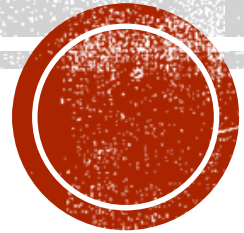
SOFTWARE DEFINED VIRTUAL NETWORKING SECURITY

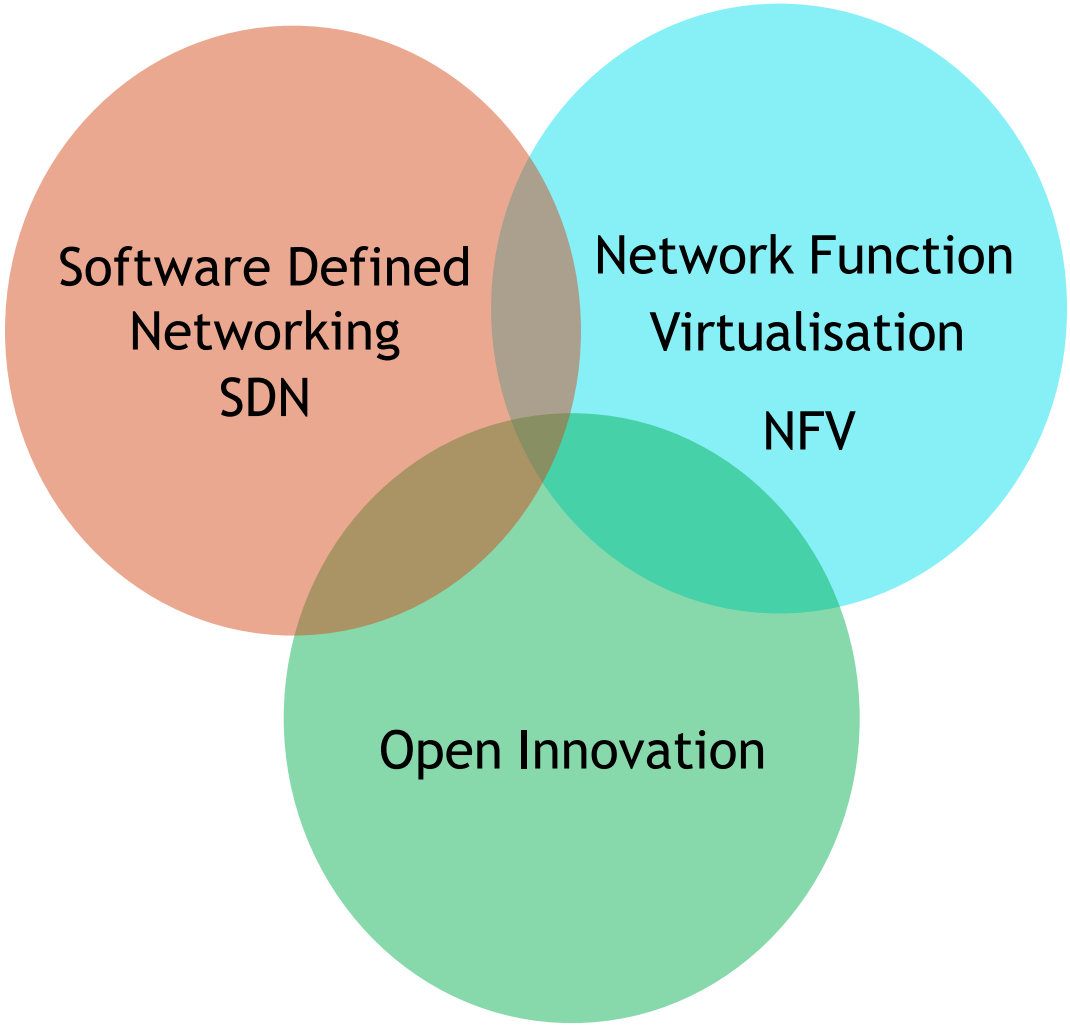
CHAPTER 3 SDN AND NFV

Dijiang Huang, Ankur Chowdhary, and Sandeep Pisharody

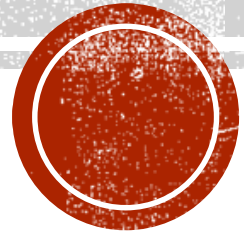


OUTLINE

- Software Defined Networking
  - Concept of SDN
  - OpenFlow – a SDN Implementation
  - Open vSwitch – a flow-based virtual switch
- Network Function Virtualization
  - Concept of NFV
  - Case Study: OpenStack



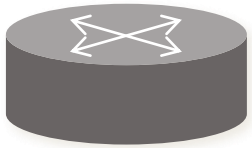
CONCEPTS OF SDN



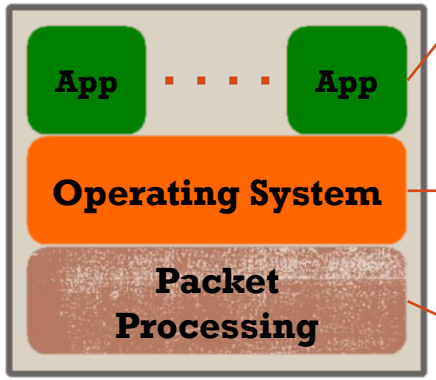
- Distributed network routing
- SDN concepts & architecture

NETWORKING DEVICES

Network Router



Protocol and operation model



- Routing
- Access Control
- Traffic Engineering
- Management

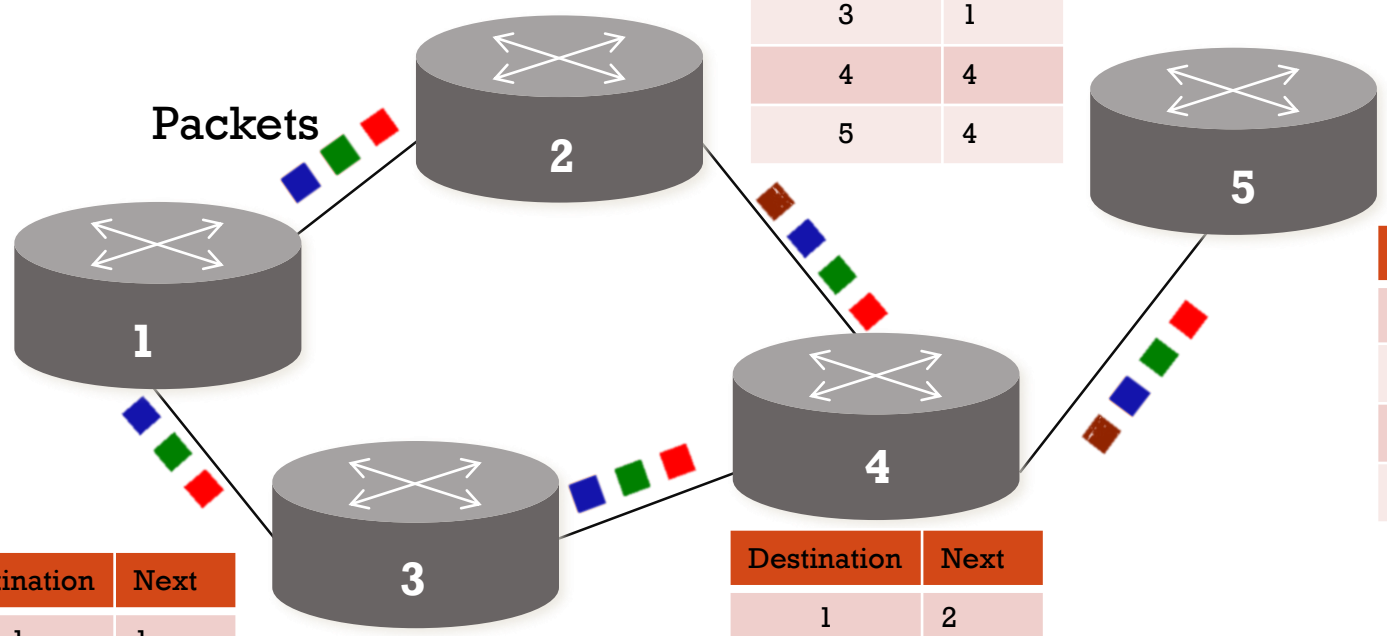
- Interface Virtualization
- Control and Management APIs

- Forwarding
- Switching

DATA PROCESSING AND CONTROL IN DISTRIBUTED SYSTEMS

An example of networking routing configuration

- A router has both control and packet forwarding functions based on the routing table established among routers
- Complicated task-specific distributed algorithm



Destination	Next
2	2
3	3
4	2
5	2

Destination	Next
1	1
3	1
4	4
5	4

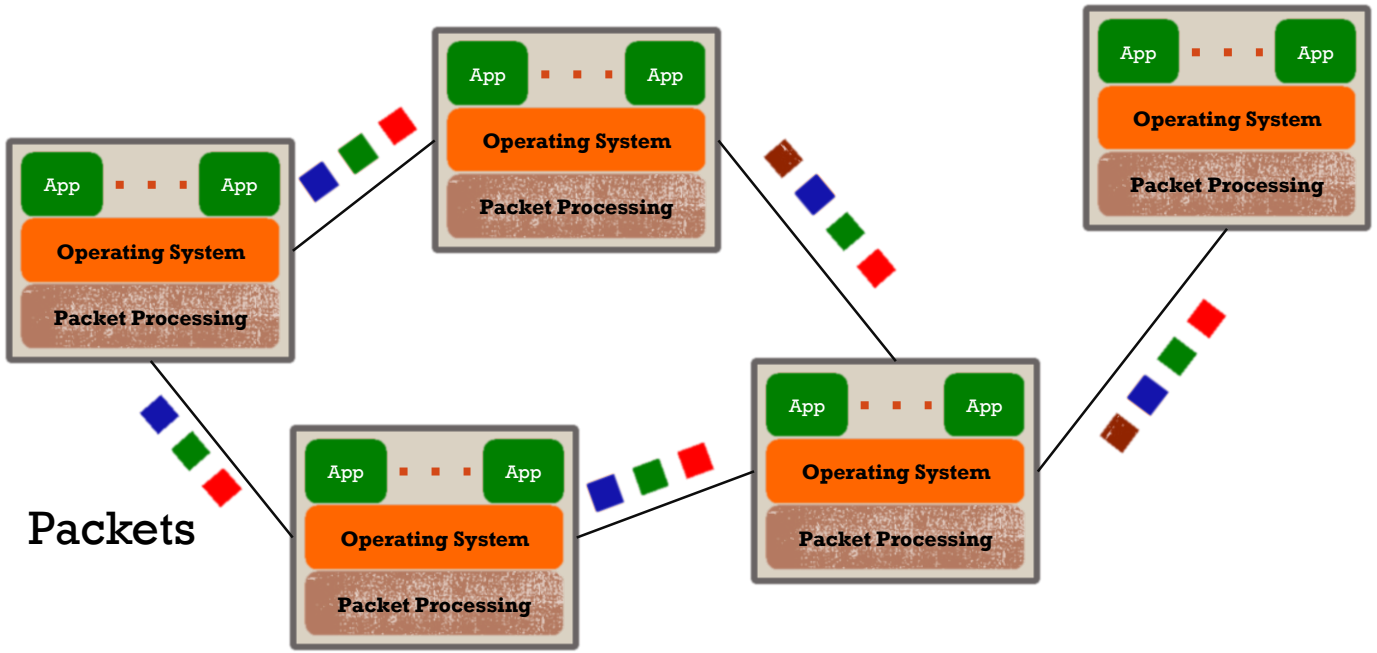
Destination	Next
1	4
2	4
3	4
4	4

Destination	Next
1	1
2	1
4	4
5	4

Destination	Next
1	2
2	2
3	3
5	5

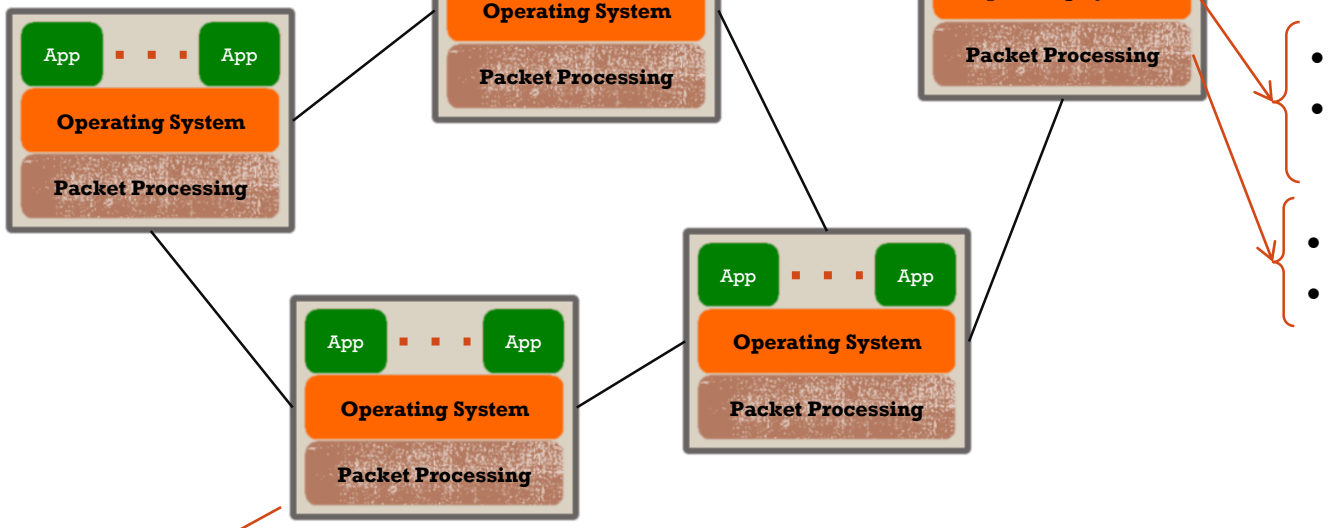
DATA PROCESSING AND CONTROL IN DISTRIBUTED SYSTEMS

- Data processing and controls are **not separated**.
- Algorithms, e.g., routing, is performed in a distributed environment.
- Each router acts **individually**, much likes our current social networks.



DISTRIBUTED NETWORKING SYSTEMS

- Close Boxes, Fully Distributed Protocols



- Routing
- Access Control
- Traffic Engineering
- Management

- Interface Virtualization
- Control and Management APIs

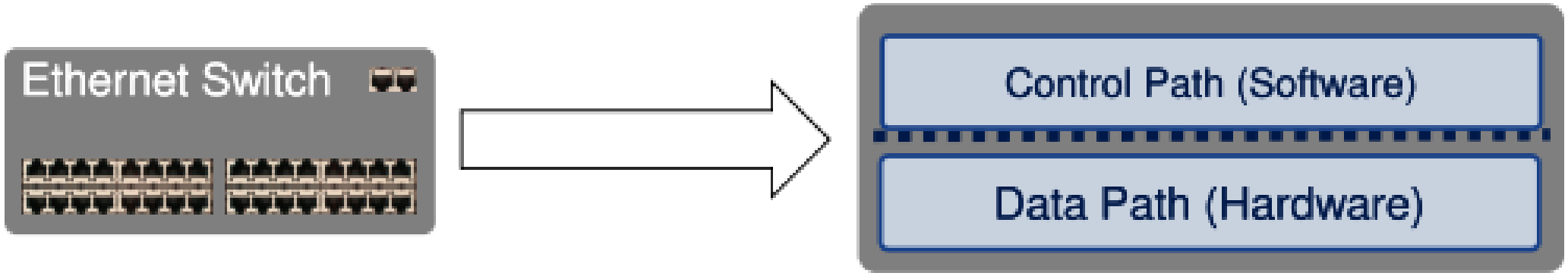
- Forwarding
- Switching

Closed Operating Environment

- Traffic control decisions are mostly made **locally**, i.e., usually require a negotiation procedures among network entities (e.g., routing).
- Not easy for testing and experiments, i.e., usually requires a separate testing environments.

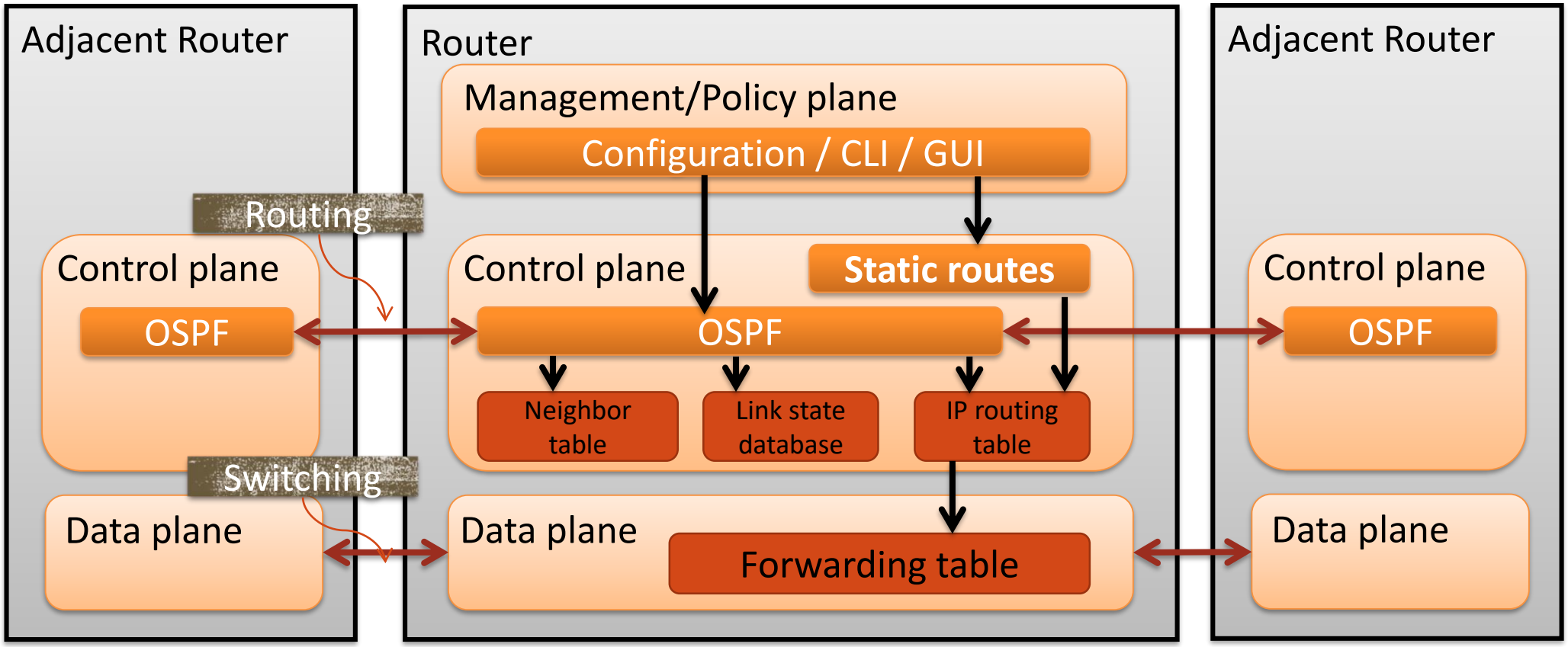
Traditional Network Node: Switch

- Two “planes”
  - **Control Plane**: computing the forwarding state
    - Involves coordination with rest of system
  - **Data Plane**: forwarding packets
    - Based on local forwarding state



TRADITIONAL NETWORK NODE: ROUTER

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



PROBLEMS OF COMPUTER NETWORKS?

- Networks used to be simple: Ethernet, IP, TCP….
- Many complex functions backed into infrastructure
  - OSPF, BGP, multicast, differentiated services, traffic Engineering, NAT, firewalls, …
- For examples, new **control** requirements led to great complexity
  - Isolation ➔ VLANs, ACLs
  - Traffic engineering ➔ MPLS, ECMP, Weights
  - Packet processing ➔ Firewalls, NATs, middleboxes
  - Payload analysis ➔ Deep packet inspection (DPI)
  - …..

SDN DEFINITIONS

- SDN is an approach to building computer networks that separates and abstracts elements of these systems.
  – *from Wikipedia*

- In the Software Defined Networking architecture, the control and data planes are **decoupled**, network intelligence and state are **logically centralized**, and the underlying network infrastructure is **abstracted** from the applications.
  – *from ONF White Paper*

TWO "PLANES" OF SDN

- Two fundamental terms to begin understanding the SDN

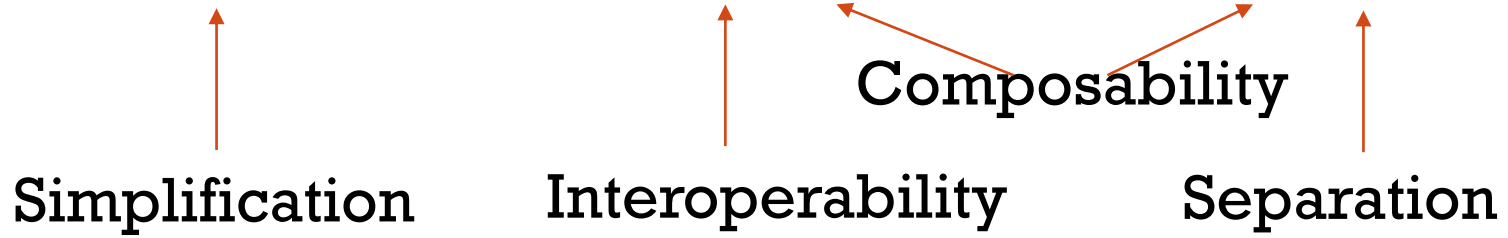
Processing Plane	Where it runs	How fast these processes run	Type of processes performed
Control Plane	Switch CPU (smart but slow)	In the order of thousands of packets per second	Routing protocols (i.e., OSPF, IS-IS, BGP), Spanning Tree, SYSLOG, AAA (Authentication Authorization Accounting), NDE (Netflow Data Export), CLI (Command Line interface), SNMP
Data Plane	Dedicated Hardware ASIC (fast but dumb)	Millions or Billions of packets per second	Layer 2 switching, Layer 3 (IPv4 | IPv6) switching, MPLS forwarding, VRF Forwarding, QOS (Quality of Service) Marking, Classification, Policing, Netflow flow collection, Security Access Control Lists

HOW TO SIMPLIFY THE NETWORKING PROBLEM?

- How to get a simpler, more systematic design for the so complicate network control mechanisms?

- The power of Abstraction
  - "Modularity based on abstraction is the way things get done." – Barbara Liskov

- **Abstractions → Interfaces → Modularity**



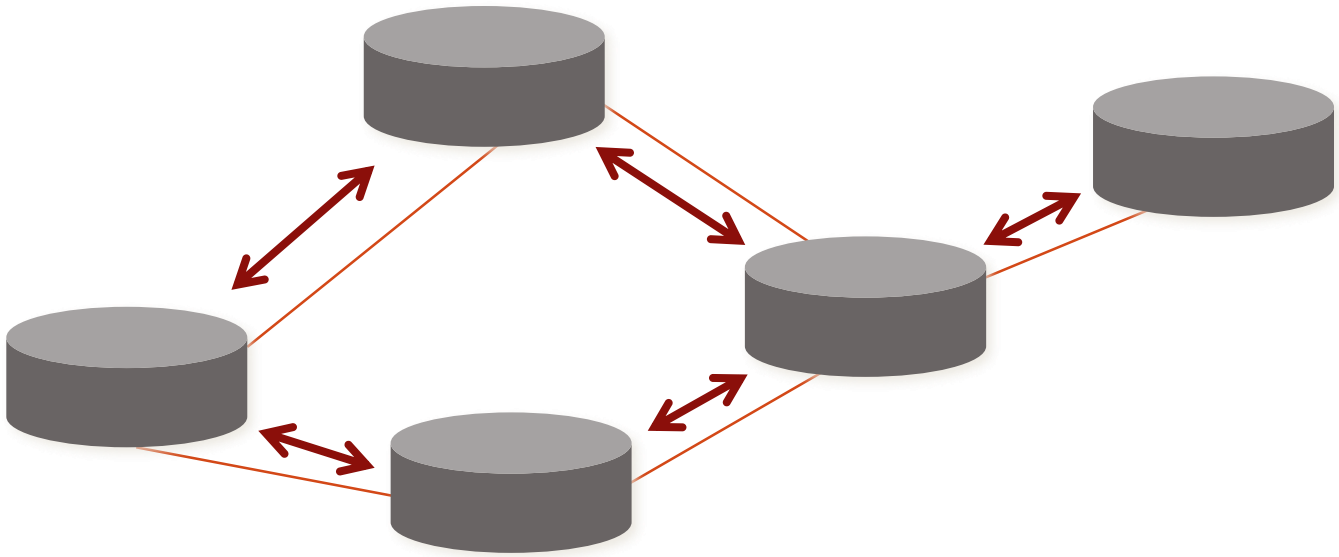
SDN ABSTRACTIONS

- SDN is defined precisely by three abstractions:
  Forwarding, Distribution, Configuration
  - Abs#1: Be compatible with low-level hardware/software
    - Need an abstraction for general **forwarding model**
    - **OpenFlow** is current proposal for forwarding standard
    - Configuration in terms of **flow entries**: <header, action>
  - Abs#2: Make decisions based on entire network
    - Need an abstraction for **distributed network state**
    - **Global network view** abstraction
    - **Network OS** (controllers) queries network devices to form "view" and sends commands to them to control forwarding
  - Abs#3: Compute the configuration of each physical device
    - Need an abstraction that **simplifies configuration**
    - **Network virtualization**: map abstract configuration to physical configuration

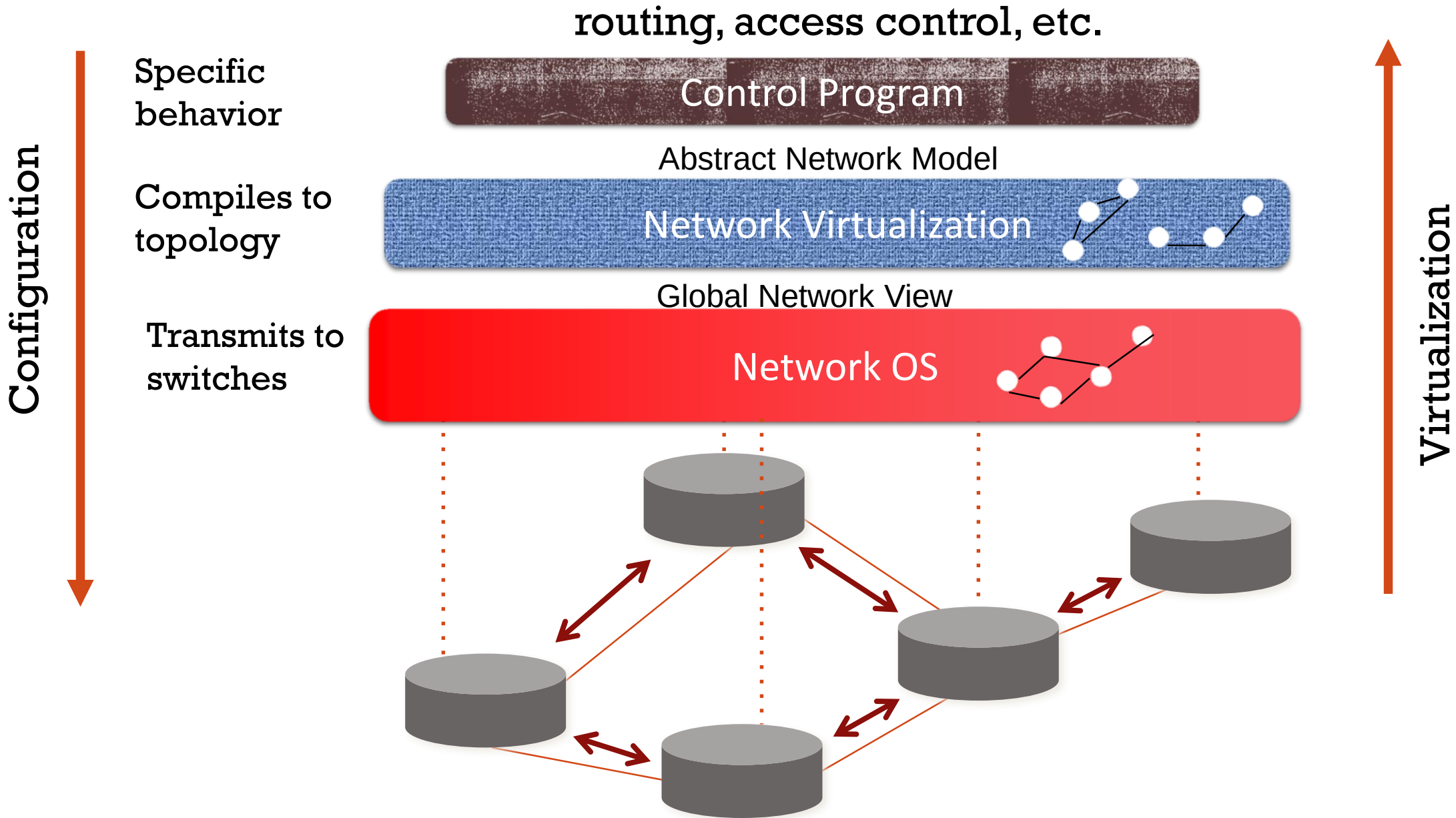
TRADITIONAL CONTROL MECHANISMS

Distributed algorithm running between neighbors

Complicated task-specific distributed algorithm

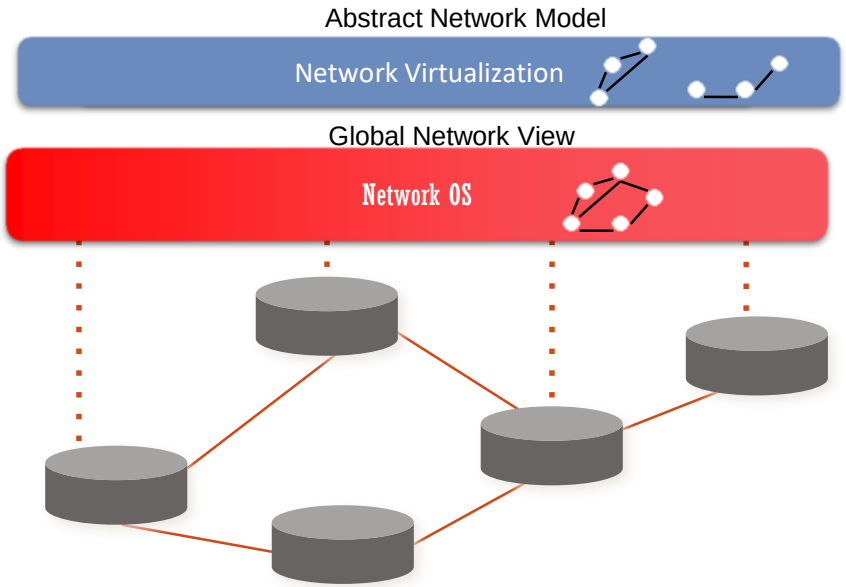


SOFTWARE DEFINED NETWORK (SDN)



How to process the SDN requests?

- Write a simple program to configure a simple model
  - Configuration merely a way to specify what you want
- Examples
  - ACLs: who can talk to who
  - Isolation: who can hear my broadcasts
  - Routing: only specify routing to the degree you care
    - Some flows over satellite, others over landline
  - TE: specify in terms of quality of service, not routes
- Virtualization layer "compiles" these requirements
  - Produces suitable configuration of actual network devices
- NOS then transmits these settings to physical boxes



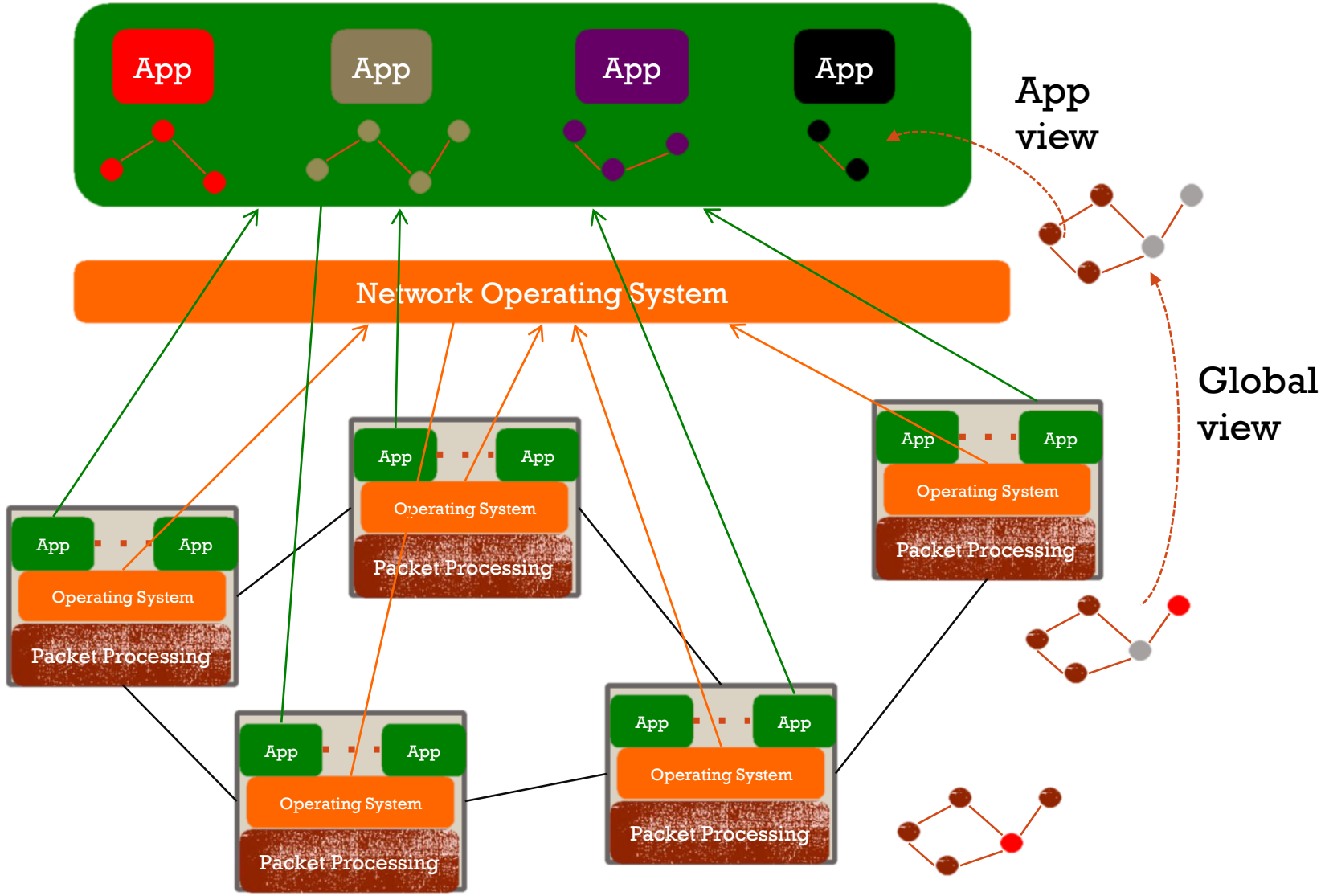
CLEAN SEPARATION OF CONCERNS

- **Control program:** express goals on abstract view
  - Driven by **Operator Requirements**
- **Virtualization Layer:** abstract view ←→ global view
  - Driven by **Specification Abstraction** for particular task
- **NOS:** global view ←→ physical switches
  - API: driven by **Network State Abstraction**
  - Switch interface: driven by **Forwarding Abstraction**

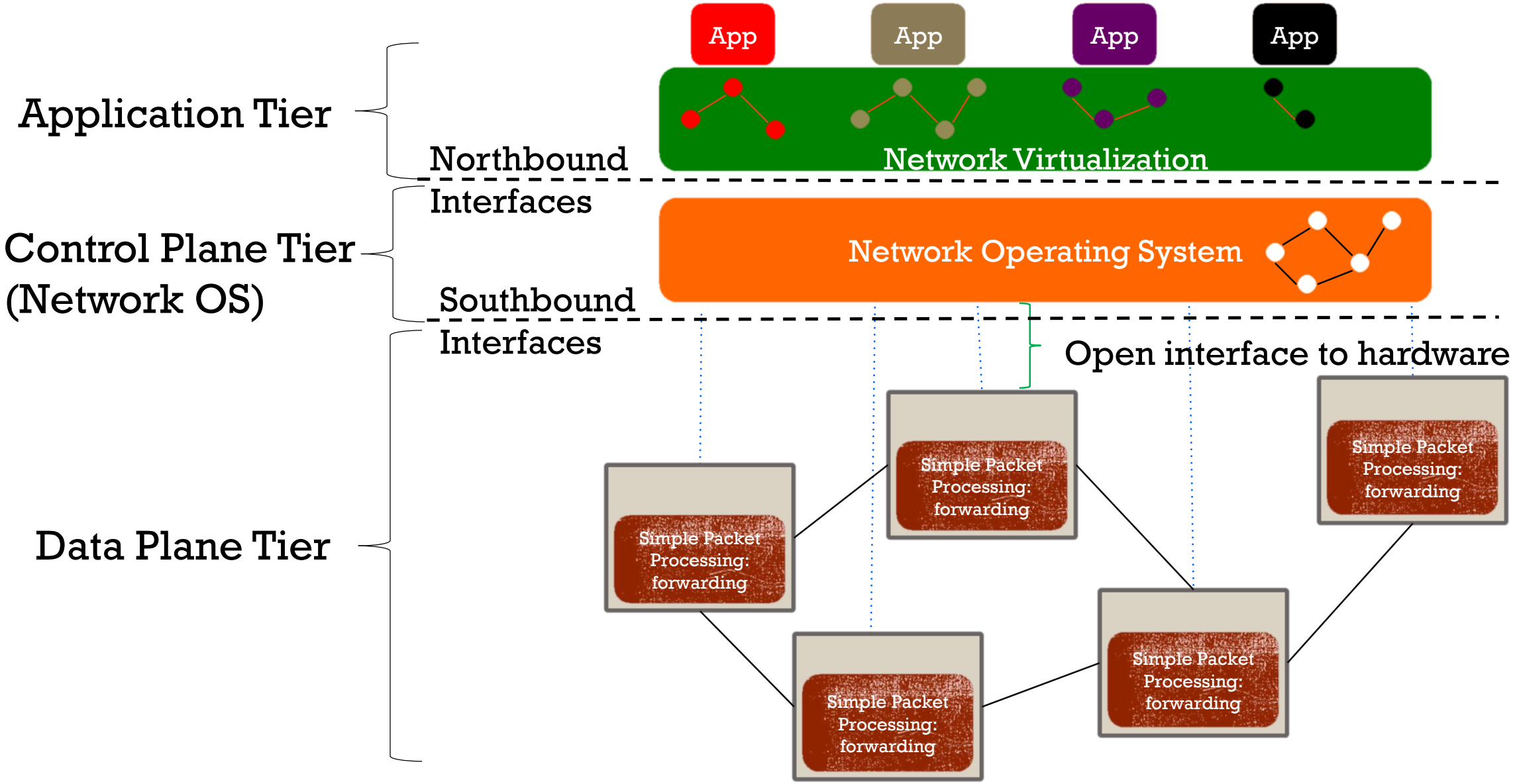
SOFTWARE-DEFINED NETWORKING

- Data Plane Tier
  - Packet forwarding (as per flow table), packet manipulation (as per flow table), statistics collection
- Control Plane Tier (Network OS)
  - Data plane resource marshaling, common libraries (e.g., topology, host metadata, state abstractions)
- Application Tier
  - Virtual network overlays, network slicing (delegation), tenant-aware broadcast, application-aware path computation, integration with other software packages, policy, security, traffic engineering.

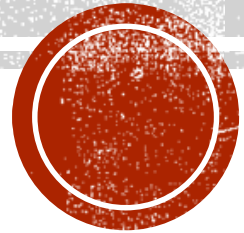
FROM TRADITIONAL NETWORKING TO SDN



ABSTRACT SDN MODEL

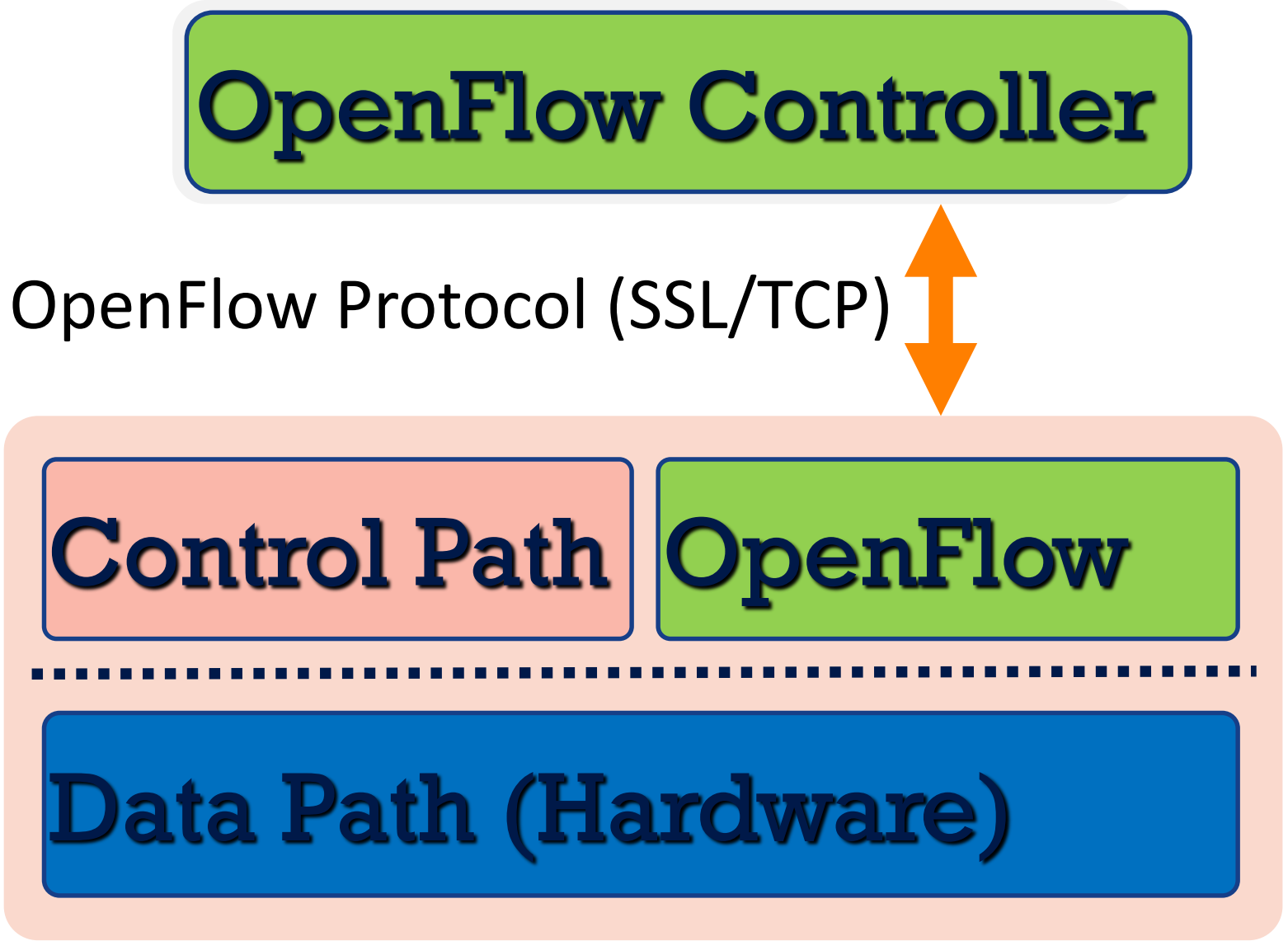


OPENFLOW – AN SDN IMPLEMENTATION



- **OpenFlow**
- Open vSwitch (OVS)

OPENFLOW

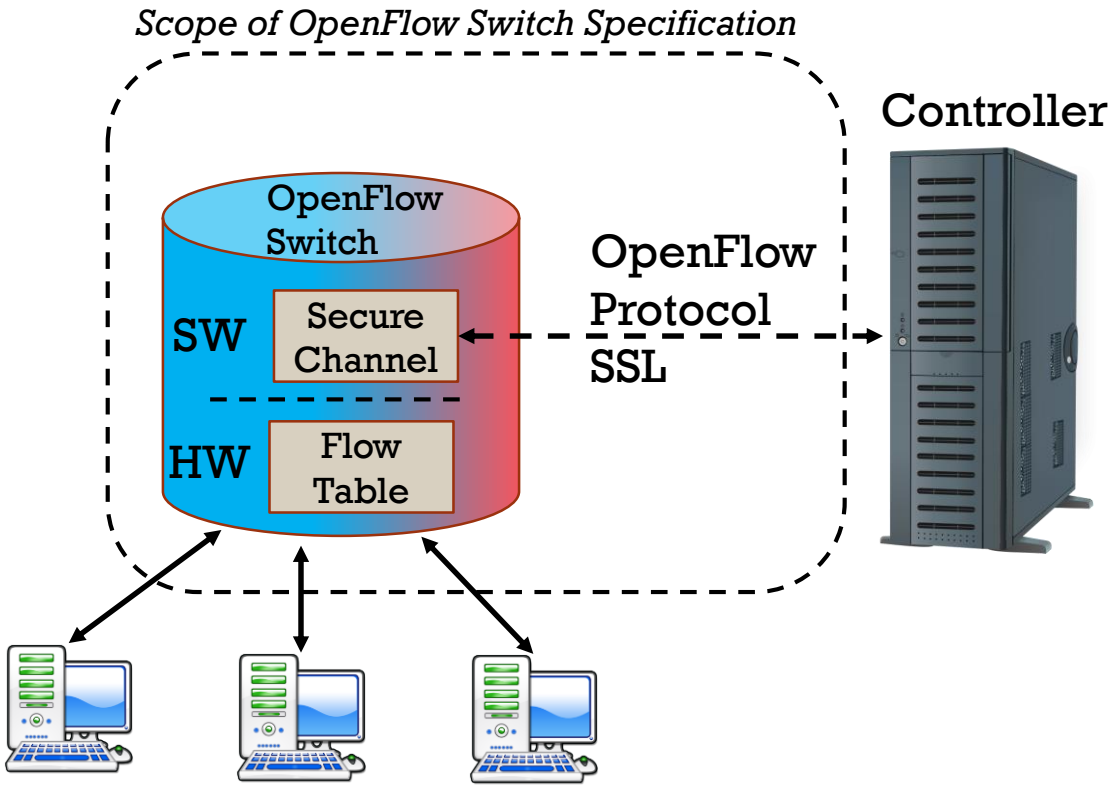

OpenFlow Controller
OpenFlow Protocol (SSL/TCP)
Control Path
OpenFlow
Data Path (Hardware)

OPENFLOW SWITCH

- OpenFlow-compliant switches come in two types:
  - **OpenFlow-only**: support only OpenFlow operation, in those switches all packets are processed by the OpenFlow pipeline, and cannot be processed otherwise.
  - **OpenFlow-hybrid**: support both OpenFlow operation and normal Ethernet switching operation, i.e. traditional L2 Ethernet switching, VLAN isolation, L3 routing (IPv4 routing, IPv6 routing…), ACL and QoS processing.
  - The **OpenFlow pipeline** of every OpenFlow switch contains multiple flow tables, each flow table containing multiple flow entries.

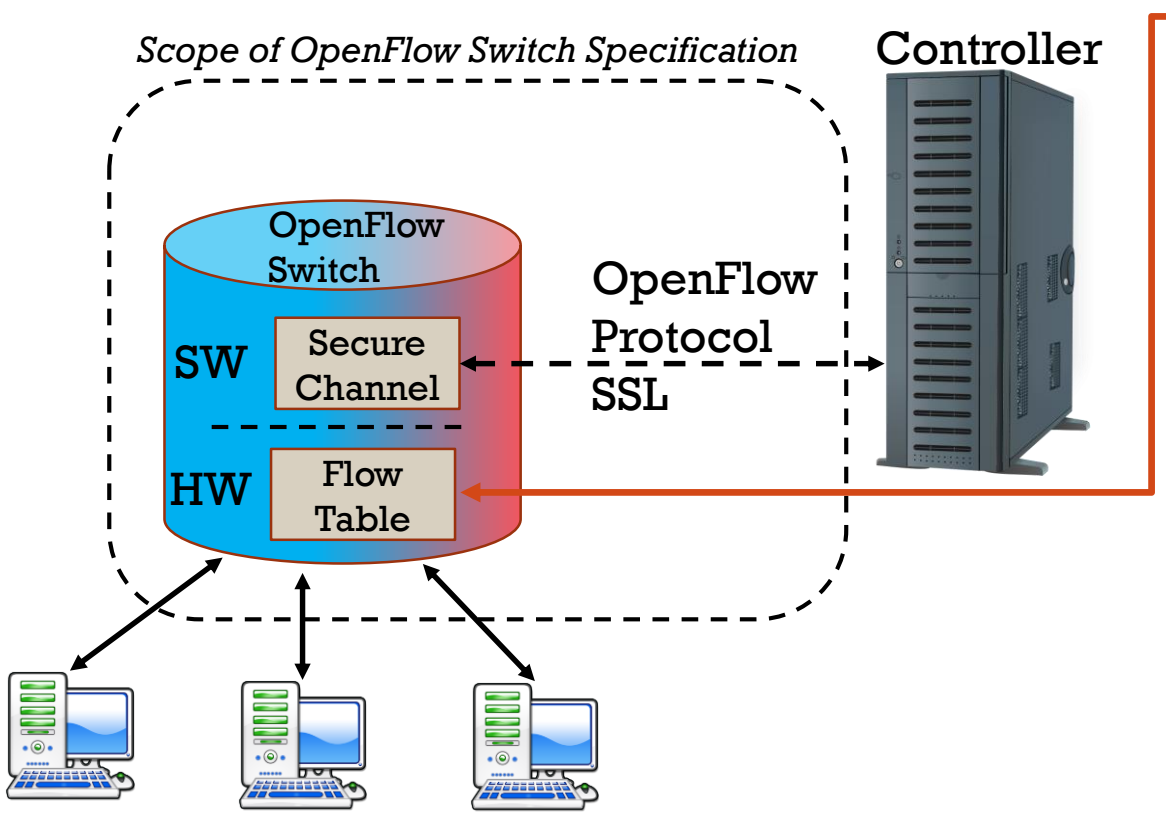
OPENFLOW

- OpenFlow is a Layer 2 communications protocol that gives access to the forwarding plane of a network switch or router over the network.



Idealized OpenFlow Switch. The Flow table is controlled by a remote controller via a secure channel.

OPENFLOW SWITCH V1.0



Flow Table

Rule	Action	Stats

Packet + byte counters

1. Forward packet to port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline
5. Modify field.

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport

+ mask

The Stanford Clean Slate Program, http://cleanslate.stanford.edu

FLOW ENTRY EXAMPLES

Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f:..	*	*	*	*	*	*	*	port6

Flow Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
port3	00:20..	00:1f..	0800	vlan1	1.2.3.4	5.6.7.8	4	17264	80	port6

Firewall

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	*	*	*	22	drop

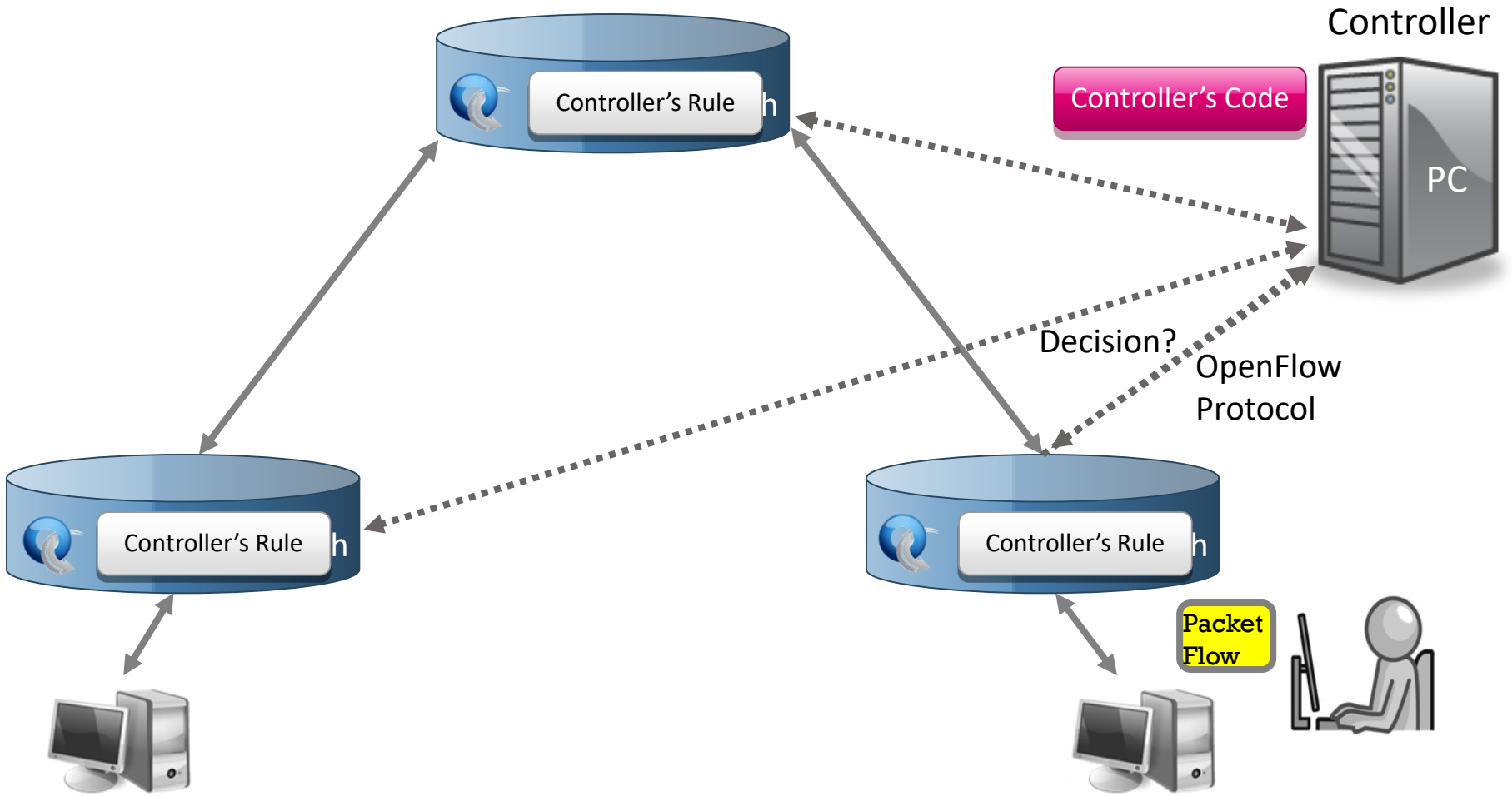
Routing

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	5.6.7.8	*	*	*	port6

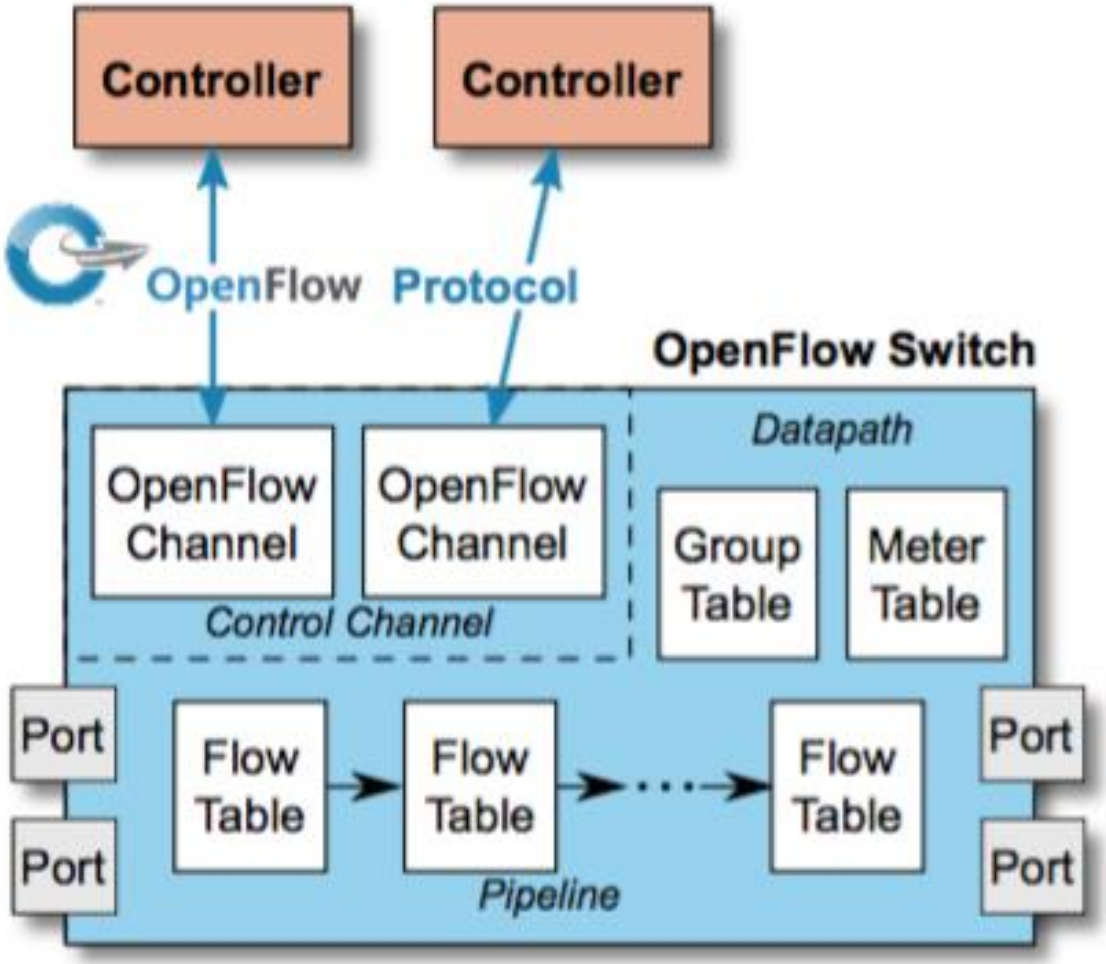
VLAN Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f..	*	vlan1	*	*	*	*	*	port6, port7, port9

OPENFLOW FLOW RULES SETUP

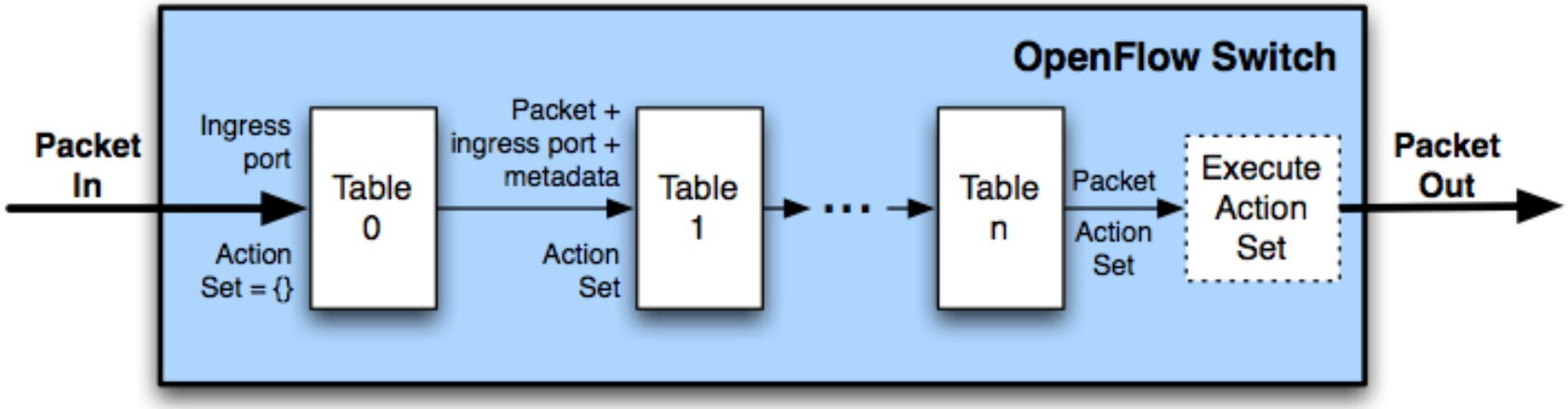


OPENFLOW V 1.5 (MAIN COMPONENTS)

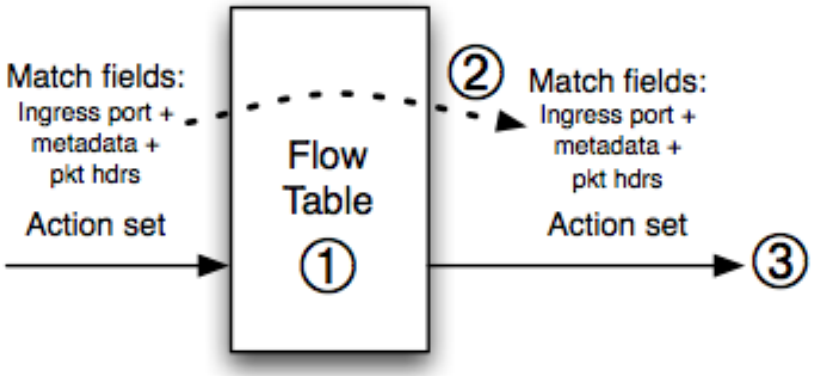


ONOS Openflow 1.5 implementation:
https://wiki.onosproject.org/display/ONOS/OpenFlow+1.5+Implementation

OPENFLOW MATCHING PROCESS



(a) Packets are matched against multiple tables in the pipeline



① Find highest-priority matching flow entry

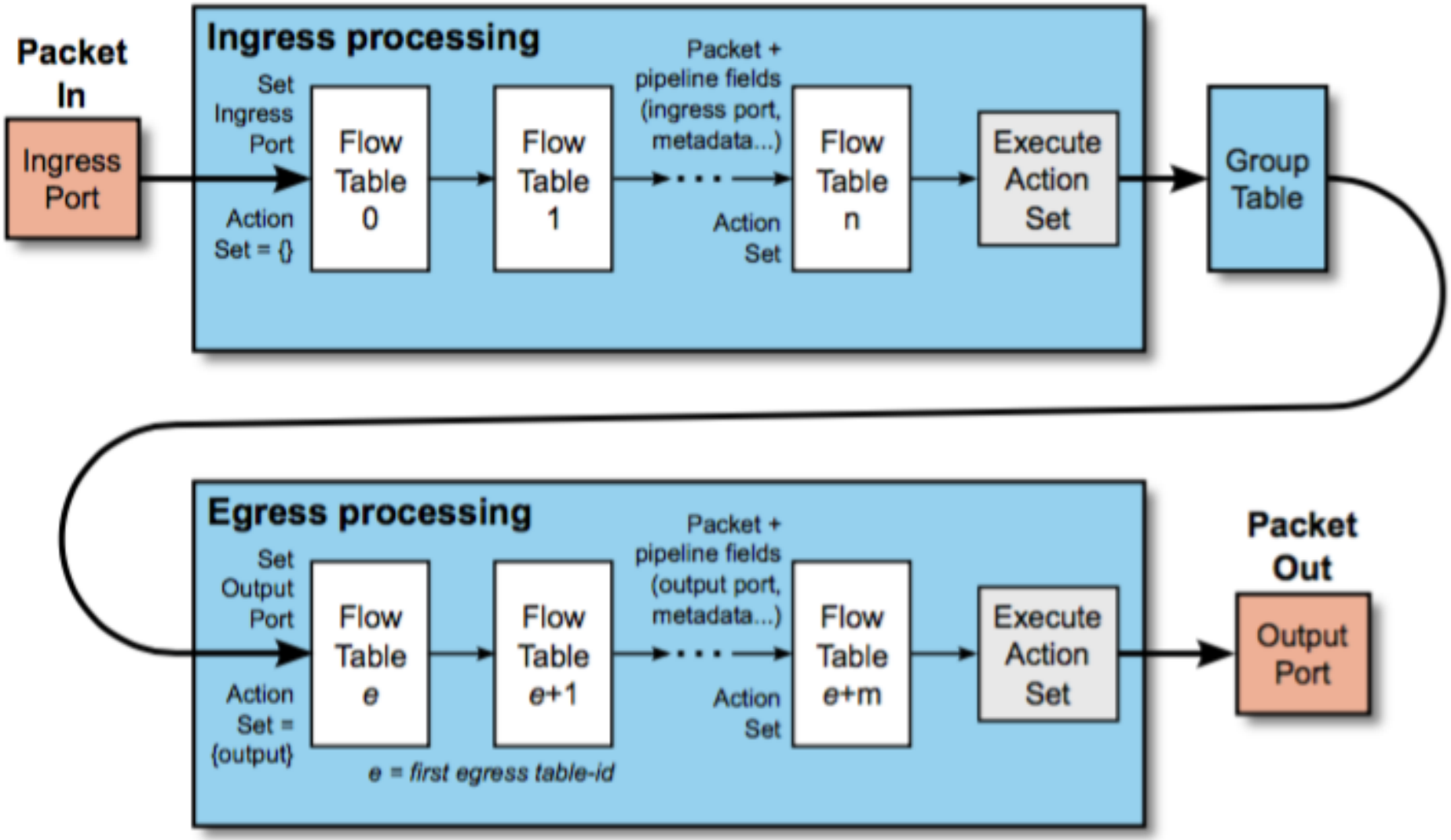
② Apply instructions:
- i. Modify packet & update match fields (apply actions instruction)
- ii. Update action set (clear actions and/or write actions instructions)
- iii. Update metadata

③ Send match data and action set to next table

(b) Per-table packet processing

ONF, "*OpenFlow Swith Specification v.1.3.0*", June, 2012

OPENFLOW V 1.5 (PROCESSING PIPELINE)

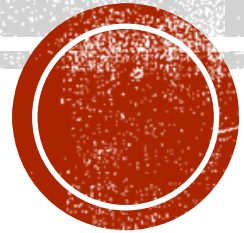


ONOS Openflow 1.5 implementation:
https://wiki.onosproject.org/display/ONOS/OpenFlow+1.5+Implementation

OPENFLOW CONTROLLERS

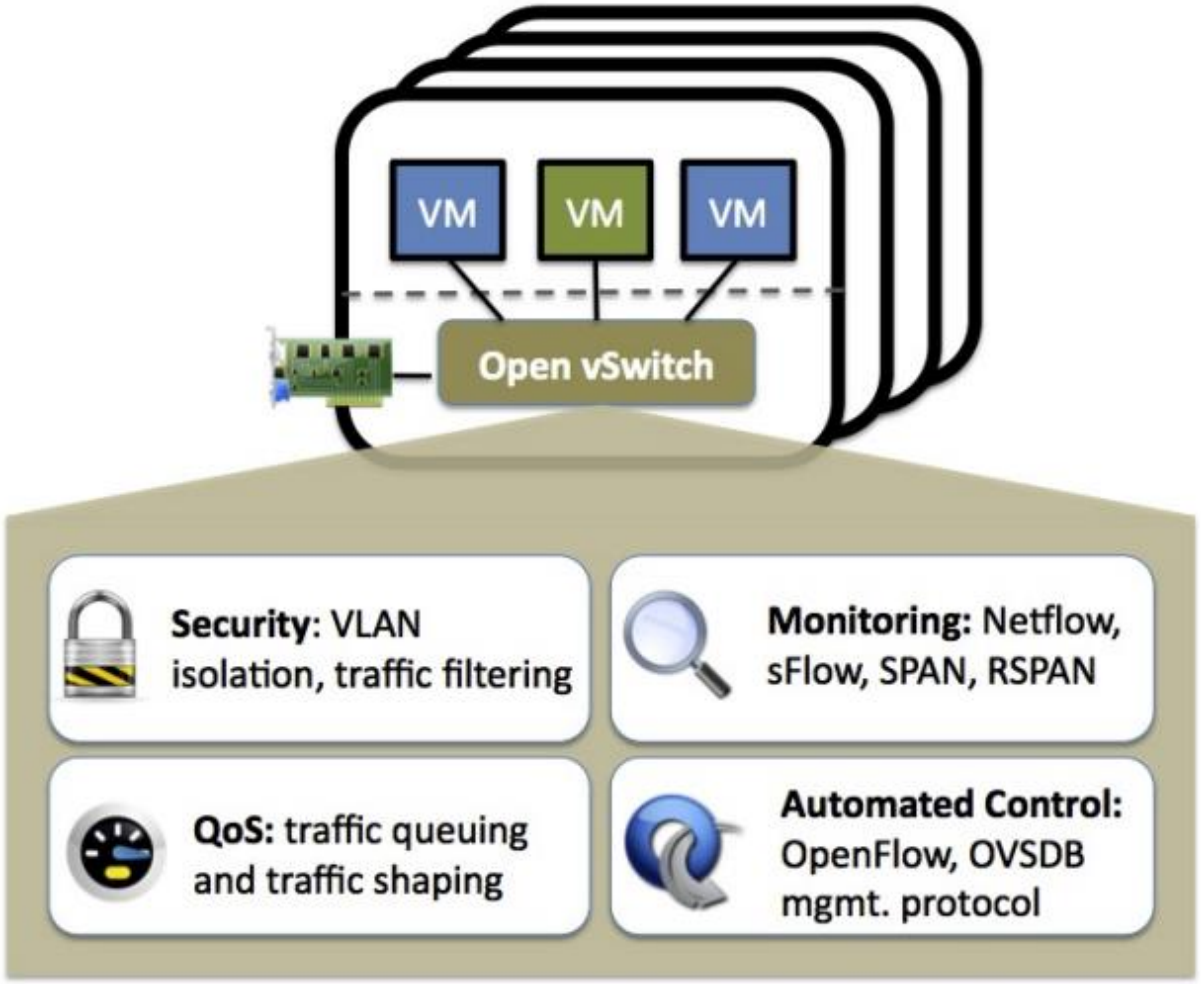
Name	Language	Platform(s)	Original Author	Notes
Openflow Reference	C	Linux	Stanford/Nicira	Not designed for extensibility
Nox/Pox	Python, C++	Linux	Nicira	Actively developed
Beacon	Java	Win, Mac, Linux, Android	David Erickson (standford)	Runtime modular, web UI framework, regression test framework
Trema	Ruby, C	Linux	NEC	Includes emulator, regression test framework
Ryu	Python	Linux	NTT	Supports Openstack and Openflow 1.3
Floodlight	Java	any	BigSwitch	Support Openstack and java based
ONOS	OSGi/Java	any	ONF	Support Openflow 1.5

OPENFLOW – AN SDN IMPLEMENTATION



- OpenFlow
- **Open vSwitch (OVS)**

WHAT IS OPEN VIRTUAL SWITCH?



Source Open vSwitch 2.11.90:
https://rtdcommunitypublic.blob.core.windows.net/media/pdf/openvswitch/latest/openvswitch.pdf

WHAT IS OPEN VSWITCH (CONTINUE)?

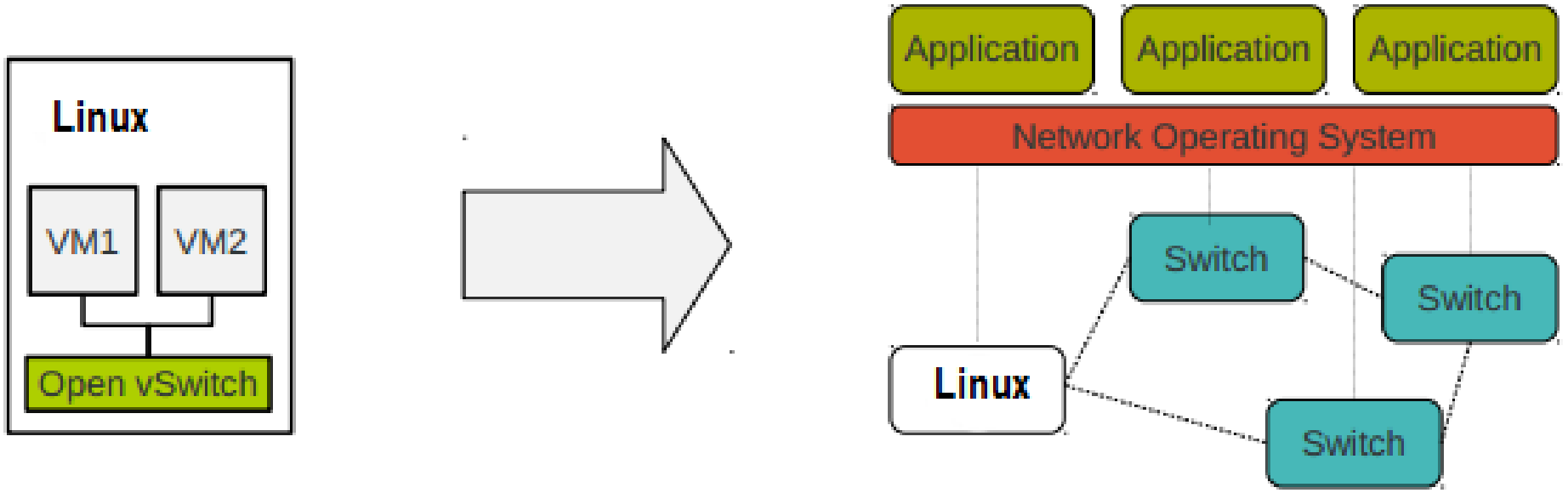
- Open vSwitch (OVS) is an OpenFlow-based multilayer software switch licensed under the open source Apache 2 license (User Space) and GPL (Kernel).
- OVS is a **virtual switch** for hypervisors providing network connectivity to VMs.
- It exposes **standard control** and **visibility interfaces** to the virtual networking layer.
- It was designed to support distribution **across multiple physical servers**.
- OVS supports multiple Linux-based virtualization technologies including Xen/XenServer, KVM, and VirtualBox.

OPEN VSWITCH'S FEATURES

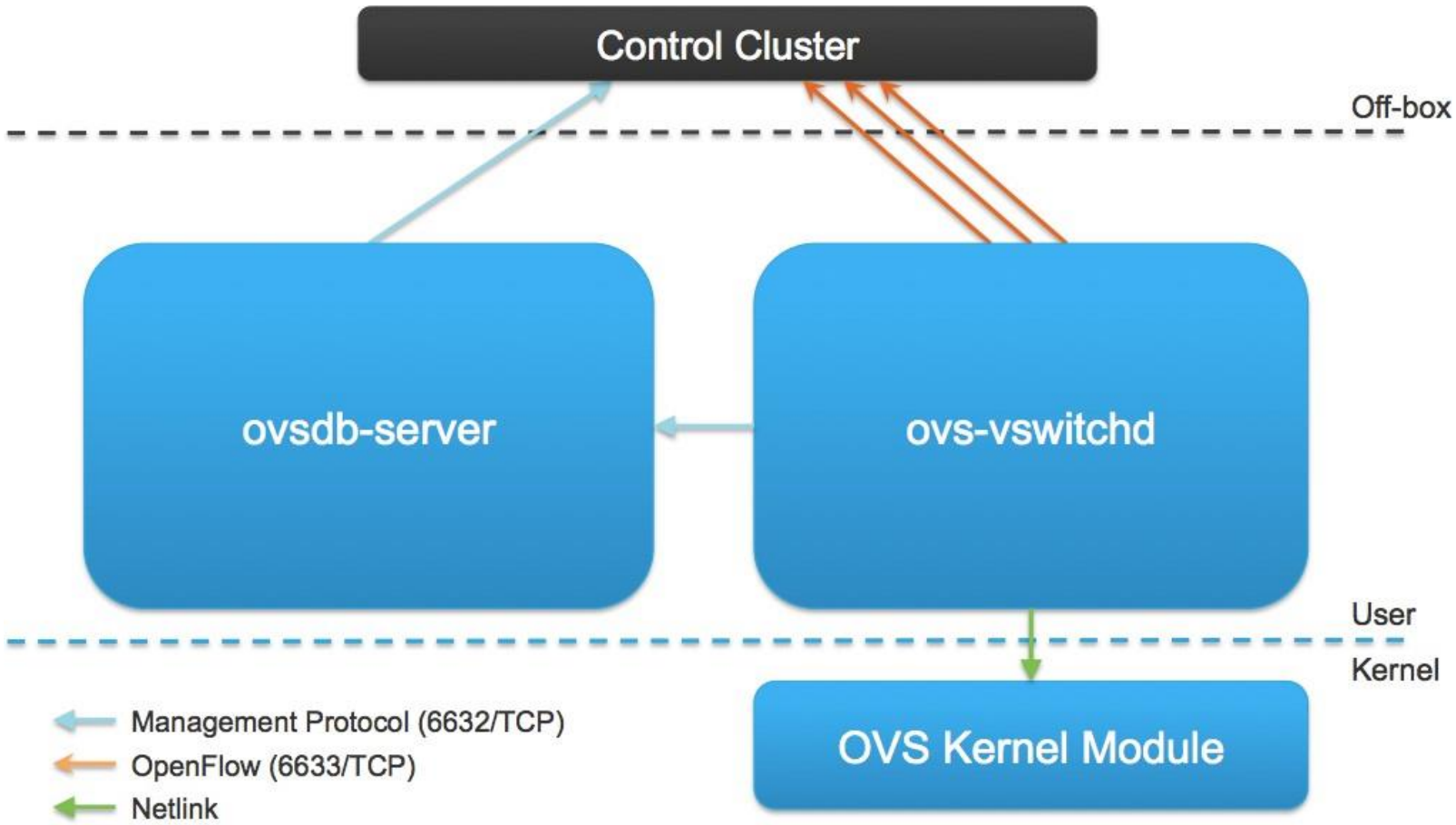
- Visibility:
  - NetFlow, sFlow, Mirroring (SPAN/RSPAN/ERSPAN)
- Control:
  - Centralized control through OpenFlow
    - Missed flows go to central controller
  - Fine-grained ACL and QoS (Quality of Service) policies
    - L2-L4 matching and actions to forward, drop, modify, and queue
- Forwarding:
  - LACP (Link Aggregate Control Protocol)
  - Port bonding
  - Standard 802.1Q VLAN model with trunk and access ports
  - GRE, GRE over IPSEC, Ethernet-over-GRE and CAPWAP tunneling
- Compatibility layer for Linux bridging code
- High-performance forwarding using a Linux kernel module

WHY OPEN VSWITCH?

- Open vSwitch enables Linux to become part of a SDN architecture.

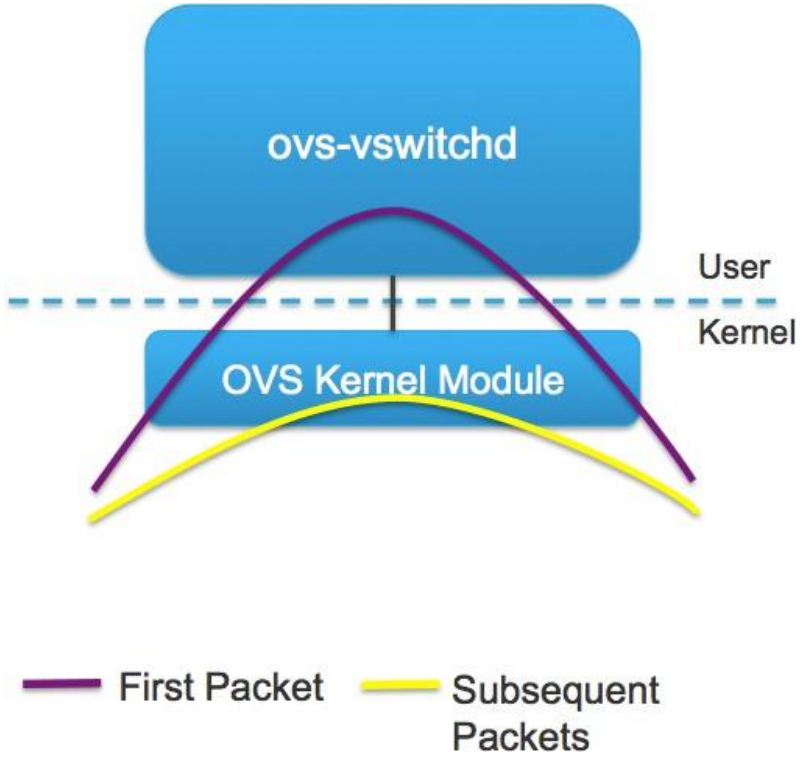


COMPONENTS

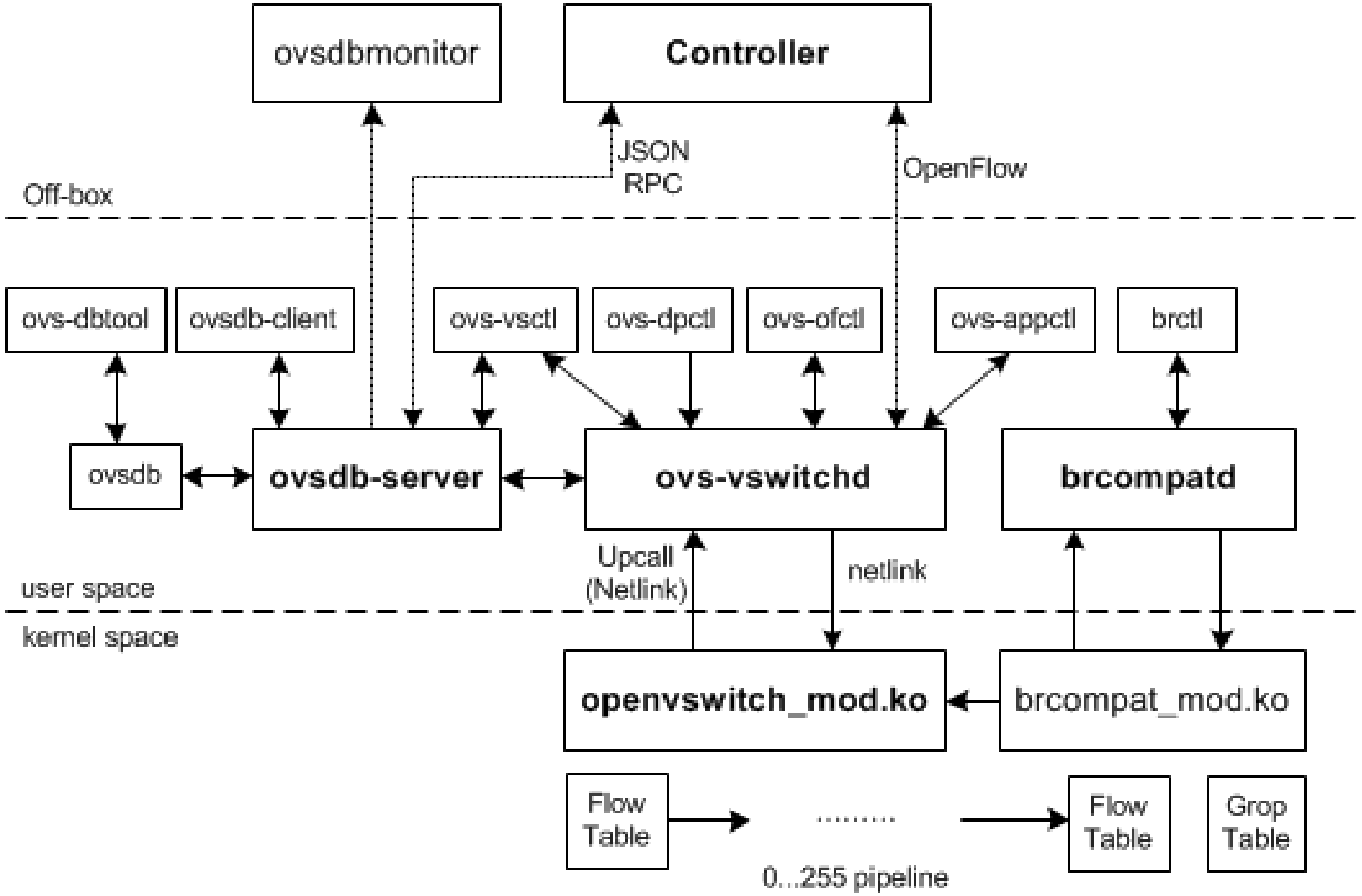

Control Cluster
Off-box
ovsdb-server
ovs-vswitchd
User
Kernel
OVS Kernel Module
Management Protocol (6632/TCP)
OpenFlow (6633/TCP)
Netlink

OPEN VSWITCH DESIGN

- Decision about how to process packet made in userspace
- First packet of new flow goes to ovs- vswitchd, following packets hit cached entry in kernel



MAIN COMPONENTS OF OVS



FORWARDING COMPONENTS

- Forwarding Components
  - ovs-vswitchd (control plane, slow path)
    - A daemon that implements the switch, along with a companion Linux kernel module for flow-based switching.
    - Forwarding logic (learning, mirroring, VLANs, and bonding)
    - Remote configuration and visibility
  - openvswitch_mod.ko (data plane, fast path)
    - Packet lookup, modification, and forwarding
    - Tunnel encapsulation/decapsulation

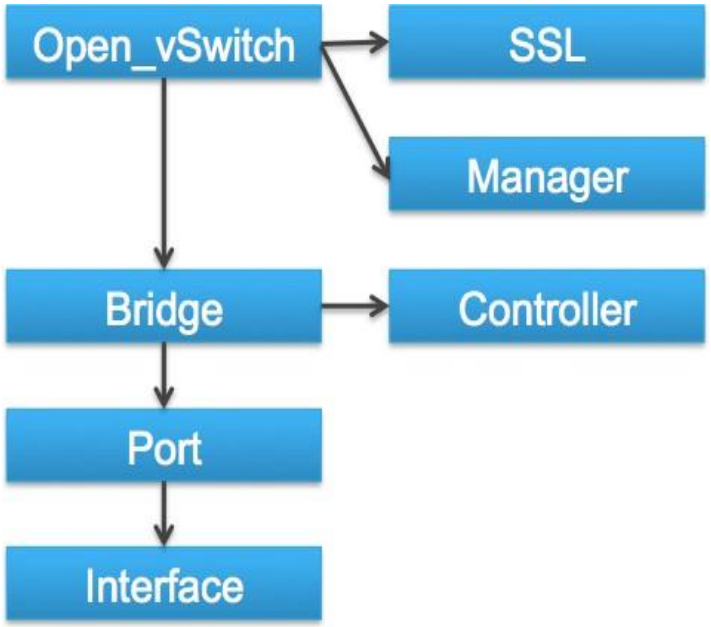
OTHER MODULES AND TOOLS

- ***ovsdb-server***: a lightweight database server that *ovs-vswitchd* queries to obtain its configuration.
- ***ovs-brcompatd***: a daemon that allows *ovs-vswitchd* to act as a drop-in replacement for the Linux bridge in many environments, along with a companion Linux kernel module to intercept bridge *ioctls*.
- ***ovs-dpctl***: a tool for configuring the switch kernel module.
- ***ovs-vsctl***: a utility for querying and updating the configuration of *ovs-vswitchd*.

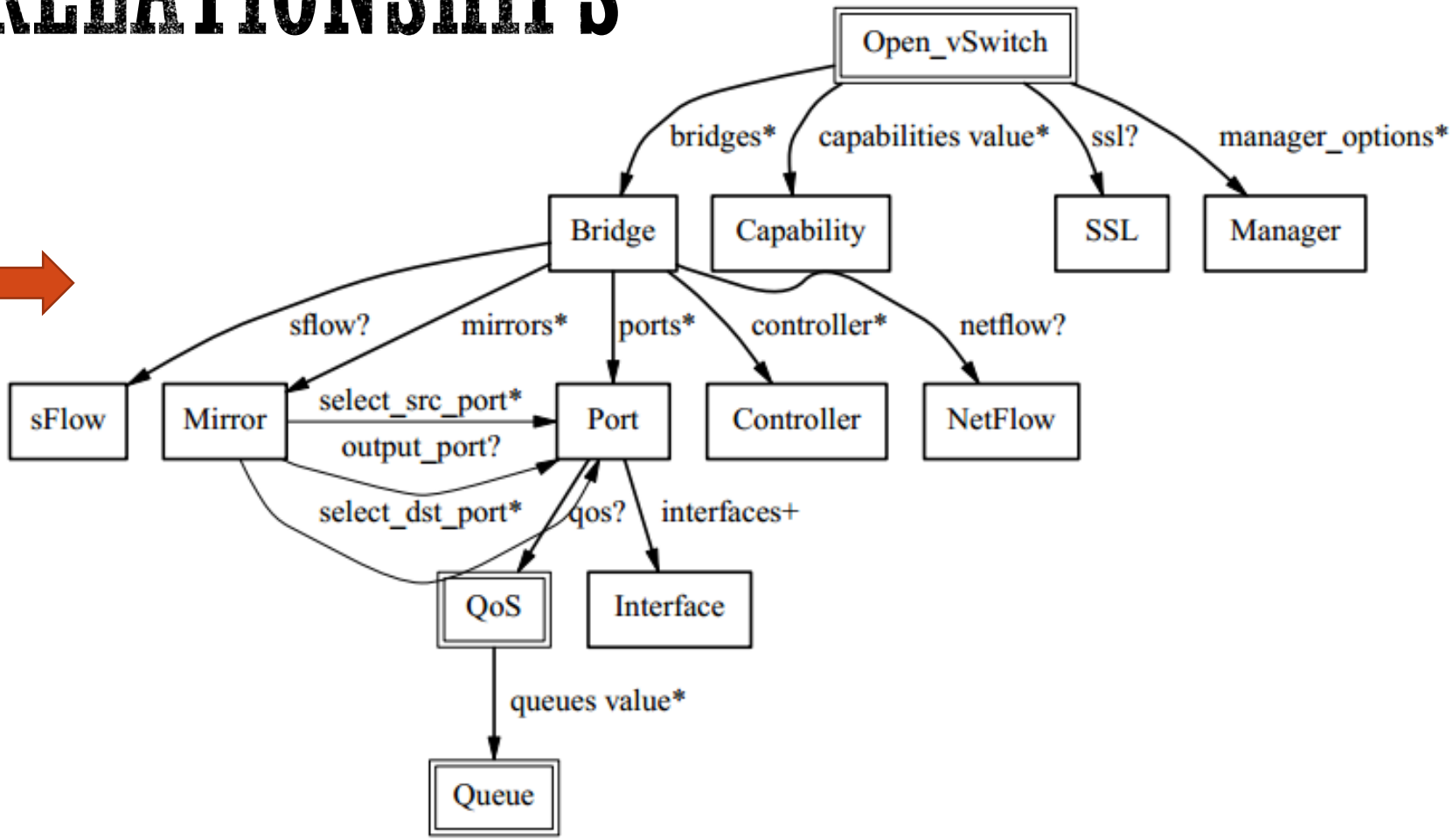
OTHER MODULES AND TOOLS

- ***ovs-appctl***: a utility that sends commands to running Open vSwitch daemons.
- ***ovs-ofctl***: a utility for querying and controlling OpenFlow switches and controllers.
- ***Ovsdbmonitor***: a GUI tool for remotely viewing OVS databases and OpenFlow flow tables.
- ***ovs-controller***: a simple OpenFlow controller.
- *ovs-pki*: a utility for creating and managing the public-key infrastructure for OpenFlow switches.

OVSDB TABLE RELATIONSHIPS



Simplified view

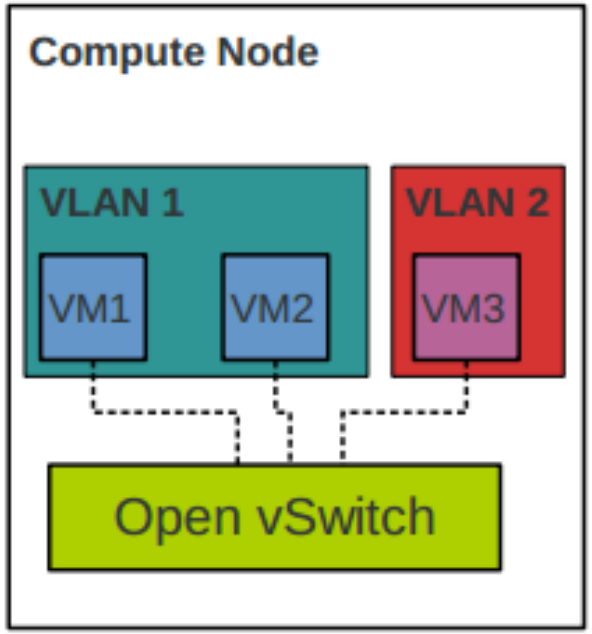


"Open_vSwitch" is the root table and there is always only a single row. The tables here are the ones most commonly used; a full entity-relationship diagram is available in the *ovs-vswitchd.conf.db* man page.

FEATURE EXAMPLE– SECURITY/L2 SEGREGATION

- VLAN isolation enforces VLAN membership of a VM without the knowledge of the guest itself.

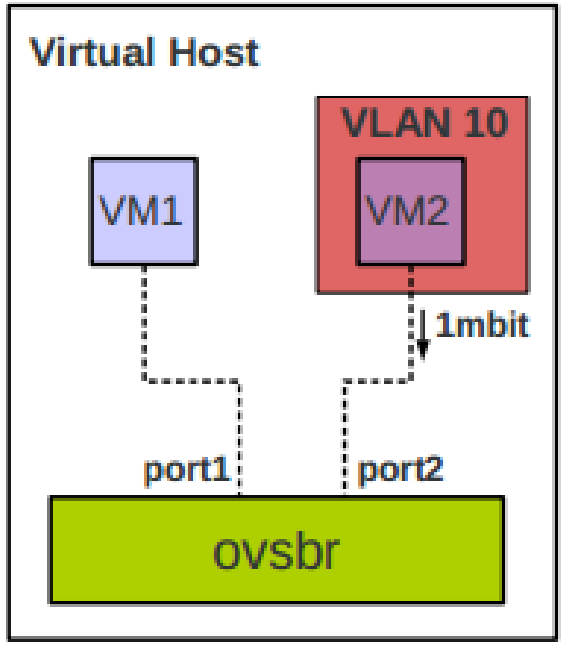
```
# ovs-vsctl add-port ovsbr port2 tag=10
```



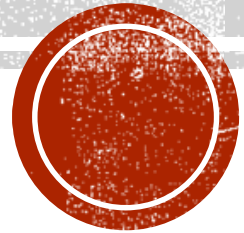
FEATURE EXAMPLE – QUALITY OF SERVICE

- Uses existing traffic control layer
  - Policer (Ingress rate limiter)
  - HTB, HFSC (Egress traffic classes)
- Controller can select traffic class

```
# ovs-vsctl set Interface port2 \
  ingress_policing_rate=1000
```



NETWORK FUNCTION VIRTUALIZATION (NFV)



- **NFV Overview**
- OpenStack Overview

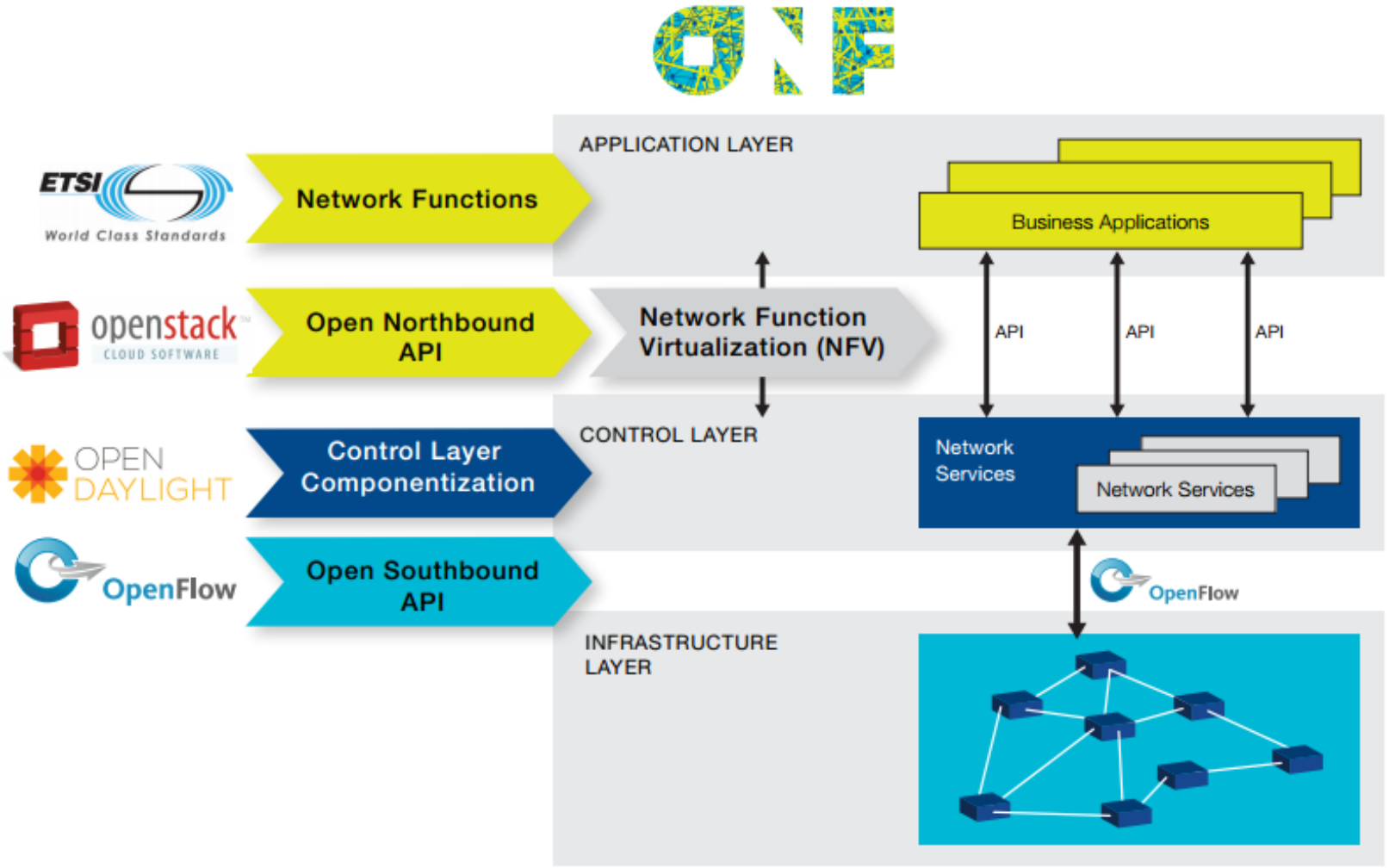
THE RELATION BETWEEN NFV AND SDN

- NFV can be achieved using non-SDN mechanisms
  - Already used in datacenters
- NFV supports SDN by providing infrastructure upon which the SDN can be implemented
  - Aligns closely with SDN objective to use commodity servers and switches

DIFFERENCES BETWEEN NFV AND SDN

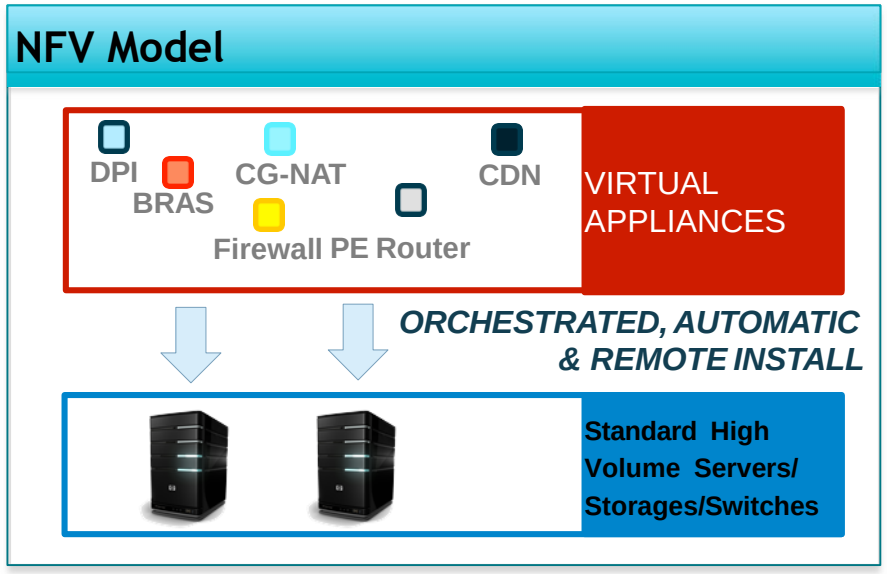
Software Defined Networking (SDN)		Network Function Virtualization (NFV)
Separate control and data, centralize control and programmability of network	Basic Concept	Relocate network functions from dedicated appliances to generic servers
Campus, data center / cloud	Target Location	Service provider network
Commodity servers and switches	Target Devices	Commodity servers and switches
Cloud orchestration and networking	Initial Applications	Routers, firewalls, gateways, CDN, WAN accelerators, SLA assurance
OpenFlow	New Protocols	None
Open Networking Foundation (ONF)	Formalization	ETSI NFV Working Group

OPEN PLATFORMS FOR NFV AND SDN



NFV OBJECTIVES...

"Leverage standard IT virtualisation technology to consolidate many network equipment types onto industry standard high volume servers, switches, and storage"

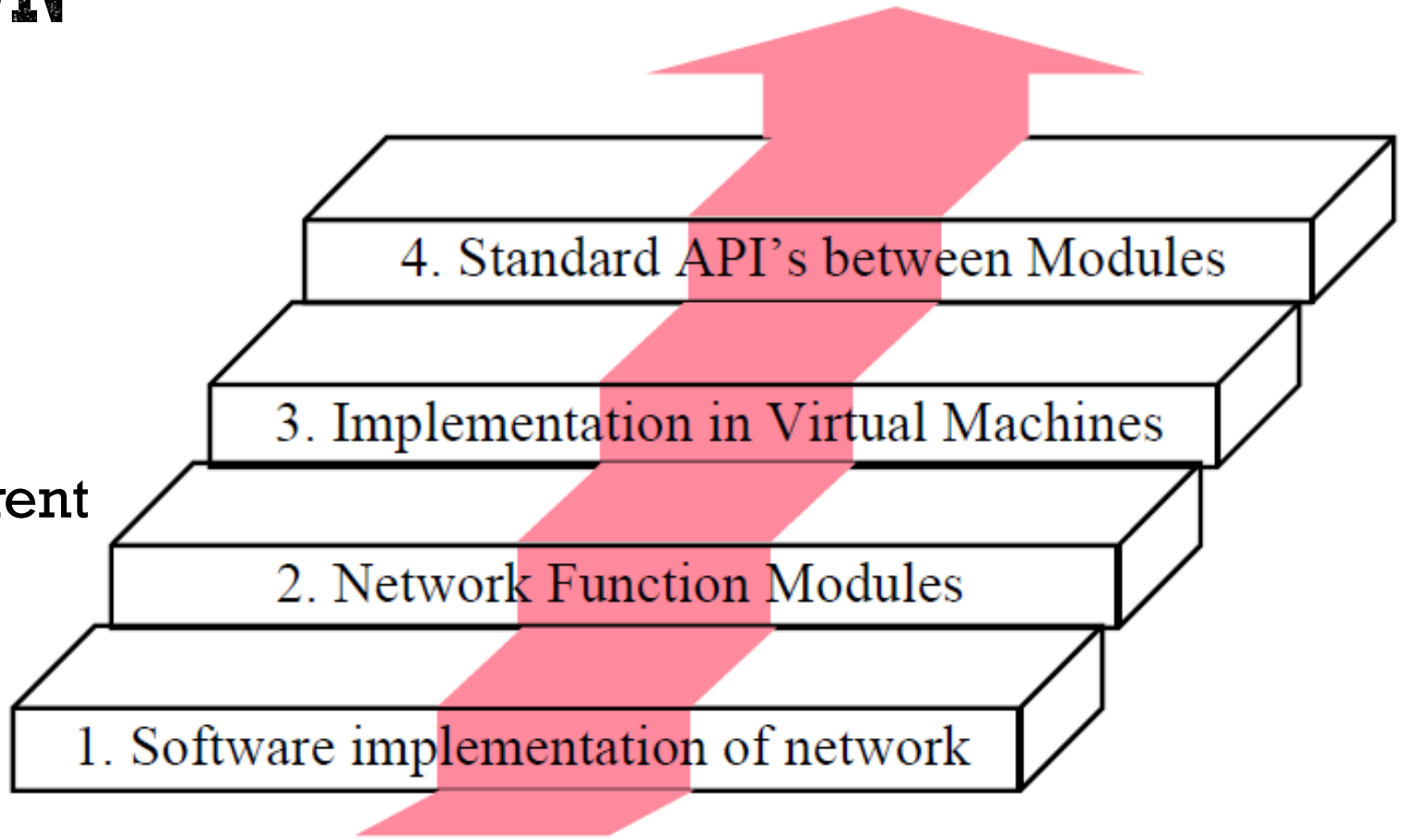


- Have fast standard hardware
  - White box implementation
- Decouple network functions from proprietary hardware appliances
  - Software implementation of network
- Standard API's between Modules

Source: Telefonica SDN-NFV: an introduction

NFV INNOVATION

- Implement network functions in software
  - Leverage standard servers and virtualization
  - Allows use of a single physical platform for different applications, users and tenants
  - Facilitate innovation
  - Ability to automate configuration and management



Source: SDN and NFV: Facts, Extensions, and Carrier Opportunities, Prof. Raj Jain

BENEFITS OF NFV

- Reduced costs (CAPEX)
  - Equipment cost reduction through consolidation on high volume industry standard servers
  - Operational cost reduction through reduced power, reduced space, improved network monitoring
  - With added flexibility dynamically provision and instantiate new services in various locations
- Increased speed of time-to-market
  - Reduce operator cycle of innovation
  - More customization and service differentiation
- Improved operational efficiency
  - Allows for multi-version and multi-tenancy
  - Allows a single platform for different applications, users and tenants
- Rapidly prototype and test new services and generate new revenue streams
  - Encourages openness
  - Encourages innovation
- Globally available workforce

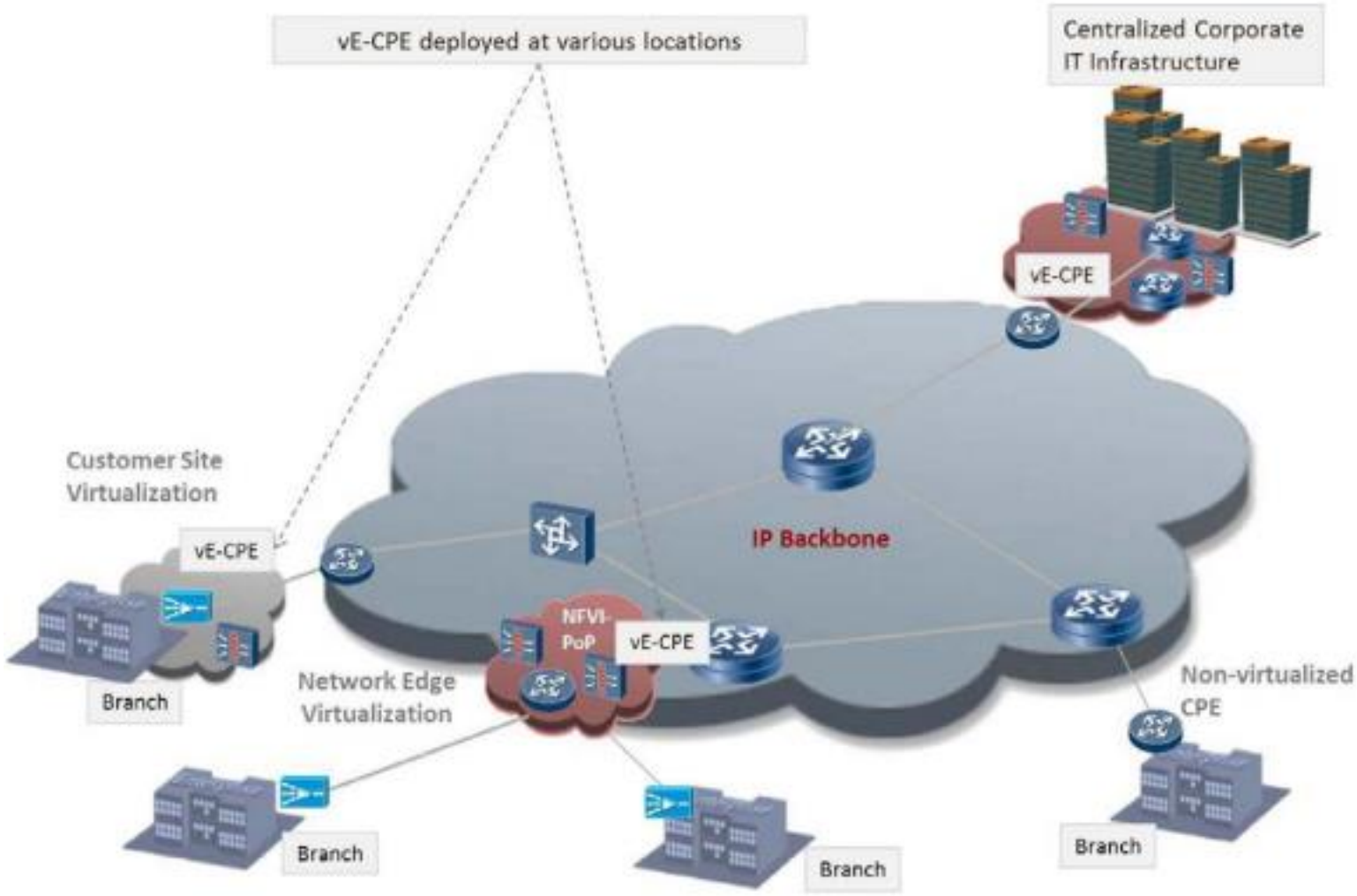
NFV USE CASES

ETSI Formalized NFV Use Cases	Potentially Virtualized Functions
Network Functions Virtualization Infrastructure as a Service	vNAT, vFW, vLB, vRR, vVPN, vRouter
Virtual Network Function as a Service (VNFaaS)	vCPE, vPE
Virtual Network Platform as a Service (VNPaaS)	vPrivateCloud
VNF Forwarding Graphs	VPE-F,
Virtualization of Mobile Core Networks and IMS	vEPC (vS/P-GW, vMME, vPCRF, vSGSN, vGGSN, vGiLan) vIMS (vP/S/I-CSCF, vMGCF, vAS)
Virtualization of Mobile Base Station	vMAC, vRLC, vPDCP, vRRC, vCOMP, vBBU
Virtualization of the Home Environment	vBNG, vRGW, vSTB
Virtualization of CDNs	vCDN,
Fixed Access Network Functions Virtualization	vOLT, vDSLAM, vONU, vONT, vMDU, vDPU

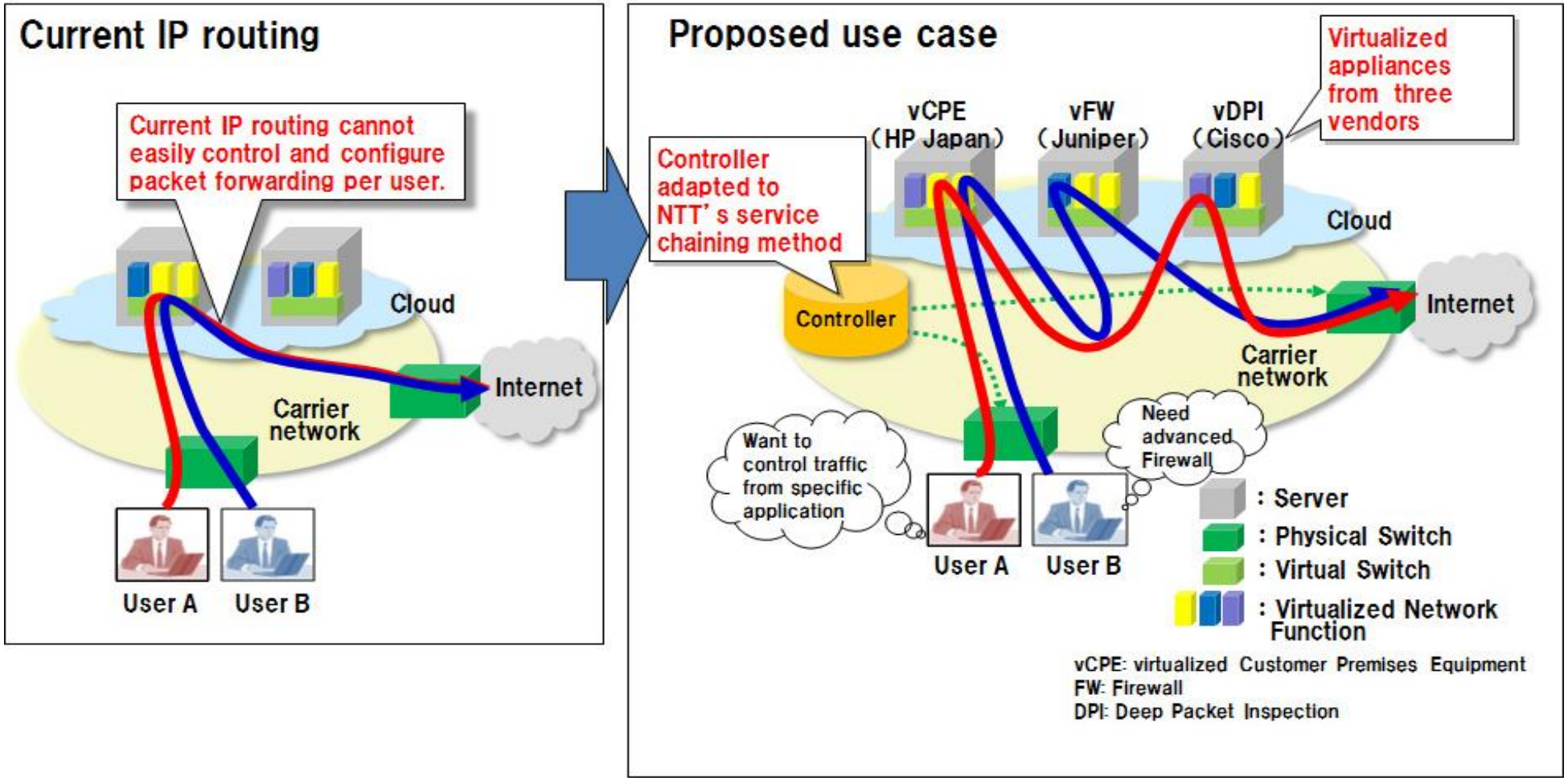
TELCO NETWORKS

- Mobile network traffic is significant
  - ARPU (Average Revenue Per User) is difficult to increase
    - Value-add services
  - LTE is considered as radio access
    - Data speeds dependent on high spectral efficiency, high peak data rates, short RTT and frequency flexibility in Radio Access Network (RAN)
  - Virtualization of base station!

EXAMPLES: VCPE



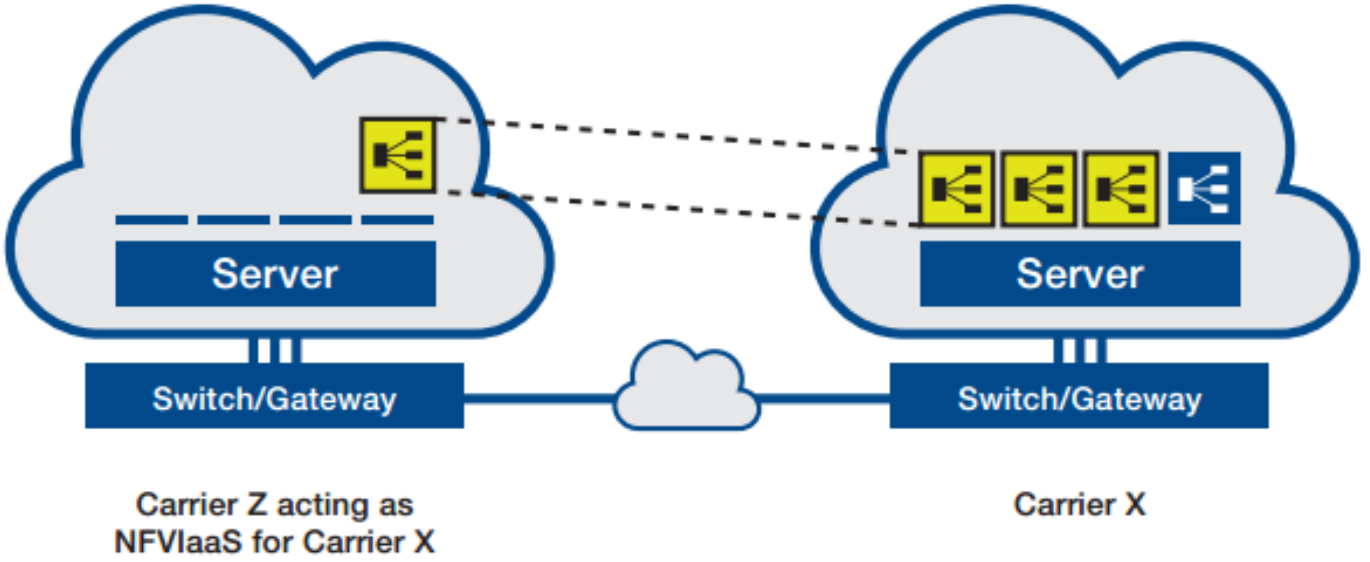
EXAMPLE: SERVICE CHAINING



Source: ETSI

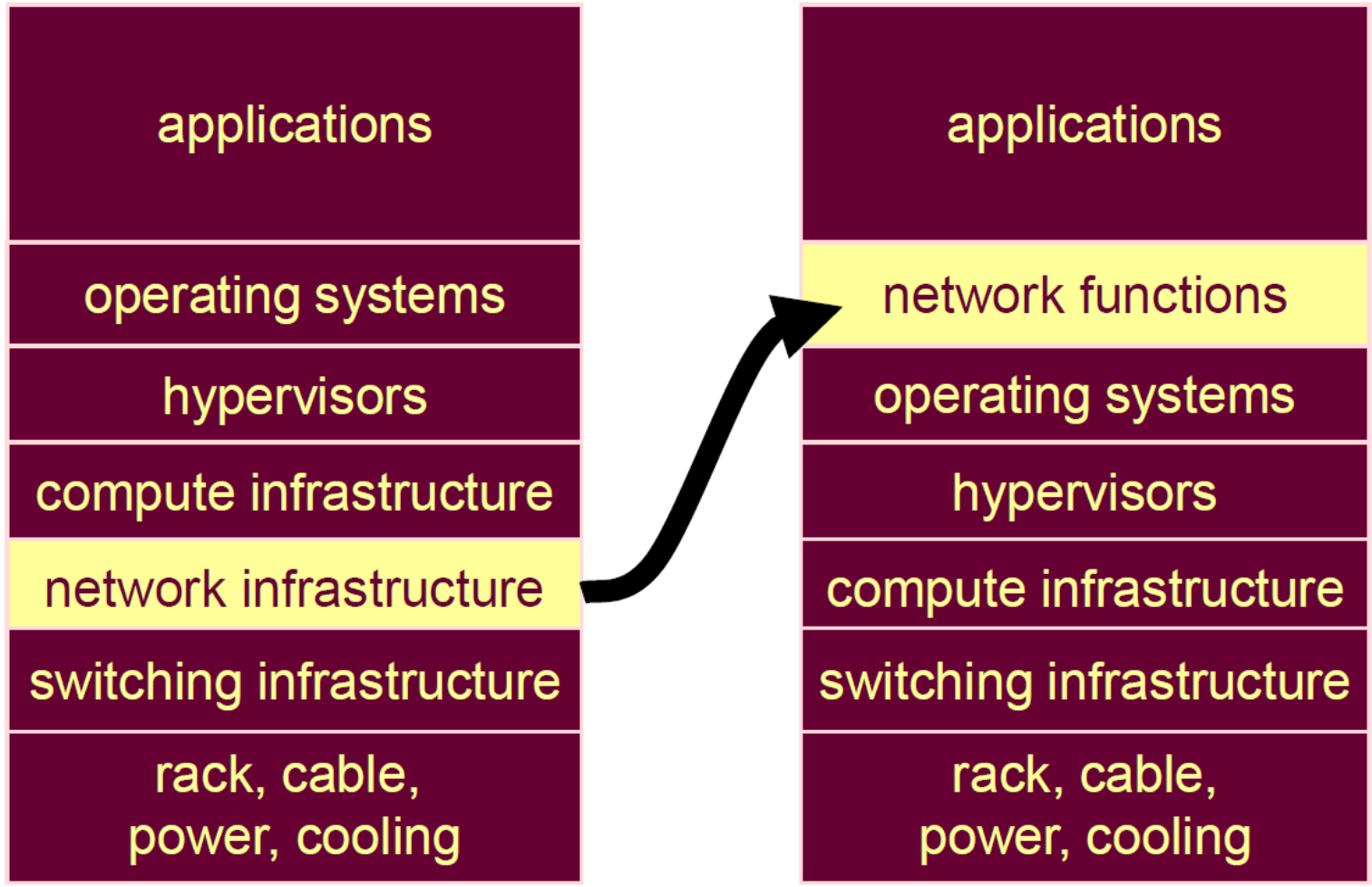
EXAMPLE: NFVIAAS

- One SP can leverage IaaS provided using NFV setup of another service provider
- Requirements:
  - A common automation framework capable of provisioning both physical and virtual infrastructures
  - A common deployment model that spans service provider and geographical boundaries

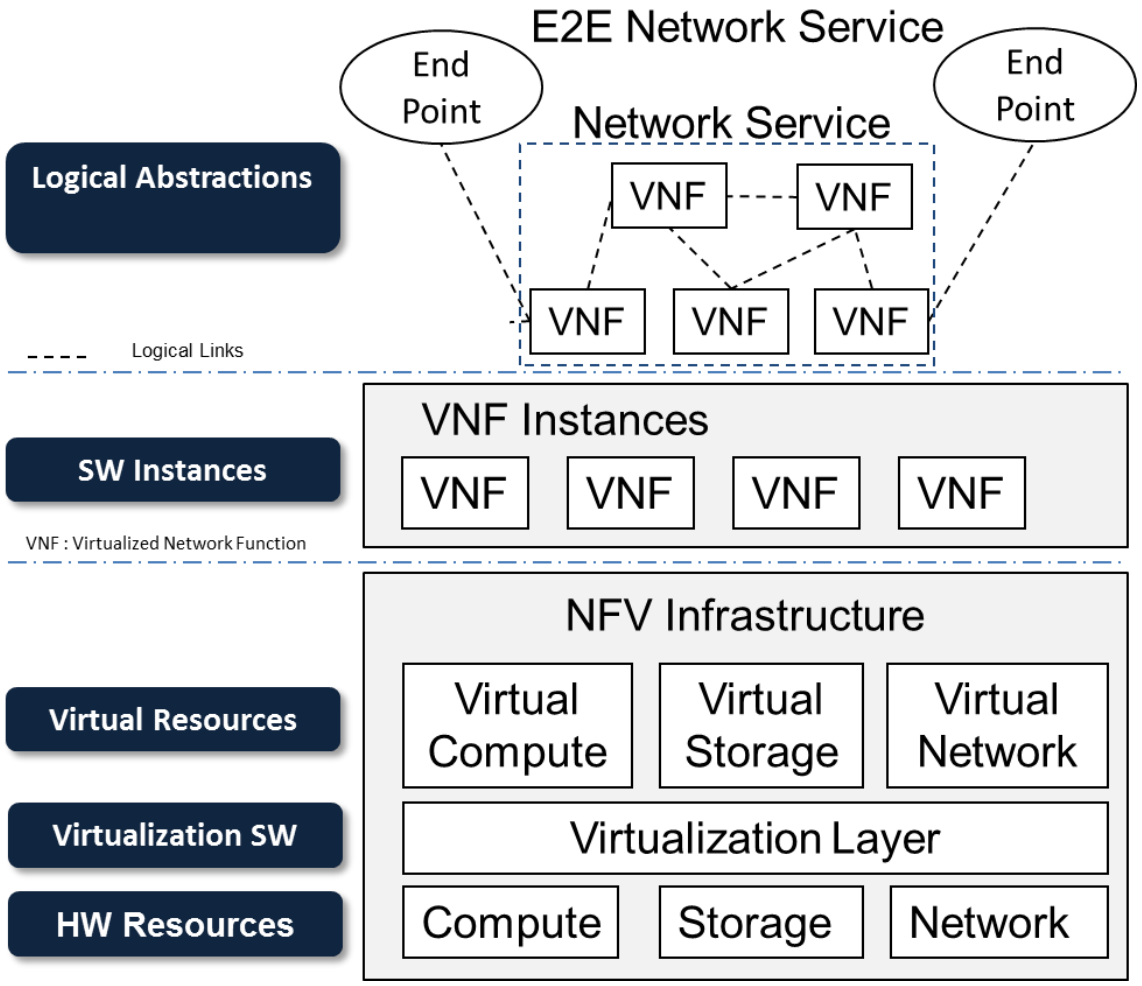


Source: ETSI

REVISITING LAYERS



NFV LAYERS

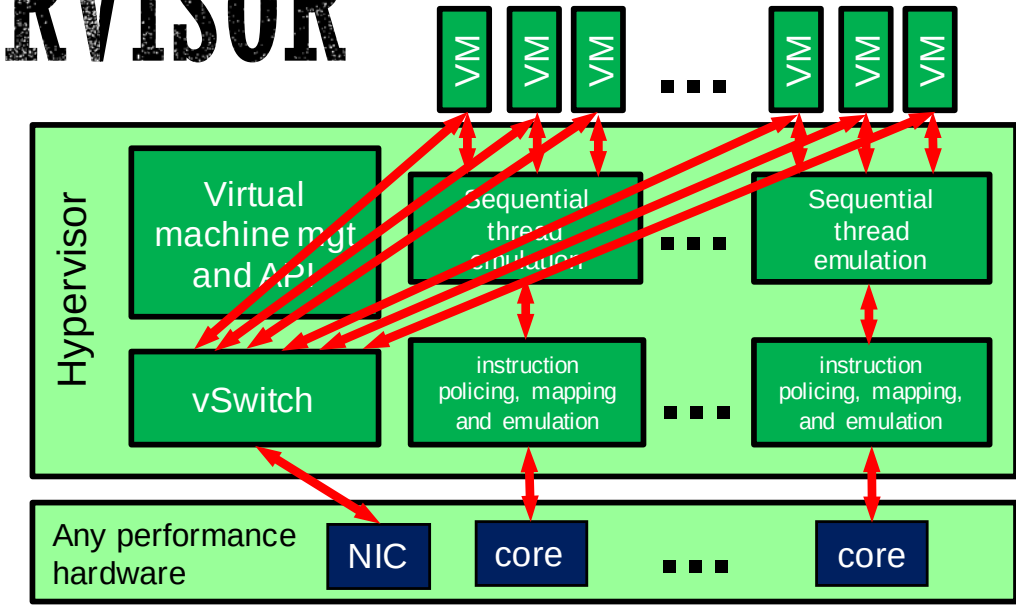

E2E Network Service
End Point
End Point
Network Service
Logical Abstractions
VNF
VNF
VNF
VNF
VNF
Logical Links
VNF Instances
SW Instances
VNF
VNF
VNF
VNF
VNF : Virtualized Network Function
NFV Infrastructure
Virtual Resources
Virtual Compute
Virtual Storage
Virtual Network
Virtualization SW
Virtualization Layer
HW Resources
Compute
Storage
Network

HYPERVISOR DOMAIN

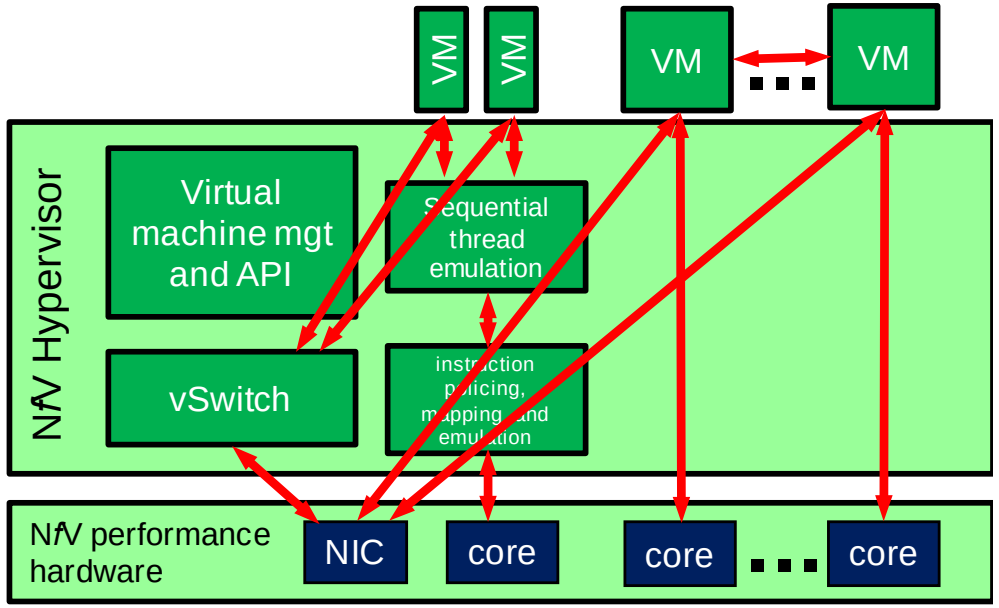
- General cloud hypervisor is designed for maximum application portability
  - Hypervisor creates
    - Virtual CPUs
    - Virtual NICs
  - Hypervisor provides
    - Virtual Ethernet switch
  - Hypervisor fully hides real CPUs and NICs

HYPERVISOR VS NFV HYPERVISOR

- General cloud hypervisor is designed for maximum application portability
  - Hypervisor creates
    - Virtual CPUs
    - Virtual NICs
  - Hypervisor provides
    - Virtual Ethernet switch
  - Hypervisor fully hides real CPUs and NICs



- NFV Hypervisor is aimed at removing packet bottlenecks
  - Direct binding of VM to core
  - Direct communication between VMs and between VMs and NIC
    - User mode polled drivers
    - DMA remapping
    - SR-IOV (single root input/output virtualization)
  - Many features already emerging in hypervisors



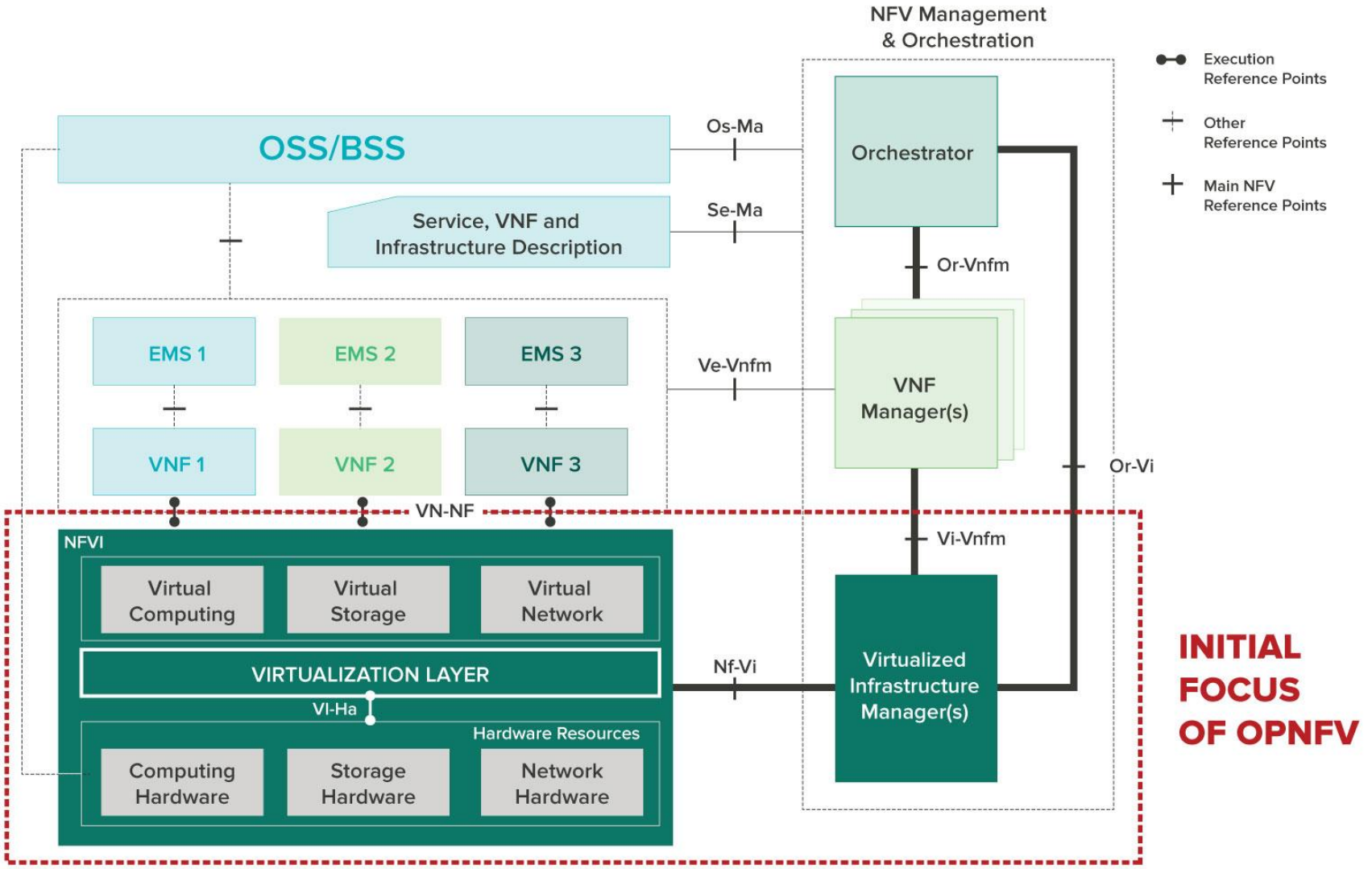
Source: Bob Briscoe, Don Clarke, Pete Willis, Andy Reid, Paul Veitch

ORCHESTRATION & MANAGEMENT DOMAIN

- Automated deployment of NFV applications
  - Orchestration console
  - Higher level carrier OSS
- Tools exist for automated cloud deployment
  - vSphere
  - Openstack
  - Cloudstack

OPEN PLATFORM FOR NFV (OPNFV)

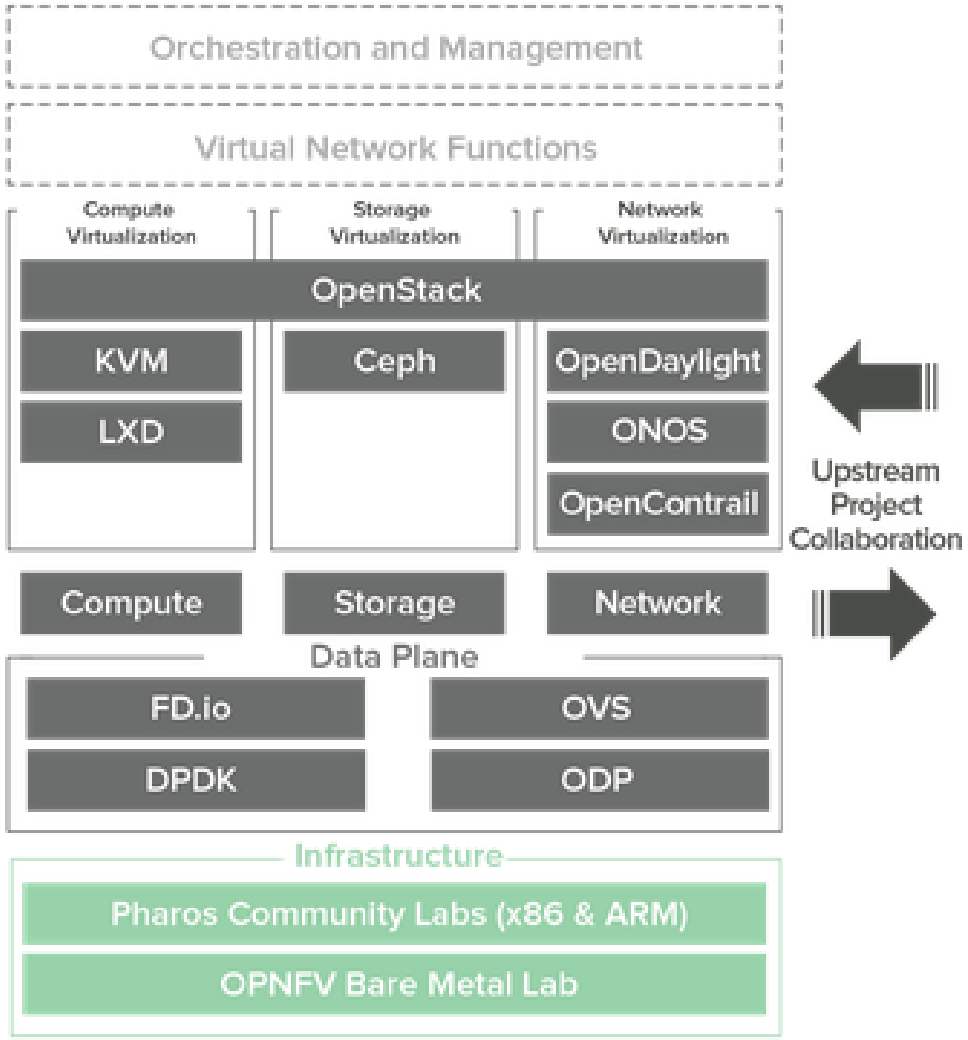
Notations:
NFVI – NFV Infrastructure
VIM – Virtualized Infrastructure Management API for other components of NFV
OSS – Operational Support System
BSS – Business Support System
EMS - Element Management System



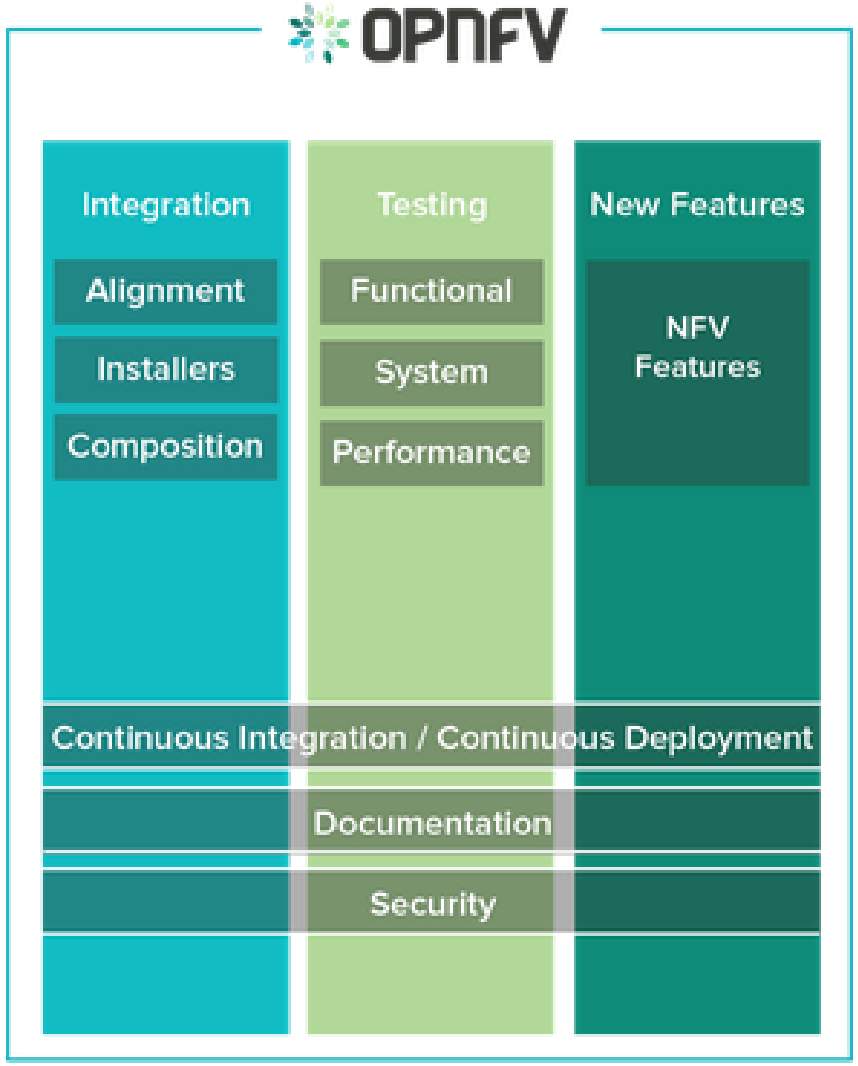
Source: Downley et al. explained an open NFV platform

OPNFV

- Develop and test an integrated open source platform
- Include participation of end users
- Contribute to and participate in relevant open source projects
- Establish an ecosystem for NFV solutions

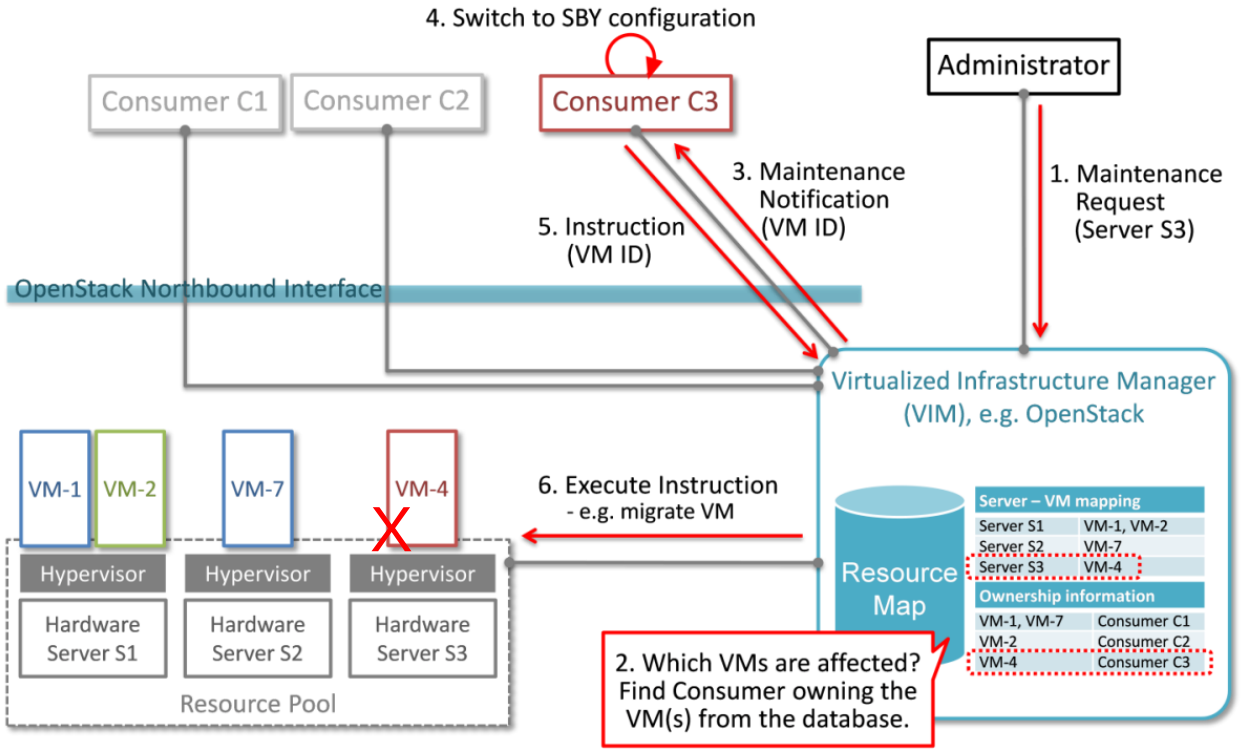


Upstream Project Collaboration



OPNFV CASE STUDY

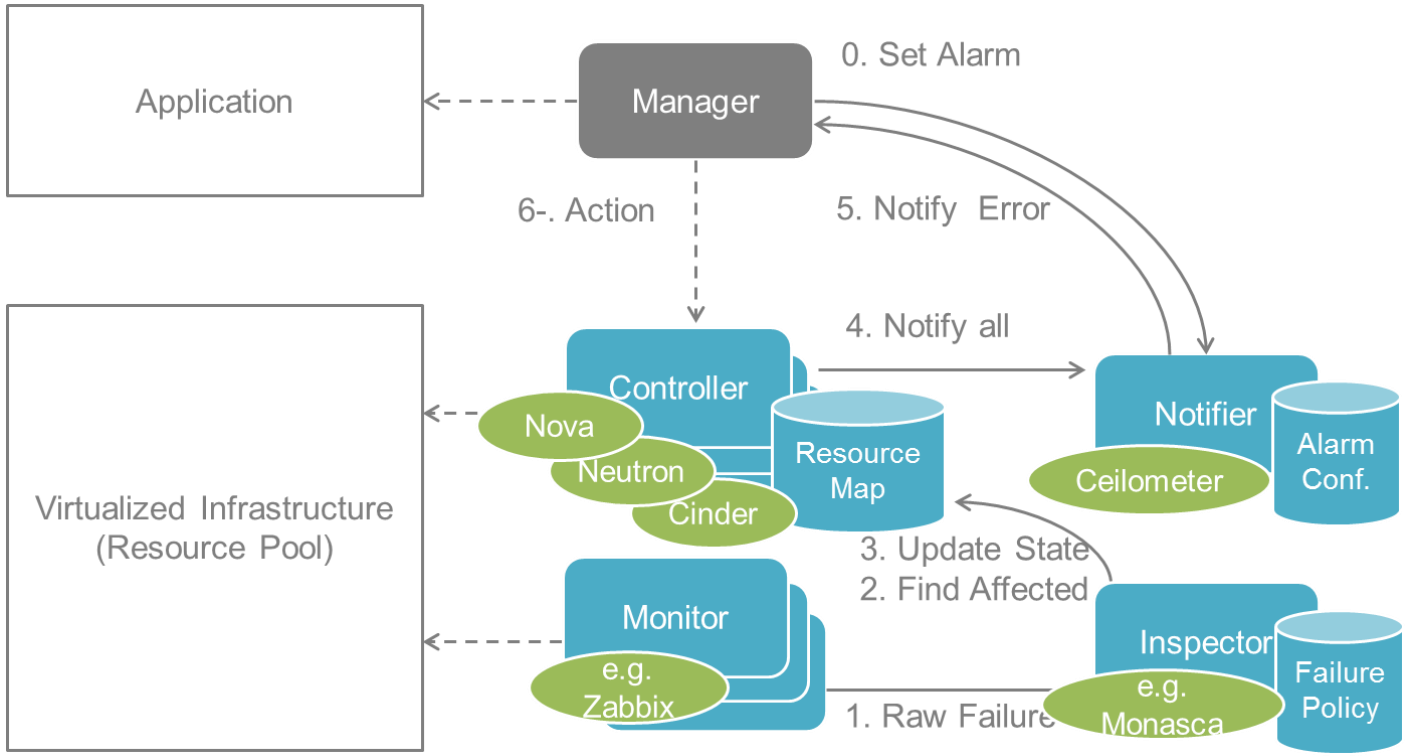
- Project Doctor
  - Fault management and maintenance framework



Source: https://wiki.opnfv.org/display/doctor

OPNFV CASE STUDY

- Project Doctor
  - Fault management and maintenance framework

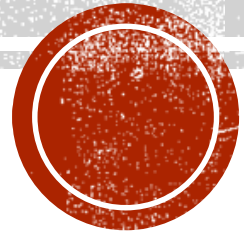


Source: https://wiki.opnfv.org/display/doctor

OPNFV MANO

- Management and orchestration modeling
  - Enable successful NFV in an increasingly divergent industry
  - Place low barrier to VNF development, on-boarding and SLA assurance
- Implement industry wide standards
- Implement interoperability between VNF vendors

NETWORK FUNCTION VIRTUALIZATION (NFV)

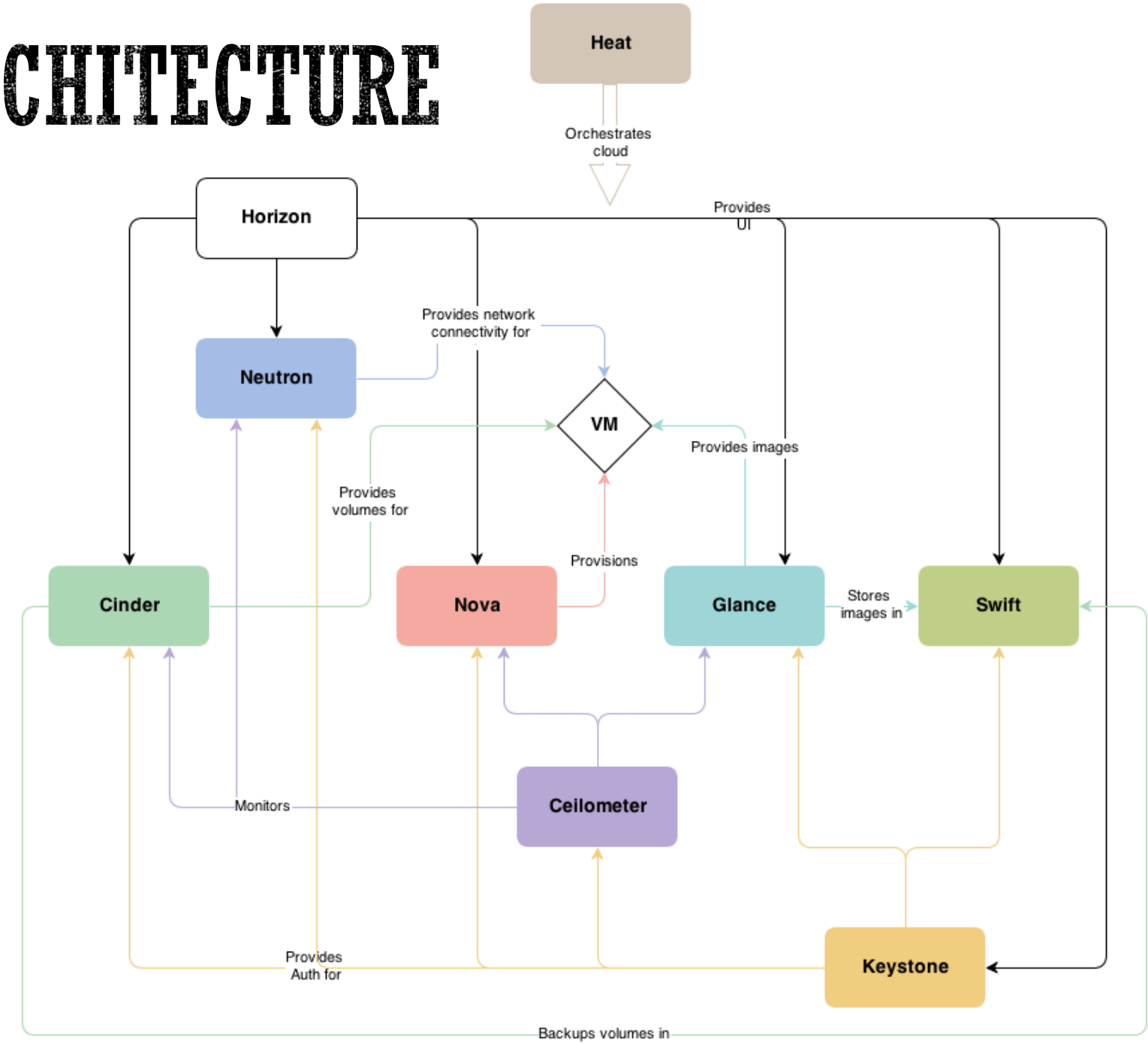


- NFV Overview
- **OpenStack Overview**

OPENSTACK

- OpenStack is a collection of open source components to deliver public and private IaaS clouds
  - Several components such as Nova, Swift, Glance, Keystone, and Neutron
- IaaS Cloud Services allows users to manage VMs, Virtual networks, storage resources

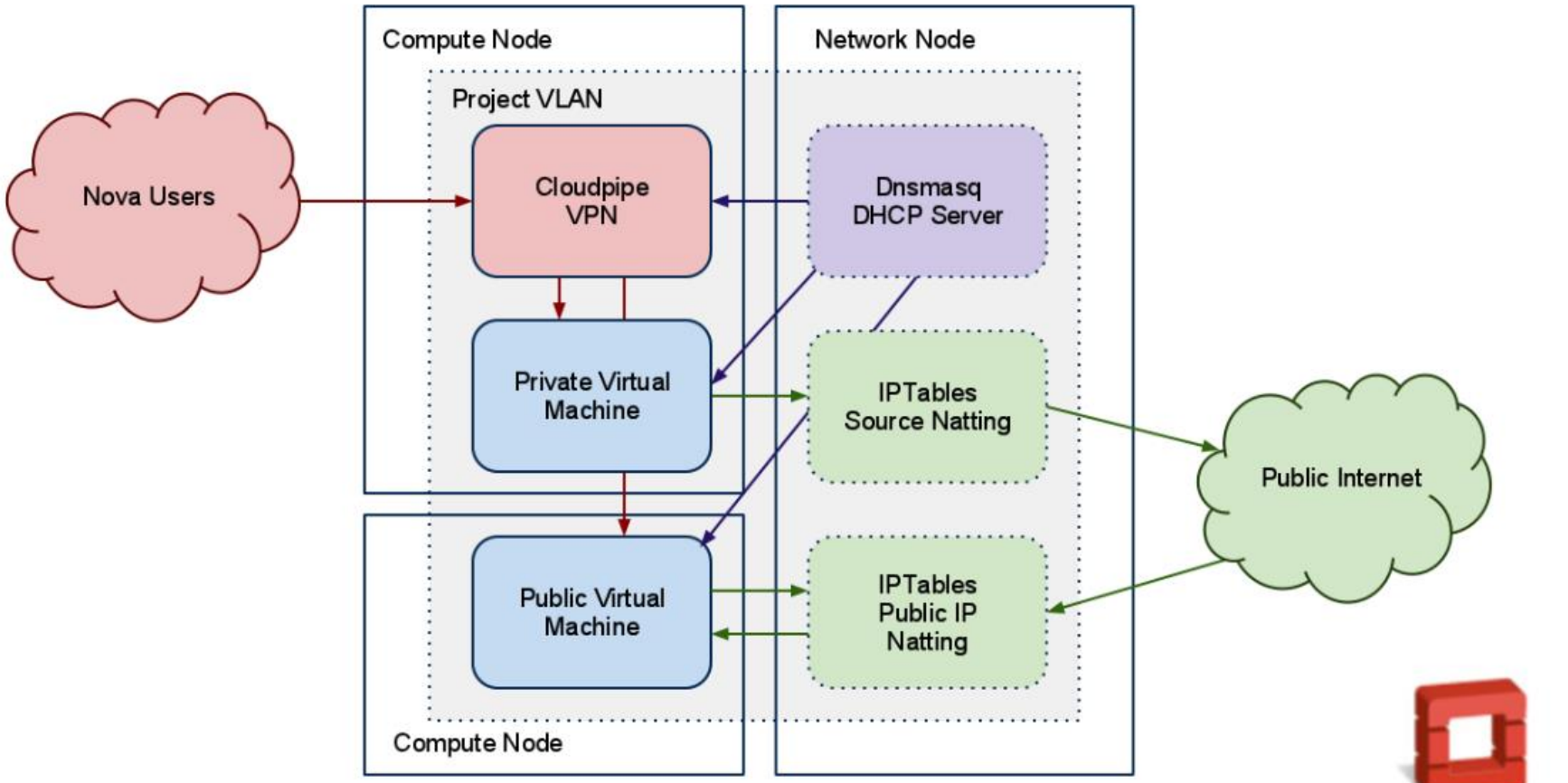
OPENSTACK ARCHITECTURE



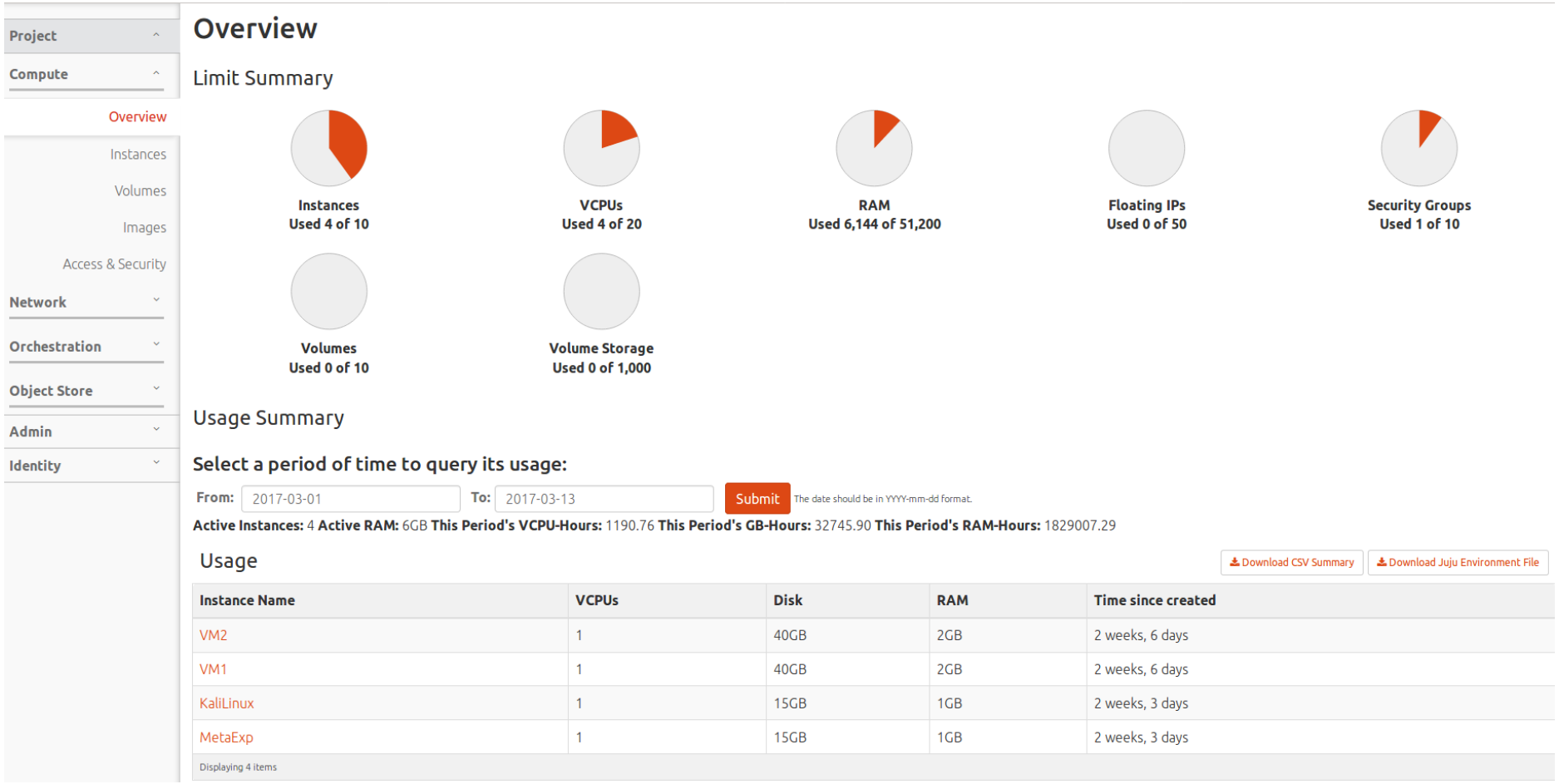
OPENSTACK PROJECTS

Service	Project name	Description
Dashboard	Horizon	Provides a web-based self-service portal to interact with underlying OpenStack services, such as launching an instance, assigning IP addresses and configuring access controls.
Compute	Nova	Manages the lifecycle of compute instances in an OpenStack environment. Responsibilities include spawning, scheduling and decomissioning of virtual machines on demand.
Networking	Neutron	Enables network connectivity as a service for other OpenStack services, such as OpenStack Compute. Provides an API for users to define networks and the attachments into them. Has a pluggable architecture that supports many popular networking vendors and technologies.
Storage		
Object Storage	Swift	Stores and retrieves arbitrary unstructured data objects via a RESTful, HTTP based API. It is highly fault tolerant with its data replication and scale out architecture. Its implementation is not like a file server with mountable directories.
Block Storage	Cinder	Provides persistent block storage to running instances. Its pluggable driver architecture facilitates the creation and management of block storage devices.
Shared services		
Identity service	Keystone	Provides an authentication and authorization service for other OpenStack services. Provides a catalog of endpoints for all OpenStack services.
Image Service	Glance	Stores and retrieves virtual machine disk images. OpenStack Compute makes use of this during instance provisioning.
Telemetry	Ceilometer	Monitors and meters the OpenStack cloud for billing, benchmarking, scalability, and statistical purposes.
Higher-level services		
Orchestration	Heat	Orchestrates multiple composite cloud applications by using either the native HOT template format or the AWS CloudFormation template format, through both an OpenStack-native REST API and a CloudFormation-compatible Query API.
Database Service	Trove	Provides scalable and reliable Cloud Database-as-a-Service functionality for both relational and non-relational database engines.

OPENSTACK ACCESS MODEL

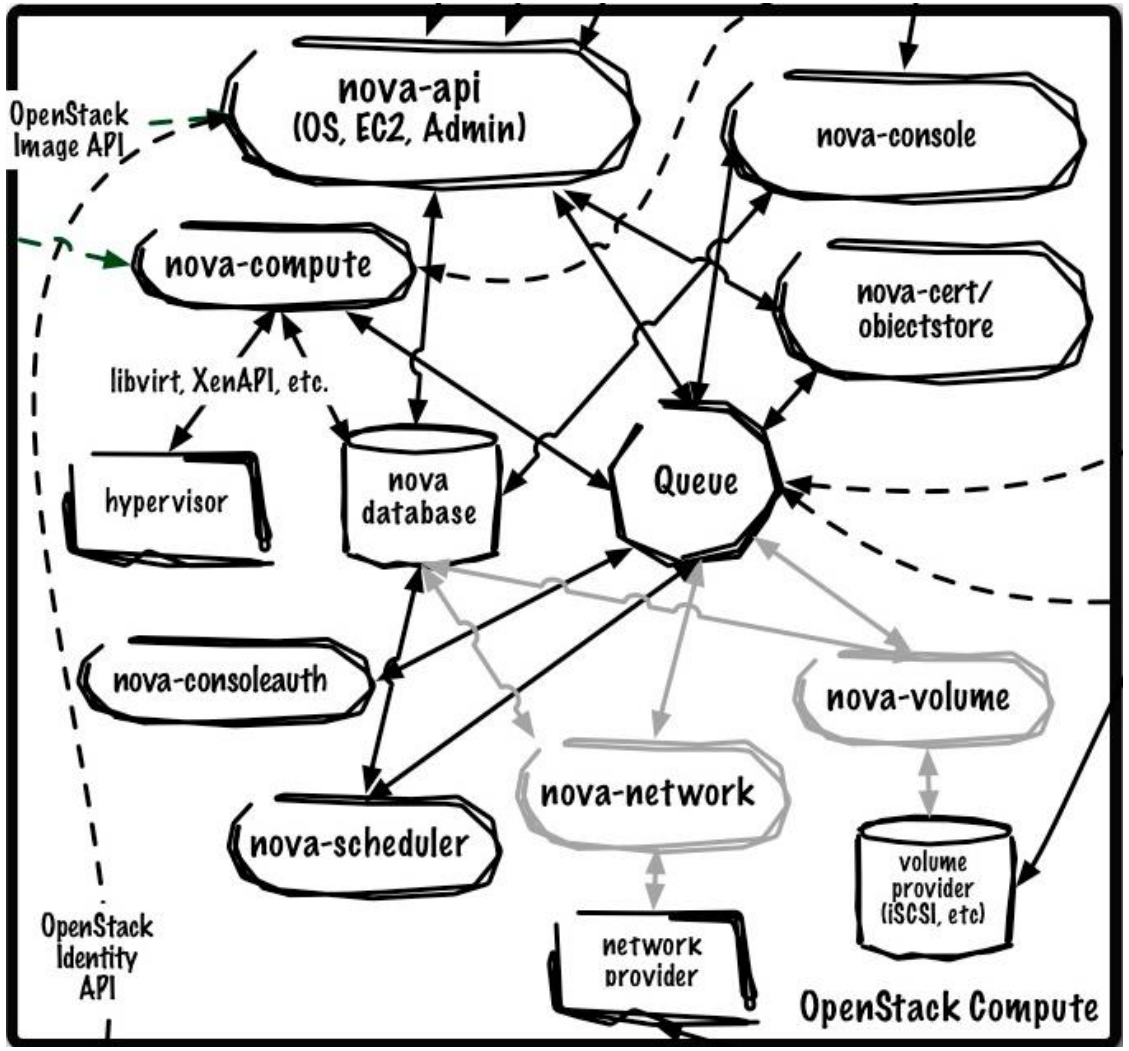


OPENSTACK HORIZON

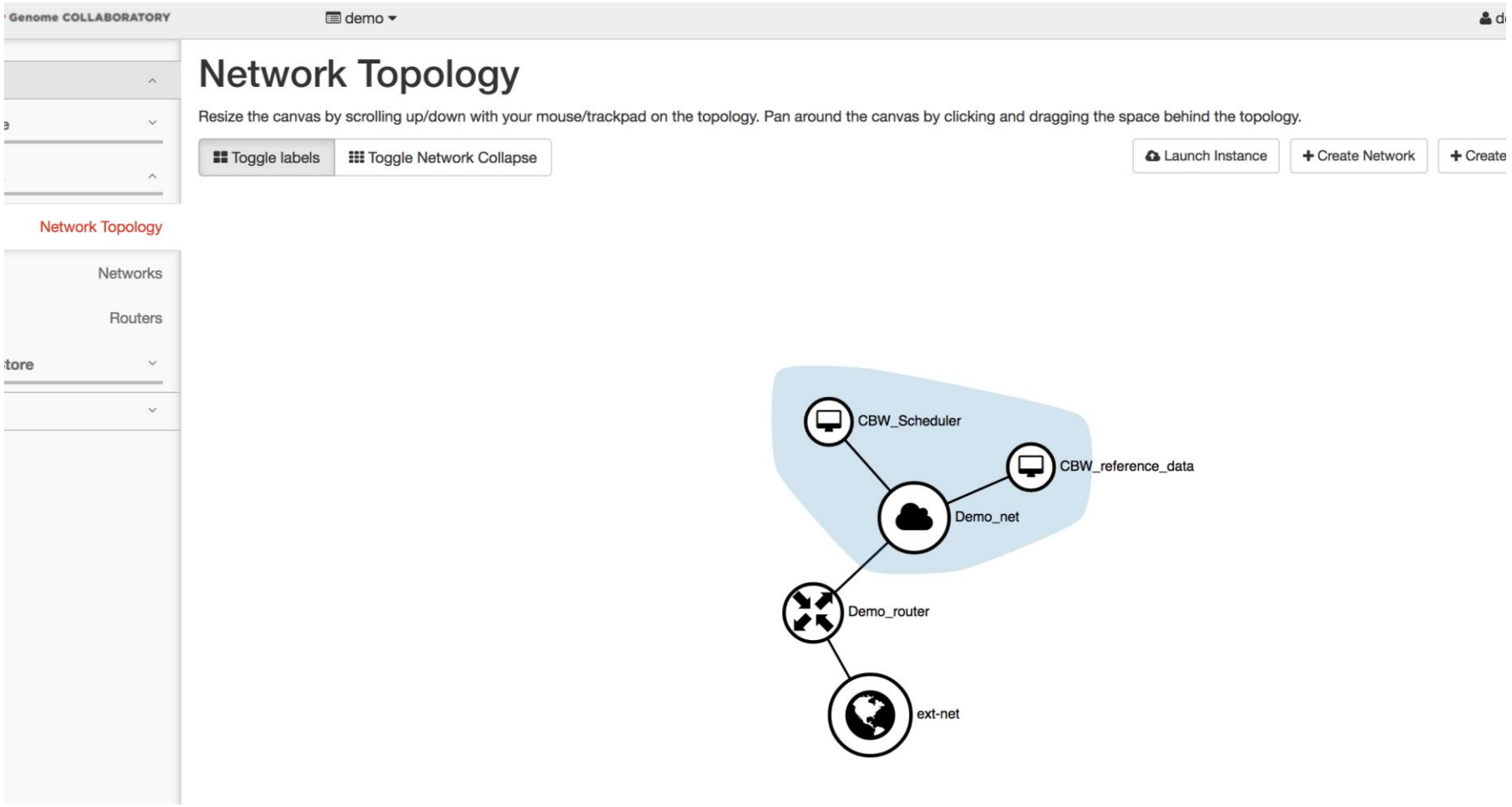


OPENSTACK NOVA

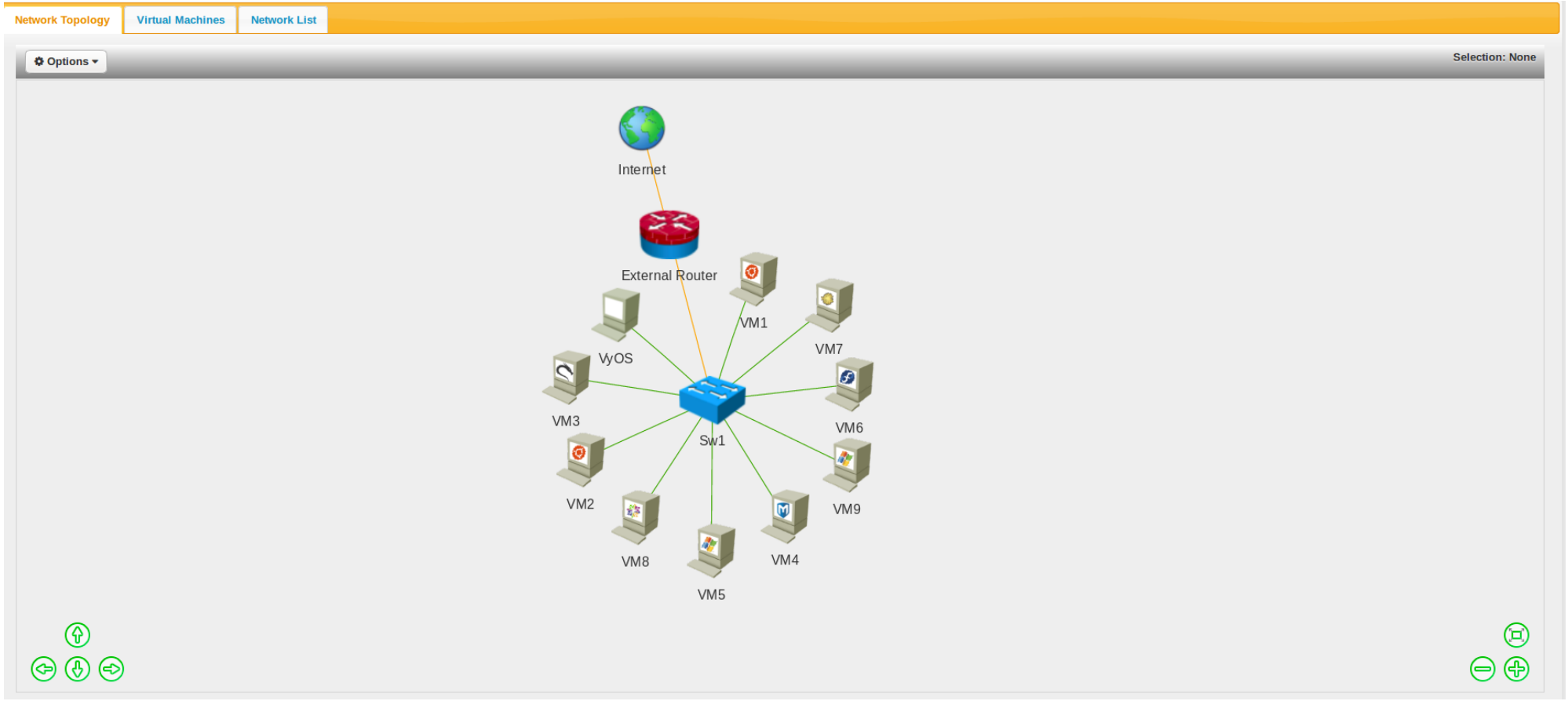
- Nova supports a wide range of virtualization solutions, such as KVM, Hyper-V, Vmware, docker, etc.
- It is a compute controller and manage virtualized server resources
  - CPU/Memory/Disk/Network Interfaces
  - API with rate limiting and authentication
- Distributed and asynchronous architecture
- Massively scalable and highly available system
- Supports live guest migration
  - Move running guests between physical hosts
  - Run, reboot, suspend, resize, terminate instances
- Security controls
  - RBAC



OPENSTACK NEUTRON

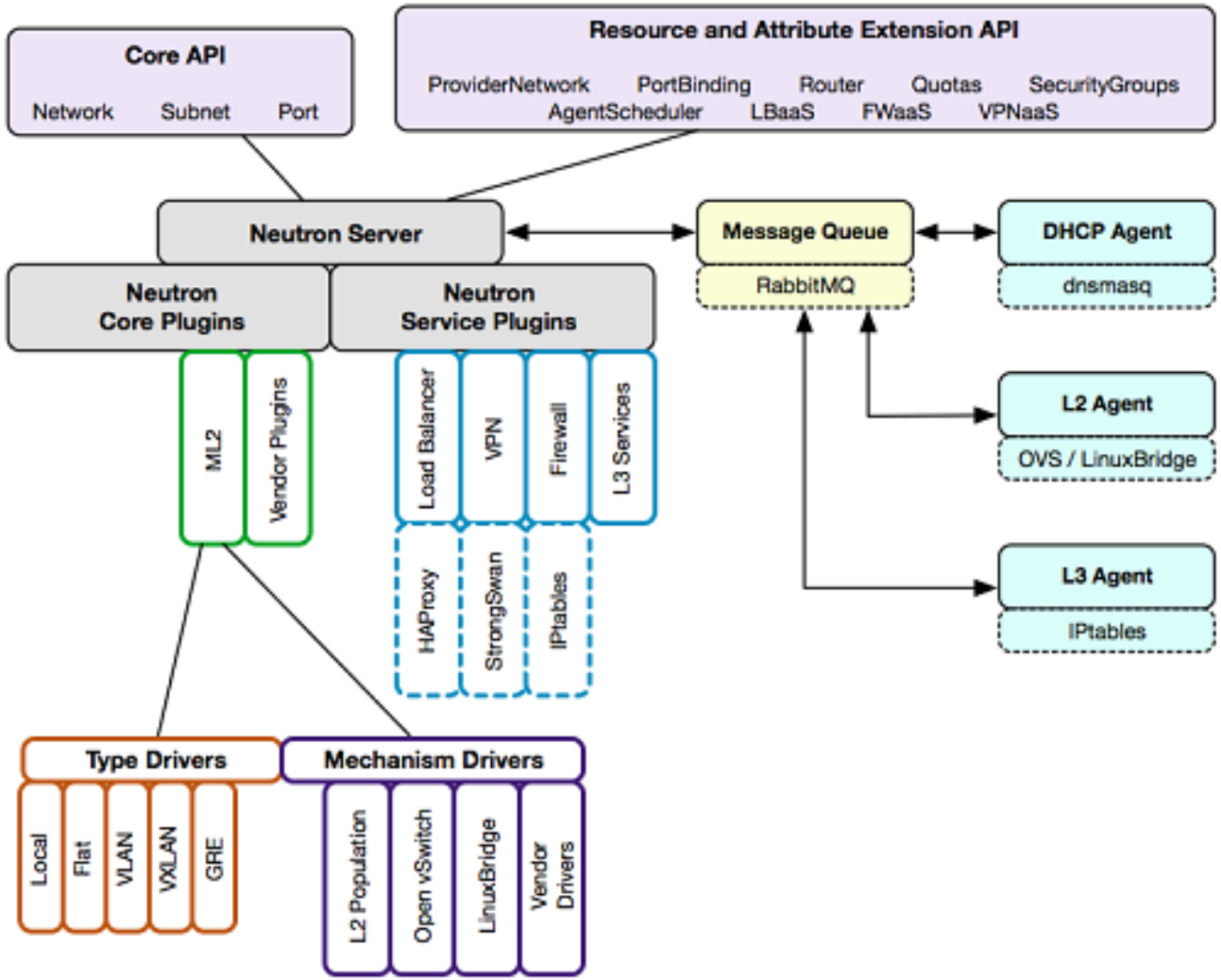


OPENSTACK NEUTRON



ThoTh Lab's virtual view of an Openstack Neutron's private virtual network.

OPENSTACK NEUTRON

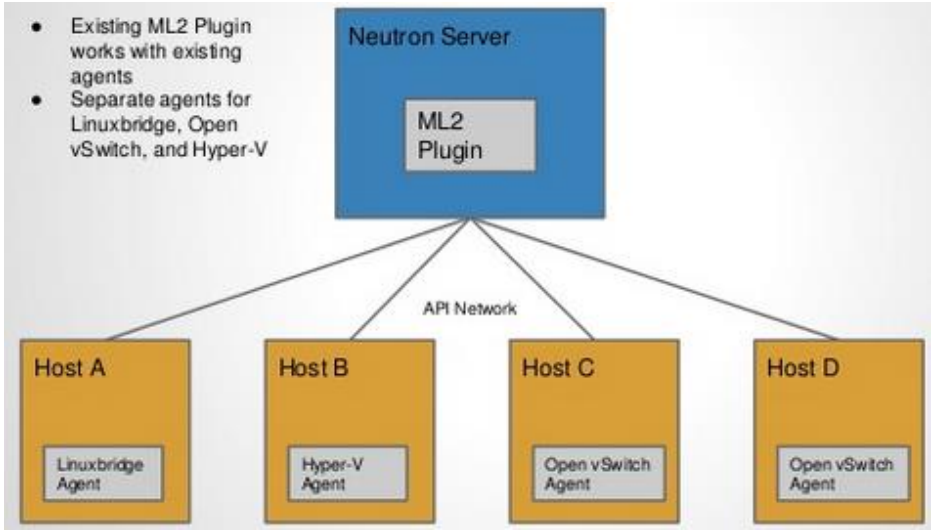


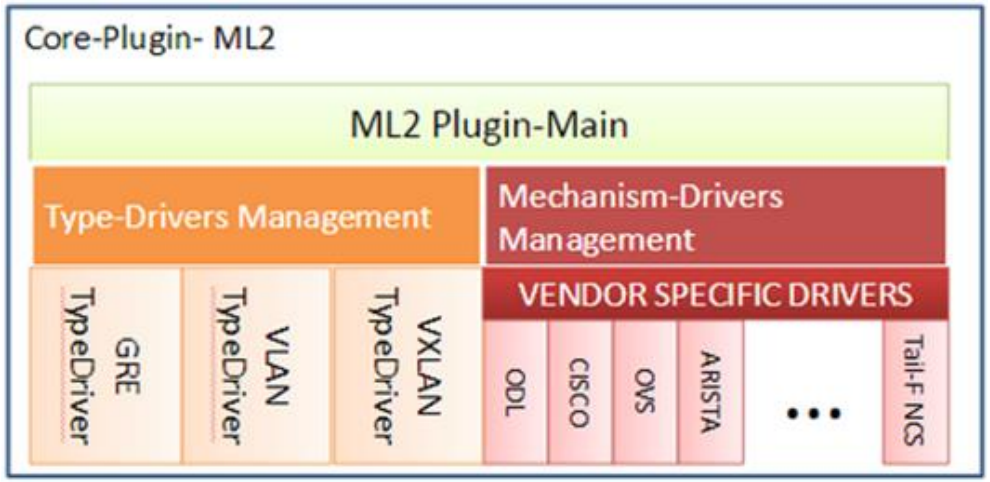
MODULAR LAYER 2

- Neutron core plugin
  - Drivers for a variety of layer 2 technologies
    - Interfacing with agents, hardware and controllers
  - Works with existing L2 agents
    - openvswitch
    - linuxbridge
    - hyperv
  - Deprecates existing monolithic plugins

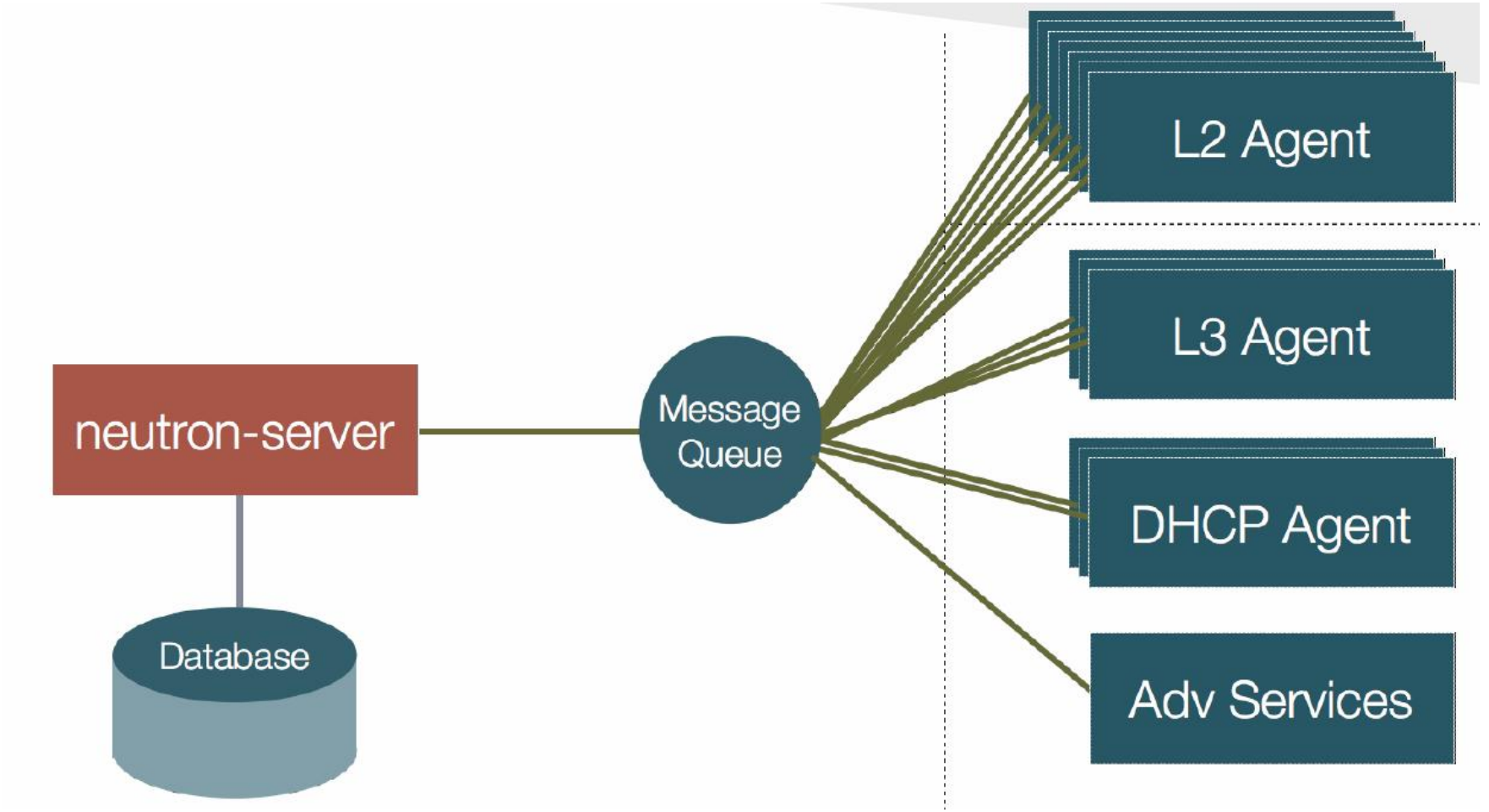
MODULAR LAYER 2

- Neutron core plugin
  - Drivers for a variety of layer 2 technologies
    - Interfacing with agents, hardware and controllers
  - Works with existing L2 agents
    - openvswitch
    - linuxbridge
    - Hyper-v
  - Deprecates existing monolithic plugins





BASIC NETWORK FUNCTION DEPLOYMENT

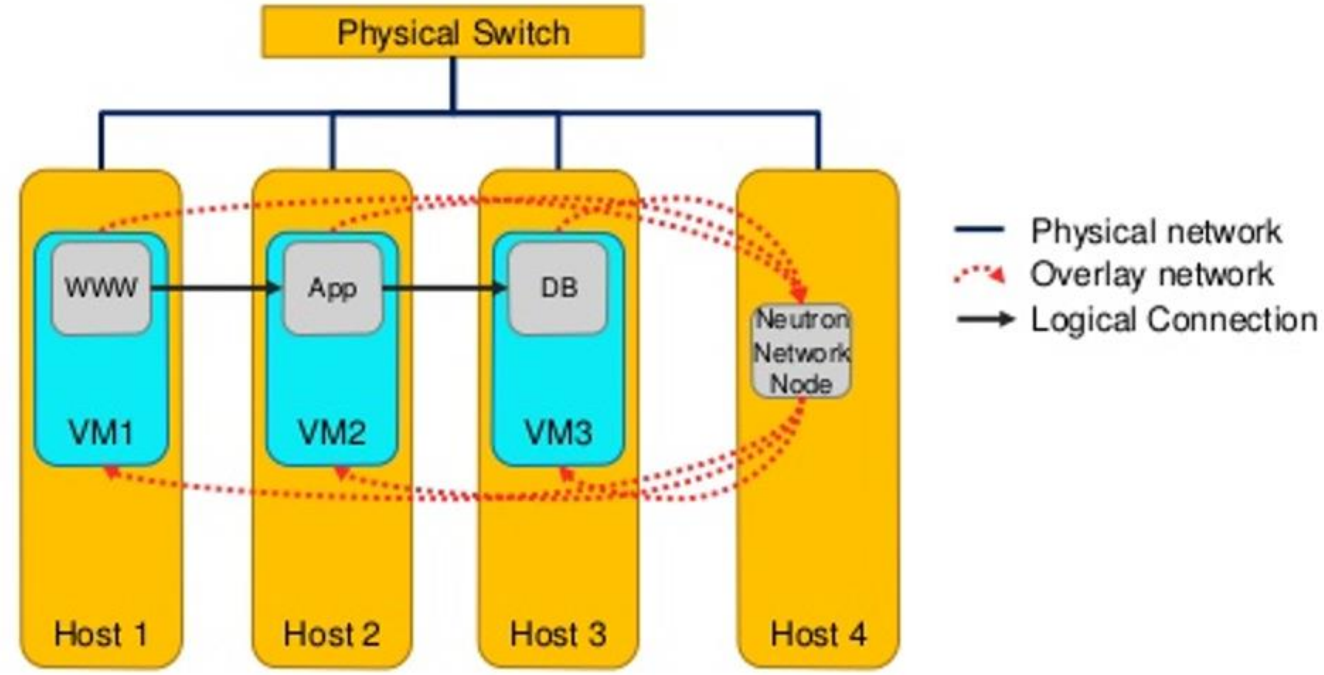


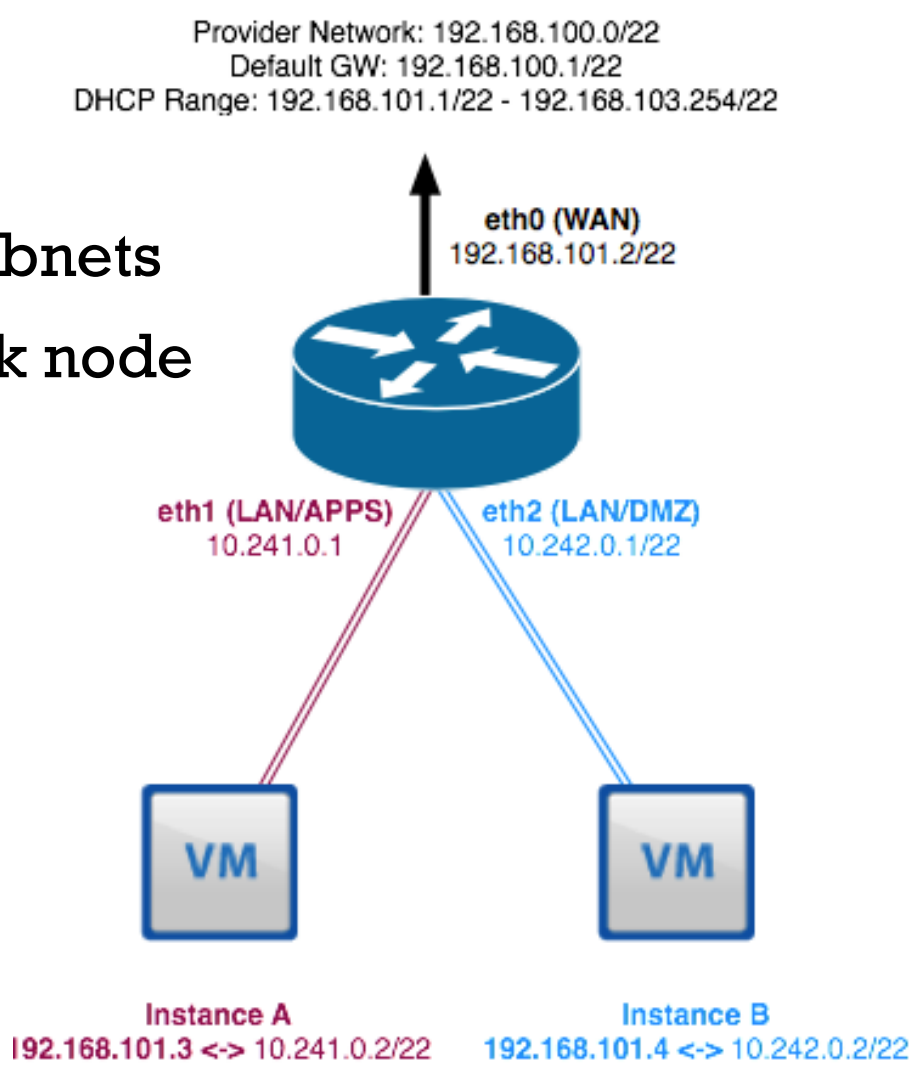
MODULAR LAYER 2

- Functional superset of the monolith plugins
  - Multiple mechanism drivers can access the same network simultaneously
  - Separates management of network types from the mechanisms for accessing those networks
  - RPC interface to L2 agents
  - Extension APIs
    - L3 router extension integrated as a service plugin
- Supports multiple mechanism drivers
  - Arista
  - Cisco
  - Linuxbridge
  - Open vSwitch
  - Hyper-V
- Supports multiple segmentation types
  - VLAN
  - GRE
  - VxLAN

L3 SERVICE

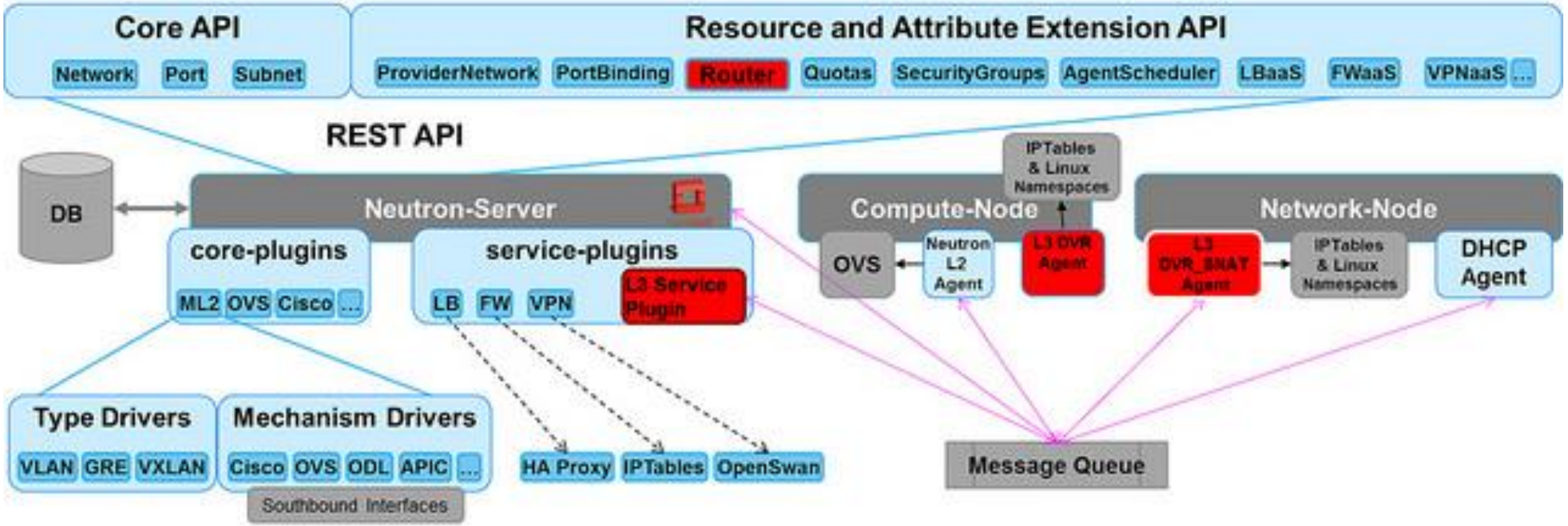
- Most east-west application traffic is between subnets
- All inter-subnet traffic goes through the network node





DISTRIBUTED VIRTUAL ROUTER (VR)

- Distribute L3 services on compute nodes
- Linux namespace is cloned to all compute nodes that participate in a tenant network



DISTRIBUTED VR

- Pros
  - Distributed the East-West traffic and the DNAT floating IP
  - Significant reduction of Network node contention
- Cons
  - Puts unreasonable load on the main message bus (sync all ARPs to all namespaces)
  - Very complex management
  - Performance impact due to added TCP stack

CONCLUSION

For more information, please refer to OpenStack website

https://www.openstack.org/