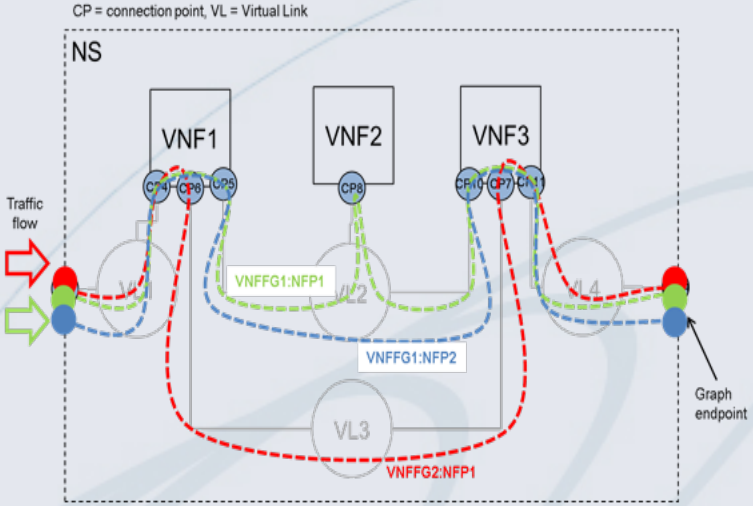
- NFVI Compute Management
- NFVI Networking Management
- NFVI Storage Management
- NFVI Hypervisor Management

Virtual Resource

- vResource Catalog Management
- vResource Capacity Management
- vResource Performance Management
- vResource Fault Management
- vResource Management

MANO Information elements





Network Service: A composition of Network Function and defined by its functional and behavioral specifications.

Virtualized Network Function: An implementation of NF that can be deployed on a Network Function Virtualization Infrastructure.

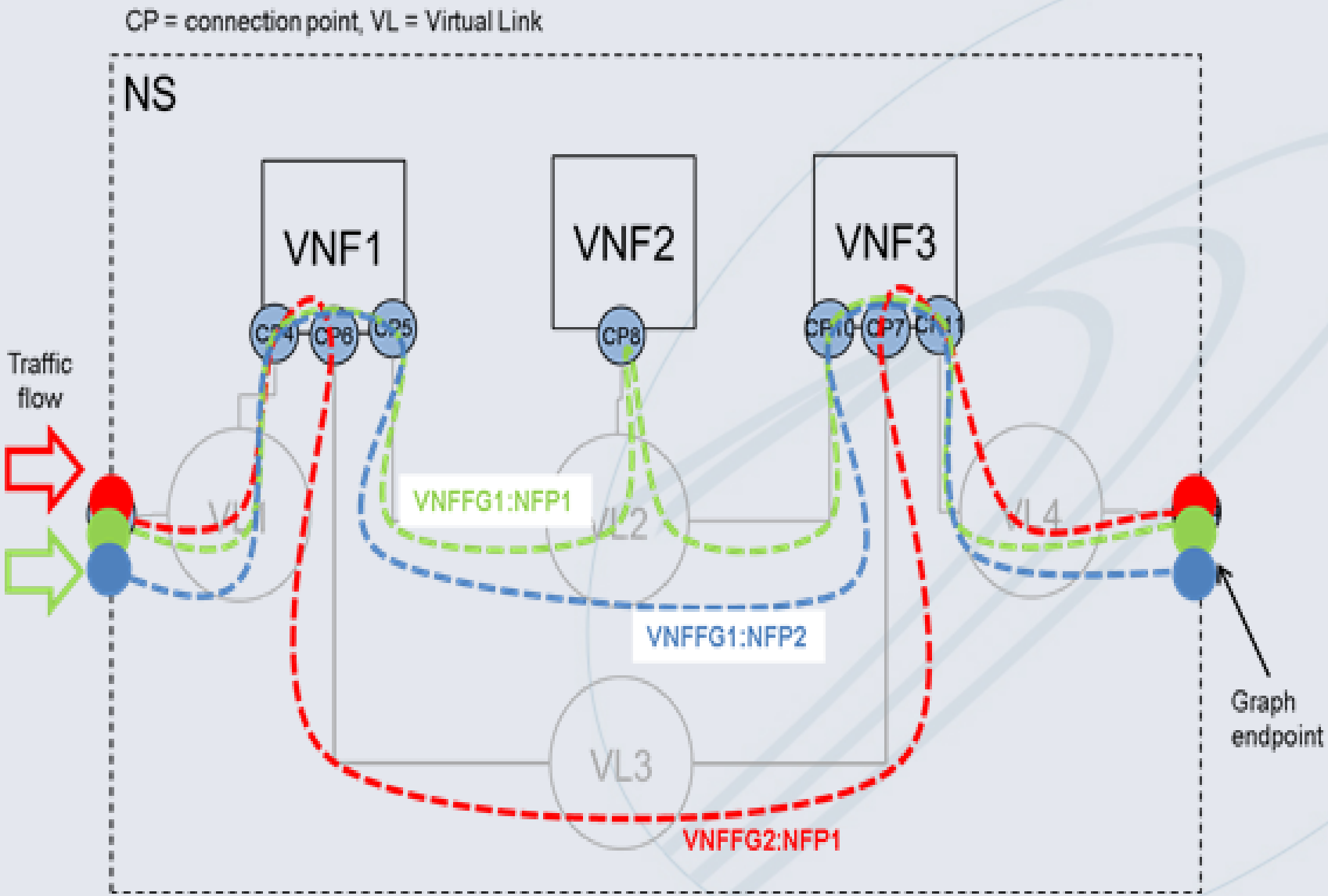
Virtual Link: Describes the resource requirements that are needed for a link between VNFs, PNFs and endpoints of the Network Service. Type of VL considered were E-LINE, E-LAN, E-TREE

VNF Forwarding Graph: Describes a topology of the Network Service or a portion of the Network Service, by referencing VNFs and PNFs and Virtual Links that connect them

Network Forwarding Path: Describes a sequence of NF to be traversed for a particular service in side the forwarding graph.

MANO uses descriptor/template mechanism to enable orchestration

NS Graph



MANO Descriptors

NSD
Id
Vendor
Version
vnfd
Vnffgd
Vld
Lifecycle_event
Vnf_depedency
Monitoring_parameters
Deployment_flavour
Auto_scale_policy
CP
pnfd

VNFD
vendor
descriptor_version
version
vdu
virtual_link
connection_point
lifecycle_event
dependency
monitoring_parameter
deployment_flavour
auto_scale_policy
manifest_file
manifest_file_security

VDU
id
vm_image
computation_requirement
virtual_memory_resource_element
virtual_network_bandwidth_resource
lifecycle_event
constraint
high_availability
scale_in_out
vnfc
monitoring_parameter
id
vm_image

vnfc
id
connection_point

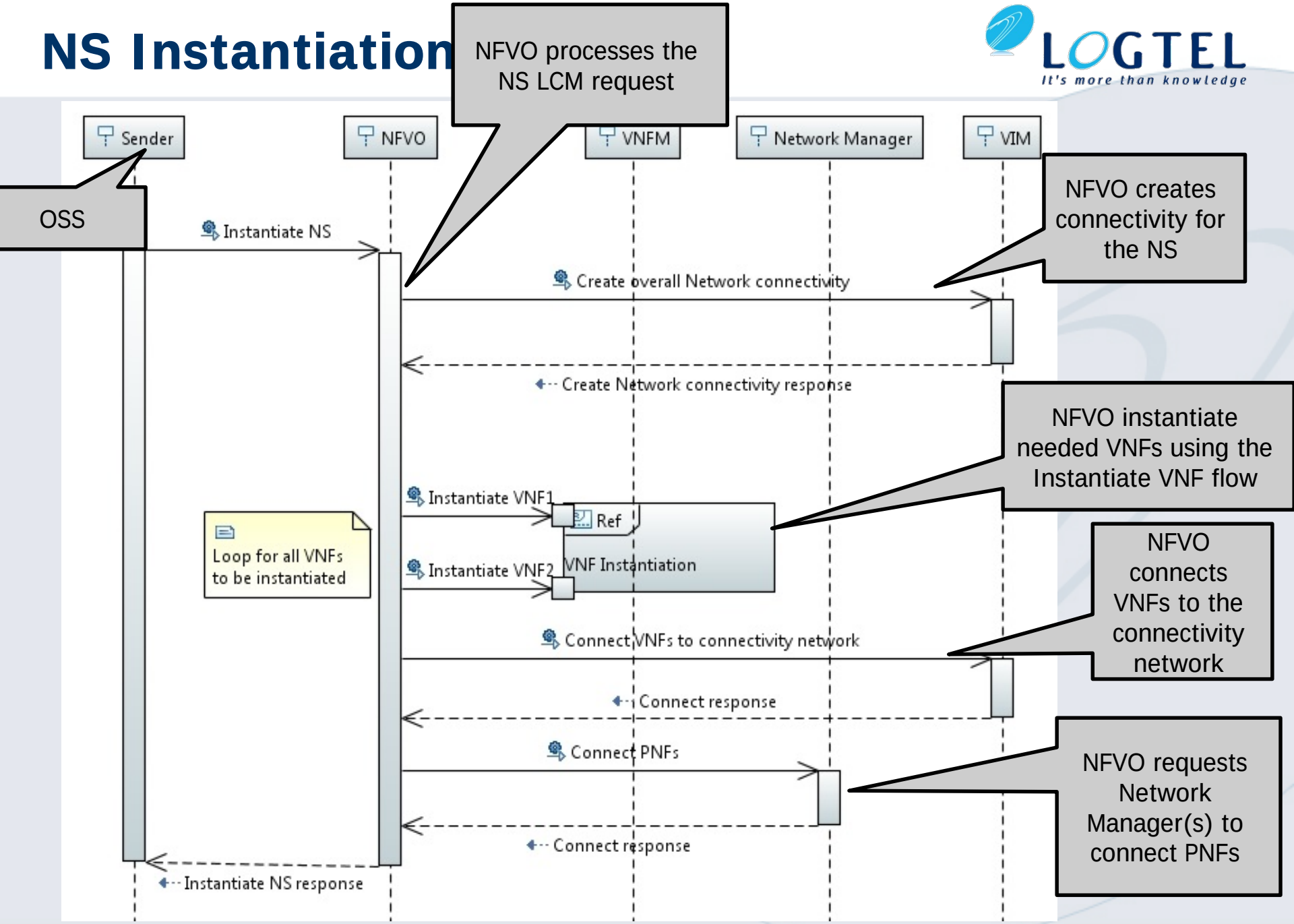
VNFFGD
Id
Vendor
Descriptor_version
Version
Constituent_vnf
Cp
Nfp
Depedent_virtual_link
Number _of_endpoints
number_of_virtual_lin k

VLD
Id
Vendor
Version
Root_requirements
Leaf_requirements
Qos
connection
Connectivity_type
Test_access

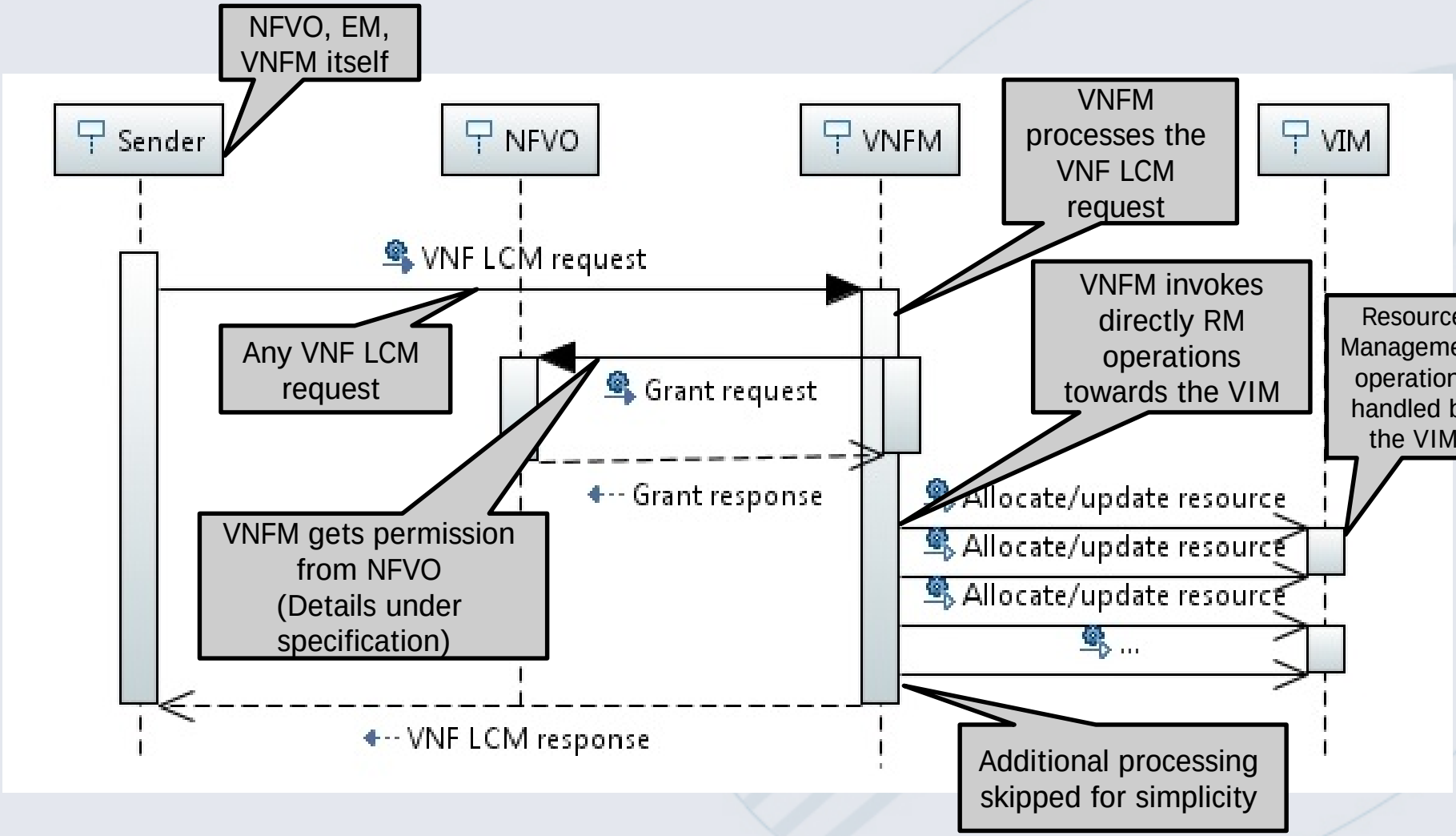
NFP
Id
policy
connection

MANO information elements describe features for all MANO constructs

NS Instantiation



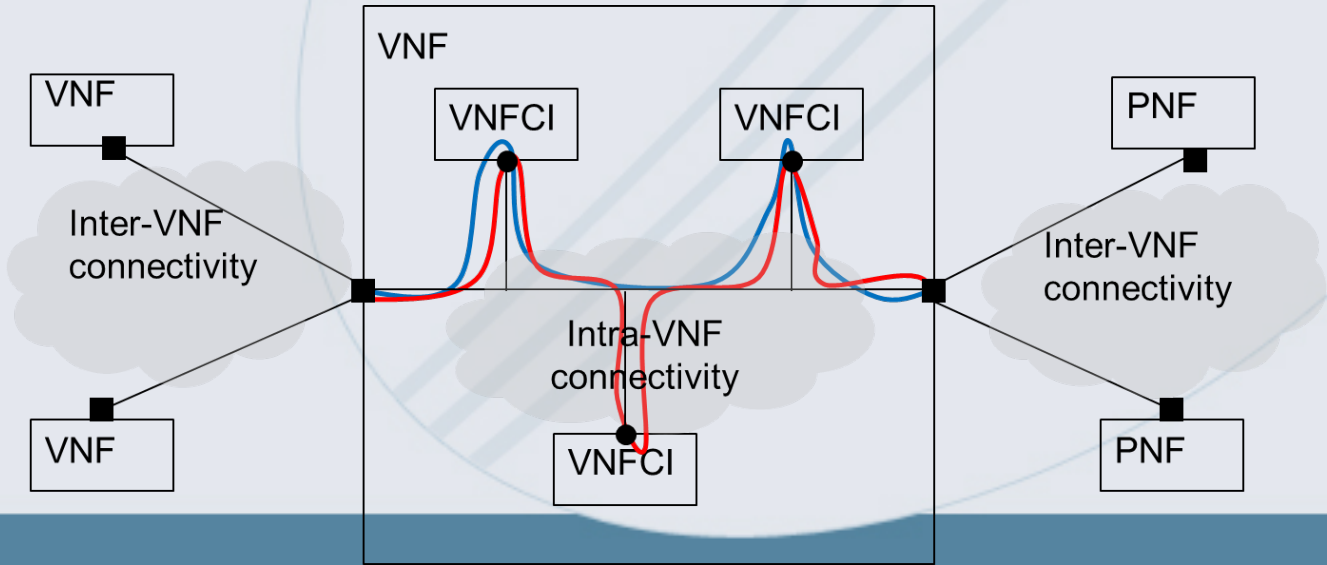
VNF related resource management done by VNFM (Direct mode)



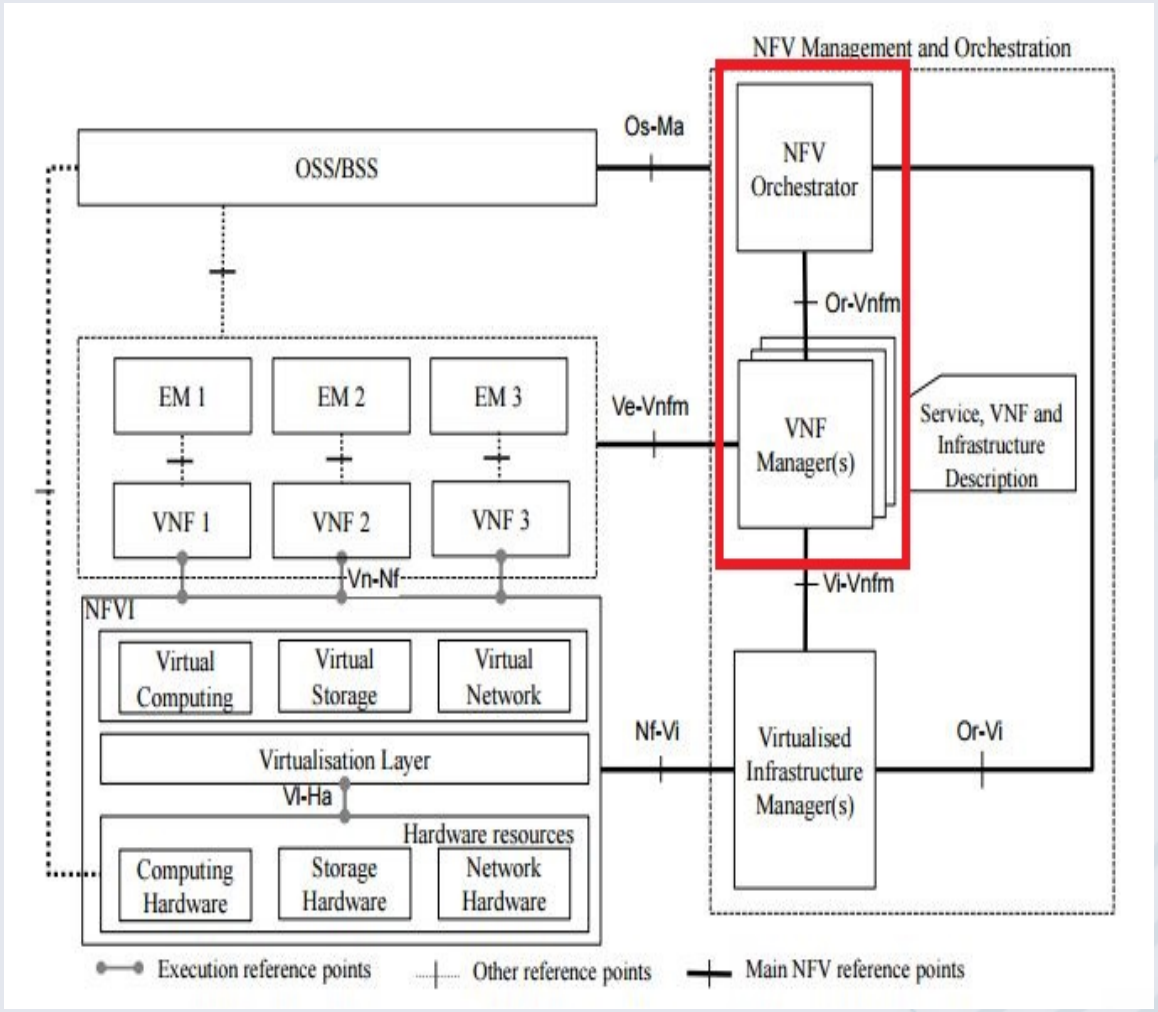
Simplified flow, focusing only on Resource Management

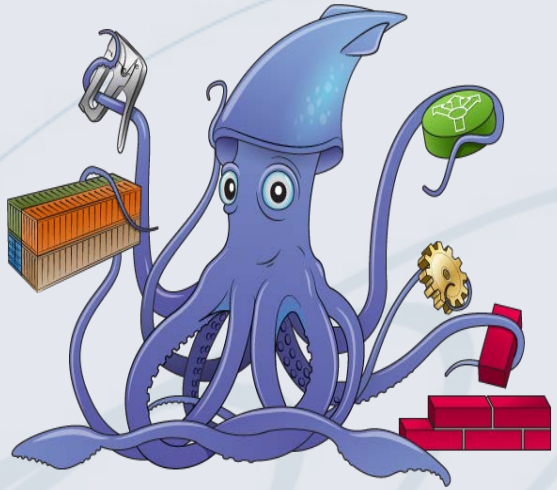
Acceleration Interfaces

- High performance networking
  - Cryptography, IPSec offloading, TCP offloading, Packet dispatching
- High performance computing
  - GPU computing, RDMA
- High performance Storage
  - NVMe, ISCSI, other persistent memory schemes
- Dynamic Optimization of Packet Flow Routing
  - Optimize intra-VNF packet forwarding and routing



What is Tacker?





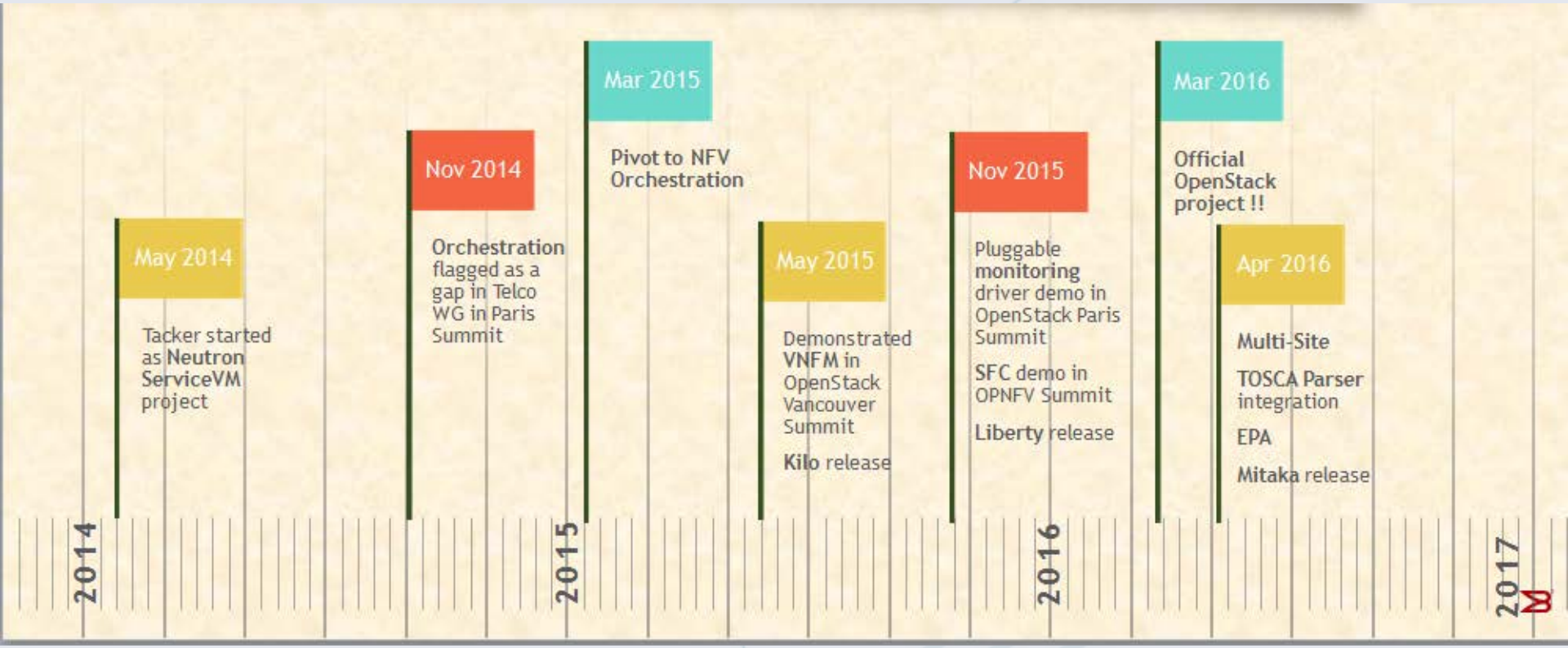
Tacker

is an official OpenStack project for NFV Orchestration and VNF anagement using standards based architectures

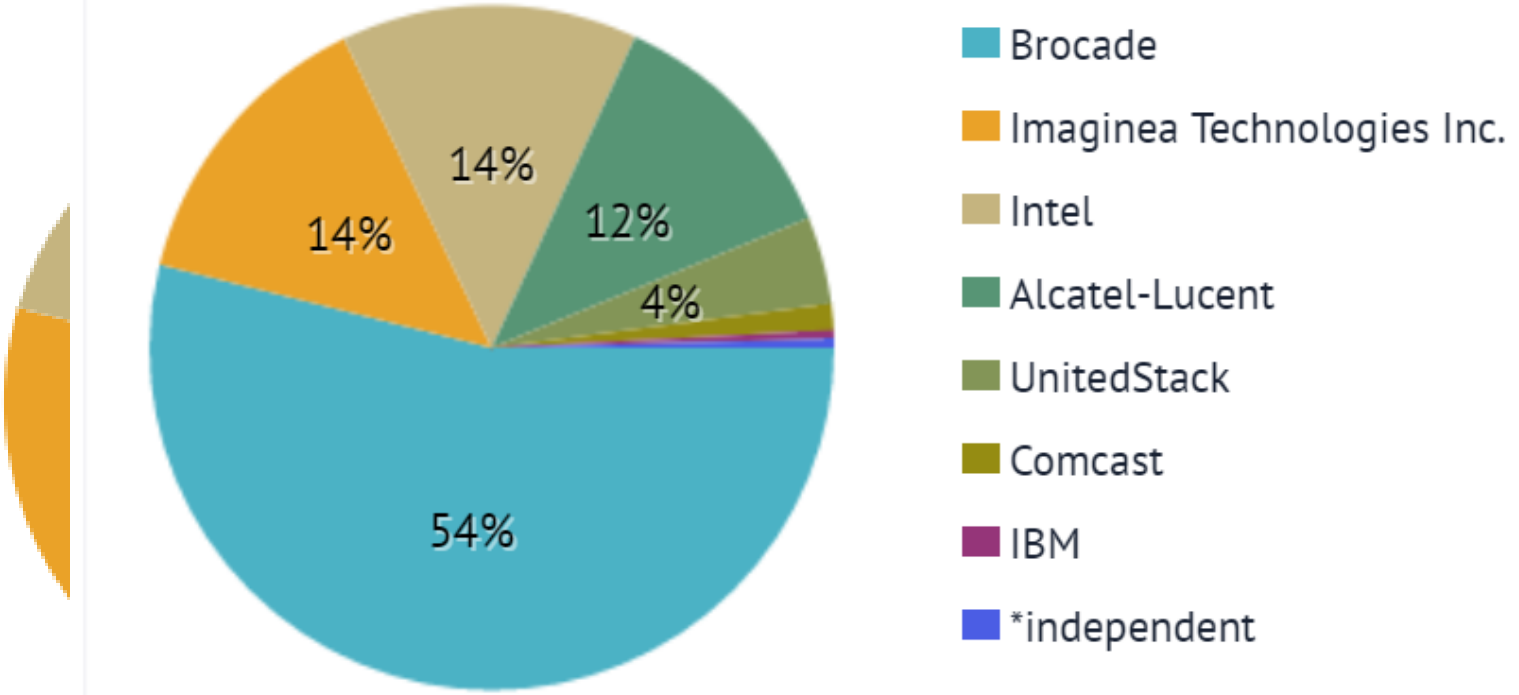
Tacker Project Overview

- Pivoted to NFV Orchestration early 2015
- Announced in OpenStack Vancouver Summit – May, 2015
- Packed sessions in OpenStack Tokyo Summit –
  - BoF, vBrownBag, Marketplace demo (vEPC, vRouter)
- Wrapping up its third release

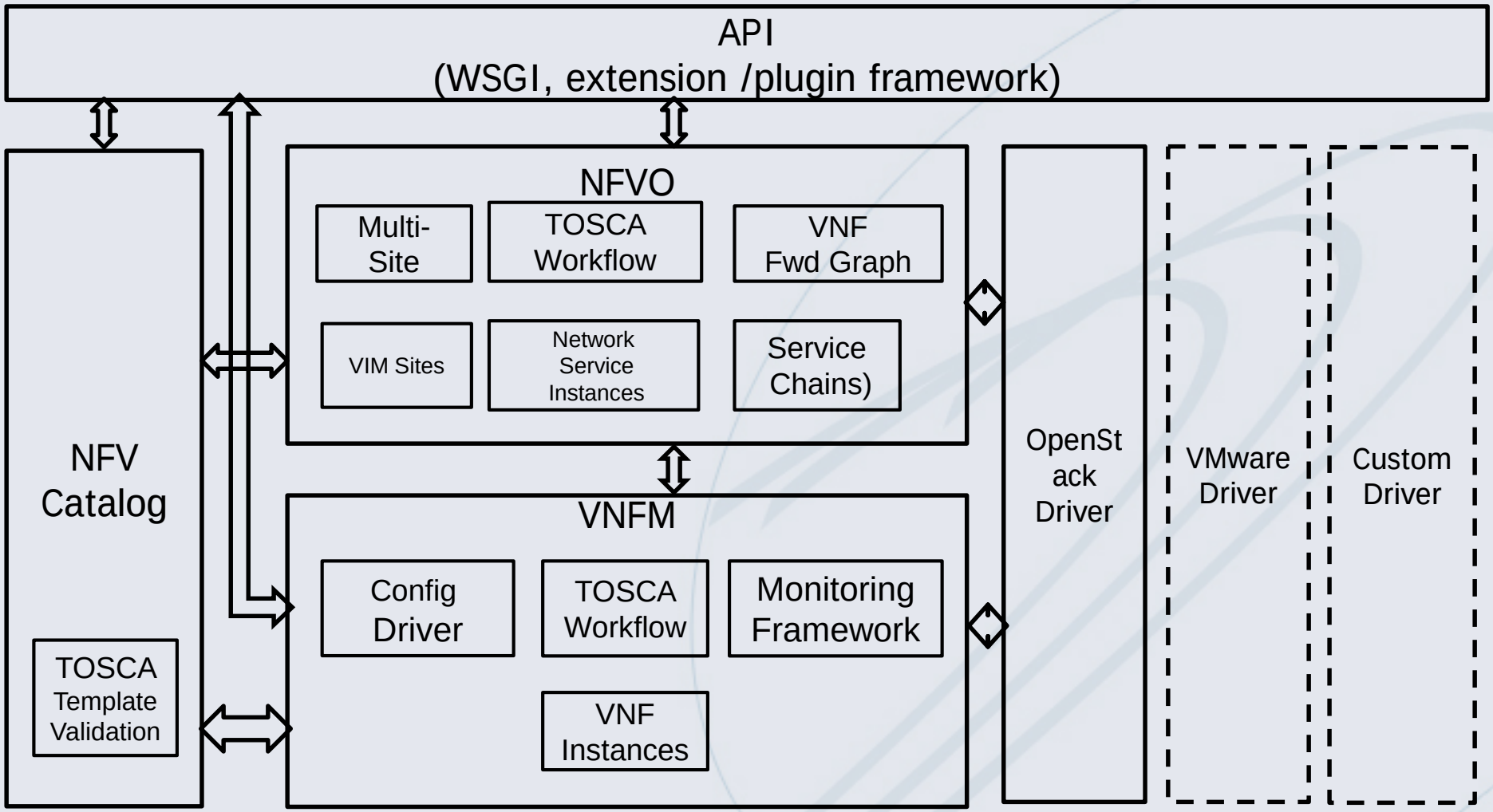
Tacker Project Timeline



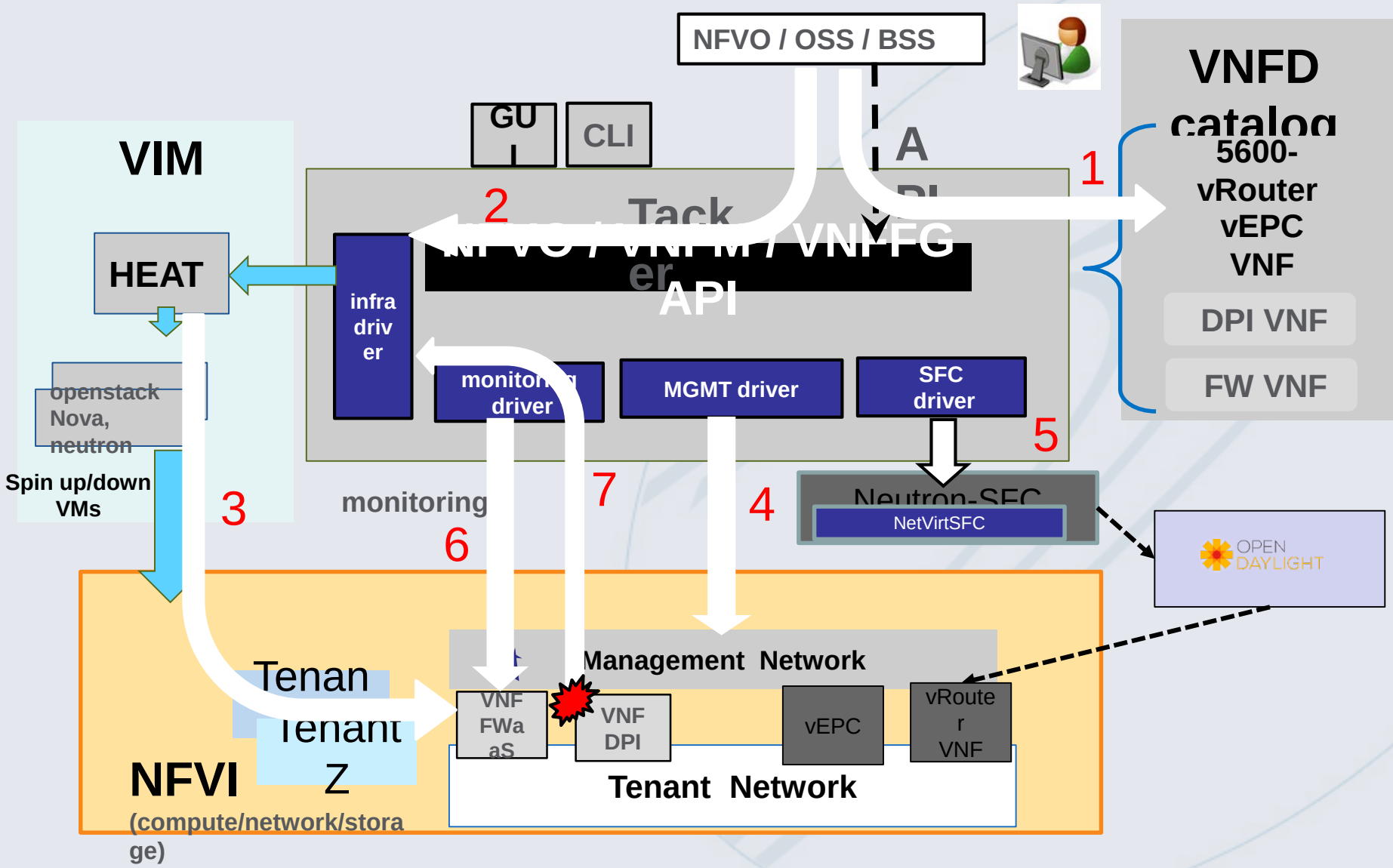
Tacker Stackalytics - Liberty



Tacker Architecture

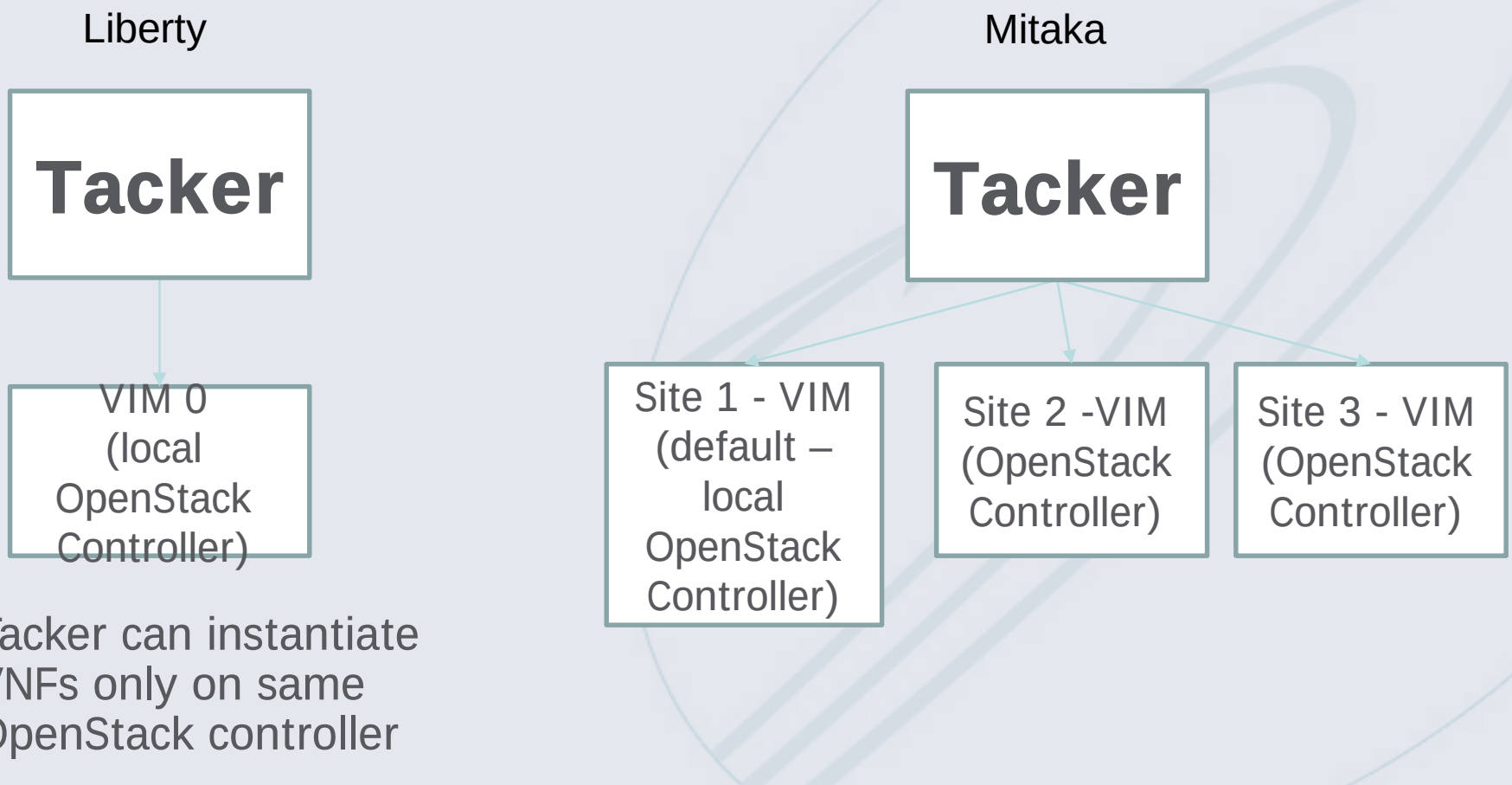


Tacker Workflow



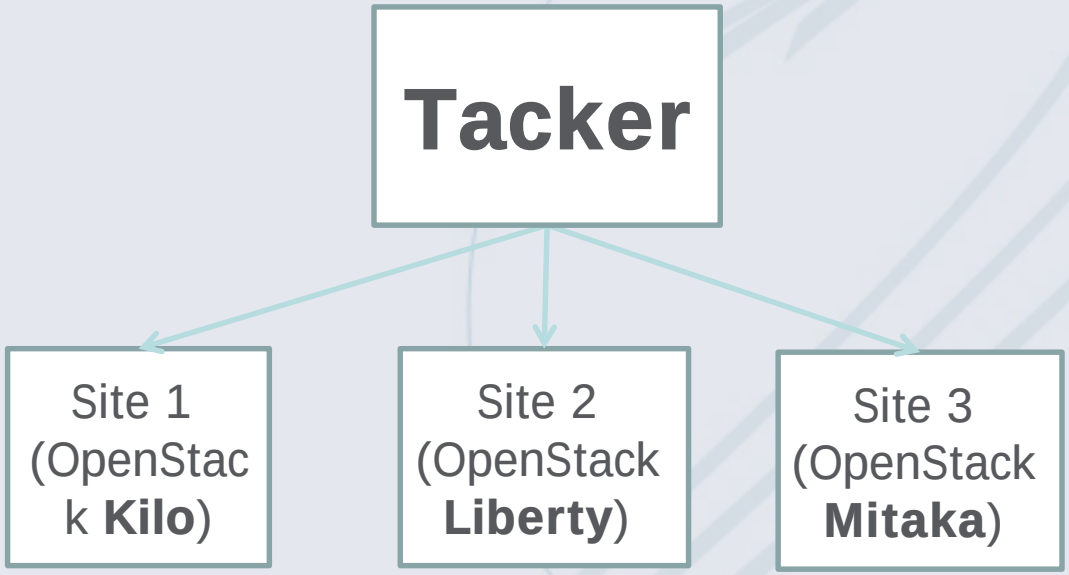
Multi-Site

- VNF placement on specific target OpenStack VIM
- Explicit Region support



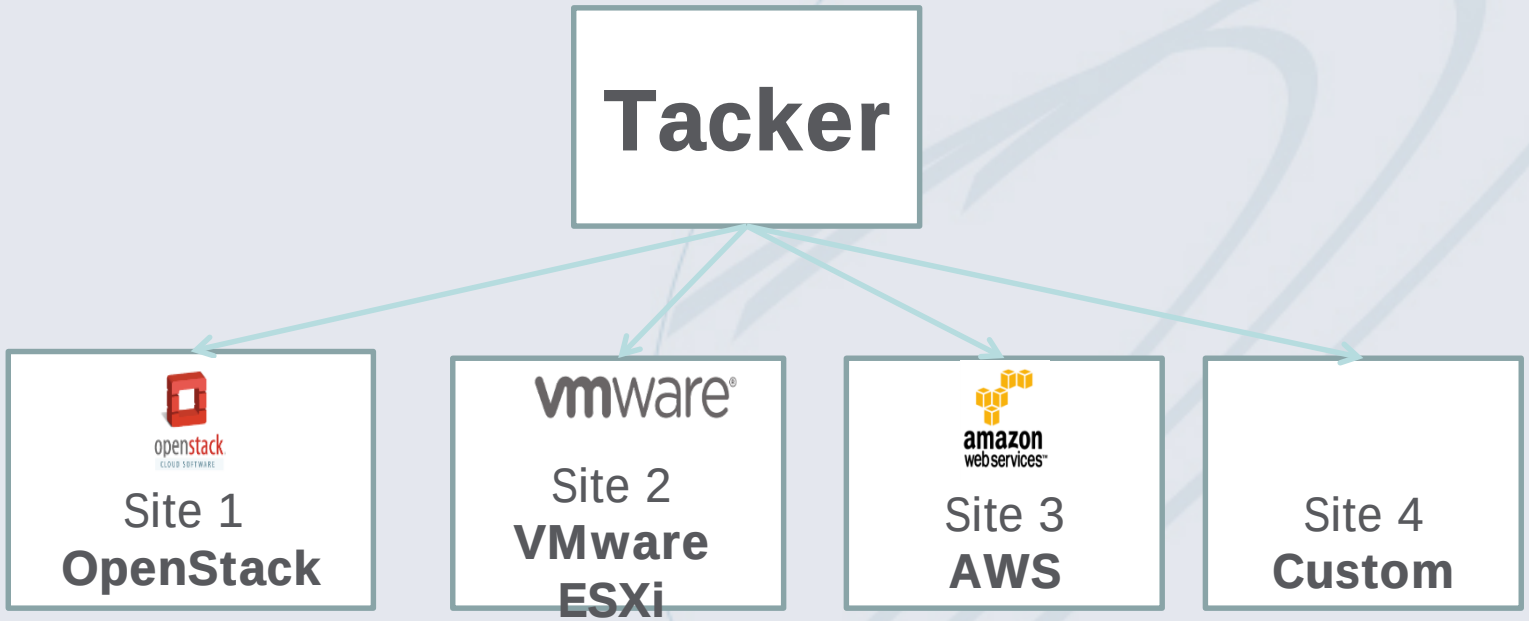
Multi-Site – multiple OpenStack versions

- Operators have multiple OpenStack VIMs with different OpenStack versions
- Detect KeyStone and HOT template version and translate accordingly
- Gracefully degrade features on older VIMs



Multi-VIM Type Support

- Orchestrate VNFs on different type of VIM's
- Introduce Tacker InfraDriver's for VMware ESXi (TOSCA -> OVF)



VNF Manager

- VNF Catalog – repository of VNF descriptors (VNFDs) in a database
- VNF Instantiation and Termination
- VNF Monitoring - Health and Performance Indicators
- Self Healing and Auto Scaling
- VNF Configuration injection during instantiation
- VNF Image management **
- Support both simple and complex VNFs
- Enforce placement policy
  - ensure efficient placement of VNFs (NUMA, CPU Pinning)

** in a future phase

NFV Orchestrator – Work in Progress

- Templatized end-to-end Network Service using *decomposed* VNFs
- Network Service Orchestration using a collection of VNFs and Forwarding Graphs
- Ability to orchestrate VNFs across Multiple VIMs - ***available in Mitaka***
- VNFs connected using Forwarding Graphs using SDN Controller or a SFC API - ***planned for Newton***
- Resource Checks and Resource Allocation
- PNF in end-to-end network service orchestration

Features

- VNF Catalog
- VNF Lifecycle Management
- VNF user-data injection
- VNF configuration injection – during Instantiation and Update
  - SDN Controller using NetConf/YANG
  - Custom management driver
- Loadable Health Monitoring Framework
- Automatic Resource Creation (flavor, image, network)

Tacker VNF Catalog

- Repository of VNF Descriptors (VNFD)

- VNF definition using TOSCA templates
  - Describes the VNF attributes
  - Glance image IDs
  - Nova properties - Placement, CPU Pinning, NUMA policy, etc
  - Performance Monitoring Policy
  - Auto-Healing Policy
- Support for multiple VMs per VNF (VDUs)
- APIs to on-board and maintain VNF Catalog
- VNFDs are stored in Tacker Catalog DB

Tacker - VNF Life Cycle Management

LOGTEL
It's more than knowledge

- Instantiation and Termination of VNF's
  - Tacker API deploys VNF from the VNF Catalog
  - Pluggable infra driver framework
    - OpenStack driver (default)
    - HEAT Driver uses in-built TOSCA to HEAT convertor
    - Instantiates one or more VMs described in TOSCA
  - Terminate VNF will delete all VMs associated with VNF instance

Tacker - VNF Auto-Configuration

- Bootstrap and start-up config of VNF

  - Tacker provides Extensible Management Driver Framework
  - Facilitates VNF configuration based on Service selection
  - Inject initial configuration using:
    - SDN Controller using NetConf/YANG
    - config-drive
    - custom mgmt-driver: connect using ssh / RESTapi and apply configuration •
  - Update configuration in active state • Extendable!

Tacker – VNF Monitoring

- VNF Closed Loop Monitoring and Healing

  - Tacker health check starts as VNF becomes ready
  - Ongoing network connectivity check
  - Auto-restart on failure – based on VNFD policy
  - Extendable, Custom Vendor and Service specific Health Monitoring Driver framework

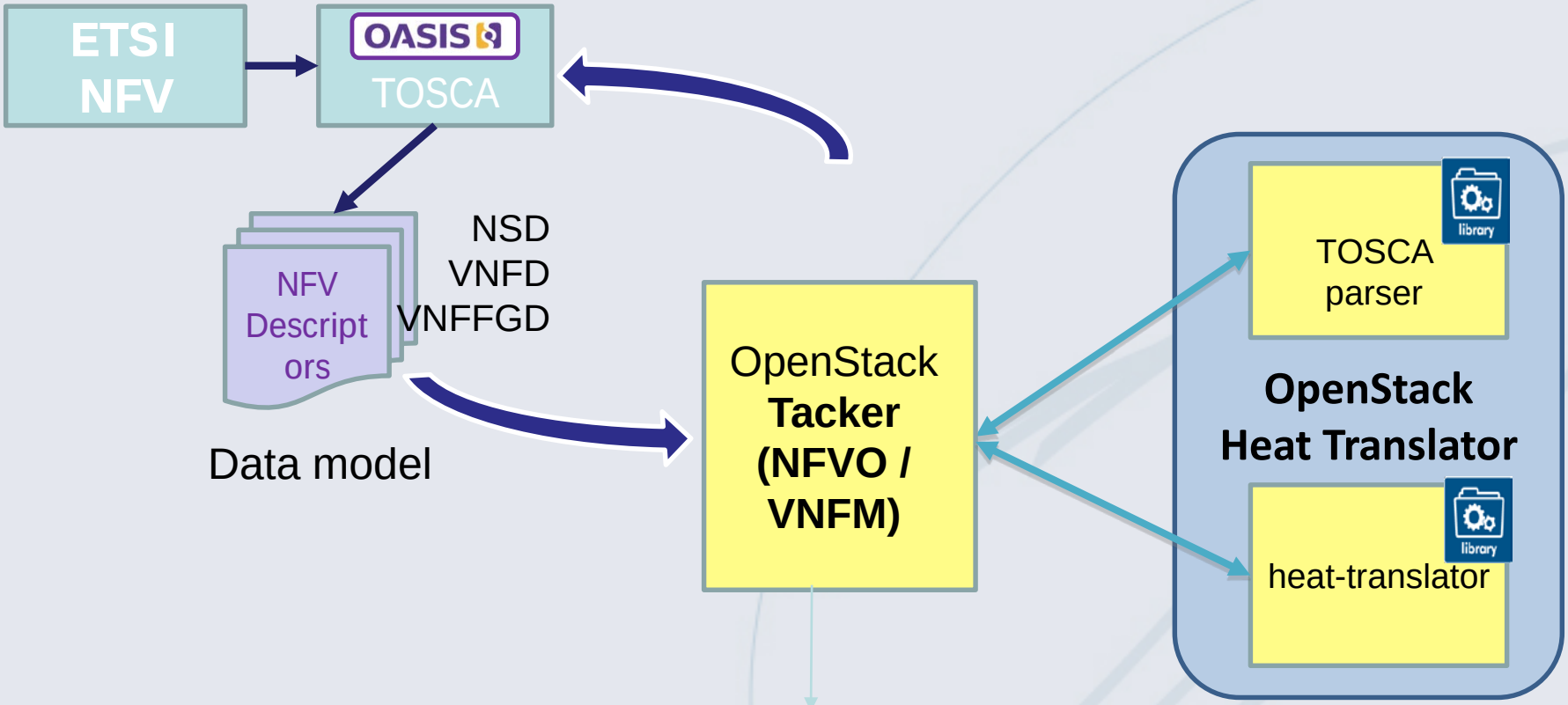
Enhanced Platform Awareness (EPA)

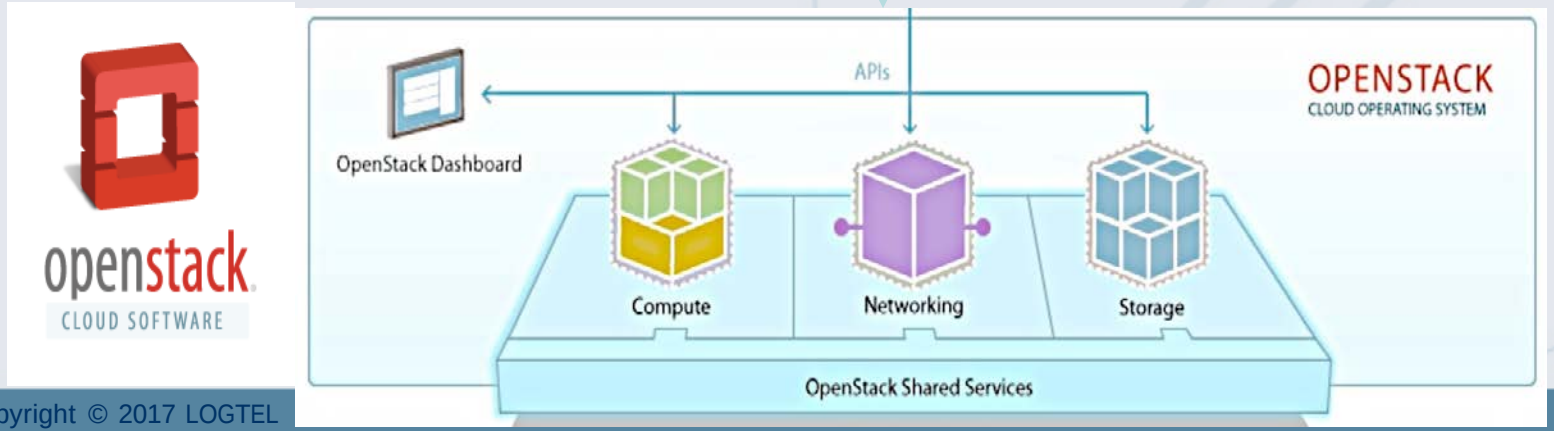
- VNF placement with performance smarts
  - CPU-Pinning
  - Huge Page
  - NUMA awareness
  - SR-IOV
  - PCI pass through

NFV Information Model / Data Model

- Tacker closely working with OASIS TOSCA NFV Adhoc Working group
- Implements the latest CSD03 version of OASIS TOSCA NFV Profile
- Participating in cross-SDO events
- Introduced TOSCA NFV Profile support into tosca-parser [1]
- Transition from in-built translator to tosca-parser in progress [2]

ETSI NFV -> OASIS TOSCA -> Tacker





Sample TOSCA NFV Profile

```
tosca_definitions_version: tosca_simple_profile_for_nfv_1_0_0

description: Template for deploying a single server with predefined properties.

topology_template:
  node_templates:
    VNF1:
      type: tosca.nodes.nfv.VNF
      properties:
        id: vnf1
        vendor: acmetelco
        version: 1.0

    VDU1:
      type: tosca.nodes.nfv.VDU

    CP1:
      type: tosca.nodes.nfv.CP
      properties:
        type: vPort
      requirements:
        - virtualLink: PrivateNetwork
        - virtualBinding: VDU1

    PrivateNetwork:
      type: tosca.nodes.nfv.VL
      properties:
        vendor: ACME Networks
```

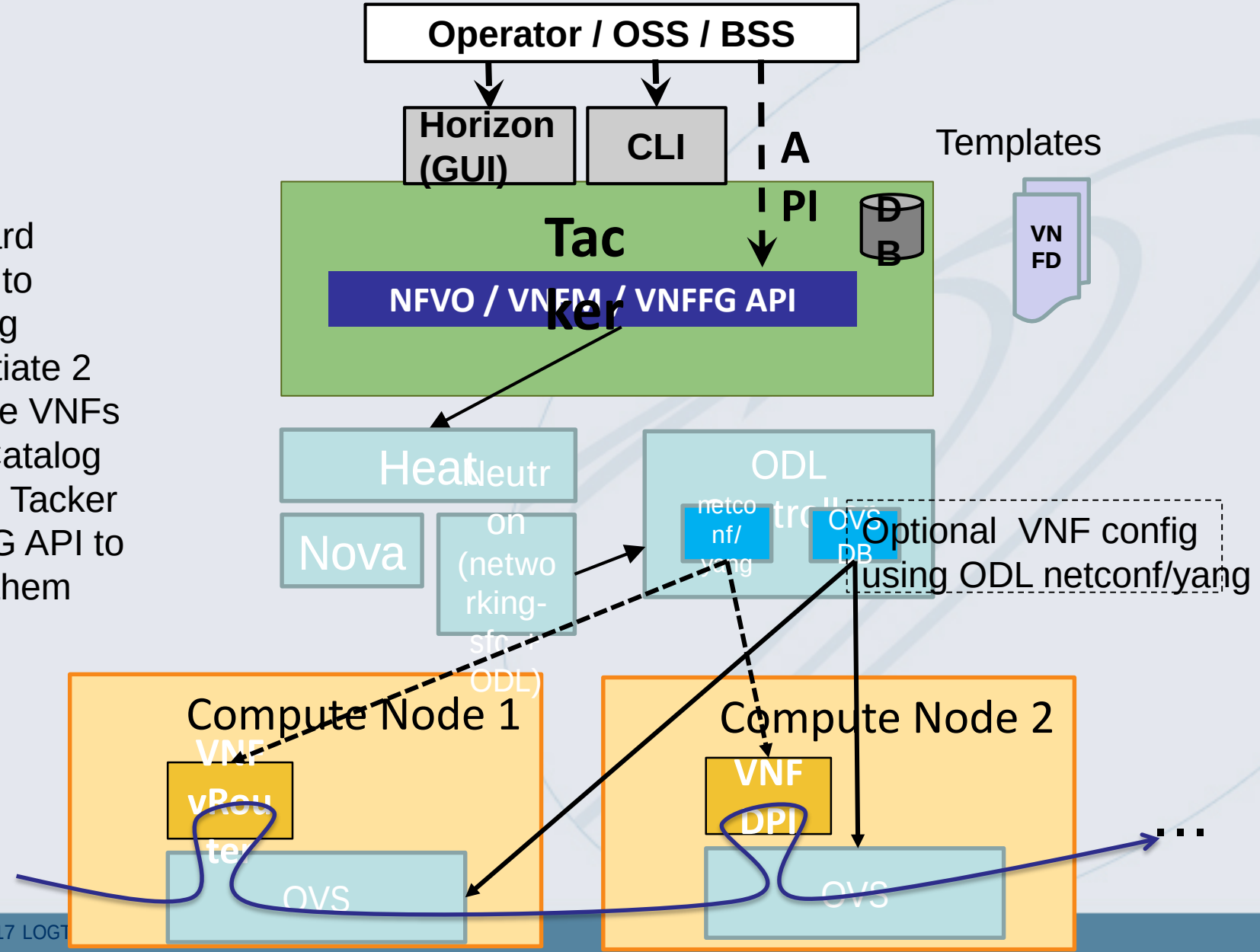
Tacker VNF Forwarding Graph Service Function Chaining Integration

- Adding Tacker VNF Forwarding Graph APIs
- APIs to
  - Define flow classification rules
  - Easy to render chains across instantiated VNFs
- Precursor to VNF Forwarding Graph Descriptor
- ODL-SFC / netvirtsfc driver support
- OPNFV Summit Demo using Tacker and ODL
- Plan to transition to neutron-sfc driver

Tacker + SFC Overview: (networking-sfc + ODL)

Workflow:

1) Onboard VNFD to Catalog
2) Instantiate 2 or more VNFs from Catalog
3) Invoke Tacker VNFFG API to chain them



VNF AND MANAGEMENT

VNFM interfaces

Resource	Operations	Description
VNFC Image	C,D	Create or delete the image associated with the VNFC in or from the VIM.
VNFC	C,R,D	Create, delete, or query the virtual instance associated with VNFC.
VNFC Link	C,D	Create, delete, query, or test the network connectivity between the two VNFCs.
VNFCPublicAccess	C,R,D,T	Create, delete, query, or test the external public access of the VNFC instance.
VNFCAdmin		With different parameters, start, stop, suspend, resume, or reboot the VNFC instance.
VNFCSaling		With different parameters, scales up, down, in, and out the VNFCs.
VNFCHealing		With different parameters, applies different approaches for VNFCHealing. For example, with fail-over parameter, it fails-over VNFC instance to the hot standby instance.
VNFConfig		With different parameters, connects to either EM or VNF control instance for VNF configuration.
Notify		Notify the subscriber with an object that contains the low level messages related to VNF lifecycle state change.

- This set of interface captures the operations that VNF Manager needs to complete for VNF life cycle management.
- The interface follows REST API design to expose system resources and apply CRUD operations on them.
- Notify API opens the notification channel between VNFM and the external systems to allow VNFM to update VNF state based on the notification sent from the cloud adaptors.

Resources

```
resources:
  <resource ID>:
    type: <resource type>
    properties:
      <property name>: <property value>
    metadata:
      <resource specific metadata>
    depends_on: <resource ID or list of ID>
    update_policy: <update policy>
    deletion_policy: <deletion policy>
    external_id: <external resource ID>
    condition: <condition name or expression or boolean>
```

```
resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      flavor: m1.small
      image: F18-x86_64-cfntools
```

```
resources:
  server1:
    type: OS::Nova::Server
    depends_on: server2

  server2:
    type: OS::Nova::Server
```

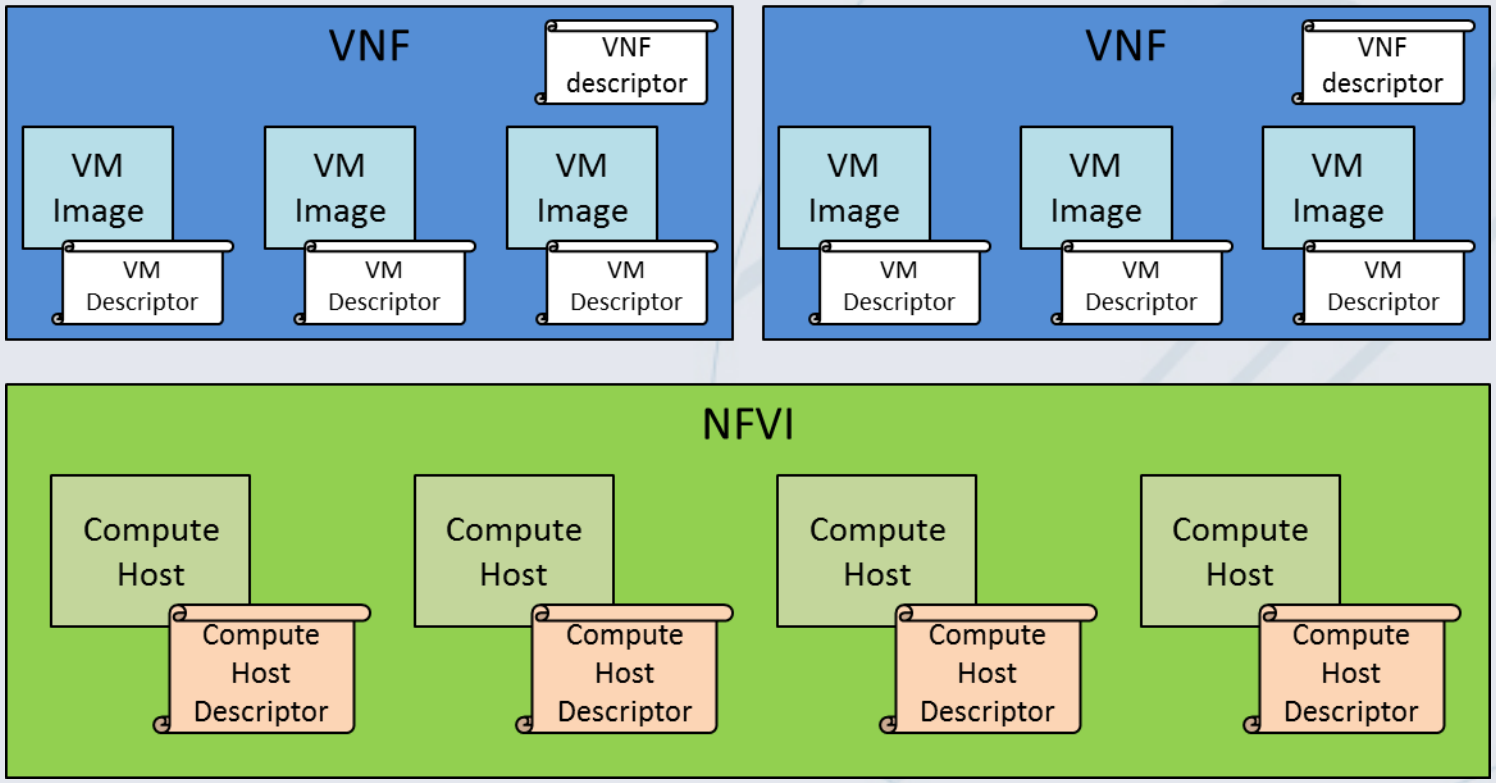
Parameters

```
parameters:
  <param name>:
    type: <string | number | json | comma_delimited_list | boolean>
    label: <human-readable name of the parameter>
    description: <description of the parameter>
    default: <default value for parameter>
    hidden: <true | false>
    constraints:
      <parameter constraints>
    immutable: <true | false>
```

```
parameters:
  user_name:
    type: string
    label: User Name
    description: User name to be configured for the application
  port_number:
    type: number
    label: Port Number
    description: Port number to be configured for the web server
```

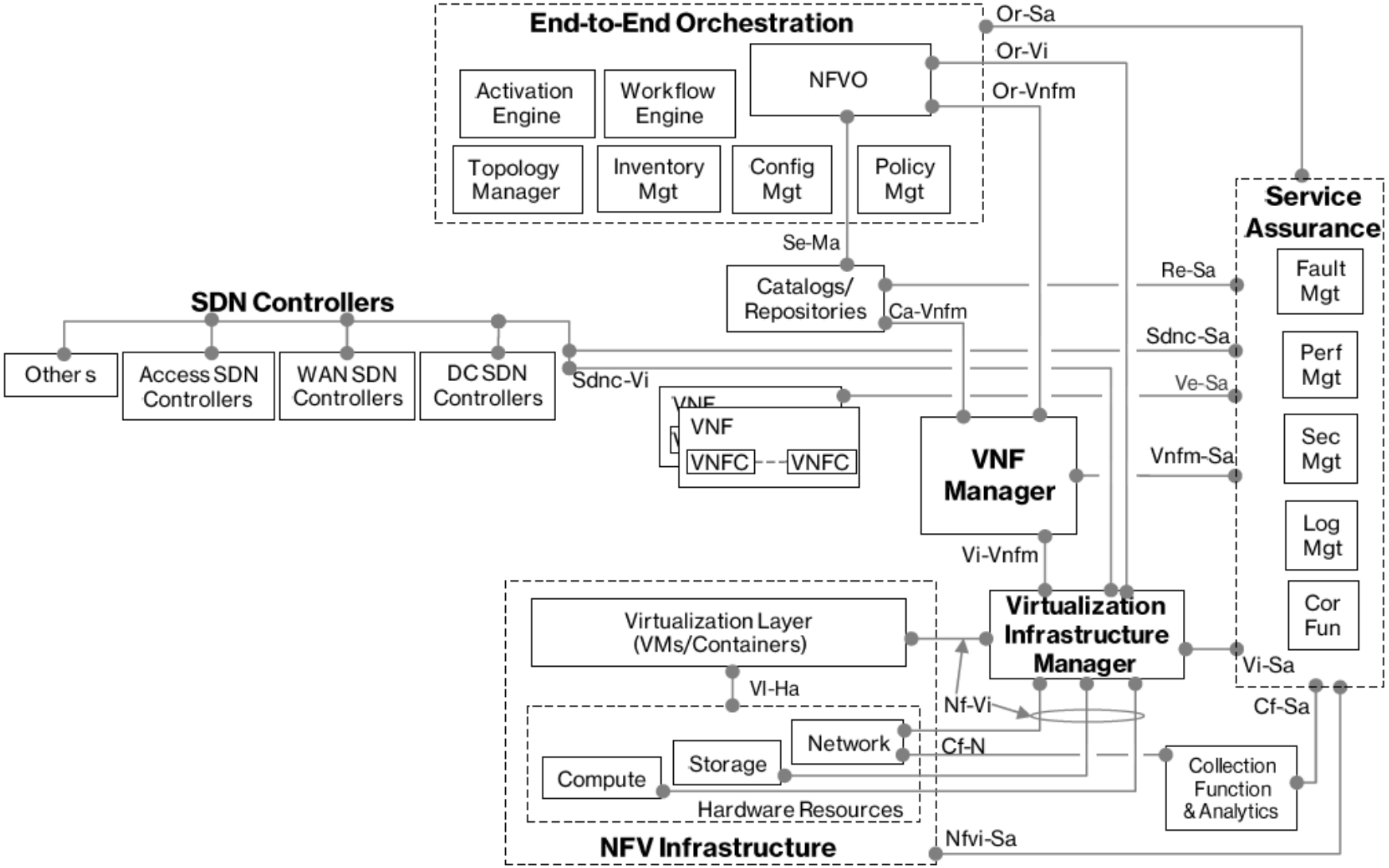
Descriptors hierarchy

- VM Descriptor
  - *NFV resources demanded by the VM* from the NFVI
- Compute Host Descriptor
  - *Capabilities and available resources* at the *Compute Host*

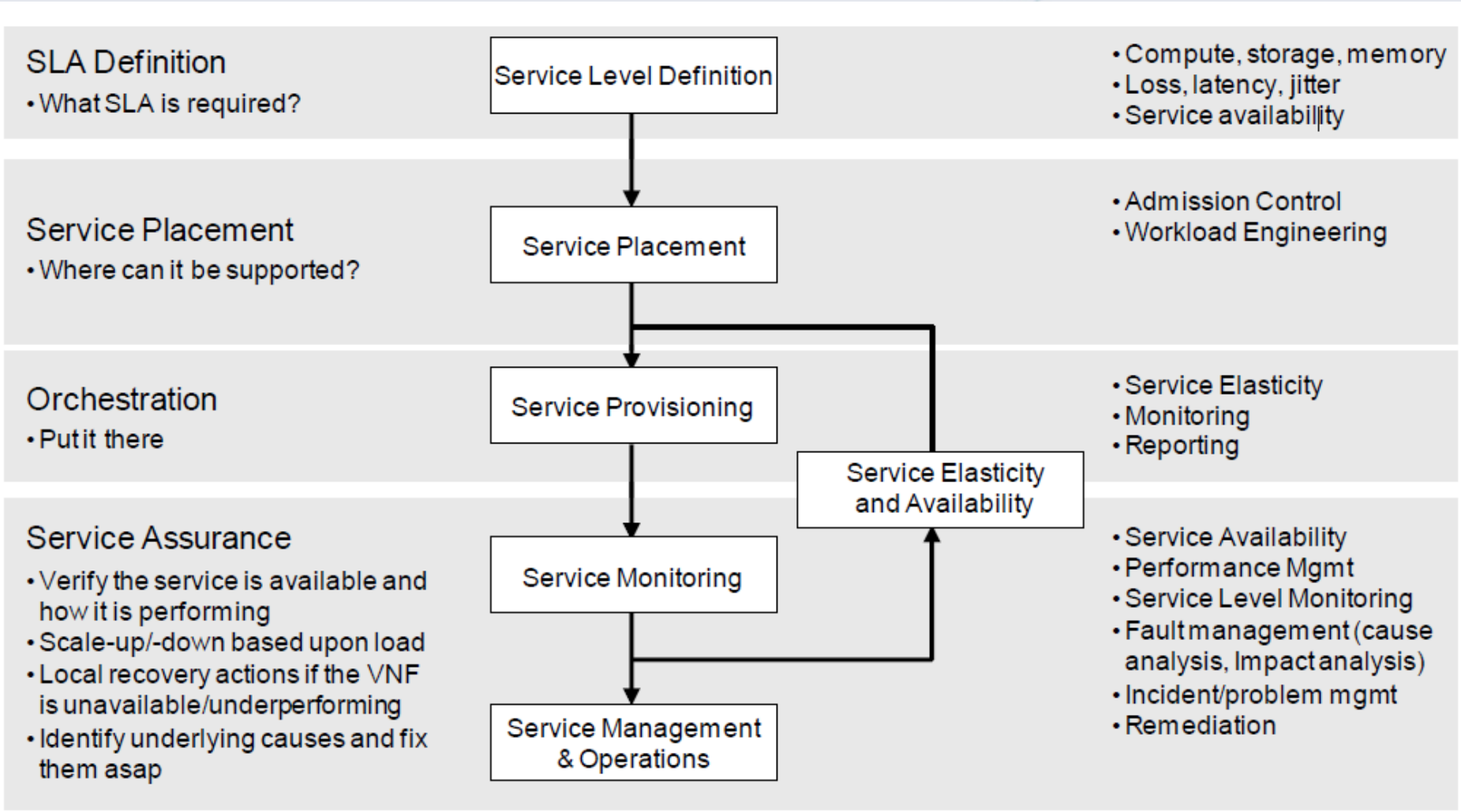


NFV
SERVICE ASSURANCE

SDN-NFV Reference Architecture v1.0



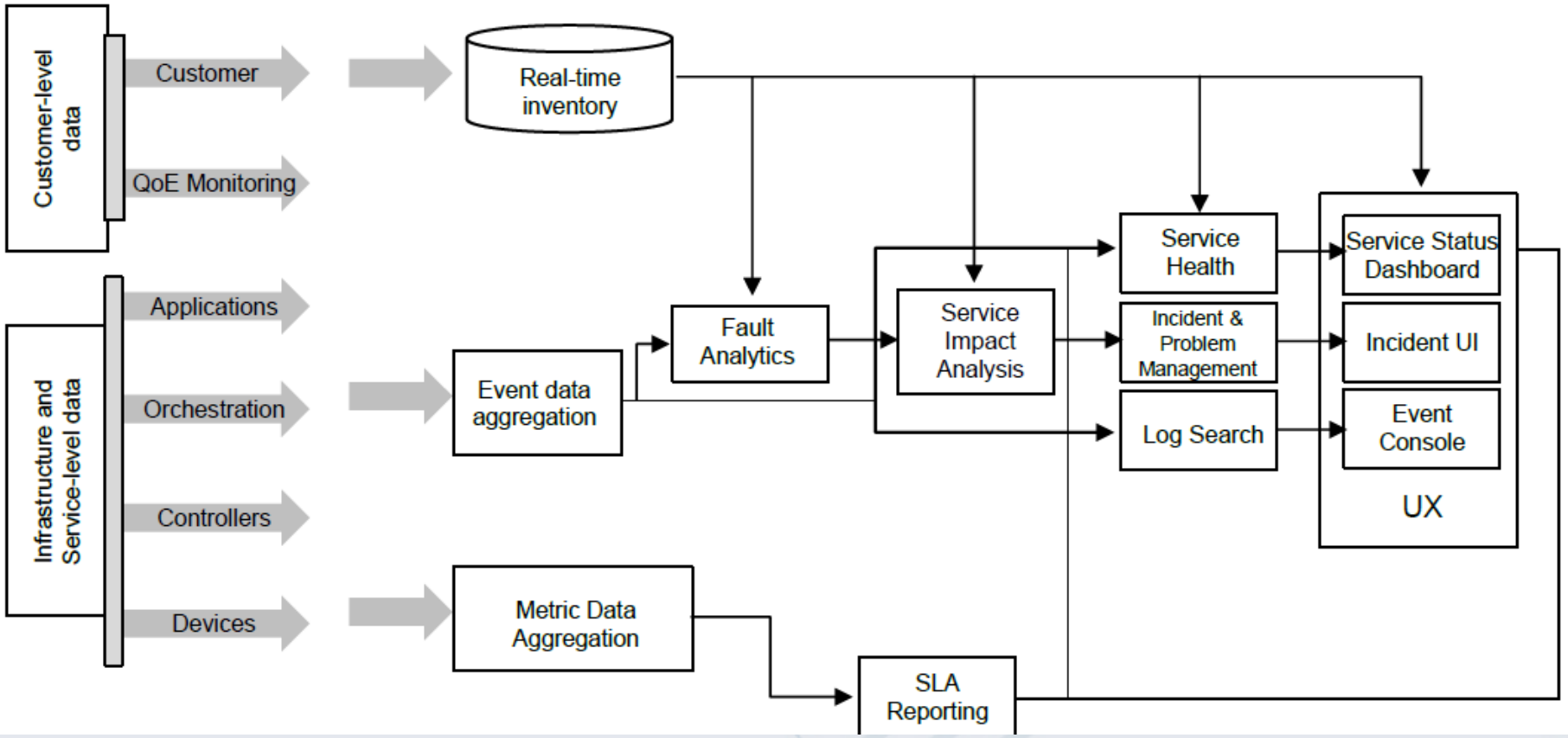
Service Assurance Lifecycle



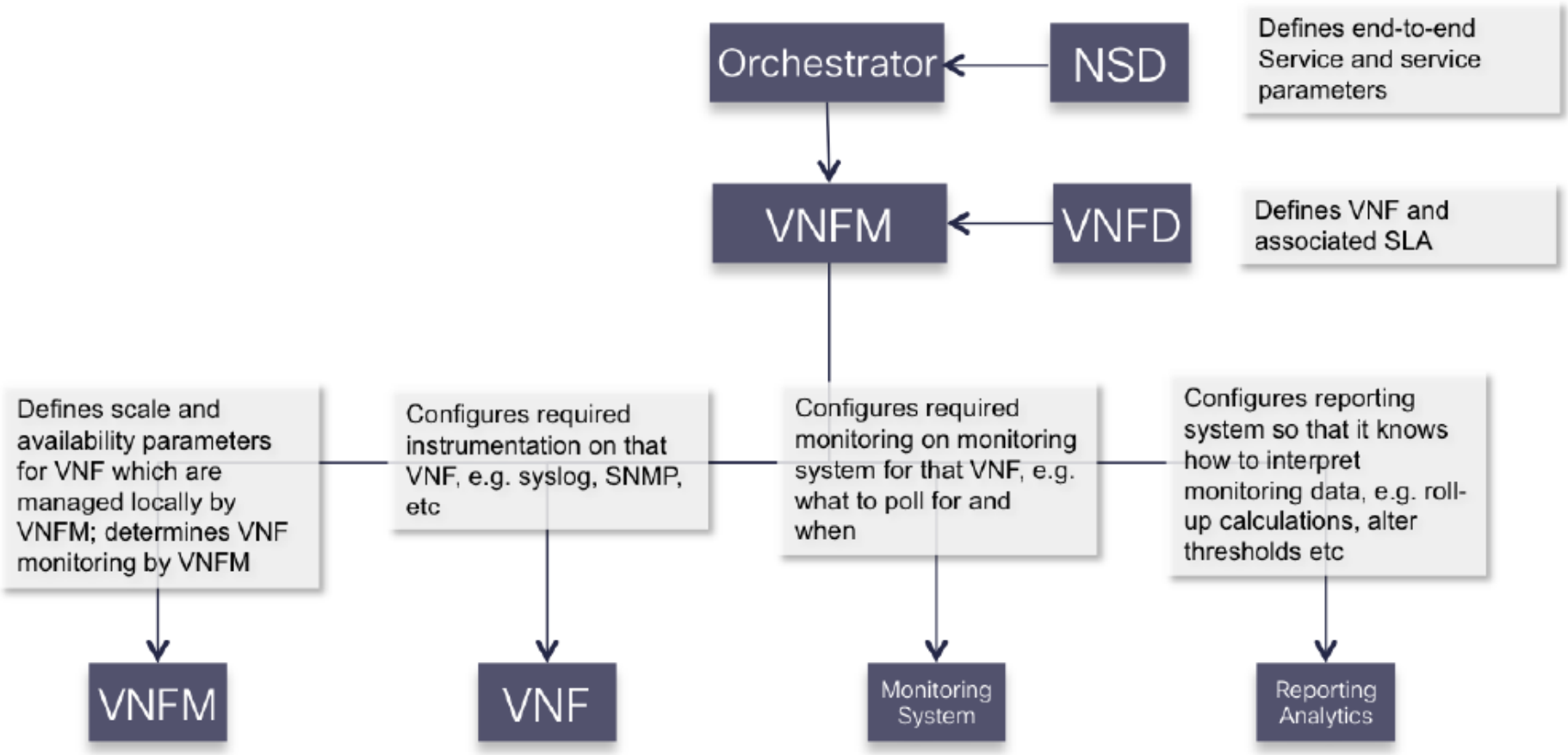
Primary Functions of a Service Assurance System

Primary Function	Mechanism	Effect
Performance Management	SLA Reporting	Monitor/report whether SLAs are being met (minimize MTTD)
	Performance Analytics	Automatically detect performance anomalies
Fault Management	Cause Isolation	Identify the underlying causes of faults
	Service Impact Analysis	Identify which services are impacted to prioritize fault resolution and minimize service impact
Incident and Problem Management	Mapping	Automatically mapping problems to trouble tickets so an engineer can fix them
	Prioritization	Prioritize trouble tickets to target operations efforts based on service impact
	Enrichment	Ensure that operations engineers are informationally equipped to minimize MTTR
	Expertise	Capture expert knowledge to try and prevent future incidents

Service Assurance Functional Architecture



Orchestrated Service Assurance



THOUGHTS & CONCLUSIONS

	ReqID	ReqChapter	ReqName	Brief Description	Explanation / Implementation Hints
1					
5	R4A-Arch-Inf-0021		Efficient FDC/ BDC ressource usage	Applications/VNFs MUST provide three types of VNFCs for user-, control-, and management-plane, if applicable. User plane VNFCs shall be able to run in the Internet and not expect any protection from the infrastructure. If the supplier does not comply now, a description is needed how and when this is achieved.	It depends on various parameters where to run which VNFC component which may also change over time.
6	R4A-Arch-Inf-0030	General Aspects	Cloud characteristics	Application/VNF SHALL be deployable in an OpenStack cloud with the following characteristics: - KVM/QEMU - uniform x86 server hardware - Ceph storage - placement zones, e.g. EHD, DMZ, MZ, SEC - Heat templates / TOSCA model description - Chef / Puppet to configure the software-stack within VMs if needed	Non support of OpenStack cloud components will not be further considered as DTAG supplier for this particular RFx.
7	R4A-Arch-Inf-0040		Virtualization layer provided by DTAG	The application MUST be implemented to be fully operational on the virtualization layer provided by DTAG. There will be no infrastructure control allowed from the application layer bypassing the orchestration systems provided by DTAG. Modifications or additions on the underlying virtualization solution is not accepted. Details of the virtualization concept provided by DTAG can be found in R4A-Arch-Inf-0090.	
8	R4A-Arch-Inf-0050	General Aspects	No HW dependency	Any HW dependency of the VNF shall be avoided, meaning the VNF shall not make any assumptions regarding the underlying infrastructure and hardware, it shall use abstractions in relation to the operating system, file system, database, etc.	
9	R4A-Arch-Inf-0060		Hardware abstraction	Applications/VNFs MUST NOT use hardware-specific functions conflicting with DTAG unified virtualization layer. It is not intended to bypass the hypervisor or adapt the hardware on a per application basis.	

The end ;-)

Samuel.dratwa@gmail.com