

RED HAT
SUMMIT

10 YEARS *and counting*
SAN FRANCISCO | APRIL 14-17, 2014

Software Defined Networking (SDN) OpenFlow and OpenStack

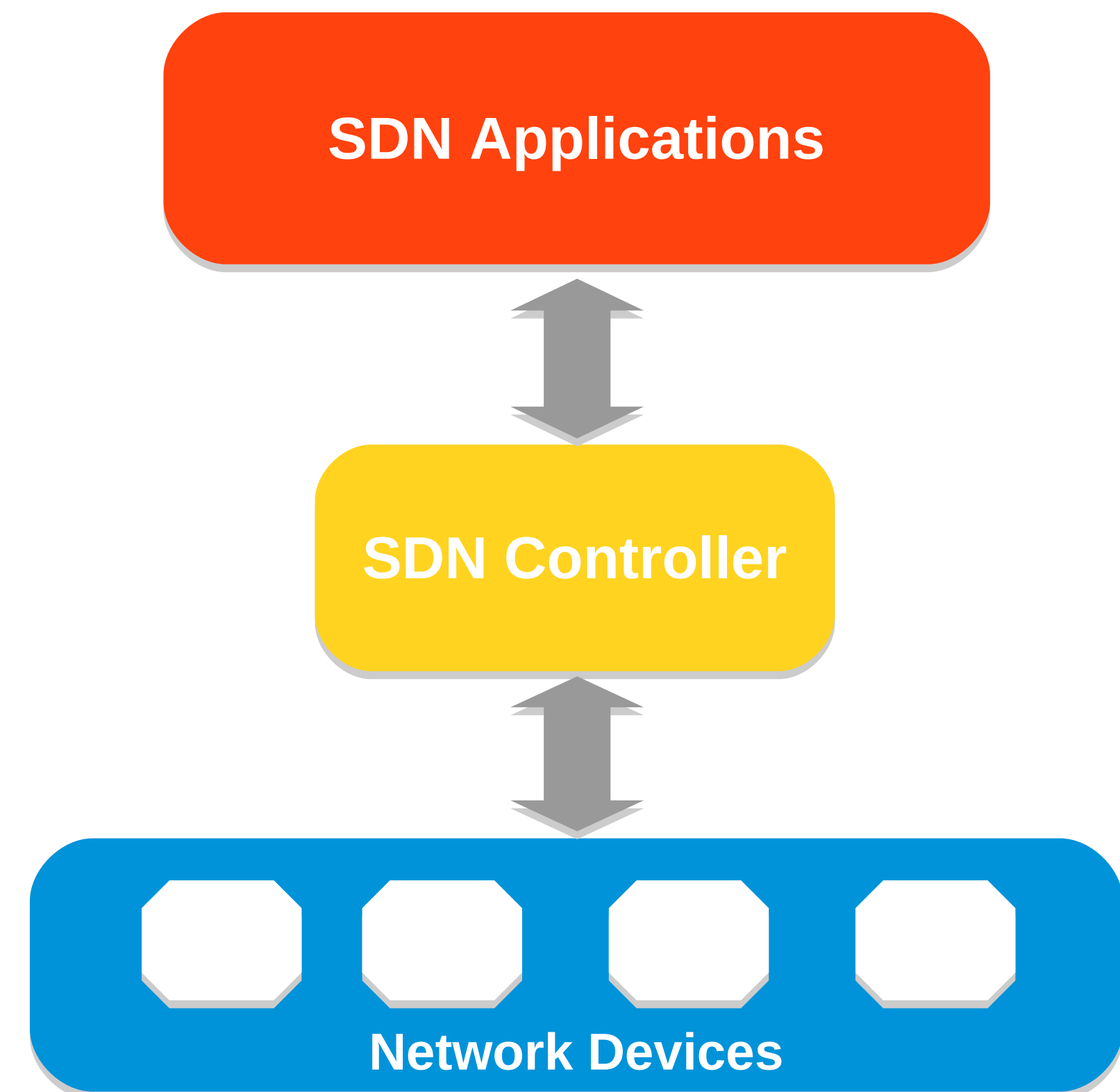
Vivek Dasgupta

Principal Software Maintenance Engineer

Red Hat

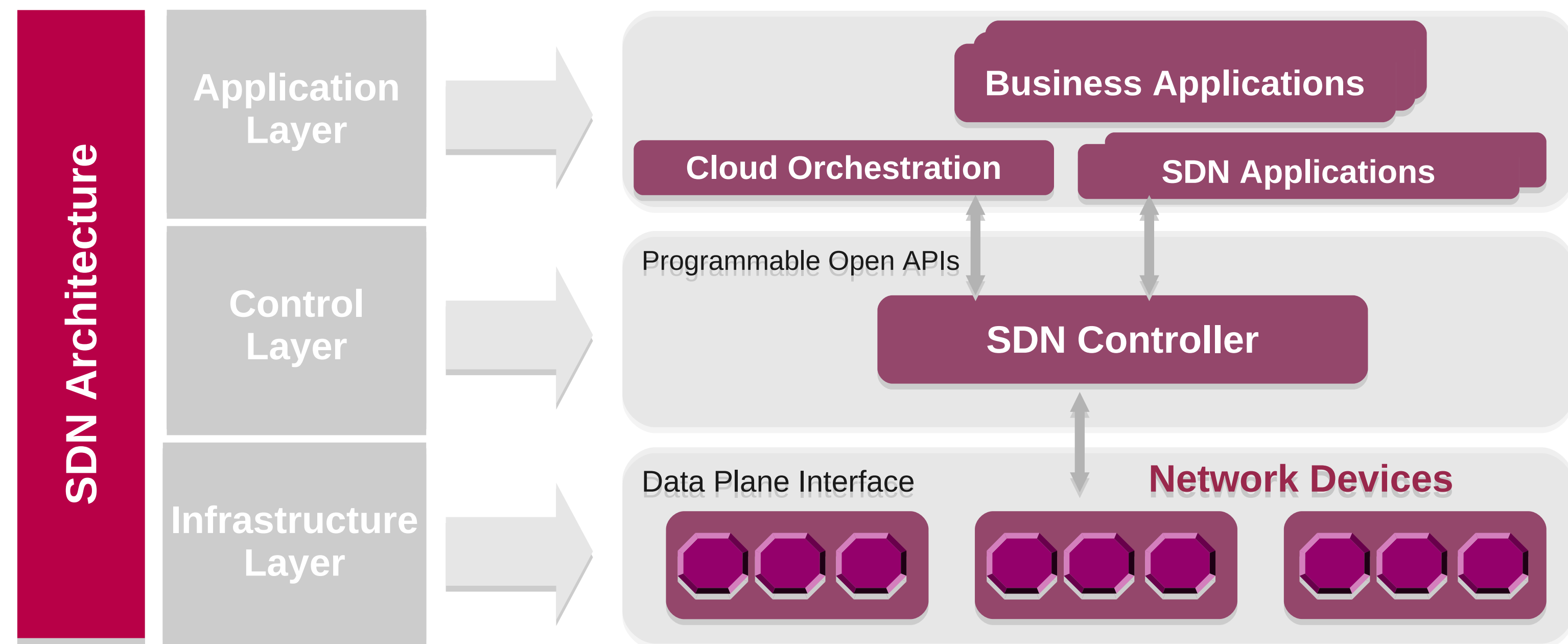
CONTENTS

- Introduction – SDN and components
- SDN Architecture, Components
- SDN Controller - OpenDayLight
- OpenFlow architecture
- Open vSwitch
- OpenStack Neutron
- OpenStack SDN
- Future Trends



Introduction – SDN and related technologies

- SDN is a technology enabling **programmable networks**
- Using software running on general purpose OS/Hardware
- SDN – Separation of control and data plane





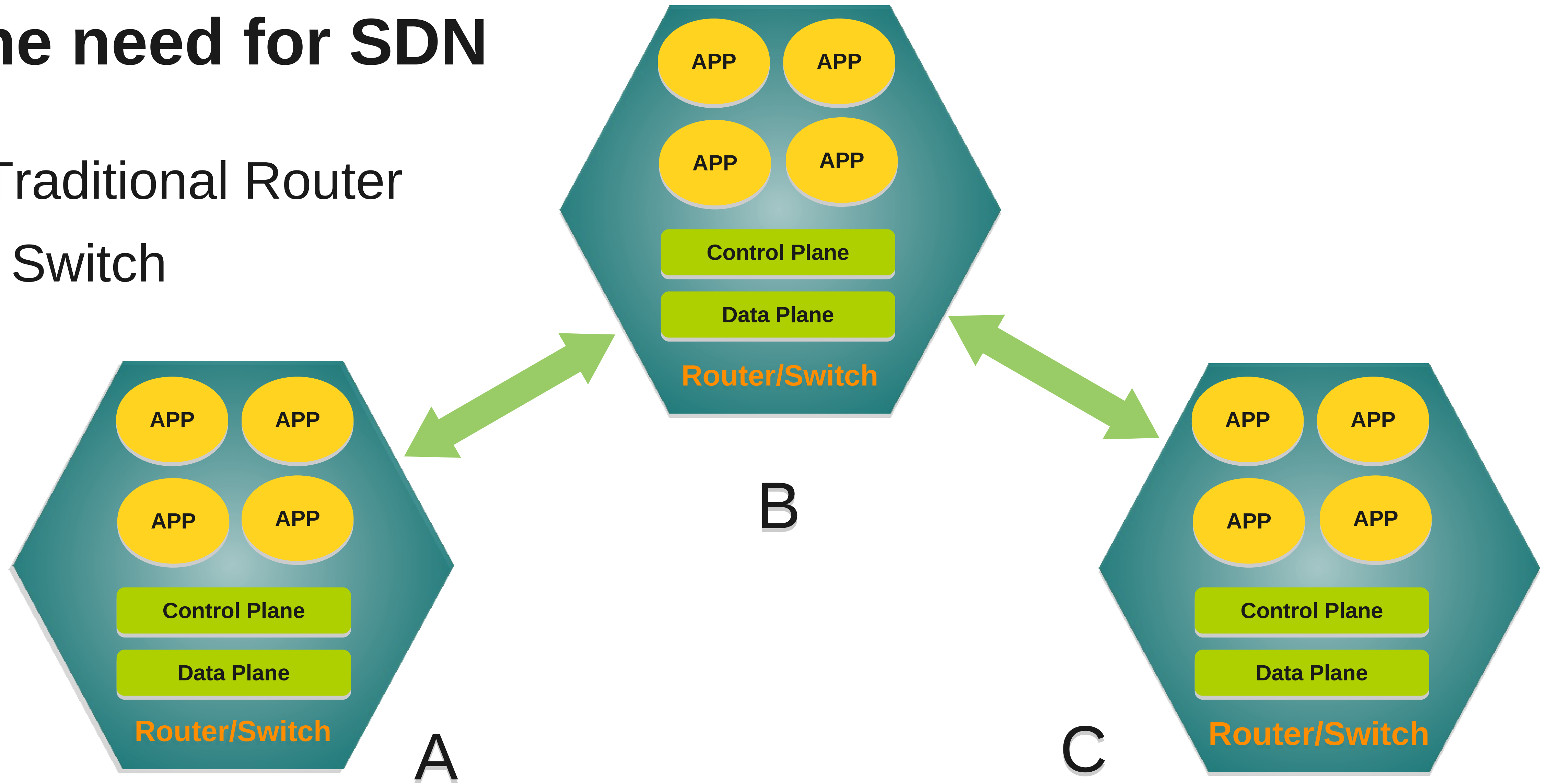
R 22L-4R

NO TRESPASSING
KEEP BACK 100'

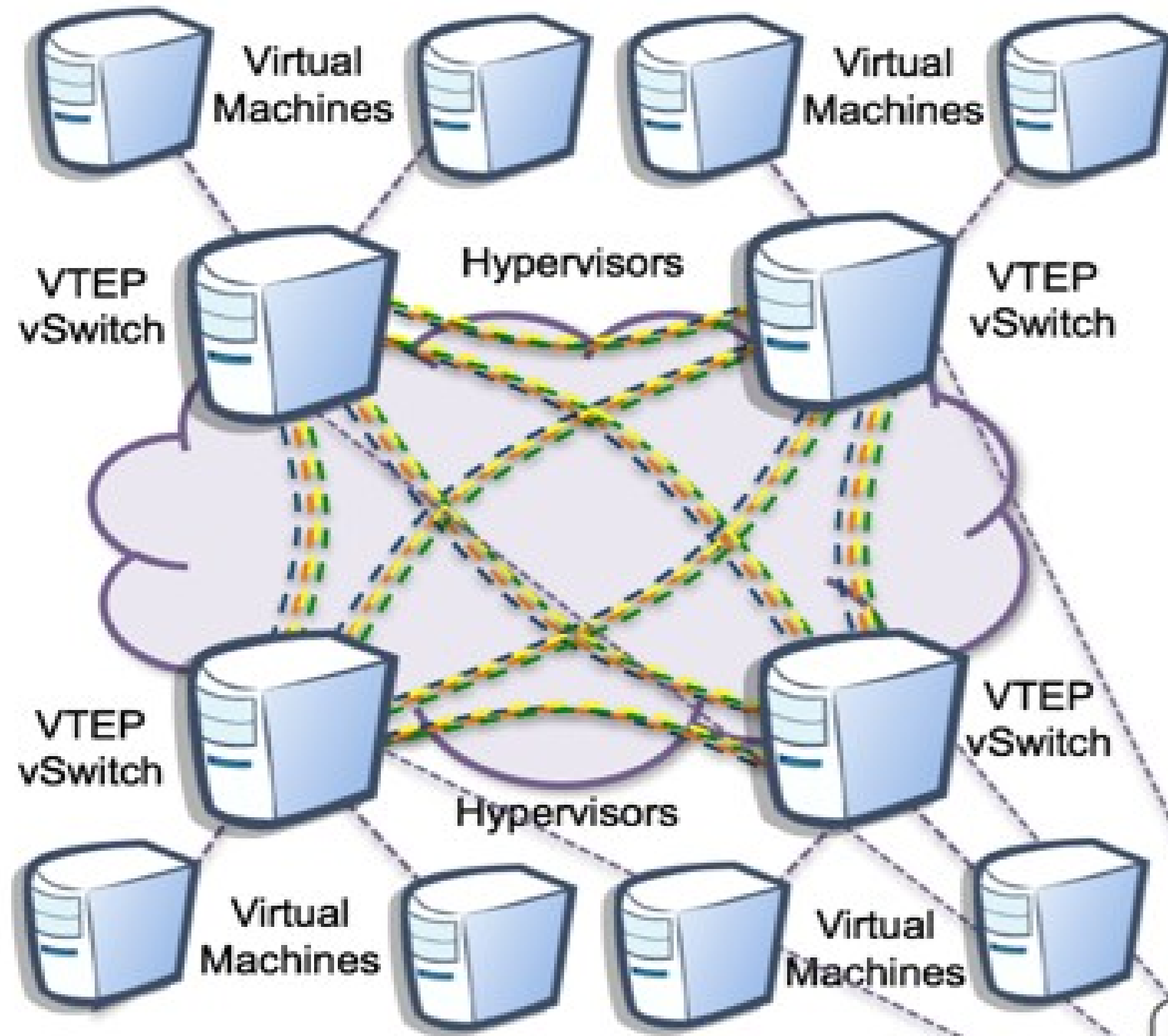
ILS

The need for SDN

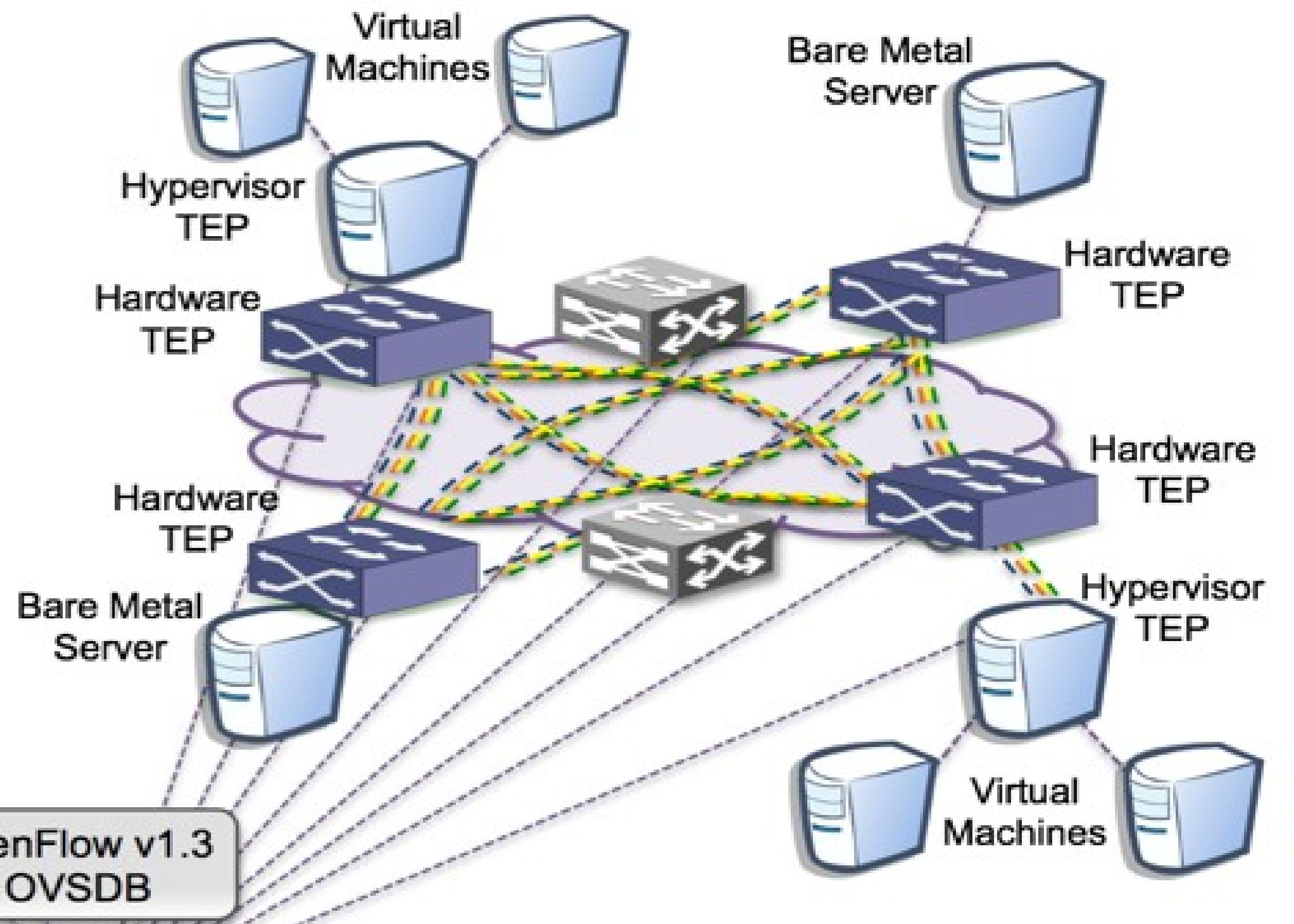
- Traditional Router / Switch



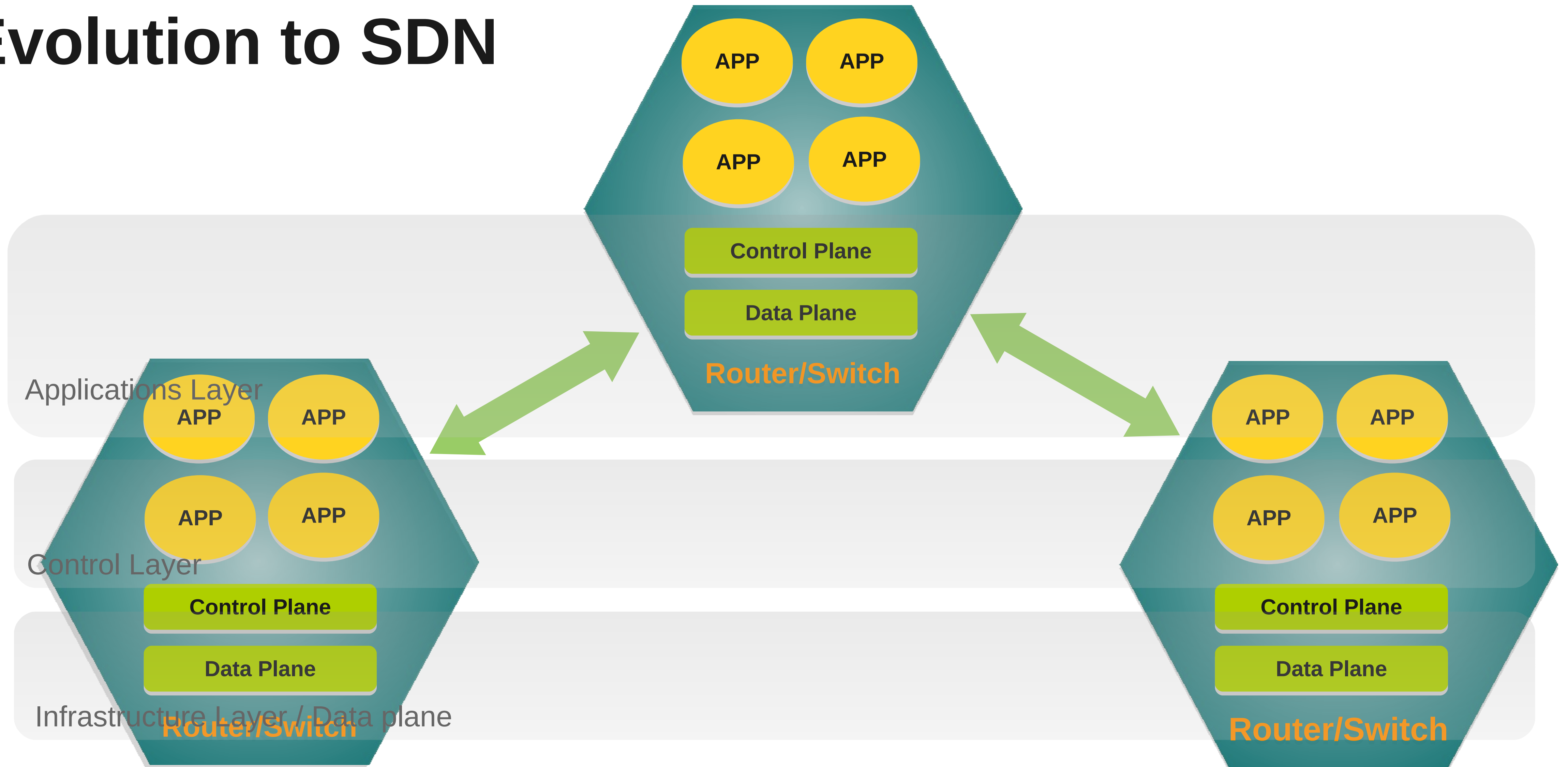
VTEP Edge Appears Directly Connected



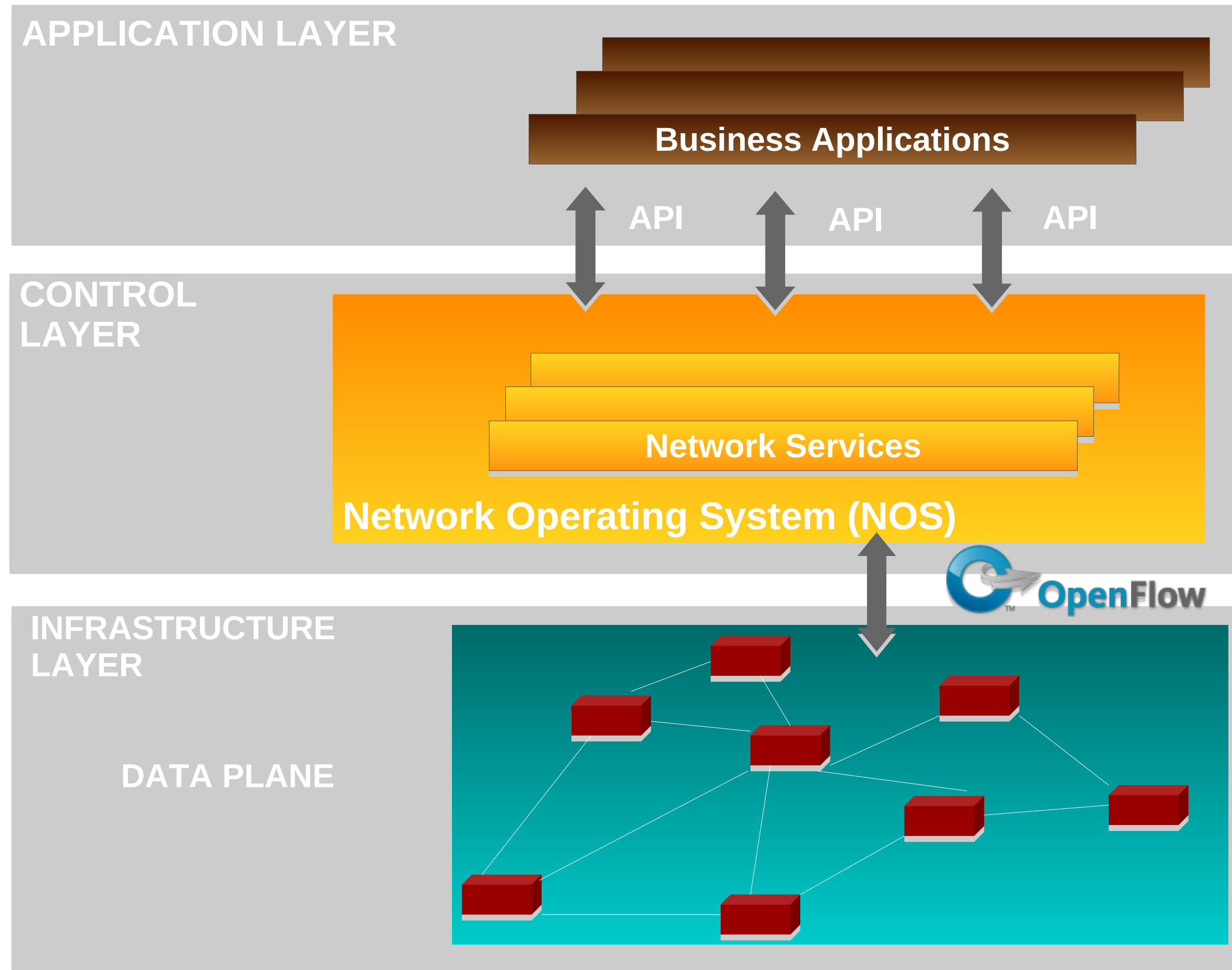
Evolving Hardware Opens New Opportunities



Evolution to SDN



SDN Architecture



- **SDN Applications**
- **SDN Control Plane**
Controller
NOS
- **SDN Data Plane**
Devices
OpenFlow





SDN Components - (Ecosystem)

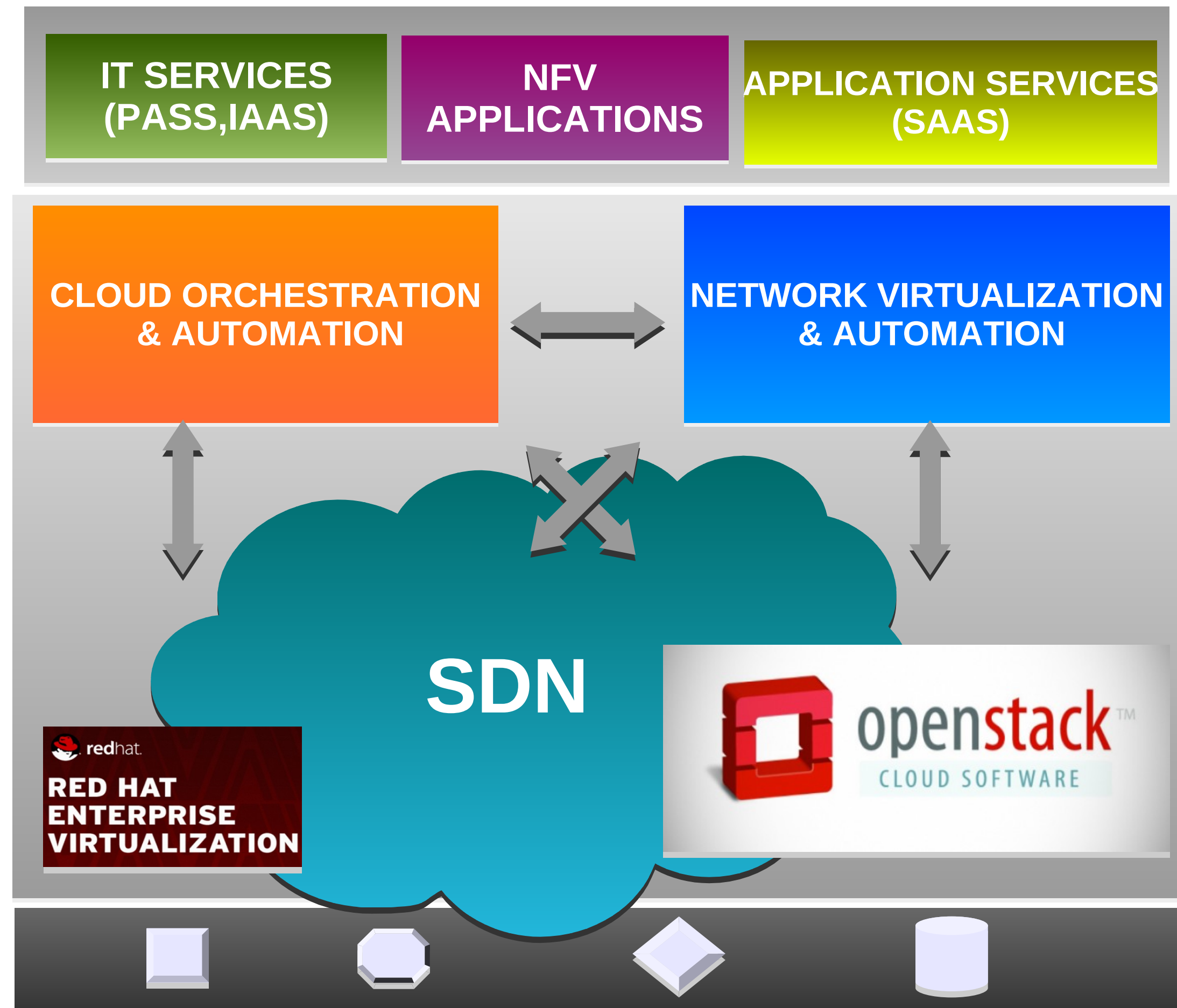
- Cloud Orchestration
- Network Virtualization
- Network Functions Virtualization (NFV)

CLOUD SERVICES AND CAPABILITY

CLOUD INTELLIGENCE AND CONTROL

OPEN DISTRIBUTED CLOUD INFRA-STRUCTURE

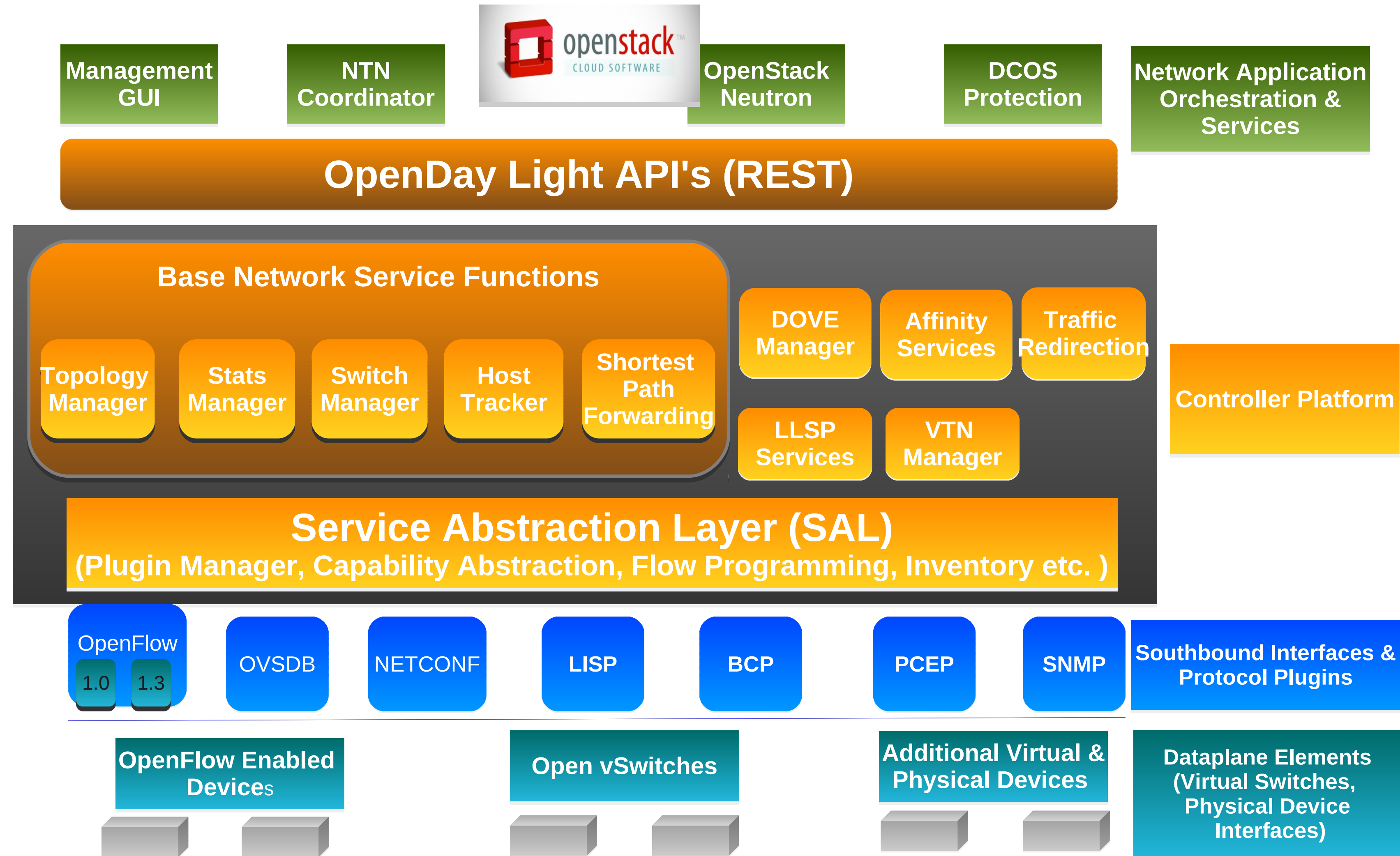
CUSTOMERS



SDN Controller OpenDayLight



- Open Source
- Southbound - OF
- Northbound -> Neutron
- Plugin -> Red Hat Enterprise Linux OpenStack Platform



RED HAT
SUMMIT

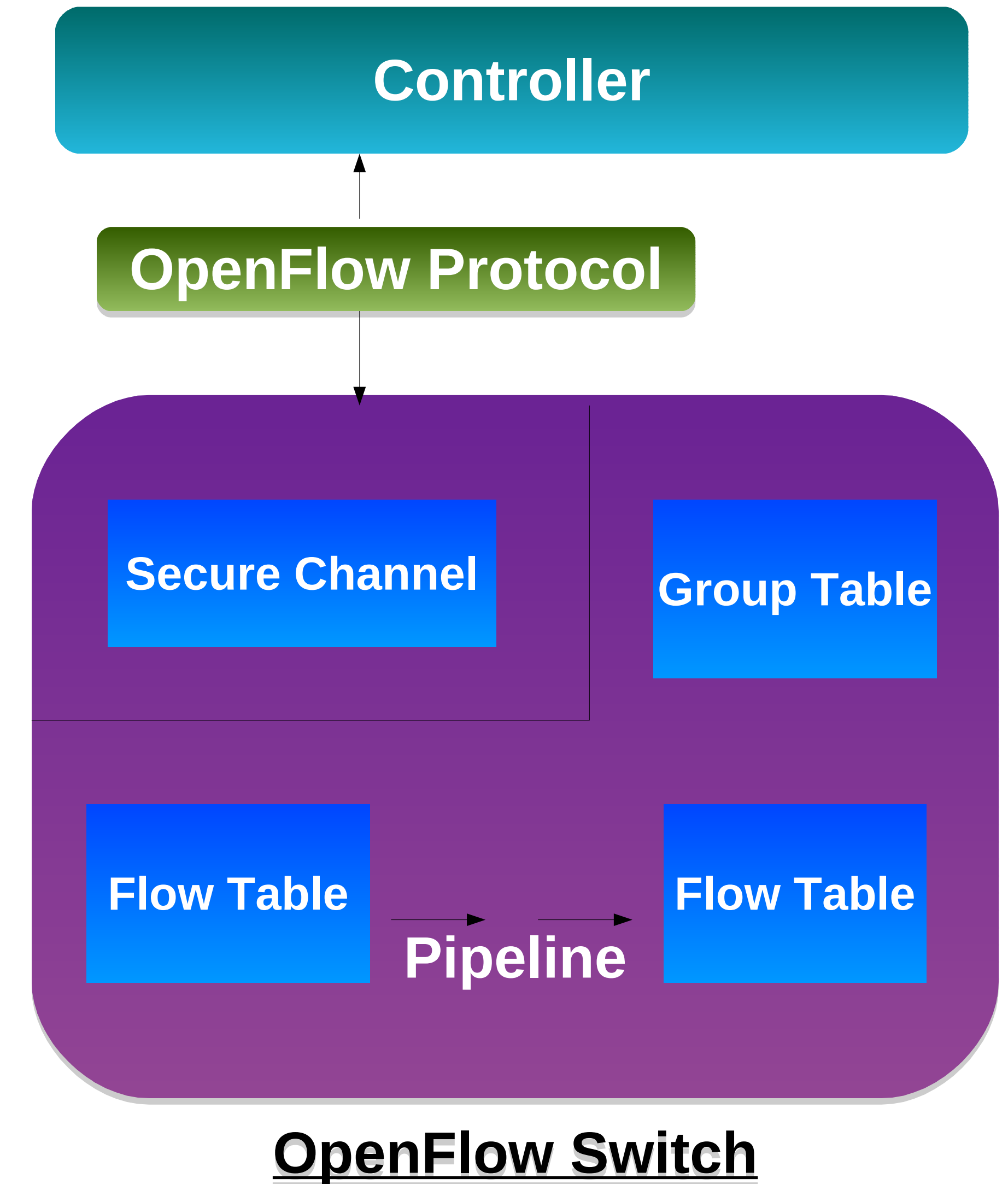
10 YEARS *and counting*

SAN FRANCISCO | APRIL 14-17, 2014

OpenFlow / Open vSwitch

OpenFlow introduction

- **Openflow** - standard for interacting with forwarding behaviours of switches
- Control the behaviour of switches dynamically and programmatically
- **Flow tables**, Group tables and OpenFlow Channel (TCP port 6633)



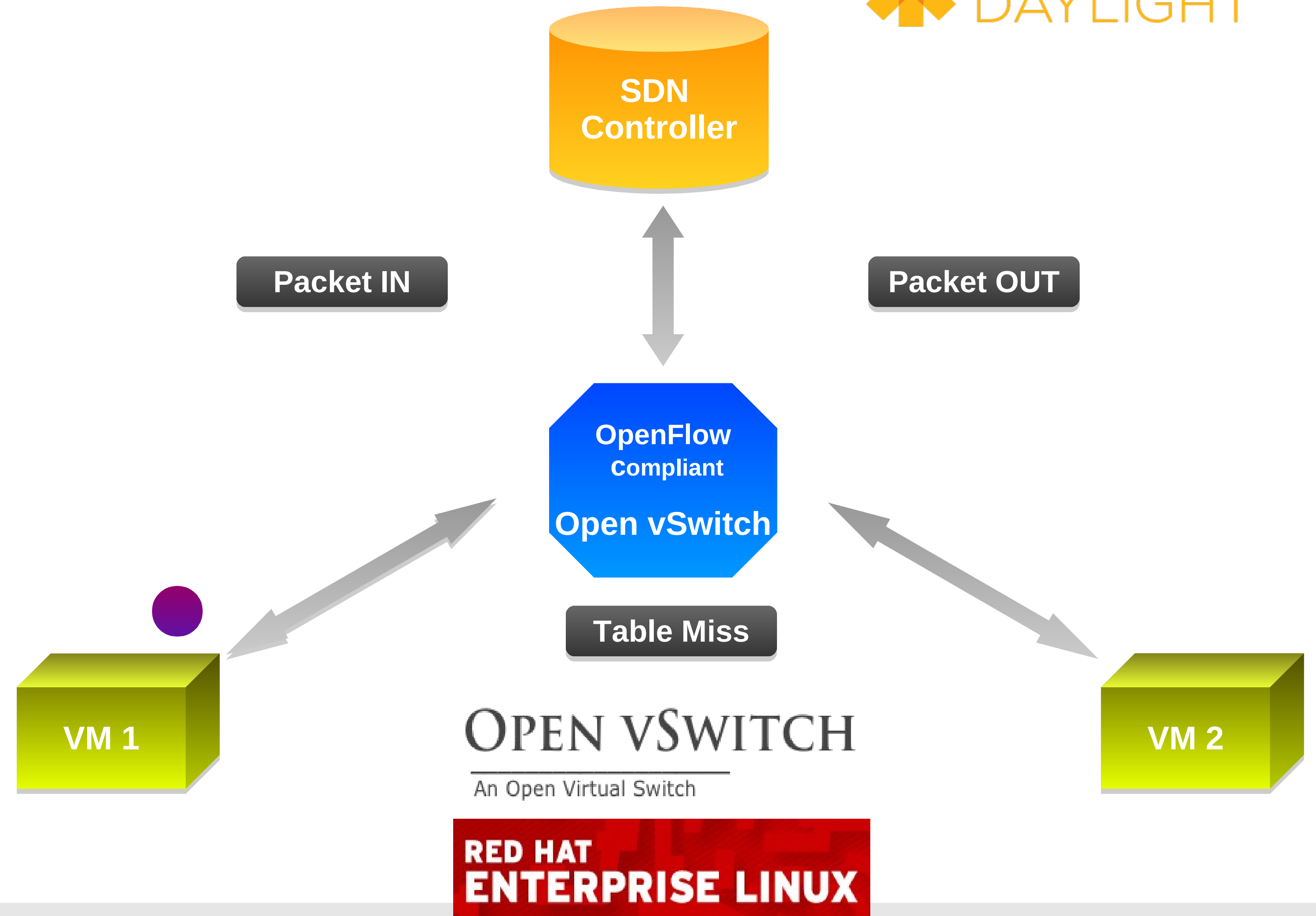
OpenFlow protocol - Messages

- **Controller to Switch** :: Switch / Flow table config, *Packet out*, Barrier, Role Req, Bundle [Controller to Switch messages]
- **Asynchronous** :: *Packet-in*, Flow-removed, Port-status, Controller Role status, Table status, Request forward [Async messages]
- **Symmetric messages** :: Hello, Echo Req/Reply, Error, Experimenter [Symmetric messages]

OpenFlow – FlowTables and Routing



- **Packet-in** message for table miss
- Controller sends a **packet-out** message specifying action
- **Buffer id** -> packet
- **Flow** modification



OpenFlow – FlowTables and Routing

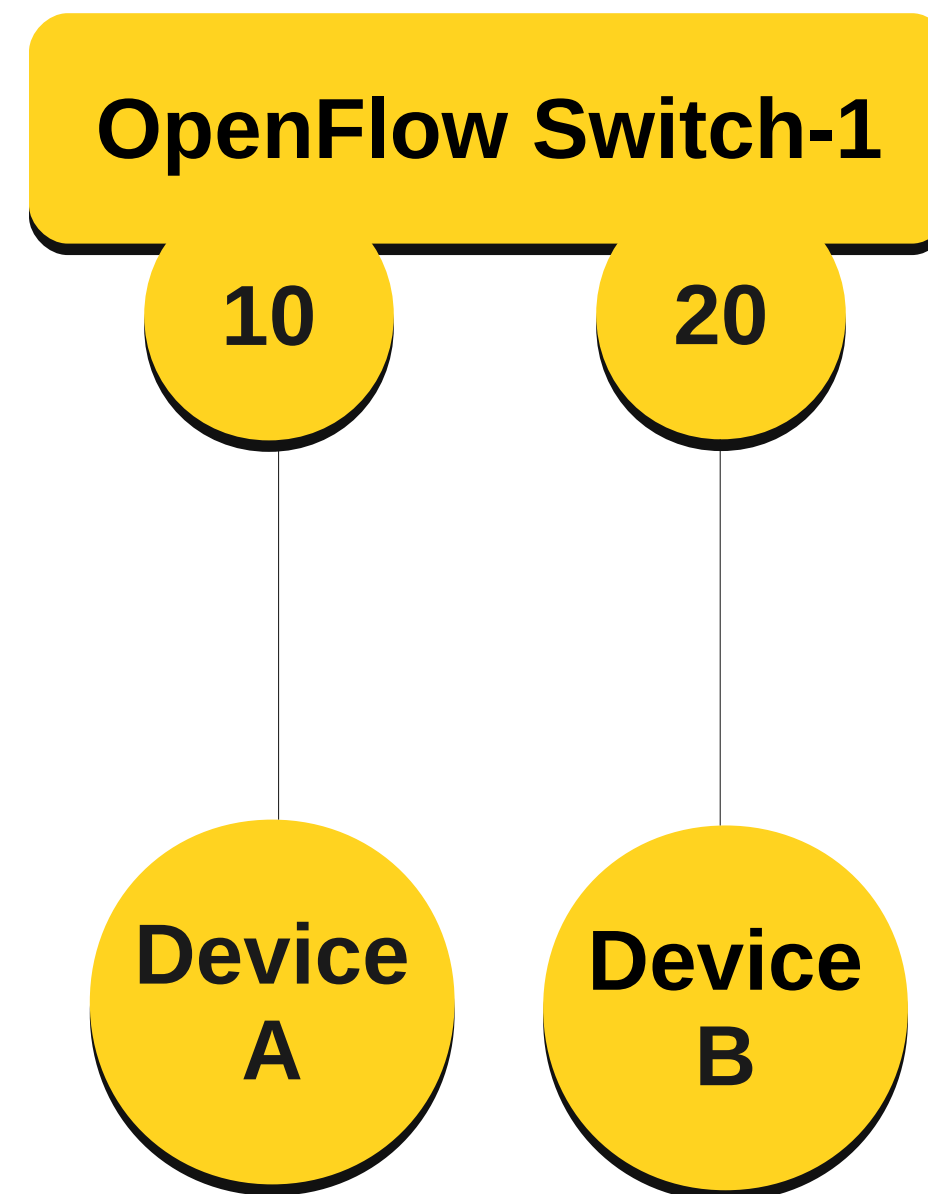
- **Flow Table Components**

Match fields, Priority, Counters, Instructions, Timeouts, Cookie

- Flow table

match vs miss

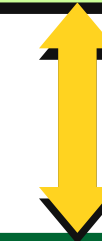
- Flow removal



Match	Action
Src. Port= 10	Fwd to Port 20
Src. Port= 20	Fwd to Port 10

OpenFlow – FlowTables

SDN Controller Software



- Group table
- Meter table
- Counters
- Instructions
- Actions

OpenFlow-enabled Network Device FlowTable compared to an instruction set

MAC Src	MAC dst	IP Src	IP dst	TCP dport	Action	Count
.	10.20	Port 1	250
.	.	.	5.6.7.8	.	.	Port 2	300
.	.	.	.	25	.	drop	892
.	.	.	192.	.	.	local	120
.	controller	11

- OpenFlow Packet Capture

The screenshot shows a Wireshark capture of an OpenFlow Packet Out. The packet list pane shows several packets, with packet 2311 selected. The packet details pane shows the following structure:

- Ethernet II, Src: Apple_94:e0:82 (10:40:f3:94:e0:82), Dst: Hewlett-_b1:ff:c0 (00:9c:02:b1:ff:c0)
- Internet Protocol Version 4, Src: 192.168.1.140 (192.168.1.140), Dst: 192.168.1.210 (192.168.1.210)
- Transmission Control Protocol, Src Port: 6633 (6633), Dst Port: 64247 (64247), Seq: 211, Ack: 389, Len: 66
- OpenFlow Protocol
 - Header
 - Packet Out
 - Buffer ID: None
 - Frame Recv Port: None (not associated with a physical port)
 - Size of action array in bytes: 8
 - Output Action(s)
 - Frame Data: ffffffff1040f394e082080600010800060400011040...
 - Ethernet II, Src: Apple_94:e0:82 (10:40:f3:94:e0:82), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
 - Address Resolution Protocol (request)
 - Hardware type: Ethernet (1)
 - Protocol type: IP (0x0800)
 - Hardware size: 6
 - Protocol size: 4
 - Opcode: request (1)
 - [Is gratuitous: False]
 - Sender MAC address: Apple_94:e0:82 (10:40:f3:94:e0:82)
 - Sender IP address: 10.1.1.1 (10.1.1.1)

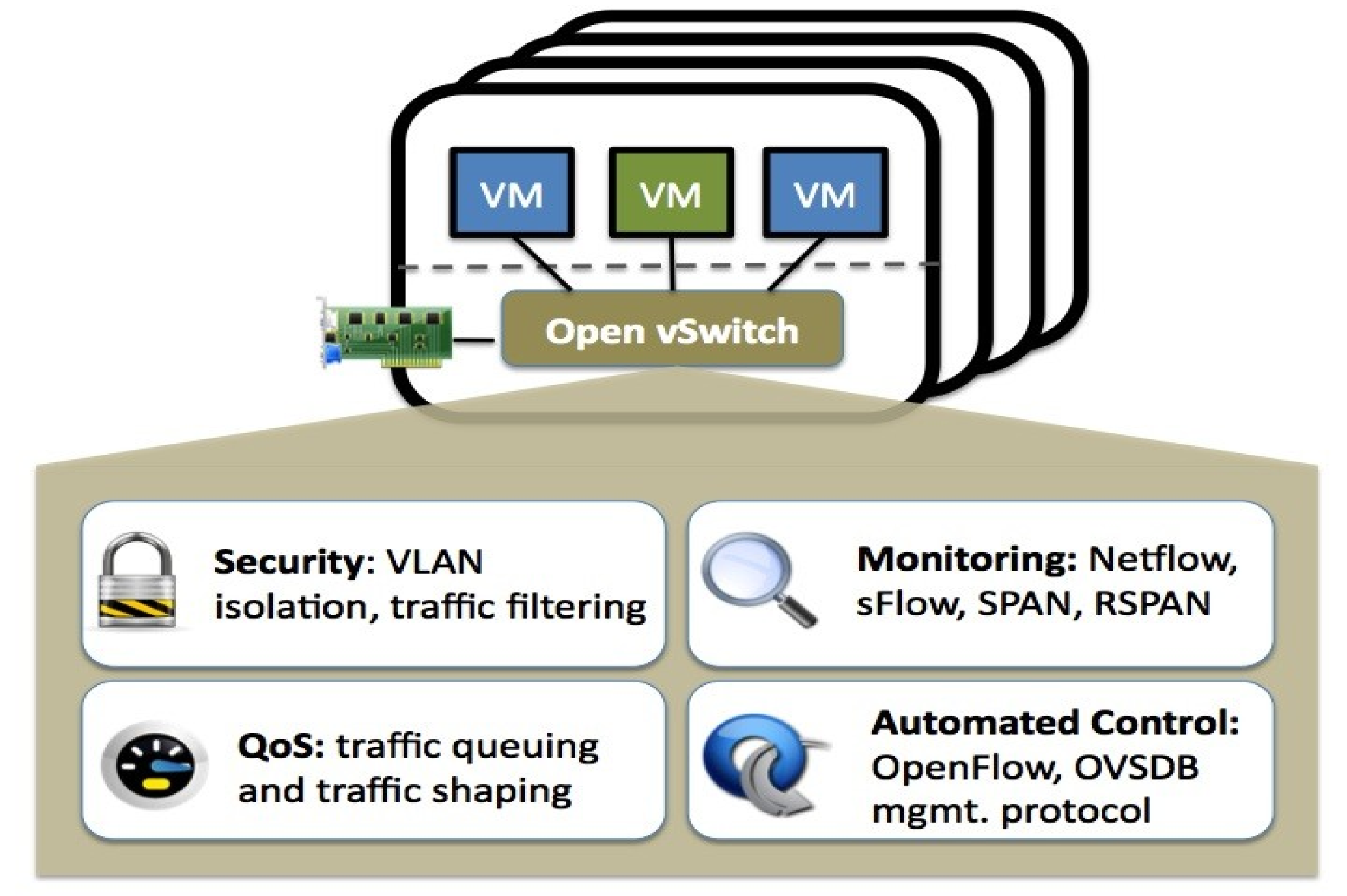
The packet bytes pane at the bottom shows the raw data in hexadecimal and ASCII:

```

0000  00 9c 02 b1 ff c0 10 40 f3 94 e0 82 08 00 45 00  .....@.....E.
0010  00 76 b4 0d 40 00 40 06 01 c6 c0 a8 01 8c c0 a8  .v..@.@.....
0020  01 d2 19 e9 fa f7 40 d8 ed 6e c8 9f 2c 7b 80 18  .....@..n...{.
0030  80 00 78 41 00 00 01 01 08 0a 4e 99 ce 58 04 12  ..xA.....N..X..
  
```

Open vSwitch

- Networking in Software
- In 2012 – total # of virtual ports surpassed physical ports
- A opensource software switch
- High performance forwarding using **Linux Kernel Module**
- OpenFlow Compliant
- Advanced switching features

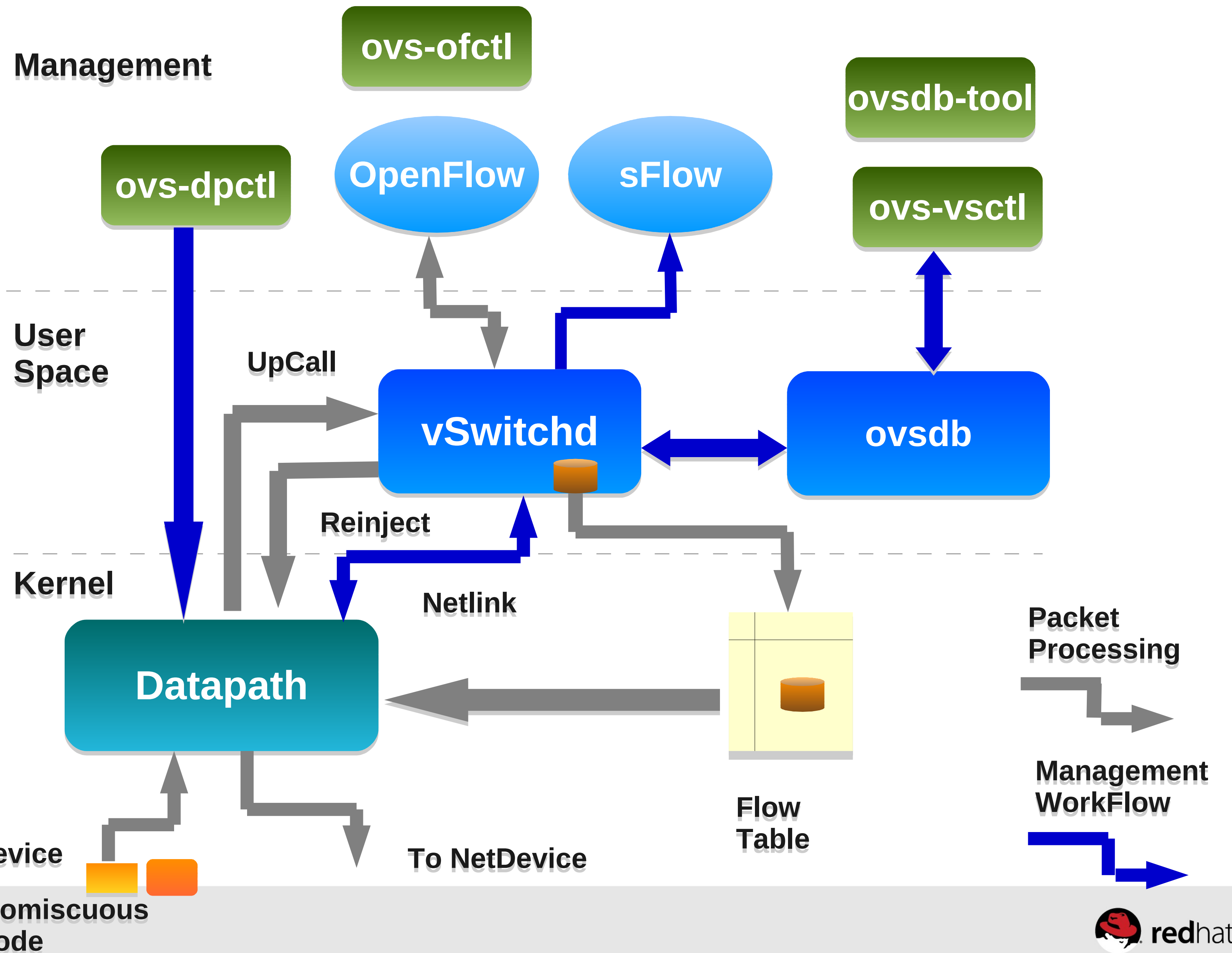


Open vSwitch

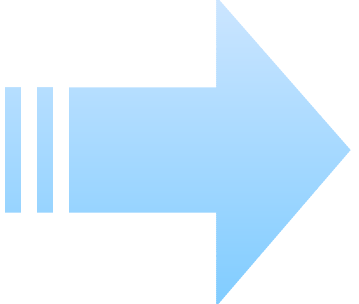
- Kernel Datapath
- Userspace daemon
- Configuration database
- Since RHEL 6.4



OPEN vSWITCH
An Open Virtual Switch



Open vSwitch operations

- Normal mode vs **Flow mode**
- Flow Table Match based on L2/L3/L4
- Forward, Drop Modify headers
- Systemtap Probe 

```
0 swapper(0): -> ovs_netdev_frame_hook
7 swapper(0): -> ovs_vport_receive
11 swapper(0): -> ovs_dp_process_received_packet
15 swapper(0): -> ovs_flow_extract
20 swapper(0): <- ovs_dp_process_received_packet
24 swapper(0): -> ovs_flow_tbl_lookup
28 swapper(0): -> ovs_flow_hash
32 swapper(0): <- ovs_flow_tbl_lookup
35 swapper(0): -> find_bucket
38 swapper(0): <- ovs_flow_tbl_lookup
41 swapper(0): <- ovs_dp_process_received_packet
45 swapper(0): -> ovs_flow_used
48 swapper(0): <- ovs_dp_process_received_packet
52 swapper(0): -> ovs_execute_actions
56 swapper(0): -> do_execute_actions
```

```
0 ovs-vswitchd(1302): -> ovs_vport_cmd_get
4 ovs-vswitchd(1302): -> lookup_vport
8 ovs-vswitchd(1302): -> ovs_vport_locate
12 ovs-vswitchd(1302): -> hash_bucket
15 ovs-vswitchd(1302): <- ovs_vport_locate
19 ovs-vswitchd(1302): -> ovs_netdev_get_name
22 ovs-vswitchd(1302): <- ovs_vport_locate
25 ovs-vswitchd(1302): <- lookup_vport
28 ovs-vswitchd(1302): <- ovs_vport_cmd_get
```

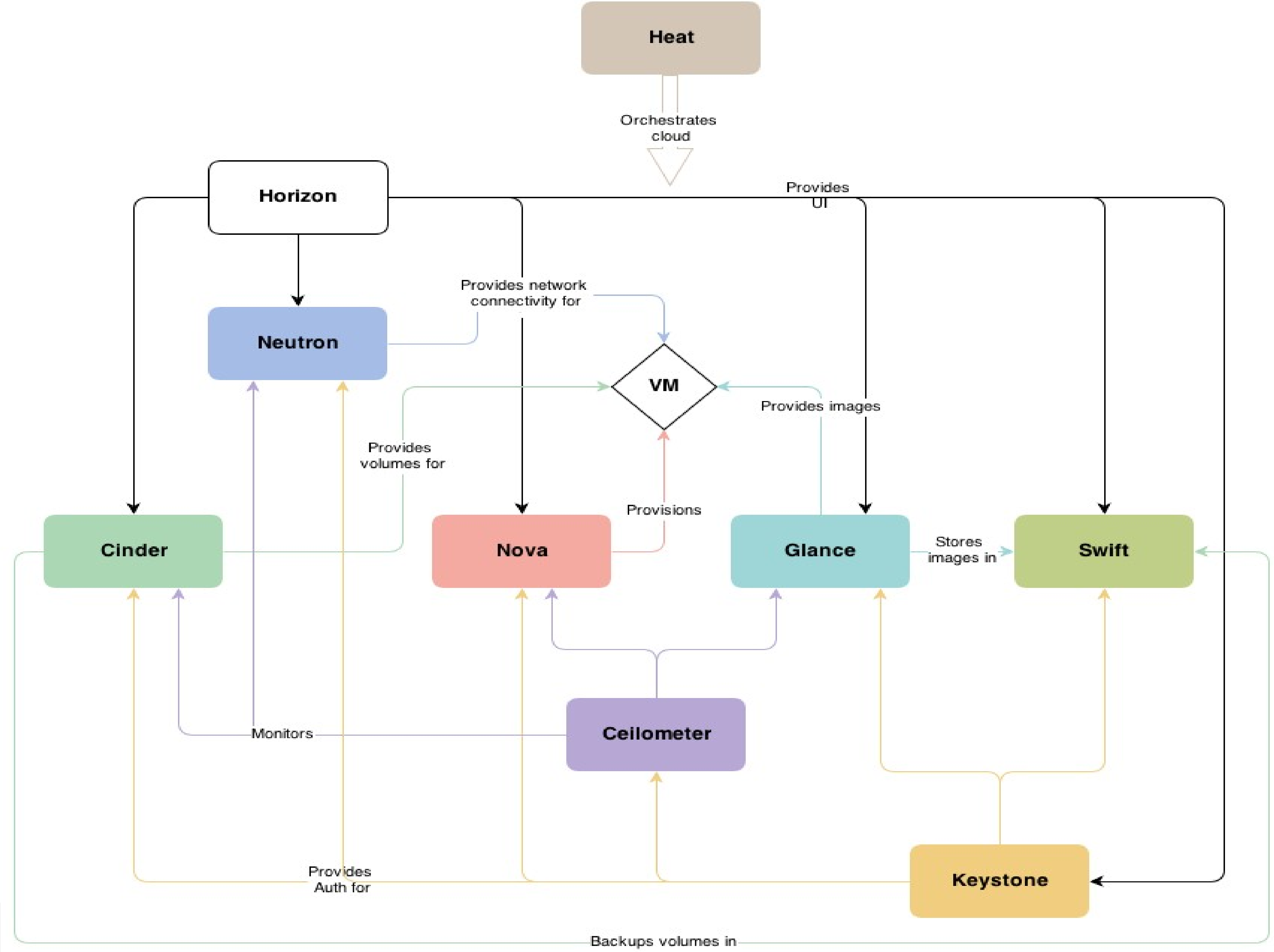
RED HAT
SUMMIT

10 YEARS *and counting*

SAN FRANCISCO | APRIL 14-17, 2014

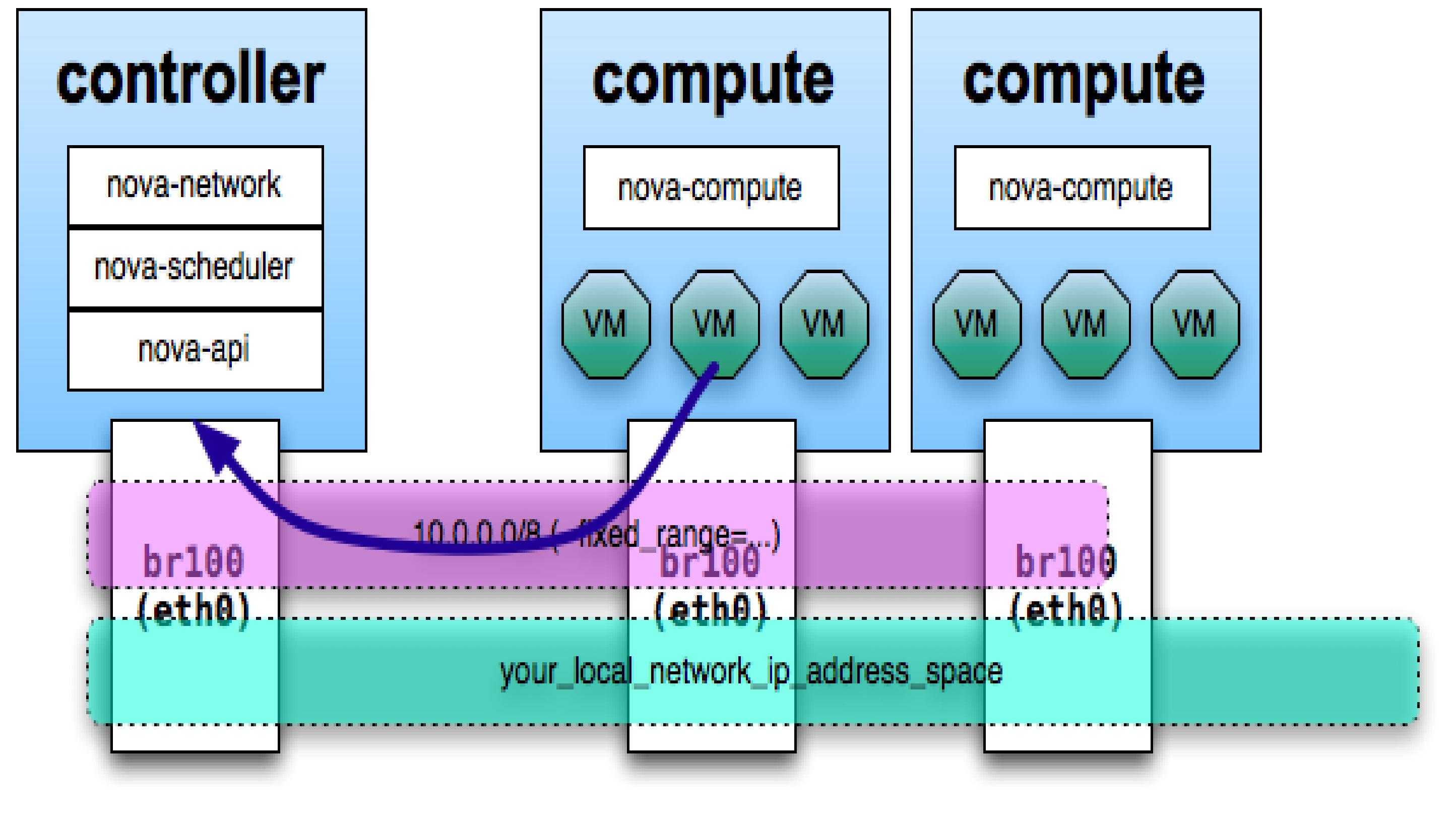
OpenStack (SDN)

OpenStack Architecture



Nova Networking

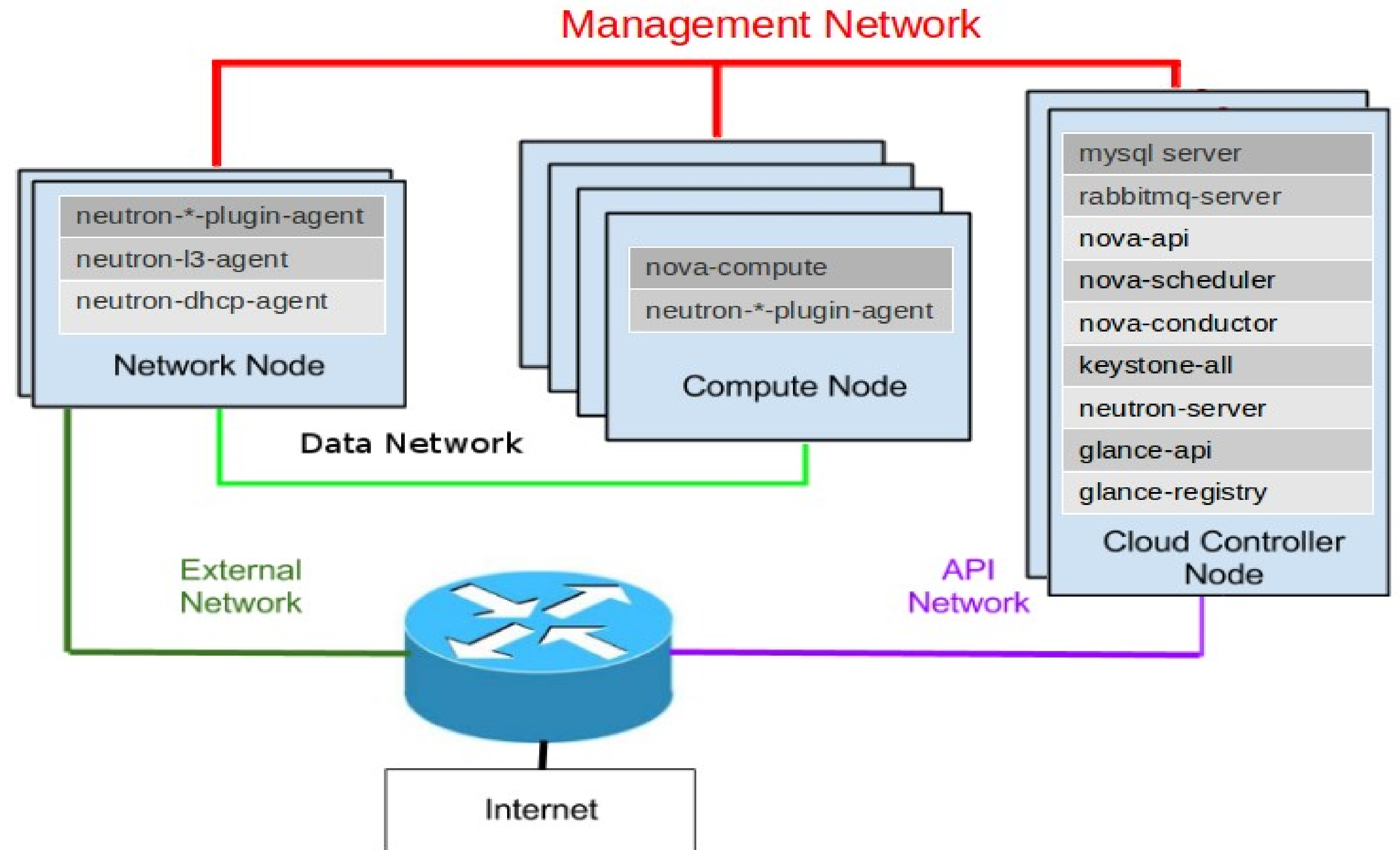
- Early days of Openstack networking
- Flat, Flat-DHCP, VLAN
- No router, firewalls etc



Neutron

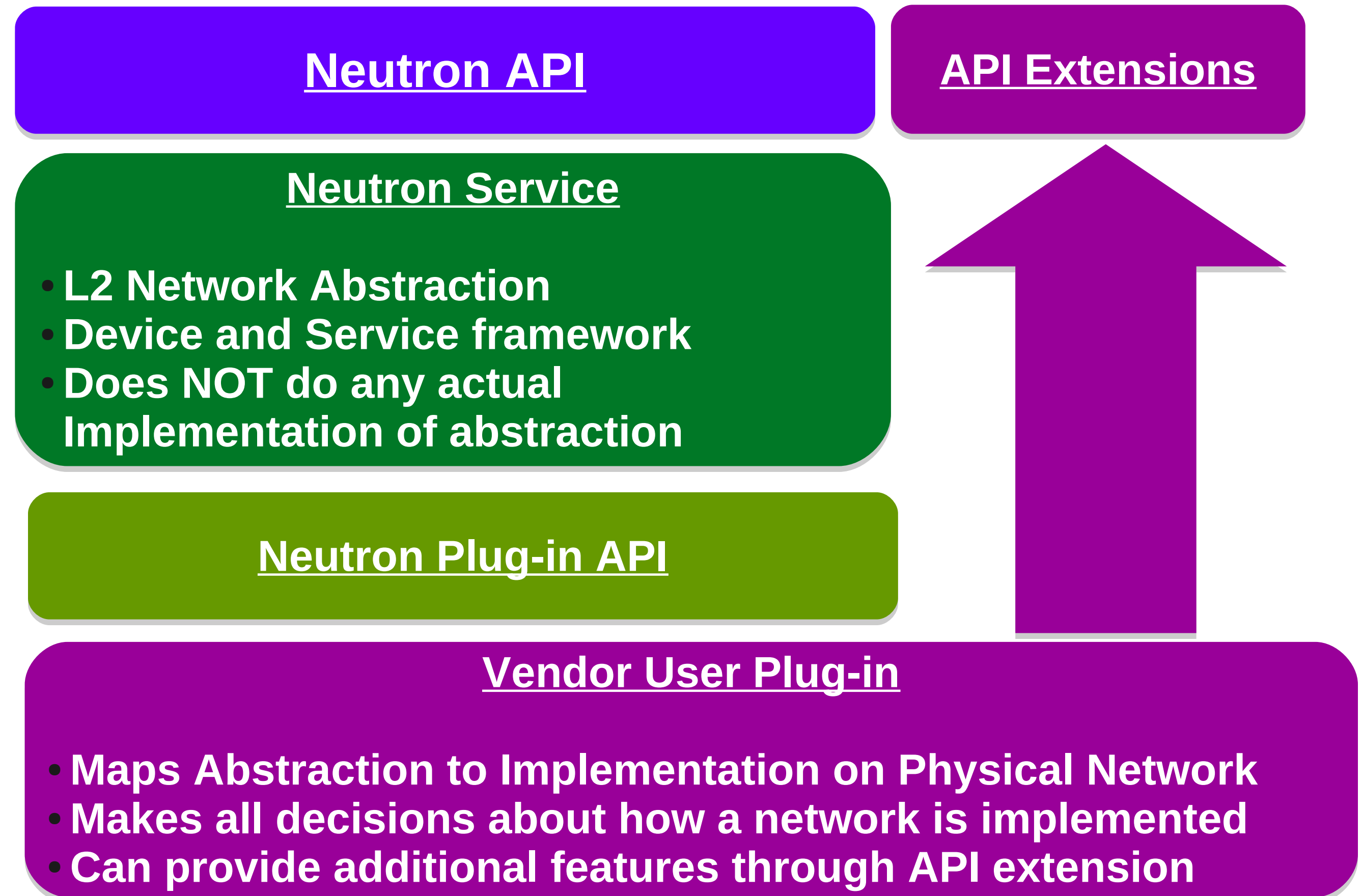


- Neutron is at the northbound side of the SDN framework
- Neutron provides network services to the Nova compute



Neutron plugin architecture

- Neutron services
- Various plugins connect to controllers or OpenFlow Switches



OpenStack Network Topology

Network Topology

Logged in as: admin [Settings](#) [Help](#) [Sign Out](#)

[Launch Instance](#) [Create Network](#) [Create Router](#)

The diagram illustrates the network topology in an OpenStack environment. It features three main vertical network segments: 'nova (external)' (green), 'private' (teal), and 'My fancy network' (purple). The 'nova (external)' segment is associated with the IP range 172.24.4.224/28. The 'private' segment is associated with 10.0.0.0/24. The 'My fancy network' segment is associated with 10.5.0.0/24. A router named 'router1' (labeled 'router') is connected to the 'nova (external)' network at IP 172.24.4.226 and to the 'private' network at IP 10.0.0.1. A server named 'My fancy n...' (labeled 'server') is connected to the 'nova (external)' network at IP 172.24.4.227 and to the 'private' network at IP 10.5.0.2. Another server named 'My other i...' (labeled 'server') is connected to the 'My fancy network' at IP 10.5.0.4.

openstack
DASHBOARD

Project Admin

CURRENT PROJECT
admin

Manage Compute

- Overview
- Instances
- Volumes
- Images & Snapshots
- Access & Security

Manage Network

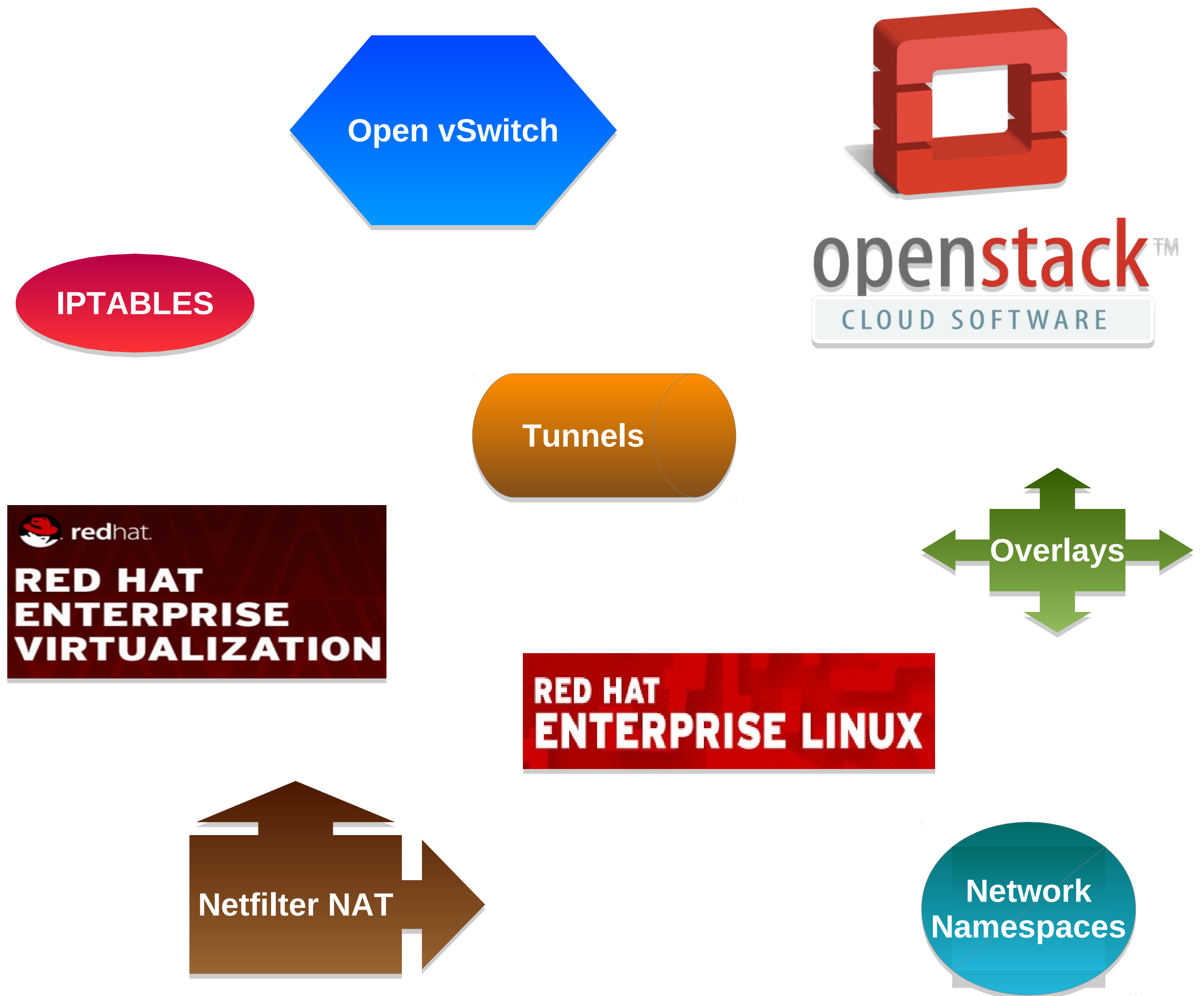
- Networks
- Routers
- Network Topology

Object Store

- Containers

OpenStack SDN

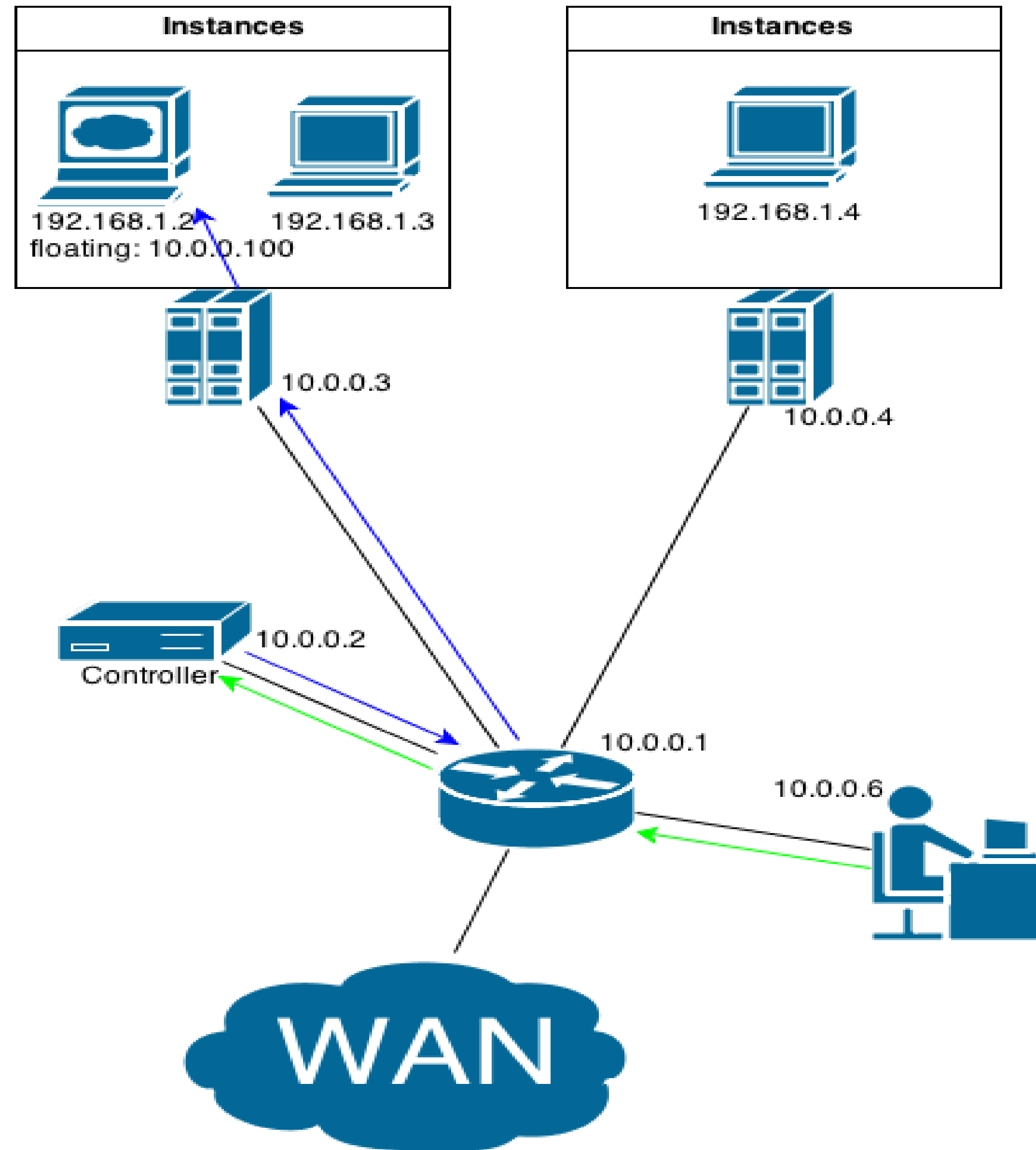
- Various Components
- Iptables
- Open vSwitch
- Overlay networks
- Tunnels – GRE/VXLAN
- Network Namespace
- Netfilter NAT for Floating IP addresses





OpenStack Floating IP

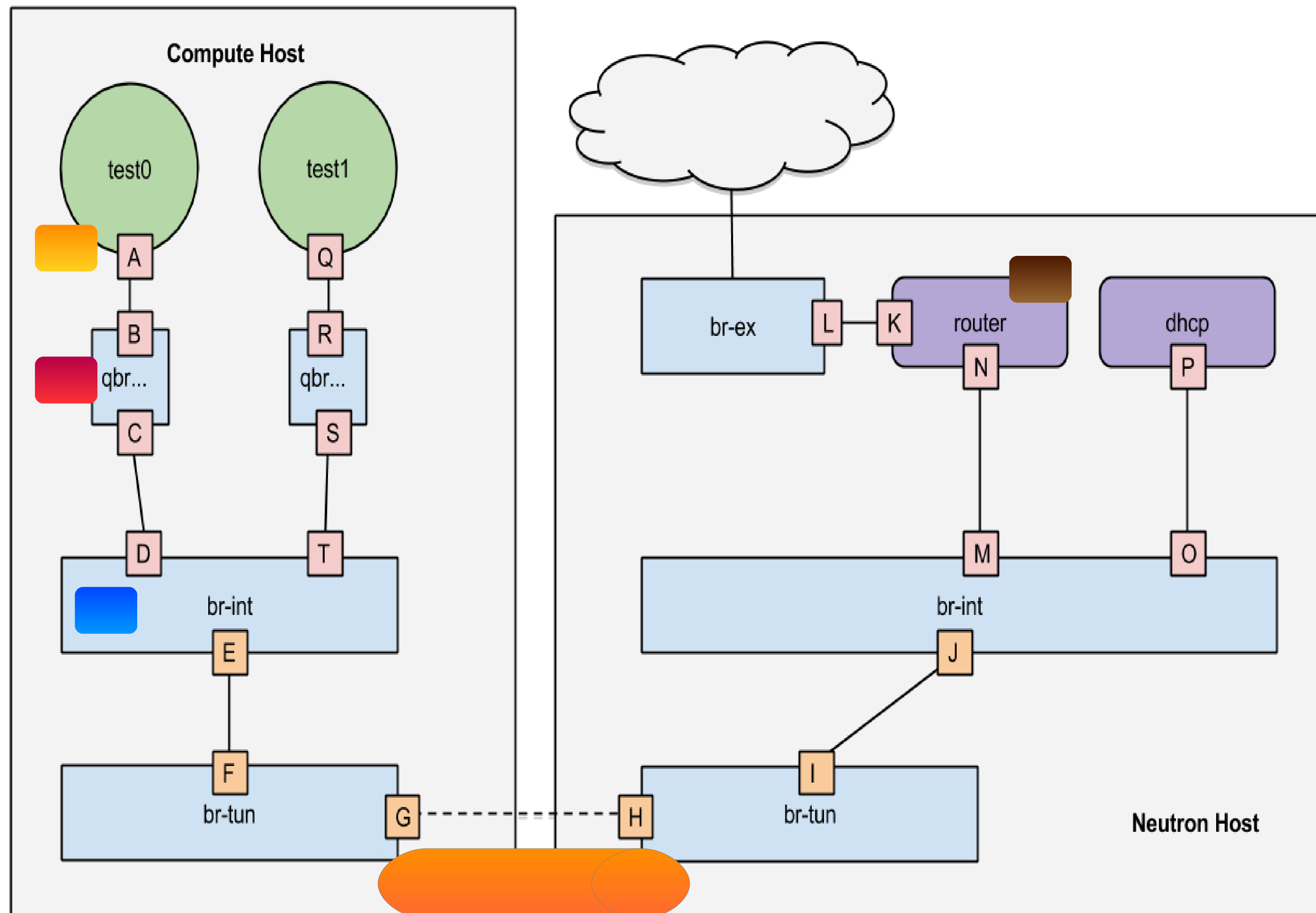
Red Hat Enterprise Linux
OpenStack Platform



Legend	
	Node
	User
	Physical router
	Virtual space with instances
	HTTP access using private IP
	HTTP access using floating IP

OpenStack SDN

- A, B, C :: Tap, Fw Bridge, **Iptables**
- D, E :: **VLAN tagging**
- F, G :: **Tunnels**
- **Open vSwitch**, GRE
- O, P :: DHCP
- M, N :: **Router**
- Netfilter NAT**



Future Trends

- Need to stick to **open standards**
- Need to have a stable SDN ecosystem
- Standardization for various components
- Allows for various vendor solutions (open/closed source)
- Scope for **Innovation at each layer**- Apps, Controller, Protocols, Devices (Physical/Virtual)
- Possible scope for **Hardware acceleration** products in SDN space
- Evolution of **NOS**



Thank You ! Questions ?

<vdasgupt@redhat.com>

Please complete this session survey available on the mobile app :: Complete 8 surveys – Win Exciting Prizes !!

Slides in PDF will be available later at
www.redhat.com/summit/2014/presentations