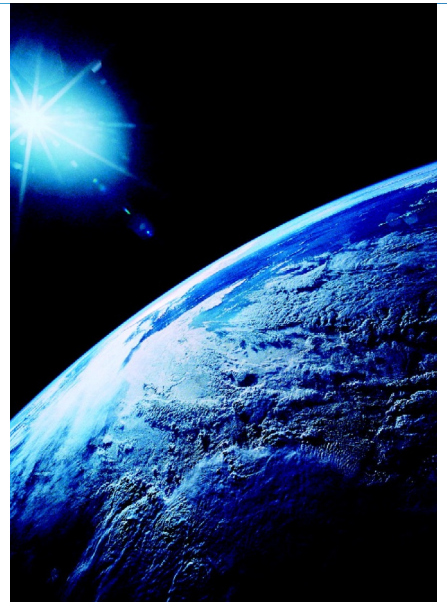
Cisco Connect

Praha, hotel Clarion
10. – 11. dubna 2013

SDN a Cisco DC Networks

Martin Diviš – Cisco

Hynek Pšenička – TO2



Síťová infrastruktura datových center O2 v proměnách času

Cisco Connect 2013 – 11. dubna 2013

Hynek Pšenička, Telefónica Czech Republic

Agenda

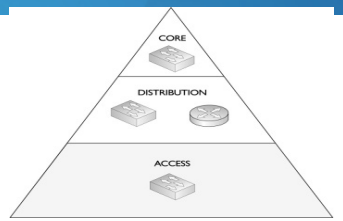
Úvod

Síťová infrastruktura v hostingových centrech Telefónica

Hostingová centra u service providera – specifické vlastnosti

Příklad z praxe

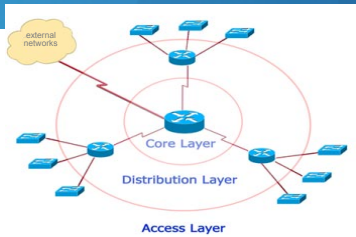
Síťová infrastruktura hostingových centrech TČR



vybudována výhradně na zařízeních firmy Cisco – přináší řadu výhod

- ***neexistence problémů s kompatibilitou mezi zařízení***
- ***podpora funkcionalit napříč celou infrastrukturou***
- **CORE vrstva**
  - ***tvořena dvojicí vysokorychlostních zařízení s nízkou latencí Cisco 7606-S***
  - ***přímé napojení na páteřní MPLS a Internet infrastrukturu pomocí 10GE***
- **DISTRIBUČNÍ vrstva**
  - ***tvořena dvojící modulárních zařízení Cisco Catalyst řady 6500-E***
  - ***propojuje přístupovou vrstvu s vrstvou páteřní (10GE propojení)***
  - ***implementuje síťové politiky a služby:***
    - ***firewall services (FWSM)***
    - ***load-balancing services (ACE module) + SSL off-load (ACE module)***
- **PŘÍSTUPOVÁ vrstva**
  - ***tvořena stacky switchů Cisco Catalyst 3750 / 3750G / 3750E a Nexus 5500 / 2200***
  - ***připojuje zákaznická / koncová zařízení do sítě***
  - ***porty o rychlostech 100Mbps (pouze metalika) / 1Gbps / 10Gbps (metalika i optika)***

Hostingová centra u service providera – specifika



Nezávislost adresace prostředí

- *virtualizace veškerých stavebních prvků infrastruktury*
- *podpora IPv4 i IPv6 (dual-stack)*
- *interconnect pro vzájemná propojení zákazníků*

Integrace různých typů konektivit

- *Internet, MPLS, VPN sítě, privátní sítě a okruhy*
- *připojení alternativních providerů*

Geografická redundance

- *propojení na Layer-2 pro fungování aplikačních a databázových clusterů*
- *vysokorychlostní propojení s nízkou latencí*

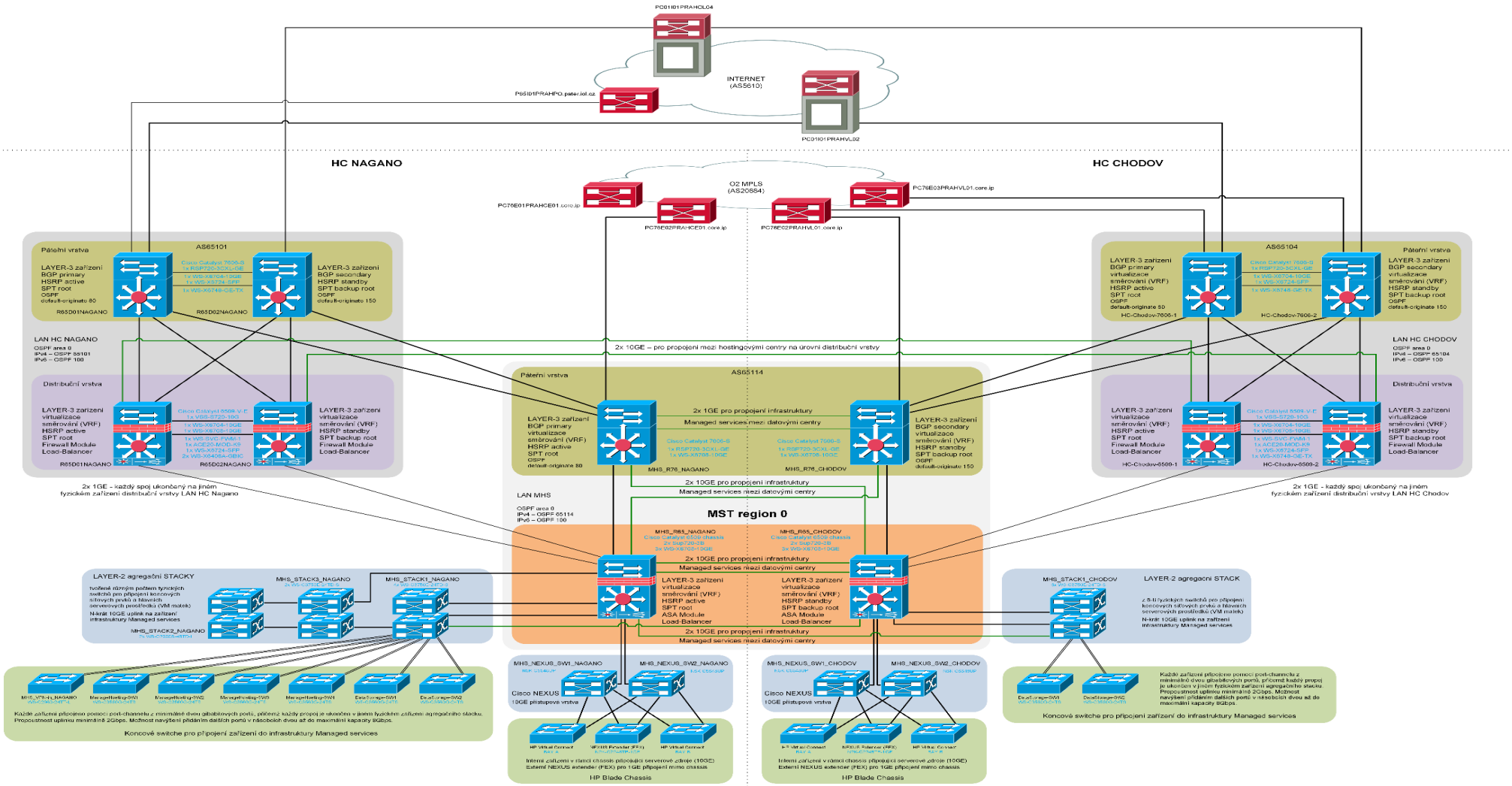
Škálovatelnost řešení

- *opakovatelná řešení s jasně nastavenými parametry služby*
- *jednorázové projekty – infrastruktura na míru podle přání zákazníka*
- *postupný rozvoj infrastruktury v závislosti na zákaznické bázi*

Out-of-band management

- *nezávislý VPN přístup a management síť pro správu prostředí i zákaznických zařízení*

Příklad z praxe – georedudatní infrastruktura v datovém centru



Děkuji za pozornost

SDN v Datových centrech

"…In the SDN 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…"

https://www.opennetworking.org/images/stories/downloads/white-papers/wp-sdn-newnorm.pdf

"…open standard that enables researchers to run experimental protocols in campus networks. Provides standard hook for researchers to run experiments, without exposing internal working of vendor devices……"

http://www.openflow.org/wp/learnmore/

"A way to optimize link utilization in my network, through new multi-path algorithms"

"An open solution for customized flow forwarding control in the Data-Center"

"An open solution for VM mobility in the Data-Center"

"A platform for developing new control planes"

"Develop solutions software speeds: I don't want to work with my network vendor or go through lengthy standardization."

"A way to reduce the CAPEX of my network and leverage commodity switches"

"A way to avoid lock-in to a single networking vendor"

"A solution to build a very large scale layer-2 network"

"A means to do traffic engineering without MPLS"

Diverse Drivers

Common Concepts

Different Execution Paths

"A means to scale my fixed/mobile gateways and optimize their placement"

"A way to define virtual networks with specific topologies for my multi-tenant Data-Center"

"A way to build my own security/ encryption solution, avoiding RSA"

"A solution to build virtual topologies with optimum multicast forwarding behavior"

"A way to configure my entire network as a whole rather than individual devices"

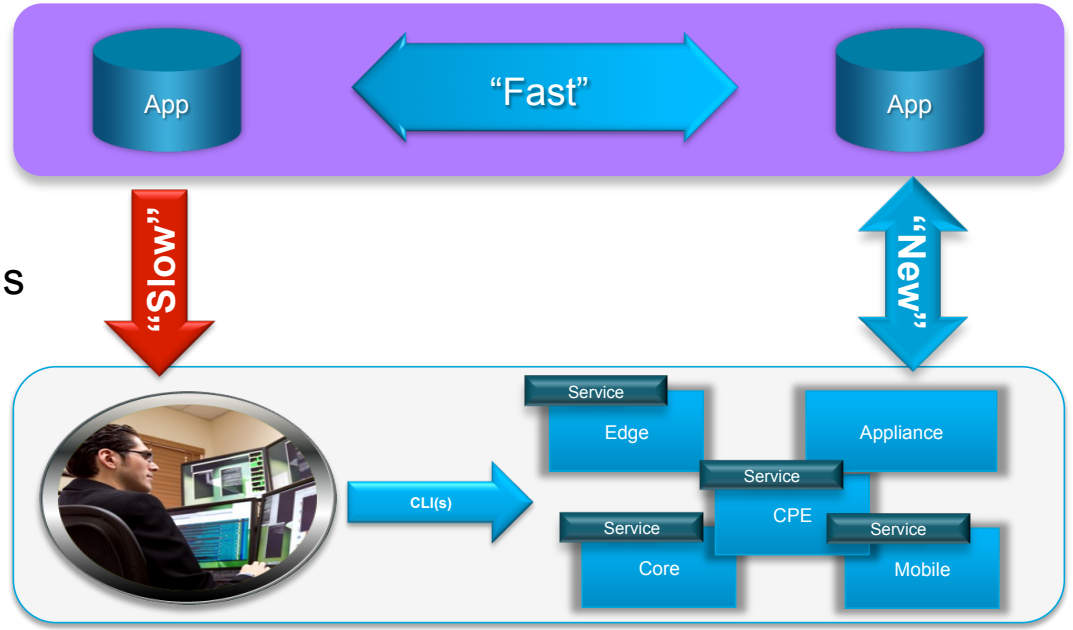
"A way to scale my firewalls and load balancers"

"A solution to get a global view of the network – topology and state"

"A way to distribute policy/intent, e.g. for DDoS prevention, in the network"

Towards Programmatic Interfaces to the Network

- Many Network Applications today:
  - OTT – for speed and agility
  - Avoid network interaction – complex and slow innovation
- New Model for Network Applications
  - Keep speed and agility
  - Full-duplex interaction with the network across multiple planes – extract, control, leverage network state

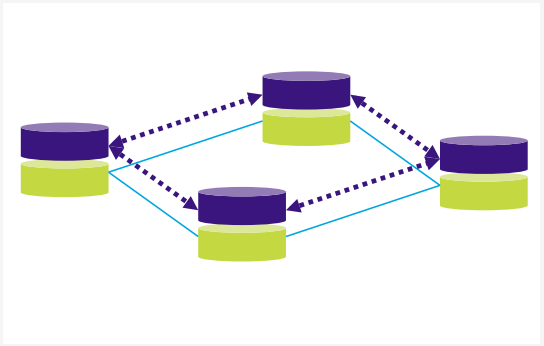


A New Programming Paradigm is Needed

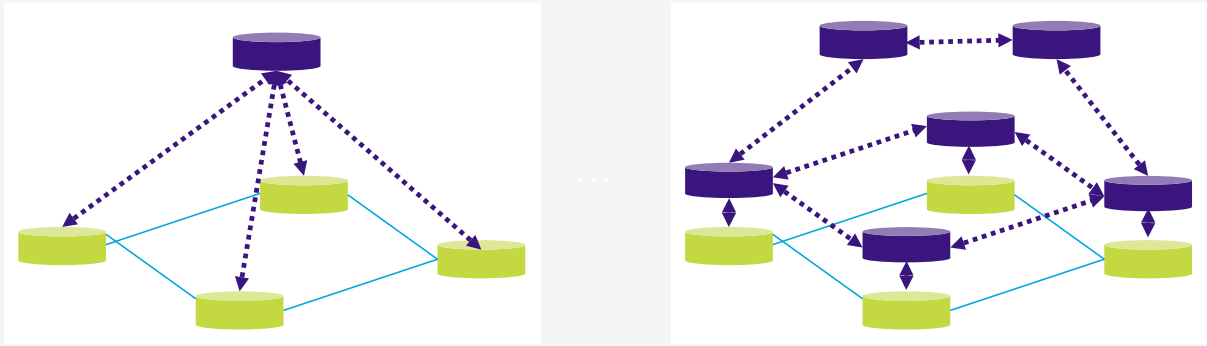
Open Network Environment for SDN

Implementation Perspective: Evolve the Control-Plane Architecture

Traditional Control Plane Architecture



Control Plane Architecture with SDN (Examples)



Enable modularization and componentization of network control- and data-plane functions, with associated open interfaces. This allows for optimized placement of these components (network devices, dedicated servers, application servers) and close interlock between applications and network functions.

Anticipated benefits include: Closely align the control plane with the needs of applications, enable a clear control-/data-plane split, improve performance and robustness, enhance manageability, operations and consistency

Control-plane component(s) Data-plane component(s)

Agents and Controllers:
The OpenFlow Protocol

OpenFlow

Original Motivation

Research community's desire to be able to experiment with new control paradigms

Base Assumption

Providing reasonable abstractions for control requires the control system topology to be decoupled from the physical network topology (as in the top-down approach)

Starting point: Data-Plane abstraction: Separate control plane from the devices that implement data plane

OpenFlow was designed to facilitate separation of control and data planes in a standardized way

Current spec is both a device model *and* a protocol

OpenFlow Device Model: An abstraction of a network element (switch/router); currently (versions <= 1.3.0) focused on Forwarding Plane Abstraction.

OpenFlow Protocol: A communications protocol that provides access to the forwarding plane of an OpenFlow Device

Nothing new under the sun

Starting point of Data-Plane Abstraction & Data- and Control Plane separation isn't new

- Ipsilon Flow Switching

  Centralized flow based control, ATM link layer

  GSMP (RFC 3292)

- AT&T "SDN"

  Centralized control and provisioning of SDH/TDM networks

- A similar thing happened in TDM voice to VOIP transition

  Softswitch → Controller

  Media gateway → Switch

  H.248 → Device interface

- IETF ForCES WG

  Separation of control and data planes

  RFC 3746 (and many others)

- GMPLS, MPLS-TP
- PBB-TE
- Multiple Cisco product examples, e.g.

  Wireless LAN Controller (WLC) – APs

  Nexus 1000V (VSM – VEM)

  Remember RSM (7200 on a stick with Catalyst 5000 as dataplane)?

OpenFlow

Basics

OpenFlow Components

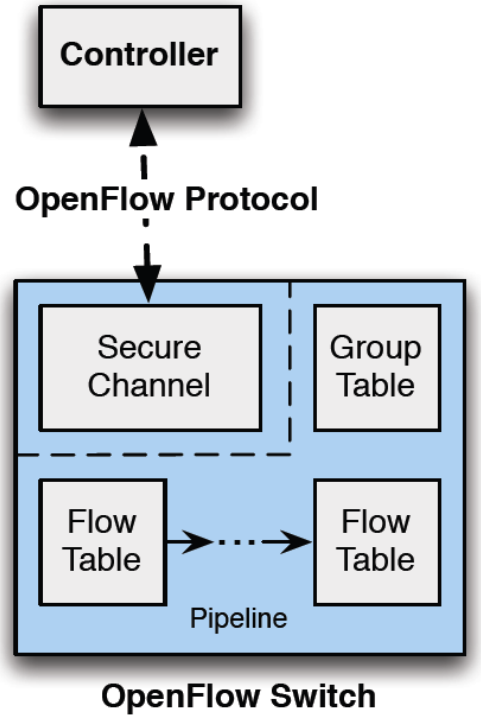
Application Layer Protocol: OF-Protocol

Device Model: OF-Device Model (abstraction of a device with Ethernet interfaces and a set of forwarding capabilities)

Transport Protocol: Connection between OF-Controller and OF-Device*

Observation:

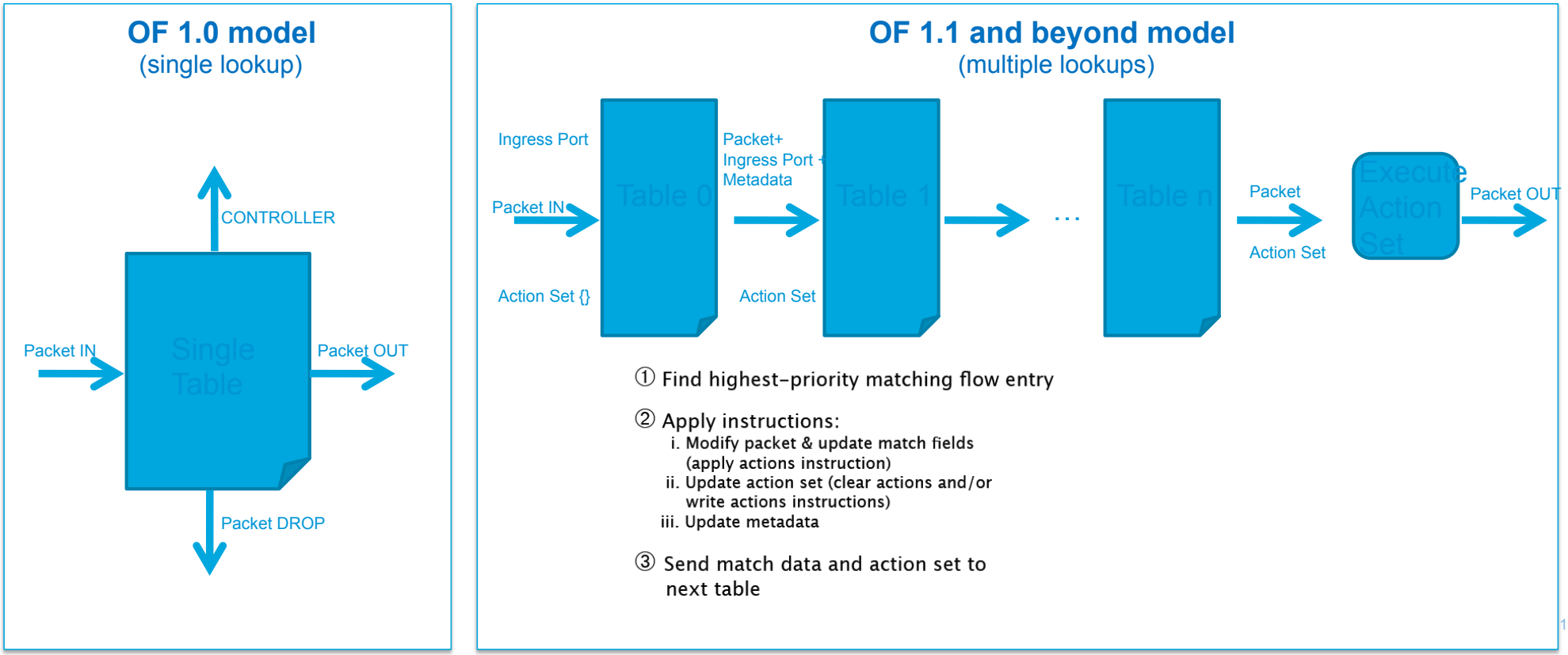
OF-Controller and OF-Device need pre-established IP-connectivity



Source: OpenFlow 1.3.0 specification, figure 1

* TLS, TCP – OF 1.3.0 introduces auxiliary connections, which can use TCP, TLS, DTLS, or UDP.

OF Processing Pipeline



Source: OpenFlow 1.3.0 specification, figure 2

OpenFlow Table

Match Fields (ingress port, packet header, metadata from previous table)

Priority (matching precedence of flow entry)

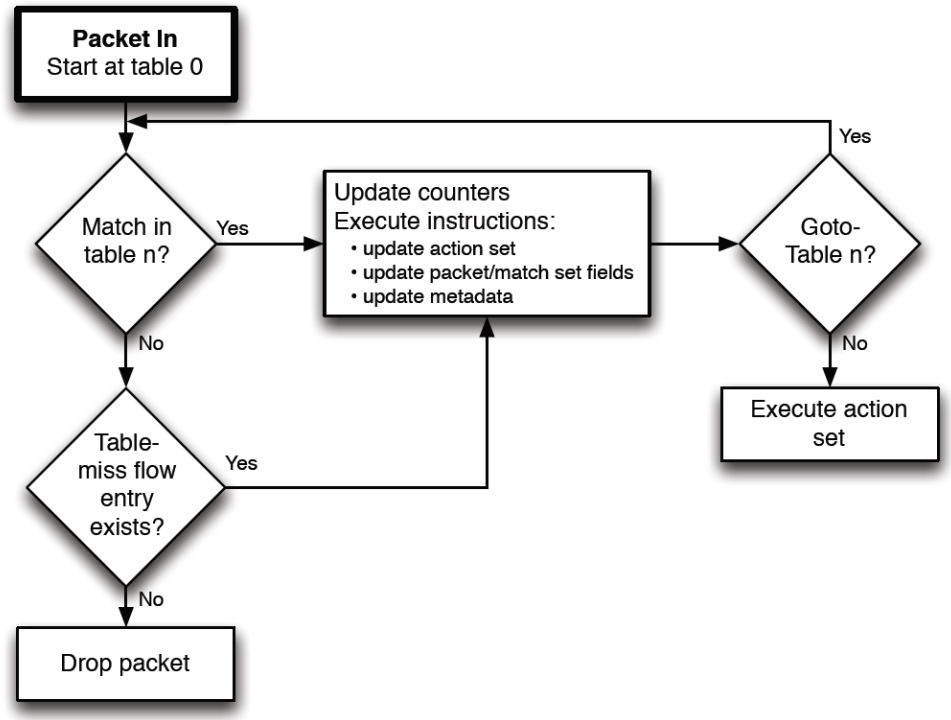
Counters (matching packets)

Instructions (modify action set, pipeline processing)

Timeouts (flow expiry)

Cookie (opaque data chosen by controller)

Packet Flow through an OpenFlow Switch



Source: OpenFlow 1.3.0 specification, figure 3

Required Match Fields

As of OF 1.3

Field	Description
OXM_OF_IN_PORT	Ingress port. This may be a physical or switch-defined logical port.
OXM_OF_ETH_DST	Ethernet source address. Can use arbitrary bitmask
OXM_OF_ETH_SRC	Ethernet destination address. Can use arbitrary bitmask
OXM_OF_ETH_TYPE	Ethernet type of the OpenFlow packet payload, after VLAN tags.
OXM_OF_IP_PROTO	IPv4 or IPv6 protocol number
OXM_OF_IPV4_SRC	IPv4 source address. Can use subnet mask or arbitrary bitmask
OXM_OF_IPV4_DST	IPv4 destination address. Can use subnet mask or arbitrary bitmask
OXM_OF_IPV6_SRC	IPv6 source address. Can use subnet mask or arbitrary bitmask
OXM_OF_IPV6_DST	IPv6 destination address. Can use subnet mask or arbitrary bitmask
OXM_OF_TCP_SRC	TCP source port
OXM_OF_TCP_DST	TCP destination port
OXM_OF_UDP_SRC	UDP source port
OXM_OF_UDP_DST	UDP destination port

OF Match Fields

OFPXMT_OFB_IN_PORT = 0, /* Switch input port. */

OFPXMT_OFB_IN_PHY_PORT = 1, /* Switch physical input port. */

OFPXMT_OFB_METADATA = 2, /* Metadata passed between tables. */

OFPXMT_OFB_ETH_DST = 3, /* Ethernet destination address. */

OFPXMT_OFB_ETH_SRC = 4, /* Ethernet source address. */

OFPXMT_OFB_ETH_TYPE = 5, /* Ethernet frame type. */

OFPXMT_OFB_VLAN_VID = 6, /* VLAN id. */

OFPXMT_OFB_VLAN_PCP = 7, /* VLAN priority. */

OFPXMT_OFB_IP_DSCP = 8, /* IP DSCP (6 bits in ToS field). */

OFPXMT_OFB_IP_ECN = 9, /* IP ECN (2 bits in ToS field). */

OFPXMT_OFB_IP_PROTO = 10, /* IP protocol. */

OFPXMT_OFB_IPV4_SRC = 11, /* IPv4 source address. */

OFPXMT_OFB_IPV4_DST = 12, /* IPv4 destination address. */

OFPXMT_OFB_TCP_SRC = 13, /* TCP source port. */

OFPXMT_OFB_TCP_DST = 14, /* TCP destination port. */

OFPXMT_OFB_UDP_SRC = 15, /* UDP source port. */

OFPXMT_OFB_UDP_DST = 16, /* UDP destination port. */

OFPXMT_OFB_SCTP_SRC = 17, /* SCTP source port. */

OFPXMT_OFB_SCTP_DST = 18, /* SCTP destination port. */

OFPXMT_OFB_ICMPV4_TYPE = 19, /* ICMP type. */

OFPXMT_OFB_ICMPV4_CODE = 20, /* ICMP code. */

OFPXMT_OFB_ARP_OP = 21, /* ARP opcode. */

OFPXMT_OFB_ARP_SPA = 22, /* ARP source IPv4 address. */

OFPXMT_OFB_ARP_TPA = 23, /* ARP target IPv4 address. */

OFPXMT_OFB_ARP_SHA = 24, /* ARP source hardware address. */

OFPXMT_OFB_ARP_THA = 25, /* ARP target hardware address. */

OFPXMT_OFB_IPV6_SRC = 26, /* IPv6 source address. */

OFPXMT_OFB_IPV6_DST = 27, /* IPv6 destination address. */

OFPXMT_OFB_IPV6_FLABEL = 28, /* IPv6 Flow Label */

OFPXMT_OFB_ICMPV6_TYPE = 29, /* ICMPv6 type. */

OFPXMT_OFB_ICMPV6_CODE = 30, /* ICMPv6 code. */

OFPXMT_OFB_IPV6_ND_TARGET = 31, /* Target address for ND. */

OFPXMT_OFB_IPV6_ND_SLL = 32, /* Source link-layer for ND. */

OFPXMT_OFB_IPV6_ND_TLL = 33, /* Target link-layer for ND. */

OFPXMT_OFB_MPLS_LABEL = 34, /* MPLS label. */

OFPXMT_OFB_MPLS_TC = 35, /* MPLS TC. */

OFPXMT_OFP_MPLS_BOS = 36, /* MPLS BoS bit. */

OFPXMT_OFB_PBB_ISID = 37, /* PBB I-SID. */

OFPXMT_OFB_TUNNEL_ID = 38, /* Logical Port Metadata. */

OFPXMT_OFB_IPV6_EXTHDR = 39, /* IPv6 Extension Header pseudo-field */

OpenFlow Actions

Output

Set-Queue* (for QoS)

Drop

Group

Push-Tag/Pop-Tag*

Set-Field* (e.g. VLAN)

Change-TTL*

*Optional

OpenFlow Ports

Physical Ports, Logical Ports, Reserved Ports

Physical Ports == Ethernet Hardware Interfaces

Logical Ports == ports which are not directly associated with hardware interfaces (tunnels, loopback interfaces, link-aggregation groups)

- Can include packet encapsulation. Logical ports can have metadata called “Tunnel-ID” associated with them

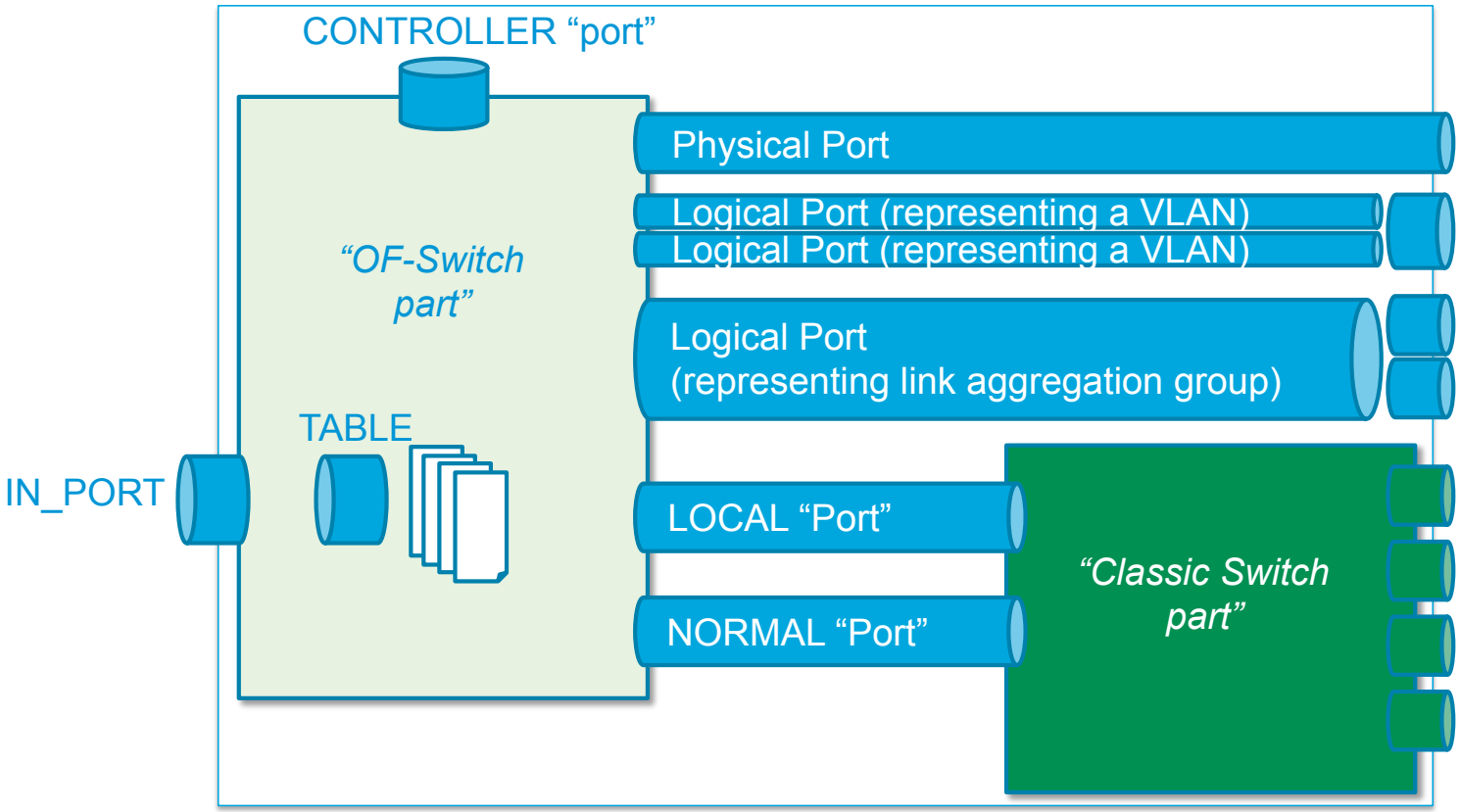
Reserved Ports

- ALL (all ports of the switch)
- CONTROLLER (represents the control channel with the OF-controller)
- TABLE (start of the OF-pipeline)
- IN_PORT (packet ingress port)
- ANY (wildcard port)
- LOCAL* (local networking or management stack of the switch)
- NORMAL* (forward to the non-OF part of the switch)
- FLOOD*

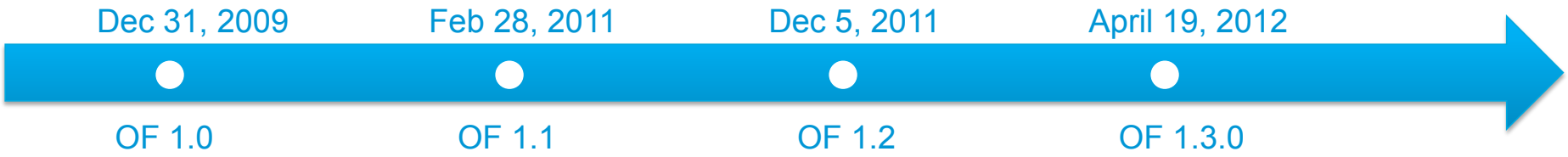
* Optional

OpenFlow Ports

Simplified View



OpenFlow Versions



- Single Table
- L2, IPv4 focused matching

- Multiple Tables
- MPLS, VLAN matching
- Groups: {Any-,Multi-}cast
- ECMP

- IPv6
- Flexible-length matching

- 802.1ah PBB
- Multiple parallel channels between Switch and Controller

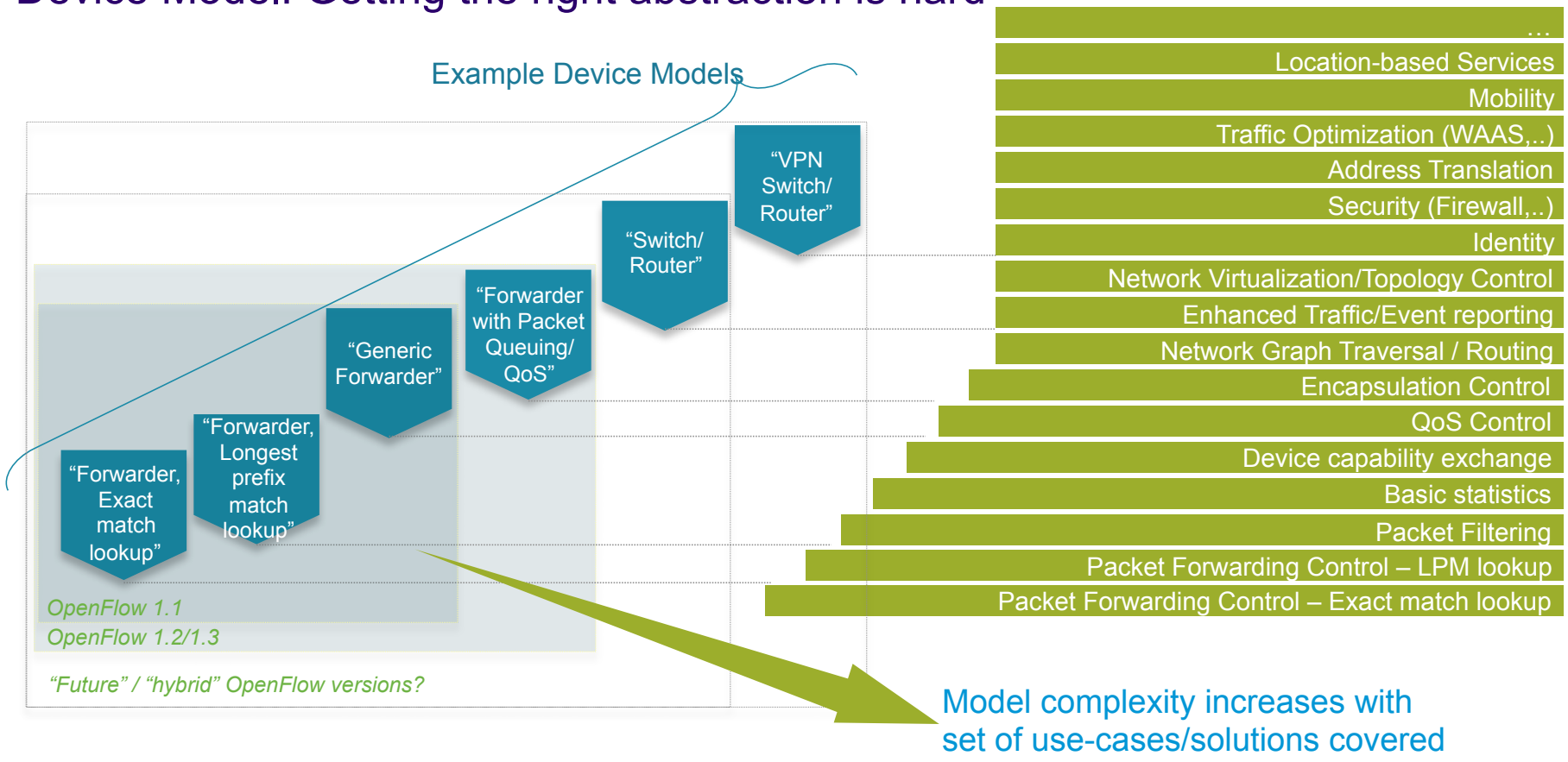
OpenFlow Evolution

Making OF functionally complete

- Topics of ongoing work
  - High availability model for device and controller (state re-sync etc.)
  - Security model (granular access control)
  - L3-forwarding model
  - Enhanced Statistics
  - Management infrastructure (evolution of OF-CONFIG)
  - Testing and certification framework
  - Hybrid device/network deployment capability (Hybrid WG)
- Longer term futures (OF “2.0”)
  - Develop a FPDL (Forwarding Plane Description Language)?

OpenFlow Evolution Challenge

Device Model: Getting the right abstraction is hard



Network Virtualization Overlays

Overview of Roles

Host based overlays	• Centralized Control • Servers perform data encapsulation and forwarding • L2 Centric
Network overlay	• Distributed control • Edge Switches perform data encap and forwarding • L2 and L3 options
Controllers	• Centralized Control / Abstraction • Resource scheduler • Policy based configuration
API	• Methods and Interfaces to implement policy • Provide abstraction of lower layer services

Problem statement:
Overlays for Network Virtualization

- Multi-tenant scale
- VM Mobility
- Span of virtual networks
- Inadequate forwarding tables in switches
- Decoupling logical and physical configuration
- Support communications between virtualized and non-virtualized devices
- Overlay design characteristics

http://tools.ietf.org/html/draft-narten-nvo3-overlay-problem-statement-01

Use-cases

- Workload anywhere –

  Optimally use server resources

- Segmentation –

  support overlapping network addresses

- Simplify management

  The physical network becomes a static one-time config (L3 routing protocol, small number of VLANs facing servers in a Rack)

  Software instantiates network services on x86 servers

OVERVIEW OF ENCAPSULATIONS VXLAN, NV-GRE, STT

Overlay Encapsulations

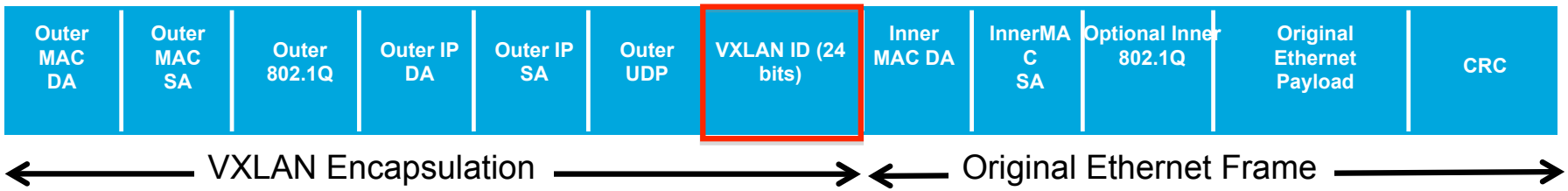
- These are stateless tunneling mechanisms
  - No static tunnel setup required on server
  - Frame format recognized by end hosts and treated as a tunneled frame (encap/decap)
- Ethernet Frames are encapsulated into an IP frame format
  - The physical network uses the outer IP/MAC header to forward tunneled traffic
  - The physical network is not aware of VM identities or the virtual networks
  - Design to accommodate L3 ToR and remove STP/VLAN constraints of physical network
- New control logic for learning and mapping VM identity (MAC address) to Host identity (IP address)

nvGRE

- IETF standard co-authored by Microsoft, Emulex and HP
- L2 Ethernet encapsulation in GRE/IP
- 24bit Tenant Network ID (TNI) used to ID a virtual network segment
- Take advantage of existing GRE standard
- Challenge w/ entropy – Require HW to use TNI in hash for ECMP
- PA = Provider Address
- CA = Customer Address

VxLAN

- IETF co-authored by VMware, Cisco, Arista, Broadcom and RedHat
- L2 Ethernet encapsulation in UDP/LISP
- 24bit virtual segment ID to identify virtual network
- Inner 5-tuple hashed to new source port in UDP header
- Control is “network assisted” for MAC learning
  Mcast (*,g) state is maintained on physical network
- Multicast used for multi-destination frames
- v1.5 removes multicast dependency



Stateless Transport Tunneling

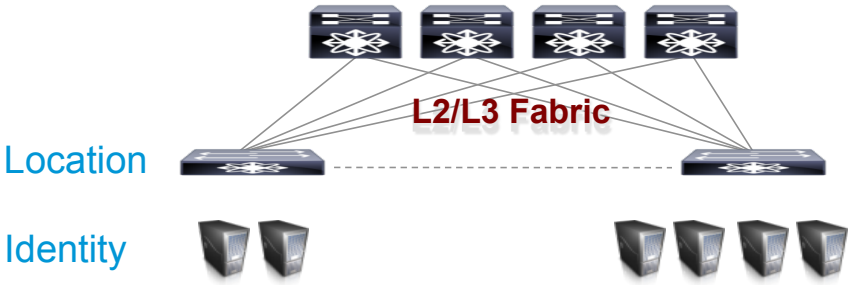
- IETF standard authored by Nicira
- L2 Ethernet frame encapsulated in IP

  Leverage TCP header to take advantage of NIC TSO/LRO
- 64bit Context to identify virtual network segment
- Inner 5-tuple hashed to new source port in STT header
- Control logic not defined in draft
- Multicast used for multi-destination (depending on implementation Mcast on the physical network may not be required)

STT was designed to take advantage of TSO and LRO engines in existing NICs today. With STT, it is possible to tunnel at 10G from the guest while consuming only a fraction of a CPU core.

Cisco Data Center Fabric & Technologies

Location Identity Separation @ scale



- Location reachability determined by traditional routing mechanisms in the Fabric
- Identity is mapped to location addresses
- All these Cisco technologies leverage Location/Identity Mapping

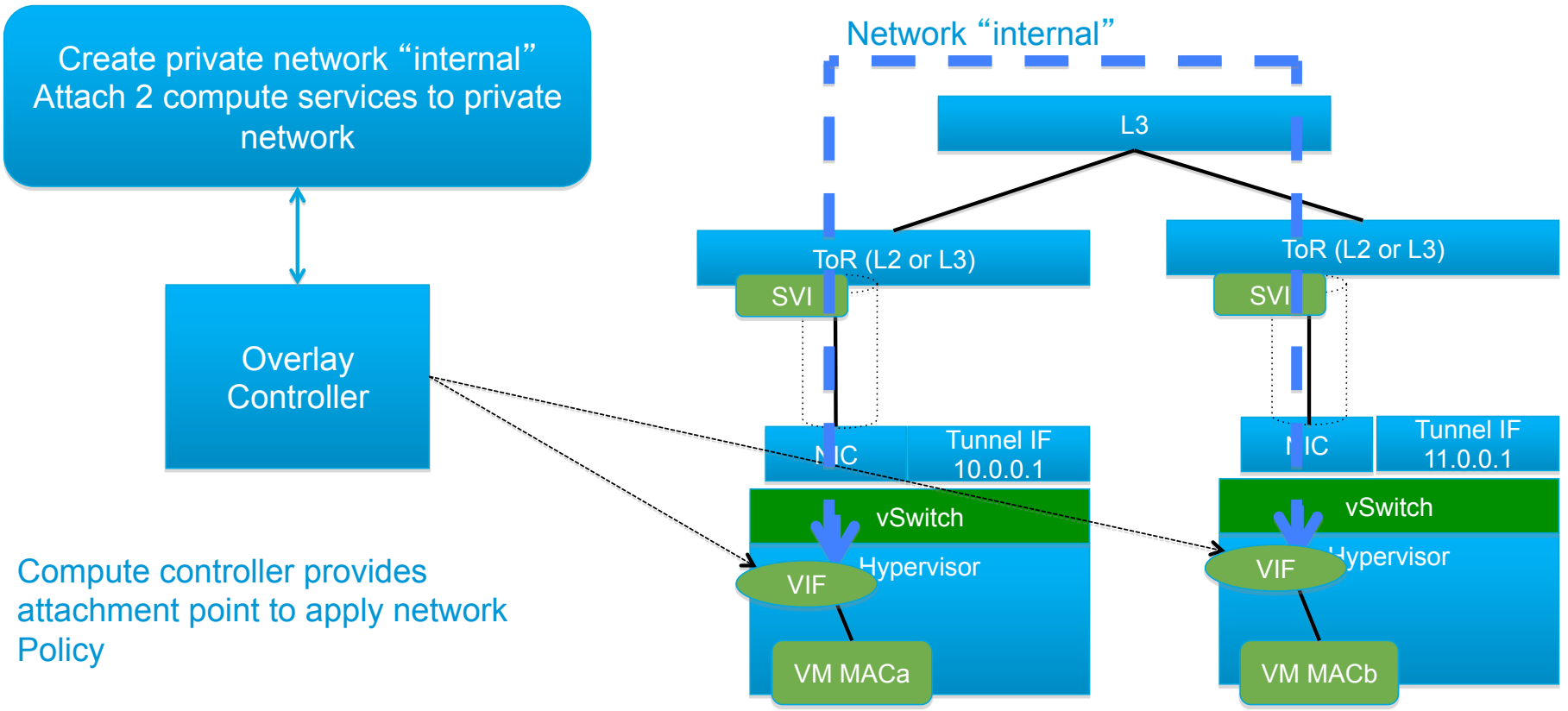
	FabricPath / TRILL	VXLAN	OTV	LISP
Location	Switch-ID (IS-IS)	IP address (IP protocols)	IP address (Unicast/Mcast)	IP address (RLOC)
Identity	Client MAC (Flooding)	Client MAC (Flooding)	Client MAC (IS-IS)	Client IP (EID)

FORWARDING MODELS (LEARNED VS PROVISIONED)

Host Overlay forwarding models

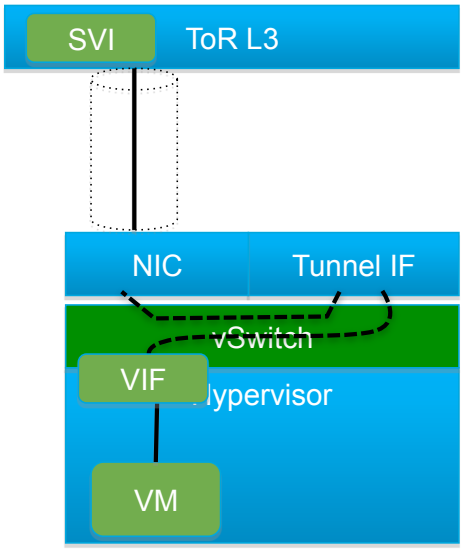
- Network assisted
  - flooding over Mcast to learn MACs
- SW Programmed
  - Track VM attach at the Hypervisor
  - External controller programs VM MAC to Host IP mapping and push down to vSwitches
- Multi-destination
  - Broadcast frames, Multicast frames,
  - Multicast transport on the physical network
- Tenant ID
  - Each virtual network supports its own address family
  - A single Tenant can have multiple virtual networks for a workload (3-tier Web, App, DB)

Host Overlay Policy instantiation – OpenStack/Quantum



Host Overlay forwarding
VM Attach

- VM attaches to virtual switch
- Hypervisor API update to vSwitch
- Policy Applied to virtual port via VIF
- Tunnel IF maintains bindings between remote VM MAC and Host IP



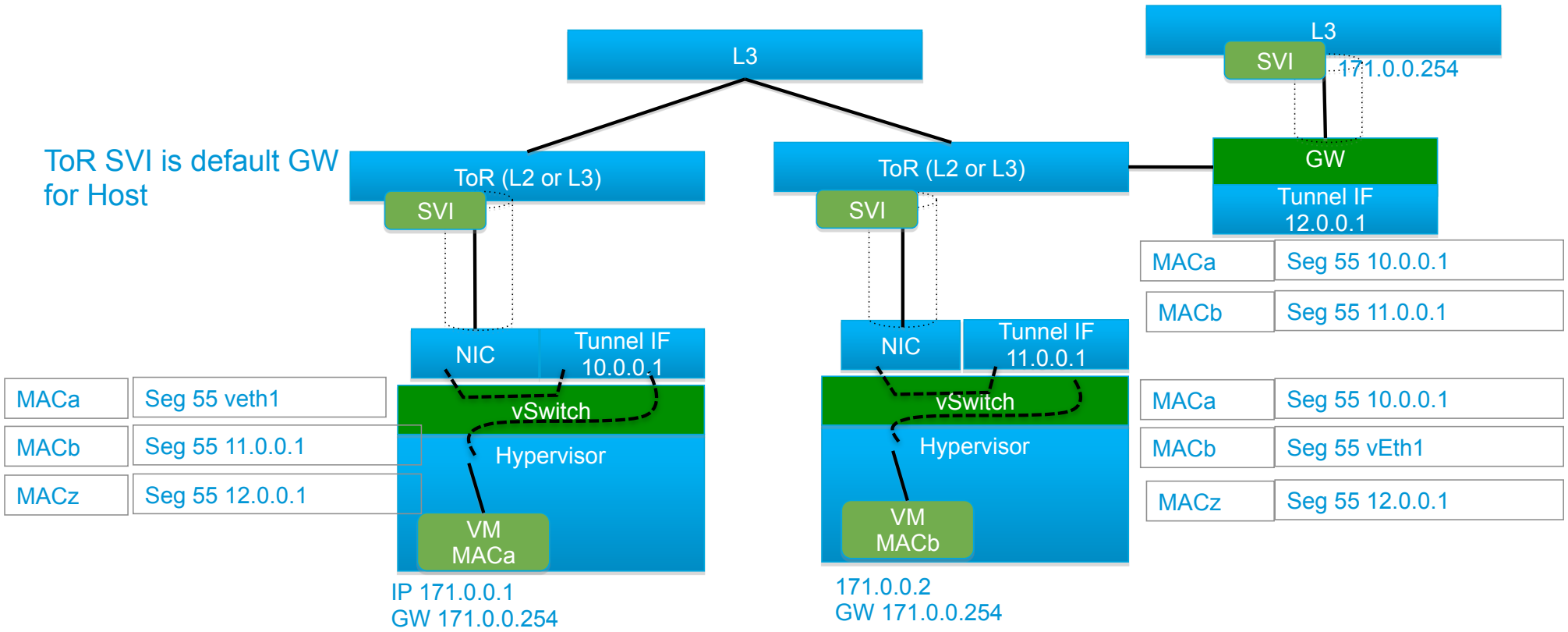
Performs Tunnel encap/decap

Mapping table for remote MACs and remote Tunnel IP

vSwitch learns local MACs and MAC of ToR SVI

Traffic destined to remote MAC passed on to Tunnel IF

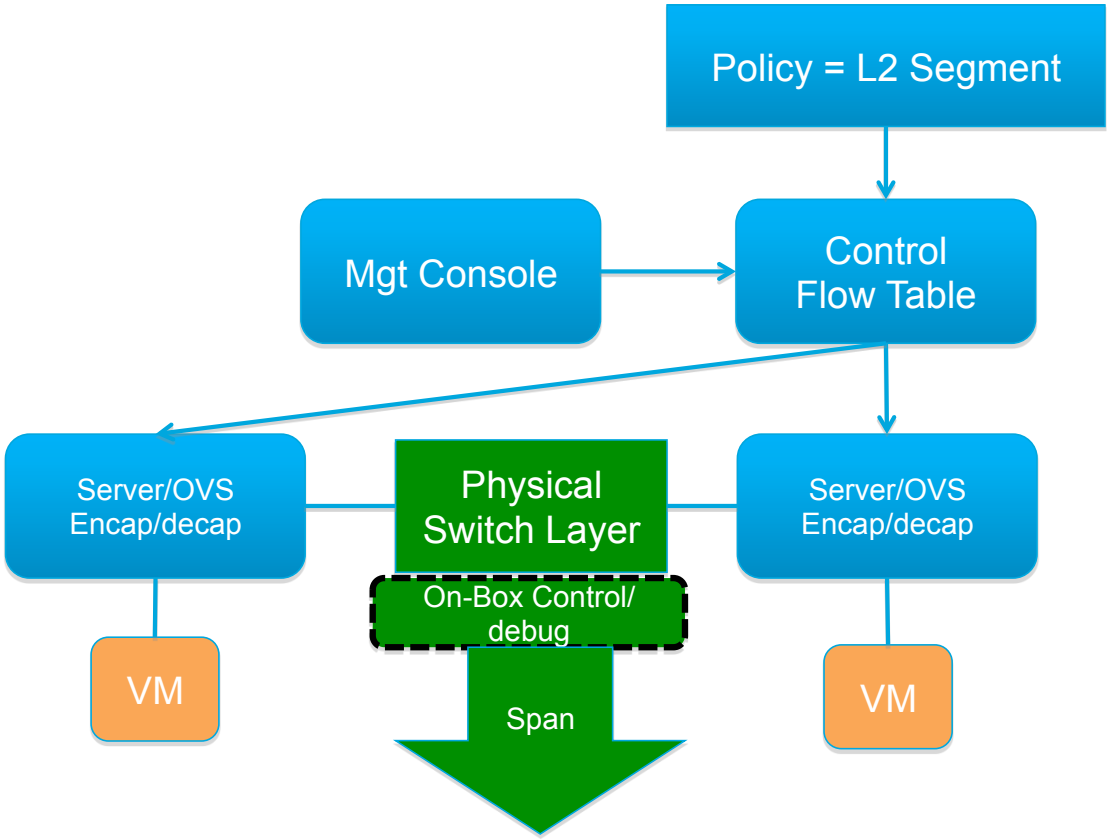
Learning complete



Operational Challenges

- How do you troubleshoot these overlay networks
- How do you debug overlays (remember ATM LANE, VPLS, …),

- Flow Table accuracy
- Data-path integrity
- Data-path performance
- Packets punted to Controller



Cisco's solution & roadmap

Cisco Vision: Exposing The Entire Network Value

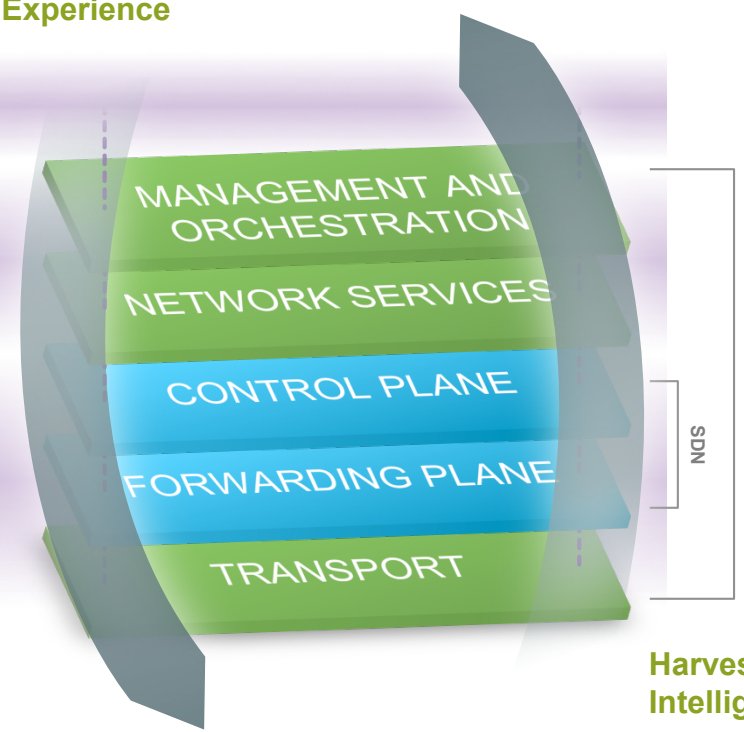
Programmatic Control across Multiple Network Planes

Program Policies for Optimized Experience

Application Developer Environment

Analysis and Monitoring, Performance and Security

Network Elements and Abstraction



Harvest Network Intelligence

Any Object

- Switch/Router
- ASIC
- Network Fabric
- Compute

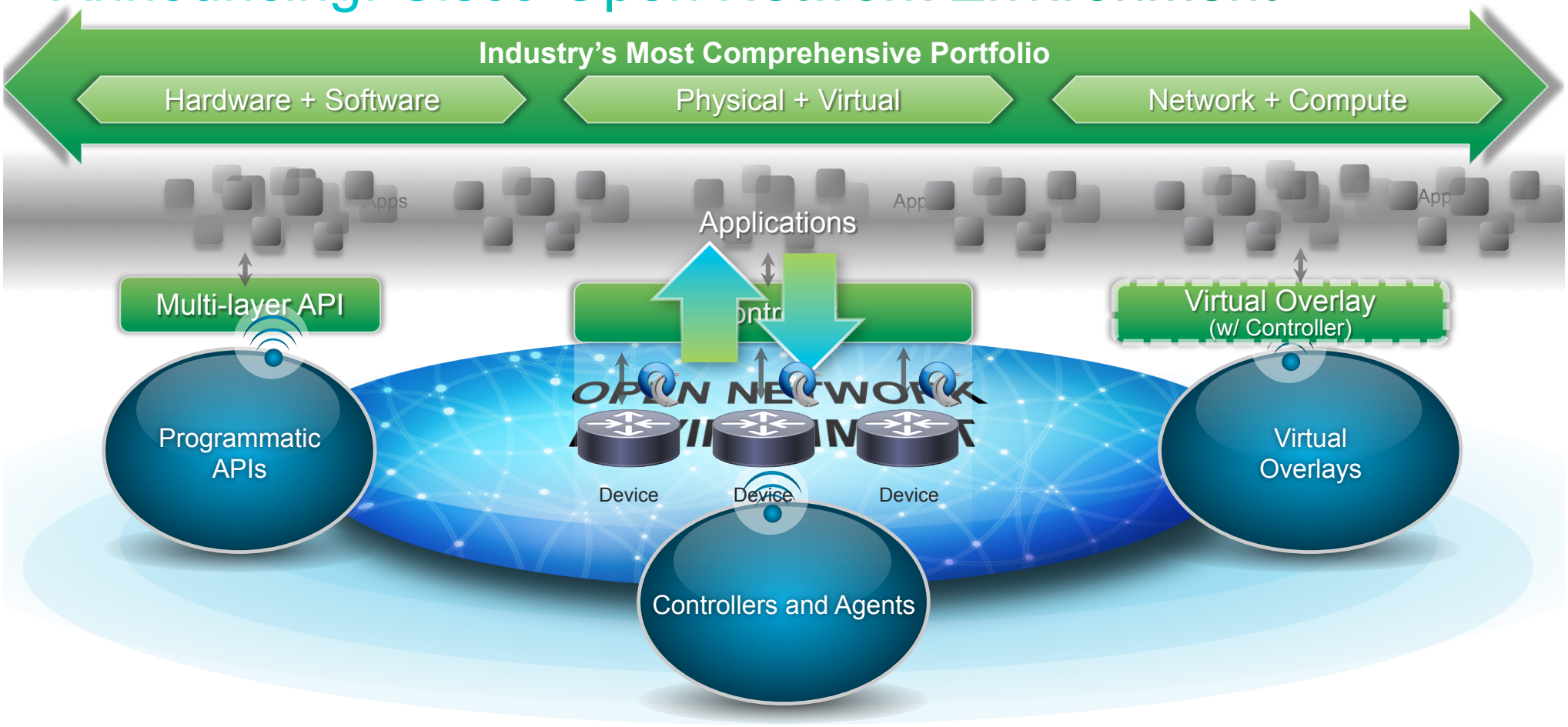
Any Service

- Cloud
- Collaboration
- Video
- Security
- Mobility

Any Layer

- L1-7
- Control/Data Plane
- Hardware/Software
- ASICs/OS

Announcing: Cisco Open Network Environment



Cisco Innovations

onePK Developer Kit

- Complete developer's kit through Cisco developer network
- Rapidly develop test and deploy Applications
- Phased availability across IOS, IOS-XR and NX-OS platforms

Programmatic APIs

Controllers + Agent Support

- Controller SW for experimentation on production networks
- Advanced Topology Independent Forwarding with integrated network slicing management
- OpenFlow experimental support on select Cisco platforms

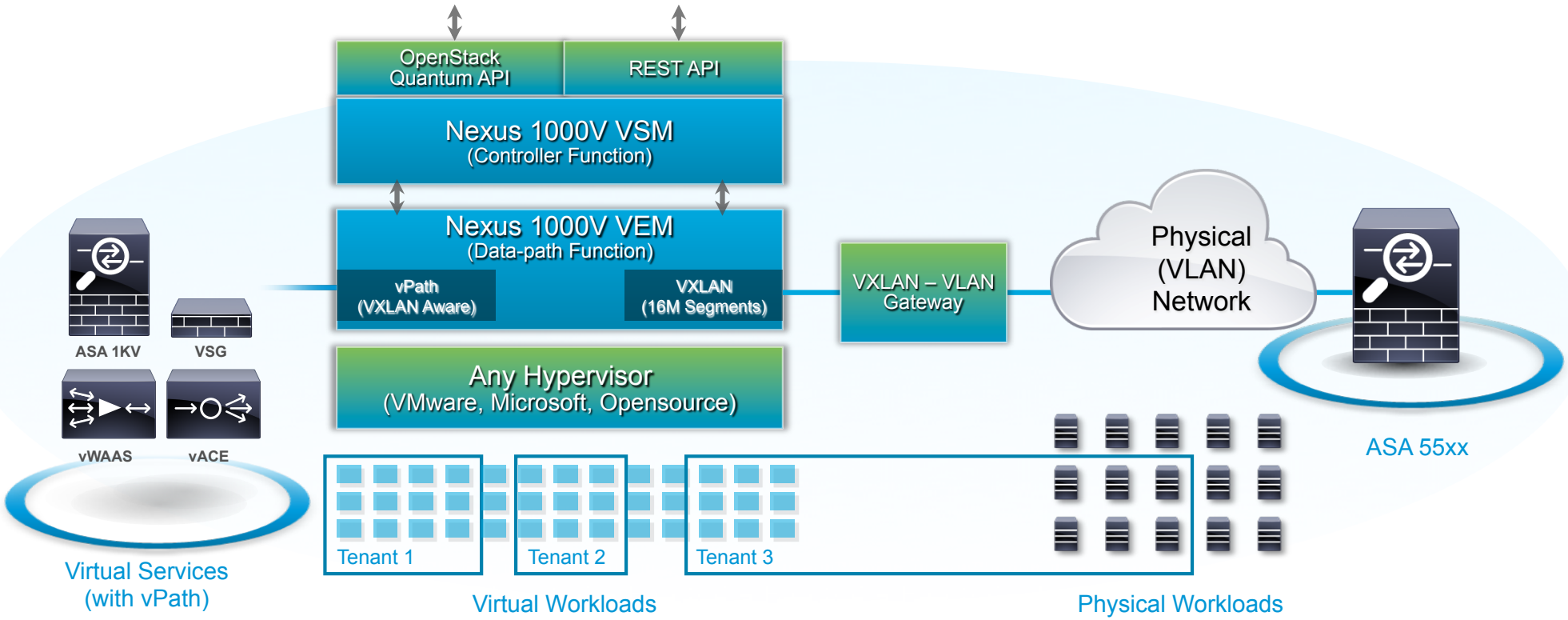
Controllers and Agents

Overlay Network Solutions

- Multi-hypervisor support on Nexus 1000V (incl. OpenSource hypervisor)
- OpenStack and REST APIs on N1KV for rapid tenant provisioning
- VXLAN-VLAN gateway (for bridging traditional environments)
- Virtual or Physical Network Services

Virtual Overlays

SDN Nexus1000v Architecture



Consistent experience across physical and virtual environments

Děkujeme za pozornost.