TOMORROW starts here.

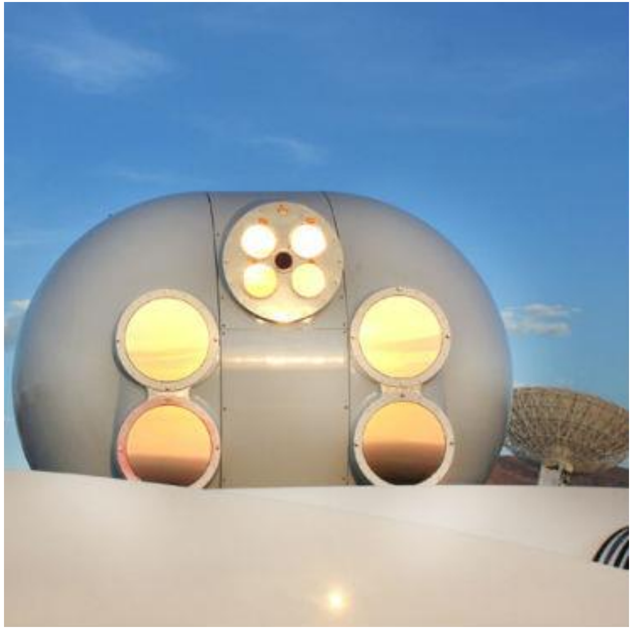
Software Defined Networks

BRKRST-2051

Alistair Crawford
Systems Engineer

Networking

Why I love IT !



Software Defined Networking – SDN

A little skeptical

What is SDN?

Software Defined Networking

> "…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…"

OPEN NETWORKING
FOUNDATION

---

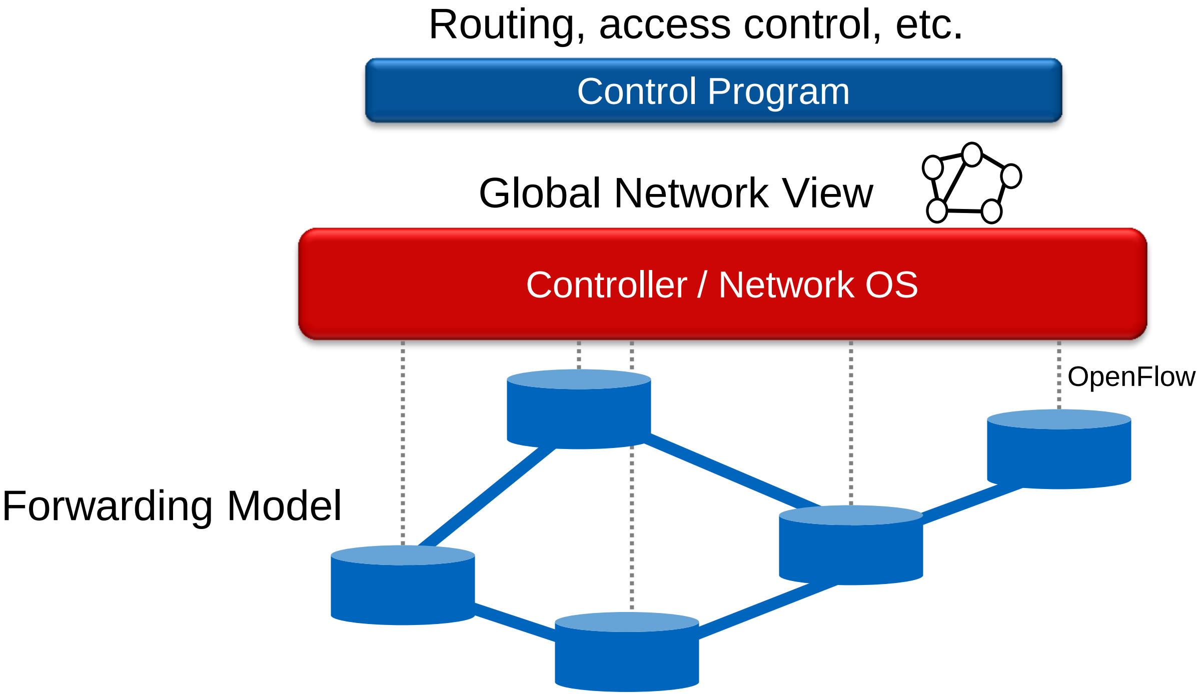
https://www.opennetworking.org/images/stories/downloads/white-papers/wp-sdn-newnorm.pdf

"…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……"

OpenFlow™

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

Original SDN Architecture



Routing, access control, etc.

Control Program

Global Network View

Controller / Network OS

OpenFlow

Forwarding Model

What is SDN **for you**?

Why SDN?

What is SDN to you?

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

"A platform for developing new control planes"

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

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

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

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

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

Diverse Drivers

"A means to do traffic engineering without MPLS"

Common Concepts

"A way to scale my firewalls and load balancers"

Different Execution Paths

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

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

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

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

Classes of Use-Cases

"Leveraging APIs and logically centralised control plane components"

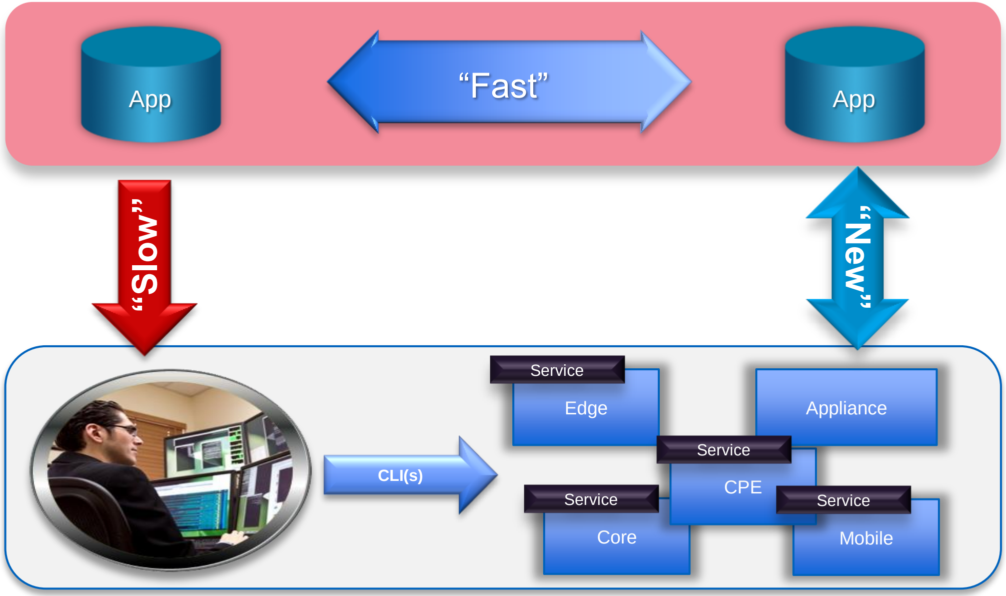
SDN origin

- 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; IaaS + MPLS-VPN)
- Federating different Network Control Points
  (LAN-WAN, DC-WAN, Virtual-Physical, Layer-1-3)

Automation of
Network Control
and Configuration
(Fulfillment and Assurance)
Virtual & Physical

Programmatic Interfaces to the Network

Today's Application Dilemma



A New Programming Paradigm Is Needed

Re-assessing the Network Control Architecture

Evolving Design Constraints on the Control Plane

Generic Network
"Internet"

Operate w/o communication guarantees
distributed system with arbitrary failures,
nearly unbounded latency,
highly variable resources,
unconstrained topologies

Optimise for reliability

Domain specific networks
(DC, SP Access/Agg, Branch, ..)

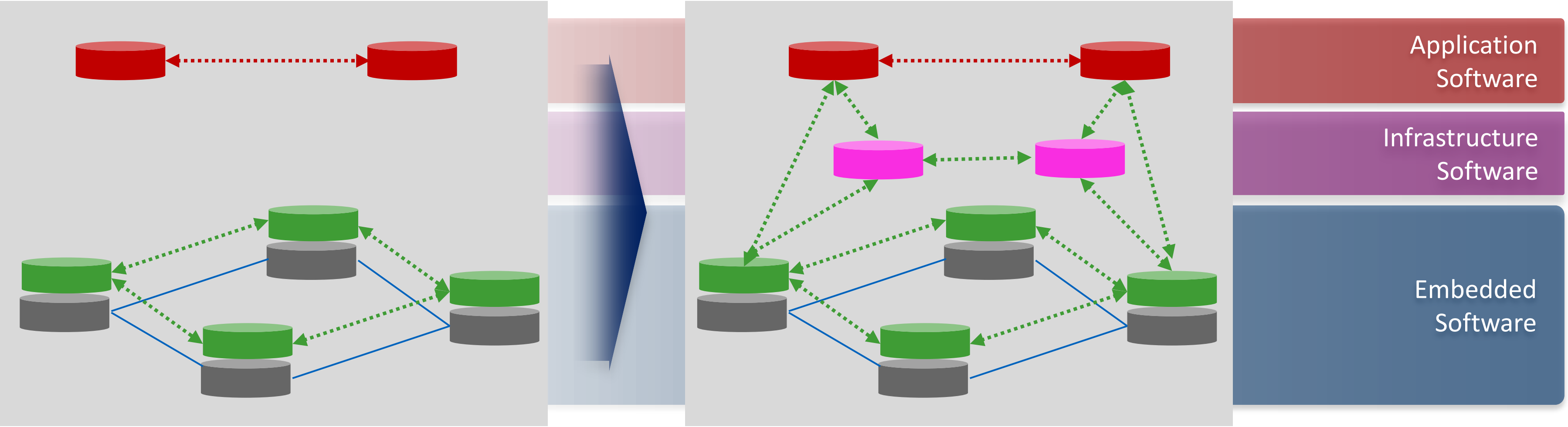
Domain specific qualities of these networks relax or evolve network design constraints
Well defined topologies,
little variety in network device-types,
no arbitrary changes in connected end-hosts, ..

Optimised for reliability *and* domain specific performance metrics

Solutions for domains differ:
DC != WAN, TOR != PE

Towards an Open Network Environment

Evolve the Control- and Management Plane Architecture

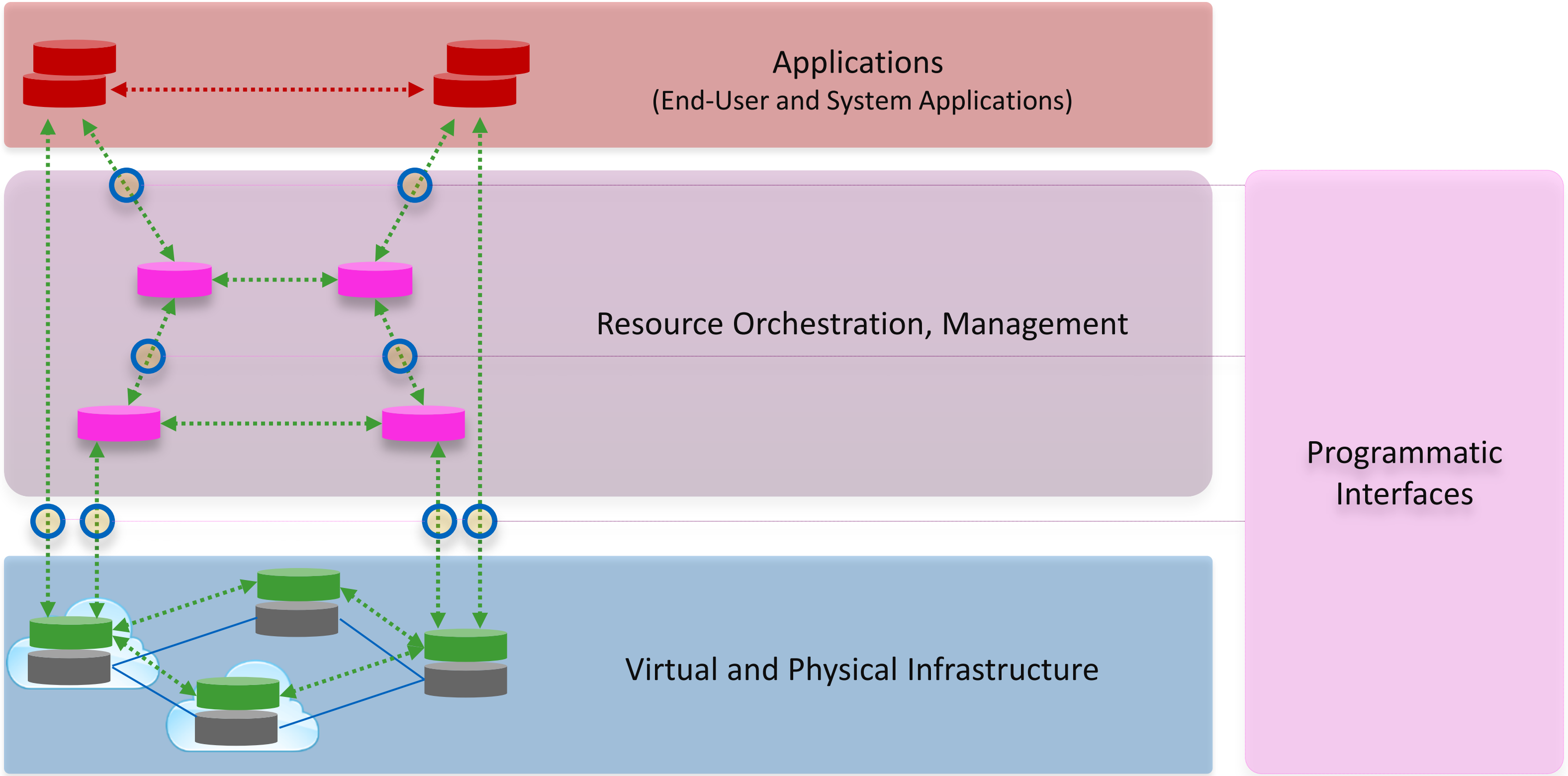


Fully Distributed Control Plane:
Optimised for reliability

Hybrid Control plane:
Distributed control combined with logically centralised control for optimised behaviour
(e.g. reliability and performance)

Open Network Environment

Approaching a definition



Applications
(End-User and System Applications)

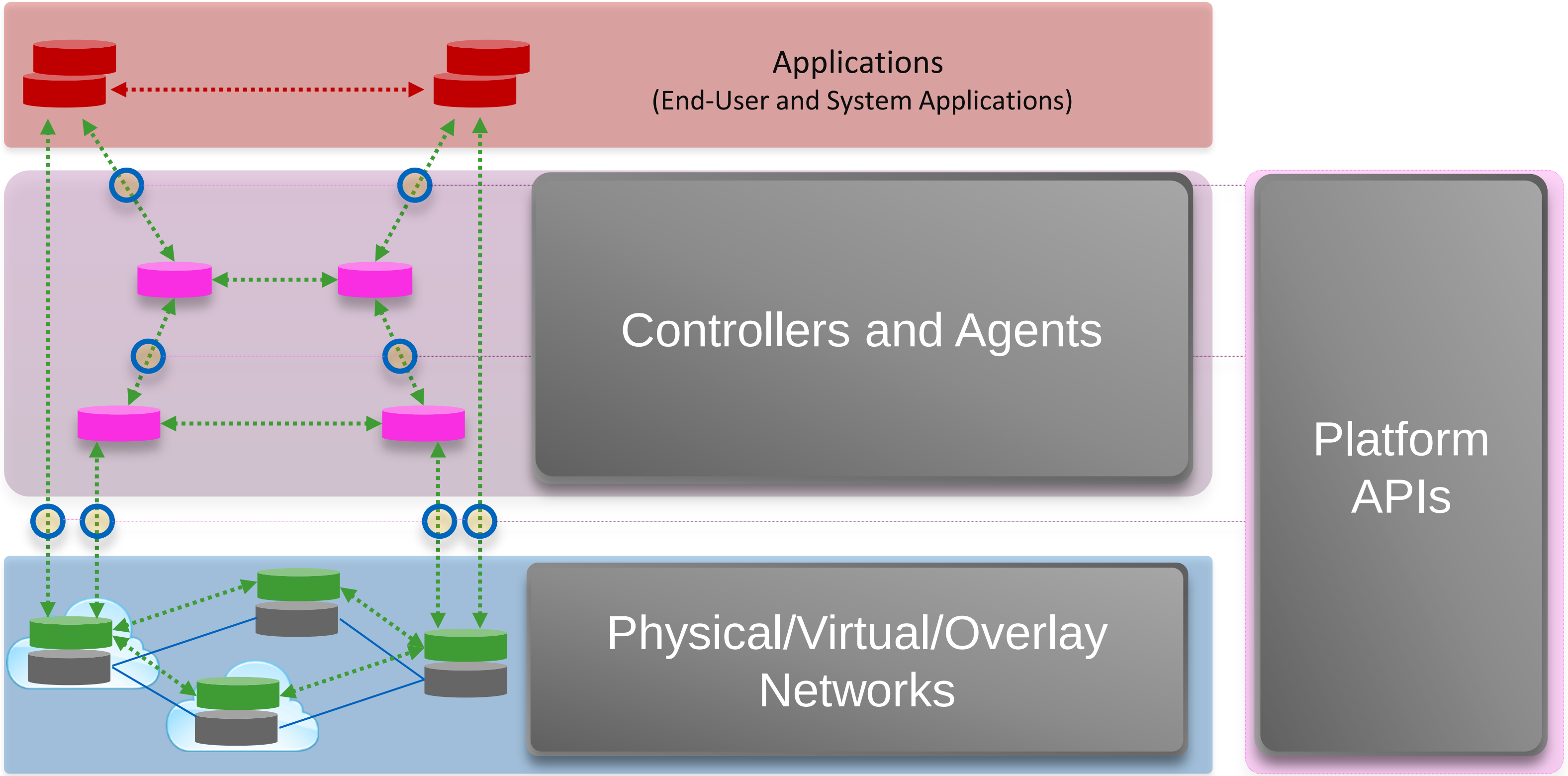
Resource Orchestration, Management

Virtual and Physical Infrastructure

Programmatic Interfaces

Open Network Environment

Introduced At Cisco Live San Diego in June 2012



Applications
(End-User and System Applications)

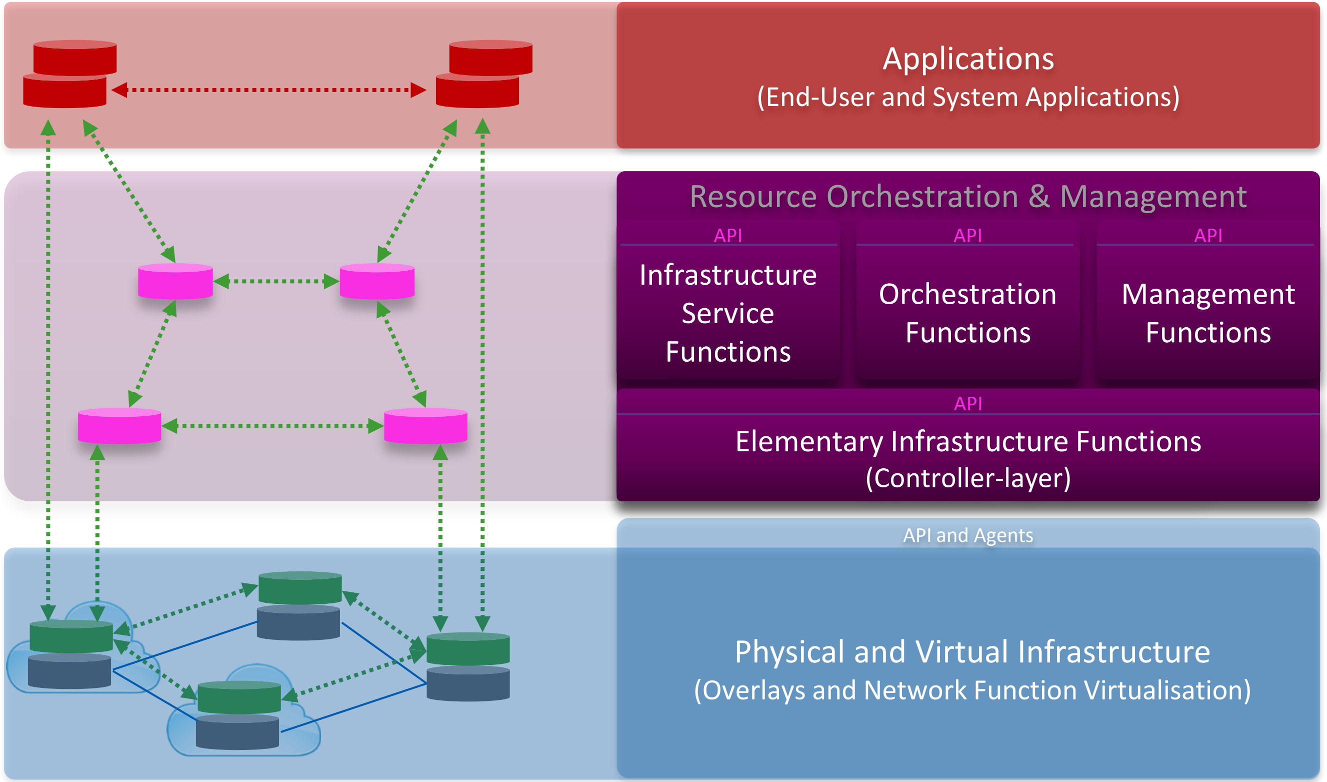
Controllers and Agents

Platform APIs

Physical/Virtual/Overlay Networks

Open Network Environment

The Next Step: Infrastructure Software Platform



Applications
(End-User and System Applications)

Application Software

Resource Orchestration & Management

API	API	API
Infrastructure Service Functions	Orchestration Functions	Management Functions

API

Elementary Infrastructure Functions
(Controller-layer)

Infrastructure Software

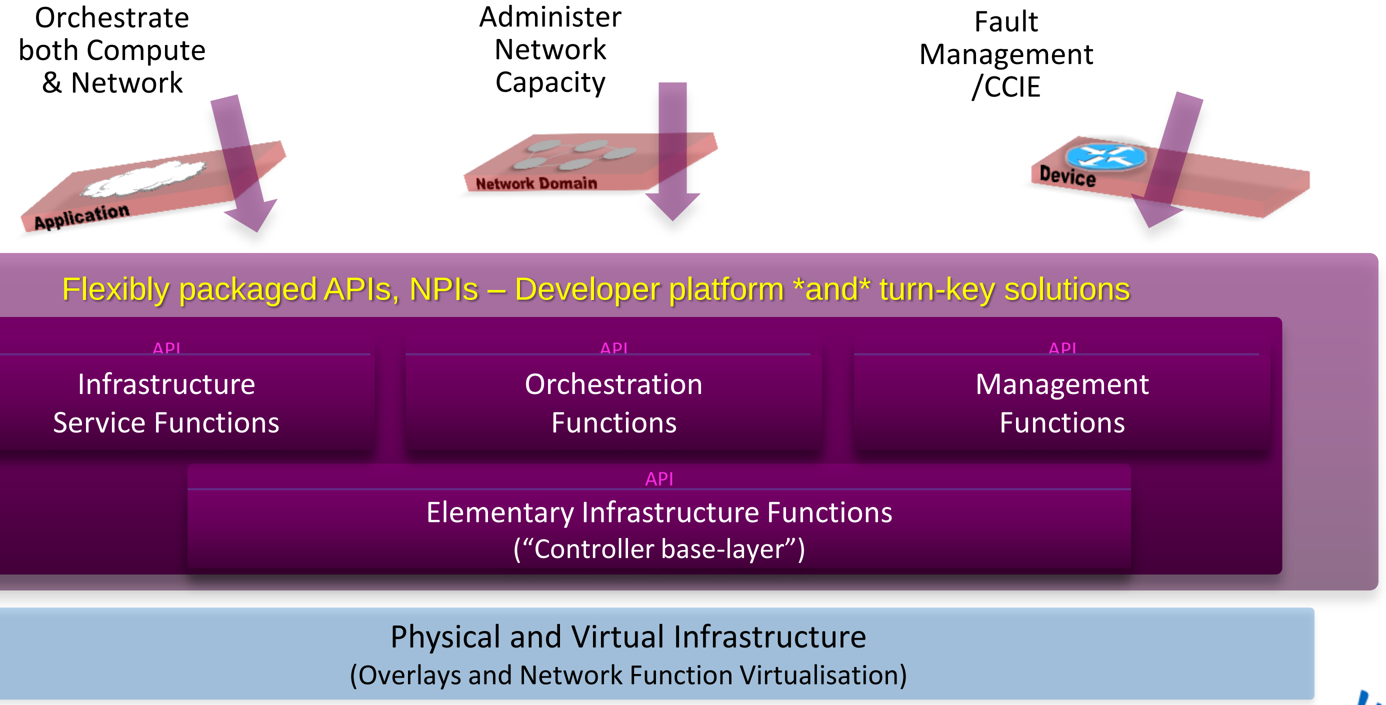
API and Agents

Physical and Virtual Infrastructure
(Overlays and Network Function Virtualisation)

Embedded Software

Open Network Environment and Unified Platform

Serving Operations Constituencies



Orchestrate
both Compute
& Network

Administer
Network
Capacity

Fault
Management
/CCIE

Flexibly packaged APIs, NPIs – Developer platform *and* turn-key solutions

Cisco Unified Platform

API
Infrastructure
Service Functions

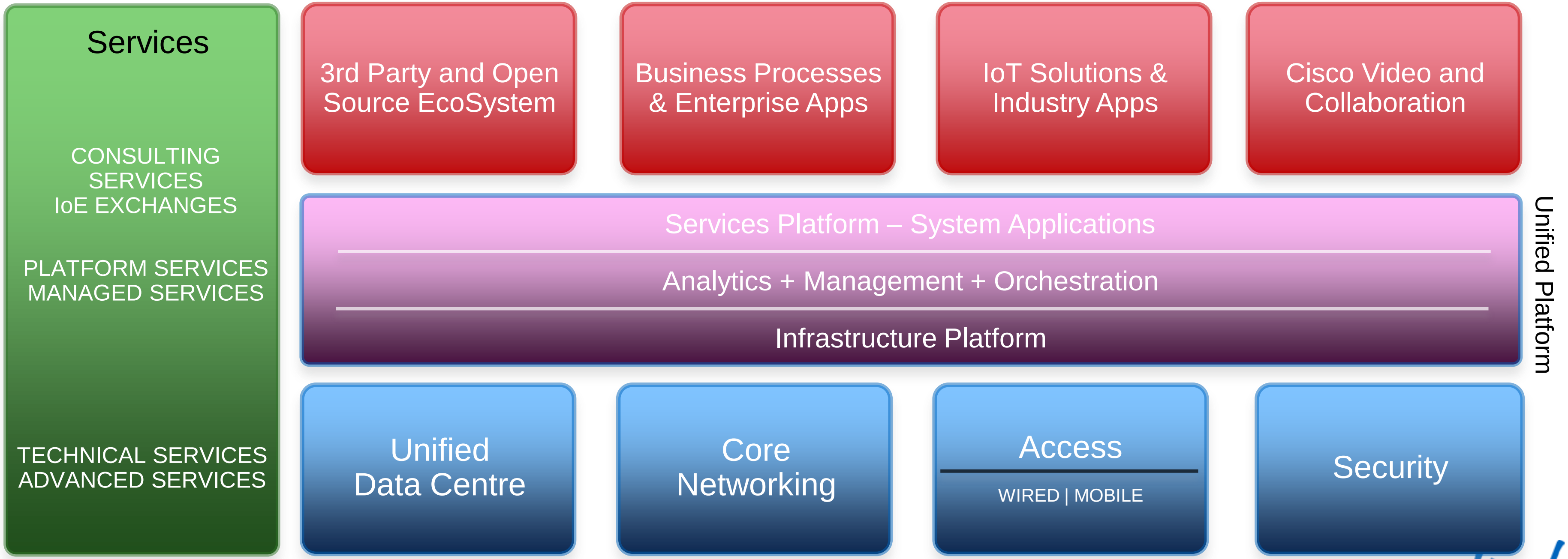
API
Orchestration
Functions

API
Management
Functions

API
Elementary Infrastructure Functions
(“Controller base-layer”)

Physical and Virtual Infrastructure
(Overlays and Network Function Virtualisation)

Cisco Unified Framework



Services

CONSULTING
SERVICES
IoE EXCHANGES

PLATFORM SERVICES
MANAGED SERVICES

TECHNICAL SERVICES
ADVANCED SERVICES

3rd Party and Open Source EcoSystem

Business Processes & Enterprise Apps

IoT Solutions & Industry Apps

Cisco Video and Collaboration

Services Platform – System Applications

Analytics + Management + Orchestration

Infrastructure Platform

Unified Platform

Unified Data Centre

Core Networking

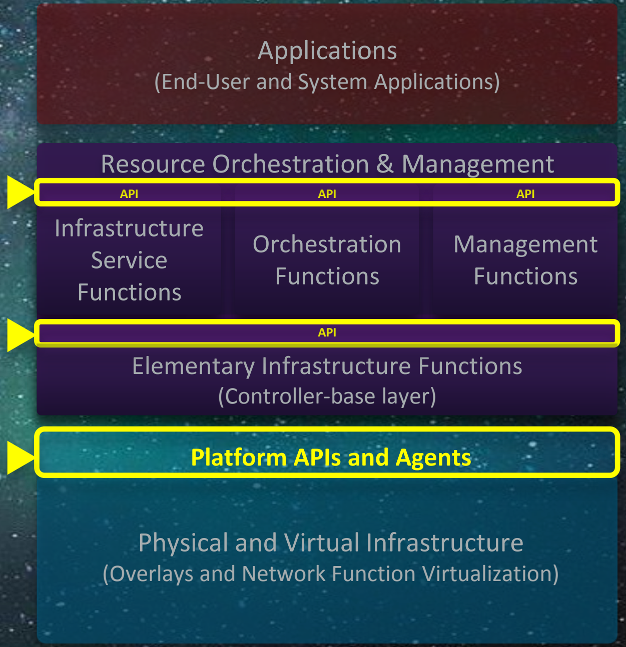
Access

WIRED | MOBILE

Security

Open Network Environment Qualities

Programmatic APIs



Applications

(End-User and System Applications)

Resource Orchestration & Management

API API API

Infrastructure Service Functions

Orchestration Functions

Management Functions

API

Elementary Infrastructure Functions

(Controller-base layer)

Platform APIs and Agents

Physical and Virtual Infrastructure

(Overlays and Network Function Virtualization)

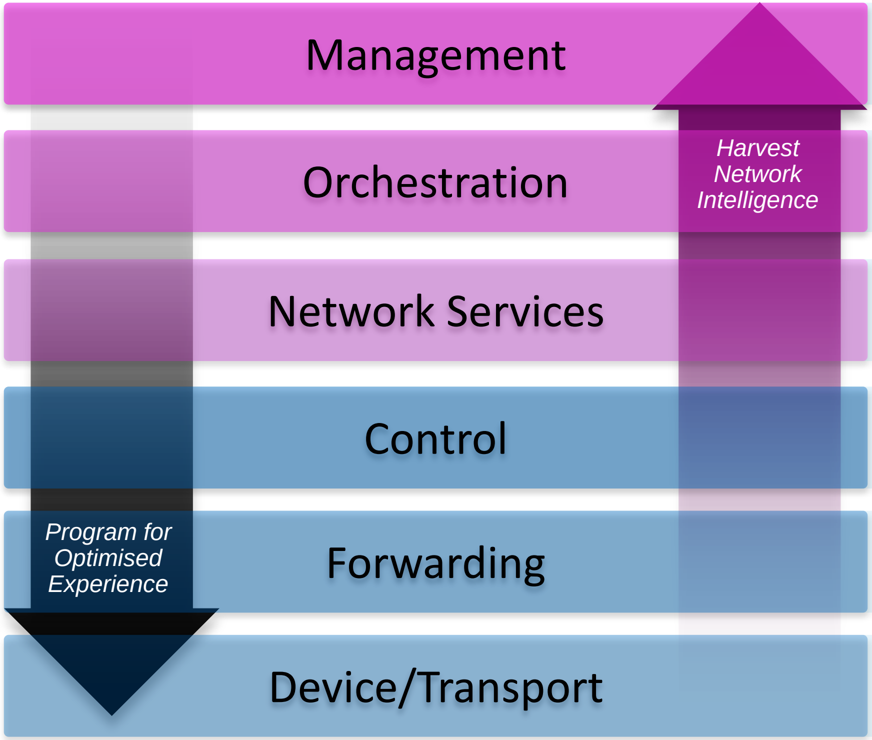
The Need for Abstractions

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

Full-Duplex, Multi-Layer/Multi-Plane APIs



Management	Workflow Management Network Configuration & Device Models, ..
Orchestration	L2-Segments, L3-Segments, Service-Chains Multi-Domain (WAN, LAN, DC)
Network Services	Topology, Positioning, Analytics Multi-Layer Path Control, Demand Eng.
Control	Routing, Policy, Discovery, VPN, Subscriber, AAA/Logging, Switching, Addressing , ..
Forwarding	L2/L3 Forwarding Control, Interfaces, Tunnels, enhanced QoS, ..
Device/Transport	Device configuration, Life-Cycle Management, Monitoring, HA, ..

Harvest Network Intelligence

Program for Optimised Experience

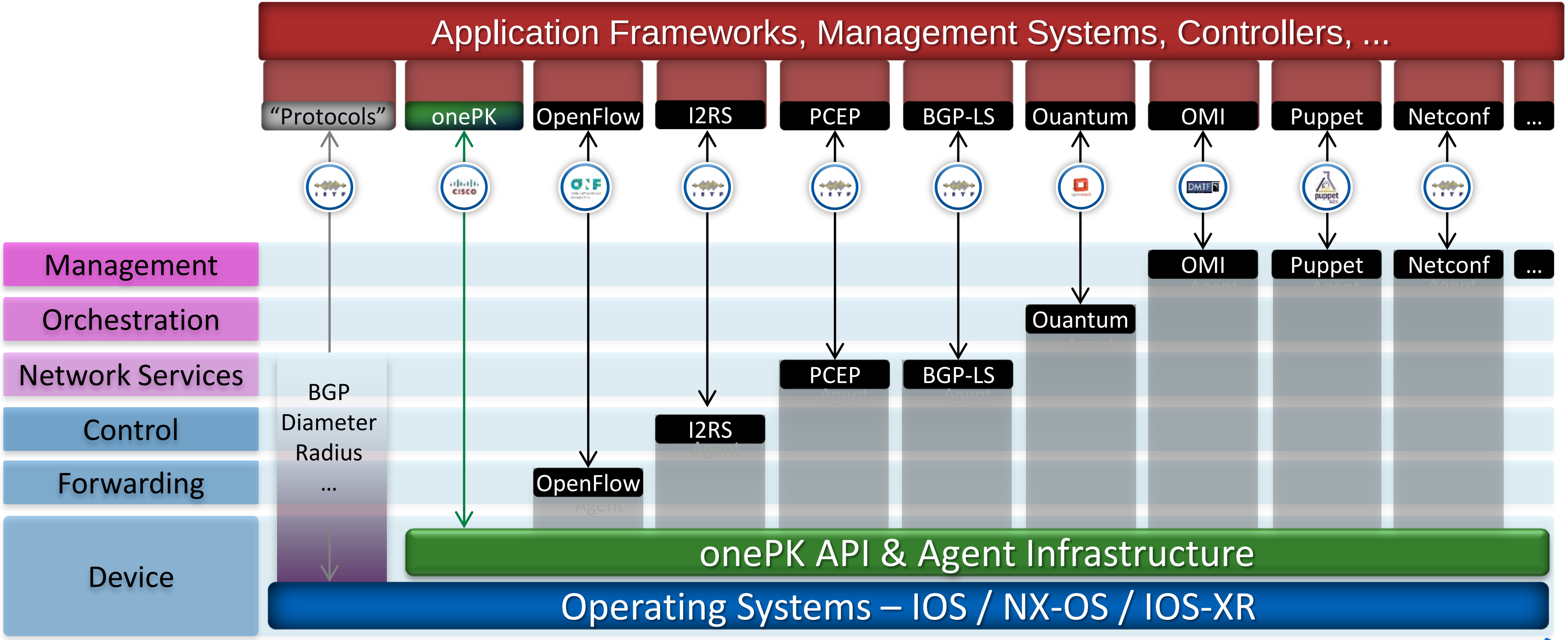
Full-Duplex, Multi-Layer/Multi-Plane APIs

Industry Examples

Management	Workflow Management Network Configuration & Device Models, ..	DMTF	Network Models - Interfaces (OMI)
Orchestration	L2-Segments, L3-Segments, Service-Chains Multi-Domain (WAN, LAN, DC)	openstack	OpenStack, Quantum API
Network Services	Topology, Positioning, Analytics Multi-Layer Path Control, Demand Eng.	IETF	Positioning (ALTO) Path Control (PCE)
Control	Routing, Policy, Discovery, VPN, Subscriber, AAA/Logging, Switching, Addressing , ..	IETF	Interface to the Routing System (I2RS)
Forwarding	L2/L3 Forwarding Control, Interfaces, Tunnels, enhanced QoS, ..	ONF OPEN NETWORKING FOUNDATION	OpenFlow Protocol
Device/Transport	Device configuration, Life-Cycle Management, Monitoring, HA, ..	ETSI	Network Function Virtualisation (NfV)

Programmatic Network Access

Agents as Flexible Integration Vehicles



Application Frameworks, Management Systems, Controllers, ...

"Protocols" | onePK | OpenFlow | I2RS | PCEP | BGP-LS | Quantum | OMI | Puppet | Netconf | ...

Management

Orchestration

Network Services

Control

Forwarding

Device

BGP
Diameter
Radius
...

OMI | Puppet | Netconf | ...

Quantum

PCEP | BGP-LS

I2RS

OpenFlow
Agent

onePK API & Agent Infrastructure

Operating Systems – IOS / NX-OS / IOS-XR

onePK for Rapid Application Development

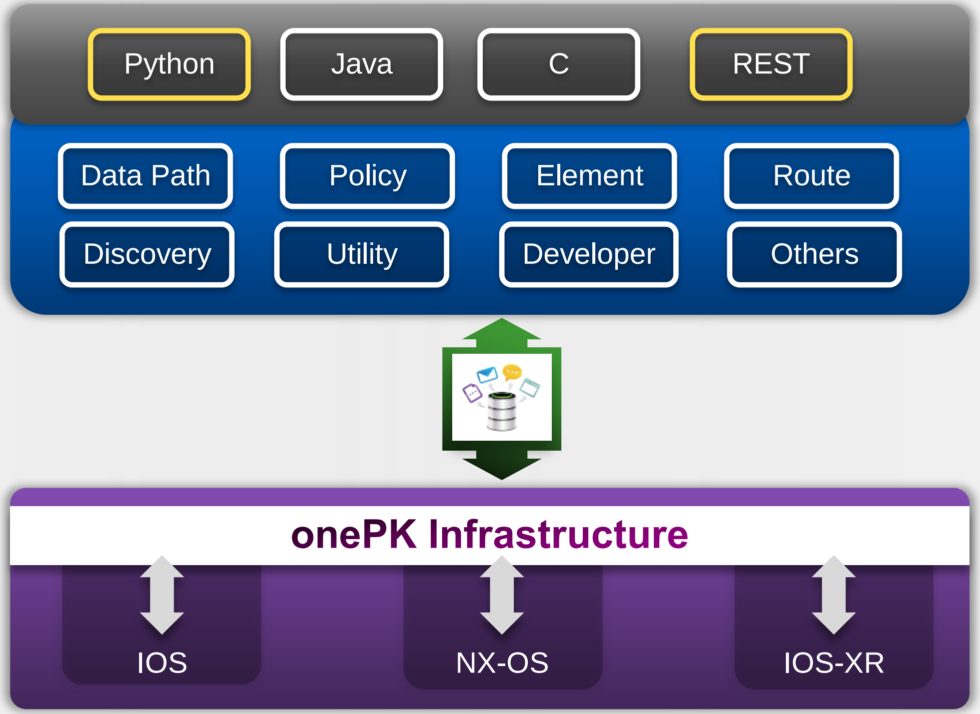
DEVELOPER ENVIRONMENT
- Language of choice
- Programmatic interfaces
- Rich data delivery via APIs

COMPREHENSIVE SERVICE SETS
- Better apps
- New services
- Monetisation opportunity

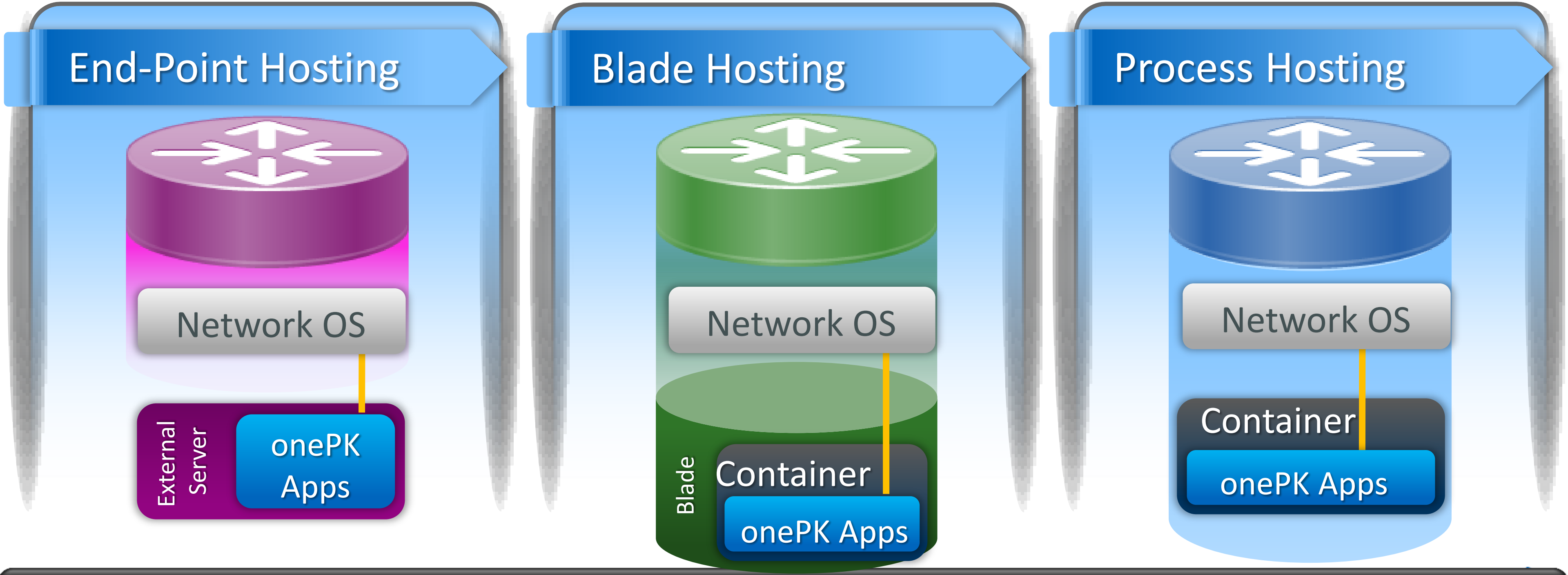
DEPLOY
- On a server blade
- On an external server
- Directly on the device

onePK

CONSISTENT PLATFORM SUPPORT
- IOS
- NX-OS
- IOS-XR



Where do I Run onePK Application?



End-Point Hosting

Network OS

External Server

onePK Apps

Blade Hosting

Network OS

Blade

Container

onePK Apps

Process Hosting

Network OS

Container

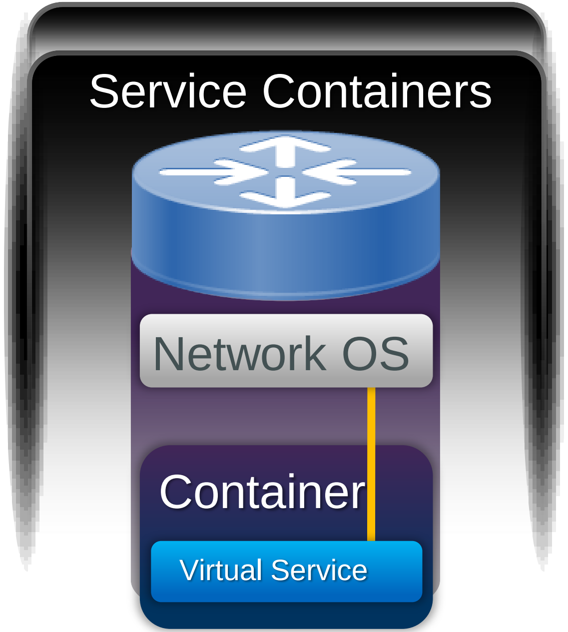
onePK Apps

Write Once, Run Anywhere

What is a Cisco Service Container?

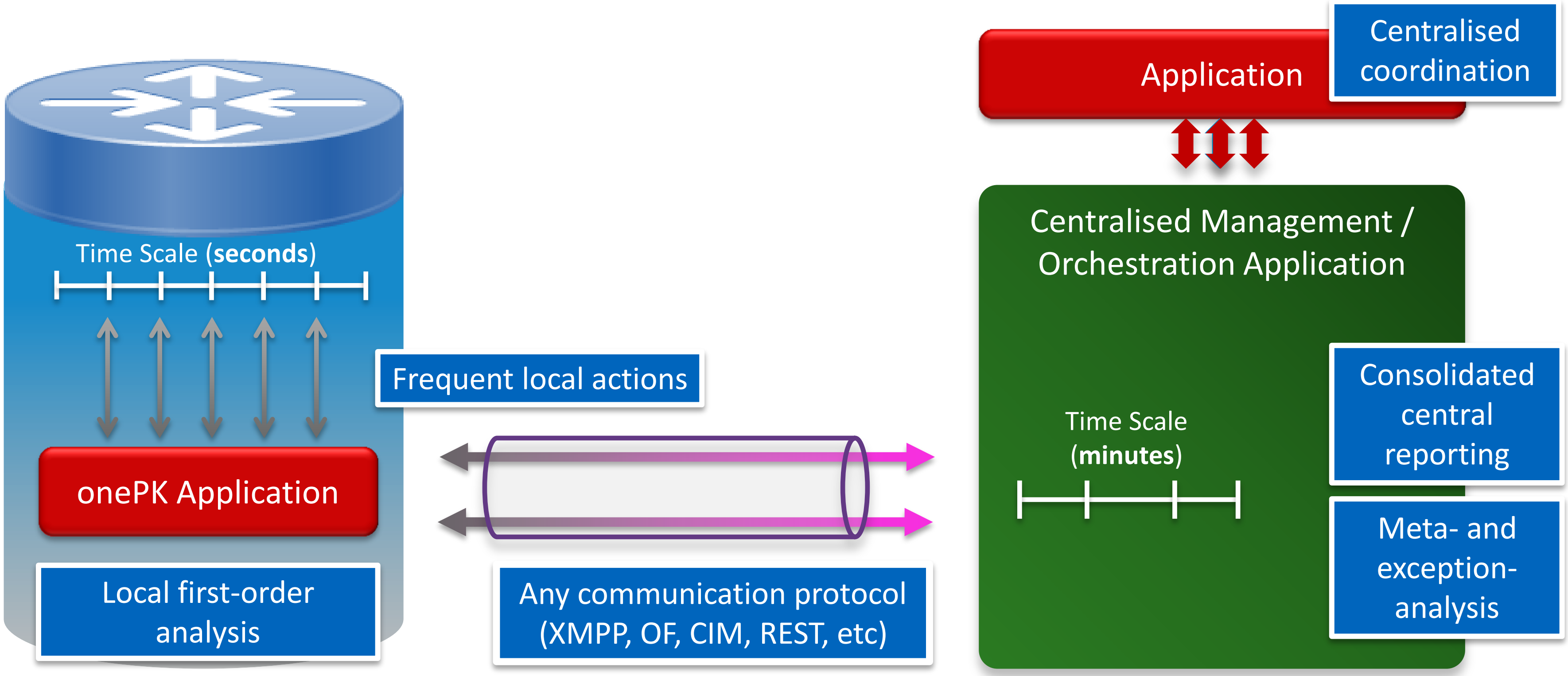
Service Containers use virtualisation technology to provide a hosting environment on Cisco routers/switches for applications which may be developed and released independent of platform release cycles.

- Virtualised environment on a Cisco device.
- Use Case Cisco Virtual Services:
  - Work/Appliance Consolidation
  - Example: ISR4451X-WAAS
- Use Case Cisco Agents:
  - Integral Router Features with decoupled release cycles
  - Example: RESTFul API
- Use Case Third Party Services (onePK applications):
  - Process Hosted onePK Applications



Network Be Nimble…

"The Agent Model"



Time Scale (**seconds**)

Frequent local actions

onePK Application

Local first-order analysis

Any communication protocol (XMPP, OF, CIM, REST, etc)

Application

Centralised coordination

Centralised Management / Orchestration Application

Time Scale (**minutes**)

Consolidated central reporting

Meta- and exception-analysis

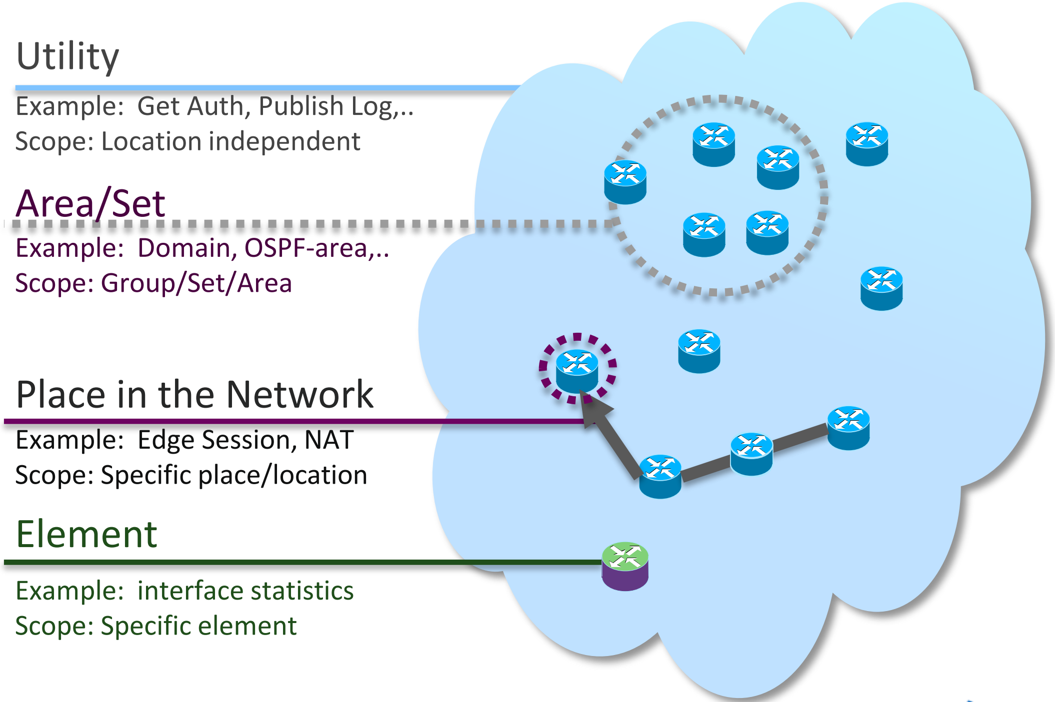
onePK APIs - Grouped in Service Sets

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

Not all Networking APIs are Created the Same

Classes of Networking APIs following their Scope

- Classify Networking APIs based on their *scope*
  - API Scopes:
    Location independent; Area; Particular place; Specific device
  - Alternate approaches like device/network/service APIs difficult to associate with use cases
  - Location where an API is hosted can differ from the scope of the API
- Different network planes could implement different flavors of APIs, based on associated abstractions



Utility

Example: Get Auth, Publish Log,..
Scope: Location independent

Area/Set

Example: Domain, OSPF-area,..
Scope: Group/Set/Area

Place in the Network

Example: Edge Session, NAT
Scope: Specific place/location

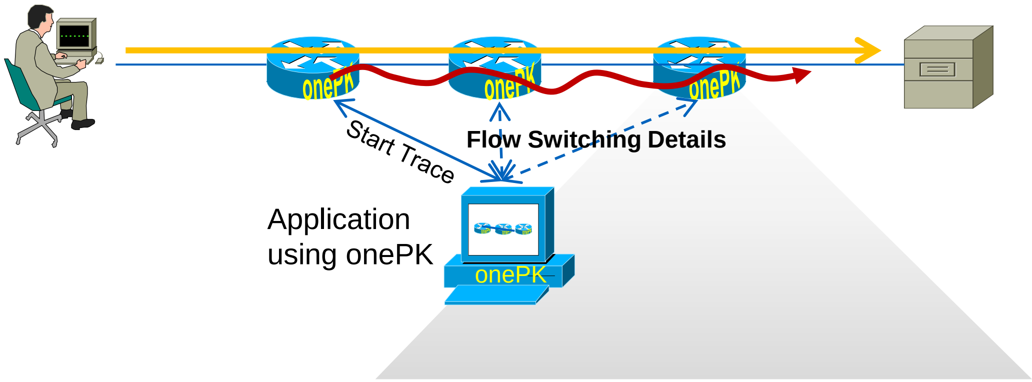
Element

Example: interface statistics
Scope: Specific element

APIs at Work – Element APIs

Example: Statistics, Diagnostics & Troubleshooting

- Objective:
  - Provide operators/ administrators/ support engineers with details about how packets flow through the network.
  - Reveal network issues
- Approach
  - NMS application leverages onePK APIs to show path of flow, timestamp, ingress/egress interfaces, interface packet counts

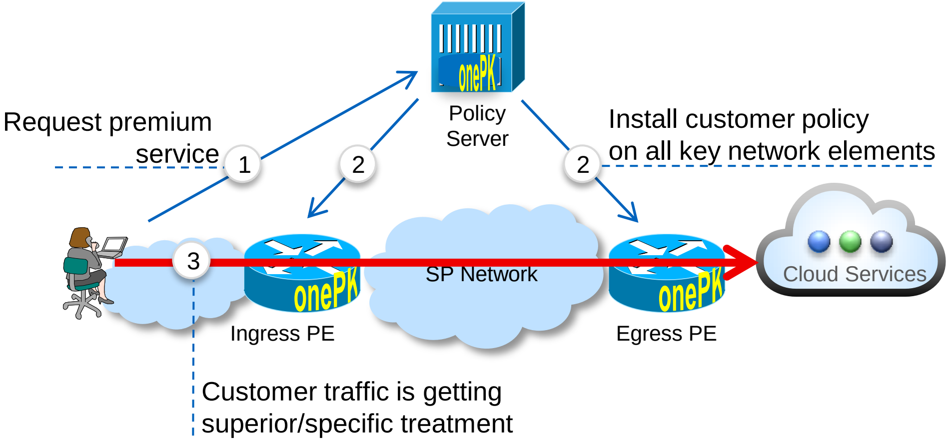


```
Ingress time: May 15, 2011 00:46:55.145
Ingress intf: Gi0/1
Ingress pkts: 30
Egress time: Jun 6, 2011 00:46:55.251
Egress intf: Gi0/0
Egress pkts: 5
```

APIs at Work – Place in the Network APIs

Example: Dynamic Bandwidth/QoS Allocation

- Business Problem
  - Enable superior experience for subscribers which access a particular cloud service
- Solution
  - Install customer policy (QoS, access control,..) using onePK on key networking elements, e.g. Provider Edge (PE) routers
  - Similarities to broadband “Bandwidth on Demand” use cases
    - Broadband: Policy controlled on Subscriber-Gateway (BRAS/BNG, GGSN/PGW, ..) only
    - Common API like onePK enables control points on all key networking devices



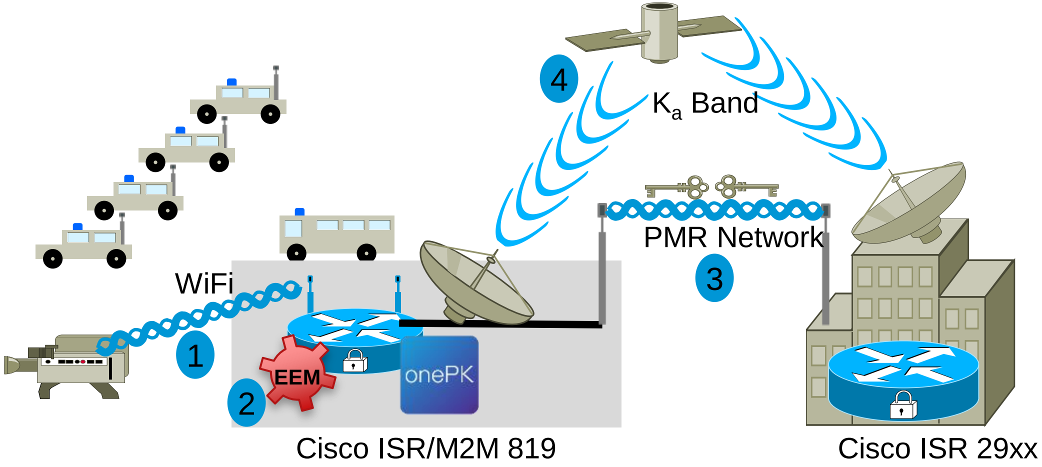
APIs at Work – Place in the Network APIs

Example: Emergency Response Network

Problem: How to deliver secure, trusted, robust, cost-effective broadband connectivity to mobile emergency response units?

Solution: Use Network Programming based on Cisco onePK and Cisco IOS Embedded Event Manager to integrate low-cost, high-bandwidth options with accredited legacy radio connectivity

Design: Pramacom (the key customers: Ministry of Interior of Czech Republic and Ministry of Interior of Slovak Republic)

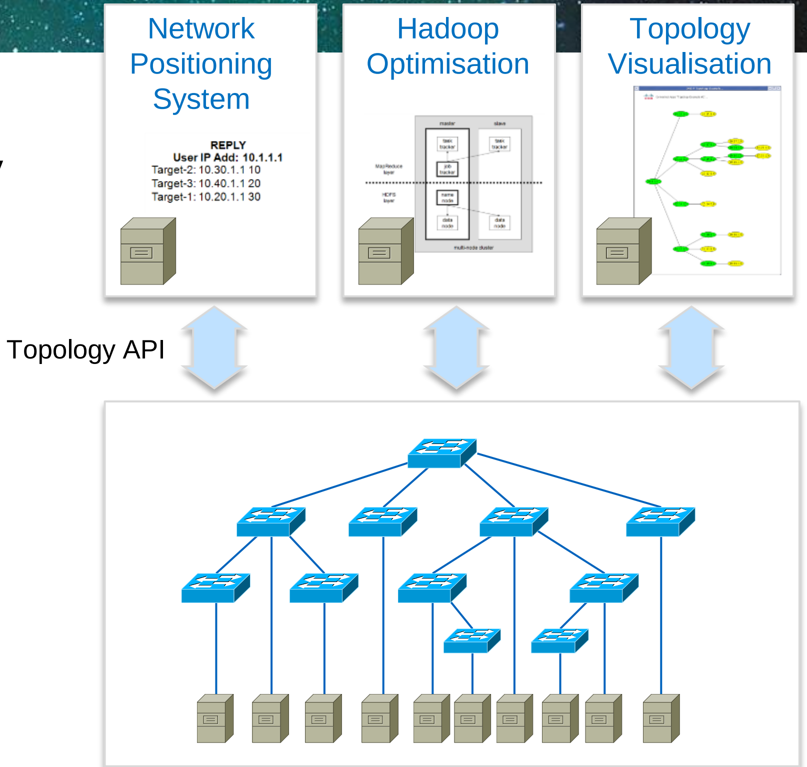


1. Connect high-bandwidth forward clients via WiFi
2. Use Cisco IOS EEM for onboard system integration and adaptation
3. Use Cisco onePK to redirect IKE key exchange out-of-band via legacy radio
4. Secure IPSec tunnel via cost-effective high bandwidth K_a Band
5. Reliable, secure emergency response network saving ~4M€ operating cost annually

APIs at Work – Area APIs

Examples: Topology graph

- Business Problem
  - Several problems require a view of the network topology (area, domain, or whole network)
  - Examples:
    - Locate optimal service out of a given list
    - Optimise Load Placement
    - Visualise the active Network Topology
- Solution
  - Topology API to expose network topology to applications, such as
    - NPS (for service selection)
    - Hadoop (for optimal job placement)
    - NMS (for topology visualisation)



Network Positioning System

REPLY

User IP Add: 10.1.1.1

Target-2: 10.30.1.1 10

Target-3: 10.40.1.1 20

Target-1: 10.20.1.1 30

Hadoop Optimisation

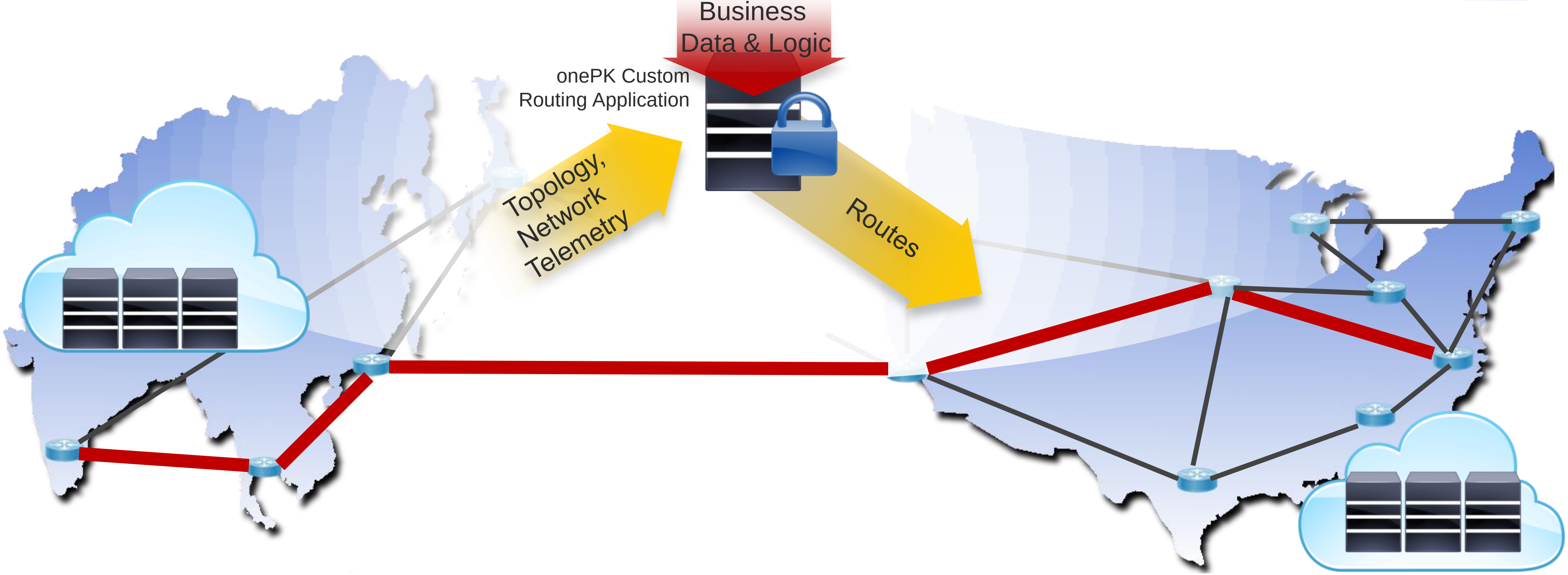
Topology Visualisation

Topology API

Example: Custom Routing

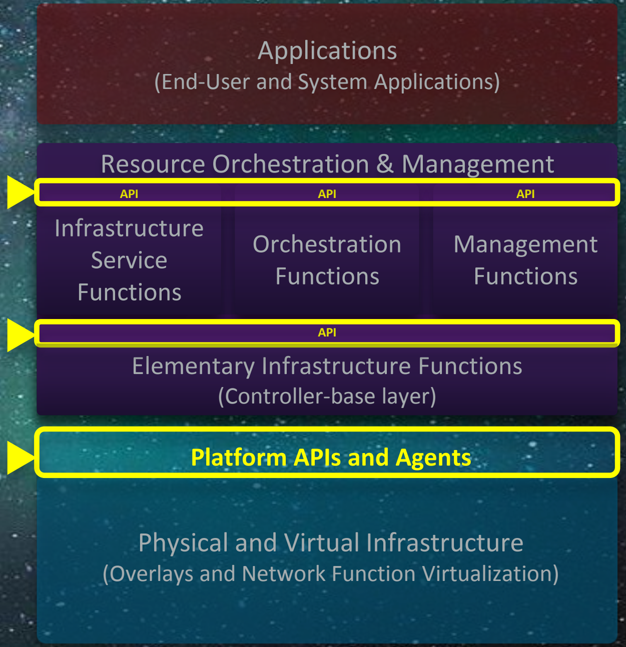
Data Centre Traffic Forwarding Based on a Custom Algorithm

onePK



Unique Data Forwarding Algorithm Highly Optimised
for the Network Operator’s Application

Open Network Environment Qualities
Programmatic APIs
ONF's OpenFlow Protocol



Applications
(End-User and System Applications)

Resource Orchestration & Management

API
API
API

Infrastructure Service Functions
Orchestration Functions
Management Functions

API

Elementary Infrastructure Functions
(Controller-base layer)

Platform APIs and Agents

Physical and Virtual Infrastructure
(Overlays and Network Function Virtualization)

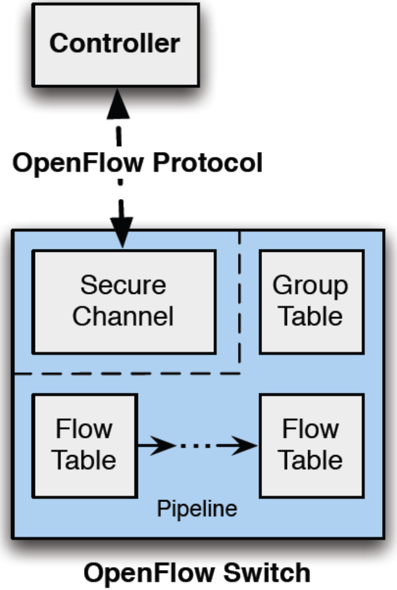
OpenFlow

- 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 <= 1.4.0) focused on Forwarding Plane Abstraction.
  - *OpenFlow Protocol*: A communications protocol that provides access to the forwarding plane of an OpenFlow Device

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



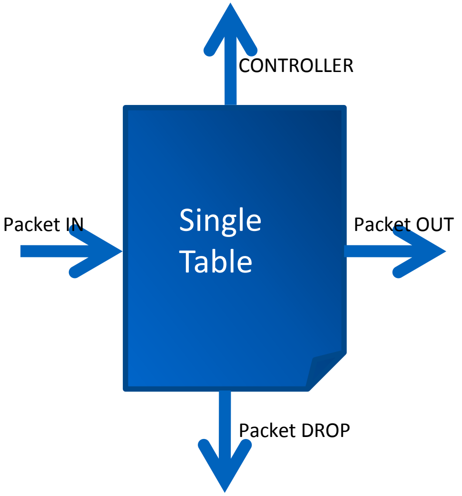
Source: OpenFlow 1.3.1 specification, figure 1

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

OF Processing Pipeline

OF 1.0 model

(single lookup)



CONTROLLER

Packet IN

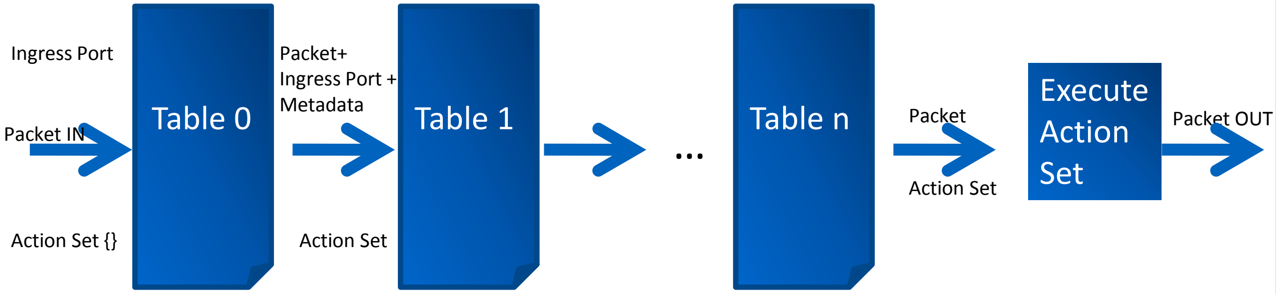
Single Table

Packet OUT

Packet DROP

OF 1.1 and beyond model

(multiple lookups)



Ingress Port

Packet IN

Action Set {}

Table 0

Packet+
Ingress Port +
Metadata

Action Set

Table 1

…

Table n

Packet

Action Set

Execute Action Set

Packet OUT

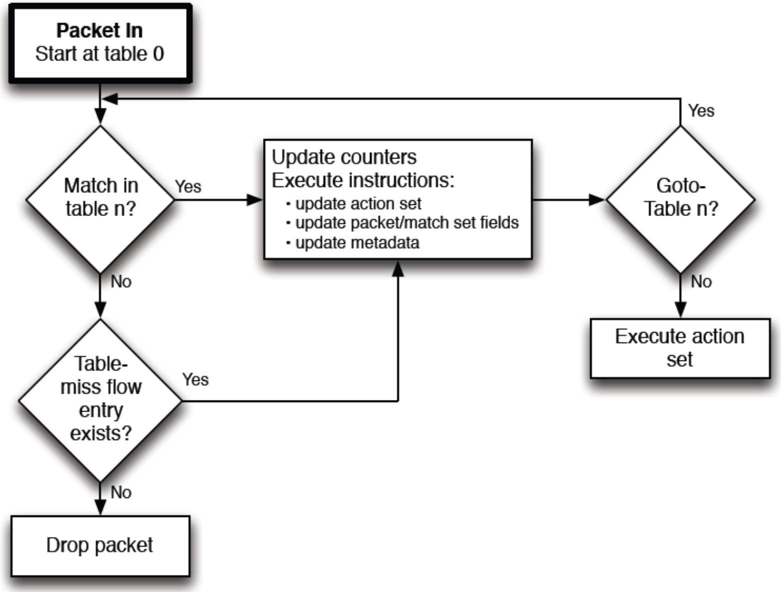
① Find highest-priority matching flow entry

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

③ Send match data and action set to next table

Source: OpenFlow 1.3.1 specification, figure 2

Packet Flow Through an OpenFlow Switch



Source: OpenFlow 1.4.0 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

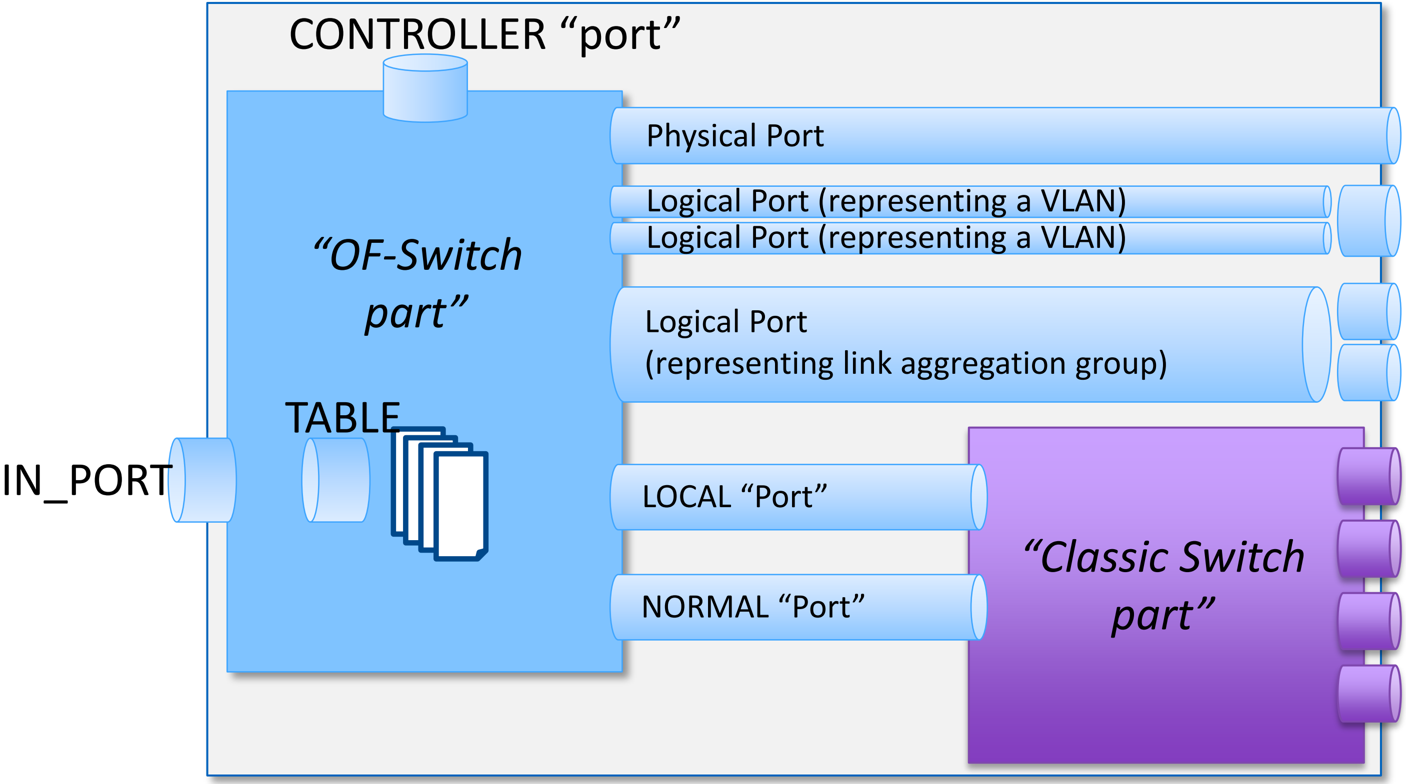
Physical Ports, Logical Ports, Reserved Ports

- 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



CONTROLLER “port”

Physical Port

Logical Port (representing a VLAN)

Logical Port (representing a VLAN)

Logical Port
(representing link aggregation group)

“OF-Switch part”

TABLE

IN_PORT

LOCAL “Port”

NORMAL “Port”

“Classic Switch part”

OpenFlow Ports

CONTROLLER port and NORMAL “port”

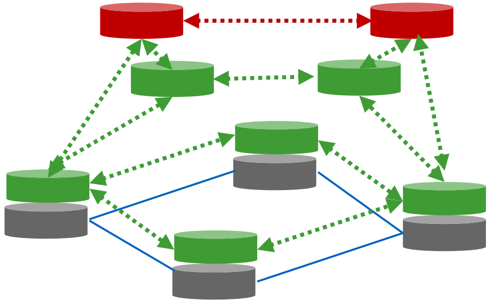
- CONTROLLER
  - Forward packets to Controller
  - For “reactive” mode of operation
  - Considerations
    - Latency for decision making
    - Bandwidth between OF-switch and OF-controller
    - Speed at which rules can be installed/removed

- NORMAL
  - More of a concept than a real “port”: Hand packets to “classic” part of the switch
  - Forwarding operation in the classic part is TBD
    - Xconnect?
    - L2-Bridge (use Dest-MAC to forward packet to o/if)?
    - L3-Route (requires L3-next hop info as meta-data from OF, or rely on classic routing protocol)?

Integration with Existing Networking Devices

The “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



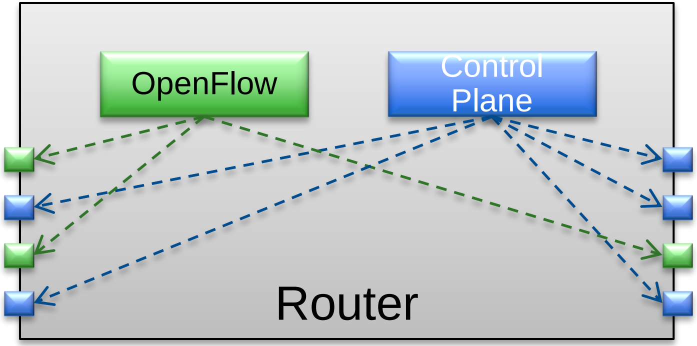
A Couple Of Hybrid Switch Use Cases

- 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”

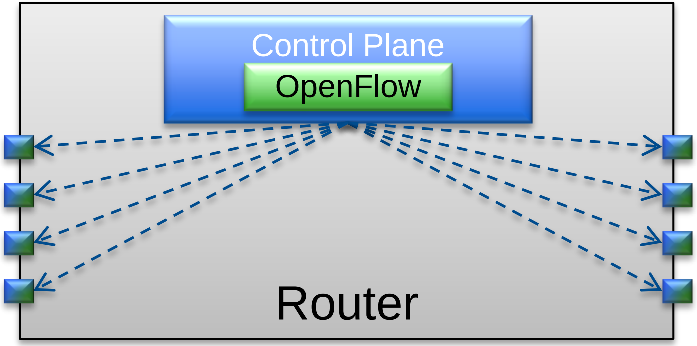
(aka “vertical partitioning”*)



- 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”

(aka “horizontal partitioning”)



- 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)

* See: ONF Architecture Draft 0.0.1

OpenFlow Versions

Status

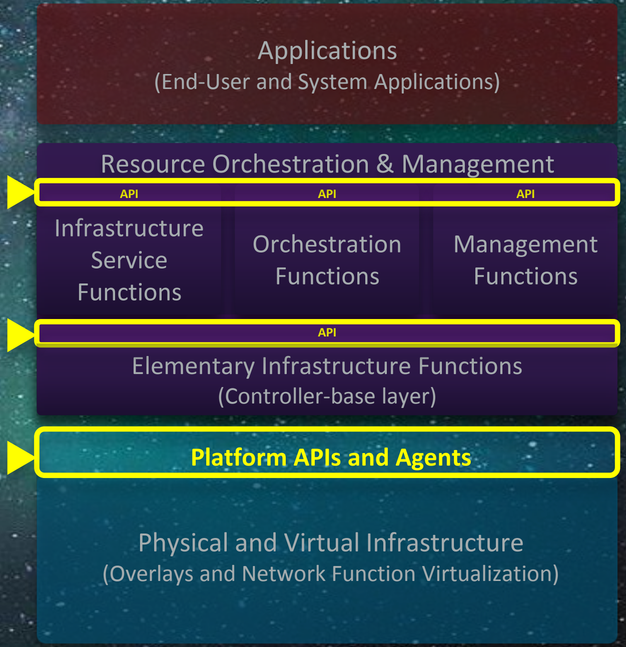
Dec 31, 2009	**OF 1.0**	• Single Table • L2, IPv4 focused matching
Feb 28, 2011	**OF 1.1**	• Multiple Tables • MPLS, VLAN matching • Groups: {Any-,Multi-}cast • ECMP
Dec 5, 2011	**OF 1.2**	• IPv6 • Flexible TLV matching • Multiple Controllers
Apr 19, 2012	**OF 1.3.0**	• 802.1ah PBB • Multiple parallel channels between Switch and Controller
Jun 7, 2012	**OF 1.0.1**	• Bug fixes
Sep 6, 2012	**OF 1.3.1**	• Bug fixes
Apr 25, 2013	**OF 1.3.2**	• Bug fixes
Oct 14, 2013	**OF 1.4.0**	• Flexible ports • Synch across flow tables • Enhanced Clustering • Flow-Monitoring

- Evolution of the specification: Mature and Evolve
  - “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 1.3.X: long term support
  - OpenFlow 1.4: extensibility, incremental improvements
  - OpenFlow 1.0.X : no work planned

Open Network Environment Qualities

Programmatic APIs

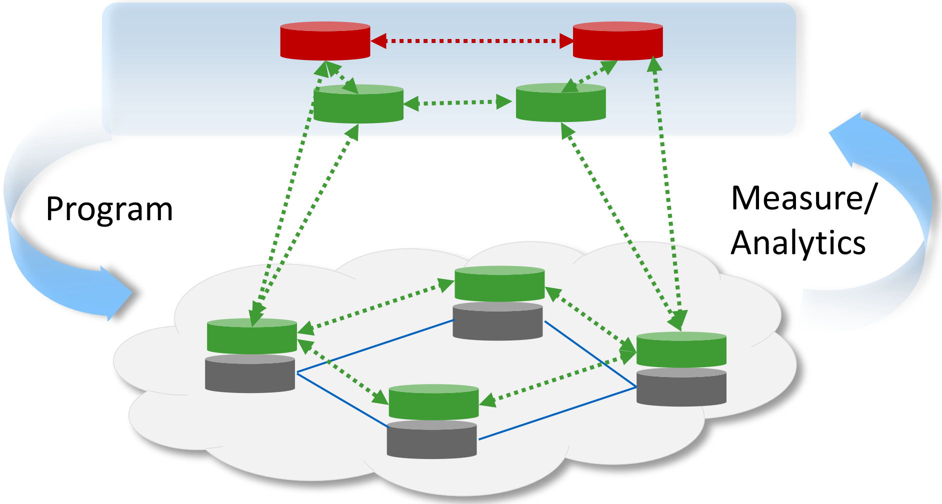
IETF's Interface to the Routing System - I2RS



Towards the "Interfaces to the Routing System"

Approach

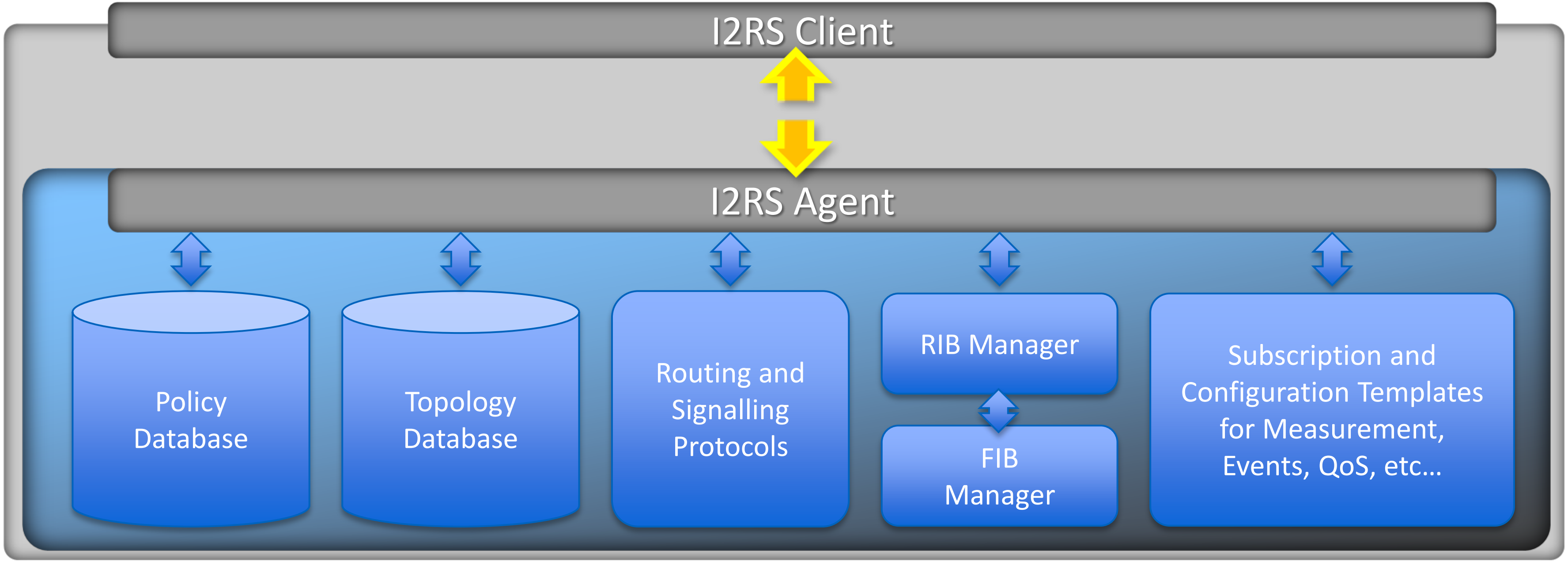
- Dynamically augment the Routing System / Control Plane based on
  - Policy
  - Flow & Application Awareness
  - Time & External Changes
- Leverage
  - Topology (active & potential)
  - Events
  - Traffic Measurements
  - …



Feedback Loop: Control & Information

I2RS: Initial Requirements

- Data Models for Routing & Signalling State
  - RIB Layer: unicast RIBs, mcast RIBs, LFIB, etc.
  - Protocols: ISIS, OSPF, BGP, RSVP-TE, LDP, PIM, mLDP, etc.
  - Related: Policy-Based Routing, QoS, OAM, etc.
- Filtered Events for Triggers, Verification & Learning Changed Router State
- Data Models for State
  - Topology model, Interface, Measurements, etc.
- Application-Friendly Interface & Protocol(s)



See also:
draft-ward-irs-framework, draft-atlas-irs-problem-statement,
draft-amante-irs-topology-use-cases, draft-keyupate-bgp-services, …

I2RS - Key Aspects & Anticipated Features

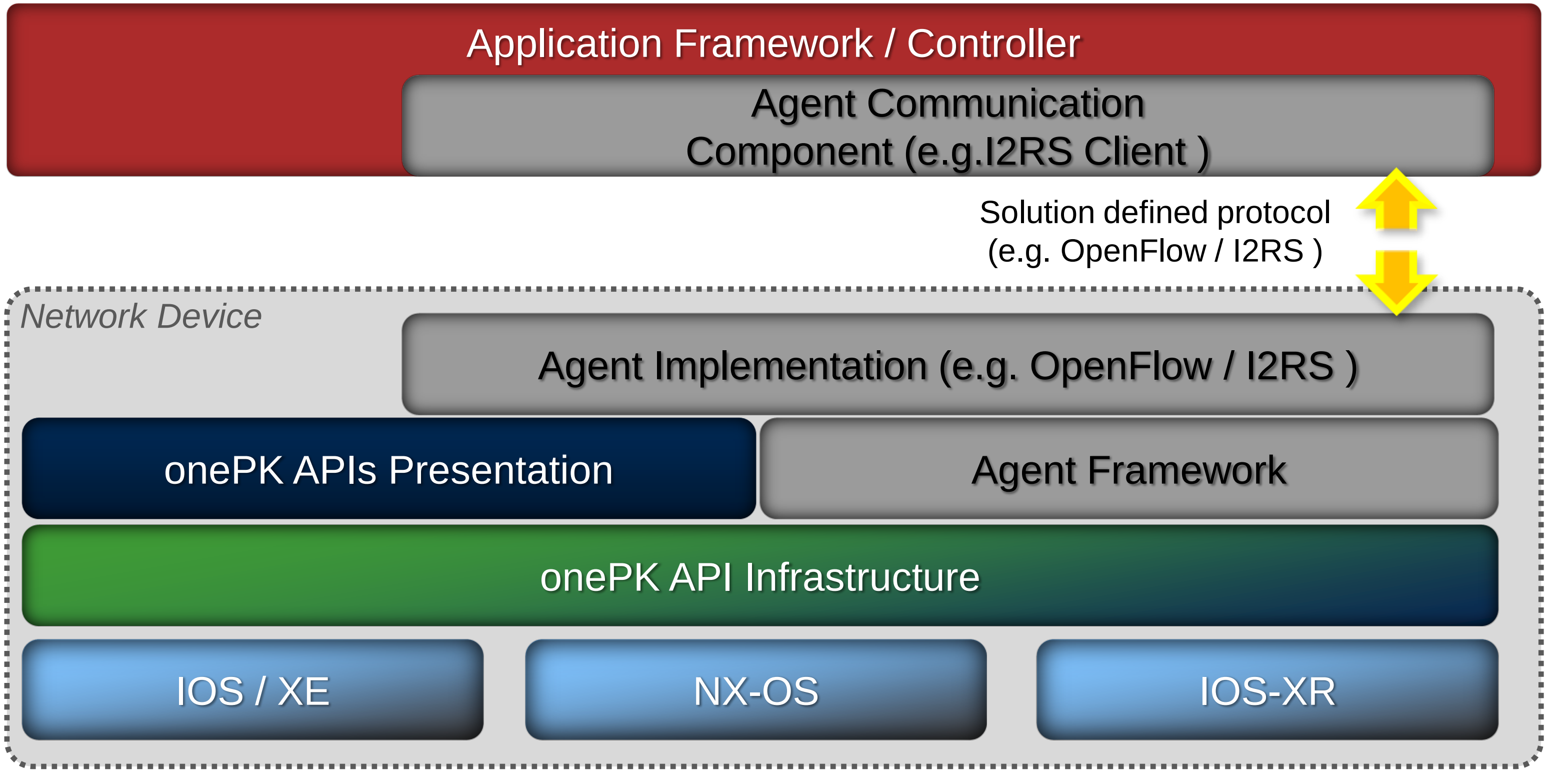
- Multiple Simultaneous Asynchronous Operations
- Duplex Communication
- High-Throughput
- Highly Responsive
- Multi-Channel (readers/writers)
- Capabilities Negotiation/Advertisement (self-describing)

- Installed state can have different lifetime models:
  - Ephemeral (until reboot)
  - Persistent
  - Time-based Persistent: Expires after specified time
  - Time-based Ephemeral: Expires after specified time
- Operations to install state have different install-time models:
  - Immediately
  - Time-Based
  - Triggered by an Event

See also: Draft I2RS Charter

Enabling OpenFlow, I2RS,… on Top of onePK

onePK Agent Framework



Application Framework / Controller

Agent Communication Component (e.g.I2RS Client)

Solution defined protocol
(e.g. OpenFlow / I2RS)

Network Device

Agent Implementation (e.g. OpenFlow / I2RS)

onePK APIs Presentation

Agent Framework

onePK API Infrastructure

IOS / XE

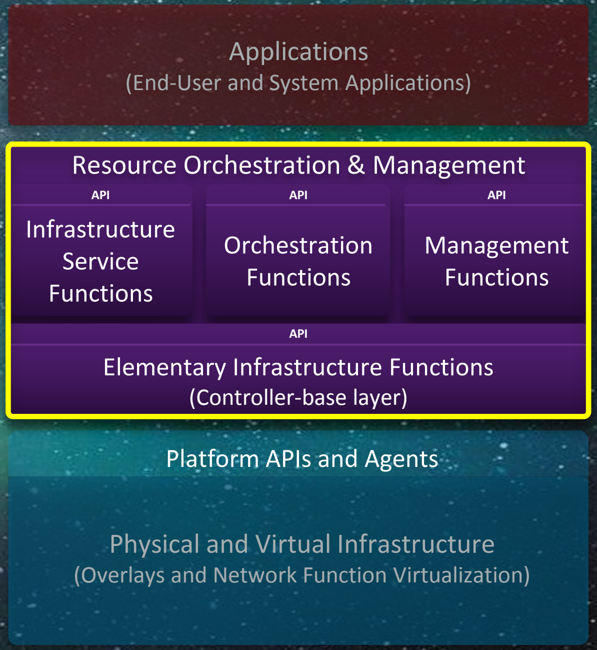
NX-OS

IOS-XR

Open Network Environment Qualities

Resource Orchestration – Controllers

Logically centralised and
fully distributed Control



Applications
(End-User and System Applications)

Resource Orchestration & Management

API

Infrastructure Service Functions

API

Orchestration Functions

API

Management Functions

API

Elementary Infrastructure Functions
(Controller-base layer)

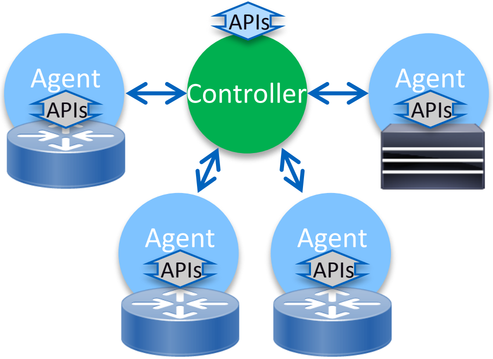
Platform APIs and Agents

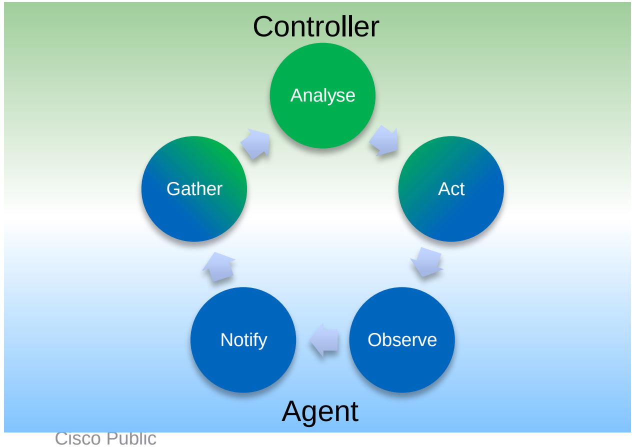
Physical and Virtual Infrastructure
(Overlays and Network Function Virtualization)

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

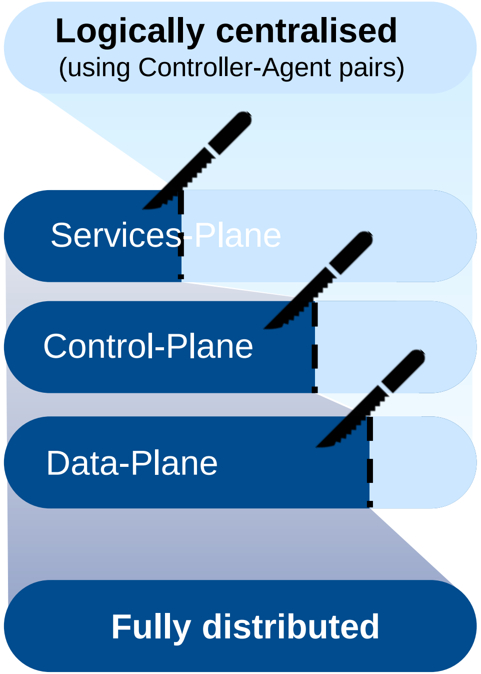




Distributed Control

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

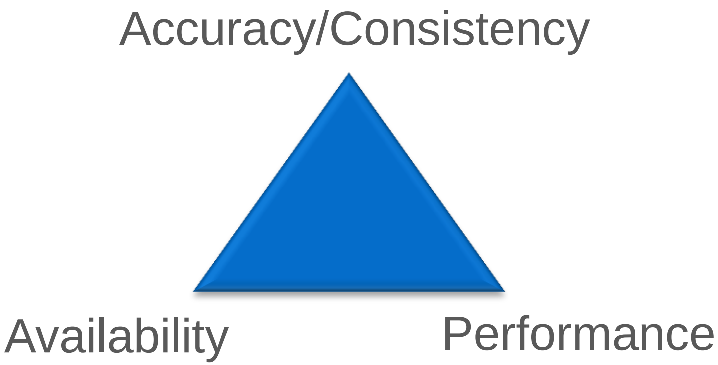
- Control loop requirements differ per function/service and deployment domain
  - “As loose as possible, as tight as needed”
  - Latency, Scalability, Robustness, Consistency, Availability
  - Different requirements per use case
    - Example:
      Topology for Visualsation (Network Management) vs.
      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

Consistency – Availability – Performance Tradeoff

Example: Network Graph Abstraction – Tradeoff differs by use case

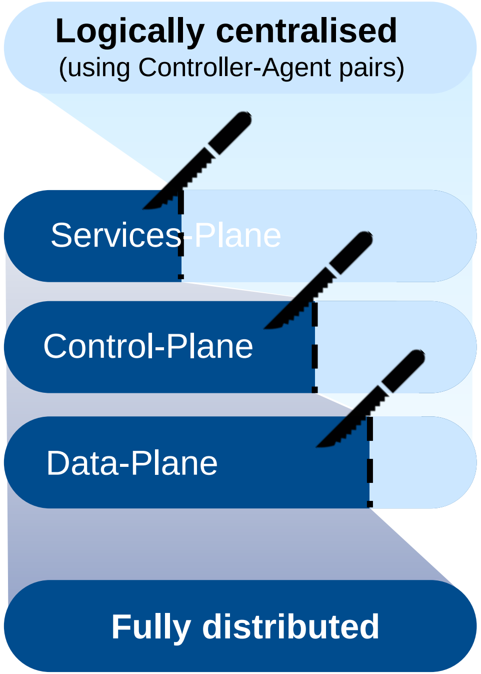


- Network visualisation
  - Loose timing and accuracy requirements
- Service/Load placement
  - Longer term heuristic algorithms used for service placement, thus limited accuracy required
- Forwarding: Generic Routing
  - Eventual consistency between forwarding and control state (TTL for temporary loop protection)
  - Sub-second convergence time: Fast reaction to all occurring events
- Forwarding: Generic Bridging
  - Strong consistency between forwarding and control state required (no loop protection in dataplane)
  - Sub-second convergence time: Fast reaction to all occurring events

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?*



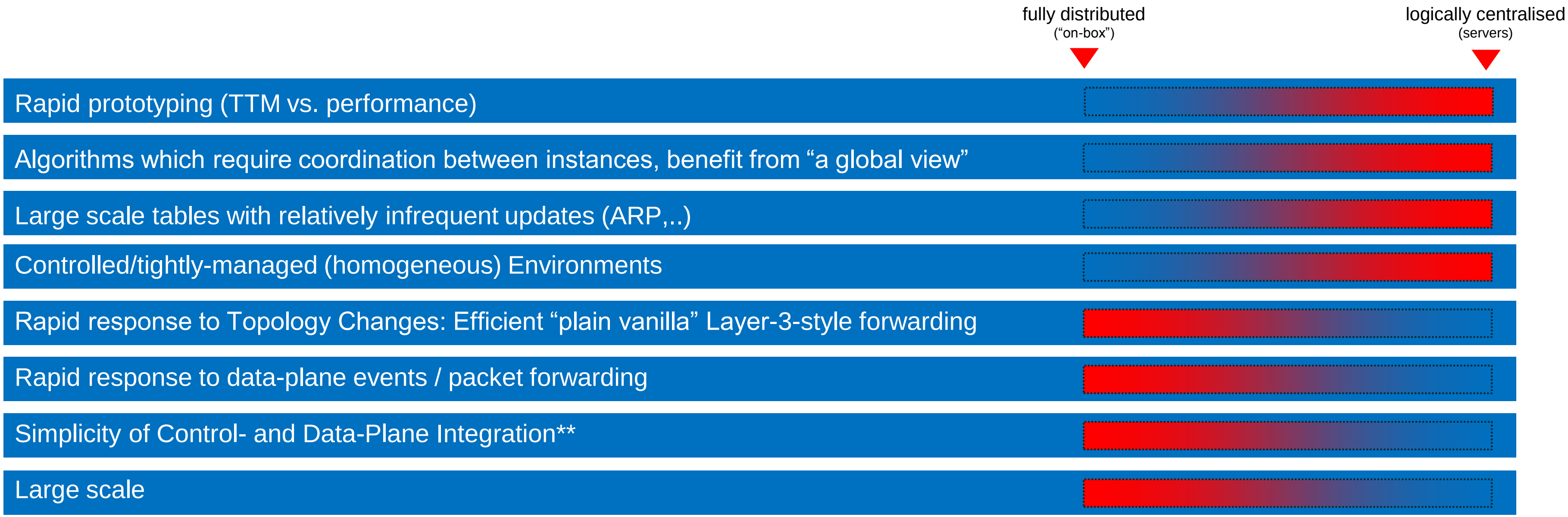
Note: Example only – Not all network planes shown

> “Subsidiarity is an organizing principle that matters ought to be handled by the smallest, lowest or least centralized competent authority.”

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

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

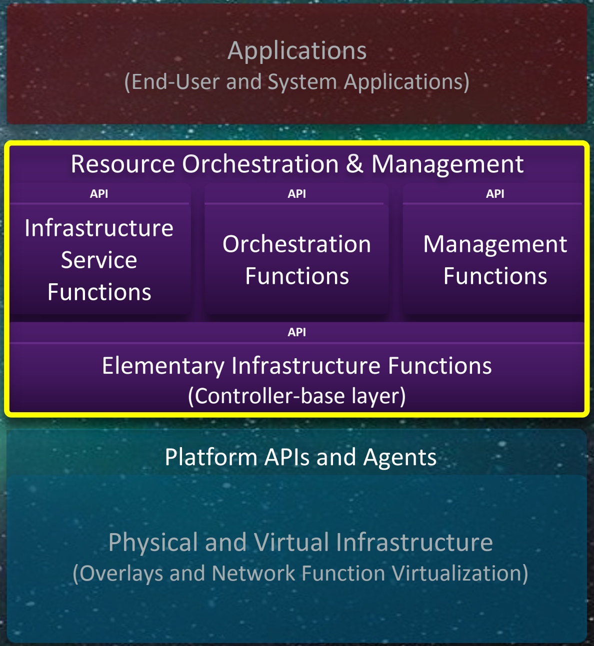
Simplicity of Control- and Data-Plane Integration**

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/

Open Network Environment Qualities
Resource Orchestration – Controllers


Applications
(End-User and System Applications)

Resource Orchestration & Management

API
Infrastructure Service Functions

API
Orchestration Functions

API
Management Functions

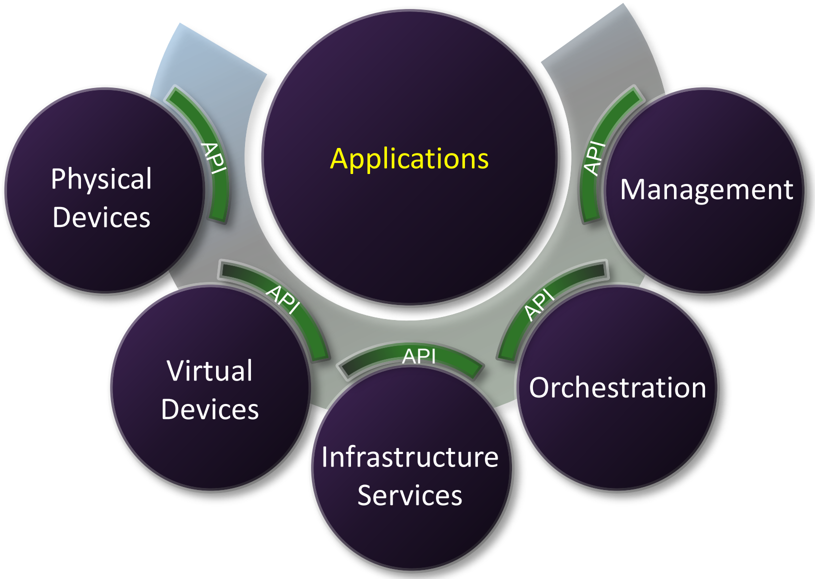
API
Elementary Infrastructure Functions
(Controller-base layer)

Platform APIs and Agents

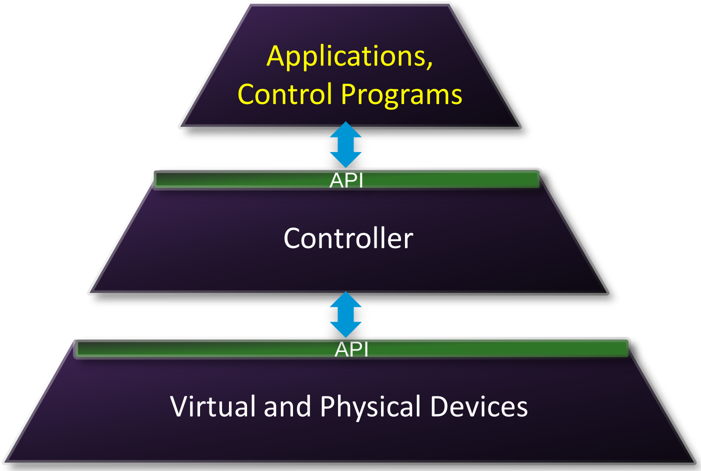
Physical and Virtual Infrastructure
(Overlays and Network Function Virtualization)

Software Architecture Perspective

Programmability supports any model: Hierarchical and Peering



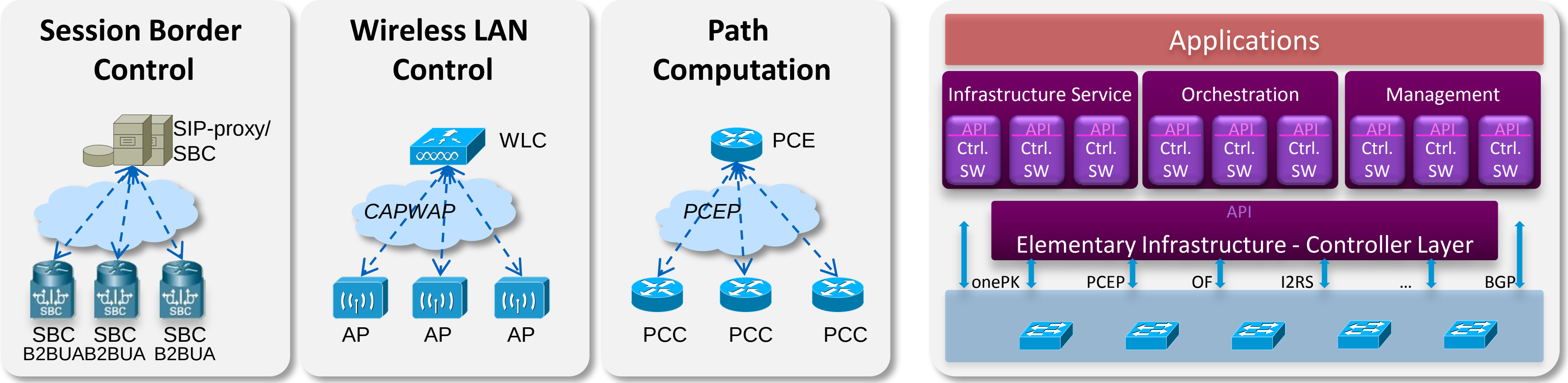
Peering Model



Hierarchical Model
(followed by original SDN)

Resource Orchestration and Control Software

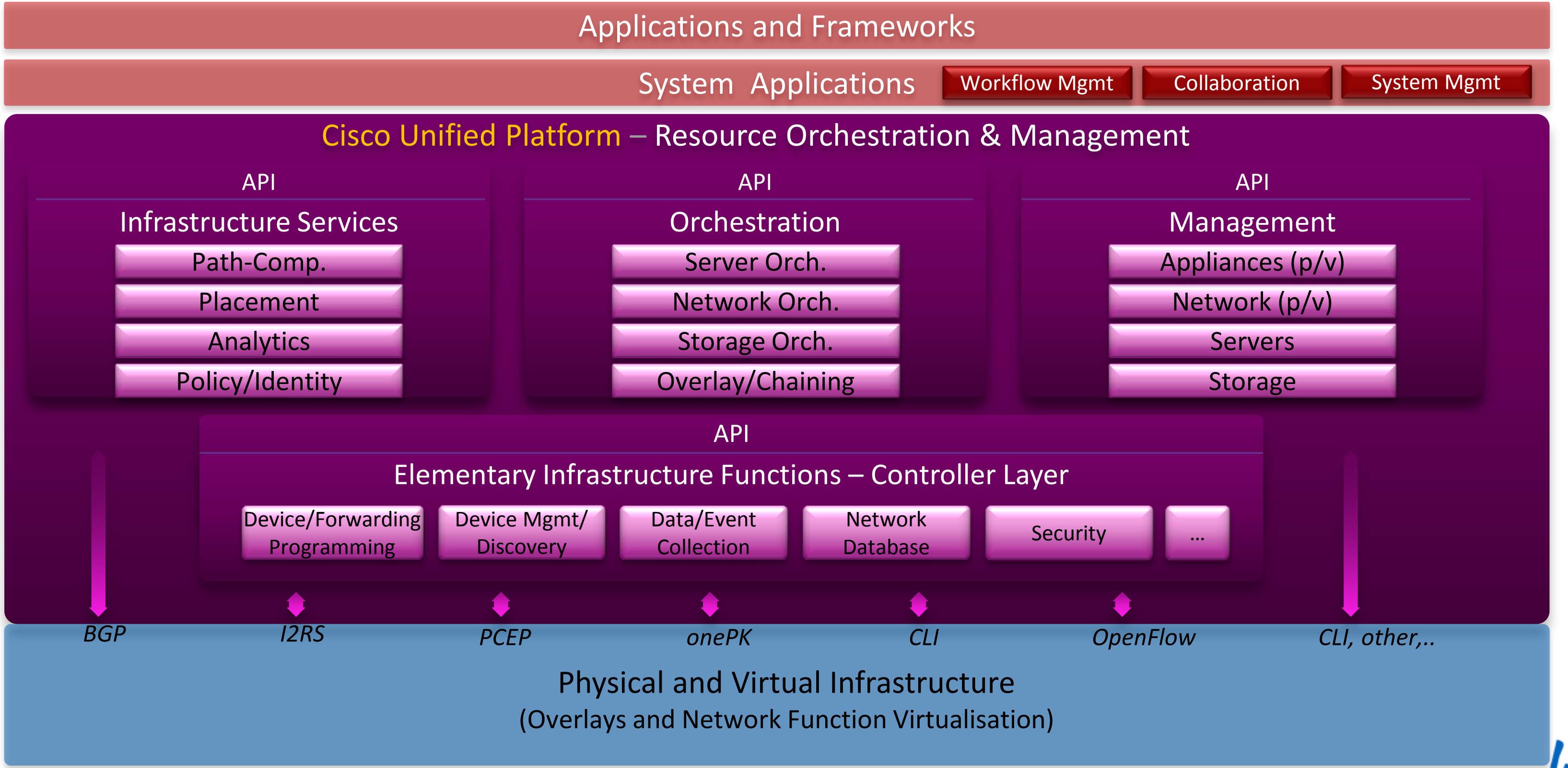
Task Specific Solutions and Generic Controller Infrastructure



- 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

Resource Orchestration and Control Software

Enabling an EcoSystem of Network Software



Applications and Frameworks

System Applications

- Workflow Mgmt
- Collaboration
- System Mgmt

Cisco Unified Platform – Resource Orchestration & Management

API

Infrastructure Services

- Path-Comp.
- Placement
- Analytics
- Policy/Identity

API

Orchestration

- Server Orch.
- Network Orch.
- Storage Orch.
- Overlay/Chaining

API

Management

- Appliances (p/v)
- Network (p/v)
- Servers
- Storage

API

Elementary Infrastructure Functions – Controller Layer

- Device/Forwarding Programming
- Device Mgmt/ Discovery
- Data/Event Collection
- Network Database
- Security
- ...

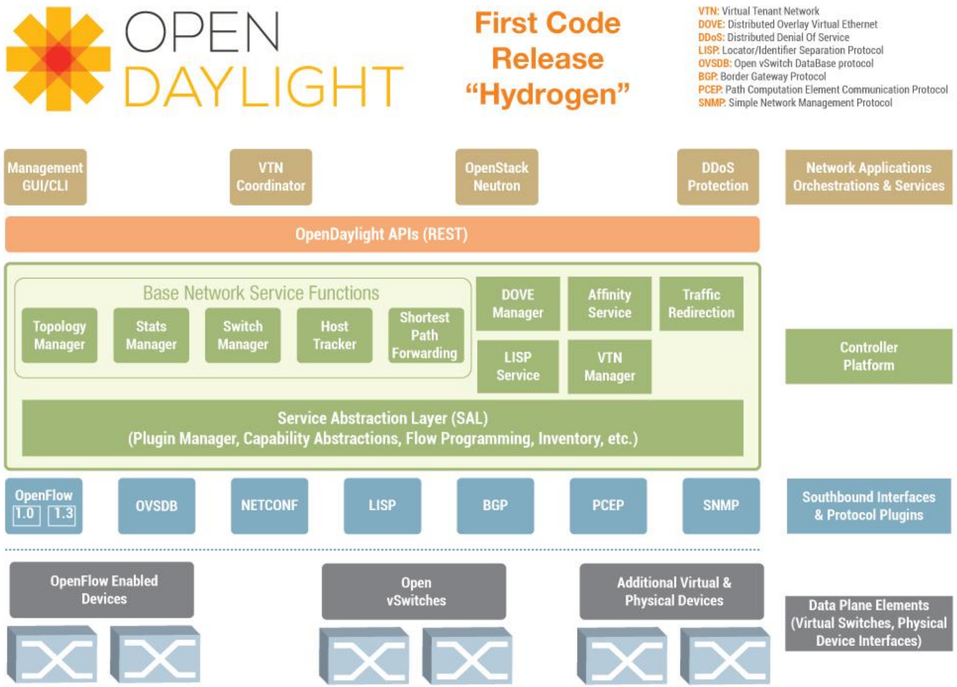
BGP *I2RS* *PCEP* *onePK* *CLI* *OpenFlow* *CLI, other,..*

Physical and Virtual Infrastructure

(Overlays and Network Function Virtualisation)

What is Project OpenDaylight?

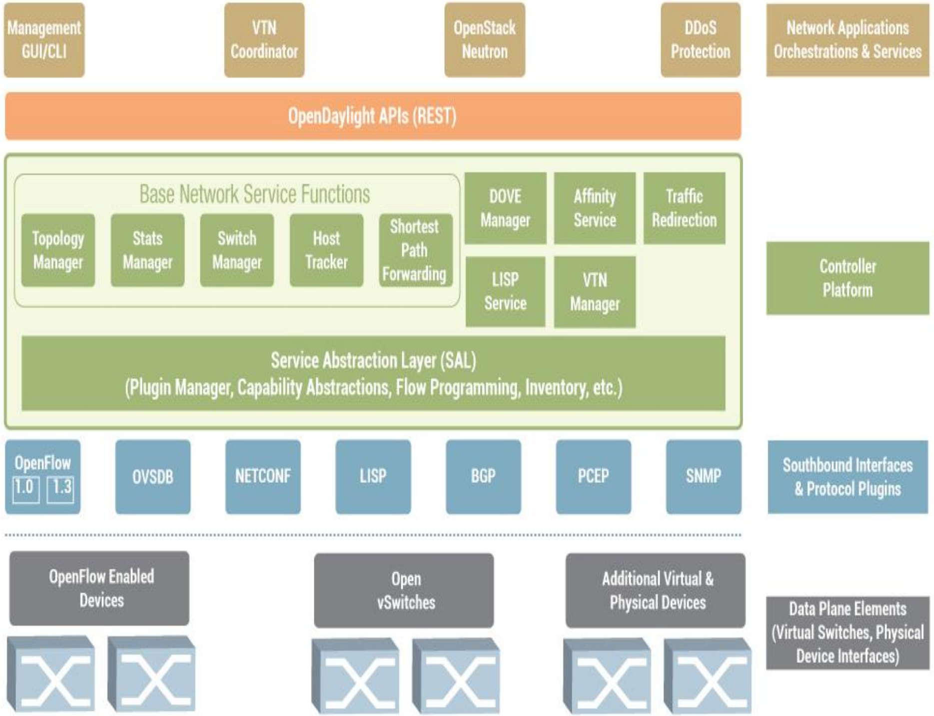
- OpenDaylight is an open source project under the Linux Foundation with the mutual goal of furthering the adoption and innovation of Software Defined Networking (SDN) through the creation of a common market-supported framework.
- www.opendaylight.org
- wiki.opendaylight.org
- 4th Feb 2014 – Hydrogen Release



Orchestration & Control: Components

Elementary Infrastructure Functions: Goals & Cisco Contribution

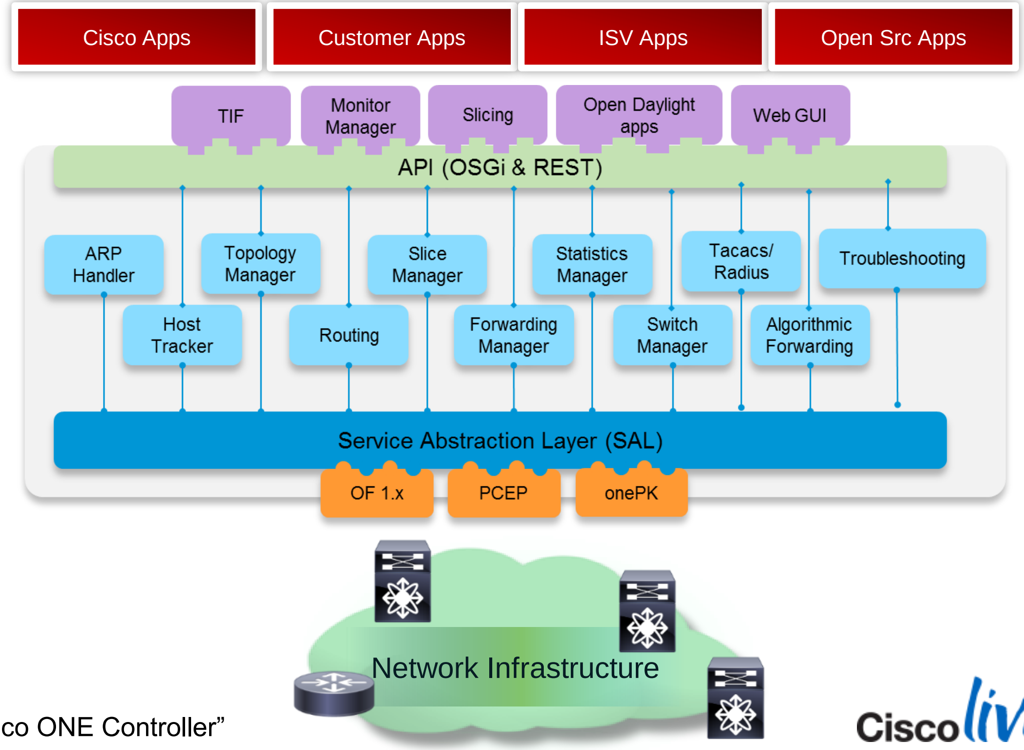
- Code: To create a robust, extensible, open source code base that covers the major common components required to build an SDN solution.
- Acceptance: To get broad industry acceptance amongst vendors and users.
- Community: To have a thriving and growing technical community contributing to the code base, using the code in commercial products, and adding value above, below and around.
- Current Cisco Contribution
  - Cisco contributes a Controller and Service Abstraction Layer that ensures the modularity and extensibility of the Controller.
  - An OpenFlow 1.0 plugin is provided on the South bound side, and Northbound API interfaces (OSGi and RESTful) will be provided for application development



Orchestration & Control – Components:

Elementary Infrastructure Functions and beyond: Extensible Network Controller (XNC*)

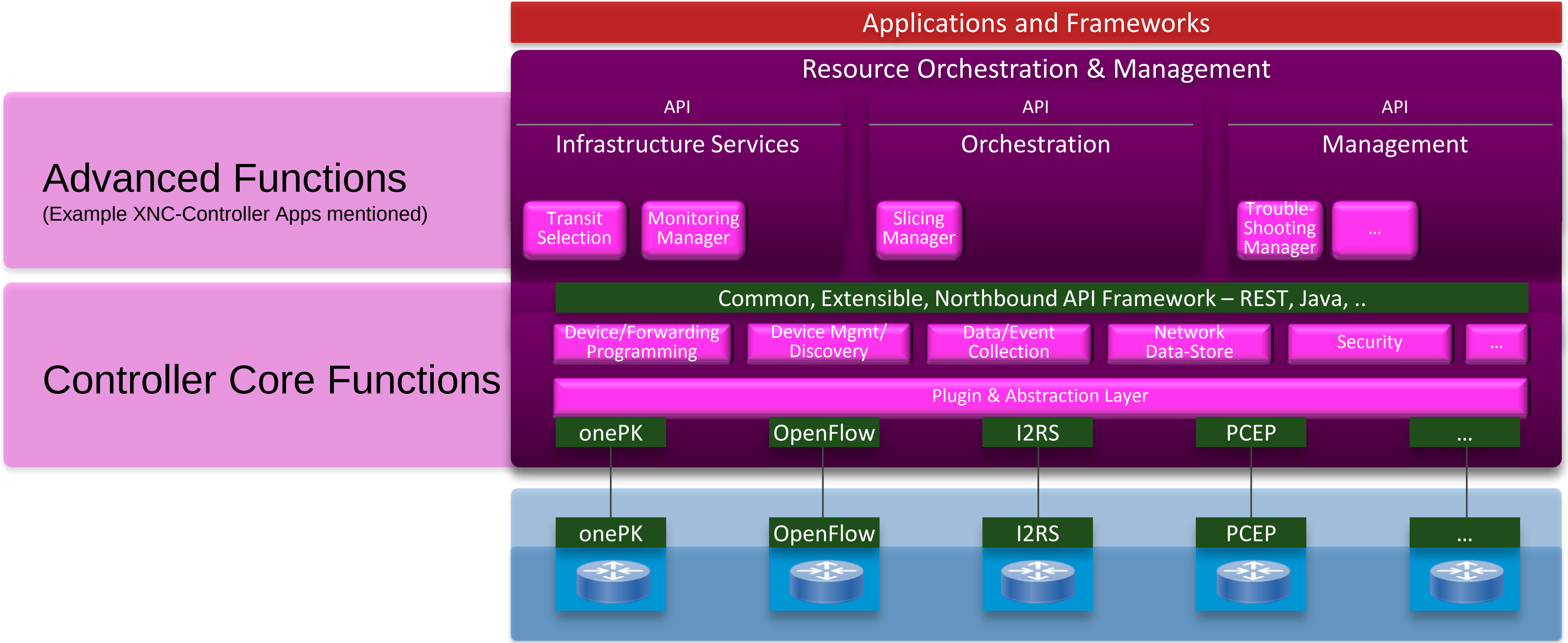
- Platform for generic network control – implements elementary infrastructure functions and enhanced apps
- Example Apps
  - Monitor Manager
  - Transit Selection (“Custom Routing”)
  - Flexible Network Partitioning and Provisioning (“Slicing”)
- Java-based



*Cisco eXtensible Network Controller is also known as “Cisco ONE Controller”

Elementary Infrastructure Functions and beyond

Extensible Network Controller (XNC) – Architecture Outline



Applications and Frameworks

Resource Orchestration & Management

Advanced Functions

(Example XNC-Controller Apps mentioned)

API

Infrastructure Services

Transit Selection

Monitoring Manager

API

Orchestration

Slicing Manager

API

Management

Trouble-Shooting Manager

...

Controller Core Functions

Common, Extensible, Northbound API Framework – REST, Java, ..

Device/Forwarding Programming

Device Mgmt/ Discovery

Data/Event Collection

Network Data-Store

Security

...

Plugin & Abstraction Layer

onePK

OpenFlow

I2RS

PCEP

...

onePK

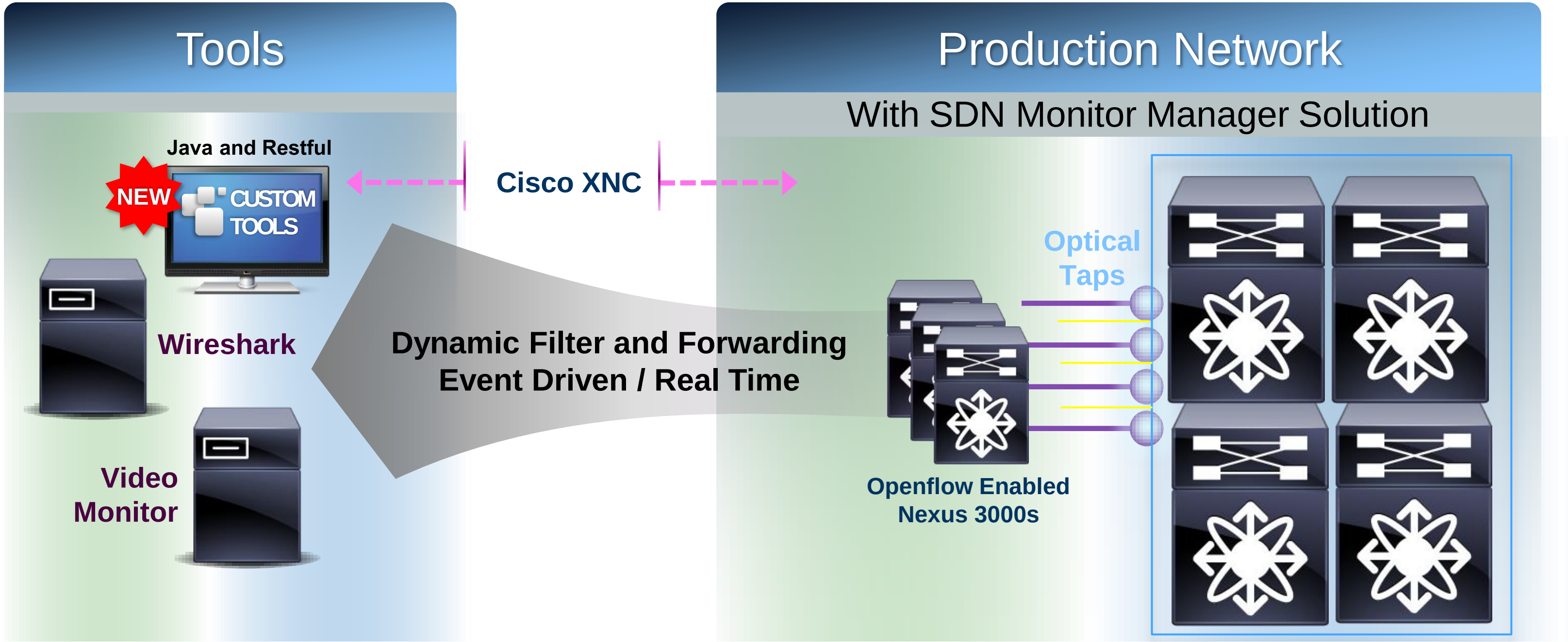
OpenFlow

I2RS

PCEP

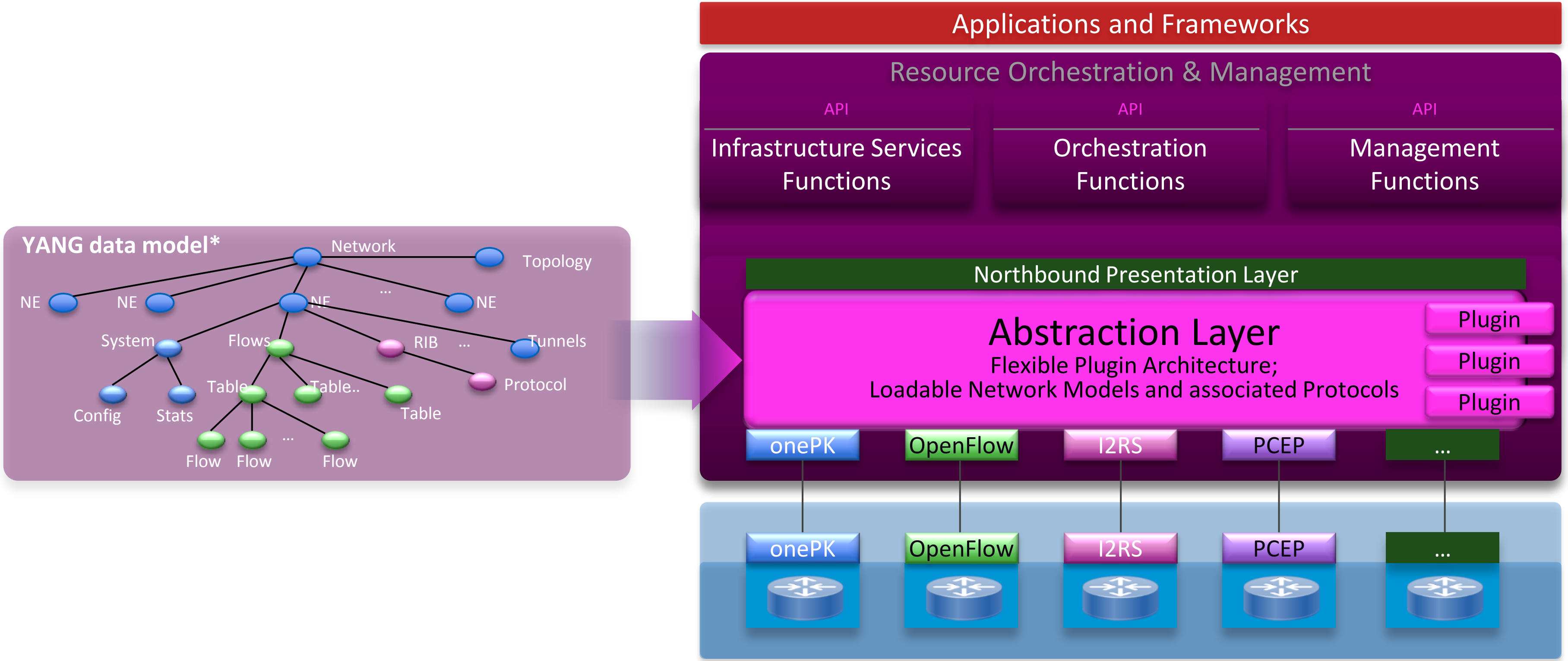
...

Monitor Manager Solution



Elementary Infrastructure Functions and Beyond

Evolving XNC



Applications and Frameworks

Resource Orchestration & Management

API	API	API
Infrastructure Services Functions	Orchestration Functions	Management Functions

Northbound Presentation Layer

Abstraction Layer

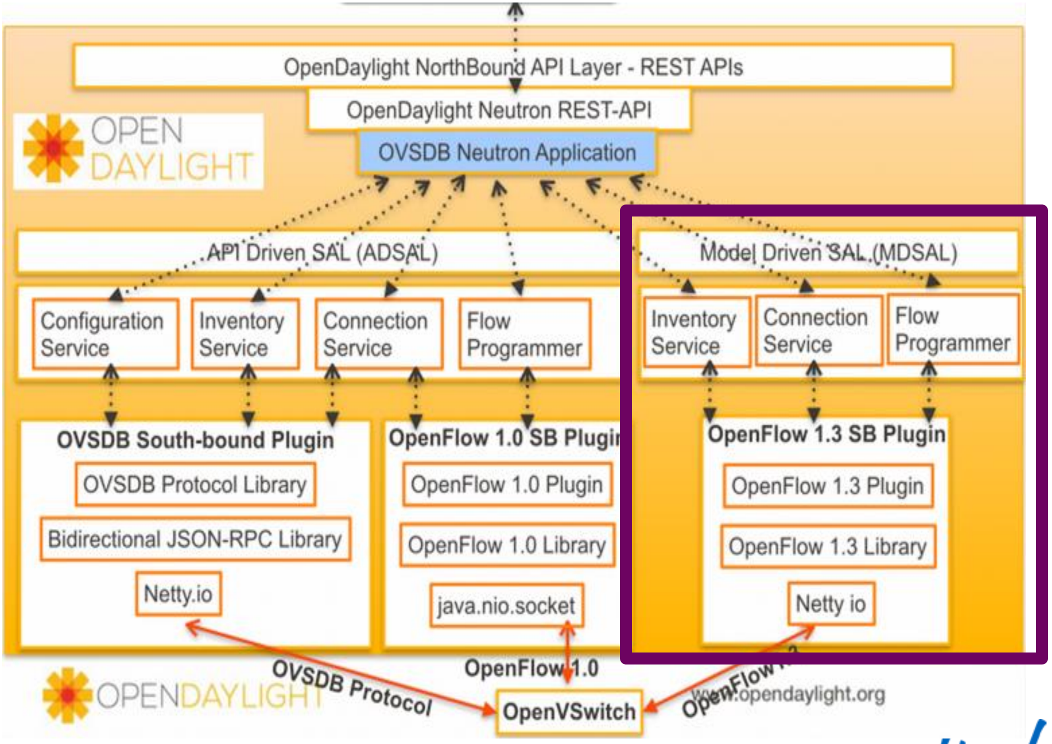
Flexible Plugin Architecture;
Loadable Network Models and associated Protocols

*http://tools.ietf.org/html/rfc6020

Elementary Infrastructure Functions and Beyond

OpenDayLight – Hydrogen

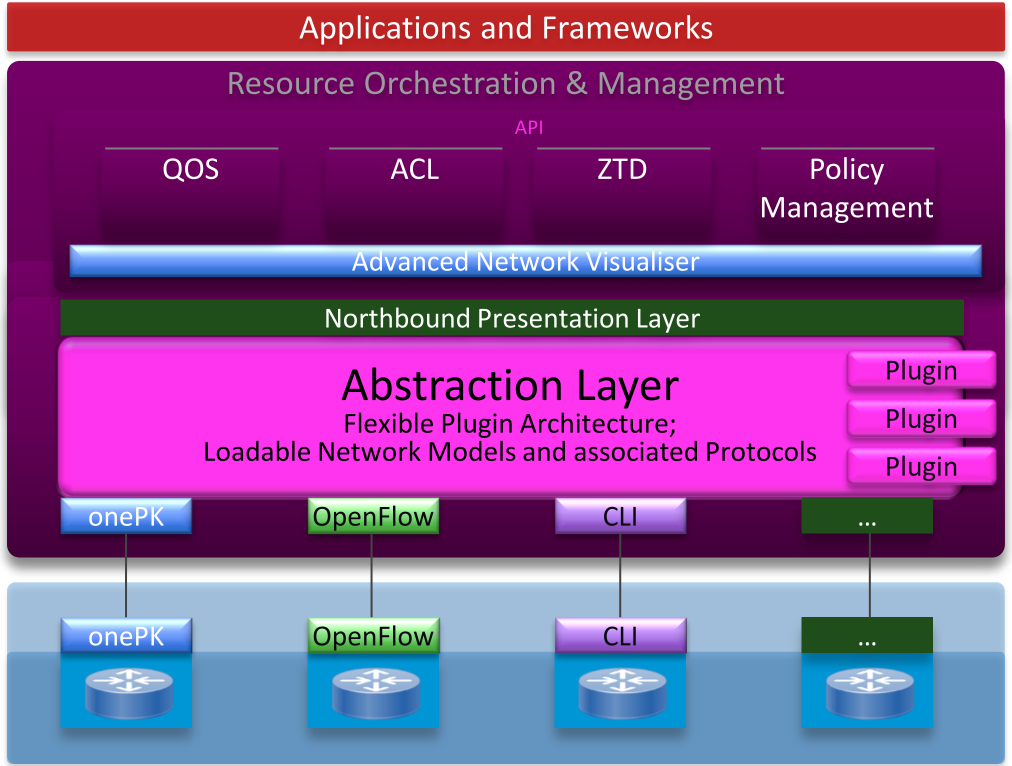
- Open Daylight - Hydrogen Controller
- Northbound Model Driven SAL coupled with model based Services
- OF 1.3 Plugin and associated API



Elementary Infrastructure Functions and Beyond

APIC – Enterprise, continuing the Architecture Evolution

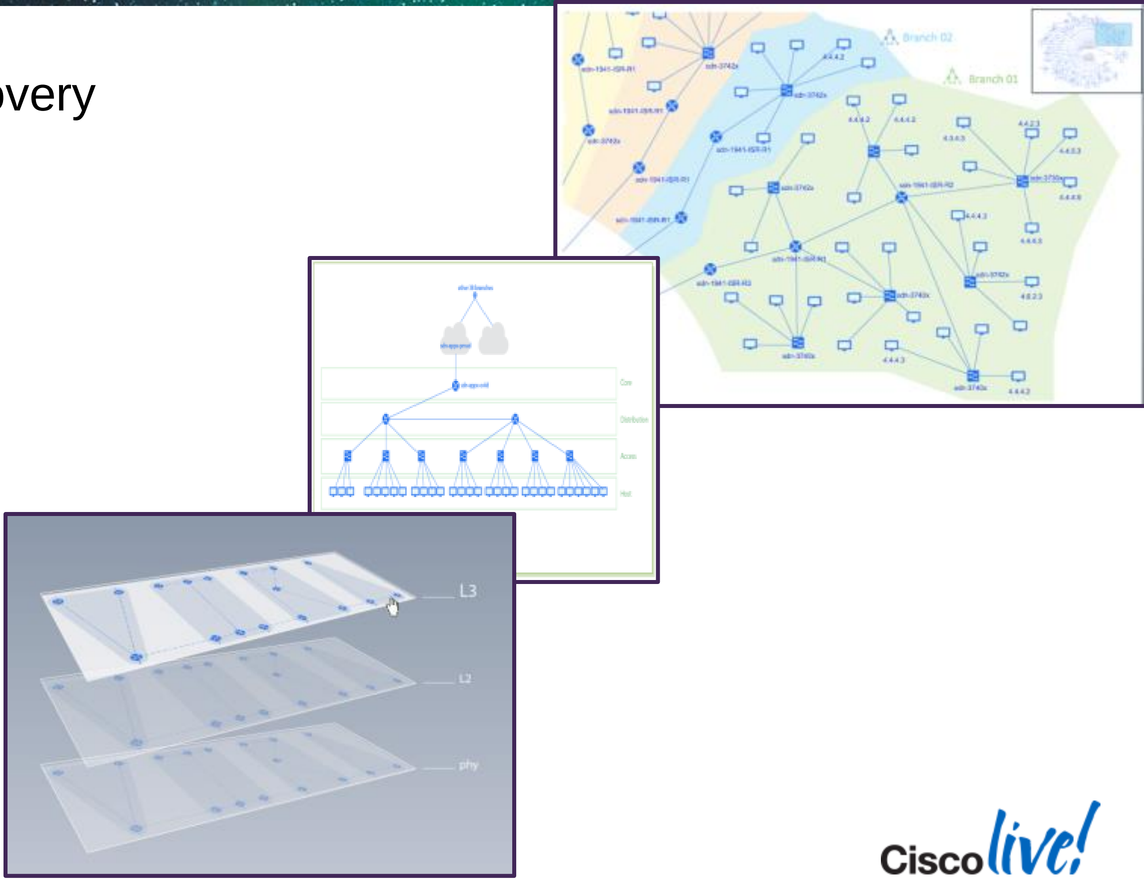
- Launched February 2014
- Enterprise specific set of “turn-key” solutions, focusing
  - Ease of Operations / Simplicity
  - Consistent Network Behaviour
  - Brownfield and Greenfield
  - Application Visibility and Control
- Examples
  - Inventory/Topology:
  - ACL Management
  - easyQoS



Orchestration, Control, Management

Example: APIC – Enterprise - Topology

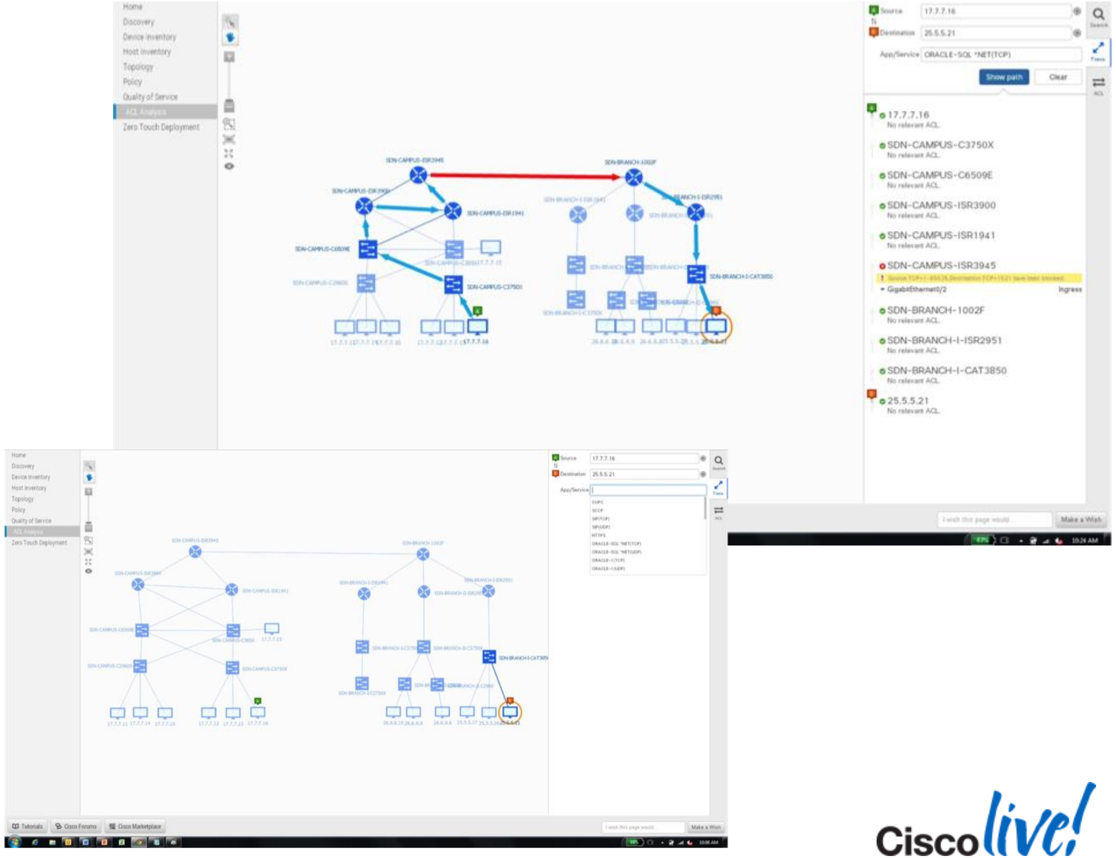
- With Inventory and discovery Services presents
- Topology 2.0
- Standard views
  - Macro
  - Micro
  - Connectivity
- More contextual view
  - L3 over
  - L2 over
  - Physical



Orchestration, Control, Management

Example: APIC – Enterprise - ACL

- ACL Management
  - Shadow ACL (duplication)
  - ACL Conflict
  - Assurance
- Flow based
  - Duplicate in flow path
  - Miss-config in flow path
- Looking forward – follow me ACL



Source 17.7.7.16
Destination 25.5.5.21
App/Service ORACLE-SQL *NET(TCP)

17.7.7.16
No relevant ACL
SDN-CAMPUS-C3750X
No relevant ACL
SDN-CAMPUS-C6509E
No relevant ACL
SDN-CAMPUS-ISR3900
No relevant ACL
SDN-CAMPUS-ISR1941
No relevant ACL
SDN-CAMPUS-ISR3945
GigabitEthernet0/2 Ingress
SDN-BRANCH-1002F
No relevant ACL
SDN-BRANCH-I-ISR2951
No relevant ACL
SDN-BRANCH-I-CAT3850
No relevant ACL
25.5.5.21
No relevant ACL

Orchestration, Control, Management

Example: APIC – Enterprise - EasyQoS

- Apps
- Classes
- Mapping
  - CVD
  - Custom

Qos Configuration

Drag and drop applications from the left into one of the four Application Classes and set Bandwidth limits for each bucket. Changes are not in effect until you hit the Save button

You can also let us do the work and set it to our Cisco Verified Design:

Set map name and Save changes: cvd

Current Map: cvd
QOS Status: disabled

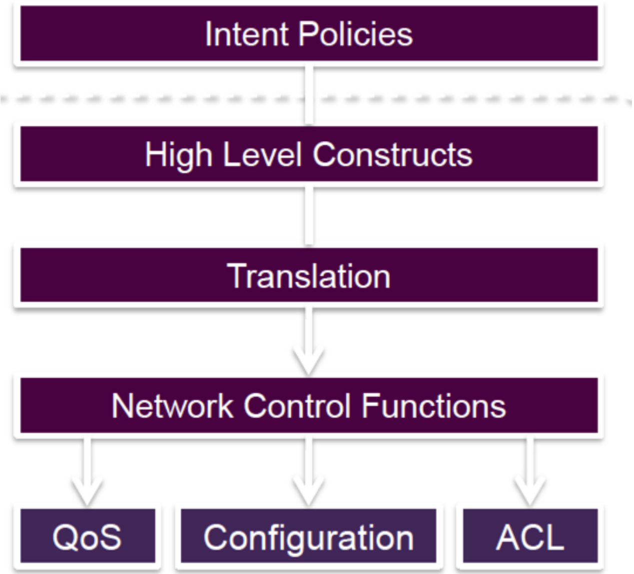
Realtime 25%	Control 25%	Transactional Data 25%	Best Effort 25%
CUPC	SCCP	CONNECTED PC BACKUP	APPLE ITUNES MUSIC SHARING(TCP)
	SIP(TCP)	FTP	APPLE ITUNES MUSIC SHARING(UDP)
	SIP(UDP)	HTTPS	BITTORRENT
		IMAP	MICROSOFT DIRECTX GAMING(TCP)
		KAZAA(TCP)	MICROSOFT DIRECTX GAMING(UDP)
		KAZAA(UDP)	MSN GAMING ZONE(TCP)
		ORACLE-1(TCP)	MSN GAMING ZONE(UDP)
		ORACLE-1(UDP)	YAHOO GAMES

Orchestration, Control, Management

Example: APIC – Enterprise – Policy Approach

- Business Intent driven Policy (intent based attributes)
  - UserID / local / device
  - App
  - Trust level
  - Experience level
  - Priority level

- Drives Network Control
  - Configuration
  - ACL
  - QoS


Intent Policies

High Level Constructs

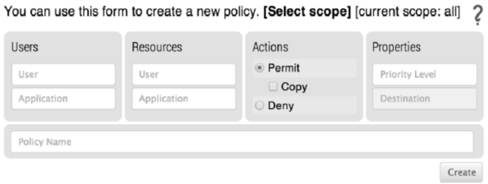
Translation

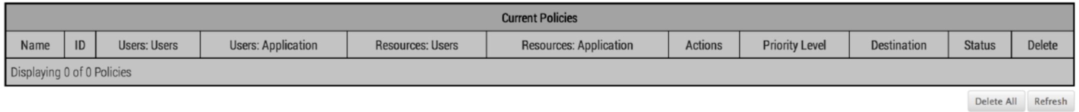
Network Control Functions

QoS

Configuration

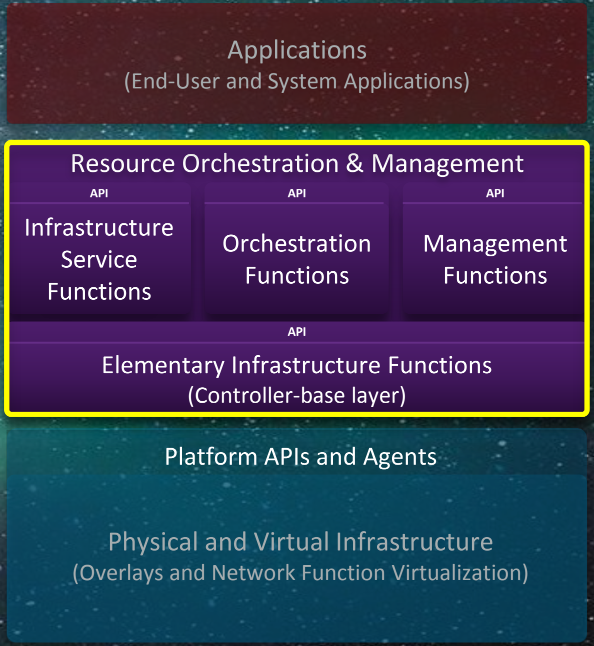
ACL


You can use this form to create a new policy. **[Select scope]** [current scope: all]



Current Policies											
Name	ID	Users: Users	Users: Application	Resources: Users	Resources: Application	Actions	Priority Level	Destination	Status	Delete
Displaying 0 of 0 Policies											

Open Network Environment Qualities
Resource Orchestration – Controllers
APIC – Data Centre


Applications
(End-User and System Applications)

Resource Orchestration & Management

API

Infrastructure Service Functions

API

Orchestration Functions

API

Management Functions

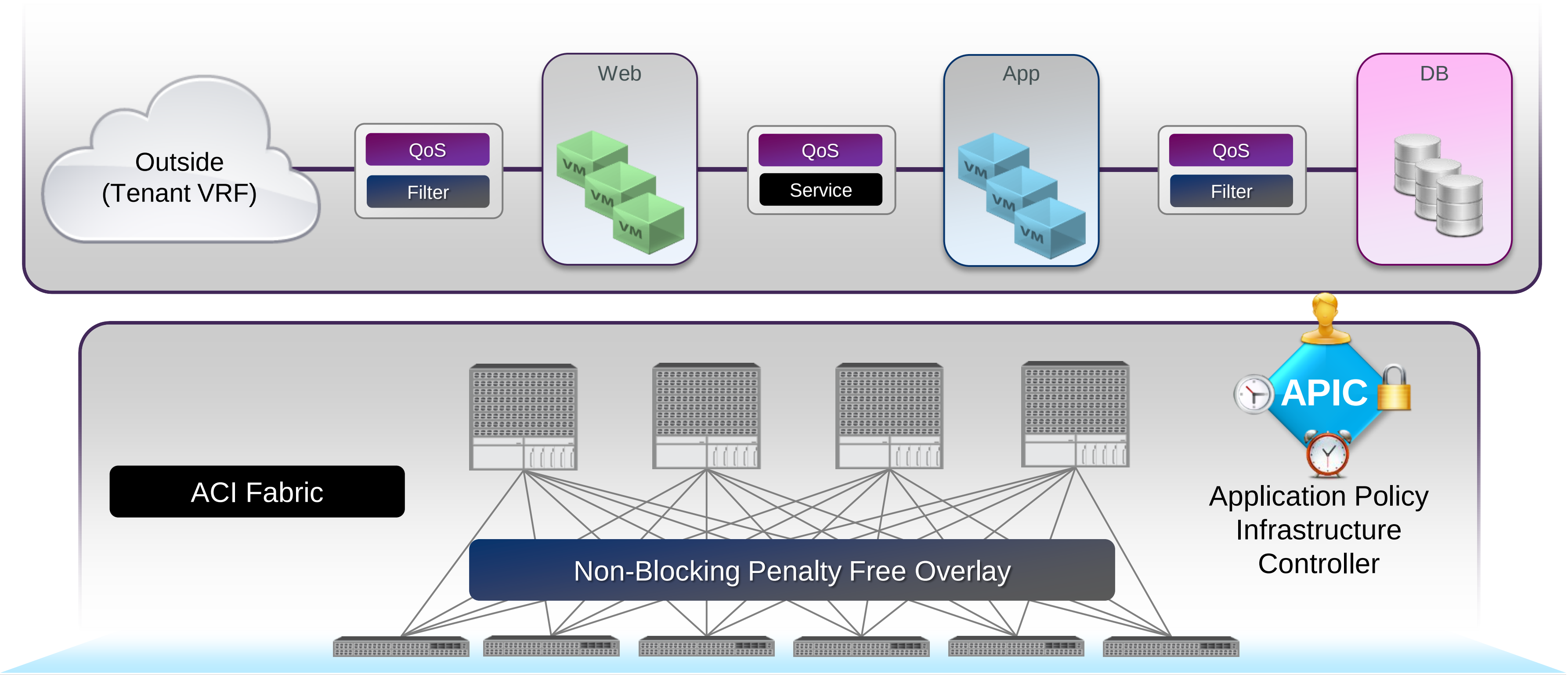
API

Elementary Infrastructure Functions
(Controller-base layer)

Platform APIs and Agents

Physical and Virtual Infrastructure
(Overlays and Network Function Virtualization)

Stateless Hardware

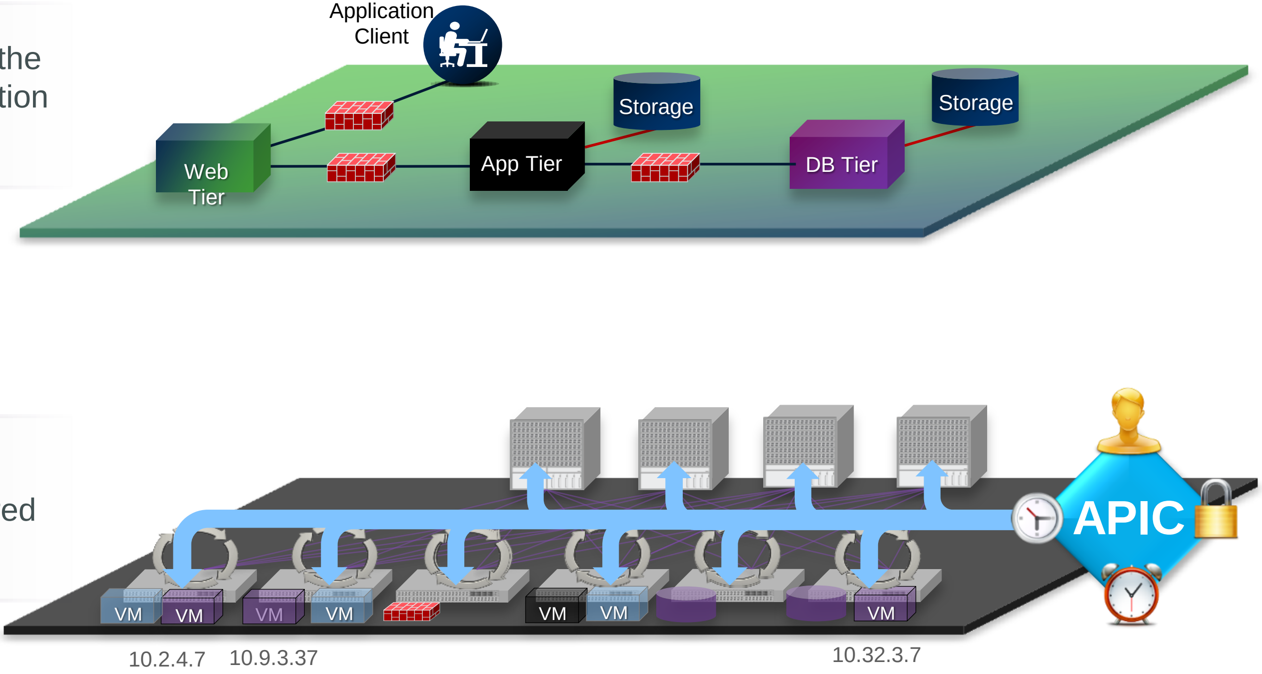


Application Policy Model and Instantiation

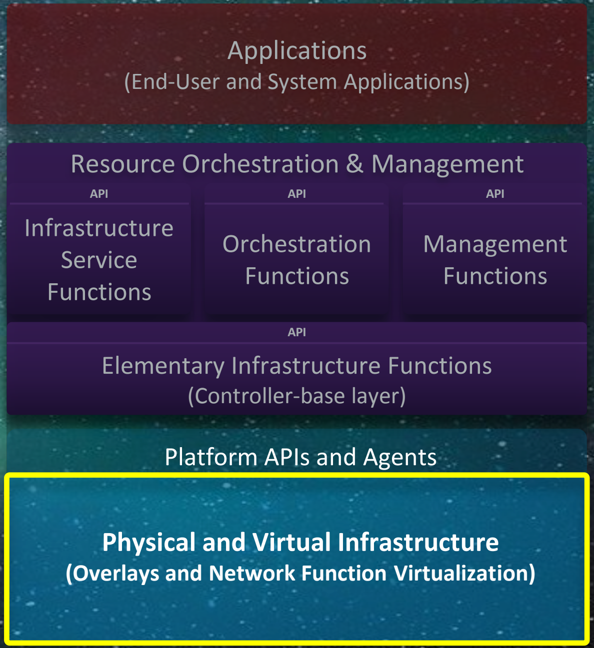
Example: APIC – Data Centre Controller

Application policy model: Defines the application requirements (application network profile)

Policy instantiation: Each device dynamically instantiates the required changes based on the policies



Open Network Environment Qualities
Network Infrastructure Virtualisation



Applications
(End-User and System Applications)

Resource Orchestration & Management

API | API | API

Infrastructure Service Functions

Orchestration Functions

Management Functions

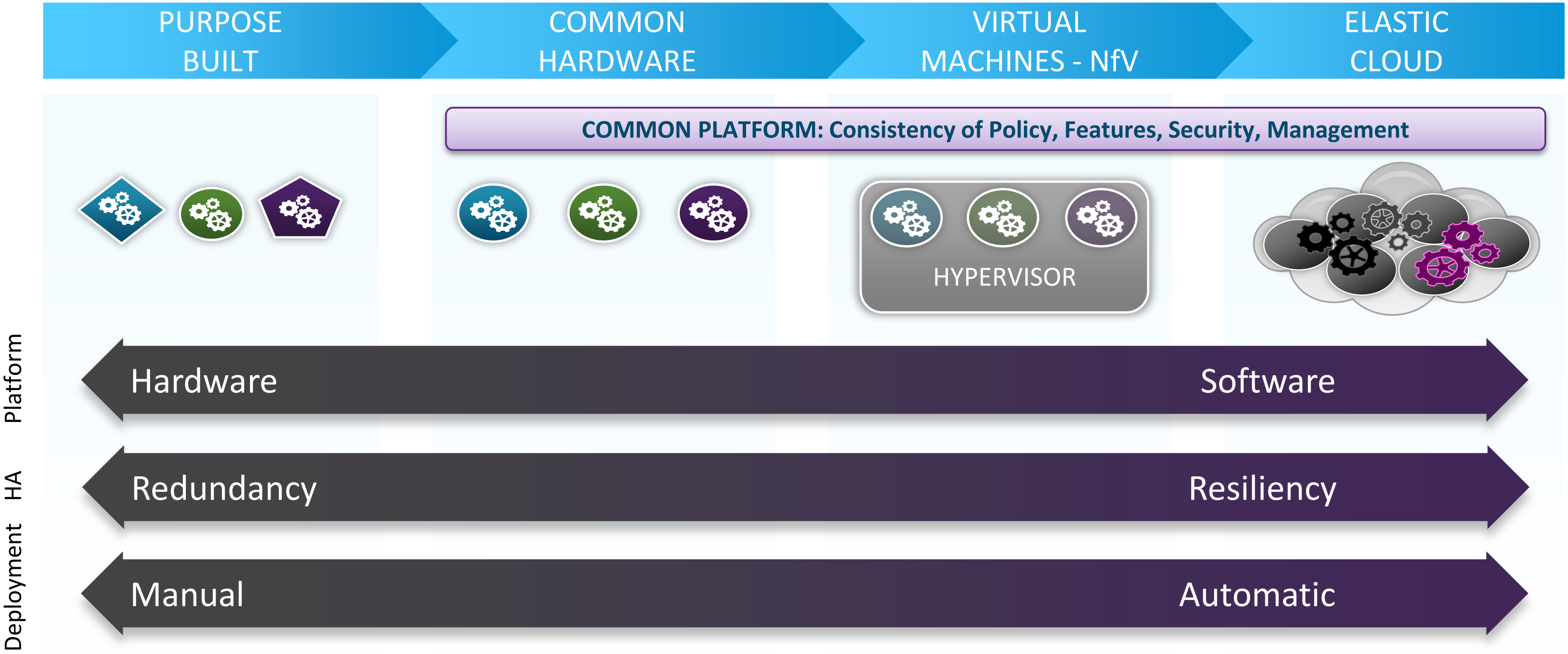
API

Elementary Infrastructure Functions
(Controller-base layer)

Platform APIs and Agents

Physical and Virtual Infrastructure
(Overlays and Network Function Virtualization)

Physical, Virtual, Cloud Evolution



PURPOSE BUILT

COMMON HARDWARE

VIRTUAL MACHINES - NfV

ELASTIC CLOUD

COMMON PLATFORM: Consistency of Policy, Features, Security, Management

HYPERVISOR

Platform: Hardware → Software

HA: Redundancy → Resiliency

Deployment: Manual → Automatic

Evolve: Engineering, Operations, Architecture

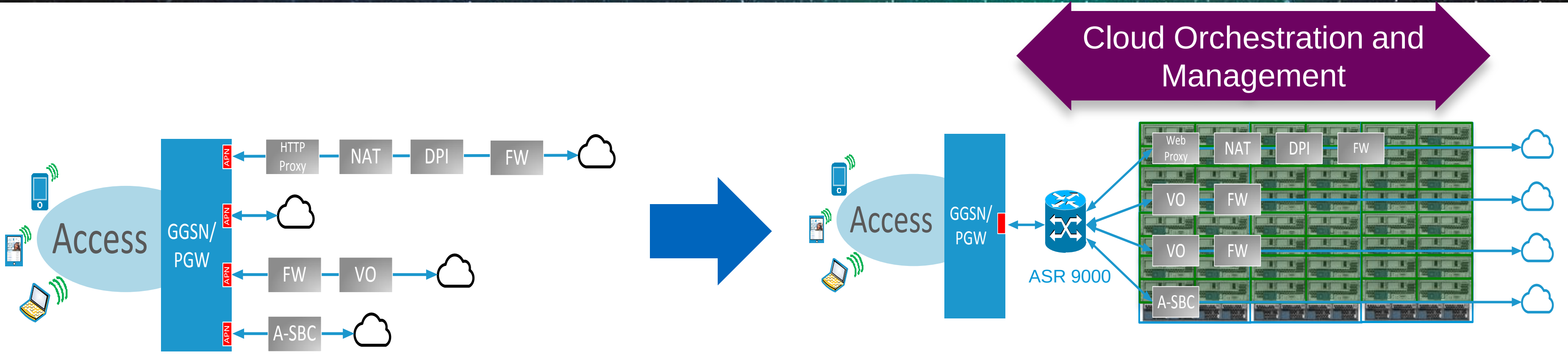
Physical and Virtualised Network Functions

Network Function Virtualisation – NFV: Examples

Nexus/Catalyst	ASR/ISR/CRS	Identity/Policy - ISE	Firewall - ASA
vSwitch (Nexus 1000v)	*vRouter (CSR1000v)*	*vISE*	*ASAv (ASA 1000v)*
WAAS	Email Security - ESA	Wireless LAN Controller	Security Gateway
vWAAS	*vESA*	*vWLC, vMSE*	*VSG*
Video Cache	Web Security - WSA	Network Analysis - NAM	IOS/XR RR
vVideoCache	*vWSA*	*vNAM*	*vRouteReflector*

NfV in Mobile Cloud Evolution

Example: virtual GI-LAN

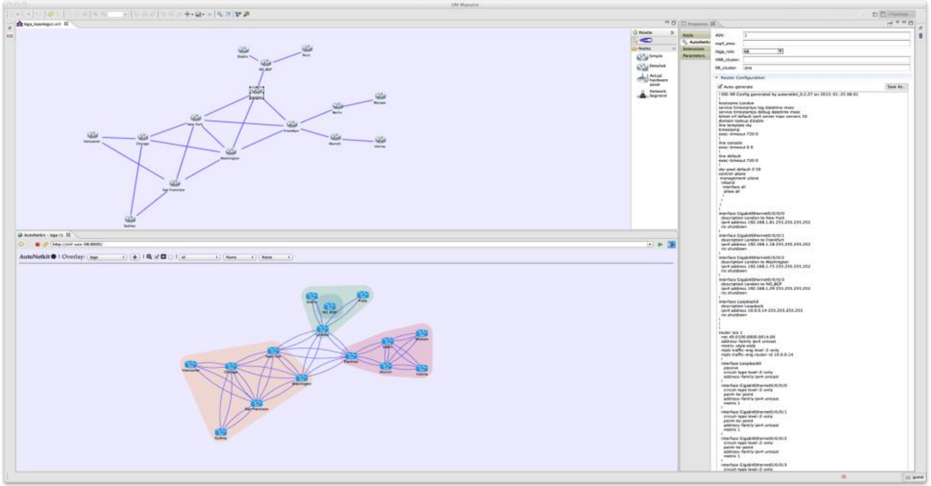


- Cost model based on subscriber count + base cost of commodity hardware
- Fault tolerance and high availability based on hypervisor tools
- Simple reconfiguration of service chains via infrastructure software and virtualisation tools
  - vertical scaling and horizontal scaling (adjusting capacity)

Cisco Modeling Labs

Development Environment for Cisco ONE

- Is a multi-purpose network virtualisation platform
- Brings virtual machines running Cisco Network Operating Systems to the customer
  - The same operating systems as used on physical Cisco products: IOS, IOS-XR, NX-OS
- Virtual Machine orchestration capabilities enables:
  - Creation of highly-accurate models of real-world or future networks – scales to thousands of virtual network devices

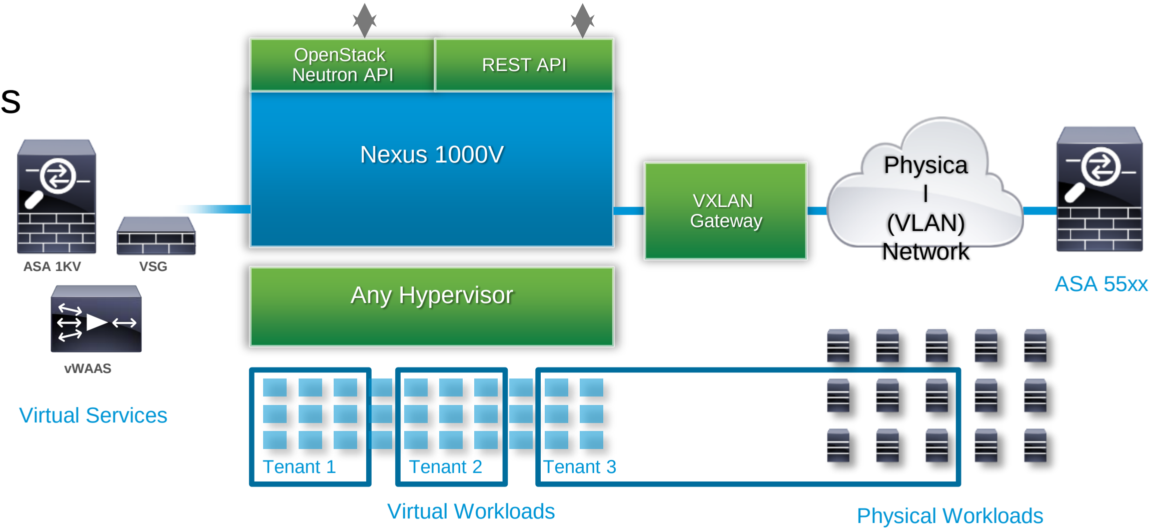


SP / Enterprise	Production Network Modeling	'What-if' Analysis	Test Lab Virtualization
Partner Community	Training and Education	Cisco onePK Virtual Testbed	Test Lab Virtualization
University and Education	Networking Research	Rapid Prototyping	Network Education

Virtualisation

Virtual Overlay Networks – Example: Nexus 1000v

- Example: Virtual Overlay Networks and Services with Nexus 1000V



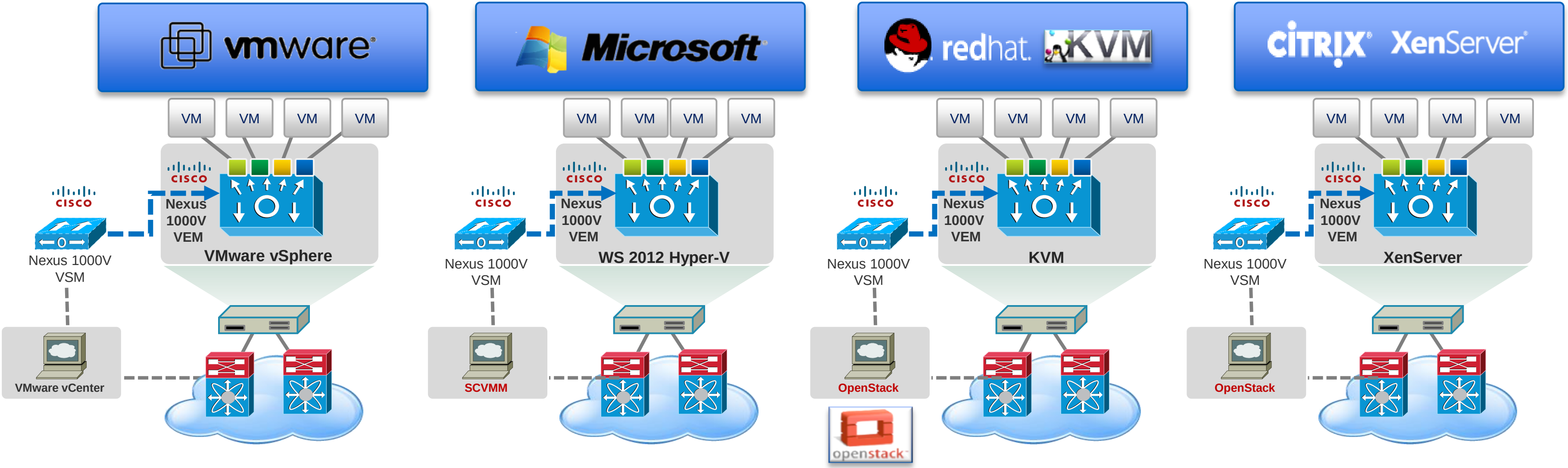
- Large scale L2 domains:
  Tens of thousands of virtual ports
- Common APIs
  - Incl. OpenStack Neutron* API's for orchestration
- Scalable DC segmentation and addressing
  - VXLAN
- Virtual service appliances and service chaining/traffic steering
  - VSG (cloud-ready security), vWAAS (application acceleration),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

*a.k.a. Quantum

Virtualisation

Host-based Virtual Overlay Networks – Hypervisor agnostic

- Example: Virtual Overlay Networks and Services with Nexus 1000V



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

Summary

What is SDN?

Software *Defined* Networking

Software Defined Networking

A Great Enabler

A Few References

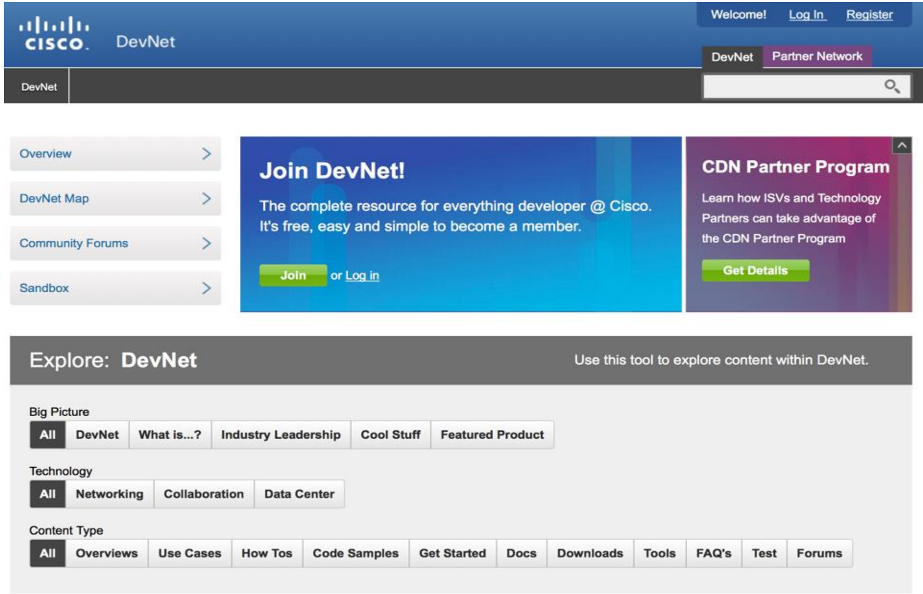
- Cisco Open Network Environment
  www.cisco.com/go/one
- Cisco Application Centric Infrastructure
  http://www.cisco.com/go/aci
- onePK
  www.cisco.com/go/onepk, developer.cisco.com/web/onepk
- OpenDayLight
  http://www.opendaylight.org/
- XNC
  www.cisco.com/go/xnc, developer.cisco.com/web/xnc/home
- APIC Enterprise
  http://www.cisco.com/go/apic_enterprise

Cisco DevNet – Cisco's New Developer Program

- All developer resources are now in one central location
  - Comprehensive API Index
  - Forums
  - Developer Sandbox
  - FAQs
  - Access to support, and more

- Interactive new portal makes finding the information and support faster and easier

- Register at https://developer.cisco.com



Join DevNet!

The complete resource for everything developer @ Cisco. It's free, easy and simple to become a member.

Join or Log in

CDN Partner Program

Learn how ISVs and Technology Partners can take advantage of the CDN Partner Program

Explore: **DevNet**

Use this tool to explore content within DevNet.

Big Picture: All | DevNet | What is...? | Industry Leadership | Cool Stuff | Featured Product

Technology: All | Networking | Collaboration | Data Center

Content Type: All | Overviews | Use Cases | How Tos | Code Samples | Get Started | Docs | Downloads | Tools | FAQ's | Test | Forums

Q & A

Complete Your Online Session Evaluation

Give us your feedback and receive a Cisco Live 2014 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 21 March 12:00pm - 2:00pm

Learn online with Cisco Live!

Visit us online after the conference for full access to session videos and presentations. www.CiscoLiveAPAC.com

CISCO TM