Cisco live!

What You Make Possible

TOMORROW
starts here.

CISCO

OpenFlow and SDN: Today and Tomorrow

BRKARC-2662

TOMORROW starts here.

Abstract

- Today there is definitely a lot of buzz around OpenFlow and Software Defined Networks (SDN) which in some minds equate to one in the same. In addition, there are additional terms that form part of the conversation like Openstack, Virtual Overlays, Network virtualisation etc. The current market conversation has loose semantics mixed in with hyperbole and hearsay that all make the work of a network practitioner and our customers harder. This session will cut through the terminology and offer a framework to both understand these trends and distill the applicability using a use case lens. We will then deep dive into Cisco's participation and leadership in this space and an assessment of current state We will then spend time to understand the broader construct these technologies fit into and Cisco's strategy in this space - Cisco ONE.

Related Sessions

Session ID	Title	Presenter
BRKSPG-2662	Software Defined Networking (SDN) Architectures and Implications	Dave Ward, VP SP Chief Architect & CTO, Cisco
BRKARC-2663	Software Defined Networking and Use Cases	Ken Hook, Sr. Product Manager, ONE/SDN Marketing, Cisco

Agenda

- Demystifying SDN
- Definitions
- Driving Factors
- Cisco ONE Strategy
- Programmability in production networks

OPEN NETWORKING FOUNDATION

"…In the SDN architecture, the control and data planes are decoupled, network intelligence and state are logically centralised, and the underlying network infrastructure is abstracted from the applications…"

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

OpenFlow™

"…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 optimise link utilisation in my network enhanced, application driven routing"

"An open solution for customised flow forwarding control in and between Data Centres"

"A platform for developing new control planes"

"An open solution for VM mobility in the Data Centre"

"A solution to automated network configuration and control"

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

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

"A means to get assured quality of experience for my cloud service offerings"

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

"A means to do traffic engineering without MPLS"

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

Diverse Drivers

Common Concepts

Different Execution Paths

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

"A way to scale my firewalls and load balancers"

"A way to optimise broadcast TV delivery by optimising cache placement and cache selection"

"A way to build my own security/encryption solution"

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

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

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

Simplified Operations – Enhanced Agility – New Business Opportunities

Classes of Use-Cases

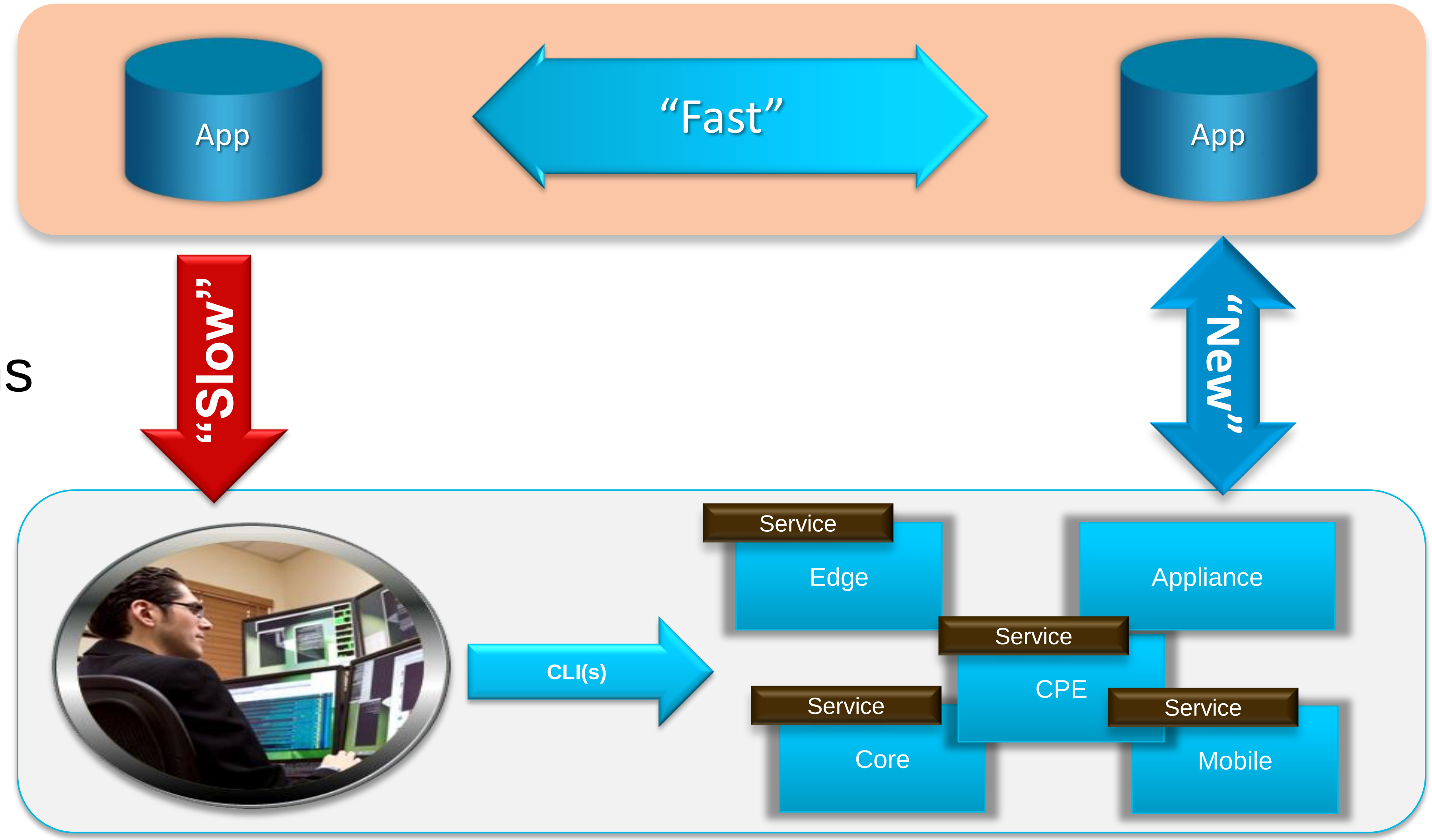
“Leveraging APIs and logically centralised control plane components”

- Custom Routing (incl. business logic) Online Traffic Engineering
- Custom Traffic Processing (Analytics, Encryption)
- Consistent Network Policy, Security, Thread Mitigation
- Virtualisation and Domain Isolation (Device/Appliance/Network)
- Federating different Network Control Points (LAN-WAN, DC-WAN, Virtual-Physical, Layer-1-3)
- Automation of Network Control and Configuration (Fulfillment and Assurance)

Towards Programmatic Interfaces to the Network

Approaching Today's Application Developer Dilemma

- 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

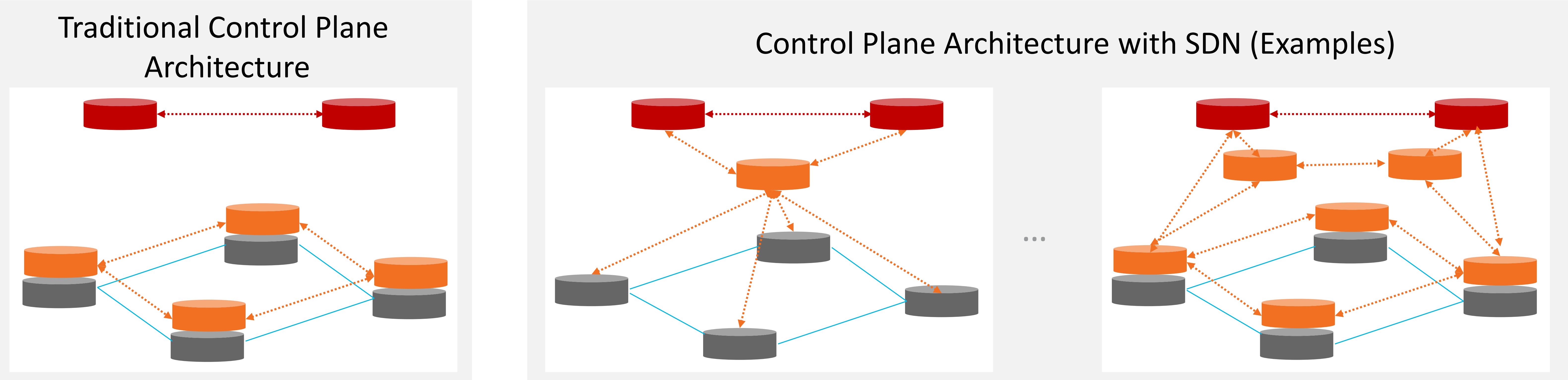
Re-assessing the Network Control Architecture Evolving Design Constraints on the Control Plane

- Classic generic networks
  - Operate without communication guarantees
    - A distributed system with arbitrary failures, nearly unbounded latency, and highly variable resources on each node in the system
  - Compute the configuration/forwarding-state of each physical device and keep the information up to date as conditions change
    - Change of conditions typically detected by the network elements themselves
  - Operate within given network-level protocol (IP, Ethernet, …)
- Domain specific networks (e.g. Data Centre, SP-Access/Agg,..)
  - Specific qualities of these networks relax or evolve network design constraints:
    - Examples: Well defined topologies; little variety in network device-types; no arbitrary changes in connected end-hosts (change always an outcome of provisioning action),…
  - Independence of network-level protocol (combined L2, L3 service delivery,…)

Different solutions for different domains: DC != WAN, TOR != PE

Towards the Open Network Environment for SDN

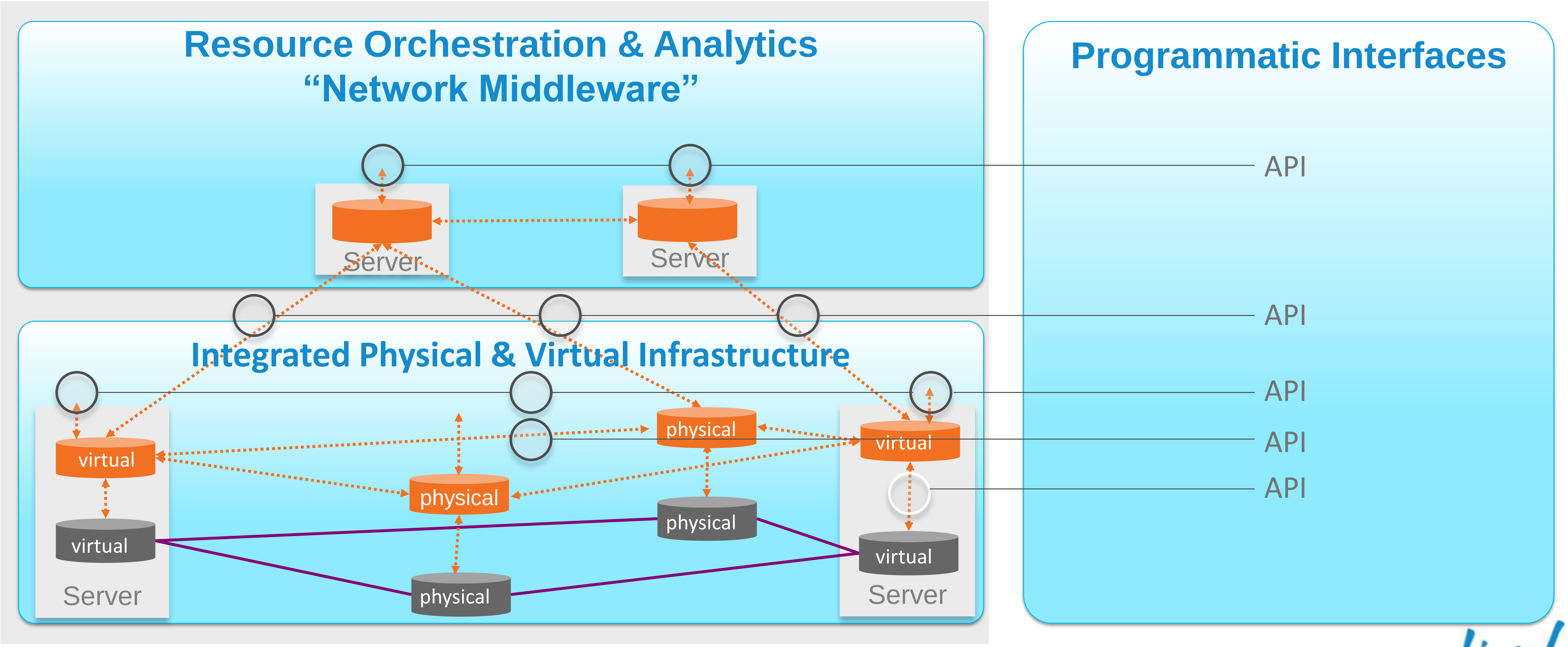
Implementation Perspective: Evolve the Control-Plane Architecture



- Enable modularisation and componentisation of network control- and data-plane functions, with associated open interfaces: Allow for optimised placement of these components (network devices, dedicated servers, application servers) and close interlock between applications and network functions; combining the benefits of distributed and centralised control plane components
- Anticipated benefits include: Closely align the control plane with the needs of applications, enable componentisation with associated APIs, improve performance and robustness, enhance manageability, operations and consistency – while maintaining benefits of standardised distributed control planes.

Application components — Control-plane component(s) — Data-plane component(s)

Open Network Environment
Approaching a Definition



Open Network Environment

Approaching a Definition

Resource Orchestration & Analytics
"Network Middleware"

"Controllers and Agents"

Integrated Physical & Virtual Infrastructure

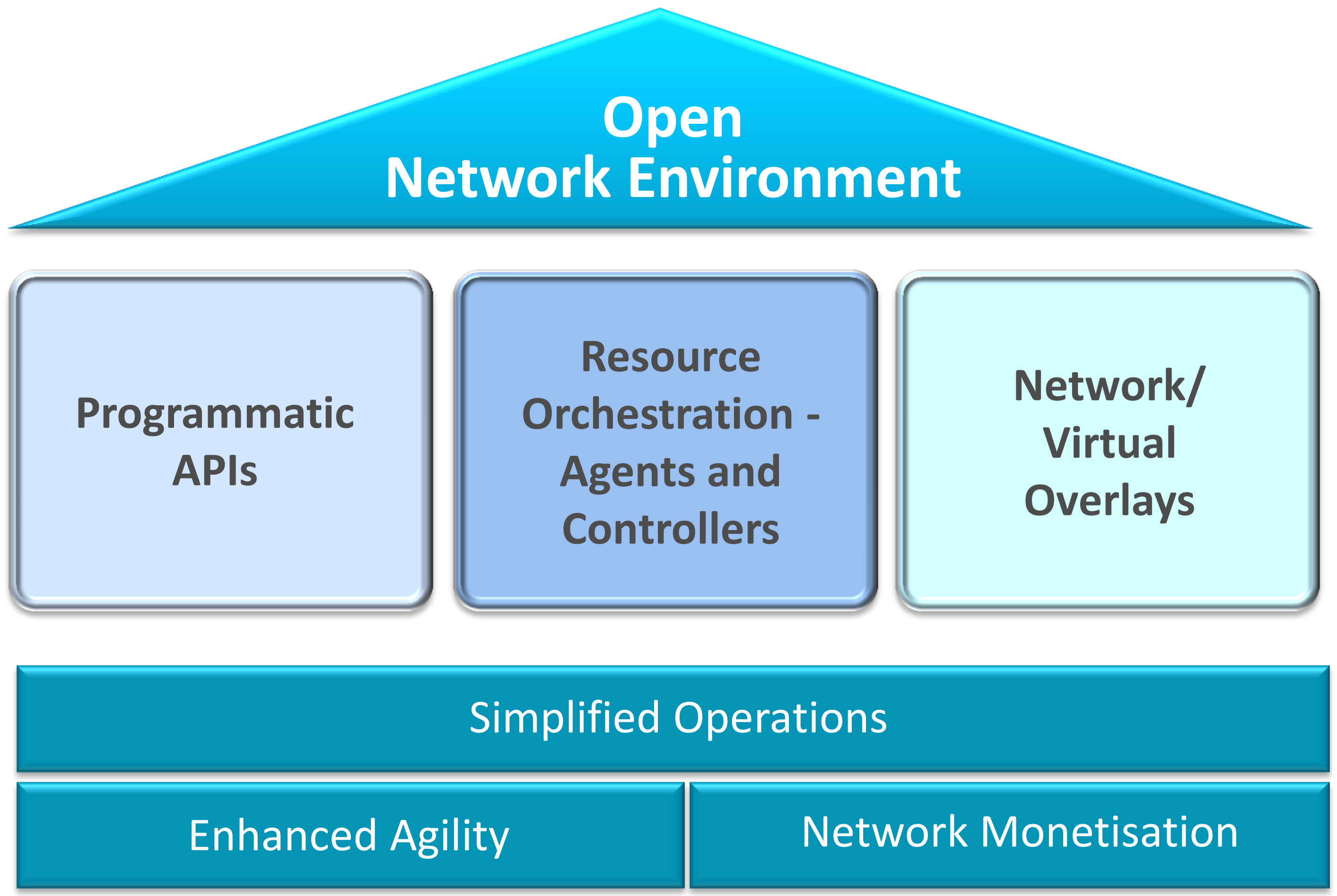
"Network/Virtual Overlays"

Programmatic Interfaces

"Platform APIs"

Open Network Environment

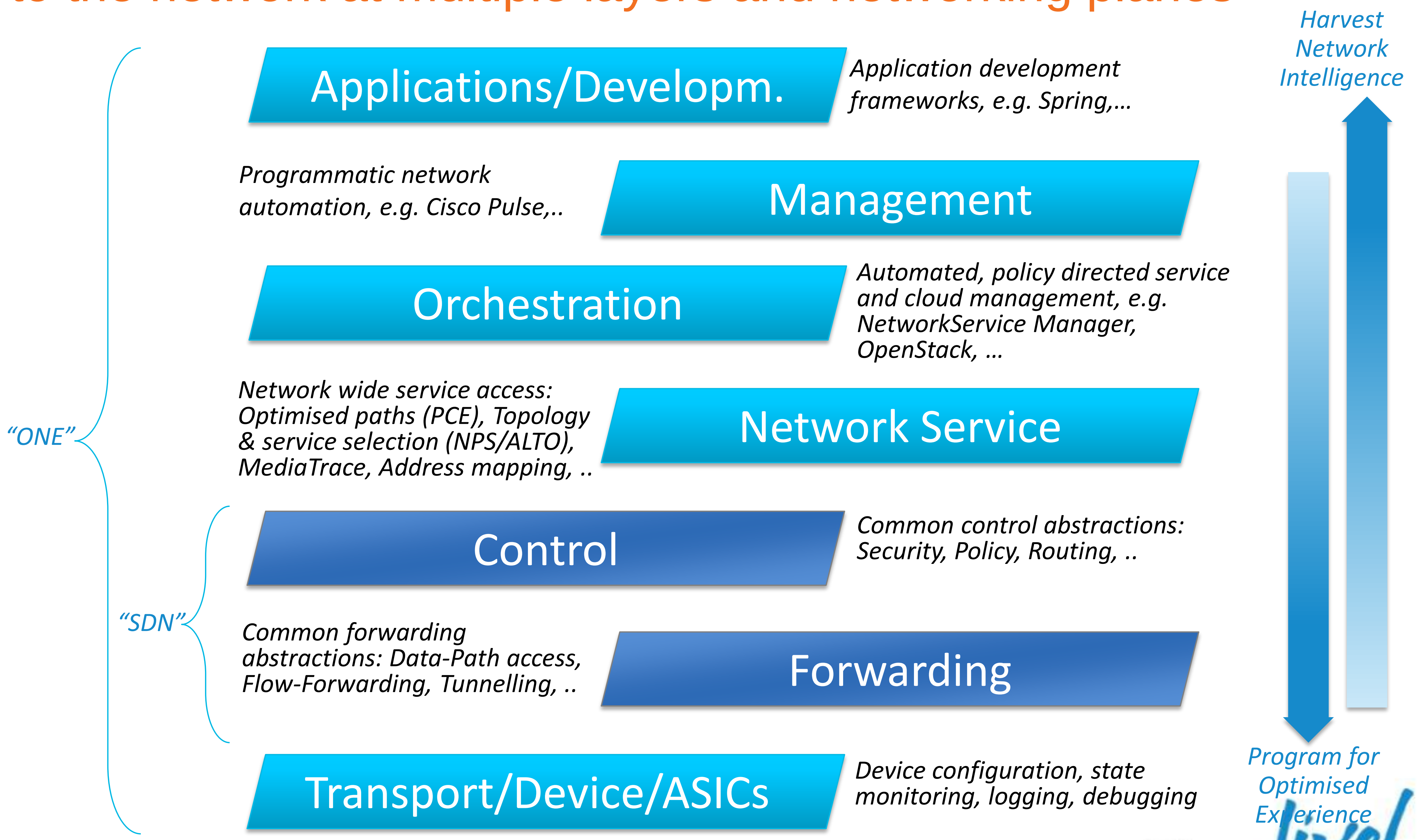
- Open Network Environment – Complementing the Intelligent Network
  - *Preserve what is working*: Resiliency, Scale and Security, Comprehensive feature-set
  - *Evolve for Emerging Requirements*: Operational Simplicity, Programmability, Application-awareness
- The Open Network Environment integrates with existing infrastructure
  - Software Defined Network concepts are a component of the Open Network Environment
  - The OpenFlow protocol can be used to link agents and controllers, and as such is component of SDN as well



Programmatic Network Access – Multiple Layers

Full-Duplex access to the network at multiple layers and networking planes

- Enable a holistic Network Programming model
- Leverage and extend infrastructure at pace of the business
- Deploy common applications across all devices
- Extend/upgrade/add features without upgrading the network operating system
- Reduced time to market by leveraging common platform for building services



Open Network Environment Qualities

Programmatic APIs

Programmatic APIs

Programmatic APIs

The Need for Abstractions

Abstractions in Networking

- Data-plane Abstractions – ISO/OSI Layering
  - Examples
    - Local best effort delivery (e.g., Ethernet)
    - Global best effort delivery (e.g., IP)
    - Reliable byte-stream (e.g., TCP)
  - Data plane abstractions are key to Internet's success
- Abstractions for the other planes (control, services, management, orchestration,..) … are missing
  - Consequences include:
    - Notorious difficulty of e.g. network management solutions
    - Difficulty of evolving software for these planes

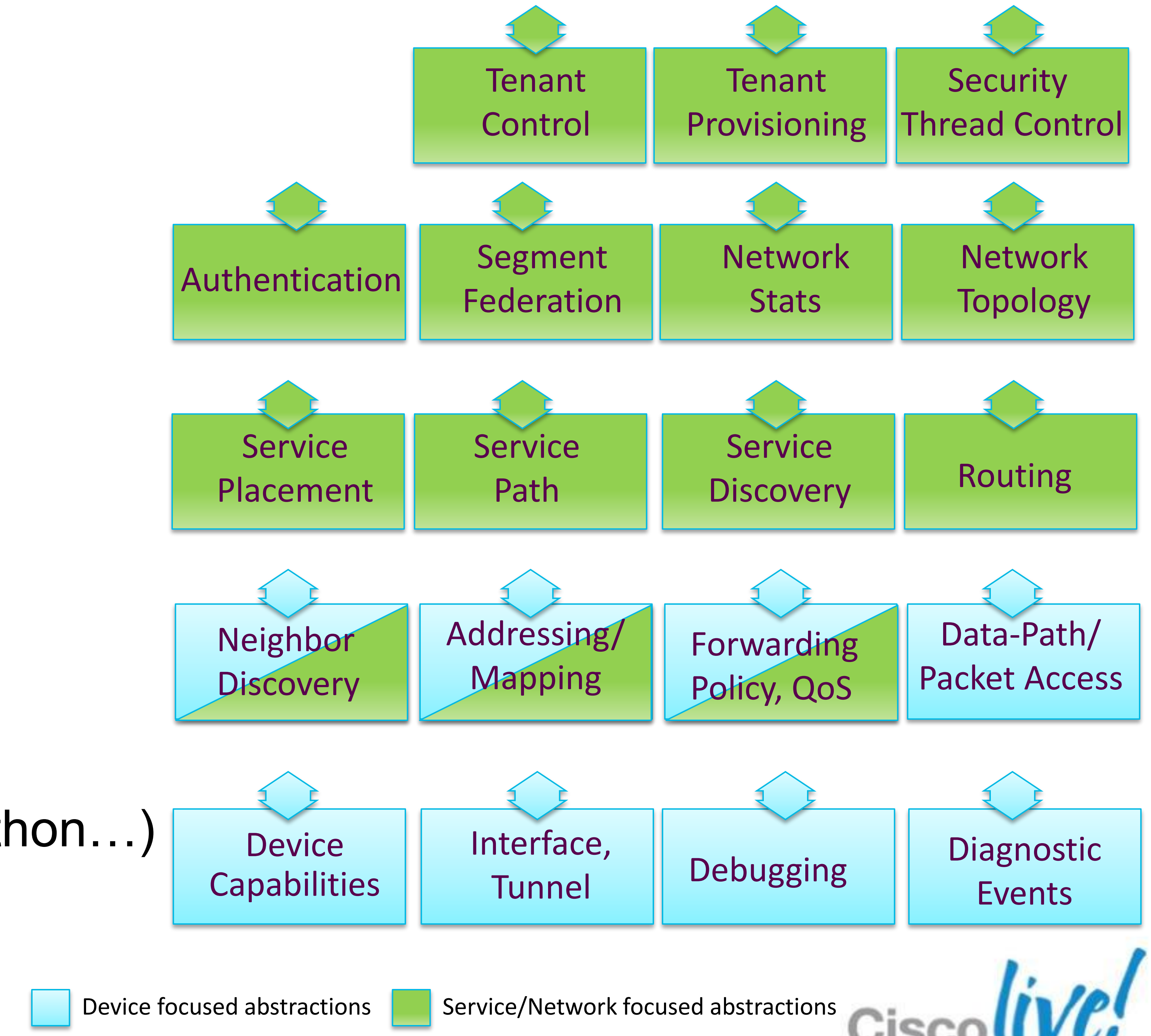
> *"Modularity based on abstraction is the way things get done"*
>
> Barbara Liskov
> Turing Award Winner

"In computer science, **abstraction** is the process by which data and programs are defined with a representation similar in form to its meaning (semantics), while hiding away the implementation details. Abstraction tries to reduce and factor out details so that the programmer can focus on a few concepts at a time. A system can have several abstraction layers whereby different meanings and amounts of detail are exposed to the programmer. For example, low-level abstraction layers expose details of the computer hardware where the program is run, while high-level layers deal with the business logic of the program."

http://en.wikipedia.org/wiki/Abstraction_Computer_Science

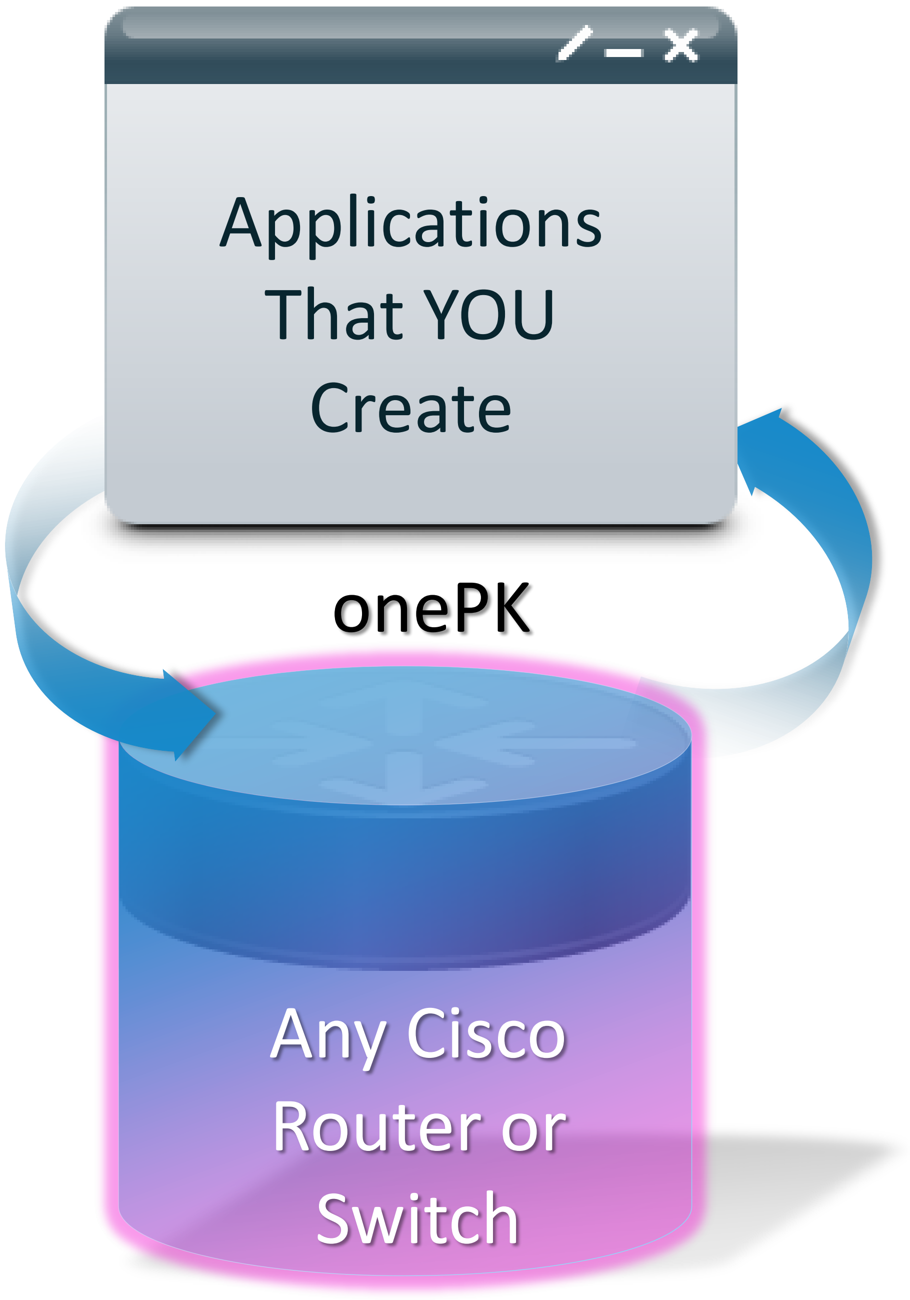
Approaching abstractions for Networking

- Abstractions allow the definition of associated APIs
  - Enable API platform kit across all platforms, to integrate with development environments
  - Accelerate development of network applications: Completely integrated stack from device to network
  - Multiple deployment modes (local and remote (blade/server) based APIs)
  - Multiple Language Support (C, Java, Python…)
  - Integrate with customer development to deliver enhanced routing, forwarding..



APIs make Abstractions available to Programmers

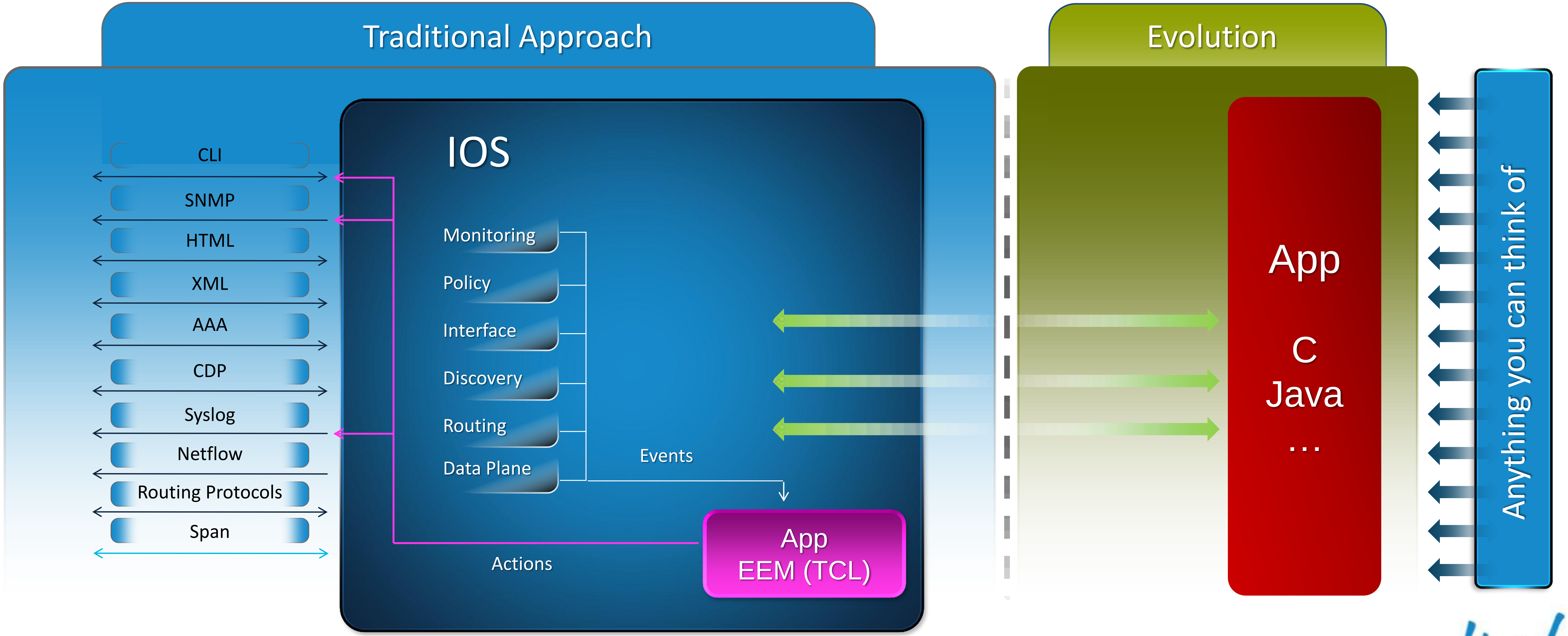
Example: Cisco's onePK (one Programming Kit) – Get your build on!



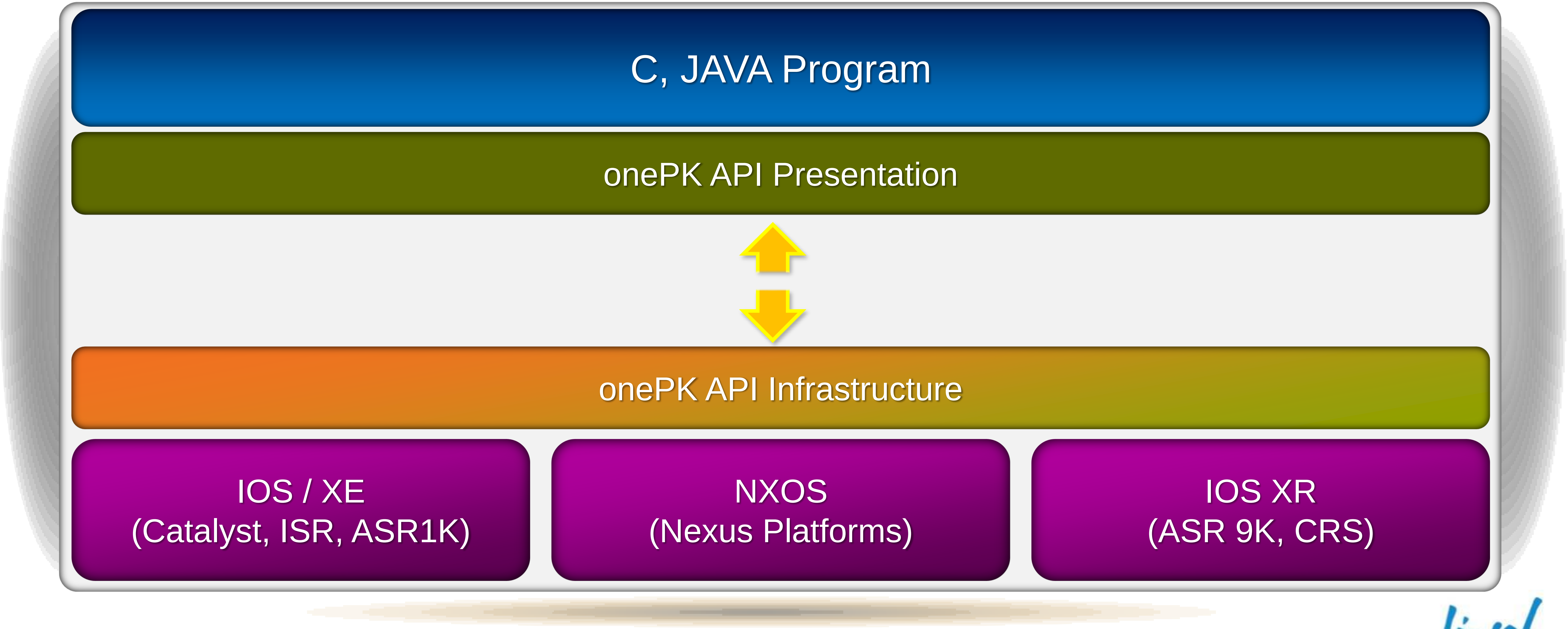
Flexible development environment to:

- Innovate
- Extend
- Automate
- Customise
- Enhance
- Modify

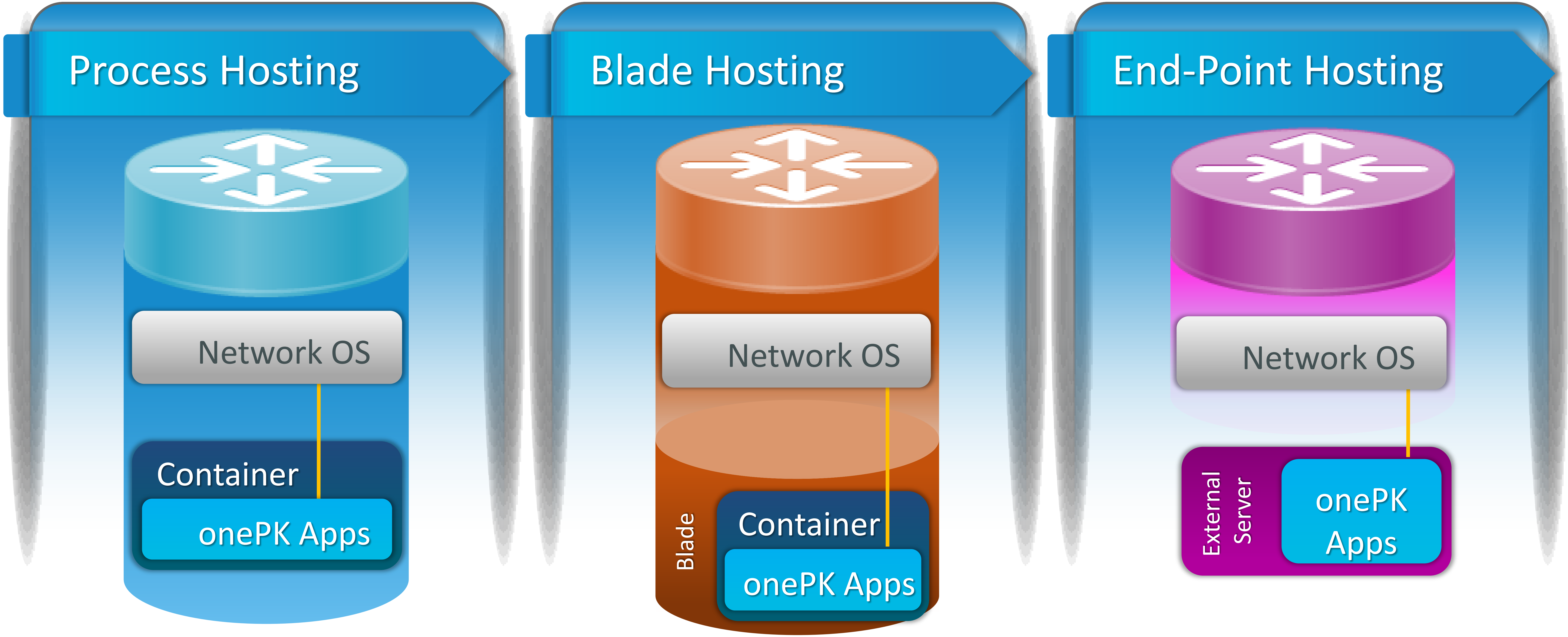
Evolving How We Interact With The Network Operating System



onePK Architecture



onePK Application Hosting Options



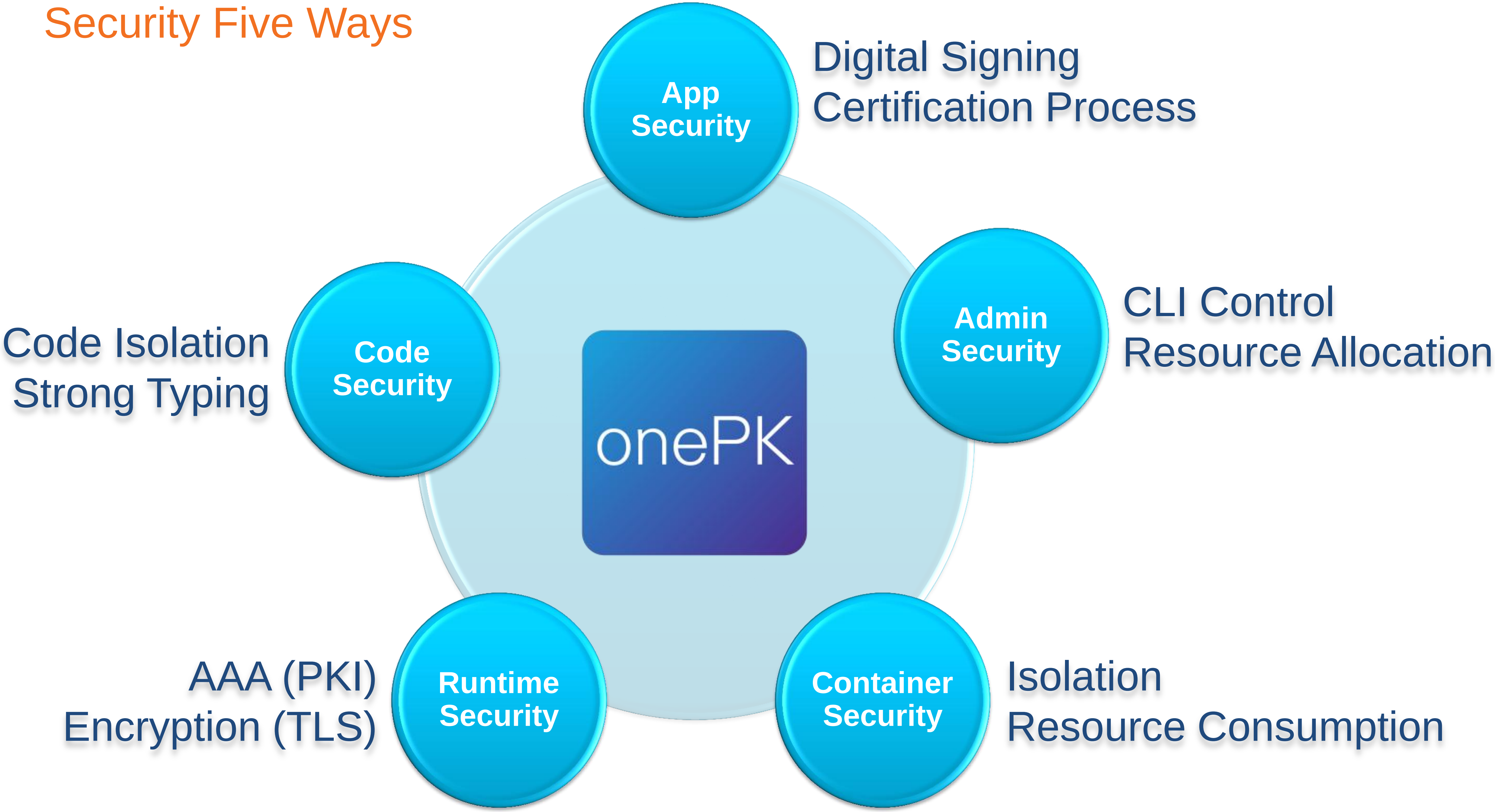
Write Once, Run Anywhere

onePK APIs are Grouped in Service Sets

Base Service Set	Description
Data Path	Provides packet delivery service to application: Copy, Punt, Inject
Policy	Provides filtering (NBAR, ACL), classification (Class-maps, Policy-maps), actions (Marking, Policing, Queuing, Copy, Punt) and applying policies to interfaces on network elements
Routing	Read RIB routes, add/remove routes, receive RIB notifications
Element	Get element properties, CPU/memory statistics, network interfaces, element and interface events
Discovery	L3 topology and local service discovery
Utility	Syslog events notification, Path tracing capabilities (ingress/egress and interface stats, next-hop info, etc.)
Developer	Debug capability, CLI extension which allows application to extend/integrate application's CLIs with network element

Yes, it is secure

Security Five Ways



Open Network Environment Qualities

Programmatic APIs

ONF's 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 standardised 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

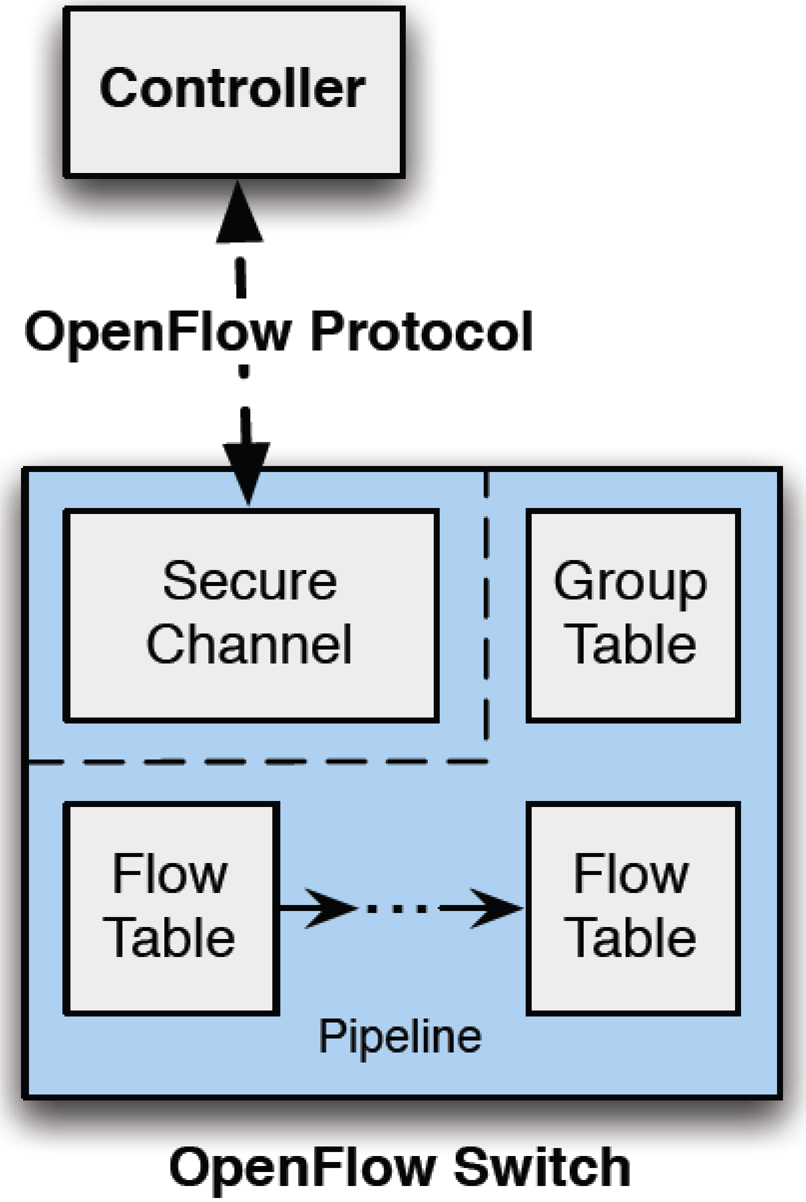
Nothing new under the sun

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

- Ipsilon Flow Switching
  - Centralised flow based control, ATM link layer
  - GSMP (RFC 3292)
- AT&T "SDN"
  - Centralised 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

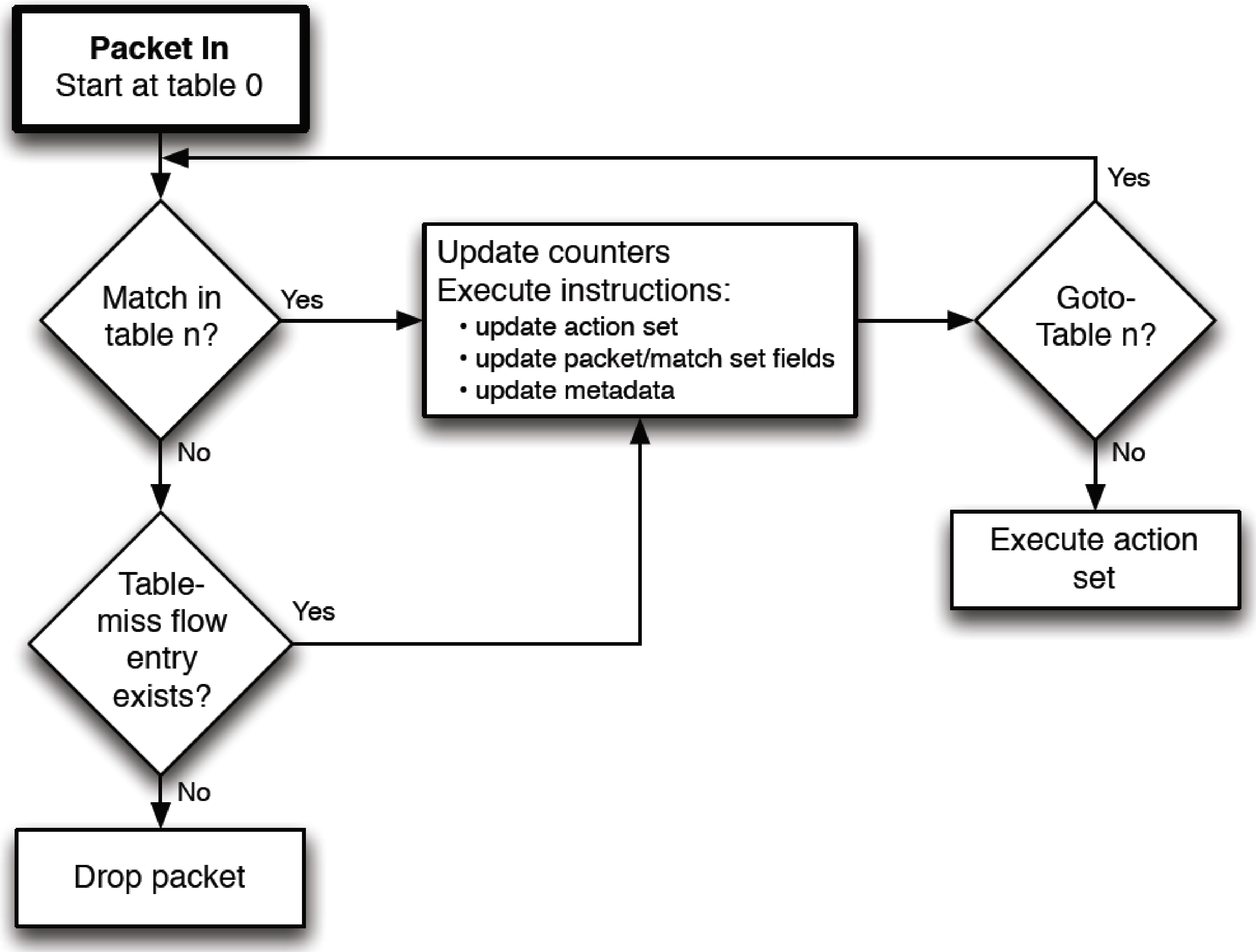
- OpenFlow Components
  - – *Application Layer Protocol*: OF-Protocol
  - – *Device Model*: OF-Device Model (abstraction of a device with Ethernet interfaces and a set of forwarding capabilities)
  - – *Transport Protocol*: Connection between OF-Controller and OF-Device*
- Observation:
  - – OF-Controller 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.

Packet Flow through an OpenFlow Switch



Source: OpenFlow 1.3.0 specification, figure 3

Required Match Fields

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

OpenFlow Actions

- Output
- Set-Queue* (for QoS)
- Drop
- Group
- Push-Tag/Pop-Tag*
- Set-Field* (e.g. VLAN)
- Change-TTL*

*Optional

OpenFlow Ports

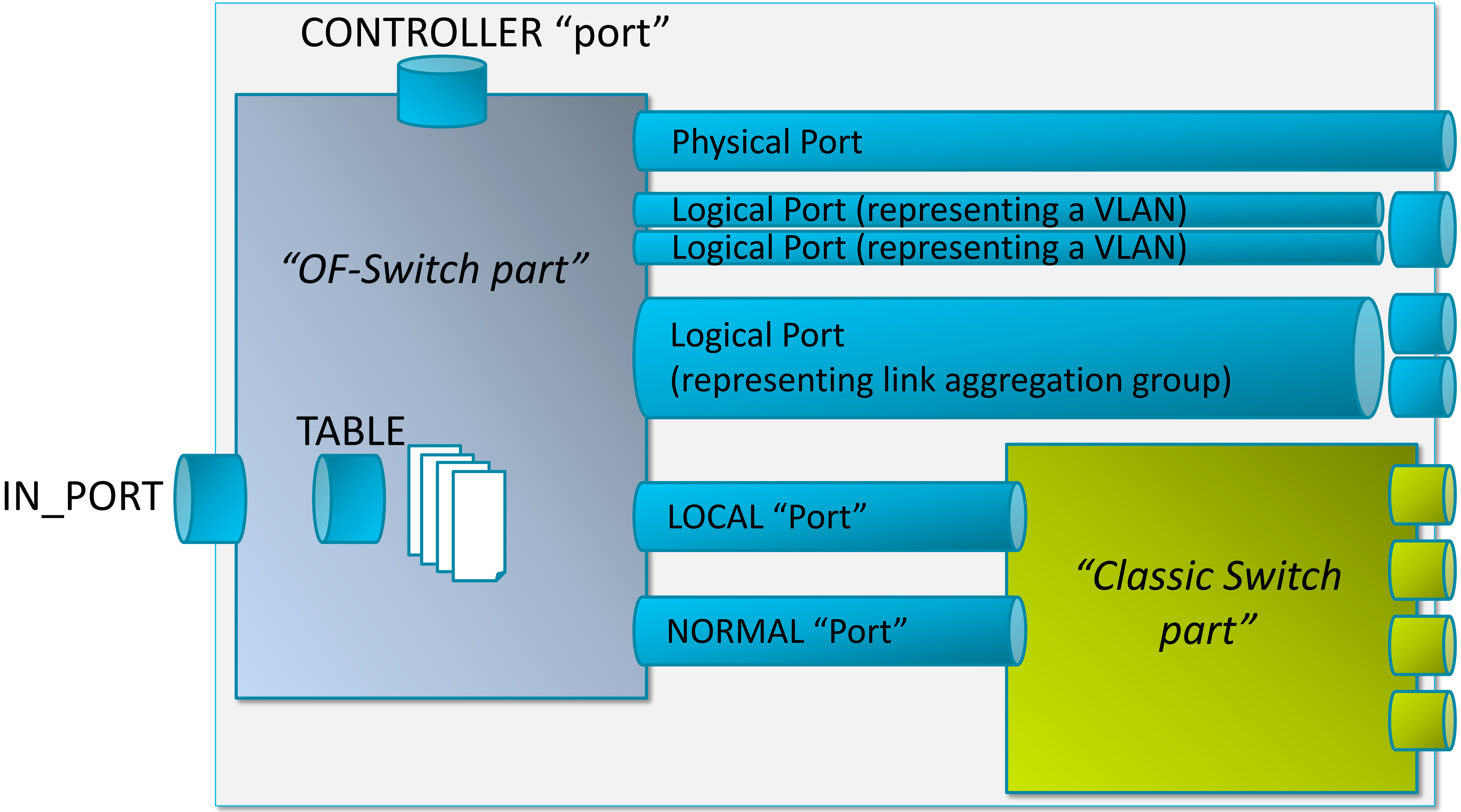
Physical Ports, Logical Ports, Reserved Ports

- 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

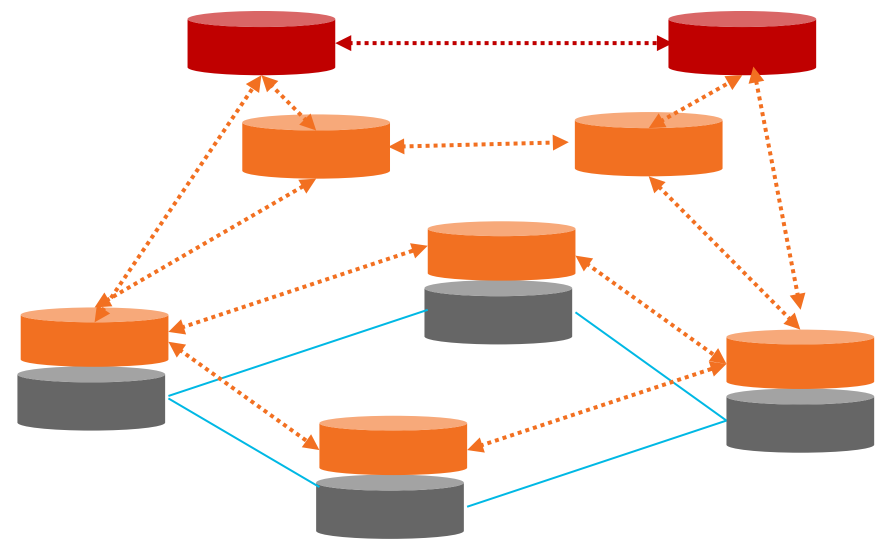
OpenFlow Ports

Simplified View



Hybrid Model

- One criticism of OpenFlow
  - OpenFlow is making all switches dumb, it requires complete re-implementation of entire control plane in the logically centralised controller (due to OpenFlow being a protocol)
- Hybrid Model acknowledges a more generic approach: Re-architect the control plane architecture *where needed*
  - Keep existing control planes on network devices and evolve/complement them – e.g. maximum scale, node & link diversity, availability combined with optimisations which follow business metrics (e.g. $-cost, geographic/political considerations, ..)
- Hybrid Model Concerns include
  - Reconciliation of state required in case multiple modules can create competing decisions (e.g. using the RIB)
  - Potentially requires the OpenFlow device model to evolve and to include additional abstractions

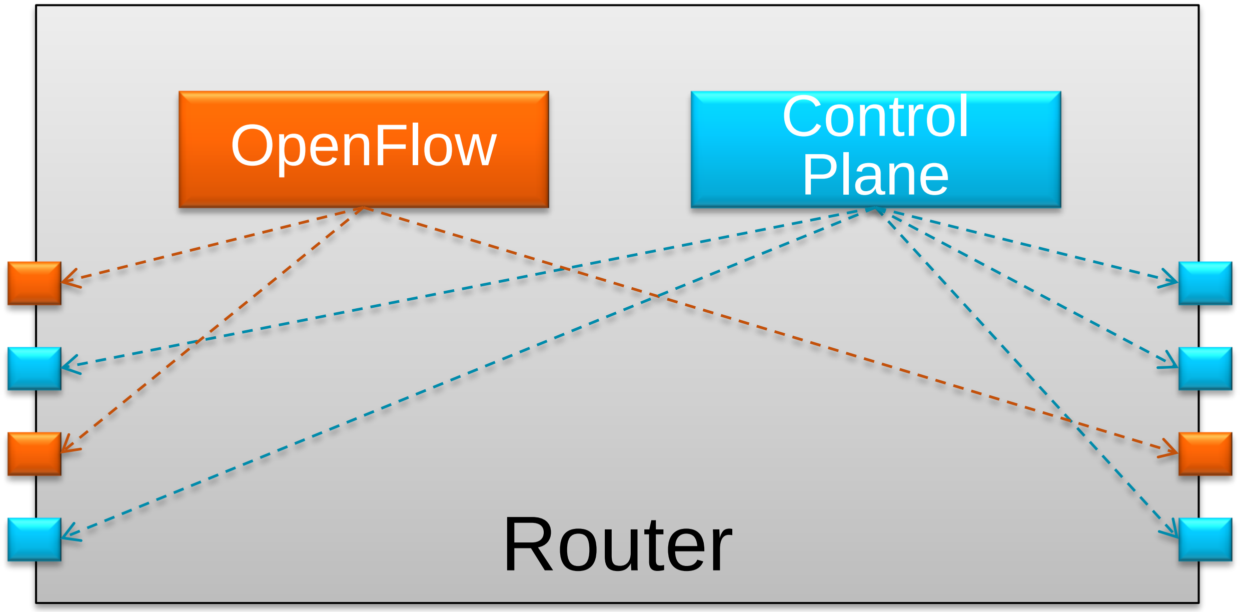


A Couple Of Hybrid Switch Use Cases

- Installing ephemeral routes in the RIB
  - Install routes in RIB subject to admin distance or …
  - Moral equivalent of static routes, but dynamic
  - May require changes to the OF protocol/model
- Edge classification
  - Use OF to install **ephemeral** classifiers **at the edge**
  - Moral equivalent of … 'ip set next-hop <addr>' (PBR)
  - Use case: Service Engineered Paths/Service Wires
    - Program switch edge classifiers to select set of {MPLS, GRE, …} tunnels
    - Core remains the same
- Service Chaining

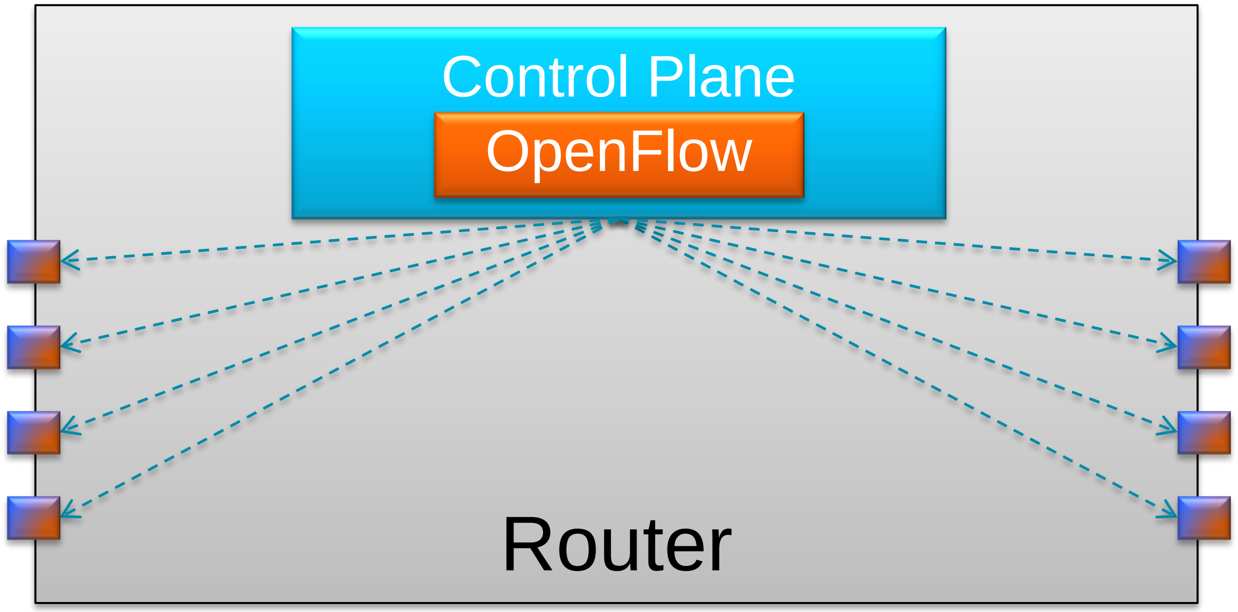
Hybrid Switch: Ships in the Night vs. Integrated

"Ships-in-the-Night"



- A subset of ports controlled by OF, another subset controlled by router's native CP – physical resources are partitioned
- Some level of integration: "OF_NORMAL":
  - Implementer free to define what "normal" is
  - May or may not be what router normally does

"Integrated"



- Use OF for feature definition – augment the native control plane
- No longer partitioning of resources
- Can operate at different abstraction levels (low-level like OF1.0 or higher level)

OpenFlow Versions

Status

Dec 31, 2009	Feb 28, 2011	Dec 5, 2011	April 19, 2012
OF 1.0	OF 1.1	OF 1.2	OF 1.3.0
• 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

- Current ONF focus is on maturing the specification
  - No new feature release in 2012
  - "Working code before new standards"
  - "ONF should not anoint a single reference implementation but instead encourage open-source implementations"; ONF board encourages multiple reference implementations

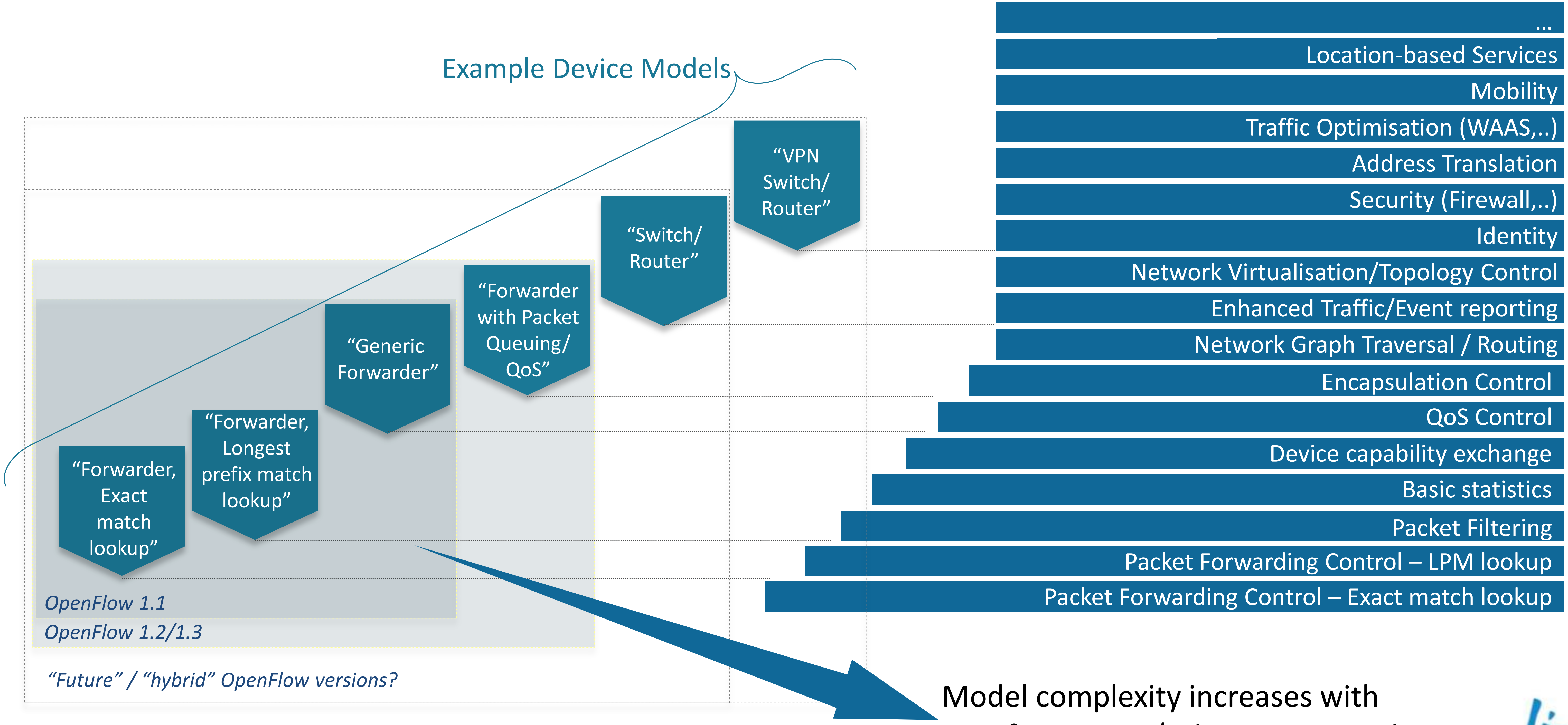
OpenFlow Evolution

Making OF functionally complete

- Examples of ongoing work
  - Hardware friendly switch model negotiations (“typed tables”)
  - Investigate OpenFlow as an interface to the Control plane of a switch (“Hybrid Switch Model – Integrated Mode; e.g. incl. Layer 3 forwarding model etc.)
  - Security model (granular access control)
  - High availability model for device and controller (state re-sync etc.)
  - OF protocol not easily extensible

OpenFlow Evolution Challenge

Device Model: Getting the right abstraction is hard



Model complexity increases with set of use-cases/solutions covered

Resource Orchestration Controllers

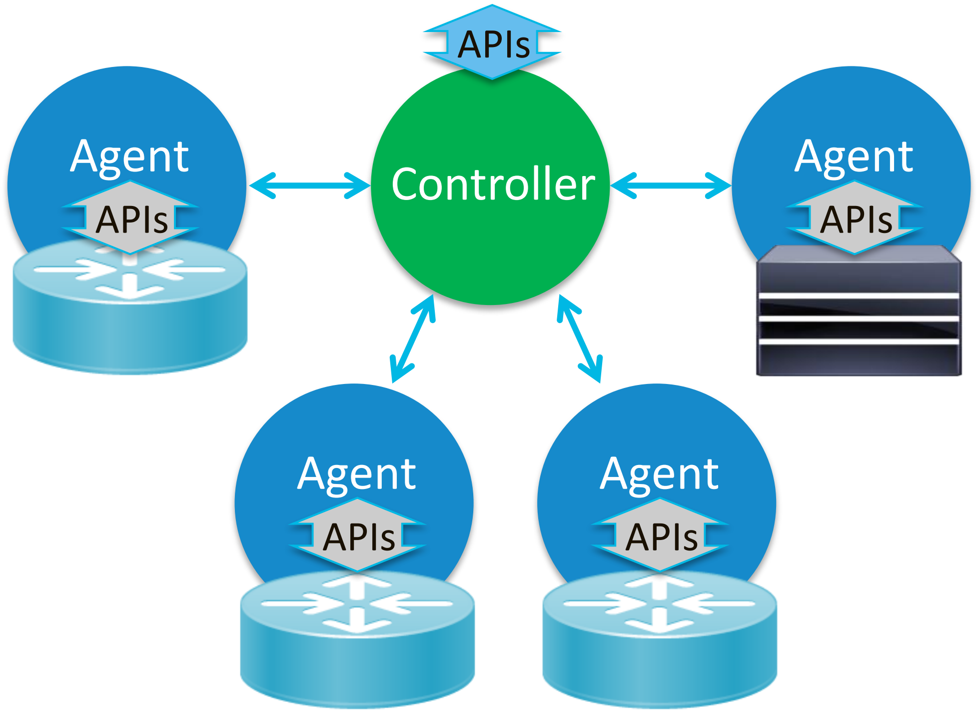
Open Network Environment Qualities

Resource Orchestration – Controllers:

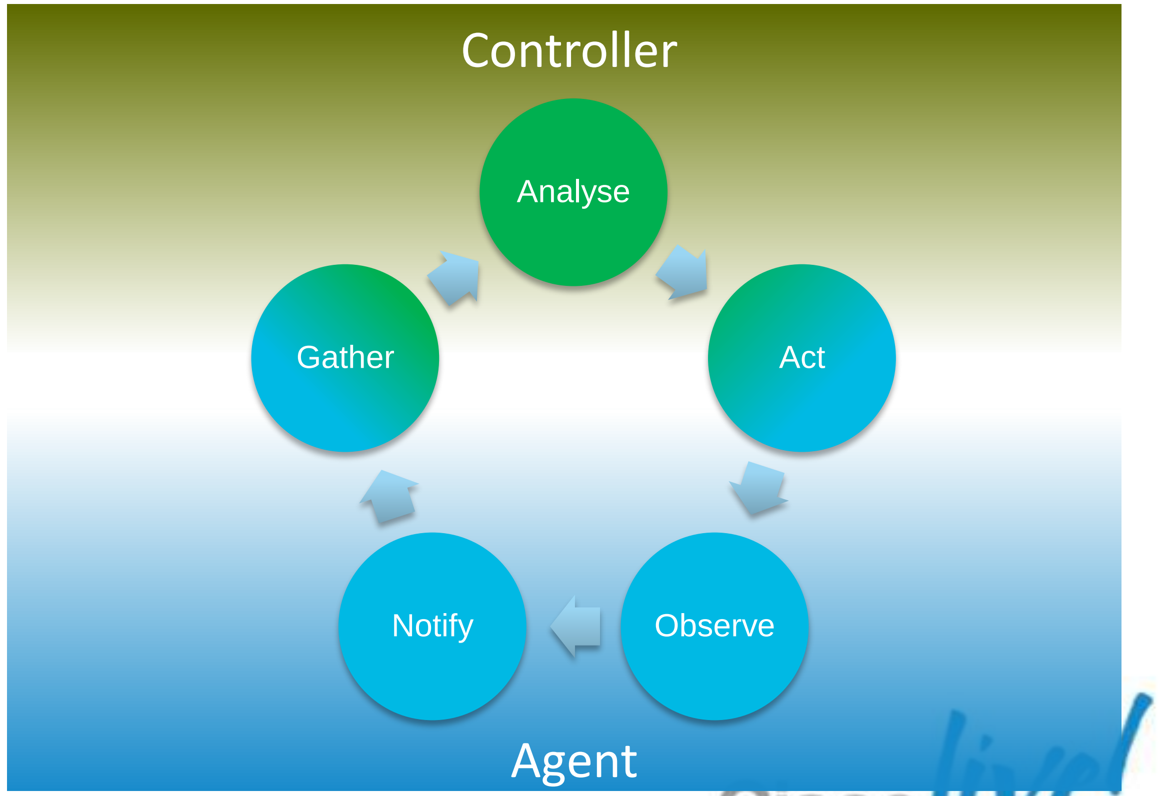
Logically centralised and fully distributed Control

Orchestration: Agents and Controllers

Consolidate State Across Multiple Network Elements



- Some network delivered functionality benefits from logically centralised coordination across multiple network devices
  - Functionality typically domain, task, or customer specific
  - Typically multiple Controller-Agent pairs are combined for a network solution
- Controller
  - Process on a device, interacting with a set of devices using a set of APIs or protocols
  - Offer a control interface/API
- Agent
  - Process or library on a device, leverages device APIs to deliver a task/domain specific function
- Controller-Agent Pairs offer APIs which integrate into the overall Network API suite

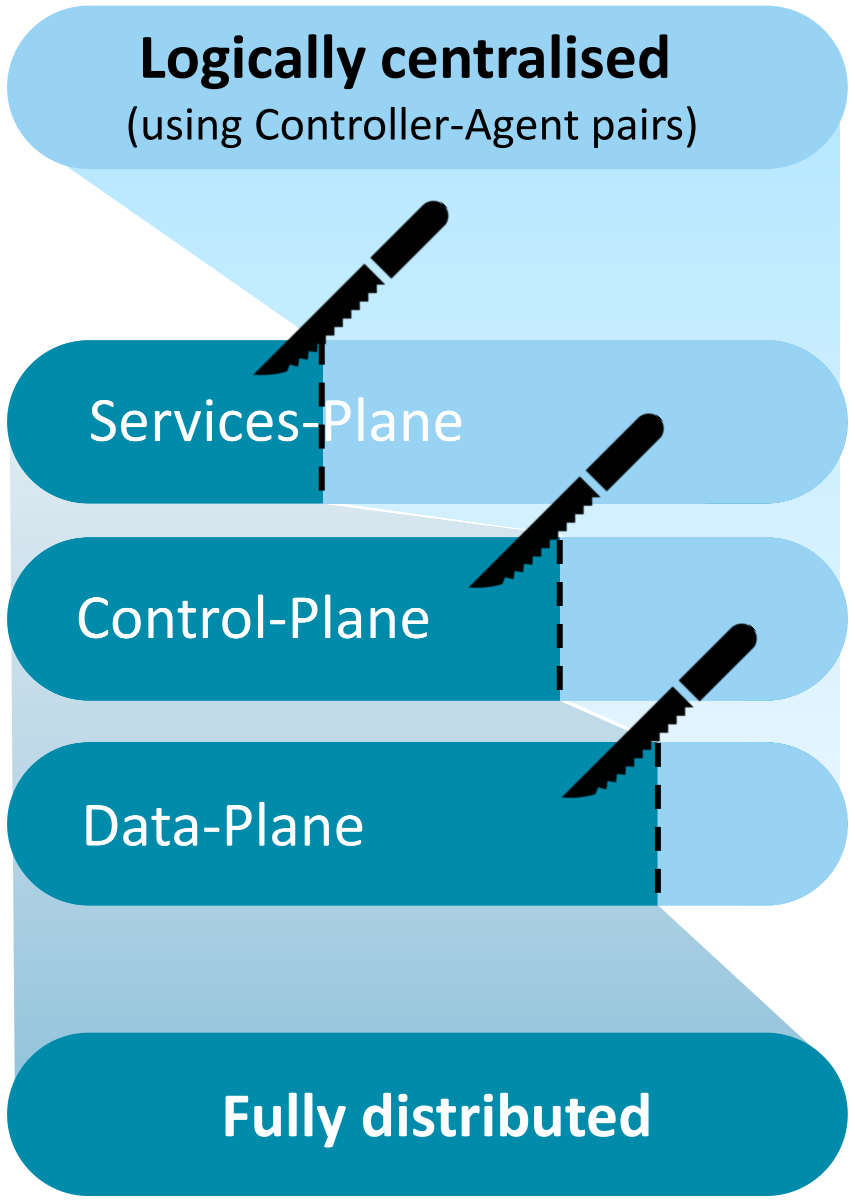


Distributed Control

Exploring the tradeoff between Agents and Controllers – and fully distributed Control

- Control loop requirements differ per function/service and deployment domain
  - "As loose as possible, as tight as needed"
  - Latency, Scalability, Robustness, Consistency, Availability
  - Different requirements per use case

    Example:
    Topology for Visualisation (Network Management) vs. Topology for Path-Computation/Routing
- *How to decide which functionality is well suited a particular control paradigm?*



Note: Example only – Not all network planes shown

“**Subsidiarity** is an organising principle that matters ought to be handled by the smallest, lowest or least centralised competent authority.” http://en.wikipedia.org/wiki/Subsidiarity

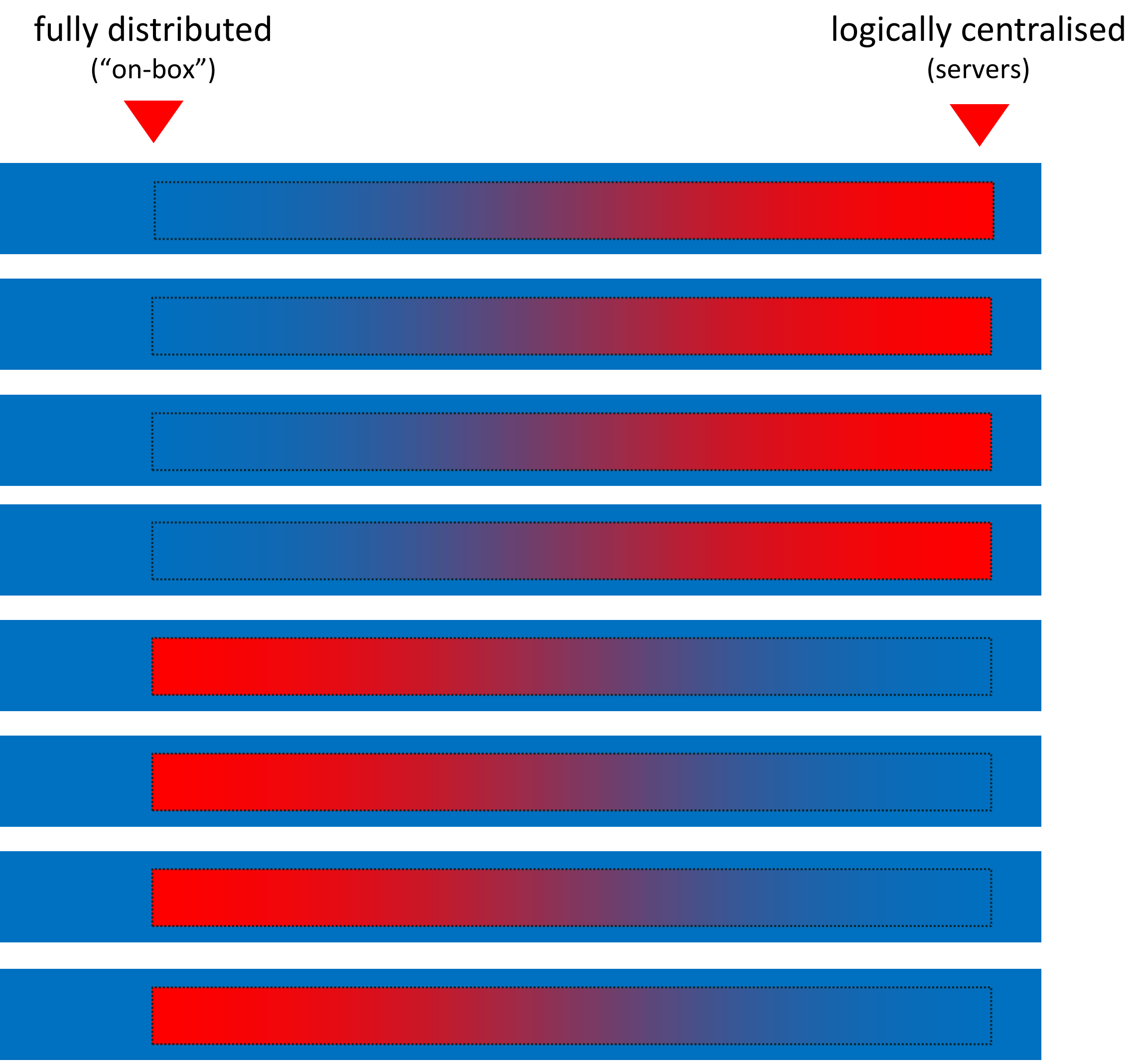
Decision making feedback loop

Considerations – Applying Subsidiarity to Networking

- Locations of event/state-source, state-processing, decision-making, action-taking can follow specific requirements
  - Fully distributed, agent/controller architectures, etc.
- Different design goals and pre-requisites
  - Required reaction time-scales
    - Fast convergence (e.g. for voice/video apps) vs. conservative reaction times (long running batch-type applications)
  - Leveraging state/information from multiple sources (network and applications) for decision making
  - Macro vs. micro-level decision making (e.g. link/device layer redundancy vs. cluster/POD level redundancy in MSDC)

Evolving the Control Plane Environment

Deployment Considerations – Applying Subsidiarity to Networking



- Rapid prototyping (TTM vs. performance)
- Algorithms which require coordination between instances, benefit from "a global view"
- Large scale tables with relatively infrequent updates (ARP,..)
- Controlled/tightly-managed (homogeneous) Environments
- Rapid response to Topology Changes: Efficient "plain vanilla" Layer-3-style forwarding
- Rapid response to data-plane events / packet forwarding
- Simplicity of Control- and Data-Plane Integration**
- Large scale

** Past experience (e.g. PSTN AIN, Softswitches/IMS, SBC): CP/DP split requires complex protocols between CP and DP.

* See also: Martin Casado's Blog: http://networkheresy.wordpress.com/2011/11/17/is-openflowsdn-good-at-forwarding/

Packet Forwarding Decision Making

An obvious observation

- Network devices delivering line rate forwarding performance need to take forwarding decisions as quickly as packets arrive
  - In case a forwarding rule exists, apply rule “at line rate”
  - In case a forwarding rule does not exist
    - Buffer the packet and create a rule (or ask someone else to create a rule)
      - How much buffer do you have?
    - Drop the packet
    - Flood the packet

Interframe Gap (IFG)	Standard (96 bit times)	Minimum on reception*
Ethernet	9.6us	4.7us
Fast Ethernet	0.96us	Not defined
Gigabit Ethernet	0.096us	0.064us
10 Gigabit Ethernet	0.0096us	0.0047us

*IFG shrinking is allowed to cope with variable network delays, clock tolerances, added preable bits

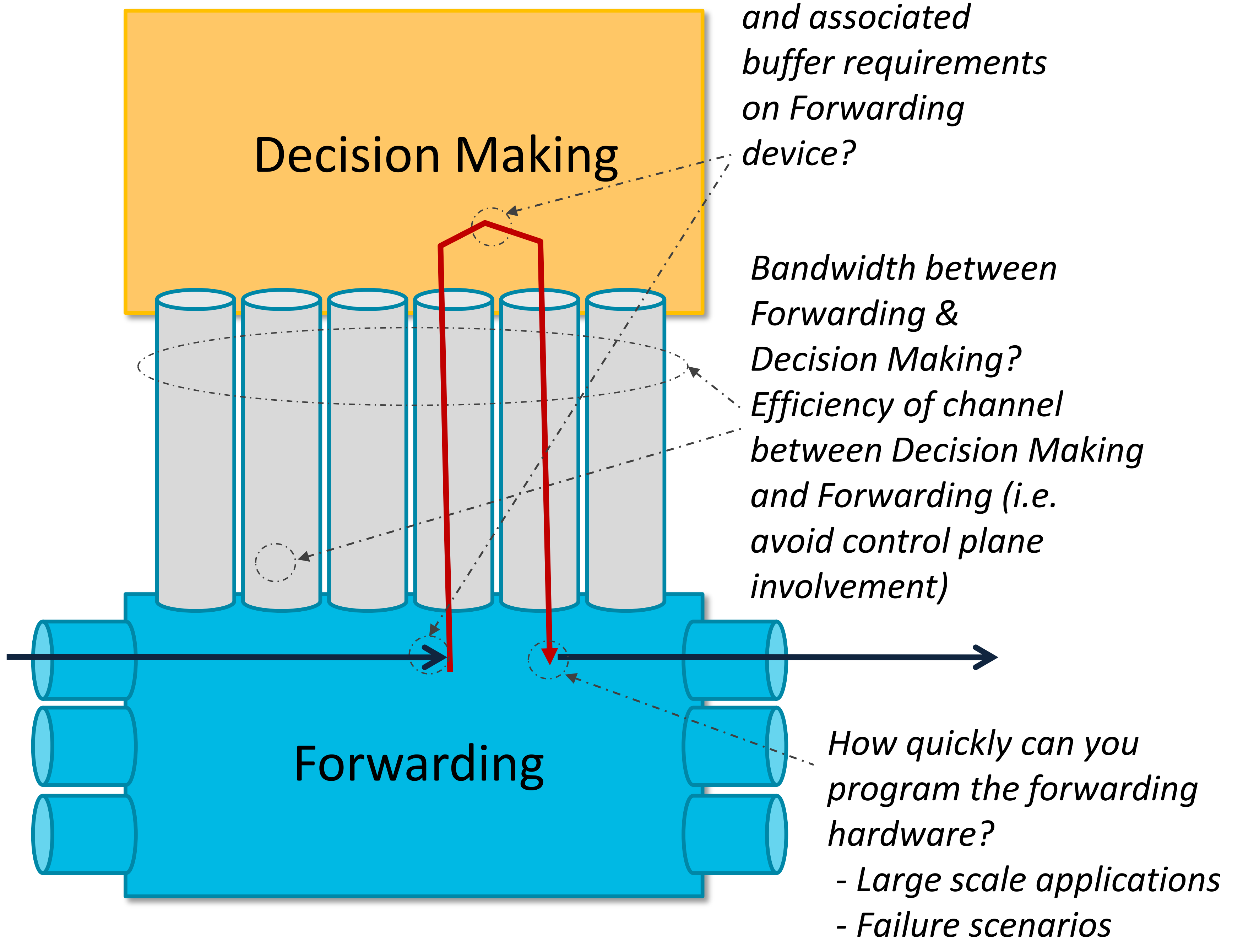
Forwarding decision making

Pro-Active and Re-Active Mode

- Pro-Active: Compute rules upfront and install in the devices
  - Typical behavior of a router
- Re-Active: Decide forwarding rules once packets arrive
  - Send first (or first few – think TCP) packets of a flow to a decision making entity (“Controller”) which creates a forwarding rule for the flow – and optionally performs forwarding on the first (or first few packets)
  - Bridges “kind-of” do this: Derive forwarding tables from packet MAC addresses: Requires hardware based learning
- Combinations
  - Default to pro-active, leverage re-active for certain exceptions (e.g. few, long living flows)

Reactive Mode: Considerations

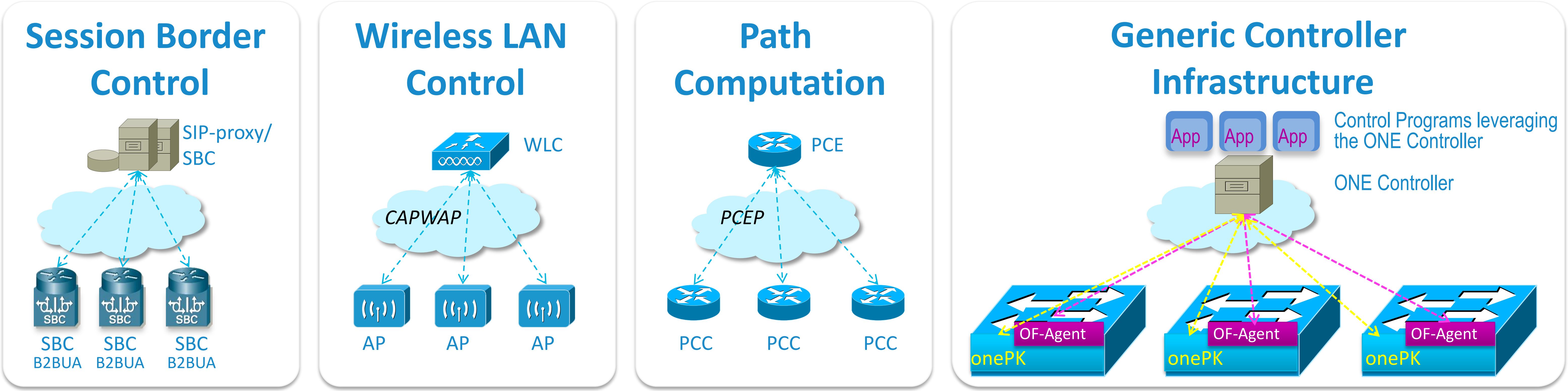
- Decision making delay and associated buffering requirements
- Flow-setup / Flow-teardown latency
- Scale
  – Bandwidth/packet forwarding requirements between forwarding entity and decision making entity
  – Total number of flows, feasibility of rule aggregation



Open Network Environment Qualities
Resource Orchestration – Controllers

Orchestration: Controllers and Agents

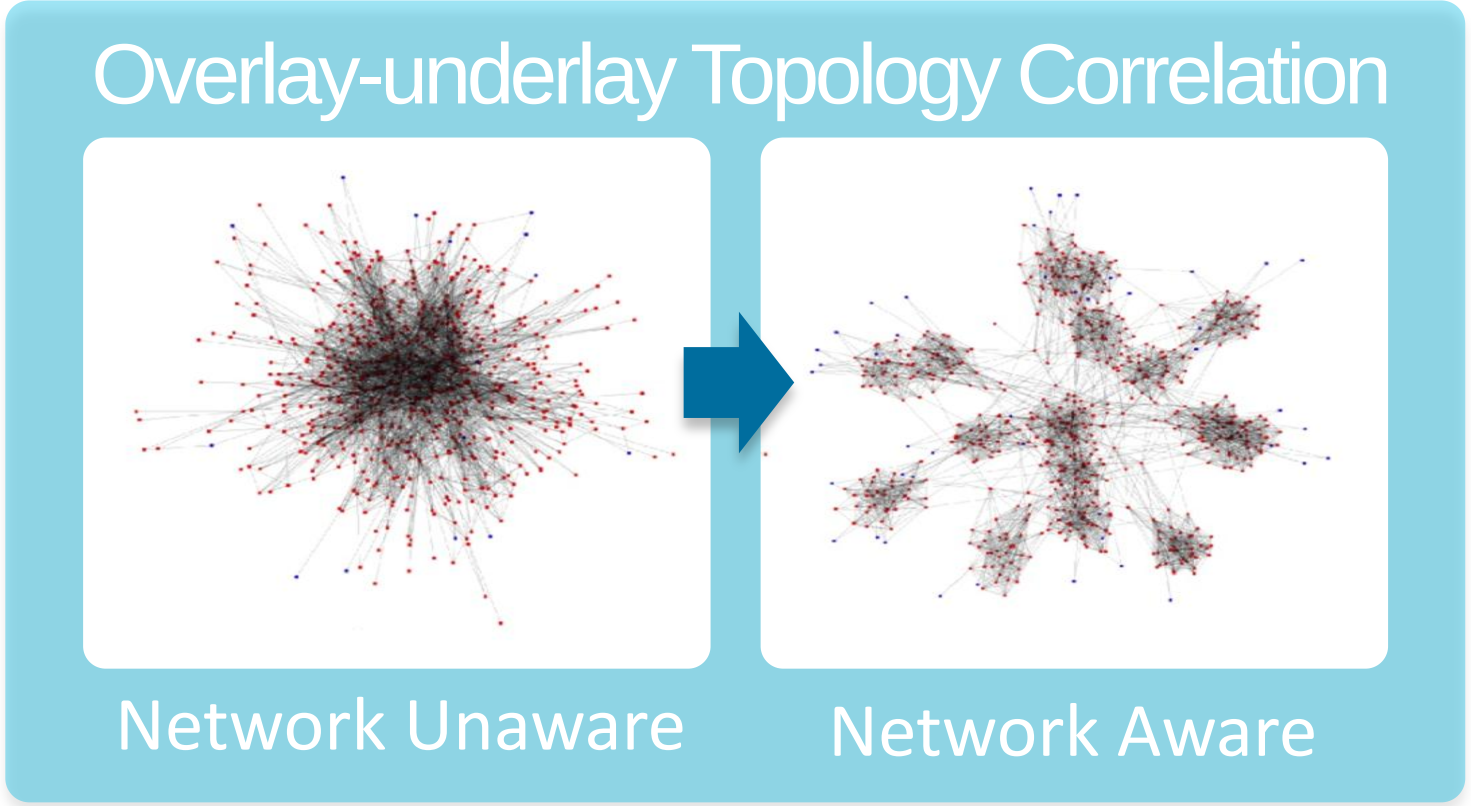
Task Specific Solutions and Generic Controller Infrastructure



- Networking already leverages a great breath of Agents and Controllers
  – Current Agent-Controller pairs always serve a specific task (or set of tasks) in a specific domain
- System Design: Trade-off between Agent-Controller and Fully Distributed Control
  – Control loop requirements differ per function/service and deployment domain
  – "As loose as possible, as tight as needed"
  – Latency, Scalability, Robustness, Consistency, Availability

Controllers as Aggregation and Decision Points Example: Network Positioning System (NPS)

- Computes the location of and distance between endpoints
  - Caching and replication are vital to optimisation of network traffic. Distribution paradigms efficiency is augmented by dynamic mechanisms that locate (and determine distance to) services and data in order to optimise infrastructure resources utilisation
  - Example: need to locate the nearest copy of a movie or the closest instance of a service among several available resources
- NPS/Proximity leverages network layer and Policy information.
  - Extended to other information sources such as: state & performance and Geo-location



Aggarwal et al. show that when the p2p overlay topology is network aware, it is highly correlated with the underlying network topology; the nodes within an AS form a dense cluster, with only a few connections going to nodes in other AS.

(Aggarwal, V., Feldmann, A., and C. Scheideler, "Can ISPs and P2P systems co-operate for improved performance?", ACM SIGCOMM Computer Communications Review (CCR), 37:3, pp. 29-40)

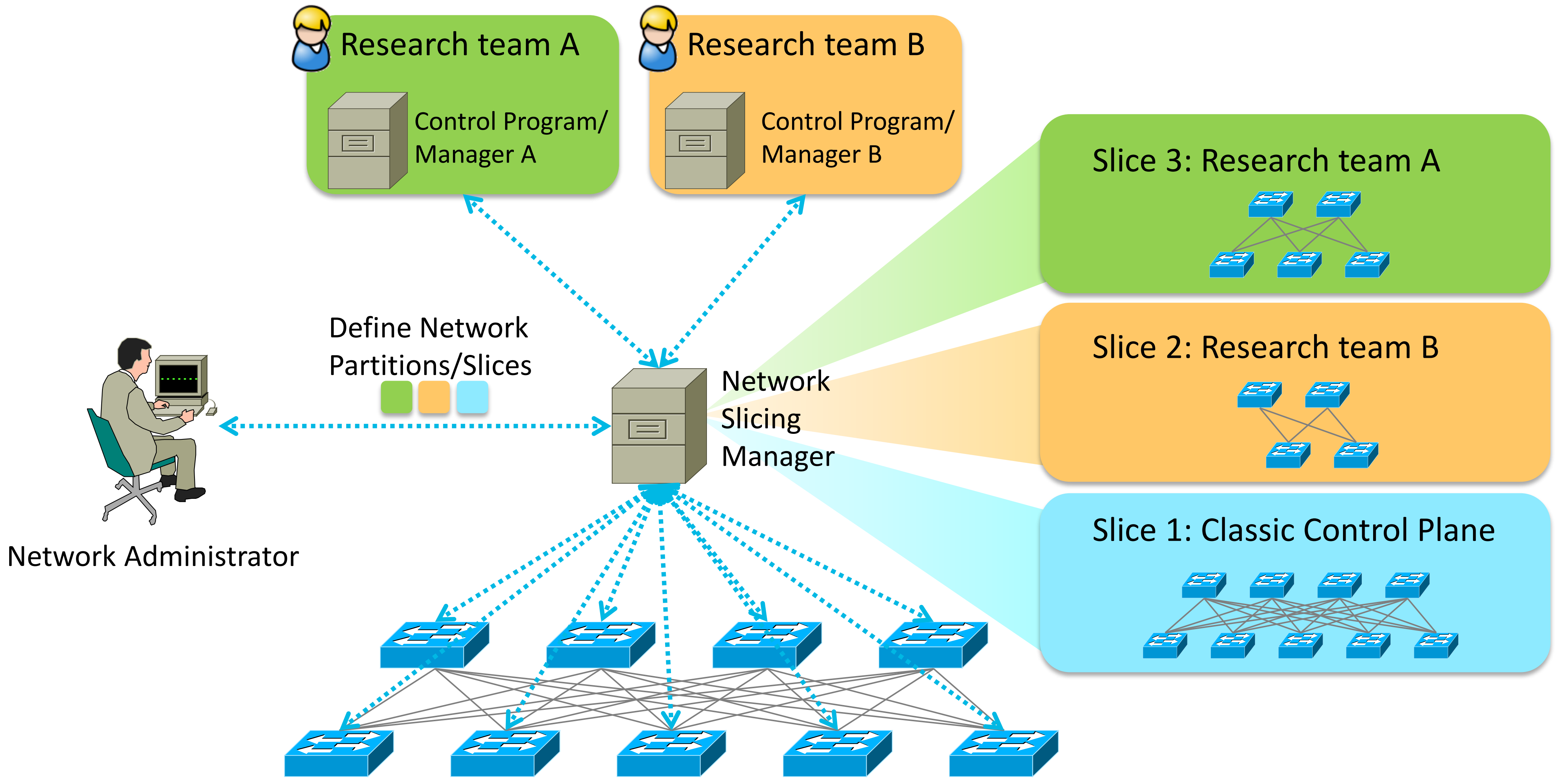
Complementing classic VPN technologies

Network Partitioning, a.k.a. “Slicing”

- VPN (L3VPN, L2VPN) technologies combine
  - Network Partitioning/Segmentation
  - Packet Forwarding Control (Control plane)
- “Slicing” refers to Network Partitioning *only*,
  i.e. no assumptions on the control plane made
  - Slices fully isolated (one slice not effecting resources and operation of other slices)
  - Several existing technologies incorporate “slicing concepts”, e.g.
    - PBB-TE – network partitioned based on I-SID/VLANs (one partition controlled by STP, another one through a NMS)
    - MPLS-TP
- ONE-Controller: “Network slicing manager”
  - Slicing manager defines/administers slices and maintains view of all slices in the network
  - Users only see their “slice” – can be used e.g. as sandbox network for a given Dept/Developer

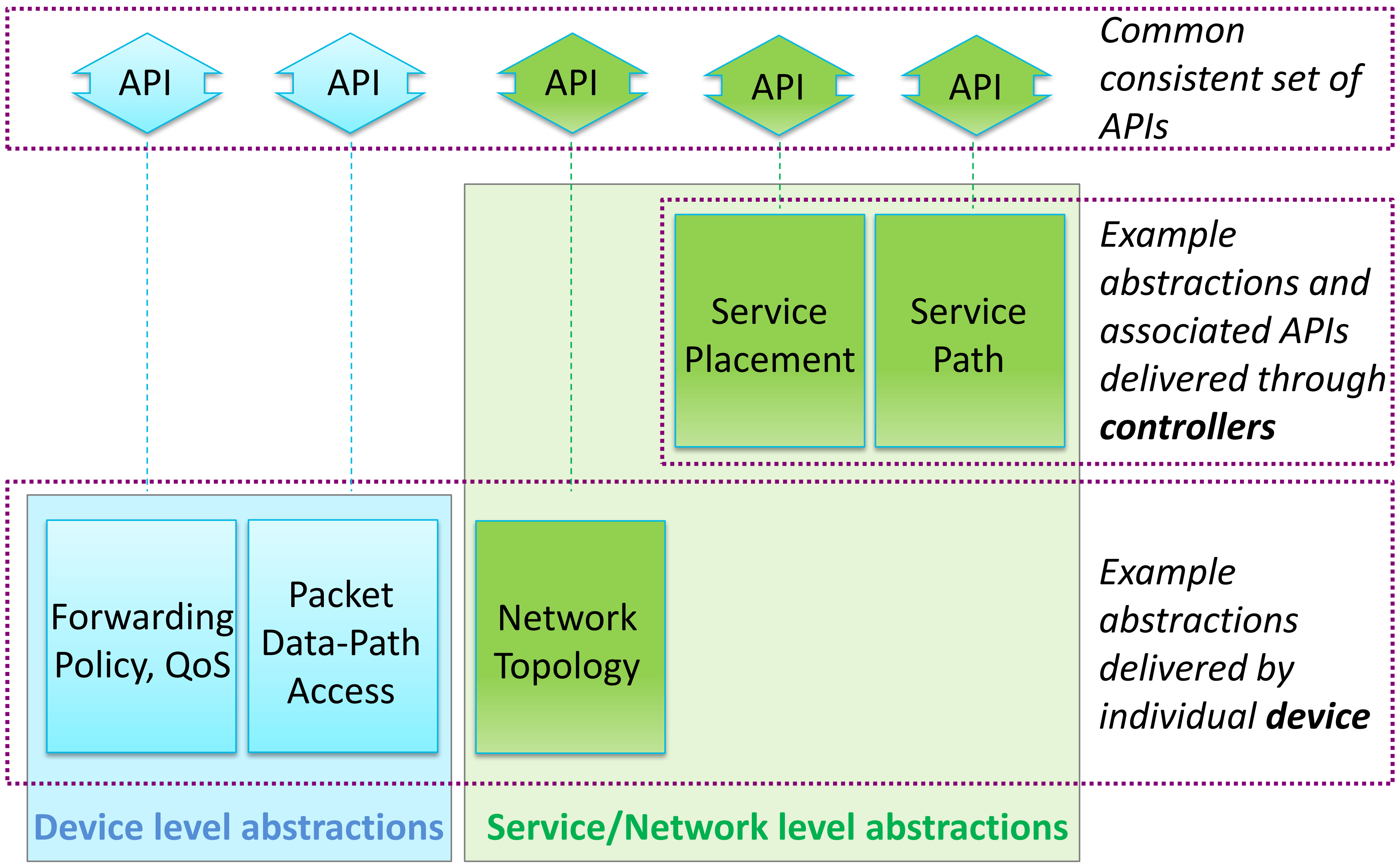
Orchestration & Virtualisation: Network Partitioning Example: Network Slicing for Research Environments

- Business Problem
  - University desires to "slice" the network into multiple partitions:
    - Production network – classic control plane
    - Several research networks – experimentation with new control algorithms, programs etc.
- Solution
  - Network Slicing Manager partitions the network based on e.g. ports or VLANs
    - Provides northbound interfaces, incl. OpenFlow (Flowvisor-like)
    - Effects of a particular control function of a partition/slice limited to that partition/slice



Devices, Controllers – and APIs

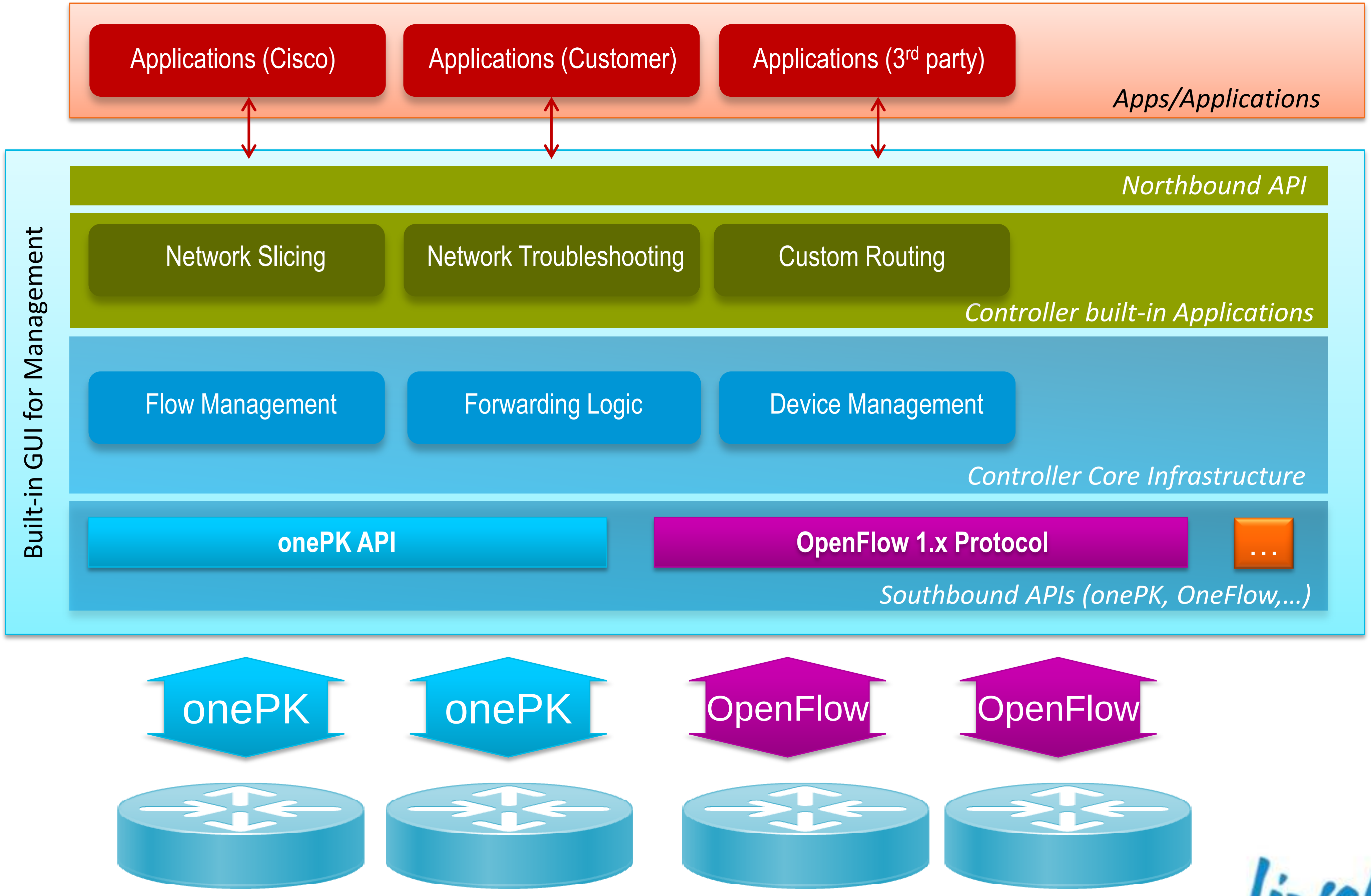
- APIs represent abstractions at different layers – complementing each other
  - – Device-layer, Network-layer etc.
  - – Devices can deliver network level abstractions and APIs as well (e.g. link state topology)
  - – Common, consistent API, different scopes



Orchestration & Control

Cisco "ONE Controller"

- Platform for generic control functions – state consolidation across multiple entities
- Current Showcase Examples
  - Flexible Network Partitioning and Provisioning ("Slicing")
  - Network Troubleshooting
  - Custom Routing
- Java-based



Network/Virtual Overlays

Open Network Environment Qualities
Network Infrastructure Virtualisation

Programmatic APIs

Resource Orchestration – Controllers

Network/ Virtual Overlays

"In computing, **virtualisation** is the creation of a virtual (rather than actual) version of something, such as a hardware platform, operating system, storage device, or network resources."

http://en.wikipedia.org/wiki/Virtualization

Network Abstractions support Virtualisation

Blurring the lines between physical and virtual entities – networks and services

Common Abstractions and common APIs across physical and virtual network elements

- Virtual Overlay Networks
  - custom endpoint addressing (e.g. for simple endpoint mobility)
  - custom topologies/segmentation
  - custom service chains

    Example: vPath

Map 'n Encap approaches to allow for flexible overlays and "identity" and "location" addresses:

- *L2-transport*: FabricPath, 802.1ah
- *IP-transport*: VXLAN, OTV, (L2-)LISP (all use the same frame format)
- *MPLS-transport*: (PBB-)VPLS, (PBB-)EVPN

- Virtual Service Nodes/Appliances/Gateways
  - VSG, vWAAS, CSR1000v, ASA 1000v, ...

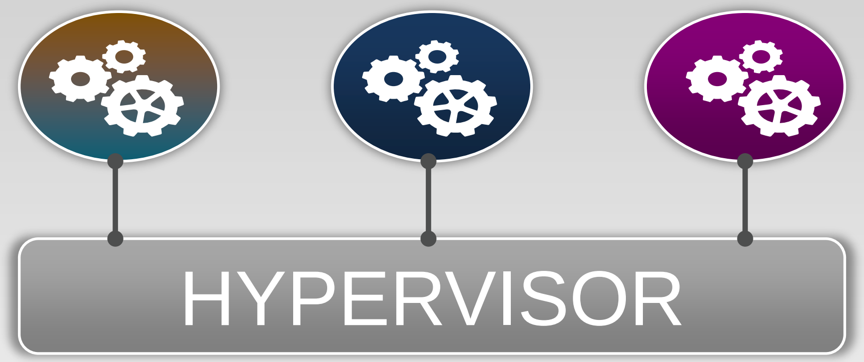
Physical | Virtual | Cloud Journey

PHYSICAL WORKLOAD

- One app per Server
- Static
- Manual provisioning

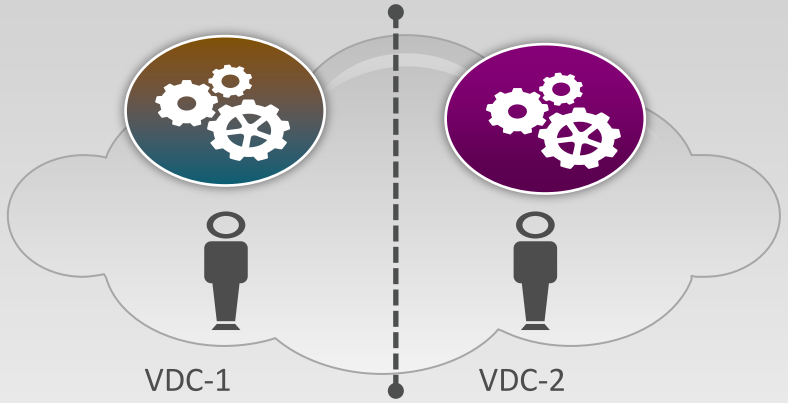
VIRTUAL WORKLOAD

- Many apps per Server
- Mobile
- Dynamic provisioning



CLOUD WORKLOAD

- Multi-tenant per Server
- Elastic
- Automated Scaling



CONSISTENCY: Policy, Features, Security, Management, Separation of Duties

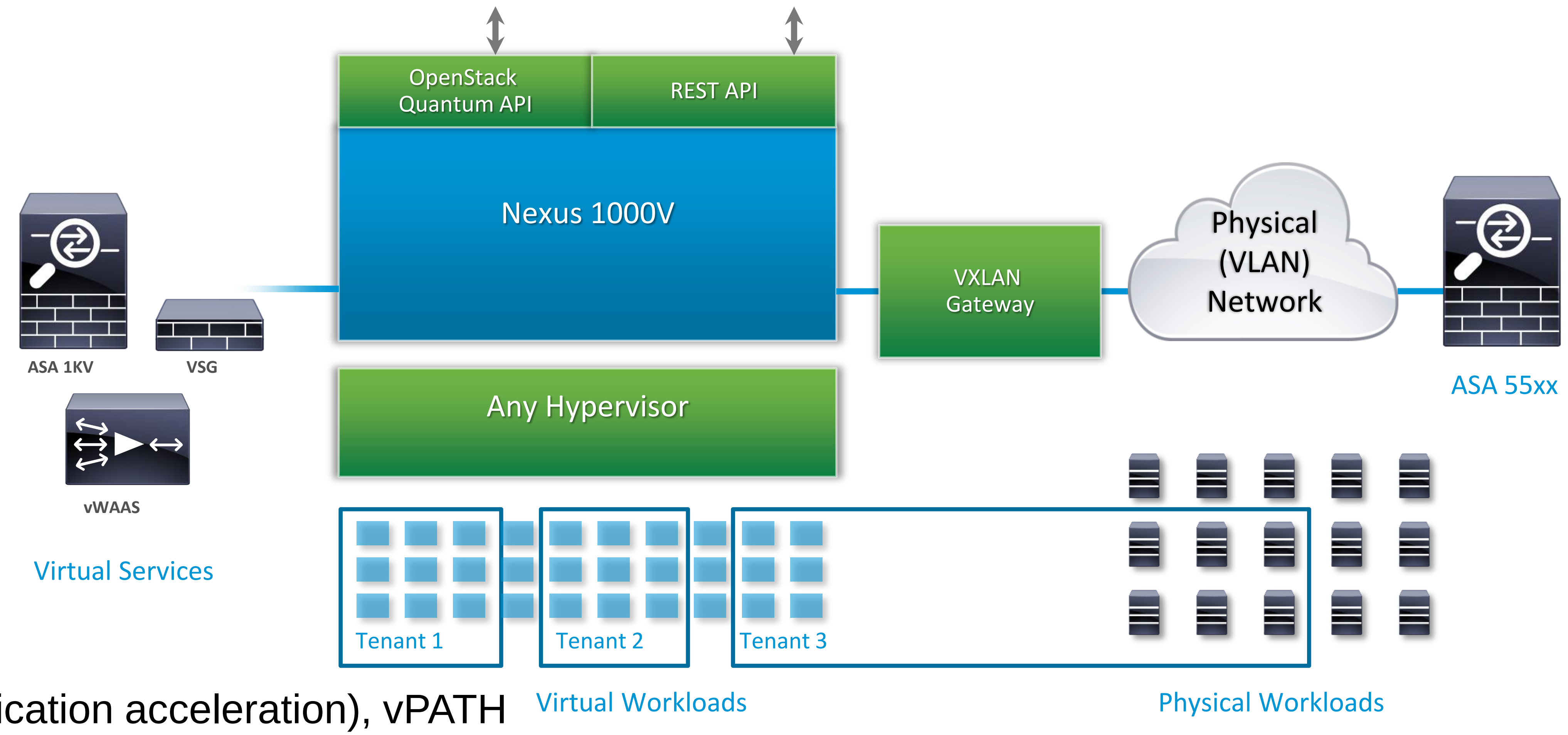
Switching	Nexus 7K/5K/3K/2K	Nexus 1000V, VM-FEX
Routing	ASR, ISR	Cloud Services Router (CSR 1000V)
Services	WAAS, ASA, NAM	vWAAS, VSG, ASA 1000V, vNAM**
Compute	UCS for Bare Metal	UCS for Virtualised Workloads

** 1H 2013

Virtual Overlay Networks

Example: Virtual Overlay Networks and Services with Nexus 1000V

- Large scale L2 domains:
  Tens of thousands of virtual ports
- Common APIs
  – Incl. OpenStack Quantum API's for orchestration
- Scalable DC segmentation and addressing
  – VXLAN
- Virtual service appliances and service chaining/traffic steering
  – VSG (cloud-ready security), vWAAS (application acceleration), vPATH
- Multi-hypervisor platform support: ESX, Hyper-V, OpenSource Hypervisors
- Physical and Virtual: VXLAN to VLAN Gateway



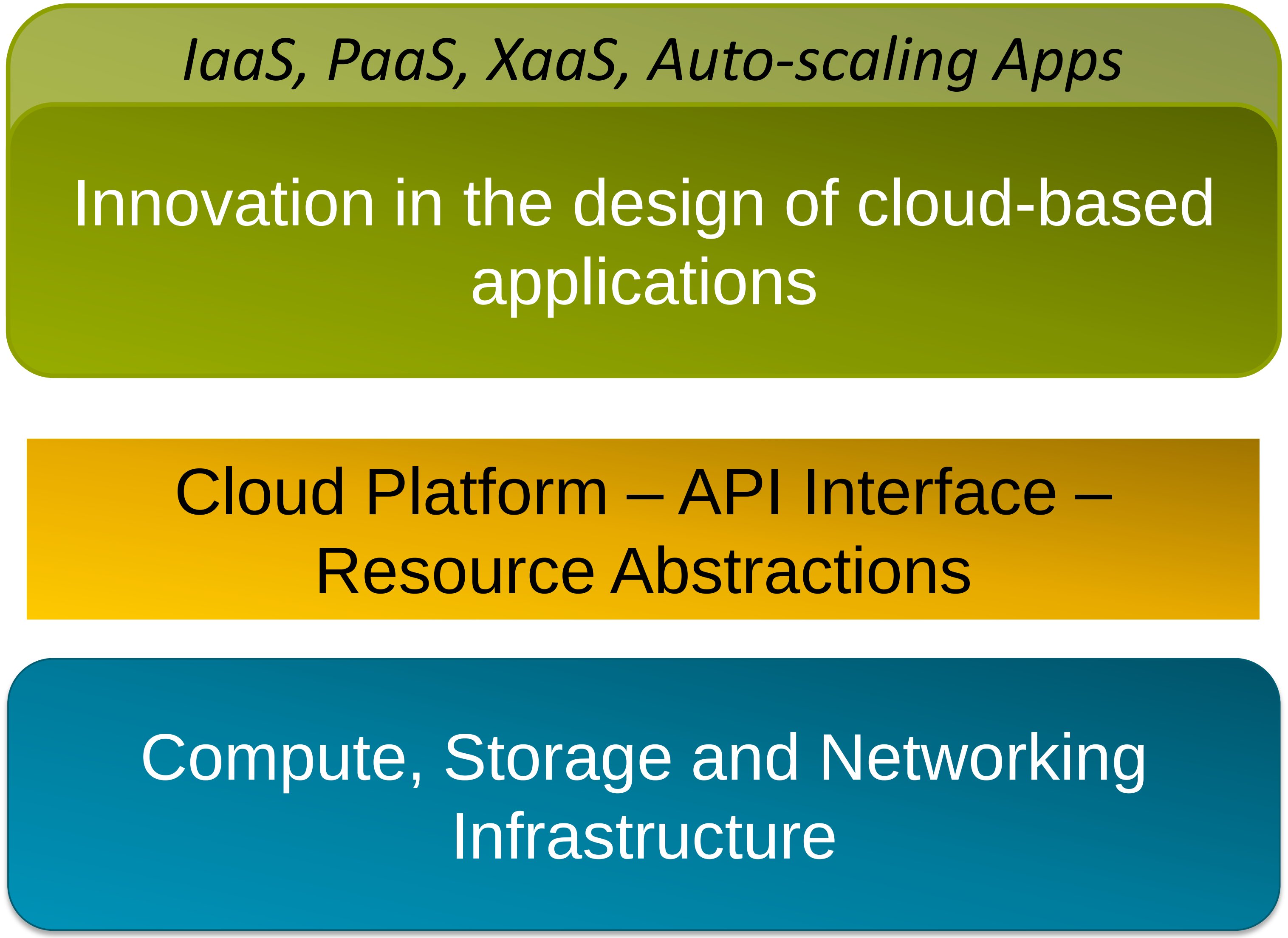
Cloud technology stacks

Multi-Hypervisor and Multi-Orchestration

Cloud Portal & Orchestration	Multi-orchestration
Virtual Network Infrastructure	Virtual Service Appliances
	Virtual Switch / Virtual Router
Hypervisor	Multi-Hypervisor
Compute Platform	UCS
Physical Network	Unified Fabric & Service Appliances
Storage Platform	Storage Eco-System

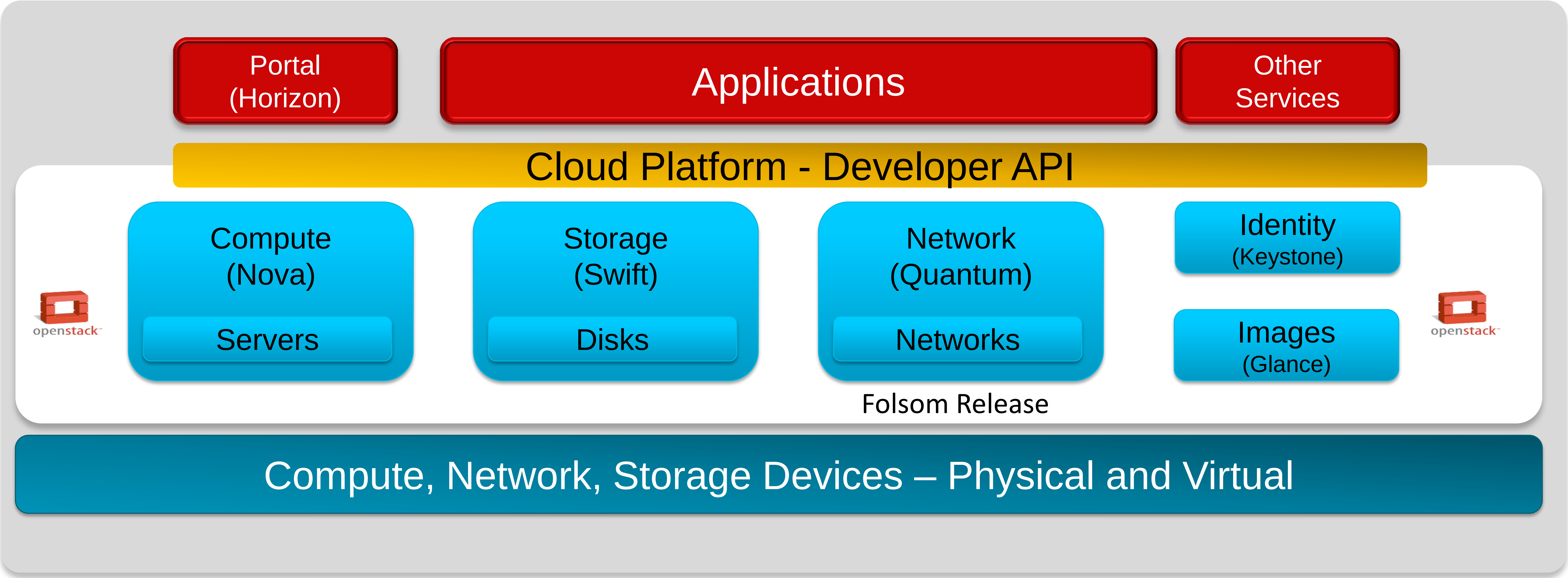
Network Service becomes a first class citizen in cloud computing and automation

- Enable full automation of Infrastructure Provisioning and Control – including the Network
  - Cloud Automation: Automation of Compute, Network, Storage resources
- Apply to automate all types of networks: physical devices, virtual devices, overlay/non-overlay networks
  - Orthogonal to whether SDN concepts are leveraged



Network Service becomes a first class citizen

Example: OpenStack with Quantum for Network Automation



Openstack is for infrastructure automation – orthogonal to whether SDN concepts are applied

Cisco & openstack

2010

July 2010
Openstack Launched
Approx. 20 Companies participated

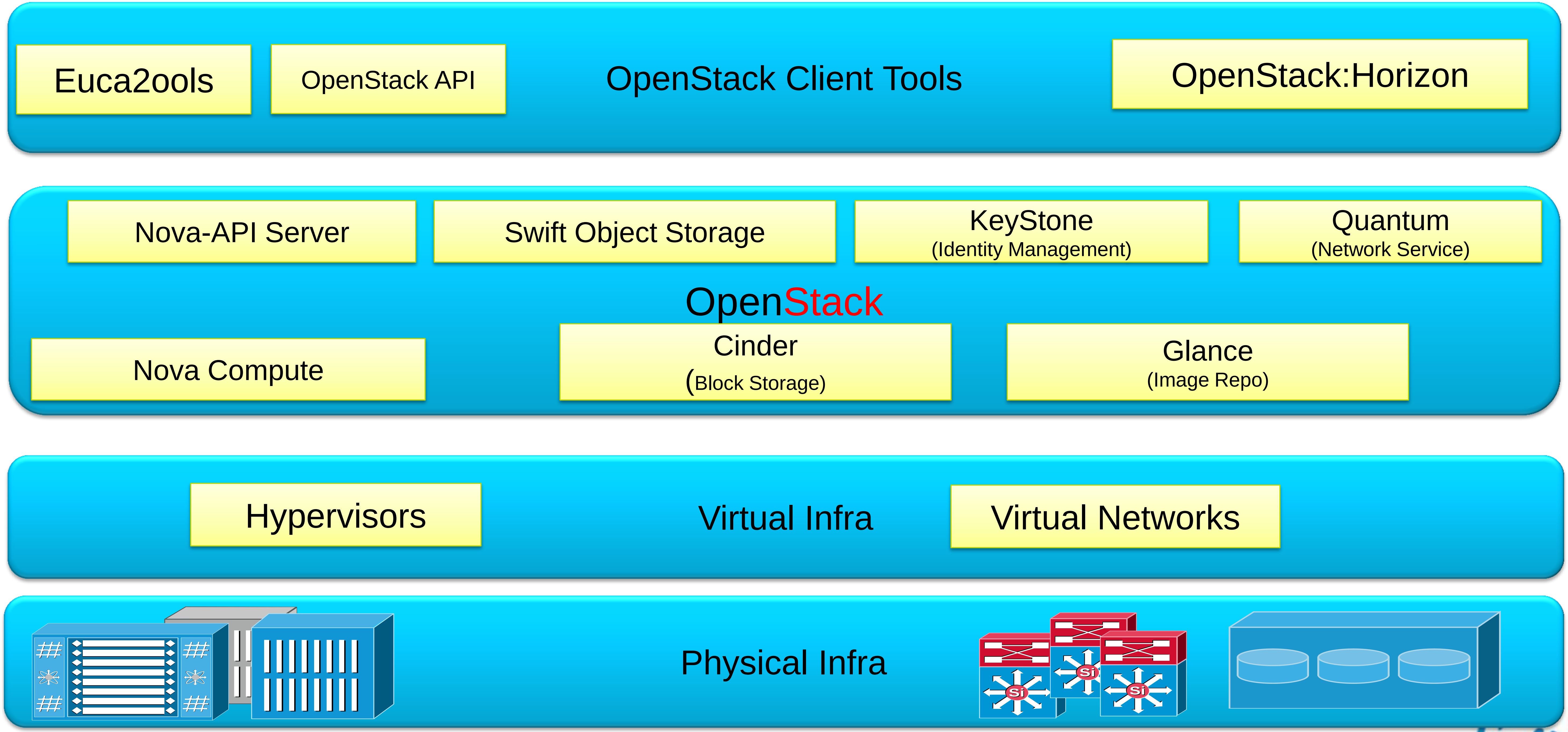
2011

- **Feb 2011**: Cisco Announces Openstack membership
- **Apr 2011**: Collaborates with Community Members for Quantum
- **Oct 2011**: Diablo Release – Cisco Contributes Code
- More than 80 companies participated

2012

- **Apr 2012**: More than 165 Companies participation
- More than 3000 Developers
- **Sep 2012** : "Folsom" Release

OpenStack

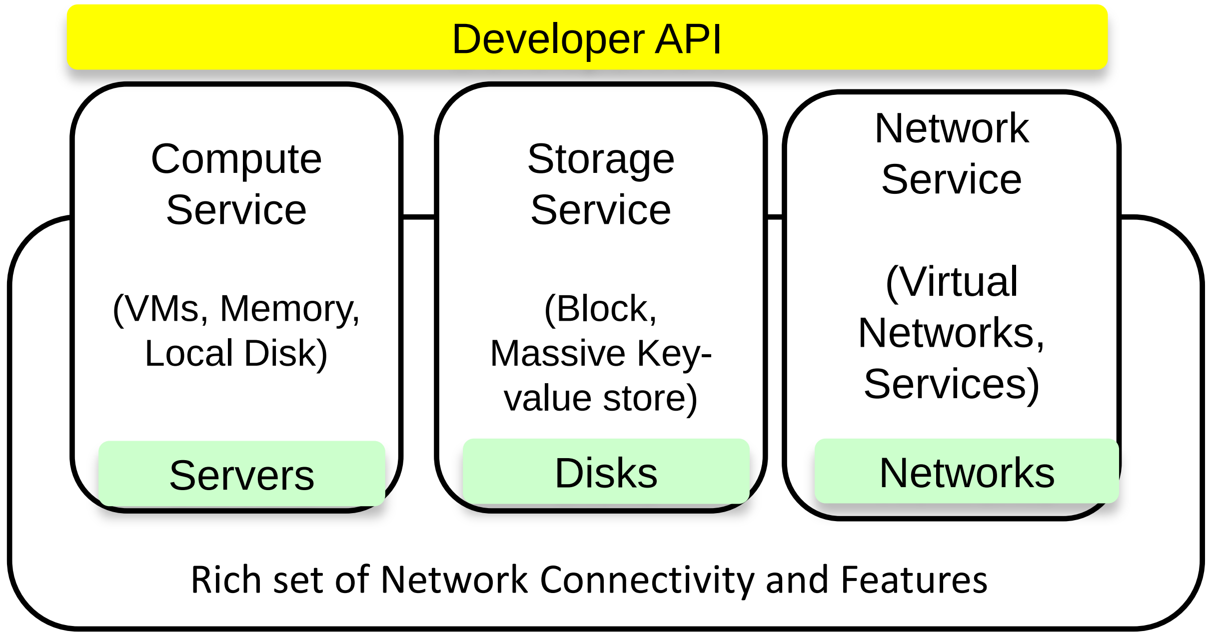

Euca2ools
OpenStack API
OpenStack Client Tools
OpenStack:Horizon
Nova-API Server
Swift Object Storage
KeyStone
(Identity Management)
Quantum
(Network Service)
OpenStack
Nova Compute
Cinder
(Block Storage)
Glance
(Image Repo)
Hypervisors
Virtual Infra
Virtual Networks
Physical Infra

Cisco's Contributions

Quantum – Network-as-a-Service

Quantum**: Network-as-a-Service*

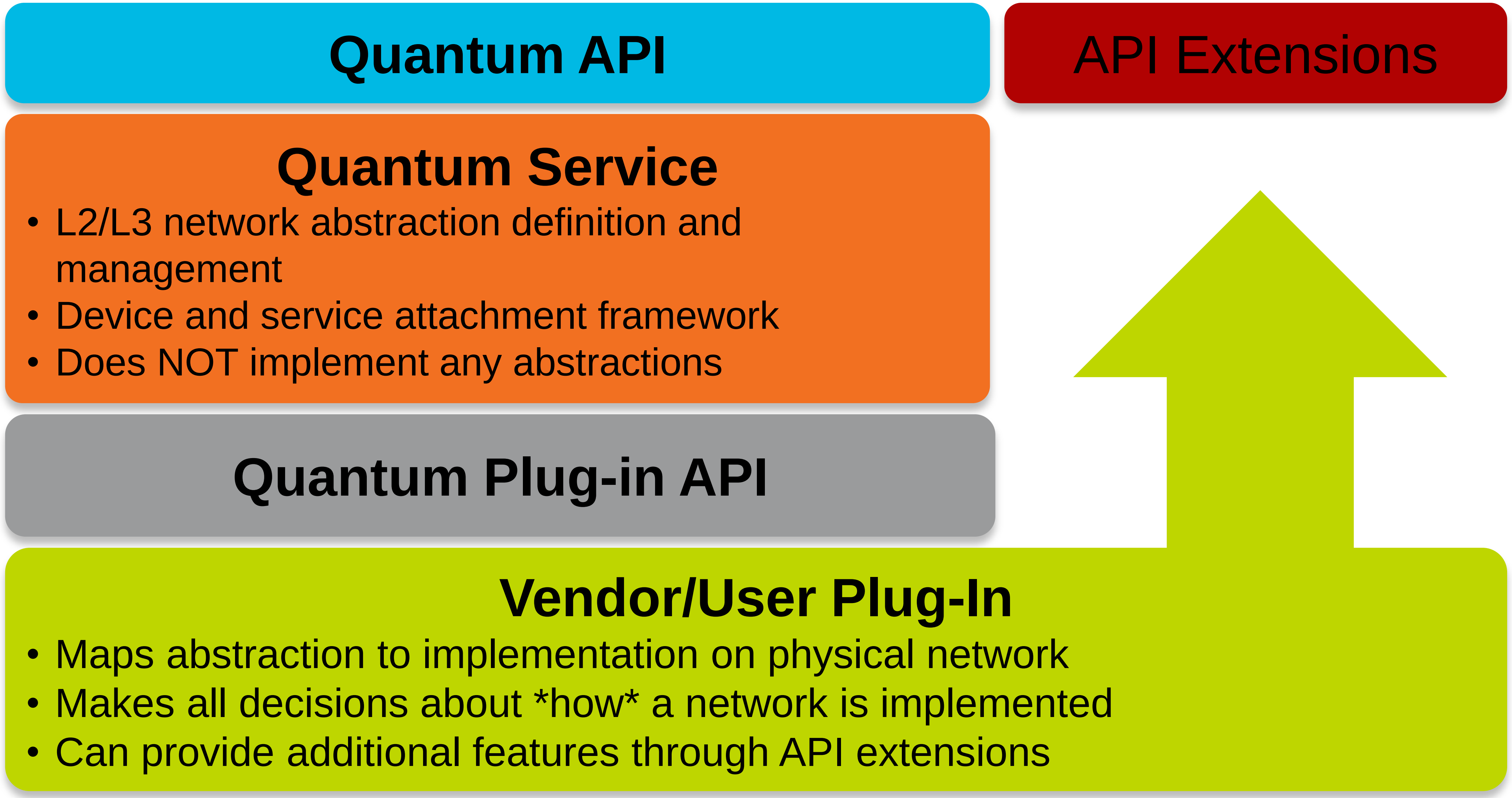
As a peer to compute and storage



* Proposed Network Services Framework in SantaClara Openstack Design Summit April 2011
** Collaborated with Openstack Community Partners in defining Quantum

Quantum Architecture

Extensible allowing vendor specific capabilities



Quantum Abstractions

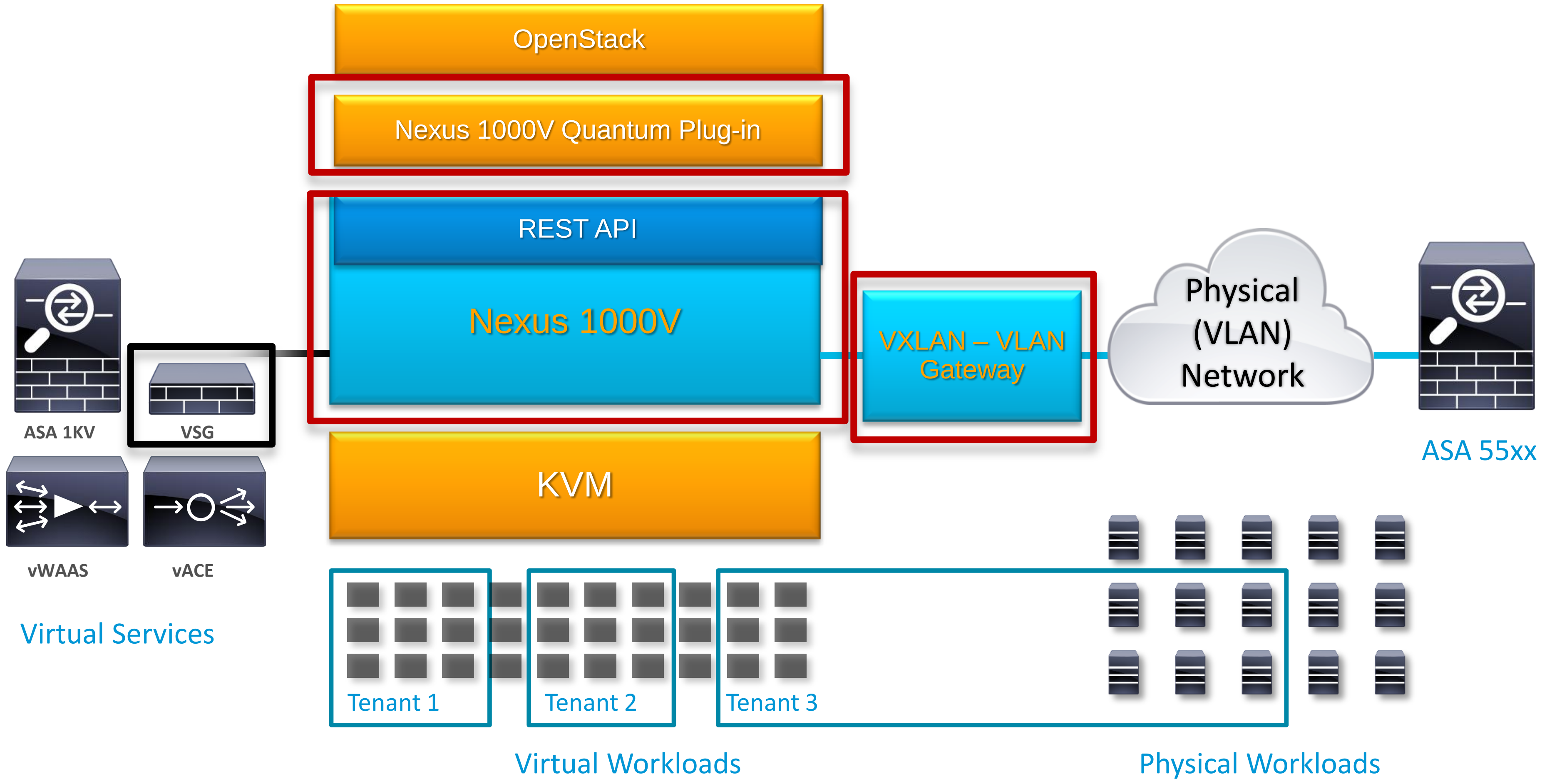
Enables extensibility

- **Virtual Networks**
  - A basic dedicated L2 network segment
  - Common realisation is a VLAN
- **Virtual Ports**
  - Attachment point for devices connecting to virtual networks.
  - Ports expose configuration and monitoring state via extensions (e.g., ACLs, QoS policies, Port Profiles, Packet Statistics)
- **Subnets**
  - IPv4 or IPv6 address blocks for VMs

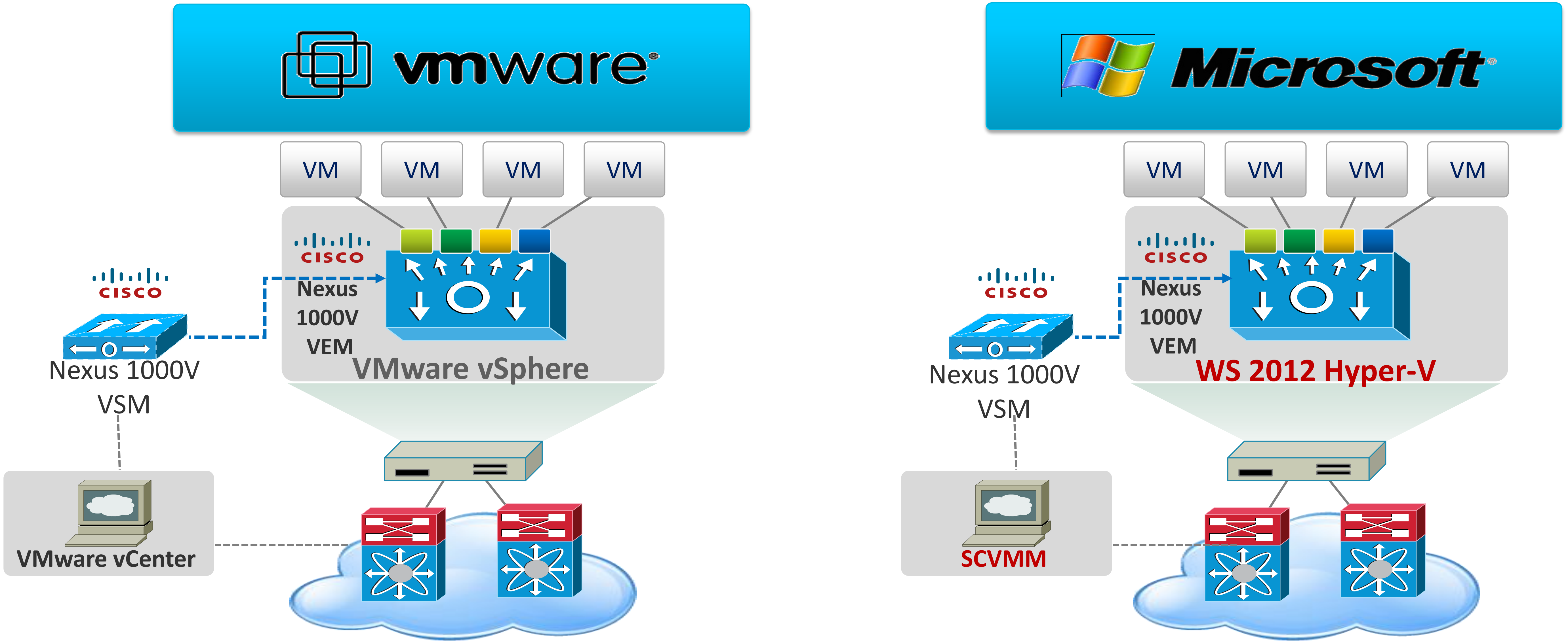
Reference: Quantum API Spec http://wiki.openstack.org/Quantum/APIv2-specification

Nexus 1000V & openstack

Quantum enables provisioning and orchestration of Nexus 1000V



Cisco Nexus 1000V for HyperV



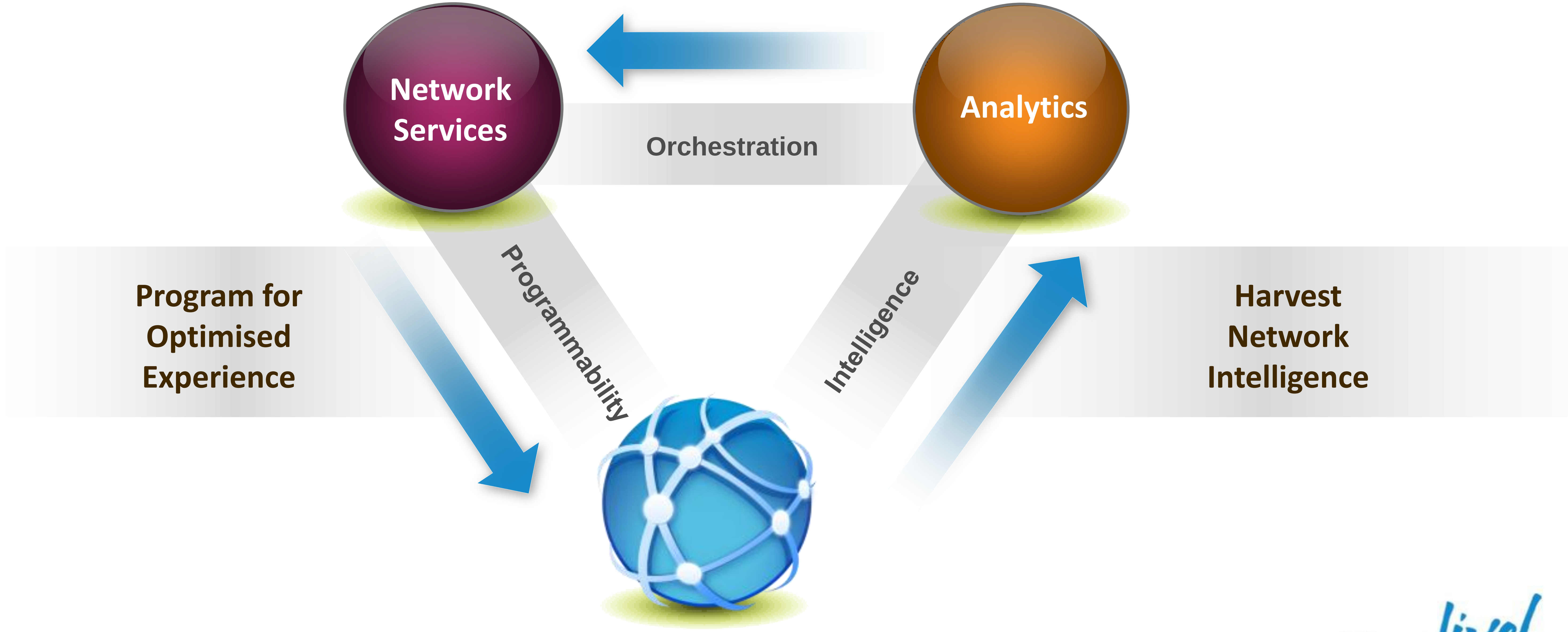
Consistent architecture, feature-set & network services ensures operational transparency across multiple hypervisors.

Open Network Environment

Broad-based Network Programmability beyond SDN

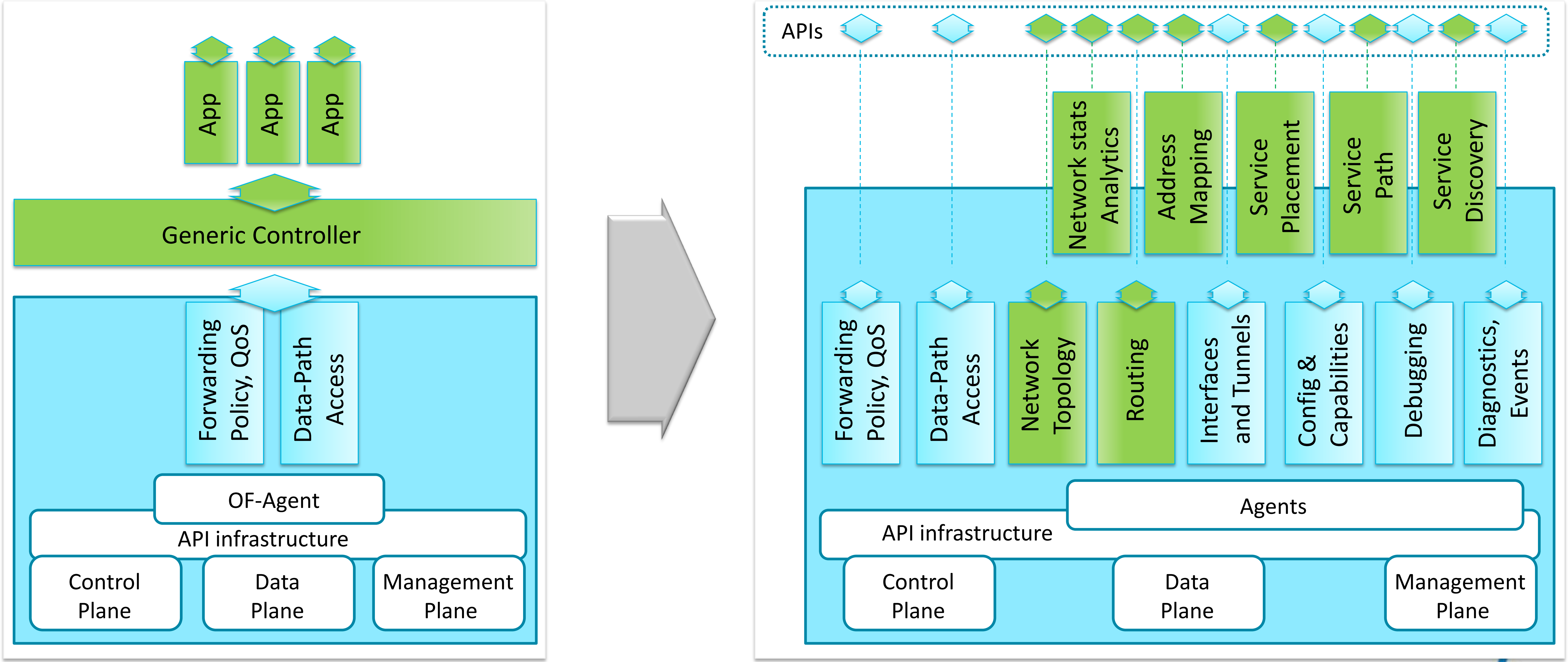
Summary: Open Network Environment

Leverage Network Value



Evolve the early SDN Model…

… acknowledge the need for diverse abstractions



Programmatic Network Control and the Gartner Hype Cycle

Technology Readiness Phases

Phase 1

Academic research (“OpenFlow”)

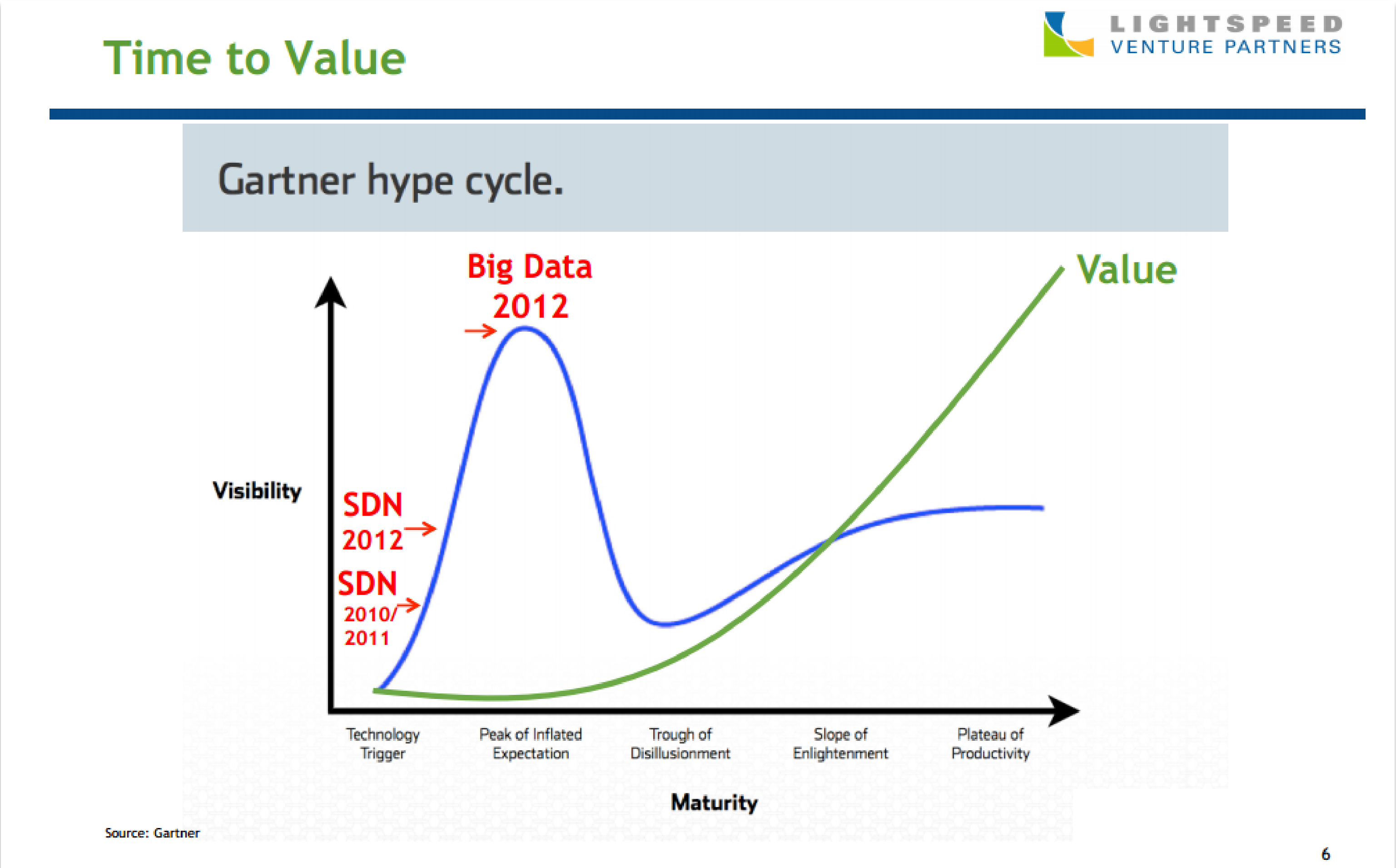
Phase 2

Broad interest from technical community + use cases (“SDN”)

Phase 3

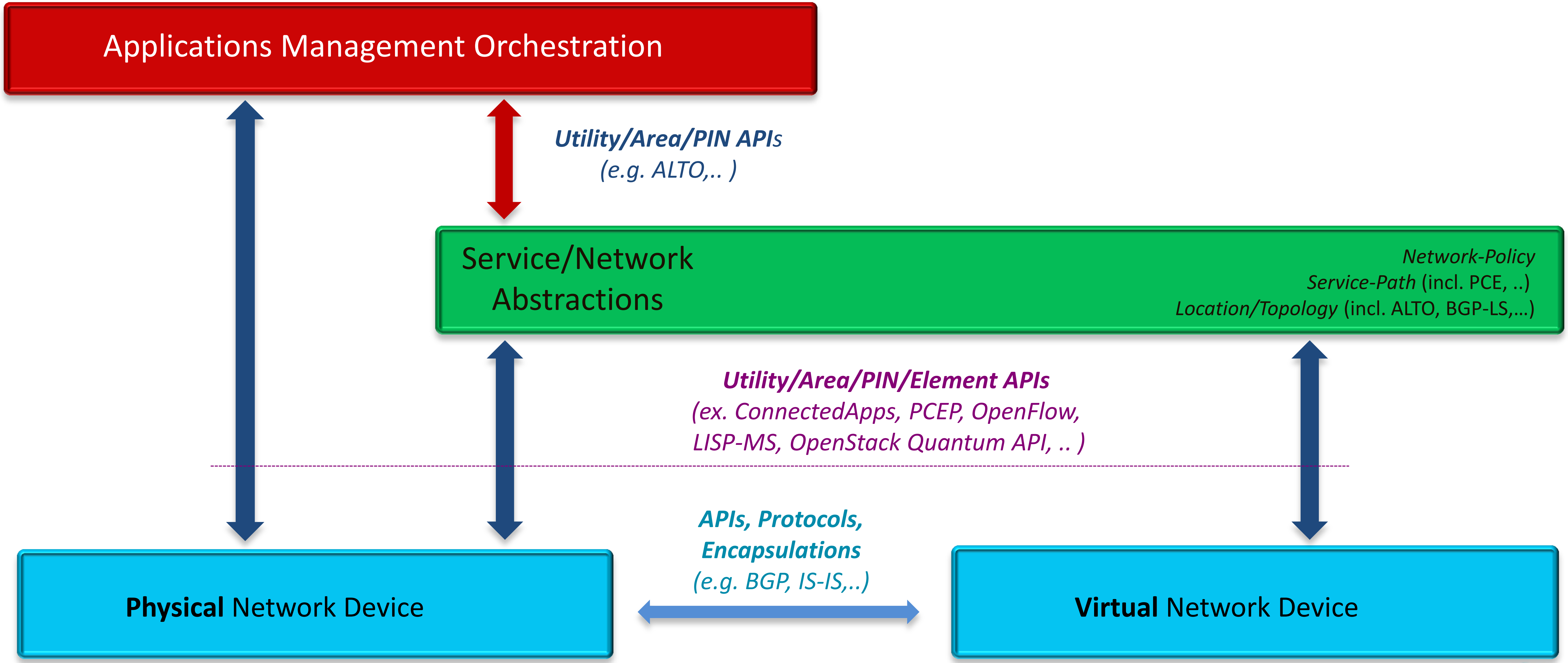
Value Proposition for mass market understood

Entering Phase 3…



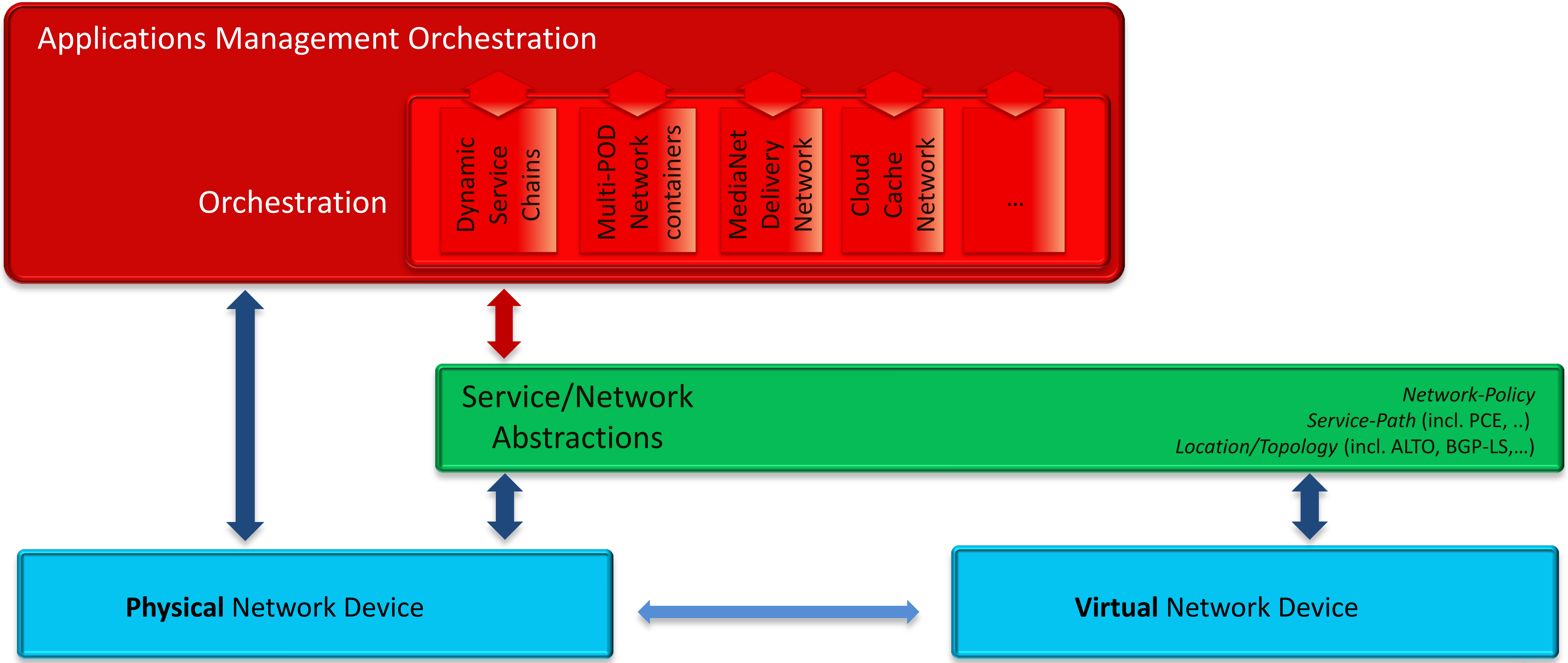
Source: John Vrionis, LightSpeed Venture, opennetsummit.org/talks/ONS2012/vrionis-wed-closing.pdf

Open Network Environment for SDN



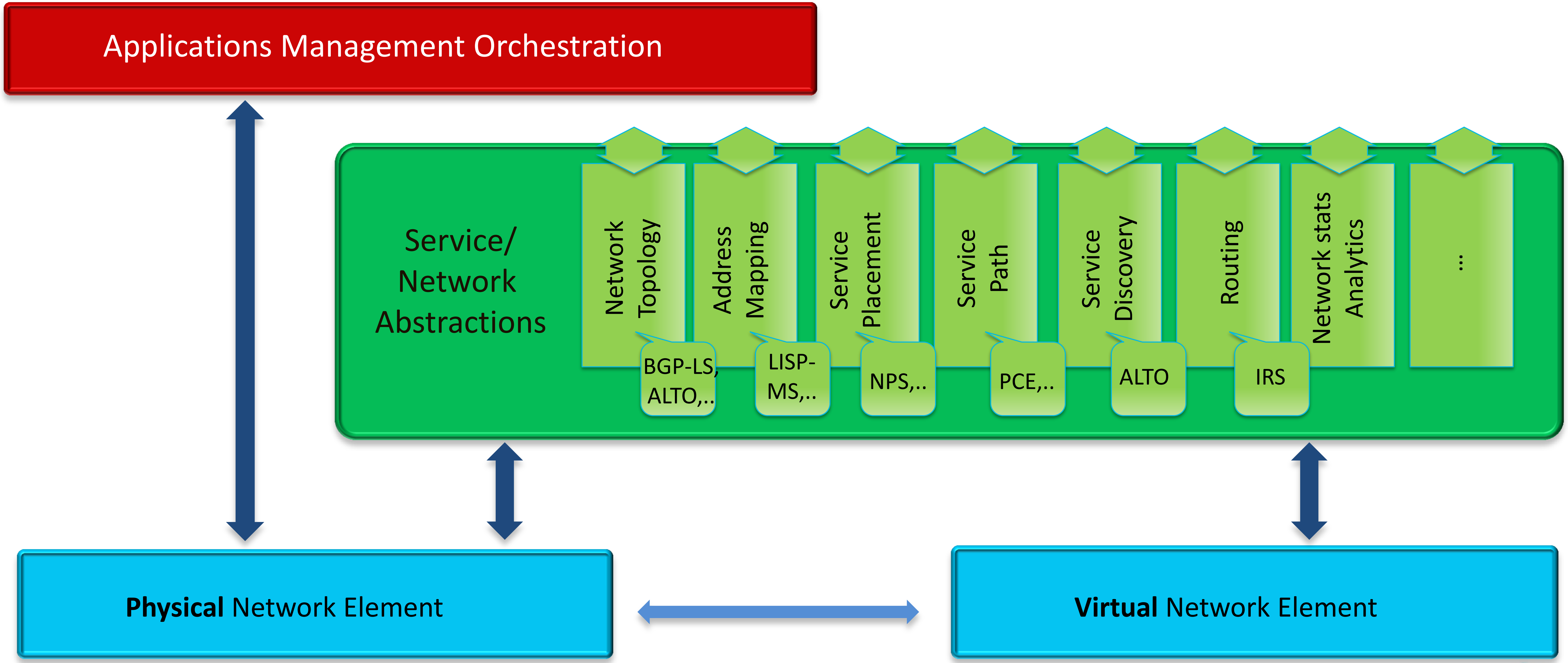
Open Network Environment for SDN

Focus: Network/Service Abstractions and associated APIs



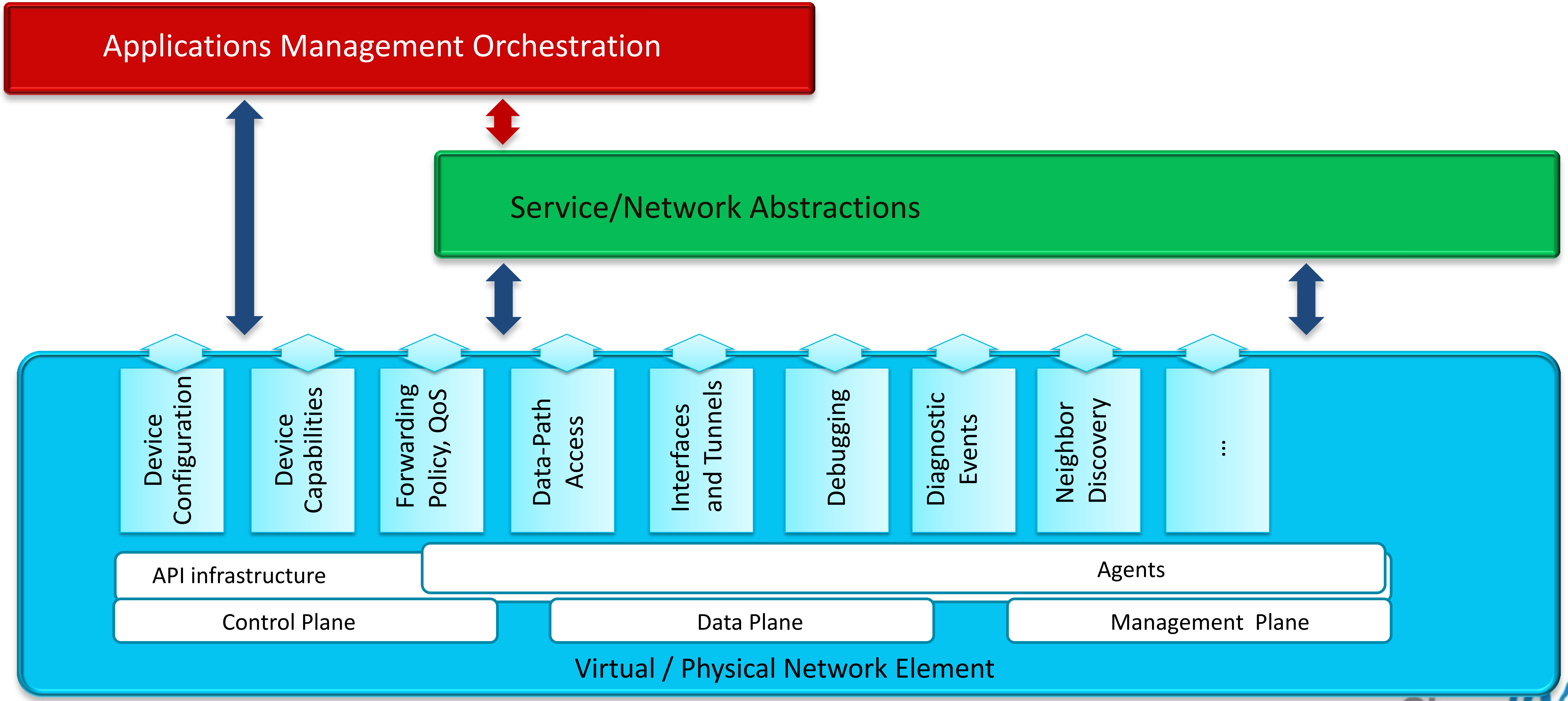
Open Network Environment

Focus: Network/Service Abstractions and associated APIs



Open Network Environment for SDN

Focus: Device-level abstractions and associated APIs



Industry Standards & Forums

802.1 Overlay Networking Projects

Open Network Research Centre at Stanford University

Technical Advisory Group, Working Groups:
Config, Hybrid, Extensibility, Futures/FPMOD/OF2.0

Initiatives:
Quantum
Donabe

Open Source Cloud Computing project

Overlay Working Groups:
NVO3, L2VPN, TRILL, L3VPN, LISP, PWE3
API Working Groups/BOFs
NETCONF, ALTO, CDNI, XMPP, SDNP, I2AEX
Controller Working Groups:
PCE, FORCES
New work items:
IRS – Interface to the Routing System

Summary

Summary: Open Network Environment

Cisco Innovations Summary announced at Cisco Live San Diego 2012

onePK Developer Kit

- Complete developer's kit for multiple Cisco Platforms, Servers, Blades
- Rapidly develop test and deploy Applications.
- Phased availability across IOS, IOS-XR and NX-OS platforms

Programmatic APIs

Controllers + Agent Support

- Engage with universities & research for campus slicing use case
- OpenFlow 1.x support on select Cisco platforms
- Controller SW

Controllers and Agents

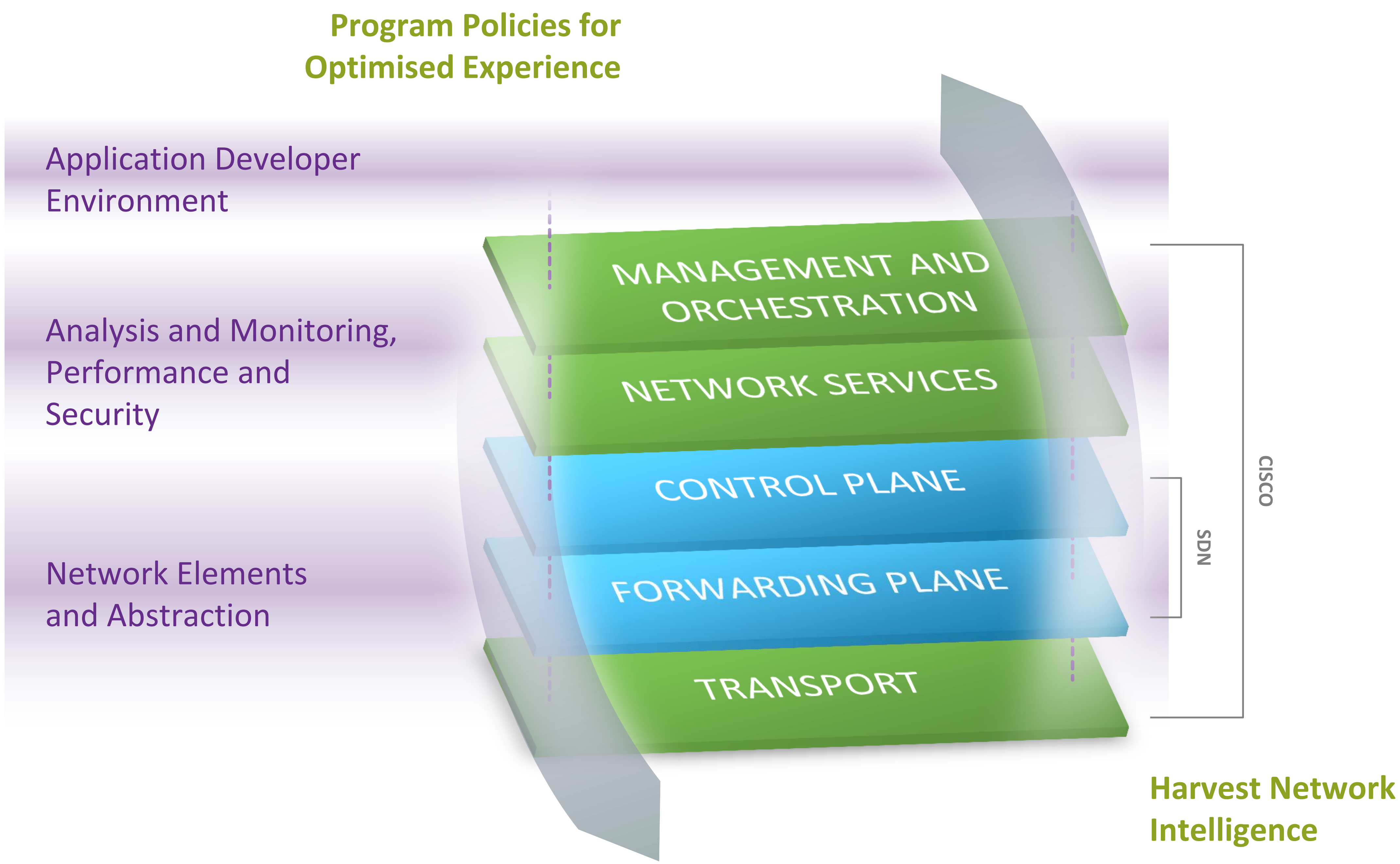
Overlay Network Solutions

- 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

Cisco Vision: Exposing The Entire Network Value

Programmatic Control across Multiple Network Planes



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

Open Network Environment – Summary

The Industry's Broadest Approach to Programmatic Access to the Network

- Evolutionary step for networking:
  Complement/evolve the Network Control Plane where needed
- Centred around delivering open, programmable environment for real-world use cases
  - No one-size-fits-all
  - Cisco will support Network Virtualisation, APIs and Agents/Controllers
  - Joint evolution with industry and academia
- Technology-agnostic
  - Not predicated on a particular technology or standard
  - Draw from Cisco technologies and industry standards
- Delivered as incremental functionality
  - Many customers will use hybrid implementations
  - Build upon existing infrastructure with investment protection

Open Network Environment
www.cisco.com/go/one

onePK
www.cisco.com/go/onepk
www.cisco.com/go/getyourbuildon

Q & A

Complete Your Online Session Evaluation

Give us your feedback and receive a Cisco Live 2013 Polo Shirt!

Complete your Overall Event Survey and 5 Session Evaluations.

- Directly from your mobile device on the Cisco Live Mobile App
- By visiting the Cisco Live Mobile Site www.ciscoliveaustralia.com/mobile
- Visit any Cisco Live Internet Station located throughout the venue

Polo Shirts can be collected in the World of Solutions on Friday 8 March 12:00pm-2:00pm

Don't forget to activate your Cisco Live 365 account for access to all session material, communities, and on-demand and live activities throughout the year. Log into your Cisco Live portal and click the "Enter Cisco Live 365" button.
www.ciscoliveaustralia.com/portal/login.ww