CORD: Central Office Re-Architected as a Data Center

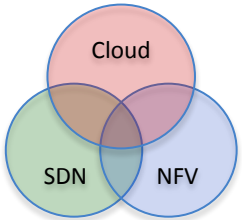
Uyen Chau
uyen@opennetworking.org

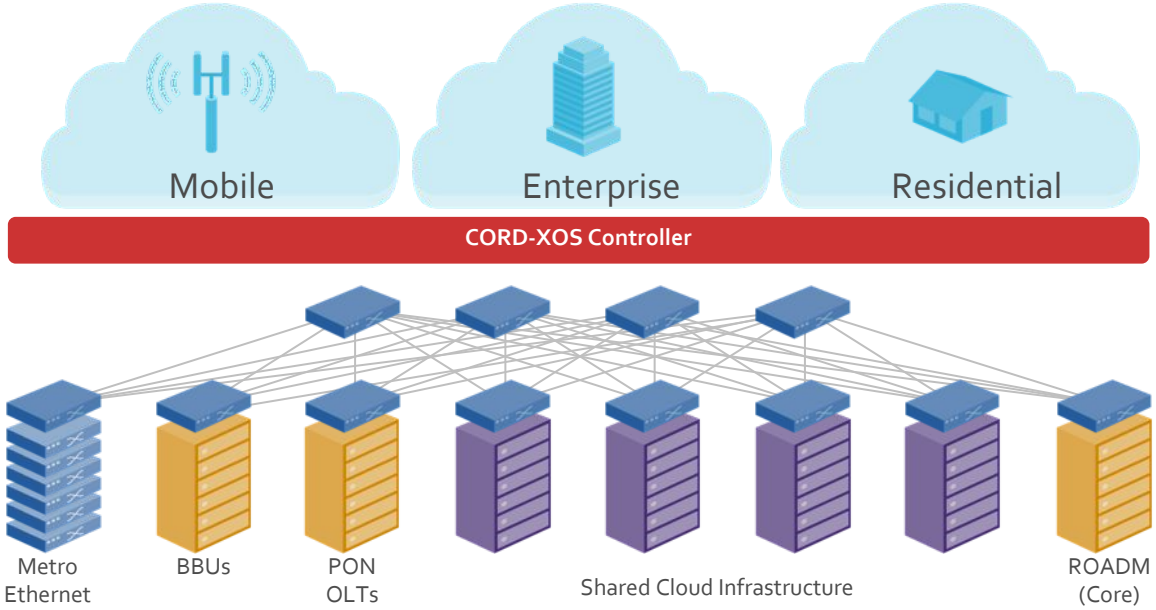
CORD High Level Architecture

Large number of COs

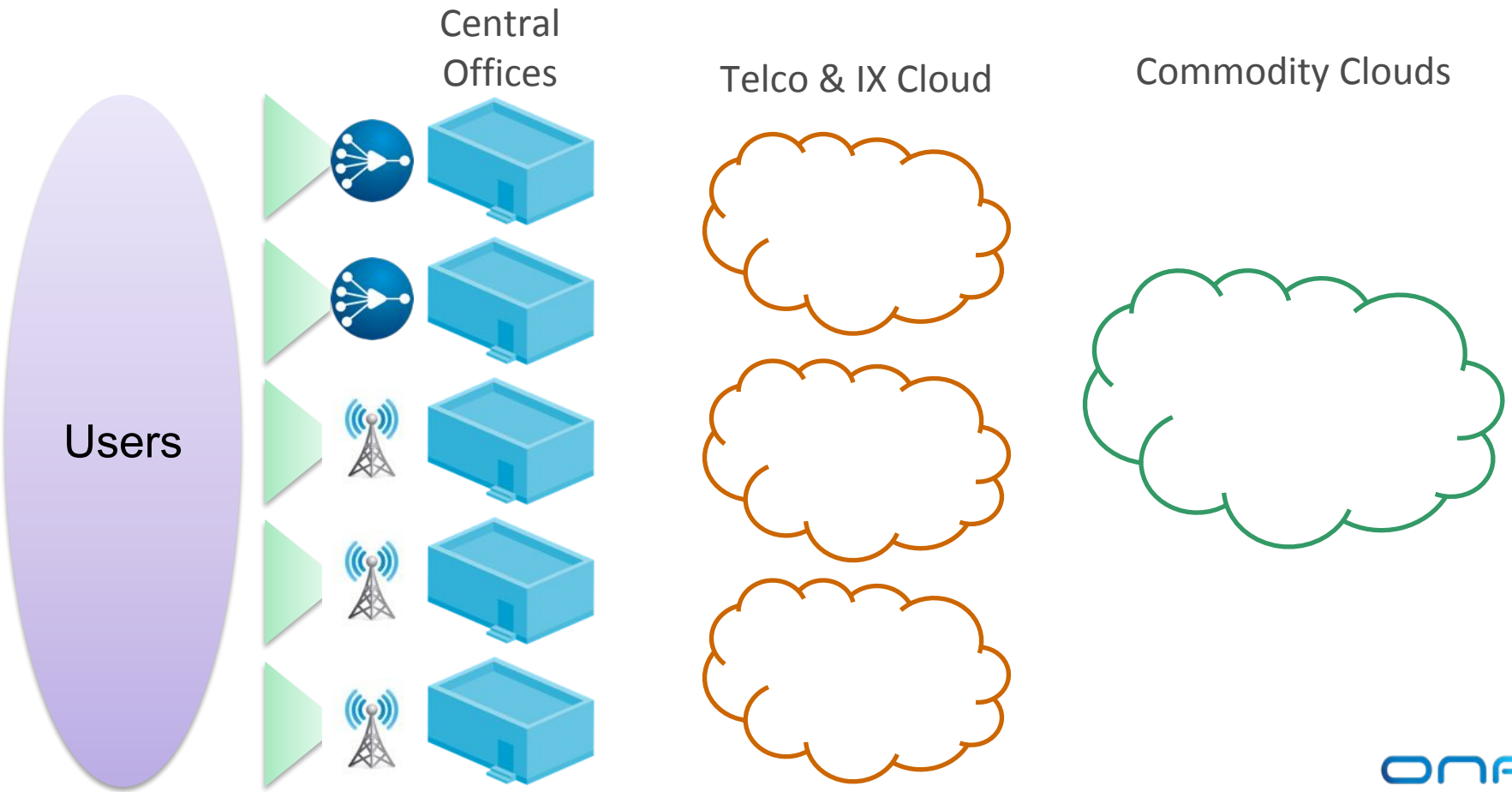
Evolved over 40-50 years

300+ Types of equipment
Huge source of CAPEX/OPEX

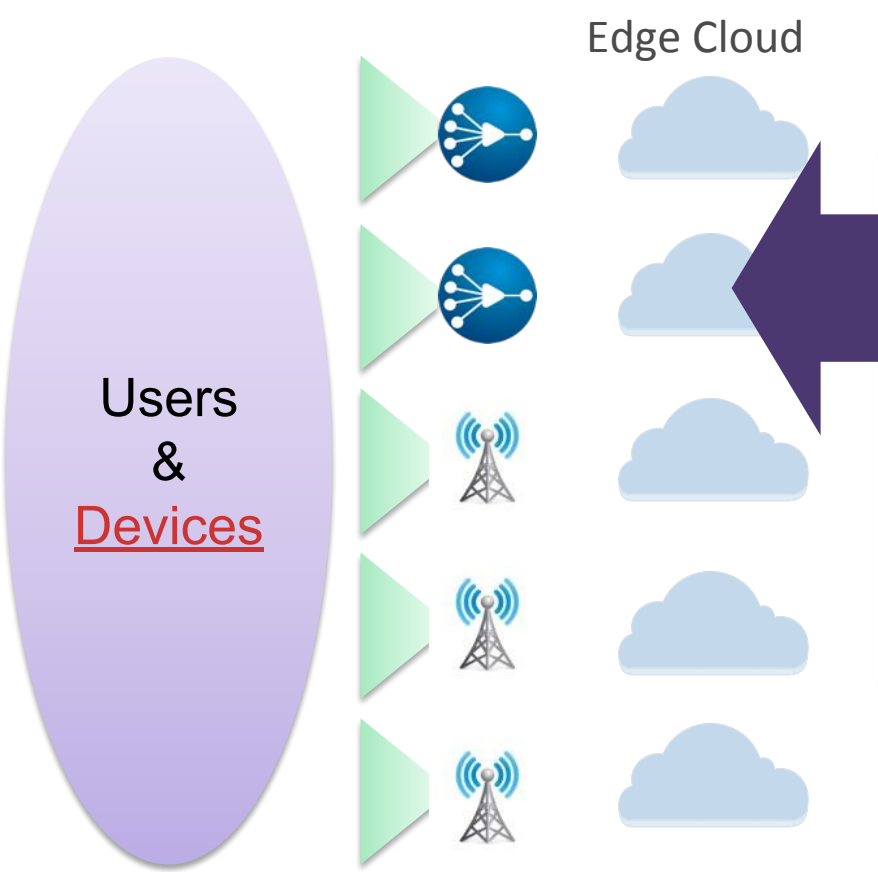




The Current Edge



Emerging Multi-Tier Cloud with New Edge



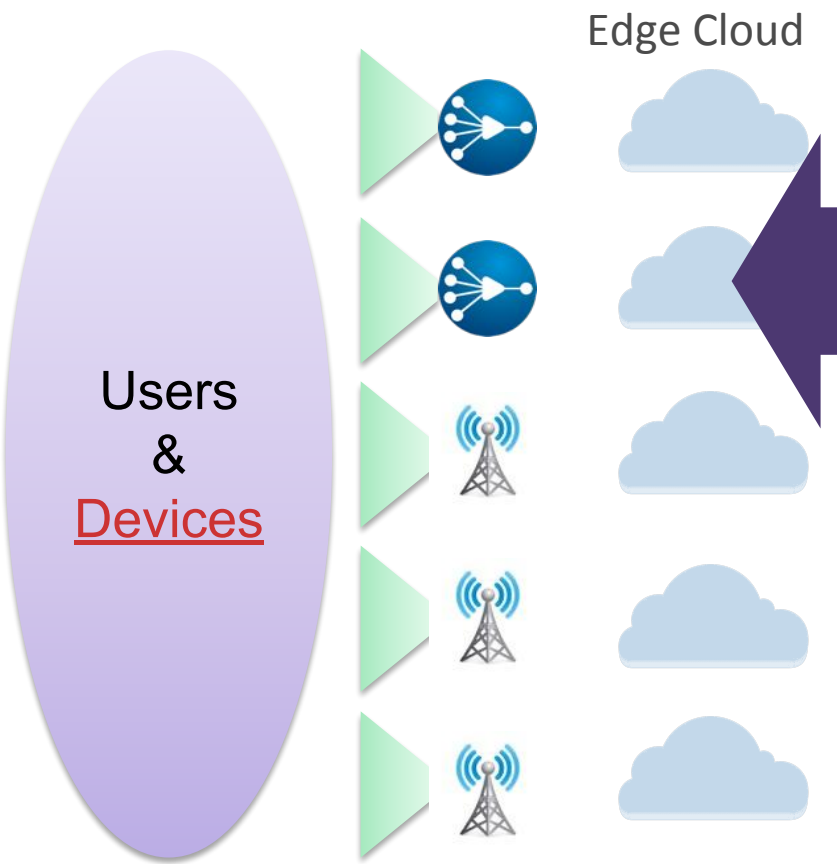
Edge Processing is Vital

- Subscriber experience dictated from here

- Human Reaction time: 210ms
- Latency to Centralized Cloud: 100 – 400ms

- Emerging Applications Require Edge Processing
  - AR visual overlays
  - Autonomous Vehicle Coordination
  - IoT Battery Life (50-75% improvement with edge processing)

Emerging Multi-Tier Cloud with New Edge



Edge Processing is similar & different

- Sum of all Edge >> all of cloud
  - The same economics apply
- Edge processing requires
  - Interfacing to wireless RAN & wireline
  - Real time performance
  - Processing of data from physical devices
  - Use machine learning: sense, infer, act
  - Leveraging cloud for learning, archiving, …

Requirements for the New Edge?

Functionality

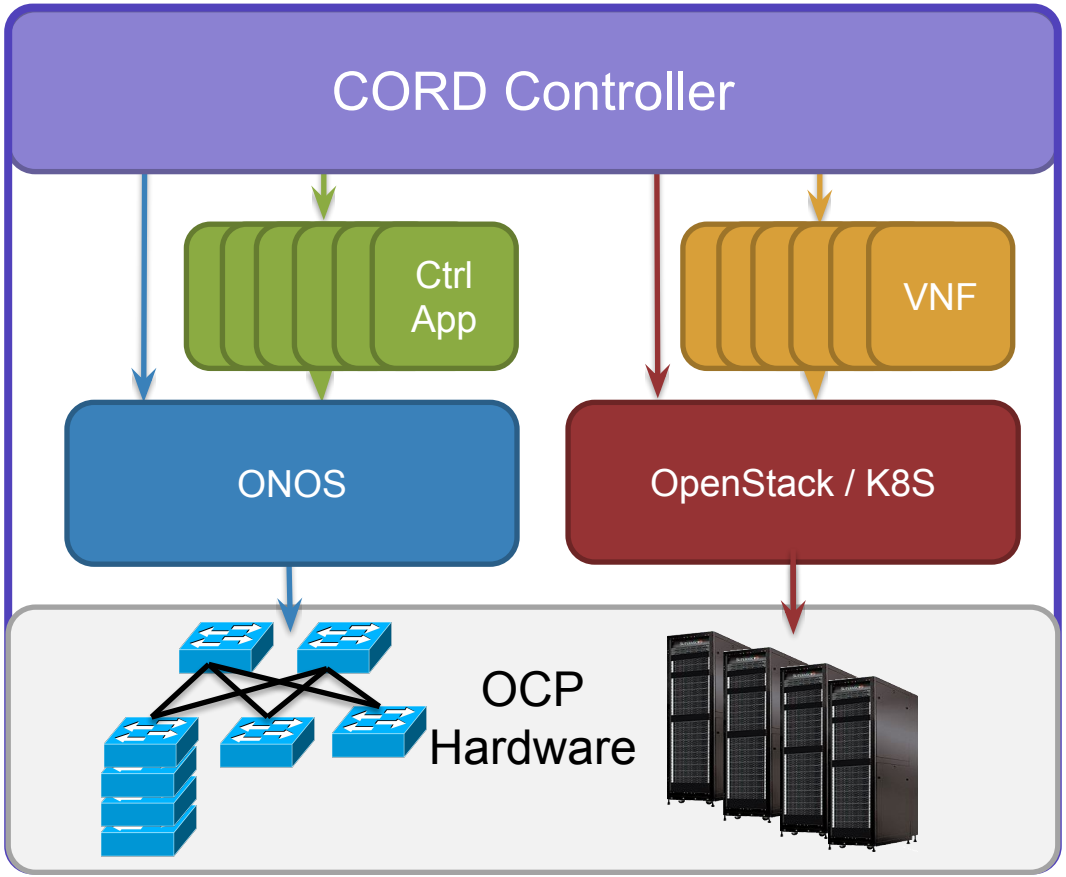
- A service delivery platform
  - For known & yet unknown services
- Many different configurations
  - Small to large
- Ability to plug-in different access devices/technologies
- Programmable control & monitoring
  - Millisecond control loops
- Economics of a datacenter
  - Space and power efficient
- Zero-touch/automated provisioning, config, & operation

Approach

- Built with
  - Merchant silicon
  - White boxes
  - Open source
- Vibrant community
- Future proof
  - Hard to predict services & access technologies
- Proprietary components as “tabasco sauce”

CORD - Platform

CORD Architecture



Control applications on ONOS (Trellis)

Provisions Services
Mediates Trust
Enforces Policies
Assembles Data Paths

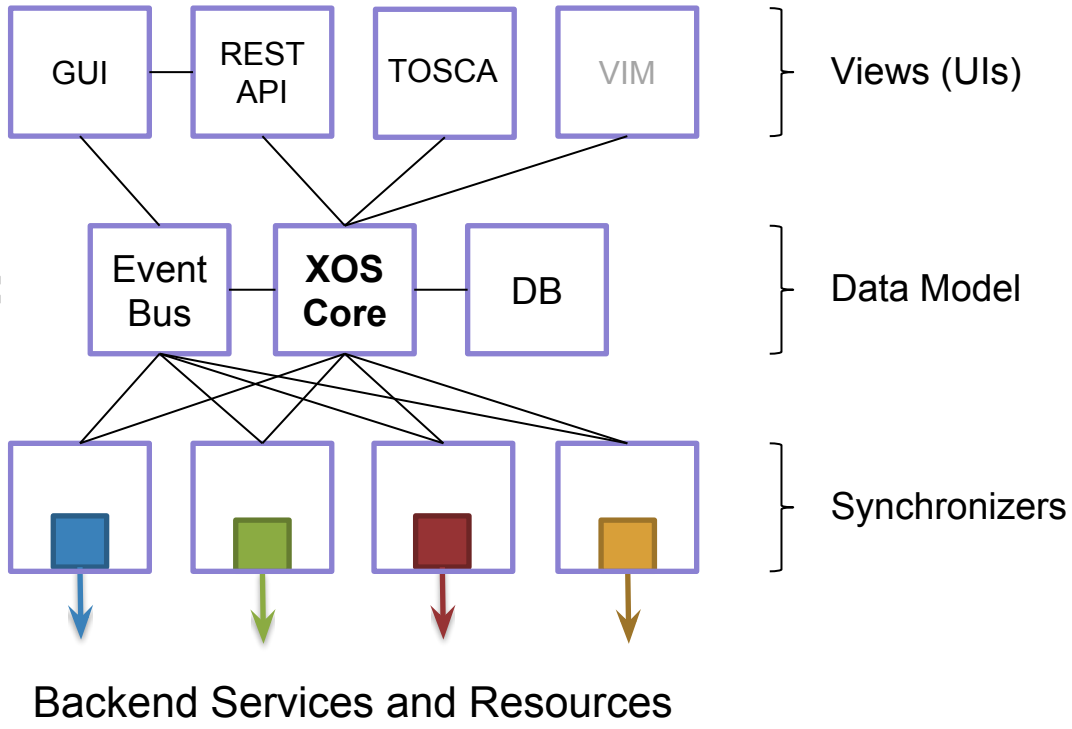
VNFs on Compute Nodes (vSG)

Commodity Servers
White-Box Switches
Merchant Silicon
Access

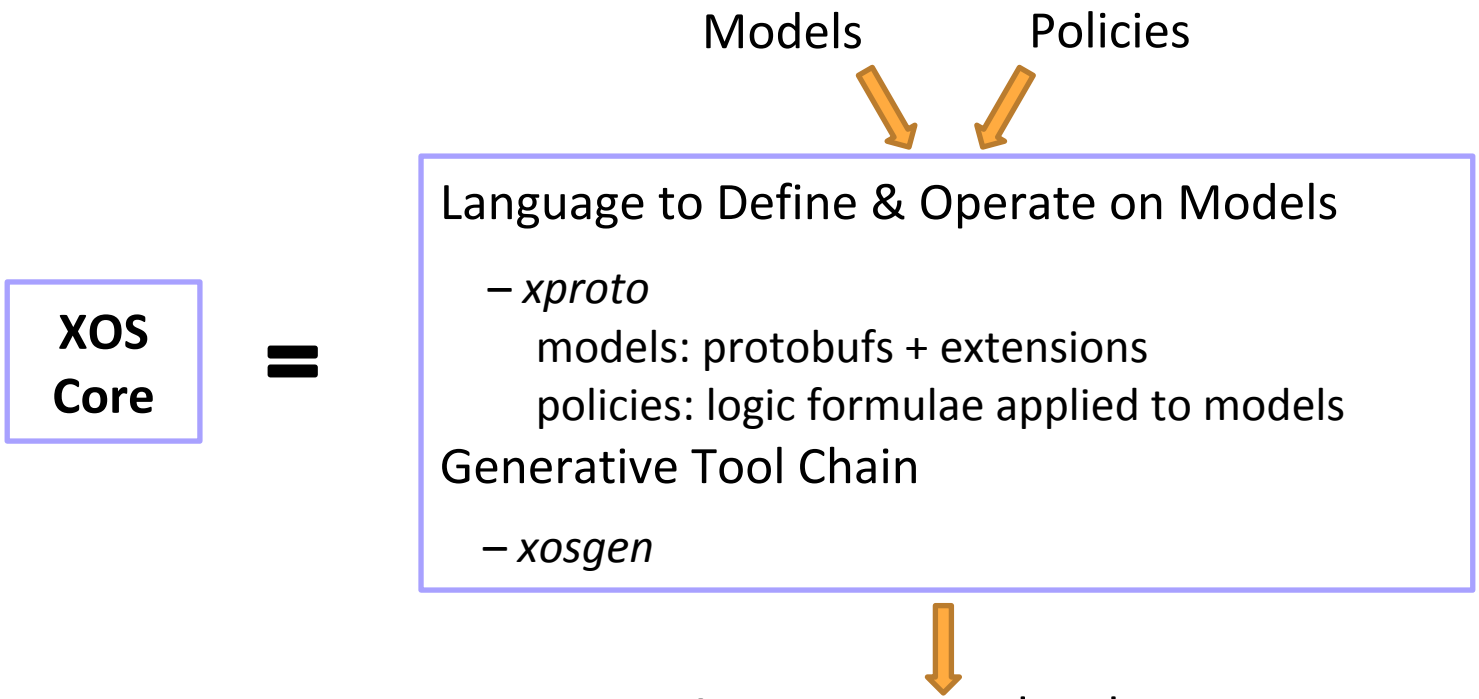
XOS - Constructed from Micro-Services

CORD Controller (XOS)

=



XOS Generative Toolchain



Synchronizer Framework

XOS auto-generates code for…

- Dependency management
- Error recovery
- Work partitioning
- Parallelization
- Logging

Service developer writes…

- A *Sync_Step( )* that is invoked when Service model changes
- An *Ansible Template* that specifies a VNF-specific playbook

Data Model

The design of CORD is driven by the data model

Modeling is how

- you make your service available on a CORD pod
- services learn about each other
- your service exposes configuration to the operator
- your service learns about resources (nodes, etc) on the pod

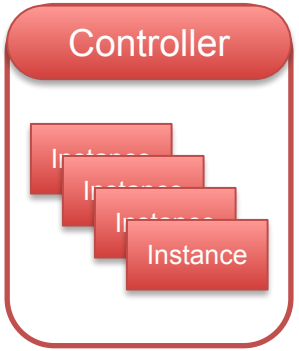
CORD also includes a set of core models

- Resources - *Instances*, *Networks*, *Slice*
- Subscribers – *CordSubscriberRoot*
- Service graph – *Service, ServiceDependency*
- Per-subscriber service chains – *ServiceInstance*, *ServiceInstanceLink*

Core Models

Instance =
(VM | Container |
Container-in-VM)

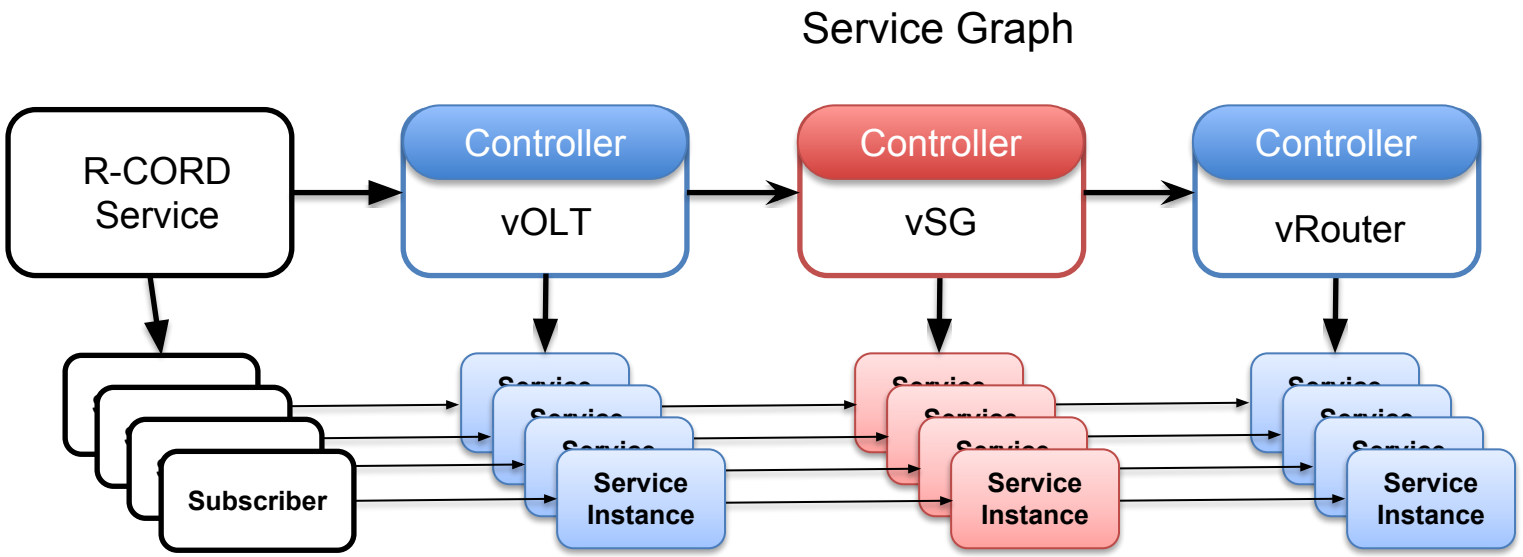
Slice =
(Instances[ ] + Networks[ ])



Service =
(Controller + Slices[ ])

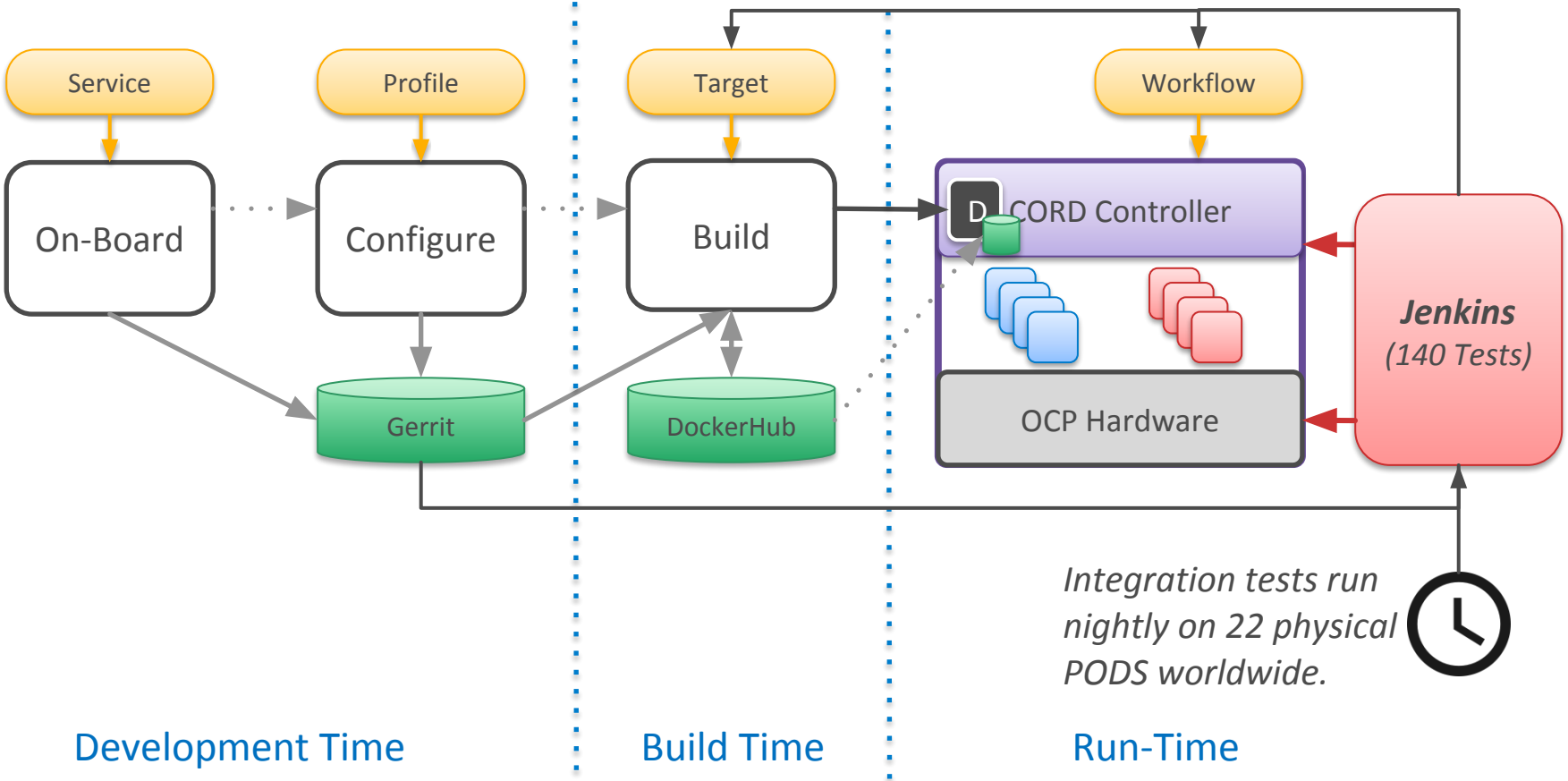
Service Graph =
(Services[ ] + Dependencies[ ])

Core Models



Service Chain =
(ServiceInstances[ ] + ServiceInstanceLinks[ ])

Multi-Stage CI/CD Pipeline



Multi-Stage Build System

Configure

```
cord_profile: rcord, mcord, ecord,…
cord_scenario: local, mock, single, cord,…
```

Build

- Fetch – onto development machine
- Build – containers (if necessary)
- Publish – to repository on head node

Multi-Stage Build System

Deploy

- Run management containers (XOS, ONOS, OpenStack) on head node
- Docker Compose today / plans to have Kubernetes help manage

Boot

- Bring up compute nodes and switches
- Leverage MAAS and PXE

Build Tooling

A set of *YAML* files represents all configuration state

→ All builds start at `build/podconfig/`*profile-scenario*`.yml`

A set of *Docker images* define the canonical representation of the system

→ Makes it easier to identify “golden” components

→ Makes it easier to iterate on a specific component during development

A set of *Ansible roles* separates configuring/installing/deploying containers

→ Makes it easy to adapt CORD to new scenarios

A sequence of *Make targets* represent build milestones

→ Makes it easy to roll back and incrementally re-build

Scenarios

Scenarios: The hardware topology and software environment to deploy.

Determines the “feature matrix” installed *external* to XOS, the inventory of nodes, and how those nodes are used

	XOS	ONOS	MaaS	OpenStack	Build VM
mock	✔				
single	✔	✔			
cord	✔	✔	✔	✔	✔
preppedpod	✔	✔		✔	

Scenarios, cont.

Scenarios are stored in `build/scenarios/<name>/`

- `config.yml` - Configuration
- `Vagrantfile` - VM definitions

Profiles

Profiles: The service graph that makes up a CORD service set

The traditional R-CORD, E-CORD, M-CORD, etc. sets of services and how they're connected together.

Stored in:
`build/platform-install/profile_manifests/<name>.yml`

CORD 5.0 Released 15-Feb-2018

Platform

- Support for dynamically loading new services and service profiles into a running system
- Refactored build to isolate Profiles (e.g., Models, TOSCA templates)
- Support for logging and diagnostics
- Preliminary integration of Kubernetes (alpha version included in release)

Use Cases

- M-CORD: MWC Demos: Integrated physical eNB + open source MME, HSS, and ProgRAN
- R-CORD: Documented how to manually bring up VOLTHA in R-CORD
- E-CORD: Automated monitoring at EVC creation

Dynamic loading of a new service

Definition and preparation phase of a VNF:

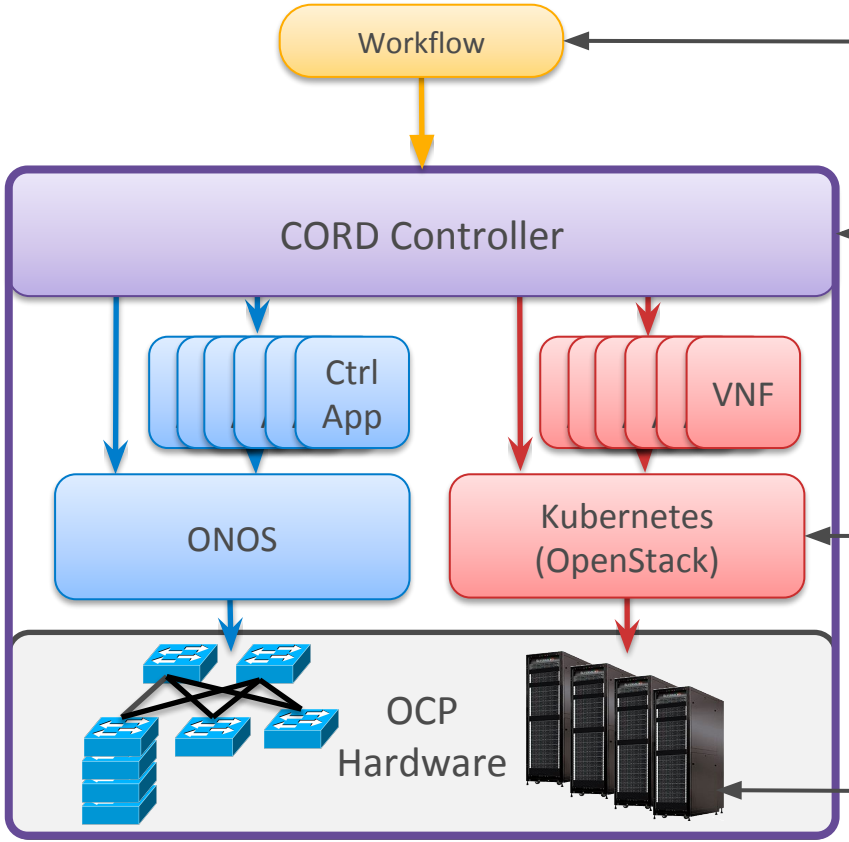
- VNF code → VM, general application, ONOS/SDN controller APP
- xProto file → Service Model Description
- XOS synchronizer → Python file to react to data model changes.
- (optional) GUI extension → Bring in new GUI elements to control your service

Onboard Phase

- Build VMs and Docker containers.
- Load the built images → VMs in Glance and containers in local docker registry
- “cord load-service” command → one-time process that onboards the code

→ **VNF is now present in CORD and ready to USE**

Automated Configuration



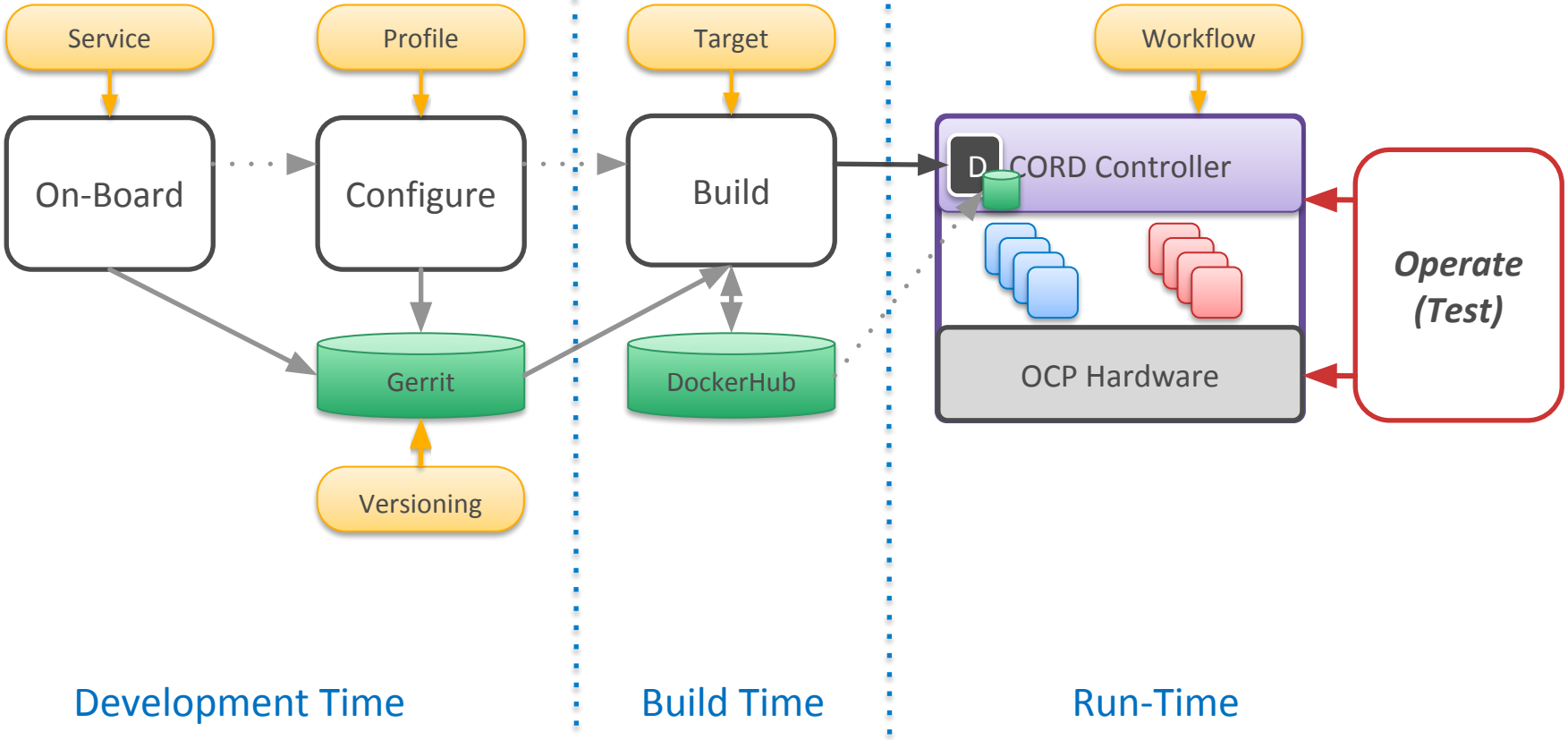
TOSCA Workflows
- Provision & Configure Services
- Runtime Operation

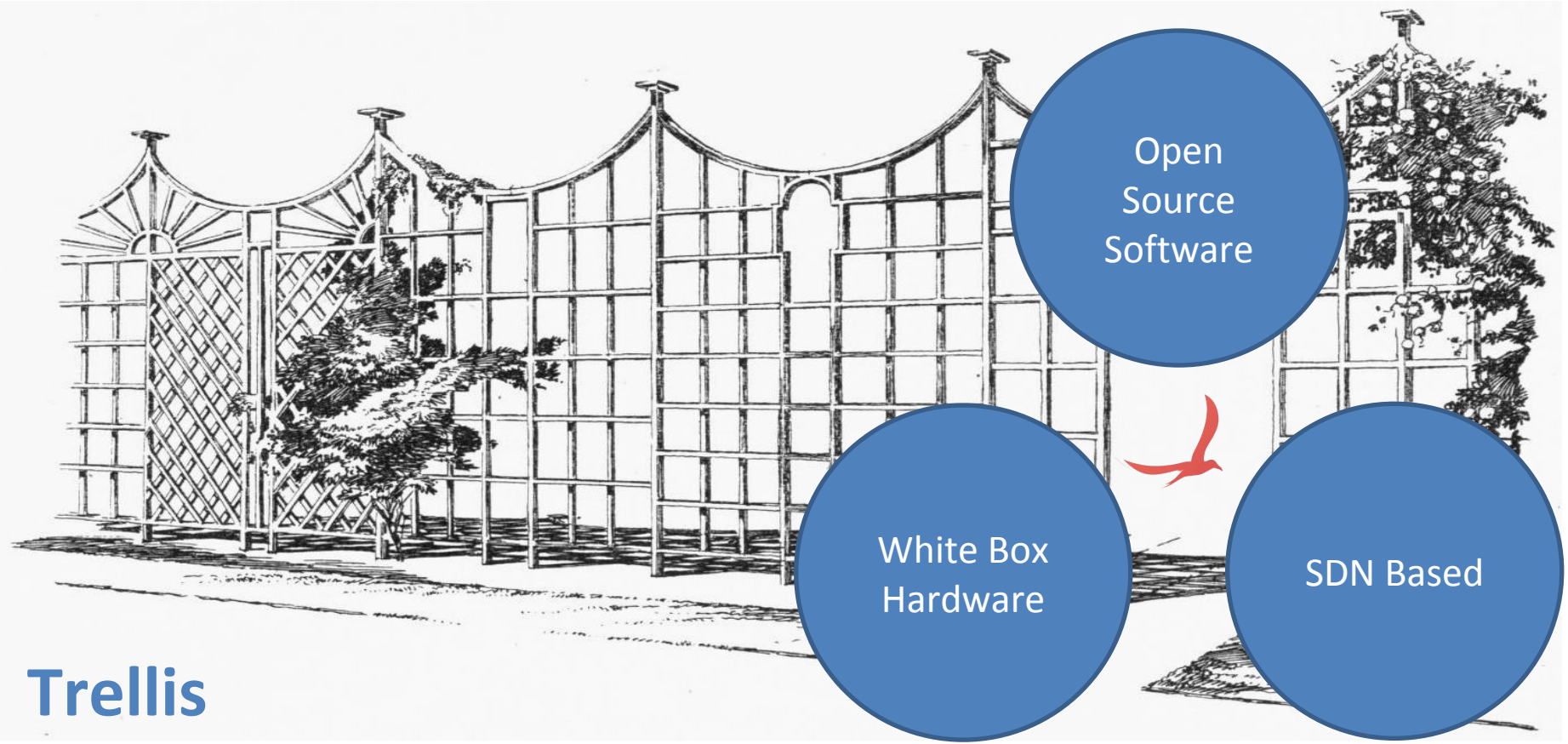
Protobuf-based Models
- Schema that Model Services
- Core set Loaded at Boot Time
- Dynamically Updated at Runtime

Helm Charts
- Containers that Implement Services
- Core set Loaded at Boot Time
- Dynamically Updated at Runtime

Kubernetes (Optionally MaaS)

Multi-Stage CI/CD Pipeline



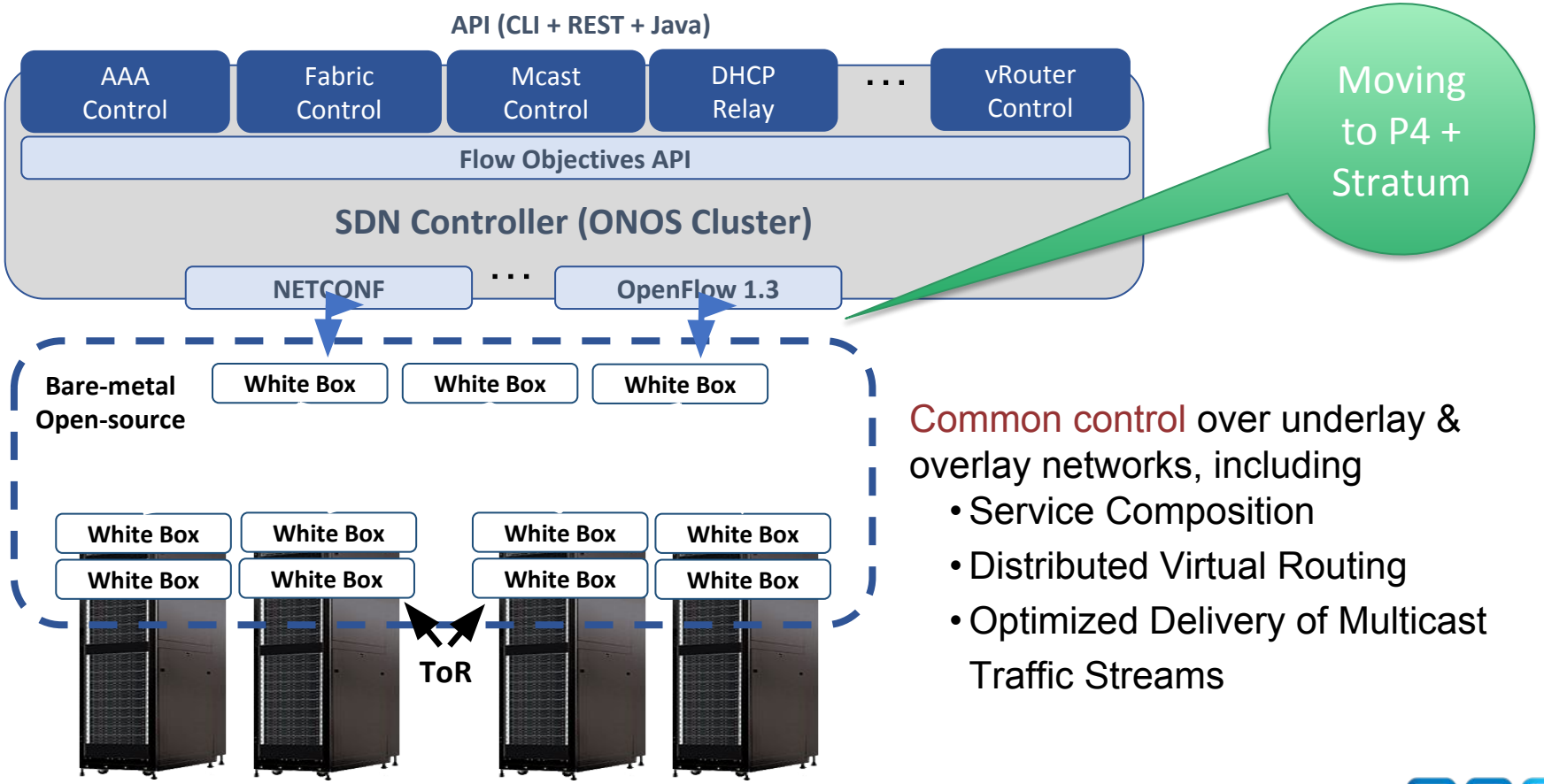


Trellis

Multi-purpose leaf-spine fabric designed for NFV

Trellis - Multi-Purpose Leaf-Spine Fabric

Trellis Fabric – Bare-metal + Open-Source + SDN



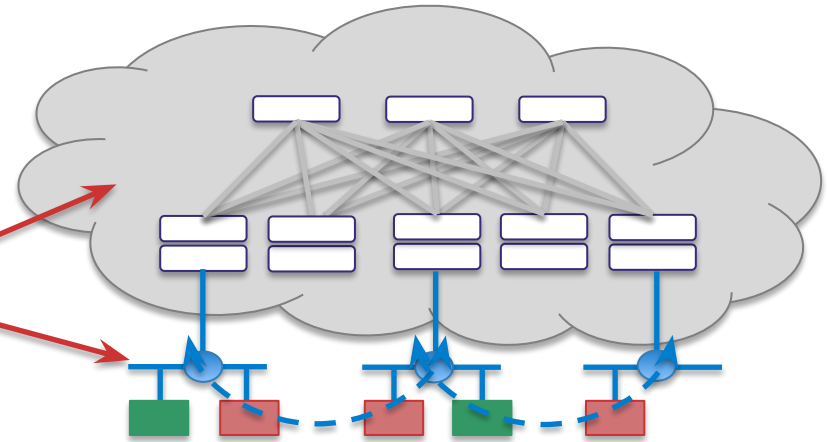
Common control over underlay & overlay networks, including

- Service Composition
- Distributed Virtual Routing
- Optimized Delivery of Multicast Traffic Streams

Trellis - CORD Network Infrastructure

Unified SDN Control
Of Underlay & Overlay

ONOS
Controller Cluster &
Apps



Datacenter Leaf-Spine
Fabric Underlay

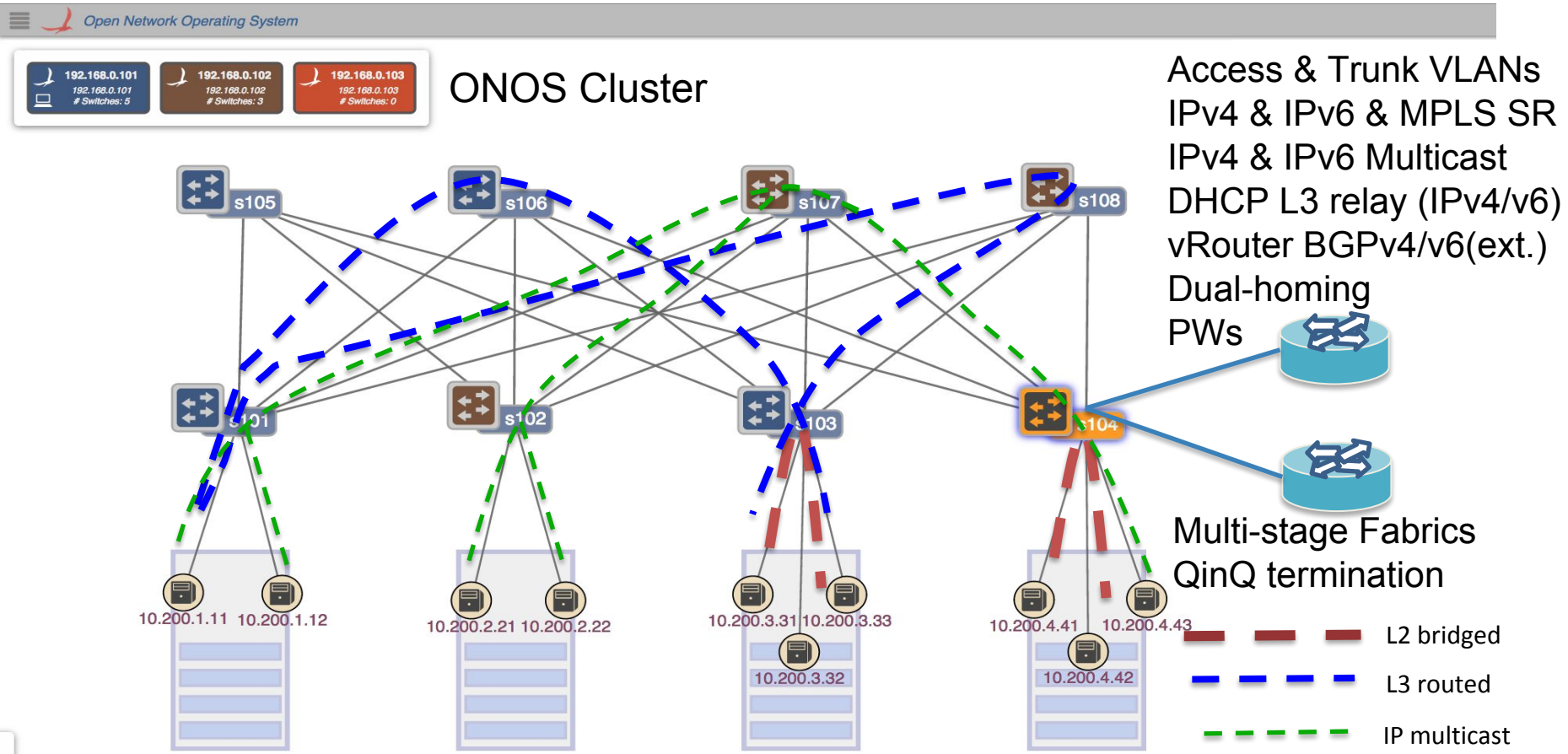
Virtual Network
Overlay

Trellis Provides Common control over underlay & overlay networks, including

- Service Composition for Tenant Networks
- Distributed Virtual Routing
- Optimized Delivery of Multicast Traffic Streams

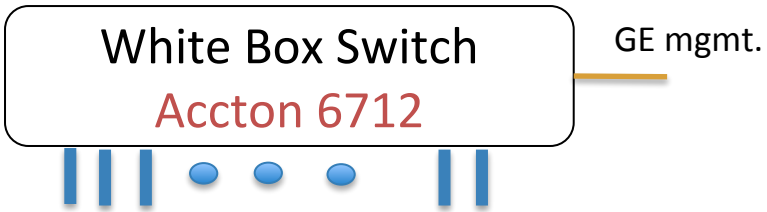
Trellis is the enabling Network Infrastructure for CORD

Trellis – Multi-purpose Leaf-Spine Fabric



Disaggregation – Bare-metal + Open-Source

Spine Switch

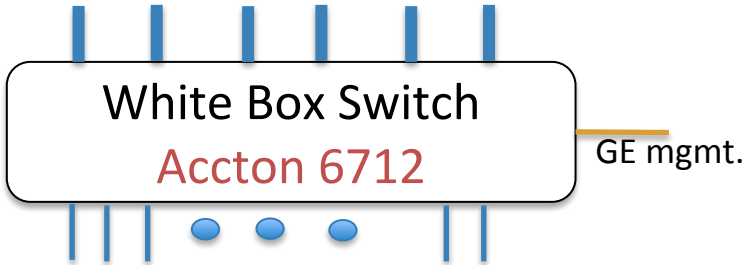


32 x 40G ports downlink to leaf switches

Leaf Switch

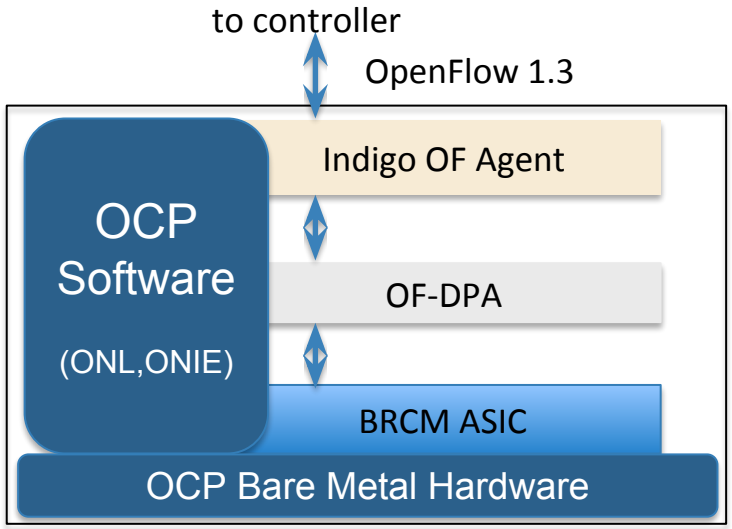
8 x 40G ports uplink to different spine switches

ECMP across all uplink ports



24 x 40G ports downlink to servers

Leaf/Spine Switch Software Stack



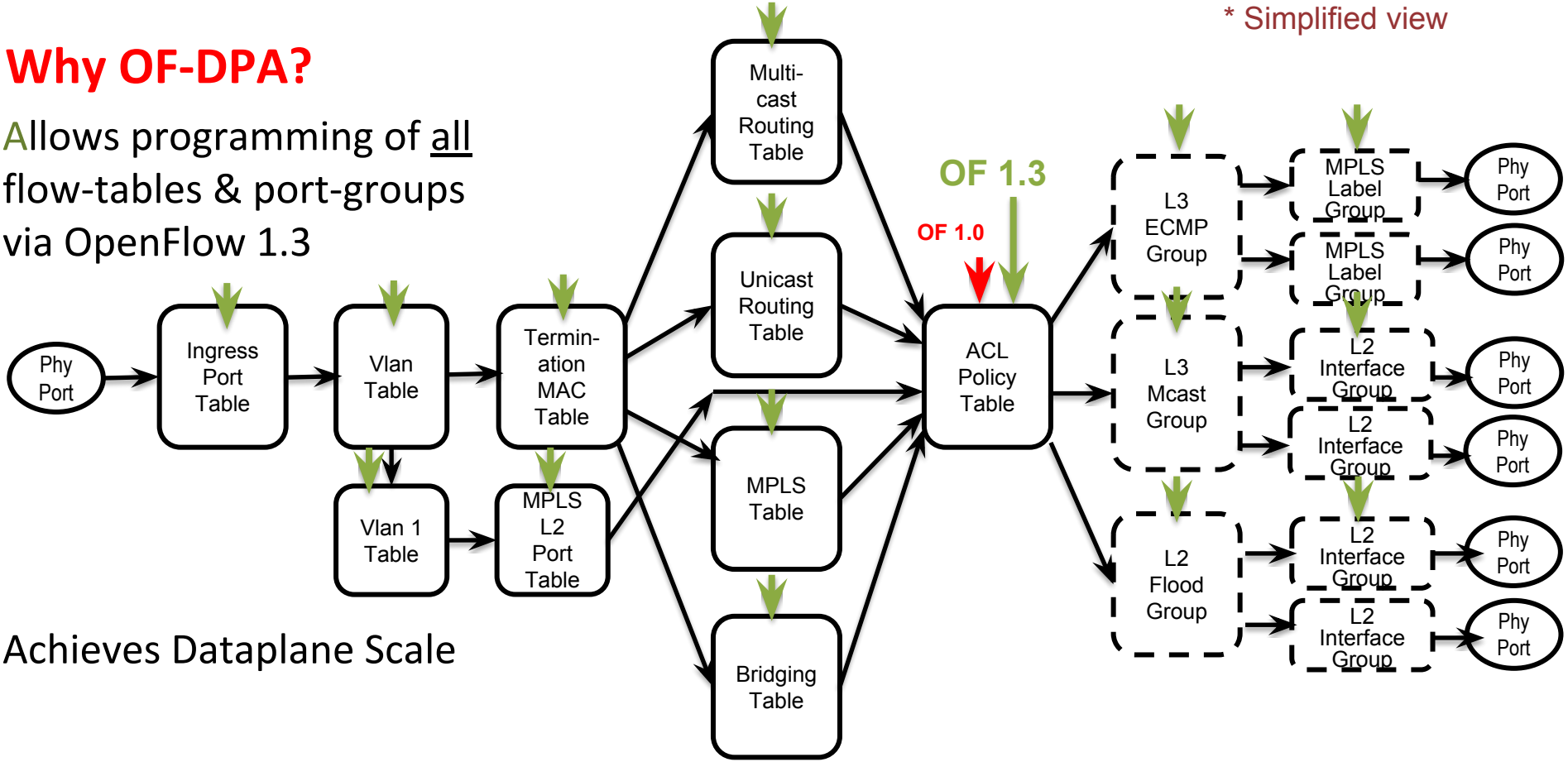
OCP: Open Compute Project
ONL: Open Network Linux
ONIE: Open Network Install Environment
BRCM: Broadcom Merchant Silicon ASICs
OF-DPA: OpenFlow Datapath Abstraction

Fabric ASIC Pipeline* (BRCM's OF-DPA)

* Simplified view

Why OF-DPA?

Allows programming of all flow-tables & port-groups via OpenFlow 1.3



Achieves Dataplane Scale

Trellis – Towards Production Readiness

- **Analyse issues** reported by Harmonic in their setups
- **Code review** patches they submit to ONOS
- Design discussions for new features
- Bringing up ONF pod housed in San Jose

- Bringing up ONF pod hosted by Flex
- QA collaboration

- **Report issues found; ~75 cases**
- Create automated tests to reproduce
- **Validate fixes** / releases
- Help with issues in their setup
- **Design discussions** for new features
- Daily scrum

- Deliver **features**; meet ops needs
- Deliver **stability**, scale, perf, tools
- 7 releases in last 4 months
- Support Comcast design/dev/QA teams – issue analysis; root-cause; **recommend best-practices/training**
- **Design discussions** for new features
- Daily scrum
- **Documentation**

- Report issues found in hw
- Validate T2 versions of switch software EdgeCore builds for us

- **Design discussions** for apps/features created by Nokia
- **Code review** submitted patches (40 changesets; 4-5 patchsets/change)
- Validate some features; report issues

Trellis – Towards Production Readiness

December ’17 – March ’18

Support Comcast

- **Support Comcast design/dev/QA/ops teams** – issue analysis; root-cause;
- **Recommend best-practices/ training**
- Design discussions for new features & architectural improvements
- **Daily scrum**
- Documentation

Support Other teams

- **Broadcom**
- **Nokia**
- **Harmonic**

Deliver New Features

- **Pseudo wires** for in-band control
- Routing in **H-Agg** based topologies
- **Multicast** improvements
- **Dual-homing** improvements
- Other smaller features
- **ISSU** architectural discussions/progress

ONOS Stability & Scale

- Focus on stability of ONOS distributed stores (**9 releases of Atomix in last 4 months**)
- Scale investigation ongoing

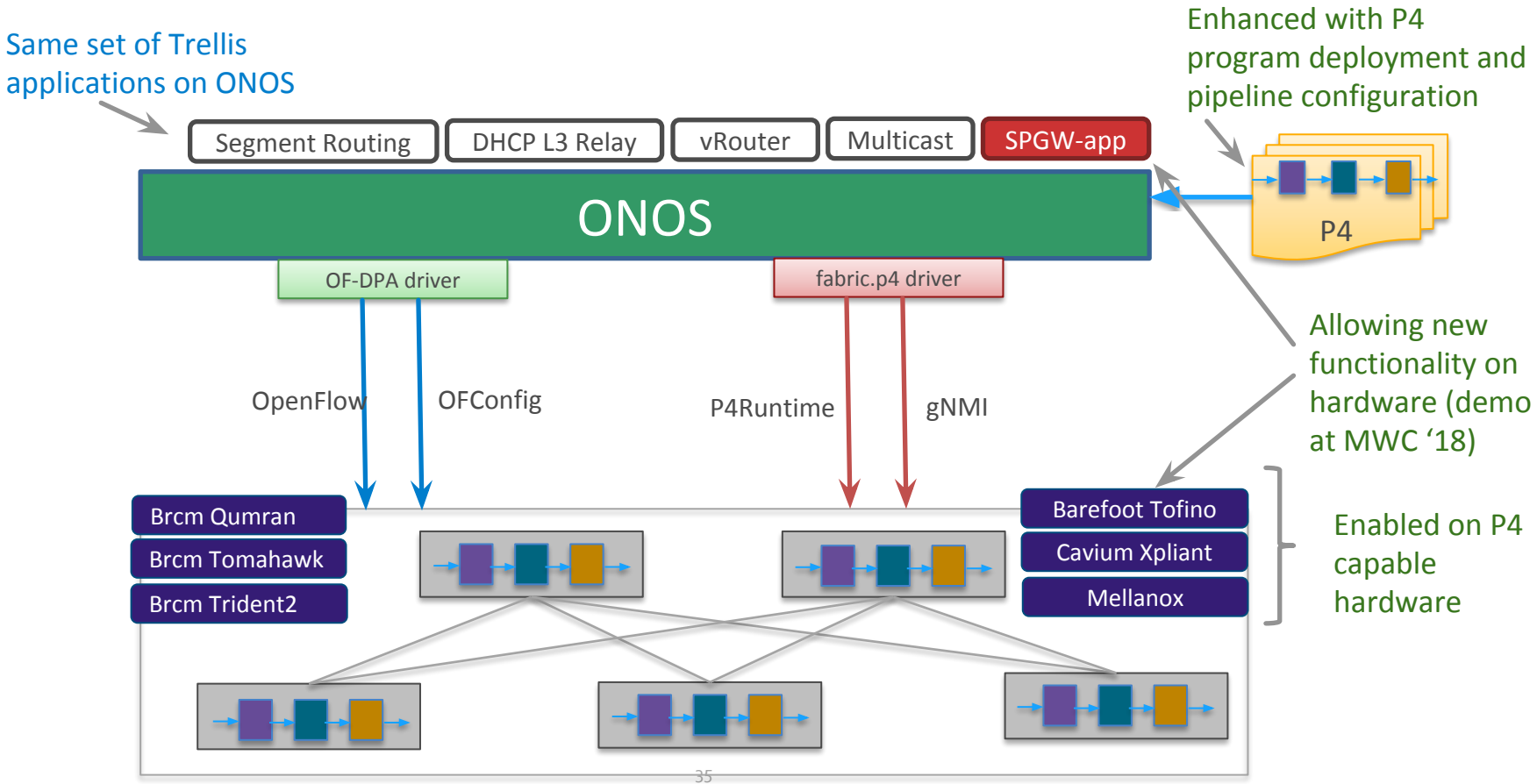
Tooling

- **T3** – Trellis Troubleshooting Tool
- **onos-diag**: Diagnostics collection tool

QA

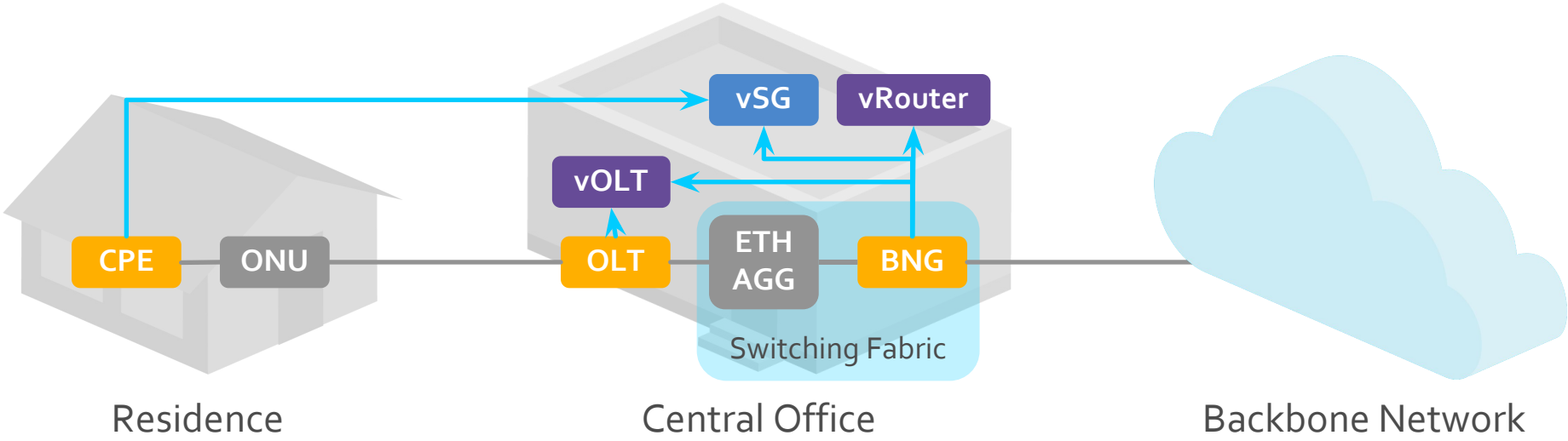
- Developing automated feature tests (**220 new tests in the last 4 months**)
- Extending framework for hardware based tests

Trellis & P4


Same set of Trellis applications on ONOS
Segment Routing
DHCP L3 Relay
vRouter
Multicast
SPGW-app
ONOS
OF-DPA driver
fabric.p4 driver
OpenFlow
OFConfig
P4Runtime
gNMI
Brcm Qumran
Brcm Tomahawk
Brcm Trident2
Barefoot Tofino
Cavium Xpliant
Mellanox
Enhanced with P4 program deployment and pipeline configuration
P4
Allowing new functionality on hardware (demo at MWC '18)
Enabled on P4 capable hardware

Residential CORD and VOLTHA

Disaggregation



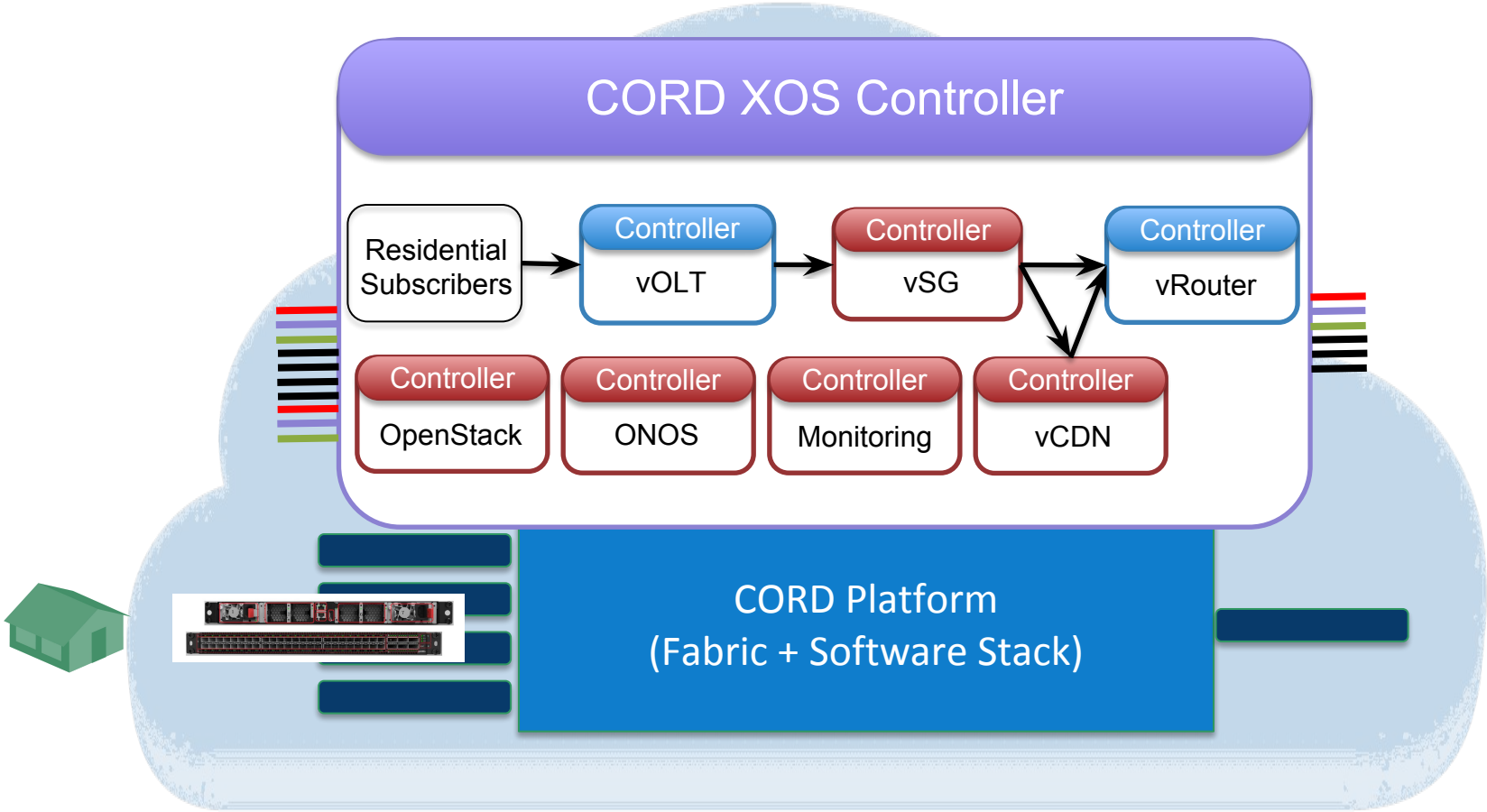
CPE: Customer Premises Equipment

ONU: Optical Network Unit

OLT: Optical Line Termination

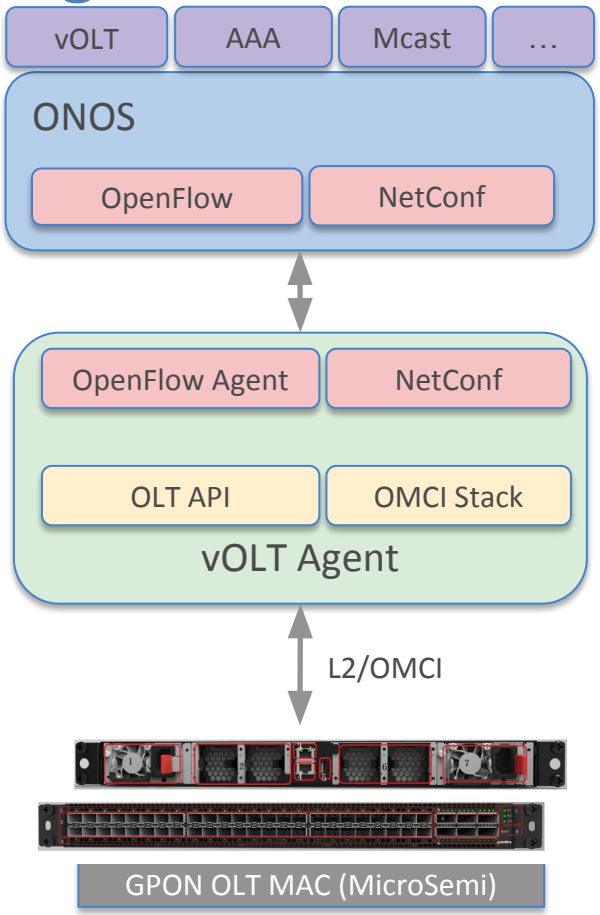
BNG: Broadband Network Gateway

Service Graph for R-CORD Use Case



Disaggregating the OLT with VOLTHA

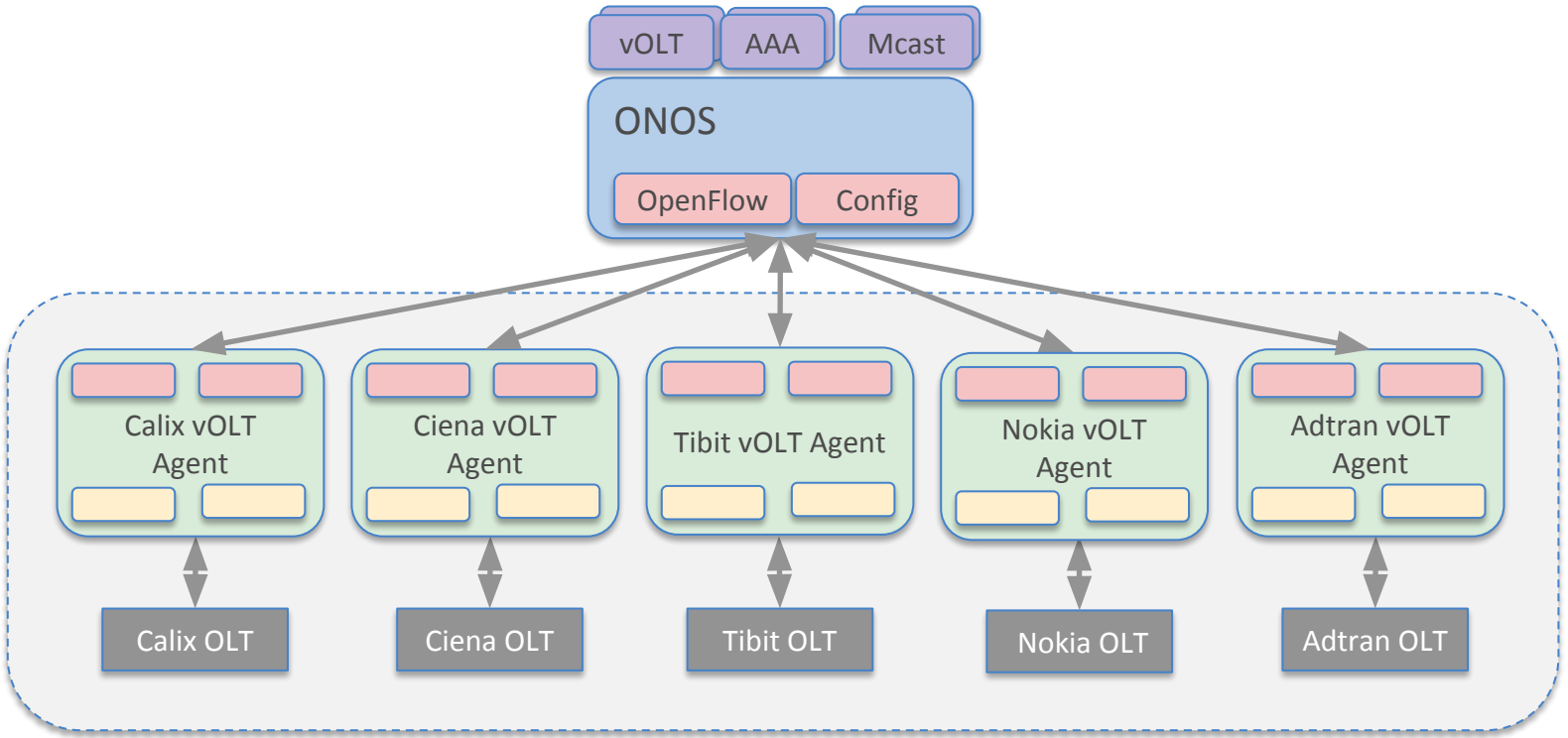
OLT Disaggregation



This is what we (the CORD community) accomplished as part of the R-CORD POCs

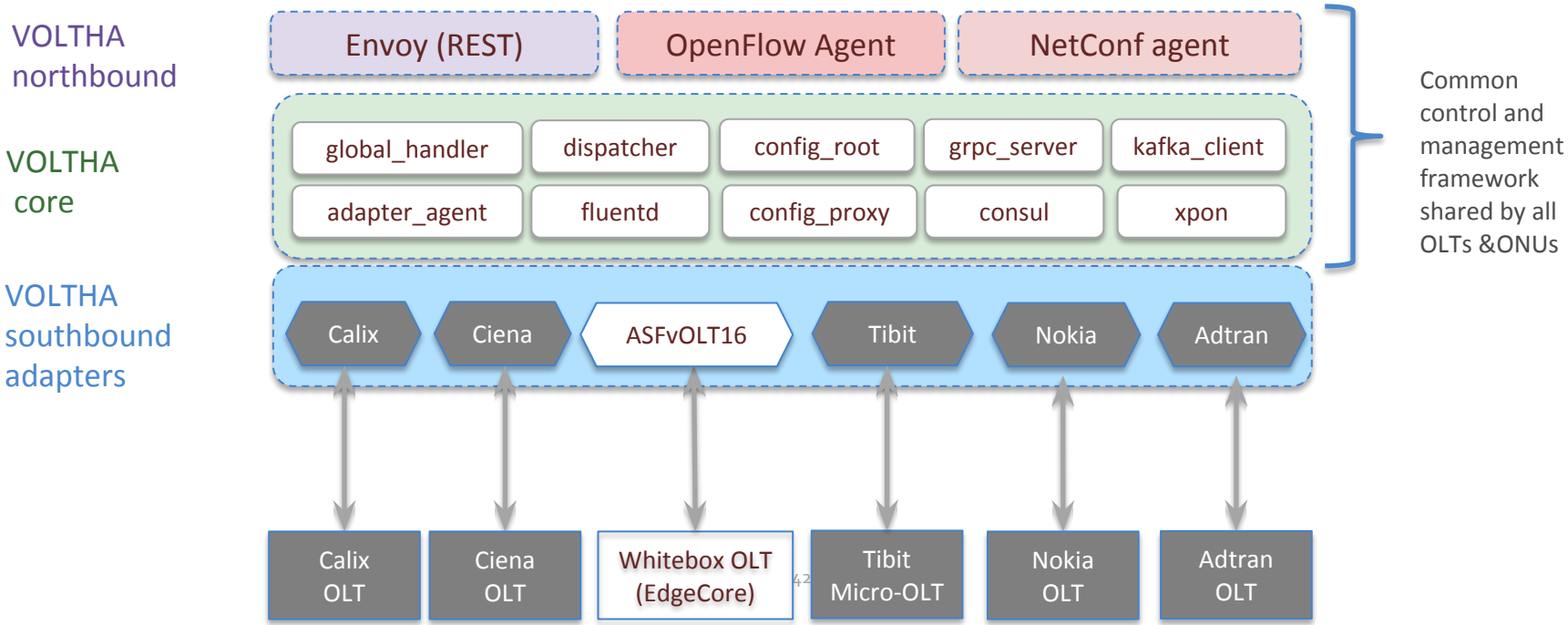
How to expand to support multiple vendors?

How do we expand support for this so that many vendors can participate and not have to rebuild the same vOLT agent stack while providing some abstraction to the control and management planes?

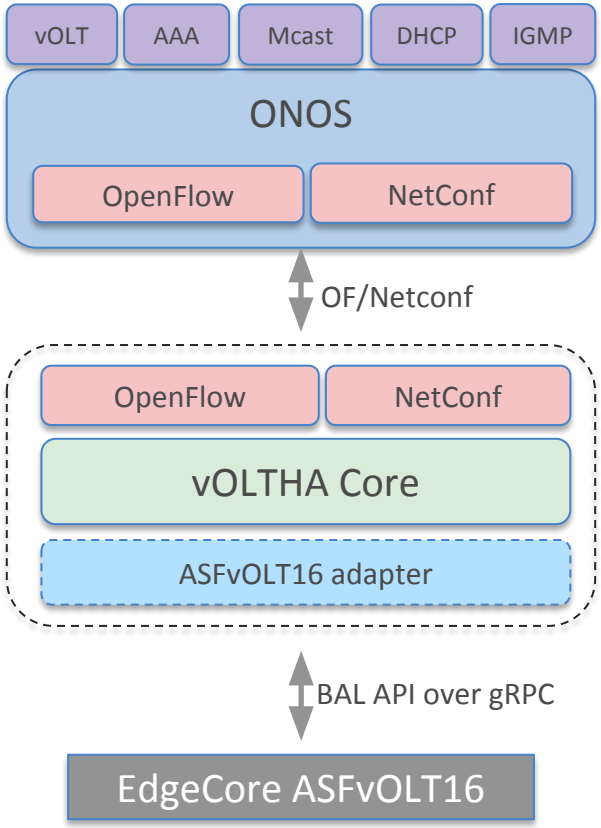


Virtual OLT Hardware Abstraction (VOLTHA)

VOLTHA hides PON-level details (T-CONT, GEM ports, OMCI etc.) from the SDN controller, and abstracts each PON as a pseudo-Ethernet switch easily programmed by the SDN controller



Virtual OLT with VOLTHA



- Legacy control plane functions run as control apps on ONOS
- VLAN provisioning, multicast, IGMP snooping/proxy, AAA (802.1X, RADIUS) , DHCP relay

- VOLTHA handles PON specifics and abstracts different HW
- ASFvOLT16 adapter uses BAL API to program device

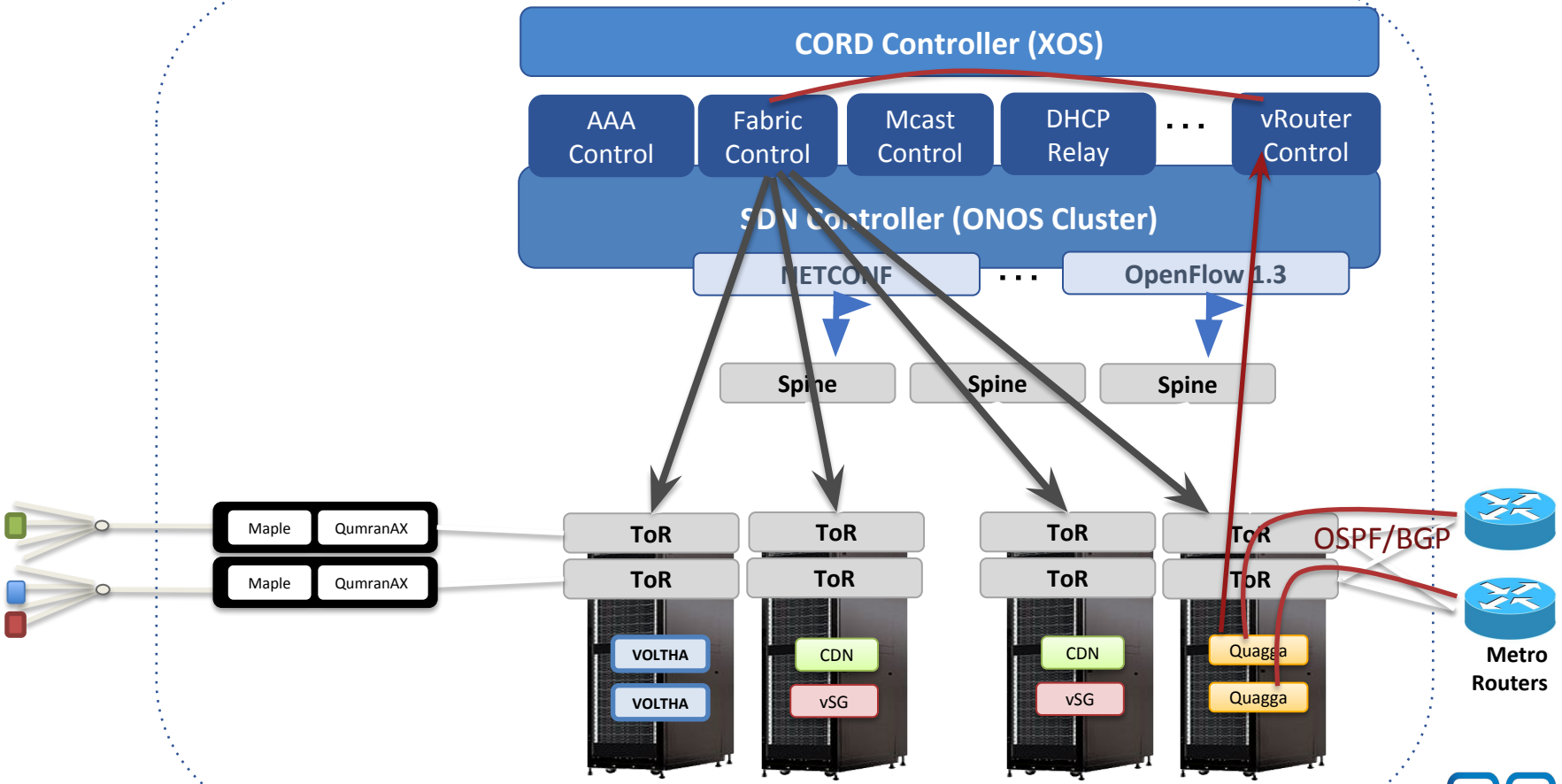
- Whitebox open HW (EdgeCore ASFvOLT16)
- 16 10G XGS-PON ports based on BRCM Maple chip
- 1x 100GBE Qumran AX switching chip

Disaggregating the BNG with vSG and vRouter

Virtual Subscriber Gateway (vSG)

- Subsumes per-subscriber functionality from CPE and BNG
- A separate vSG instance is create for each subscriber
  - providing services such as parental control, uplink bandwidth control, suspending services to the internet
- Runs as a container VNF
  - Reference implementation is a simple Docker container running DHCP, DNS, NAT on behalf of each subscriber

Upstream Connectivity with vRouter



VOLTHA roadmap

VOLTHA Roadmap - 1.3 Release (April 30 2018)

- VOLTHA High Availability
  - Migrating from Docker Swarm to **Kubernetes**
  - Explore other database redundancy framework
- Supports AT&T OpenOMCI Specification
  - Interoperability of ONUs and OLTs
- Release public docker images so can run without building code

VOLTHA – current state

- PON is abstracted as OpenFlow device that allows SDN controller to program service flows
- However, underneath configuring the PON (tconts, GEM ports, etc) relies heavily on top-down configuration
  - Several config commands/calls to bring up an subscriber's ONU in the simplest configuration
  - Breaks the simple management abstraction, exposes PON details

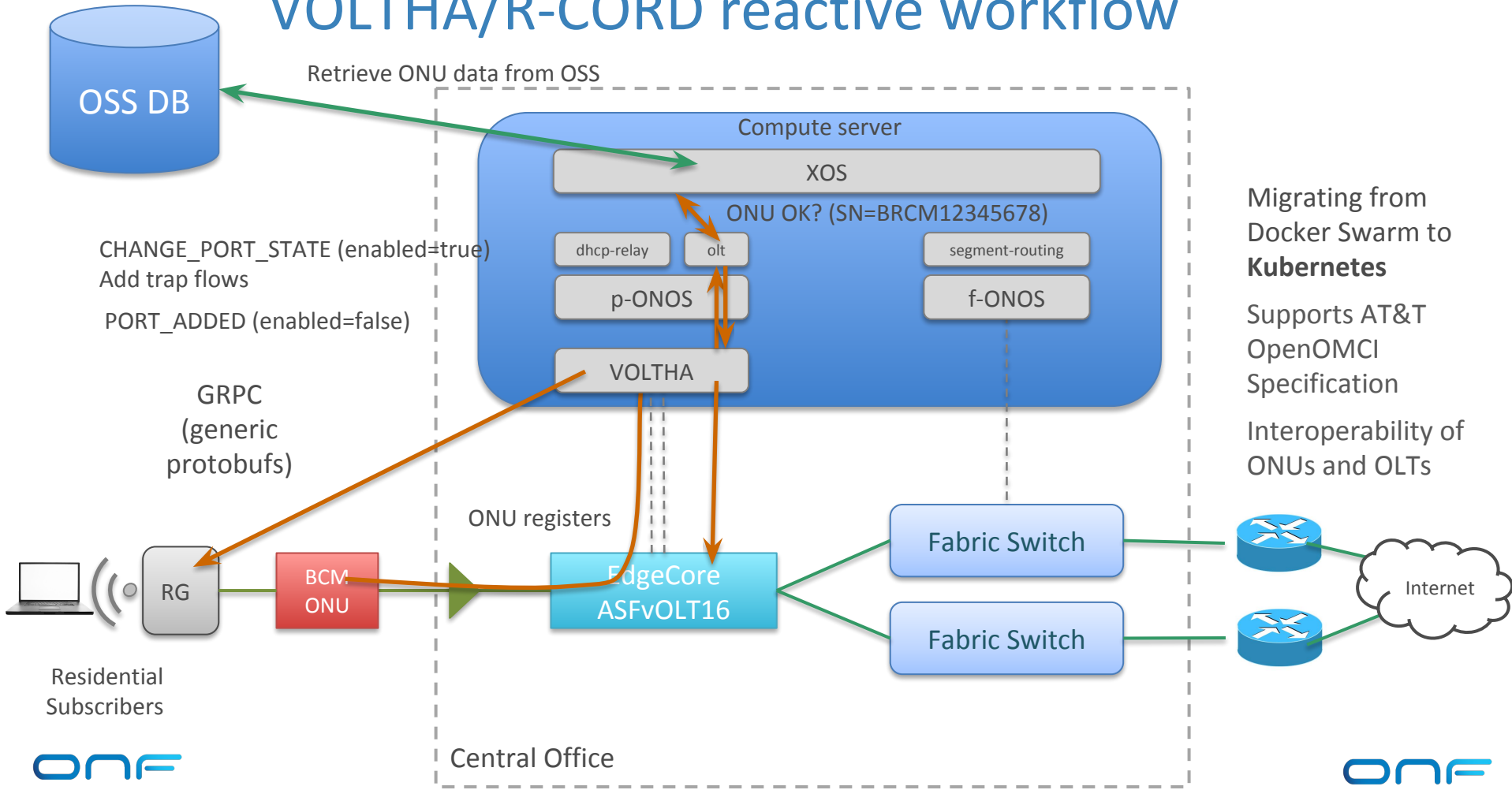
```
channel_group create -n "Manhattan" -d "Channel Group for Manhattan" -a up -p 100 -s 000000 -r
raman_none
channel_partition create -n "WTC" -d "Channel Partition for World Trade Center in Manhattan" -a up -r
20 -o 0 -f false -m false -u serial_number -c "Manhattan"
channel_pair create -n "PON port" -d "Channel Pair for Freedom Tower in WTC" -a up -r down_10_up_10 -t
channelpair -g "Manhattan" -p "WTC" -i 0 -o class_a
traffic_descriptor_profile create -n "TDP 1" -f 100000 -a 500000 -m 1000000 -p 1 -w 1 -e
additional_bw_eligibility_indicator_none
channel_termination create -i 0001bb590711de28 -n "PON port" -d "Channel Termination for Freedom
Tower" -a up -r "PON port" -c "AT&T WTC OLT"

vont_ani create -n "ATT Golden User" -d "ATT Golden User in Freedom Tower" -a up -p "WTC" -s
"BRCM12345678" -r "PON port" -o 1
ont_ani create -n "ATT Golden User" -d "ATT Golden User in Freedom Tower" -a up -u true -m false
tcont create -n "TCont 1" -r "ATT Golden User" -t "TDP 1"
v_enet create -n "Enet UNI 1" -d "Ethernet port - 1" -a up -r "ATT Golden User"
gem_port create -n "Gemport 1" -r "Enet UNI 1" -c 2 -a true -t "TCont 1"
```

VOLTHA 2.0 and beyond – Towards SDN

- Reduce dependency on top-down configuration (xPON)
  - Automatically bring up PON ports on device boot
  - Automatically detect and configure ONUs on registration
    - Allows for validation of ONUs with OSS
  - ‘Service Profile’ mechanism allows configuration of QoS parameters
- Separation of VOLTHA and Adapters into separate repos to enable independent releases
- New OpenOLT adapter and OLT software for whitebox OLTs

VOLTHA/R-CORD reactive workflow



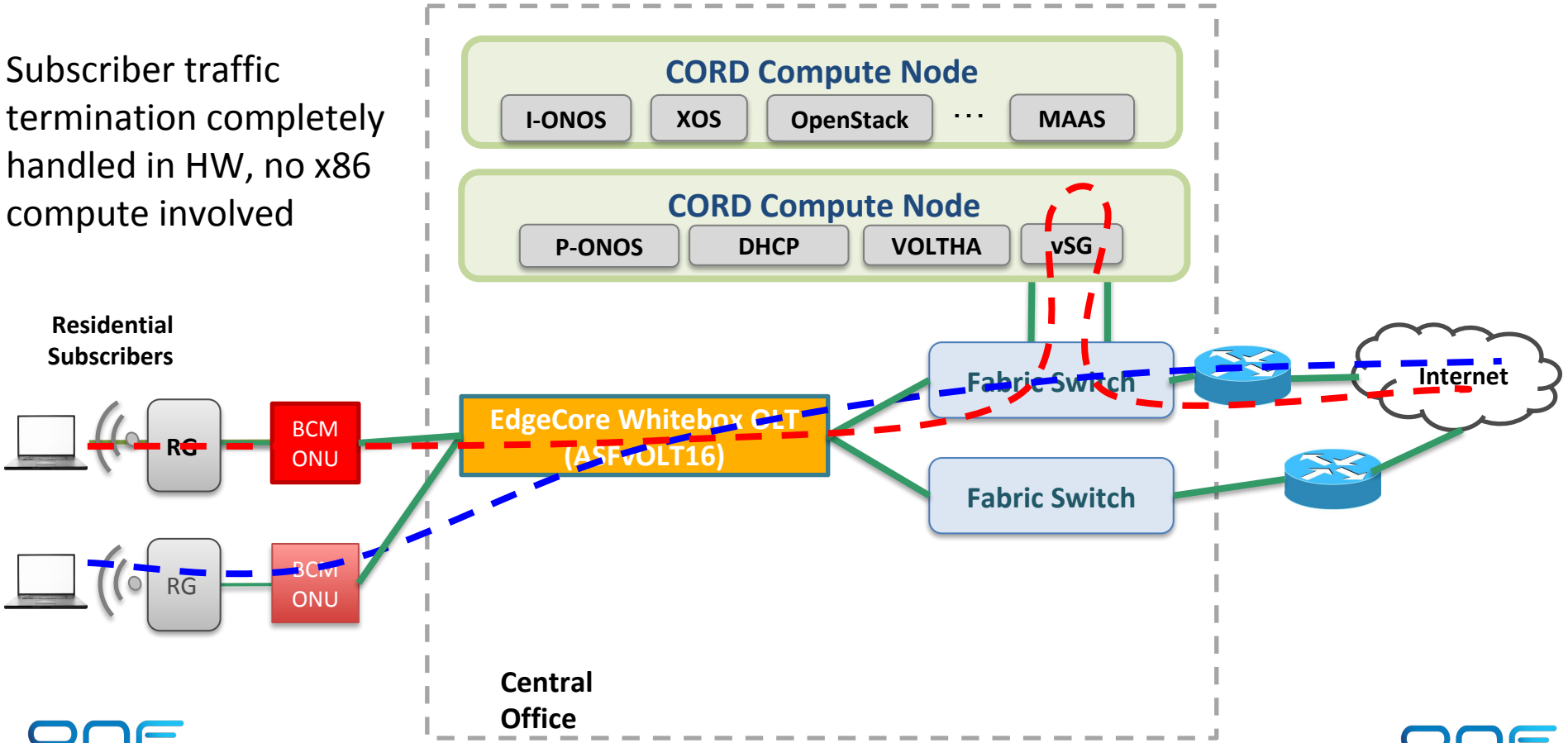
Migrating from Docker Swarm to **Kubernetes**

Supports AT&T OpenOMCI Specification

Interoperability of ONUs and OLTs

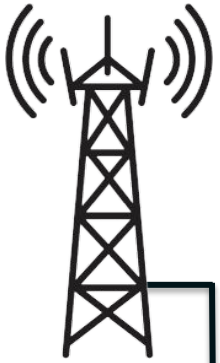
R-CORD Next: 'Fast-Path' Profile

Subscriber traffic termination completely handled in HW, no x86 compute involved



Mobile CORD

M-CORD AS THE **MOBILE EDGE**



Disaggregate RAN

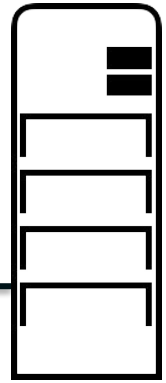
Virtualize some of its components

Realize virtualized components in the **cloud**

Disaggregate CORE

Virtualize all components

Realize virtualized components in the **cloud**



SDN'ize for programmability (xRAN)

Programmatically instantiate **RAN slices** for different use cases

XOS

ONOS | OPEN STACK / DOCKER

COMPUTE NODES

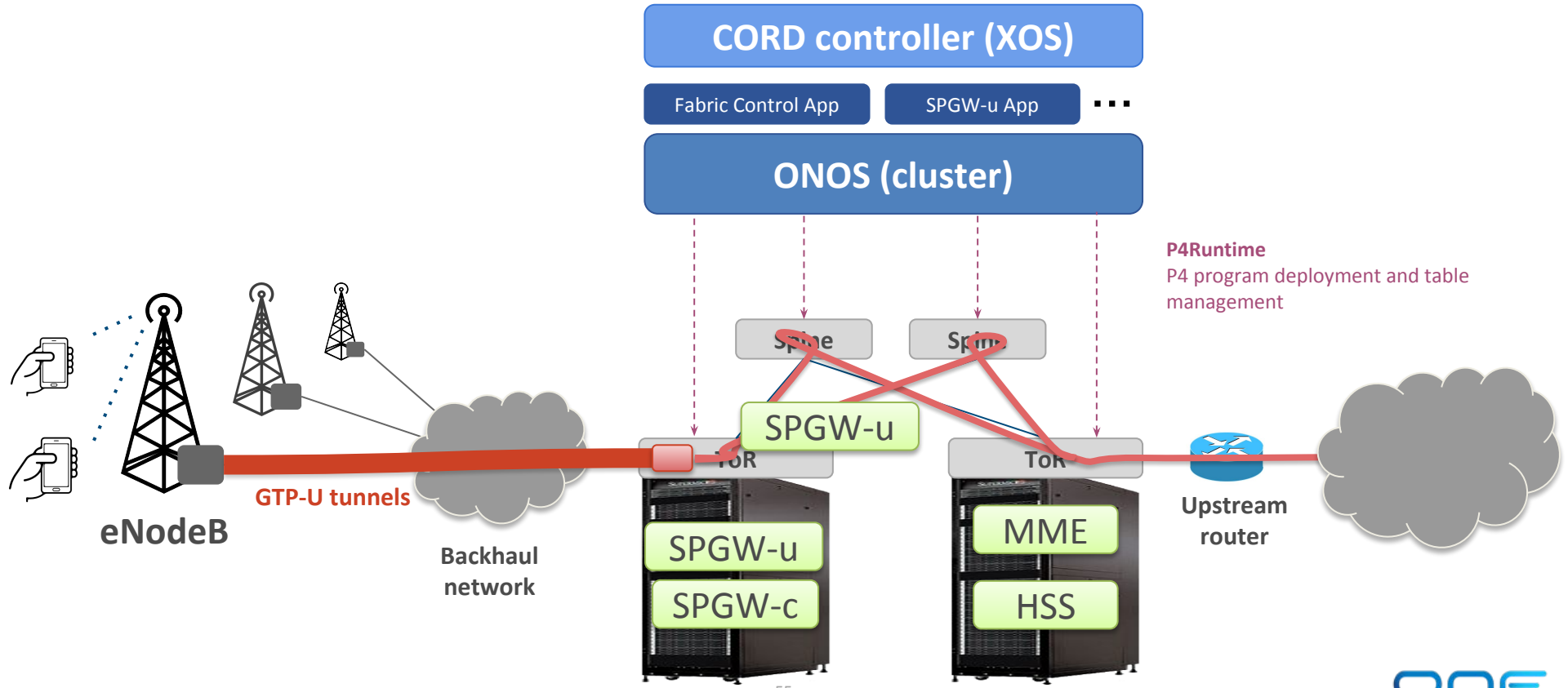
PROGRAMMABLE LEAF-SPINE FABRIC

MOBILE EDGE CLOUD

SDN'ize for user plane VNF offloading

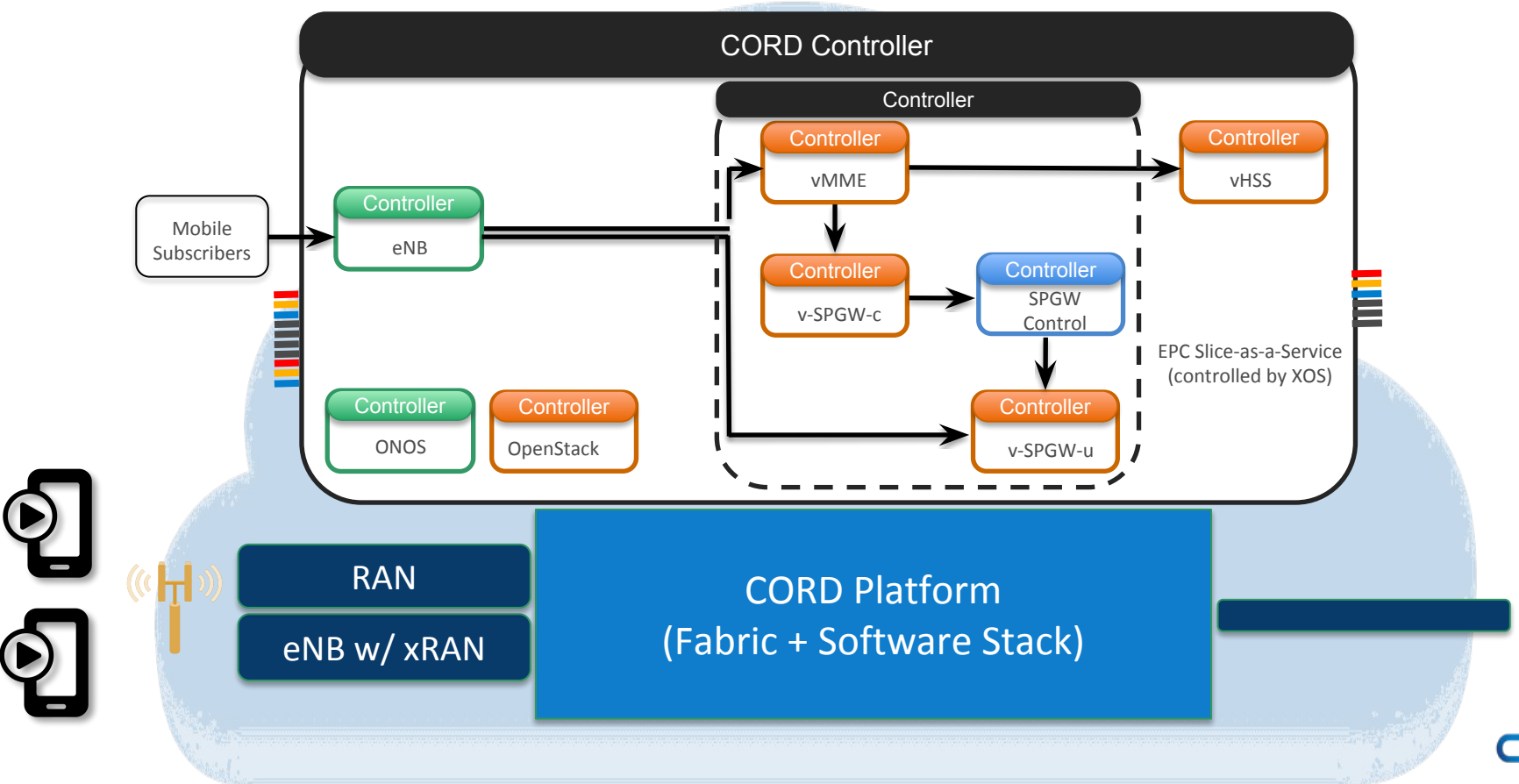
Programmatically instantiate **CORE slices** for different use cases

M-CORD Architecture



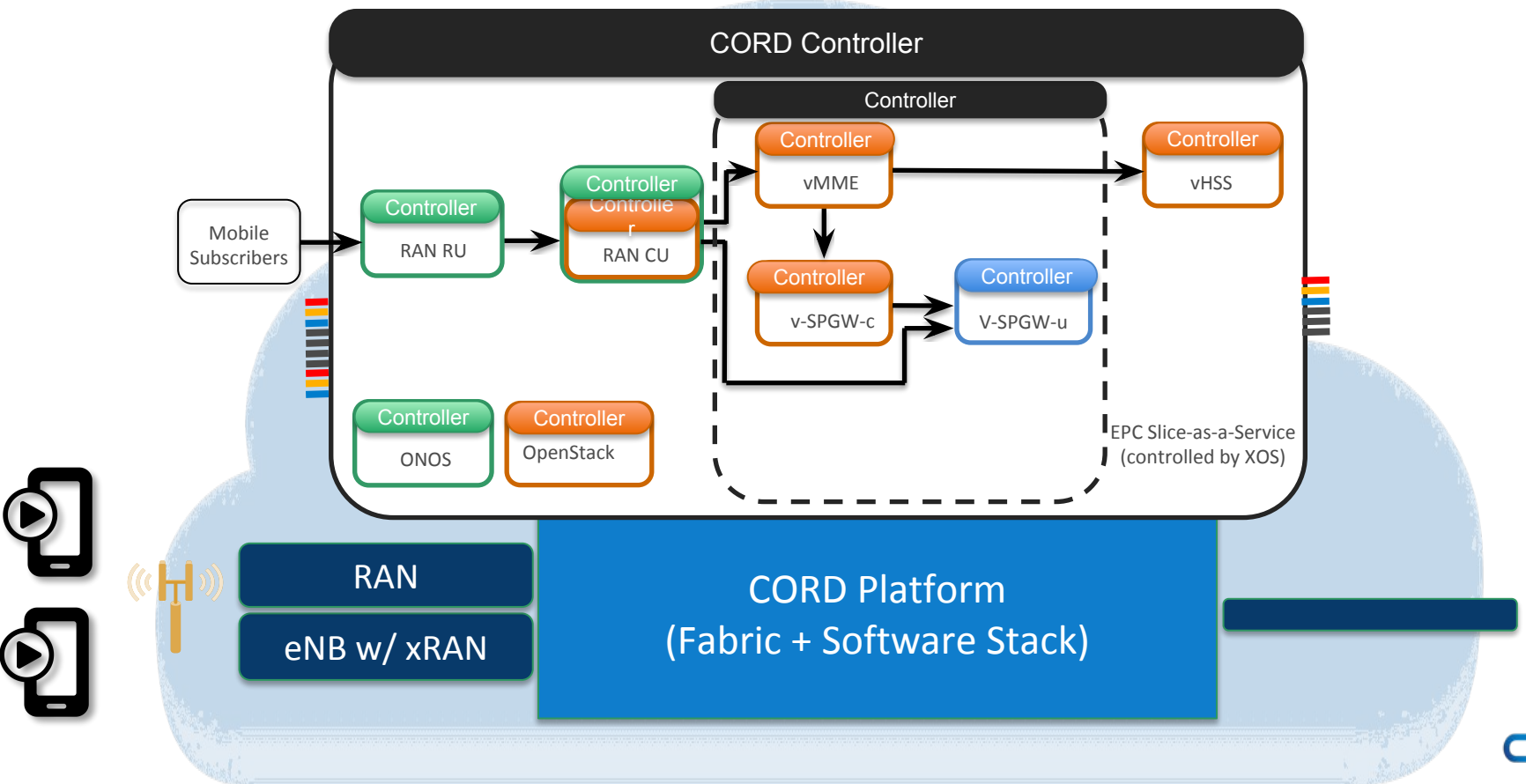
Service Graph for M-CORD

Programmable eNB and VNF-Based EPC User and Control Planes

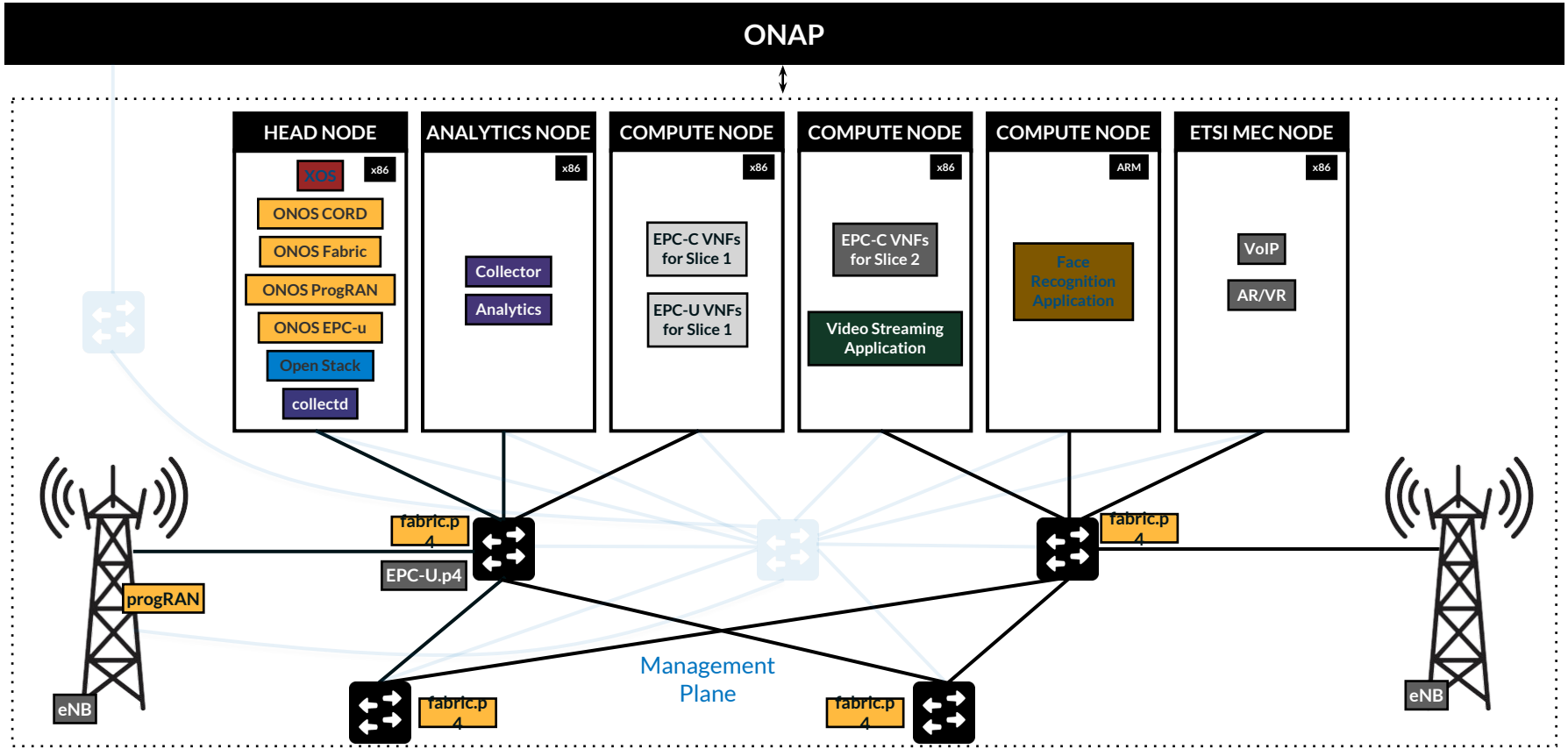


Service Graph for M-CORD

Programmable Split-RAN, P4-Based EPC User Plane and VNF-Based EPC Control Plane



M-CORD Implementation for MWC



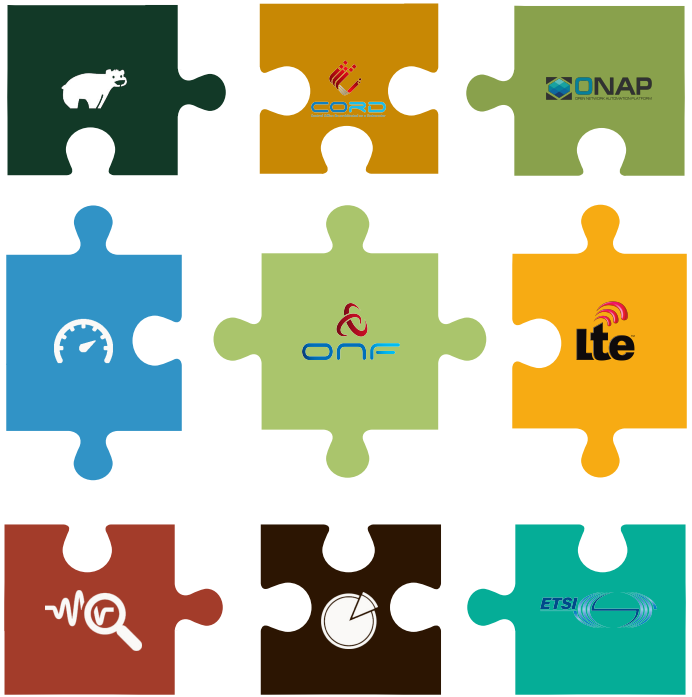
M-CORD Demos Shown at MWC

M-CORD 5.0
Architecture built on open source CORD 5.0

P4 Fabric
Utilizing a multi-vendor p4-programmable leaf-spine fabric

VNF Acceleration
Showcasing VNF acceleration by offloading to P4-fabric and GPUs

Closed Loop Analytics
Demonstrating closed loop analytics and automation



ONAP Integration
Demonstrating integration with ONAP for both design-time and run-time

Open Source EPC
Hosting an open source, virtualized, disaggregated EPC + services

E2E Network Slicing
Exhibiting ONOS-based programmable end-to-end network slicing

ETSI MEC Interoperability
Demonstrating interoperability with legacy ETSI MEC platform

Comprehensive, Highly Integrated Demo
Highlighting the Power of Open Source

M-CORD Futures

Mobility presents new challenges and requirements for the CORD architecture

As mobile devices move, due to low latency requirement, processing, and services need to be moved to the edge cloud.

This requires that state be shared across CORD pods.

CORD Next Steps

CORD 6.0 Release Planned June 2018

Platform

- Integration of Kubernetes and Helm
- Progress towards componentization - clean division between build and deploy
- Move to updated version of OpenStack (Newton or Pike), Ubuntu (16.04), and MaaS (2.3)
- Integrate support dual homing of servers and access devices
- Refactored CI/CD to take advantage of dynamic service loading

Use Cases

- R-CORD: Automated VOLTHA integration with R-CORD
- M-CORD: Integrated MWC/ONS Demos into release: eNB, MME, HSS
- Trellis: Preliminary integration of a P4 fabric

Kubernetes: Infrastructure, Platform and Deployment

- Deploy CORD over Kubernetes cluster

- Clear separation of Infrastructure deployment and Platform deployment:
  - Infrastructure: Fabric provisioning, Node OS, Ansible, MaaS, OpenStack, Kubernetes/ISTIO
  - Platform: CORD/XOS, Service Graphs

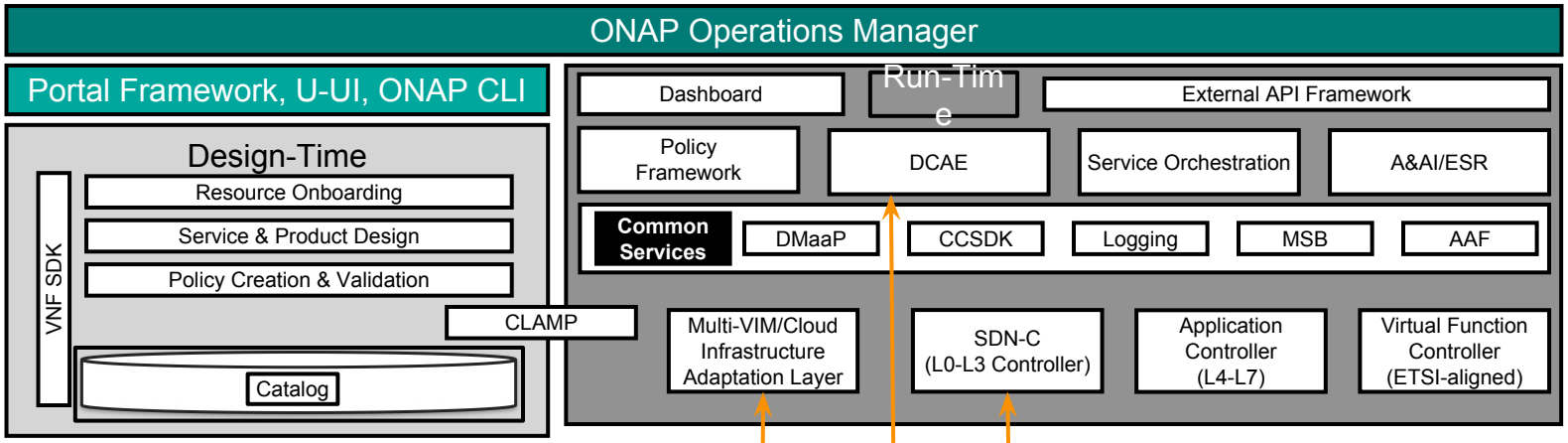
- Kubernetes Deployment based on Kubespray
  - Opensource, Ansible-based
  - Supports multiple infrastructure profiles, like VMs, bare metal and multiple OSs.

kubernetes

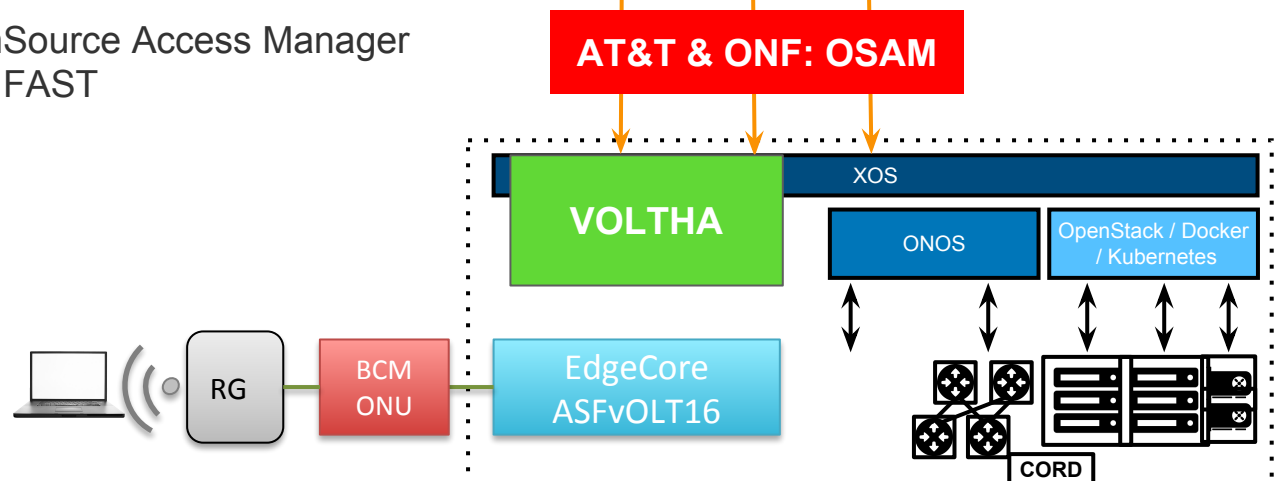
Application packaging using HELM

- Use Helm Charts for specifying XOS Core, platform components (VOLTHA) and service graph (R-CORD, M-CORD etc) packages
  - Directory with all kubernetes resources in it
  - Open Source packaging framework
  - Declarative intent specification in YAML
  - Allows application versioning, rolling updates, rollbacks

CORD-ONAP Integration



OSAM: OpenSource Access Manager
xPON and G.FAST



Useful Links

- CORD website - http://opencord.org
- Tutorials, documents, and others
  - https://wiki.opencord.org and https://guide.opencord.org
- CORD github/gerrit
  - https://github.com/opencord and https://gerrit.opencord.org
- VOLTHA - https://wiki.opencord.org/display/CORD/VOLTHA
- OSAM
  - https://wiki.onap.org/display/DW/OpenSource+Access+Manager