MSDC networks
OpenFlow / OpenStack

Minnesota OpenStack Meetup

25-mar, 2013

steve ulrich

sulrich@cisco.com / @sulrich

kyle mestery

kmestery@cisco.com / @mestery

background – about me

- networking nerd in service provider domain
  - core routing / peering / L(2|3)VPN / broadband
- past few years working with some of cisco's MSDC customers
  - i'm not really interested in compute
  - moving bits is very interesting to me

MSDC = Massively Scalable Data Center

- web scale providers
  - Google, Yahoo!, Amazon, Facebook, Microsoft
- various hosting providers
  - RackSpace, SoftLayer, etc.

MSDC characteristics

- single administrative domain with a focus on **infrastructure**
  - service provider operational orientation
  - highly automated operational environments
  - server guys and router router guys not that far from each other
- very large number of servers and VMs
  - O(100K) servers, millions of VMs
- very large compute, storage and b/w requirements
- very large number of application specific east-west flows
- highly unpredictable traffic matrix and mixture
- very mobile workloads
- warehouse scale data centers, spread across metros and around the world

MSDC design objectives

- maximize resource utilization!
  - run the networks hot / run the servers hot / do not crank up the AC
- maximize flexibility / mobility
  - run apps / analytics / experiments on the same infrastructure
- reduce costs!
  - every penny counts at scale

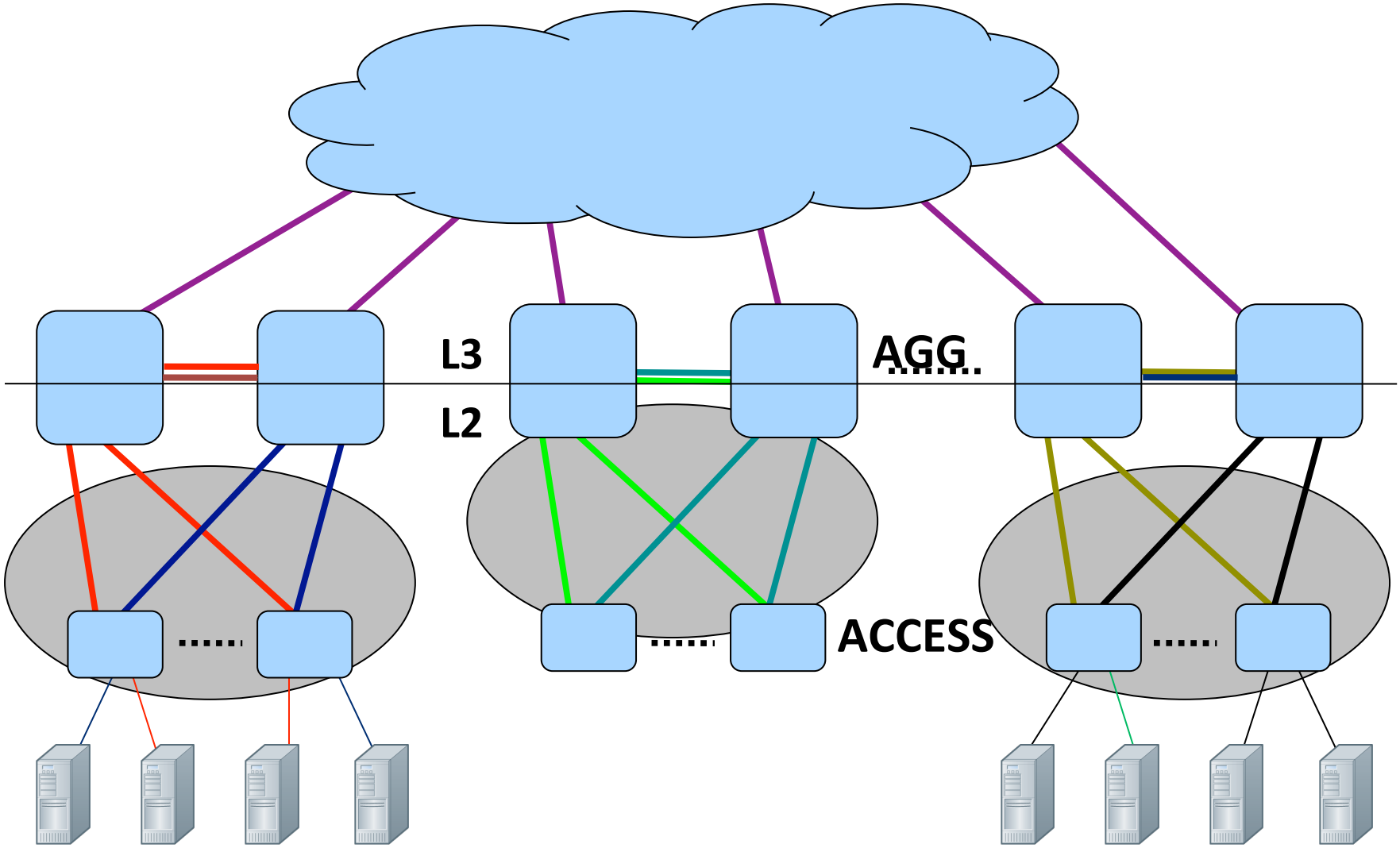
  e.g.: $0.01 / KwH = ~$1500 / year / rack == ~$1M / pod/cluster / year

WARNING

disclaimer - what follows is not comprehensive, nor a how-to. these discussion points are not necessarily appropriate for everyone's environment.

these are not problems everyone has but some of the concepts are potentially useful for thinking about how things may evolve in emergent DC networks driven by OpenStack and associated technologies.

common enterprise DC architecture



problems with ^^that picture^^

- poor utilization of network capacity
- sparse address space allocation and utilization
- big failure domains
- diverse troubleshooting methodologies
  - L2 domain + L3 domain
- lots of protocols + attendant complexity
  - STP/SPB/TRILL/FP + HSRP/VRRP + IP routing + services

MSDC network architectures commonly favor a different set of tradeoffs than most enterprises can accept

challenges - server to server capacity

- hierarchical network - servers in different L2 domains must go through the agg (L3) layer
- bandwidth between servers in different parts of a DC can be severely constrained

  application bandwidth management is a hard optimization problem

example: load balancing use case

- some LB techniques (NAT/HALF-NAT/DSR) require that all DIPs in a VIP's pool be in the same L2 domain.
- if an application grows and requires more front-end servers, it cannot use available servers in other L2 domains
  - drives fragmentation and under-utilization of resources

challenges – static network assignment

- common enterprise DC practice is to map specific applications to specific physical switches and routers
  - commonly depends on VLANs and L3 based VLAN spanning to cover the servers dedicated to the application
- derivative problems
  - VLANs drive policy

    VLAN = traffic management, security, and performance isolation
  - VLAN spanning places traffic on links high in the tree unpredictable traffic loading and oversubscription

VLAN provisioning == total hass

MSDC network requirements

- east-west traffic dominates
  - scatter/gather workloads ➔ 80% E-W / 20% N-S
- must provide deterministic latency
- must be robust – design to absorb failure
- automated, simplified management
  - setup / ongoing mgmt / debug
- must readily enable varying levels of oversubscription
  - 1:1 / 2:1 / 3:1 – you get the point
- must be scalable
  - 1K – 20K ports in a cluster/pod
- enable seamless addition of capacity and service components
- increasingly pervasive dual-stack environment
  - production traffic & management traffic

some philosophical perspectives

- crap is better
  - build out with lots of dumb, high-capacity links
  - stuff's going to break – deal.
- overlays provide mobility and reachability for available resources

  learn to love the tunnel – don't get too hung up on the flavor of encap

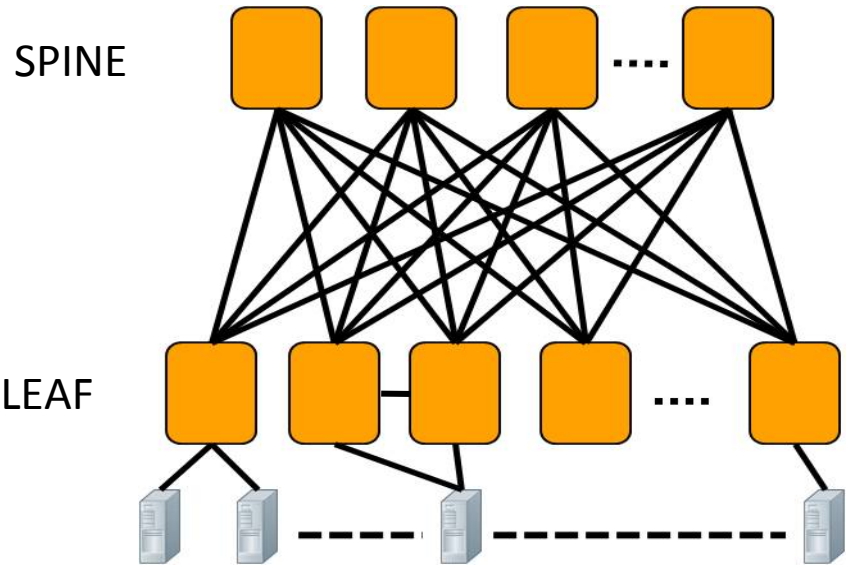
  seriously – focus on the function and the value and think about the long-term value it provides

  that said: be careful – here be some dragons
- automate everything

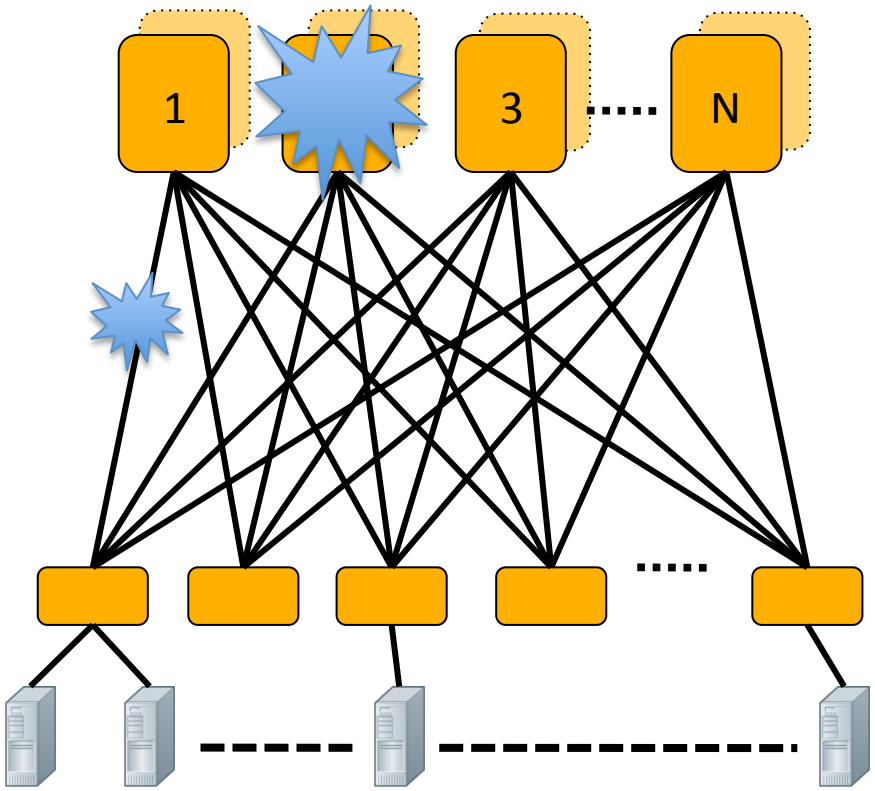
modern DC architecture

3 key components

- topology that is based on a CLOS (fat tree) network
- distributed control plane that handles forwarding
- management framework that simplifies fabric management



be robust

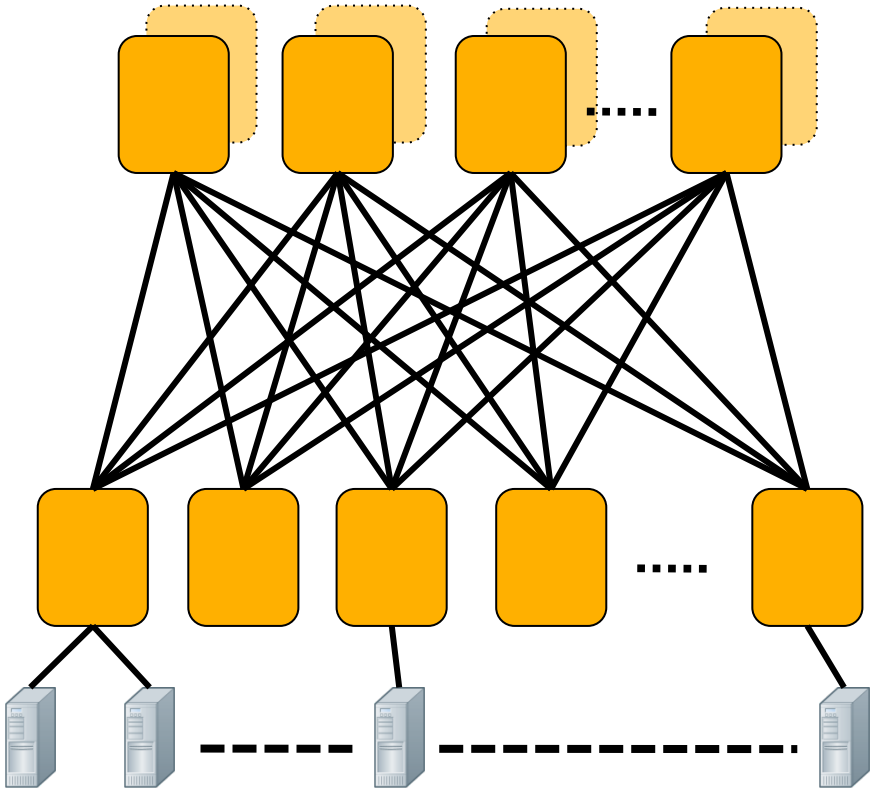


build a system that assumes things will fail.

spine width	capacity loss*
4	25%
8	12.5
16	6.25%
32	3.125

* for the loss of a single spine node

modular and scalable



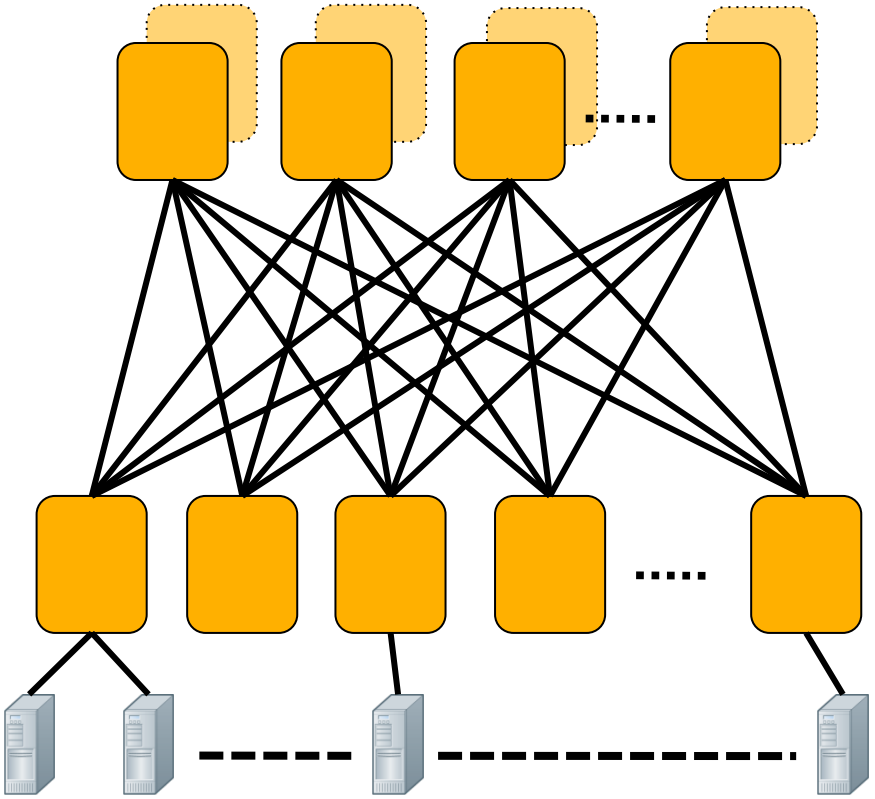
N ports – modular (or fixed)

avails full bisectional bandwidth

M ports - fixed

- supports a mix of switches and sizes
- enables incremental deployment and growth

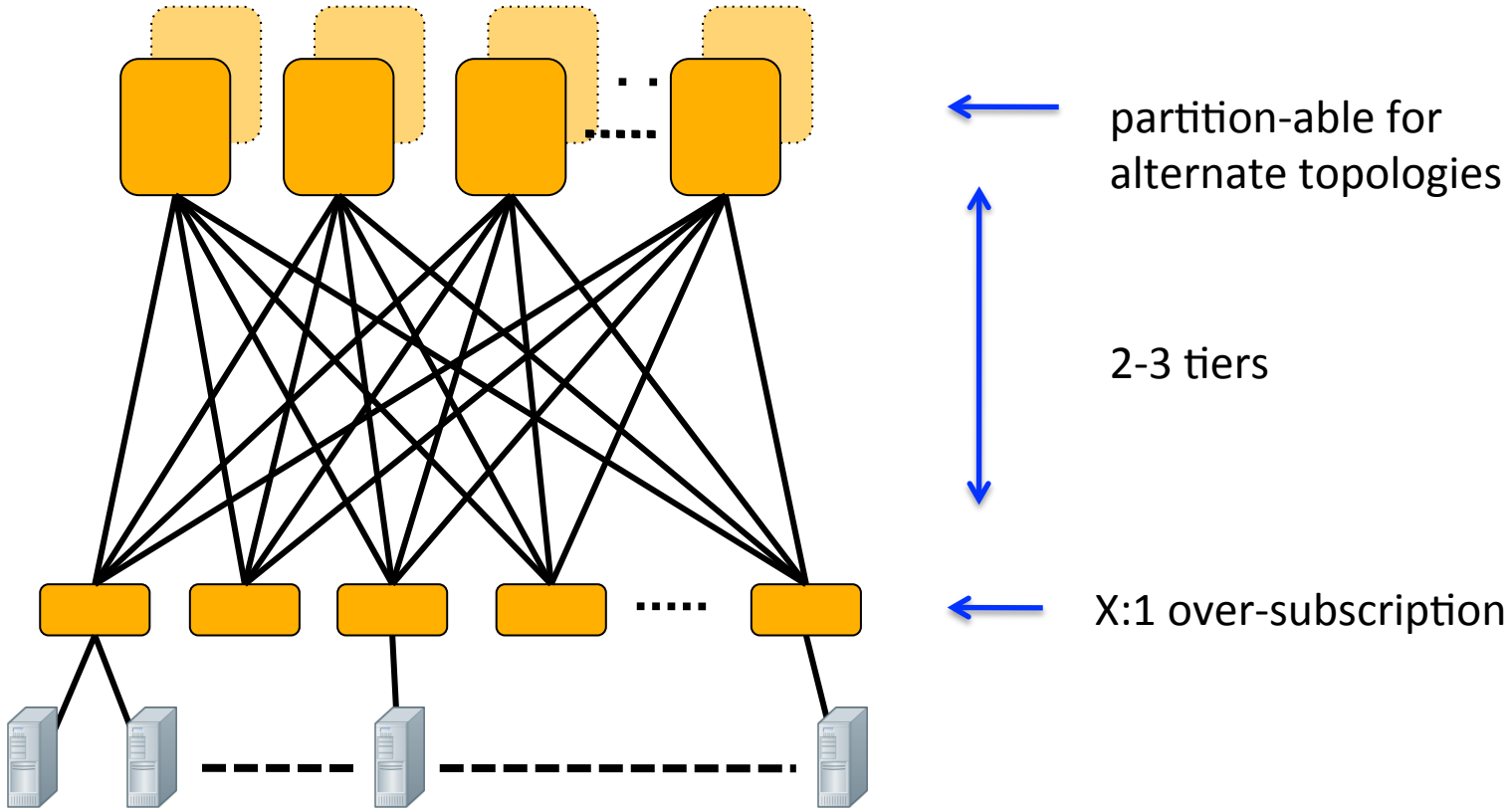
simple scaling characteristics



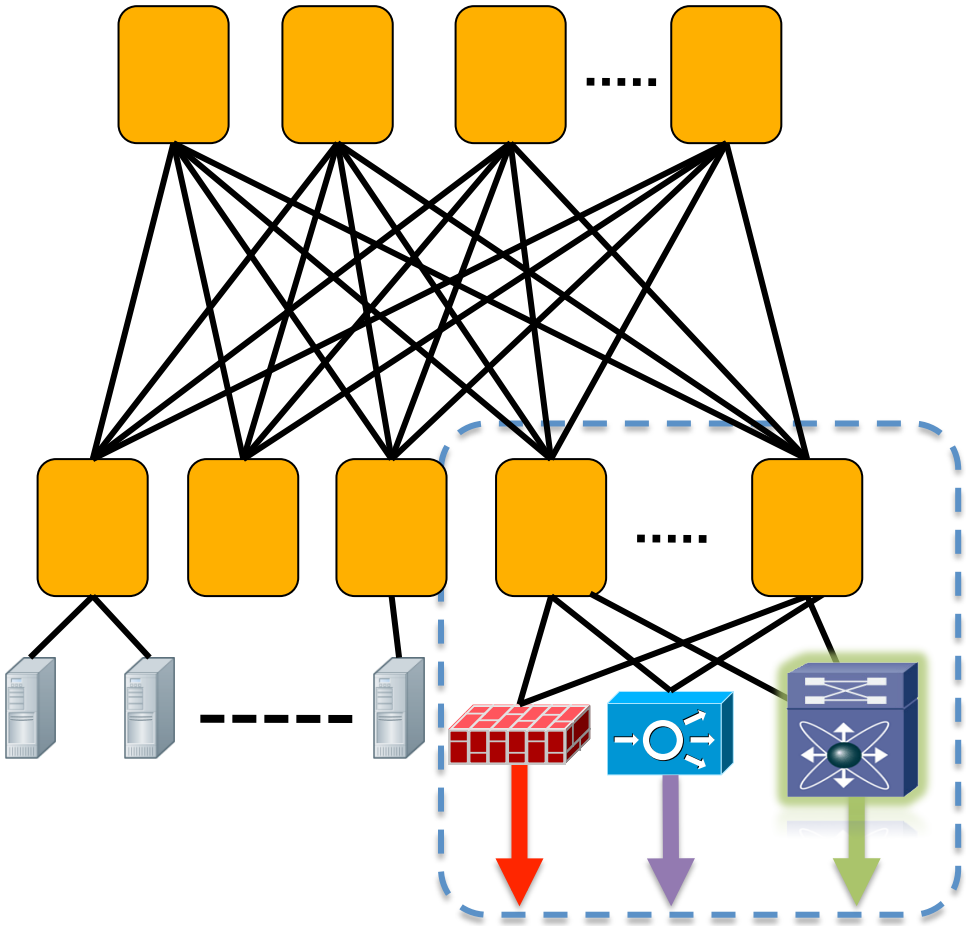
Leaf Ports	Spine Ports	Total # Servers
48	48	1,152
64	768	24,576

$$\text{maximum total ports} = \#\text{ of spine ports} \times \left(\frac{\#\text{ of leaf ports}}{2} \right)$$

flexible topology



leaf attachment



- avoid hotspots
- provides for clean separation of control protocols
- simplifies fabric management

some MSDC wrap up…

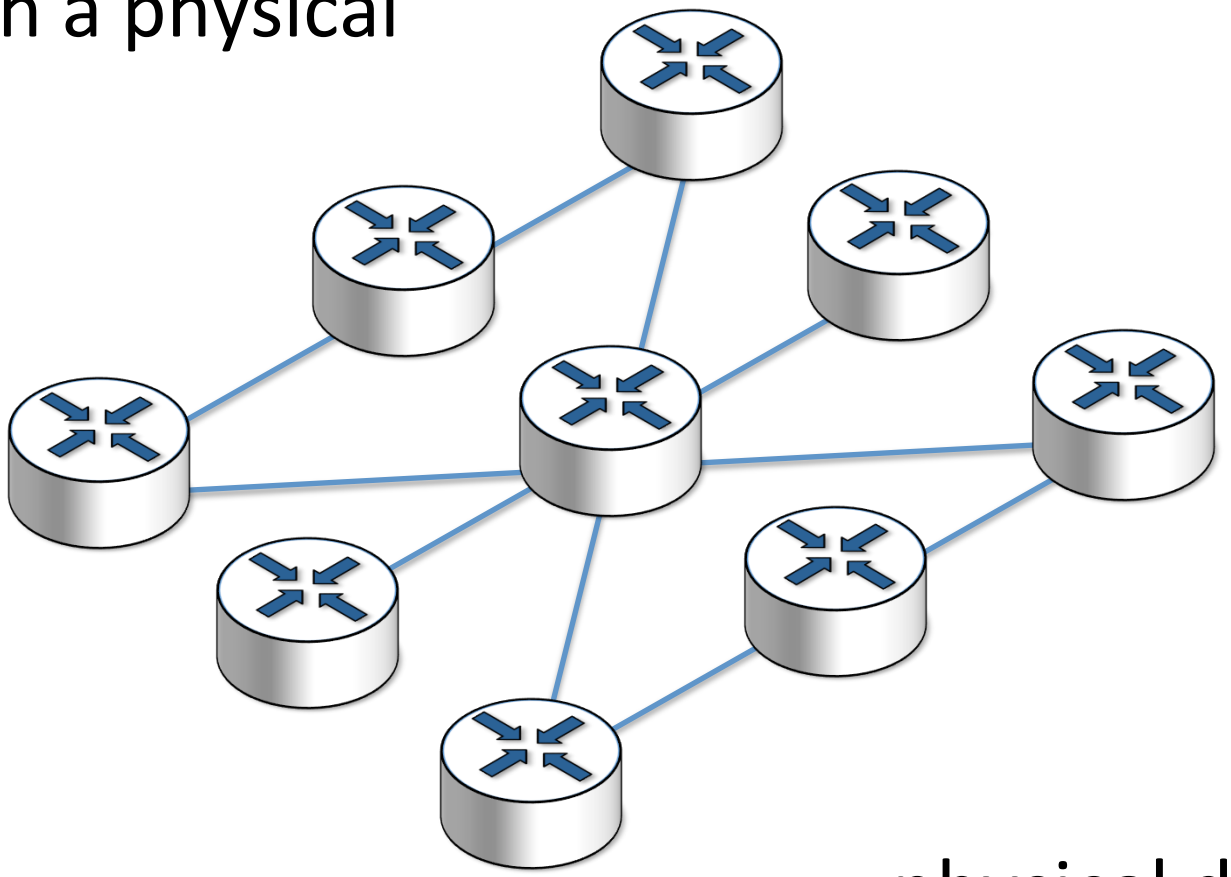
- often take different approaches to a problem space than enterprises have the ability to
- highly mobile workloads (and their placement) drive some of the same problems within OpenStack networks
  - customer / rack / service alignment is rarely right where you'd like it to be
  - worse still is application placement
- managing servers and network at scale drives pervasive automation – program your intent

OVERLAYS AND UNDERLAYS

the physical network

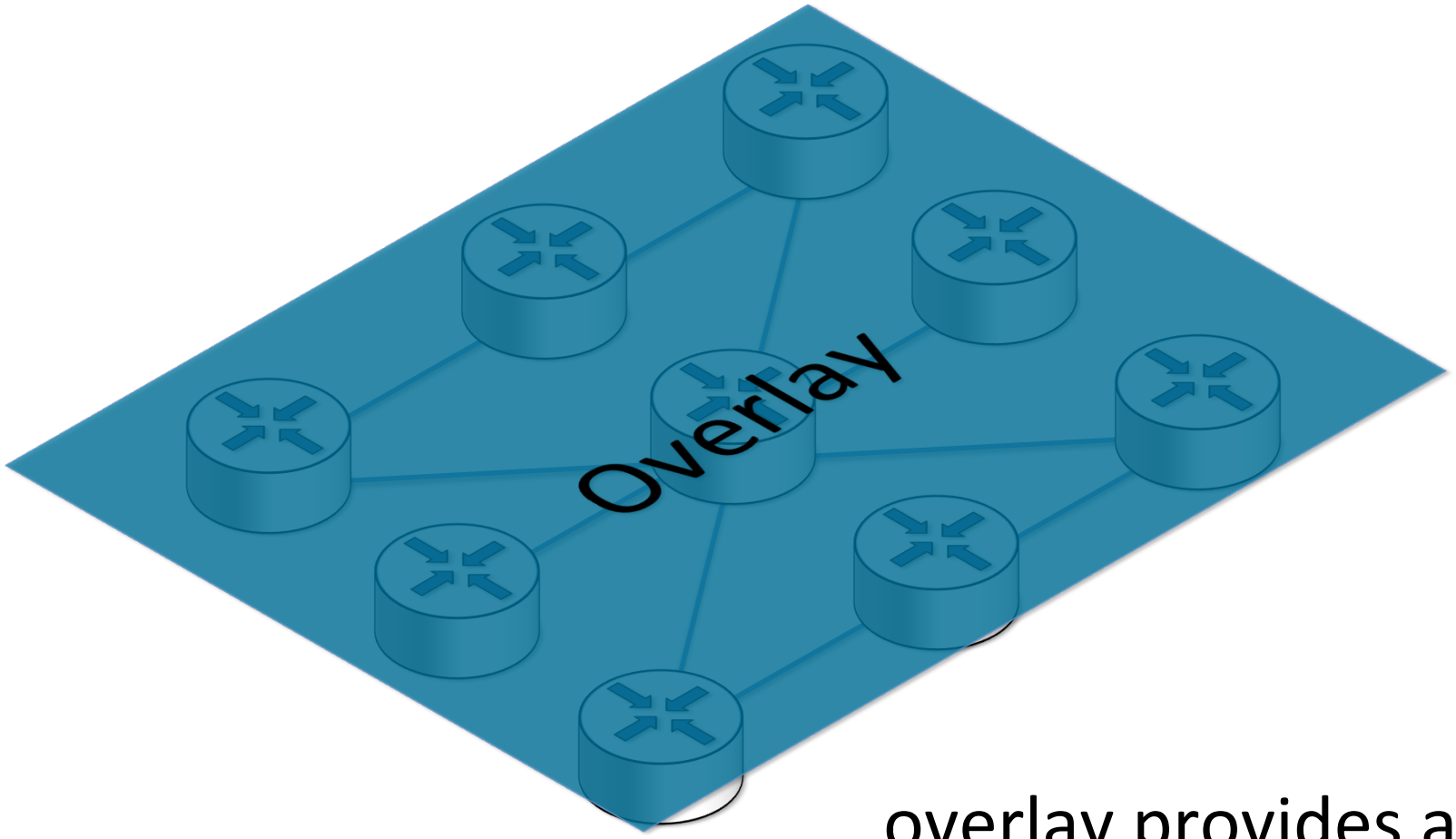
(aka the underlay)

start with a physical network



physical devices and physical connections

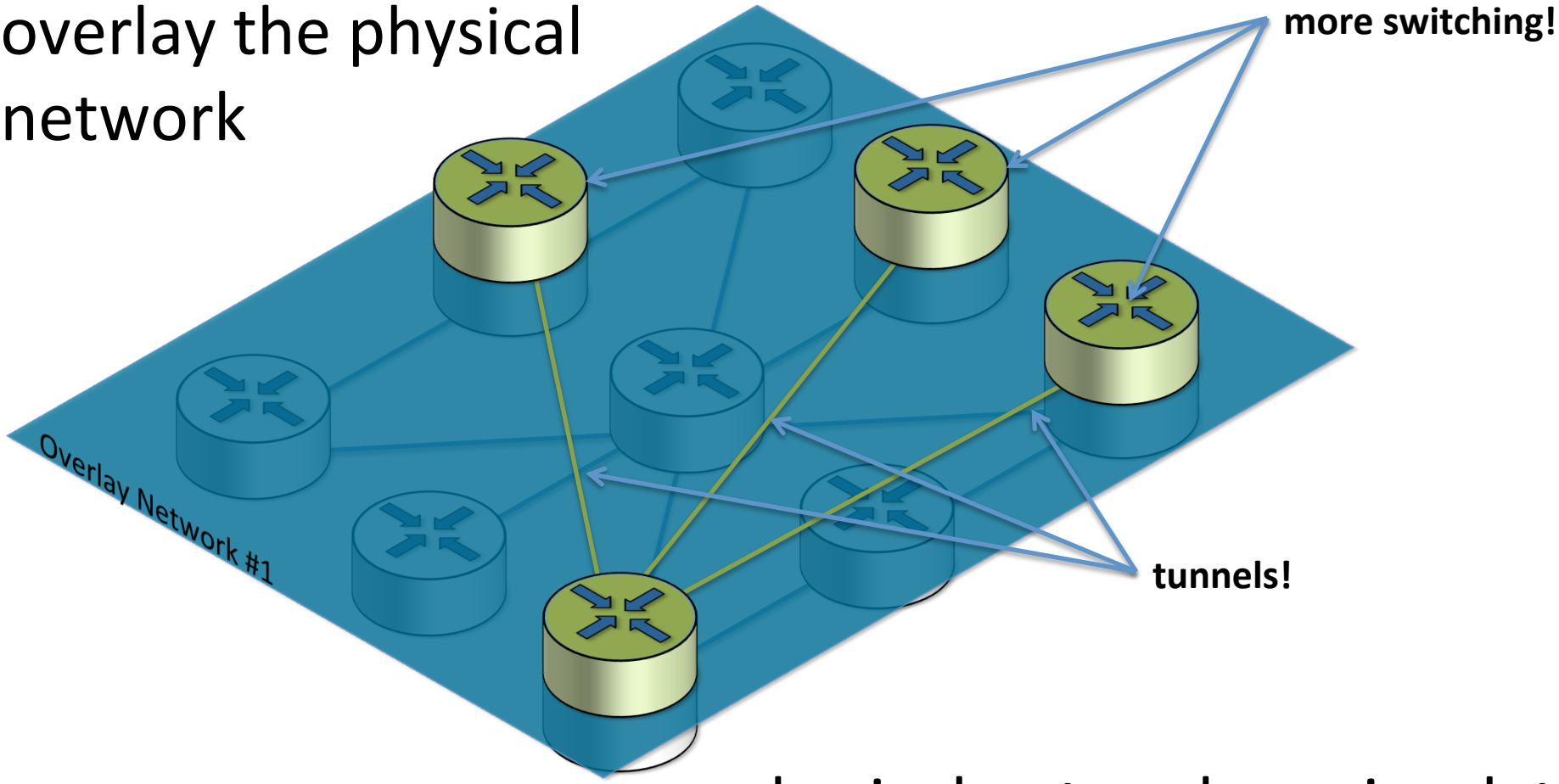
add an overlay



overlay provides a base
for the logical network

add topologies

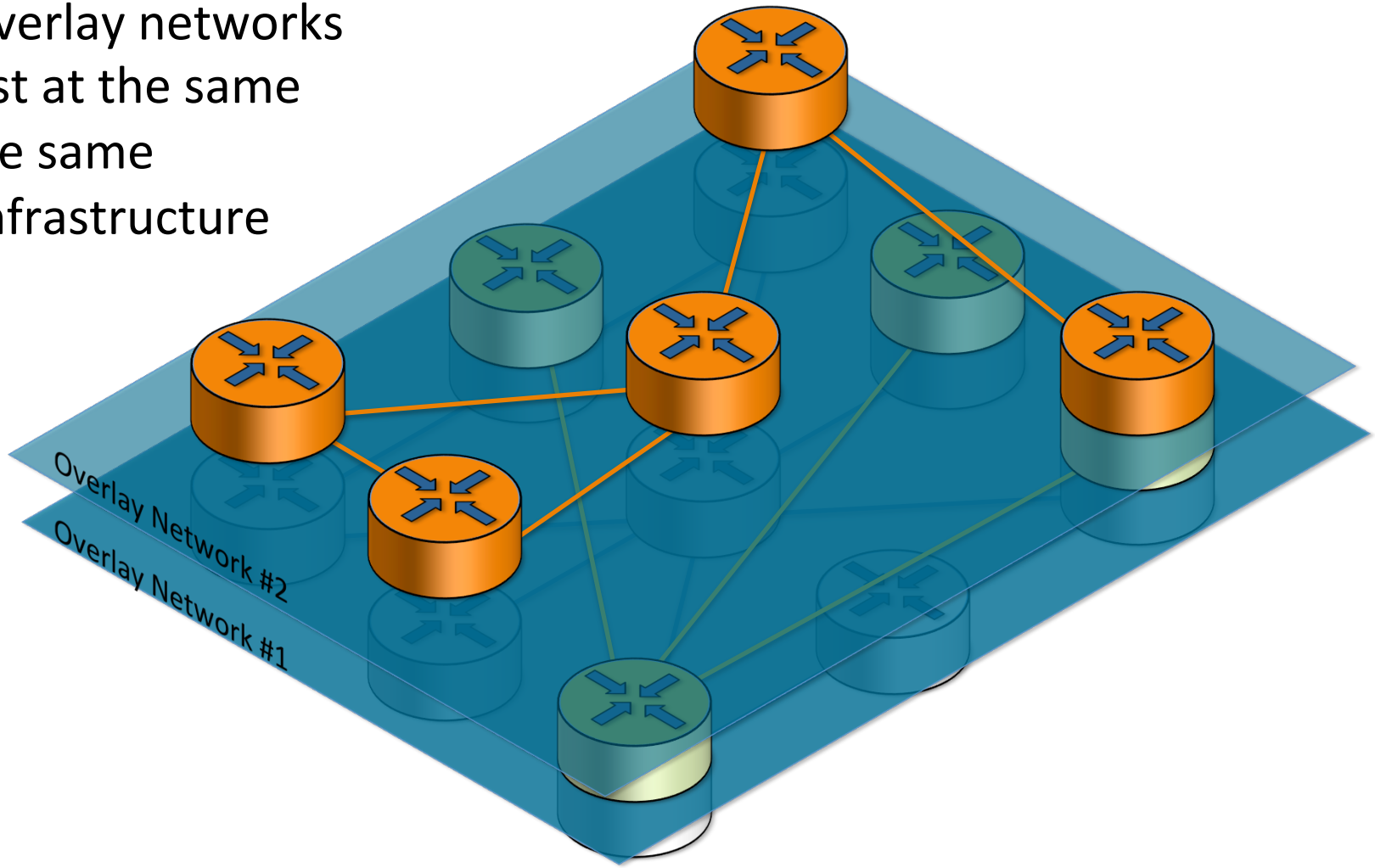
logical “switch” devices overlay the physical network



physical network carries data traffic for overlay network

lather, rinse, repeat

multiple overlay networks can co-exist at the same time on the same network infrastructure



overlays provides logical network constructs for different tenants, applications, workloads, etc.

WHAT DOES THIS ACHIEVE?

challenges - server to server capacity

MSDC approach

- maximize bisectional bandwidth through network topology and configuration

- appropriately place the workload

  enabled through the use of a programmable network topology … programmable (over|under)lay

  <u>program</u> the network to get to these states

challenges - static network assignment

MSDC approach

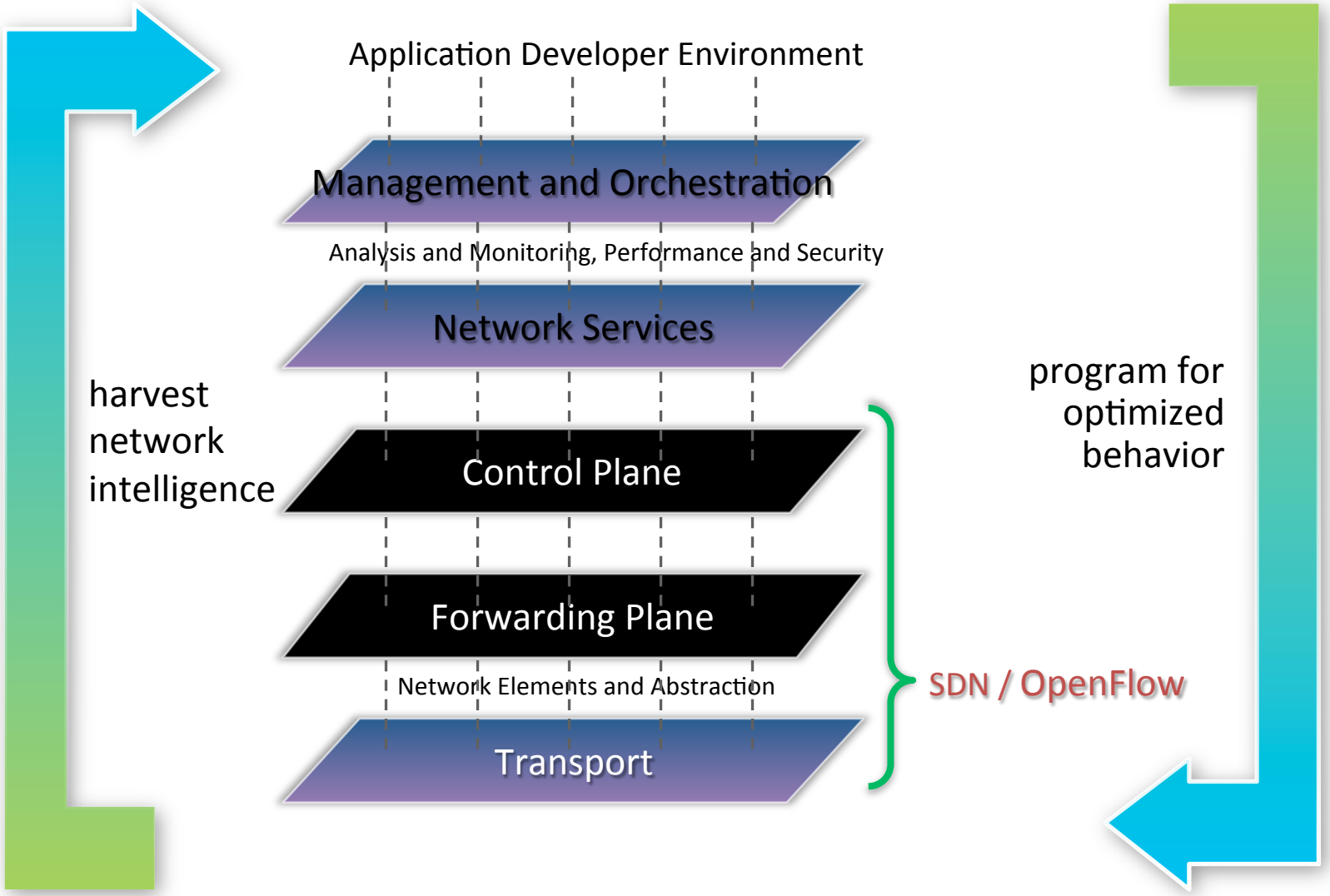
- build the topology and expand the network appropriately over the fabric to address the application requirements
- wash your hands of the issues associated with L2/L3 boundaries and scale beyond VLANs

program the network to get to these states

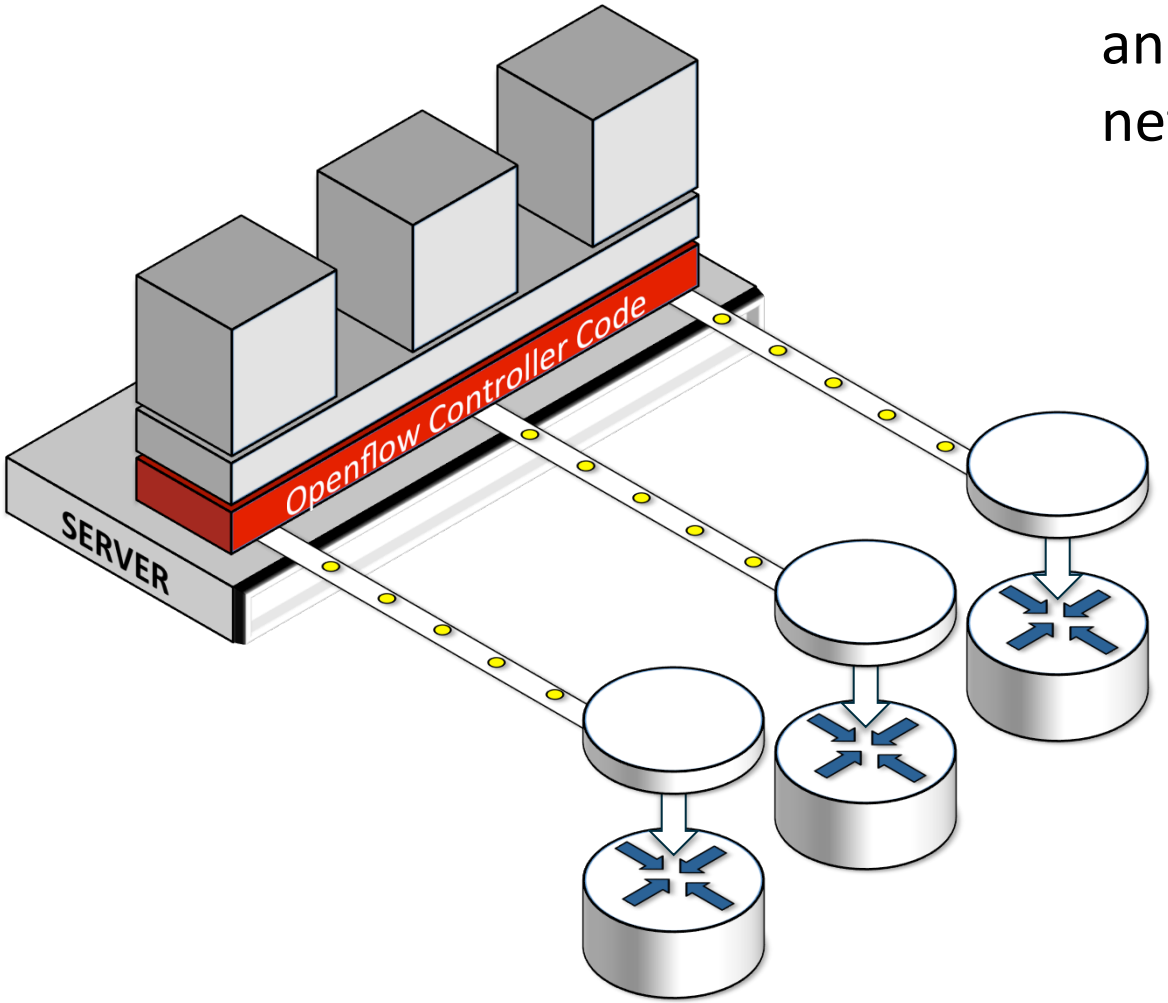
ENTER OPENFLOW (OR SDN)

programmability at multiple layers of the network

butcher what you want, where you want

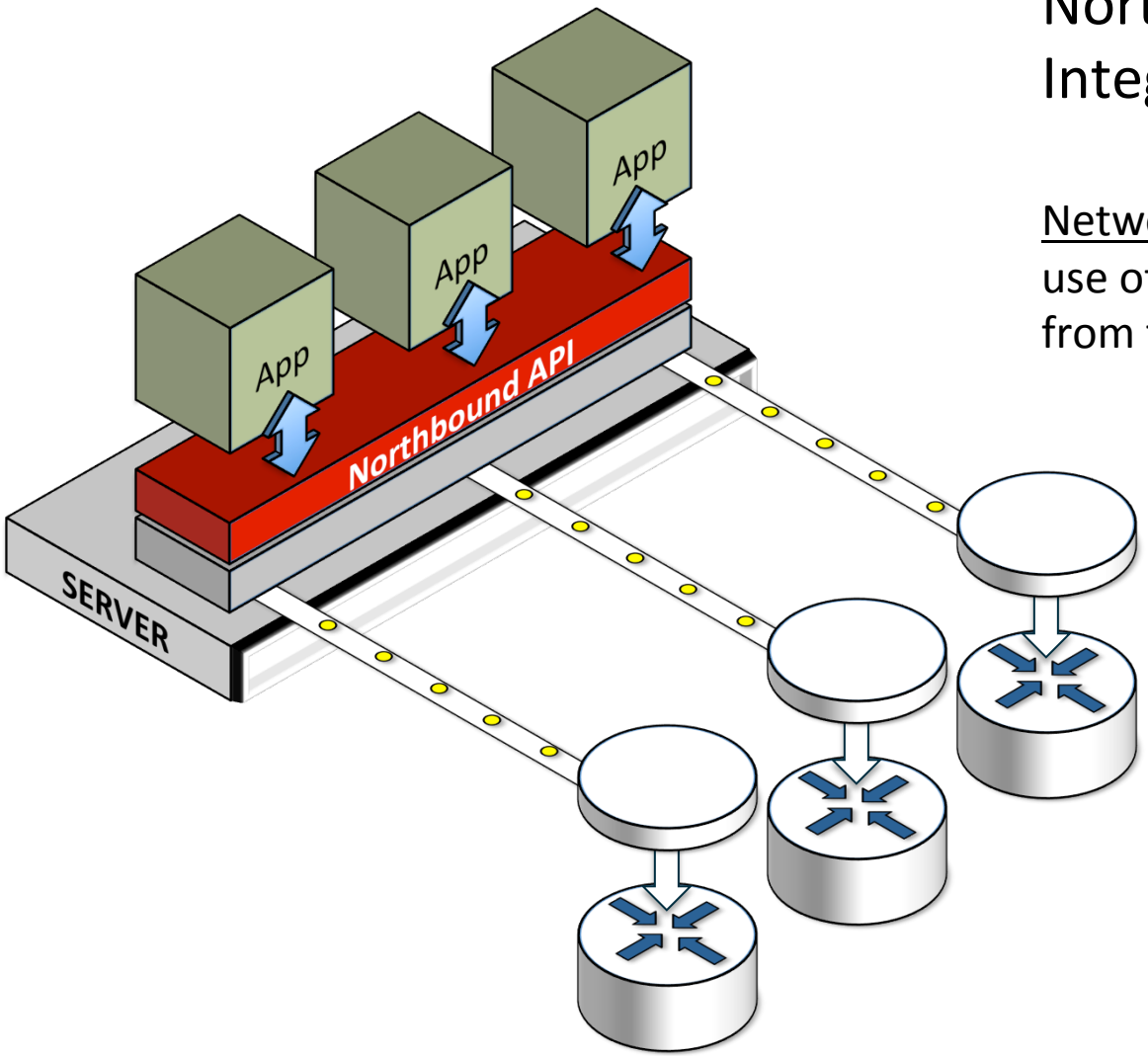


Openflow controller



central administration and operations point for network elements

Openflow Controller - Northbound API

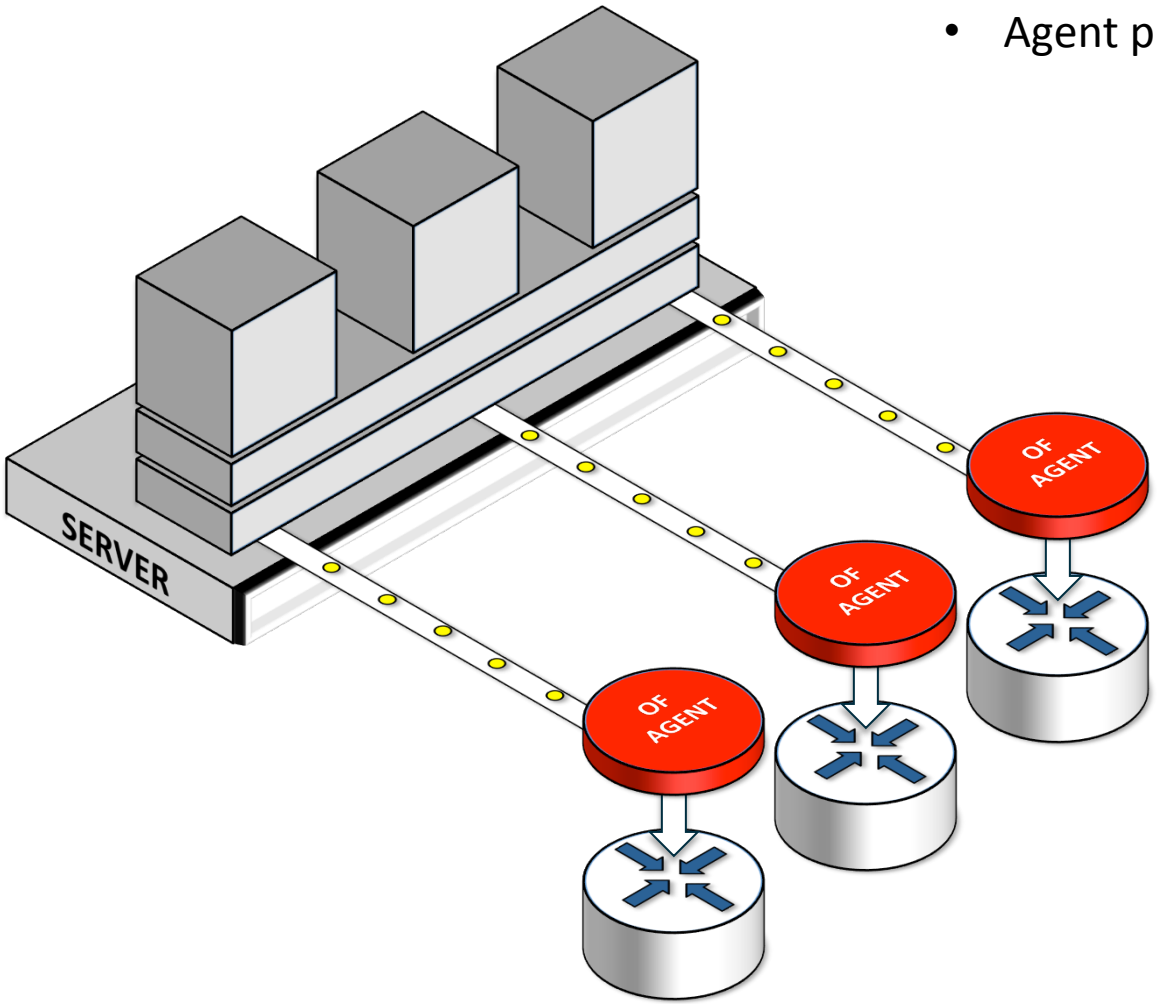


Northbound API
Integral part of Controller

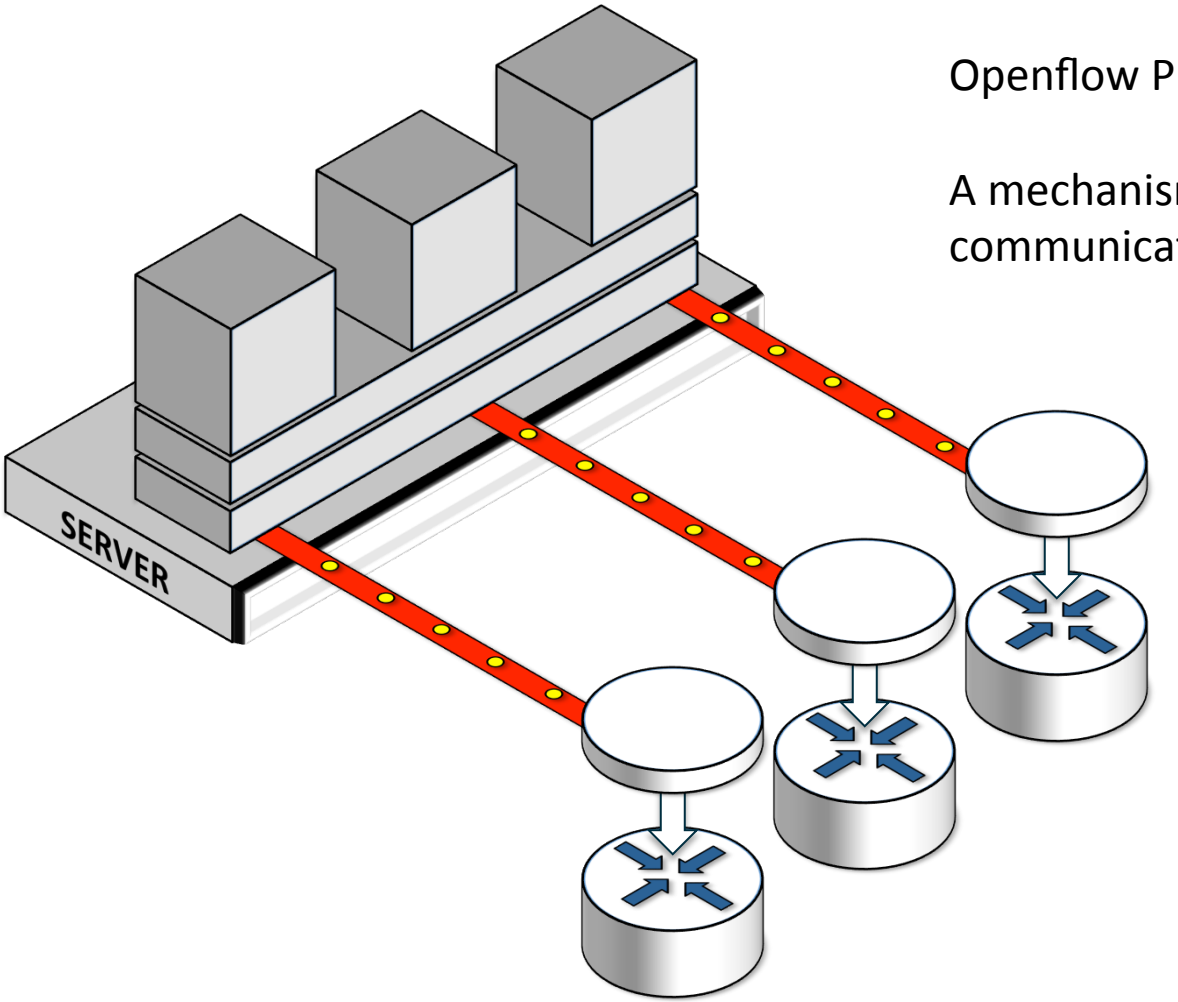
Network enabled application can make use of Northbound API to request services from the network

Openflow Device Agent

- Agent runs on the network device
- Agent receives instructions from Controller
- Agent programs device tables



Openflow protocol



Openflow Protocol

A mechanism for the Openflow Controller to communicate with Openflow Agents

what is this OpenFlow protocol?

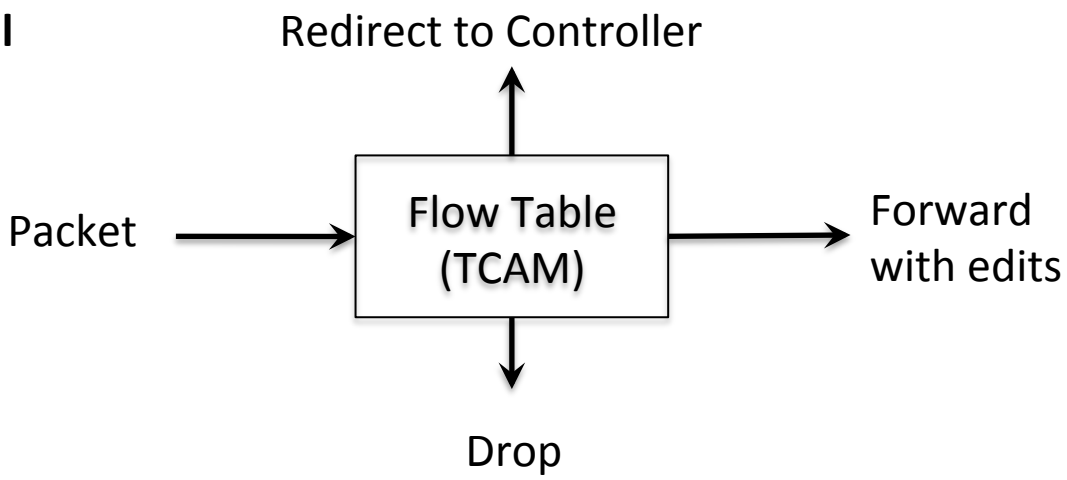
instruction set for switching

- **an information (switch) model**
  - Match/Action/Tables (MAT)

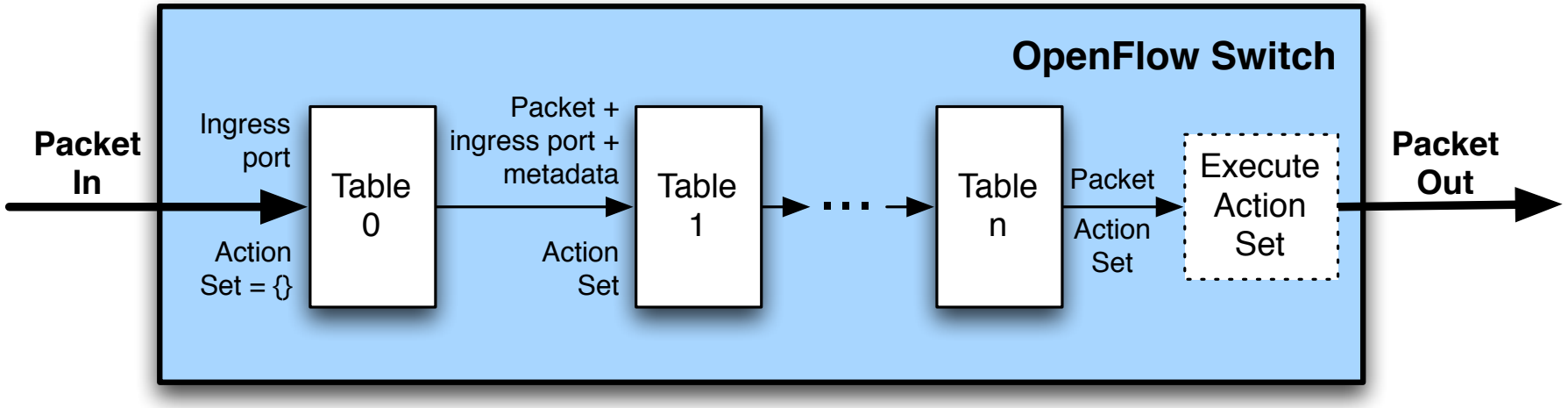
  abstract switch model – not an implementation specification
- **an application layer protocol (ALP)**
  - binary wire protocol and messages that allow programming of the MAT
- **a transport protocol**
  - SSL/TLS - transports the ALP

OpenFlow switch models

OpenFlow 1.0 model



OpenFlow 1.1 model

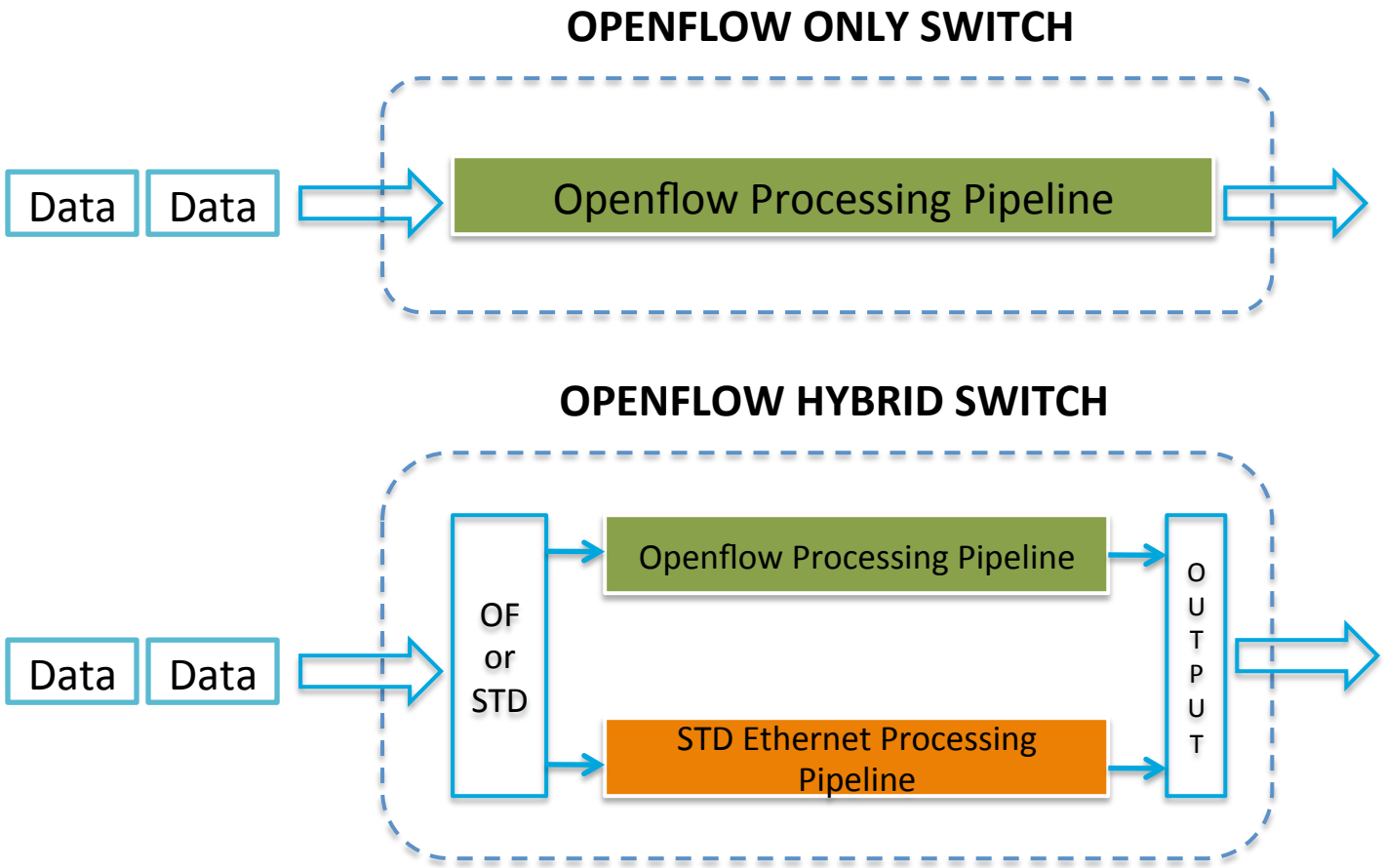


(a) Packets are matched against multiple tables in the pipeline

+ Multicast / ECMP /Anycast / MPLS

OpenFlow 1.1

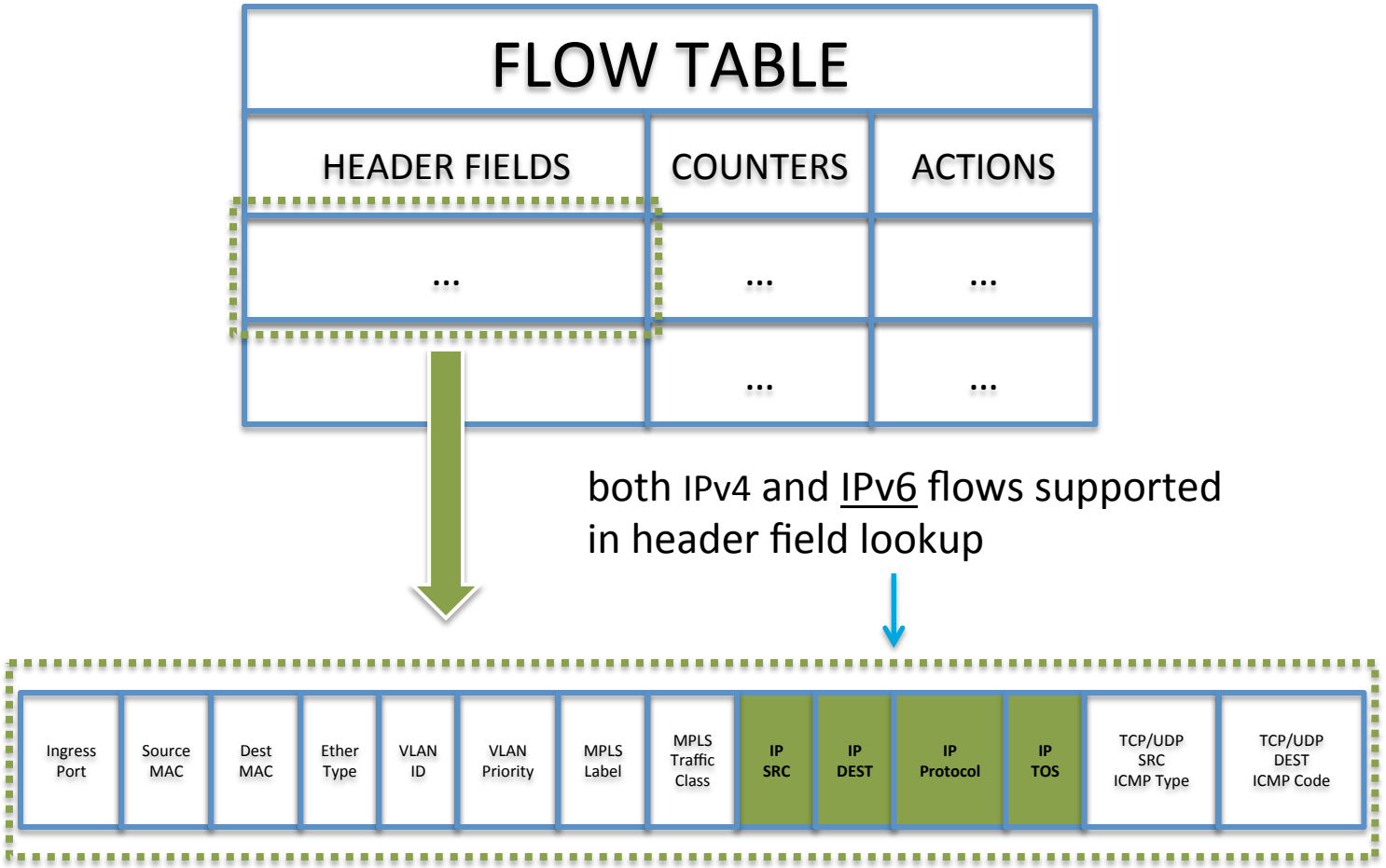
(notable addition)



Openflow v1.1 defines two processing pipeline options
OPENFLOW ONLY and OPENFLOW HYBRID

OpenFlow 1.2

IPv6 supported for lookup in flow table



Openflow 1.3

new functionality

- IPv6 extended header support (MAT)
- Match on MPLS Bottom of Stack (BoS) Bit
- Provider Backbone Bridging (PBB) support
- Duration field added for Statistics
- Support for Tunnel encapsulations (i.e. GRE)
- Ability to disable packet/byte counters on a per flow basis

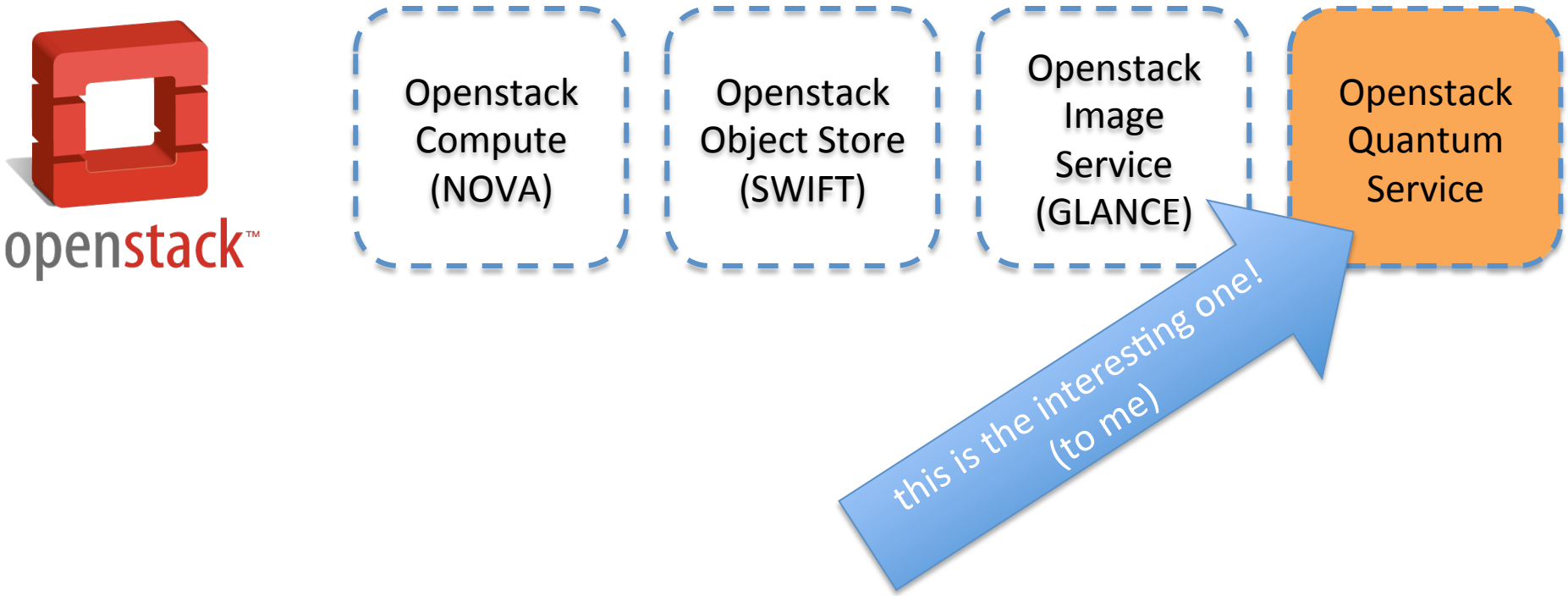
Openflow Protocol Summary

Openflow v1.0	Initial Standard - Most prevalent in the market today
Openflow v1.1	Added support for multiple flow tables Added support for MPLS Defines two operating modes - Hybrid | Pure Openflow
Openflow v1.2	adds support for IPv6
Openflow v1.3	adds support for Rate Limiting | IPv6 extension headers GRE – the version deemed *production ready*
Openflow v1.3.1	adds negotiation TLV's

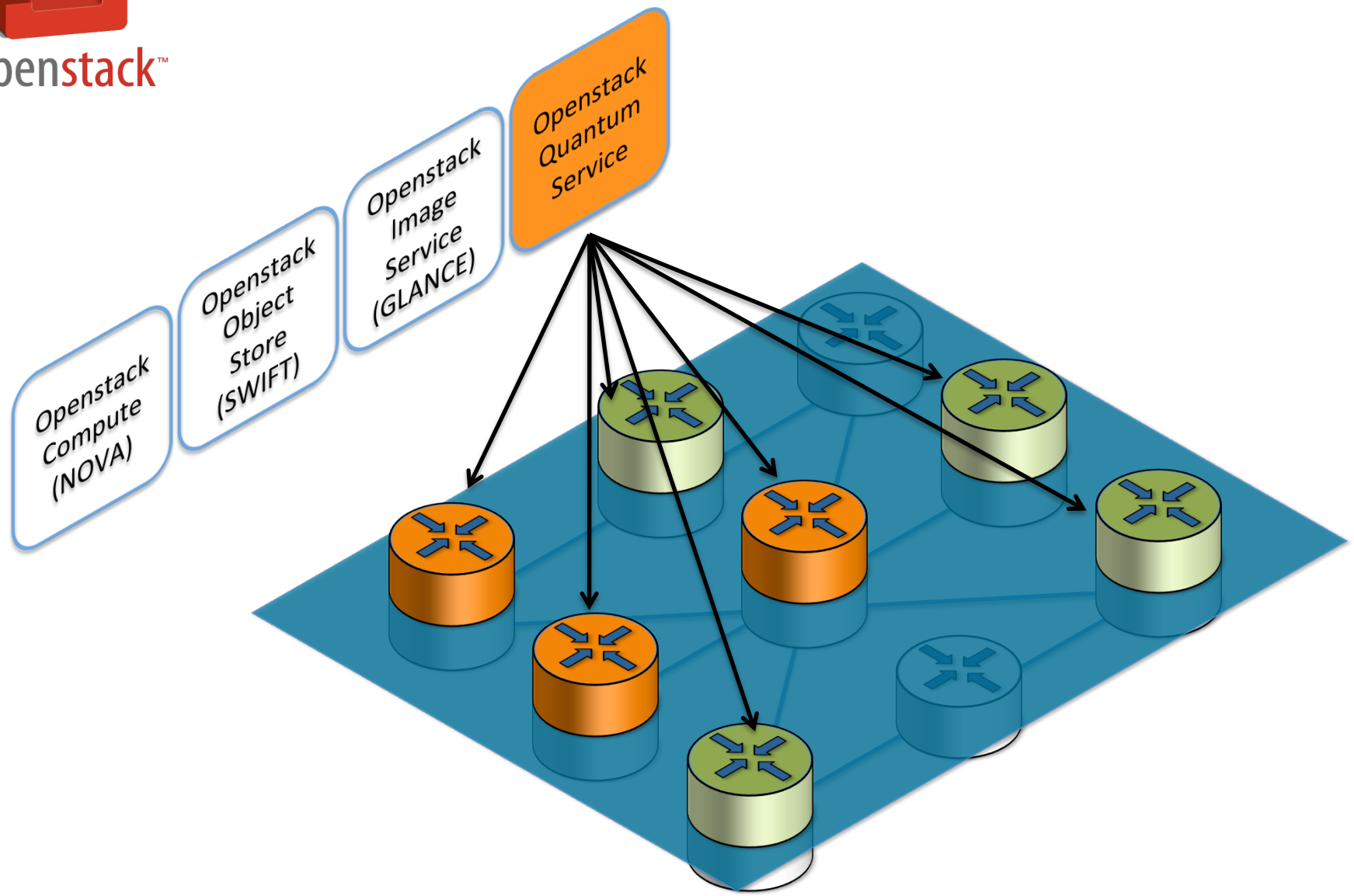
NOW ENTER OPENSTACK

OpenStack

Openstack consists of a number of components



Quantum is used to help manage the network



QUANTUM PLUGIN

OpenFlow and Quantum Plugins

- Plugins which make use of OpenFlow
  - Ryu OpenFlow Controller Plugin
  - NEC OpenFlow Plugin
  - Nicira NVP Plugin
  - Big Switch OpenFlow Controller Plugin
  - Midokura MidoNet Plugin
  - Ruijie Networks Plugin

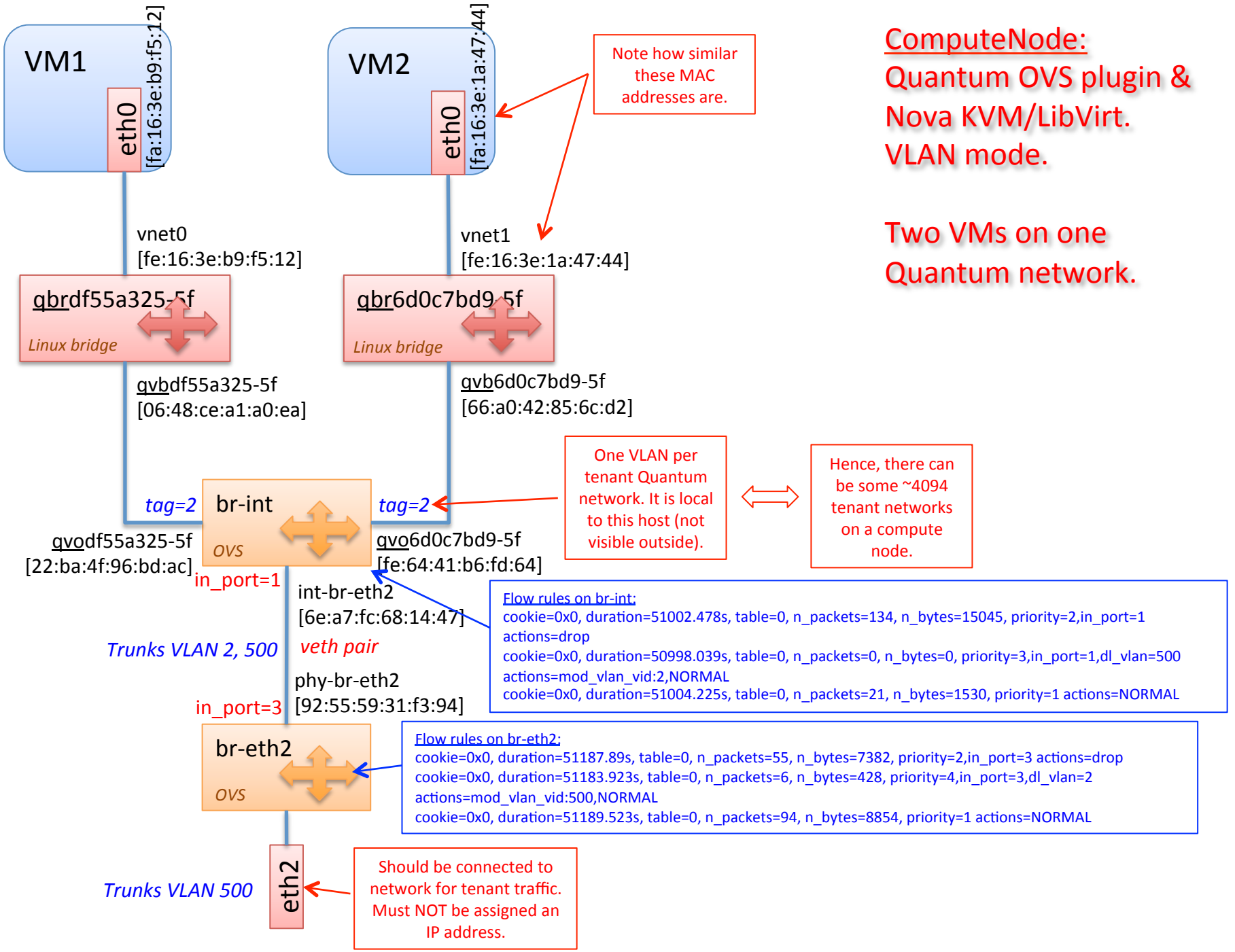
At a high level …

- Most of the OpenFlow innovation has been around Open vSwitch
  - Some hardware vendors with OpenFlow support in HW are trickling in now
- This means everyone is tied to Open vSwitch at the virtual layer …
  - Some wiggle room if you use OVSDB vs. the OVS CLI for example

How Do The Plugins Use OpenFlow?

- Most of them program Open vSwitch
  - Open vSwitch exposes OpenFlow as a northbound API
  - A large number utilize CLI commands from an agent on the host
  - Some make use of the OVSDB protocol to program hosts directly

WHAT DOES THIS ALL LOOK LIKE?


VM1
eth0
[fa:16:3e:b9:f5:12]
VM2
eth0
[fa:16:3e:1a:47:44]
Note how similar these MAC addresses are.
ComputeNode:
Quantum OVS plugin & Nova KVM/LibVirt.
VLAN mode.
Two VMs on one Quantum network.
vnet0
[fe:16:3e:b9:f5:12]
vnet1
[fe:16:3e:1a:47:44]
qbrdf55a325-5f
Linux bridge
qbr6d0c7bd9-5f
Linux bridge
qvbdf55a325-5f
[06:48:ce:a1:a0:ea]
qvb6d0c7bd9-5f
[66:a0:42:85:6c:d2]
tag=2
br-int
OVS
tag=2
One VLAN per tenant Quantum network. It is local to this host (not visible outside).
Hence, there can be some ~4094 tenant networks on a compute node.
qvodf55a325-5f
[22:ba:4f:96:bd:ac]
qvo6d0c7bd9-5f
[fe:64:41:b6:fd:64]
in_port=1
int-br-eth2
[6e:a7:fc:68:14:47]
Trunks VLAN 2, 500
veth pair
phy-br-eth2
[92:55:59:31:f3:94]
in_port=3
Flow rules on br-int:
cookie=0x0, duration=51002.478s, table=0, n_packets=134, n_bytes=15045, priority=2,in_port=1 actions=drop
cookie=0x0, duration=50998.039s, table=0, n_packets=0, n_bytes=0, priority=3,in_port=1,dl_vlan=500 actions=mod_vlan_vid:2,NORMAL
cookie=0x0, duration=51004.225s, table=0, n_packets=21, n_bytes=1530, priority=1 actions=NORMAL
br-eth2
OVS
Flow rules on br-eth2:
cookie=0x0, duration=51187.89s, table=0, n_packets=55, n_bytes=7382, priority=2,in_port=3 actions=drop
cookie=0x0, duration=51183.923s, table=0, n_packets=6, n_bytes=428, priority=4,in_port=3,dl_vlan=2 actions=mod_vlan_vid:500,NORMAL
cookie=0x0, duration=51189.523s, table=0, n_packets=94, n_bytes=8854, priority=1 actions=NORMAL
Trunks VLAN 500
eth2
Should be connected to network for tenant traffic. Must NOT be assigned an IP address.

QUESTIONS?

acknowledgements

- slide 46 – courtesy Bob Melander
- nice openflow artwork and overlay graphics shamelessly ripped off from carl solder & bryan osoro