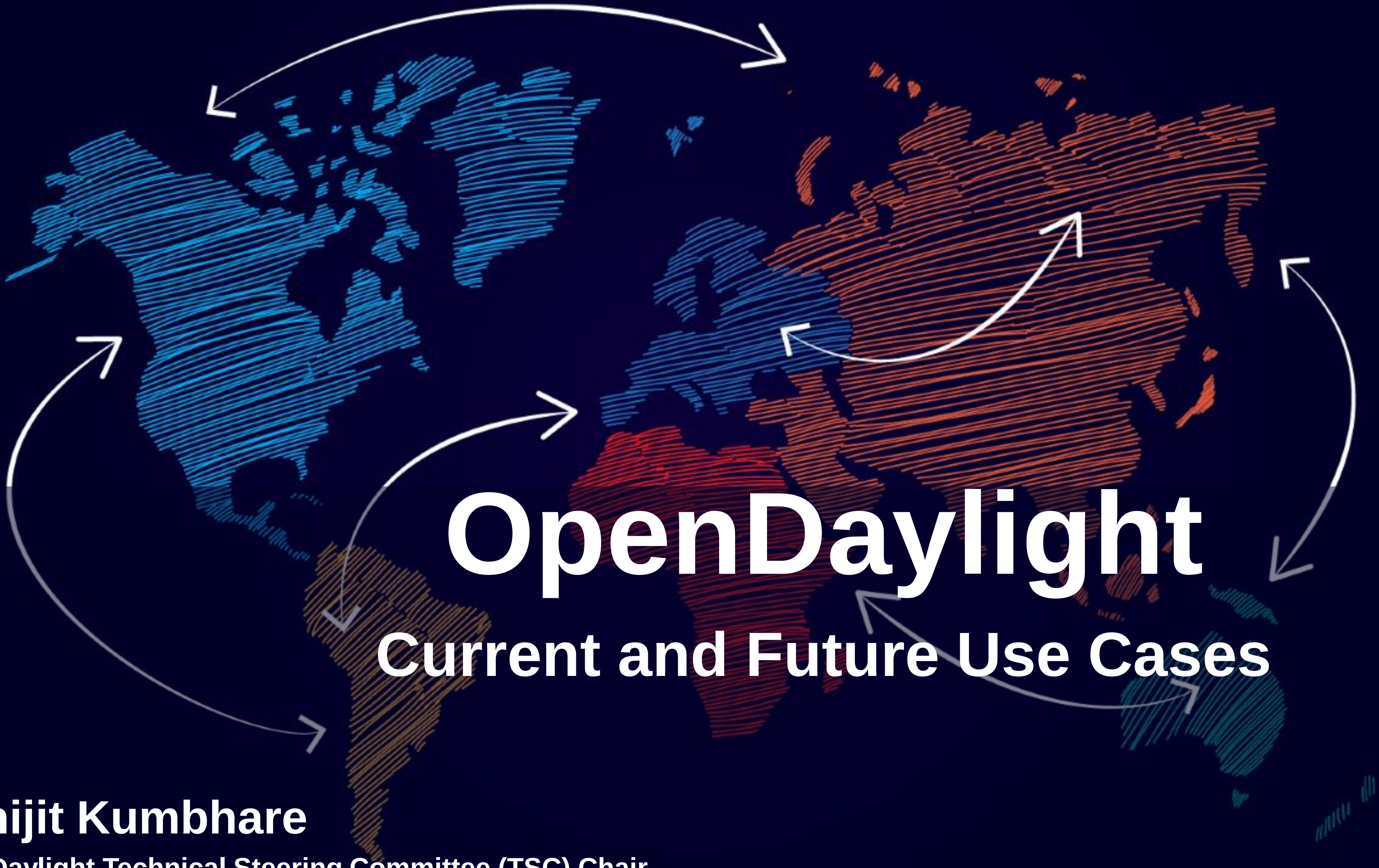
OPEN SOURCE NETWORKING DAYS



OpenDaylight

Current and Future Use Cases

Abhijit Kumbhare

OpenDaylight Technical Steering Committee (TSC) Chair

Principal Architect / System Manager, Ericsson

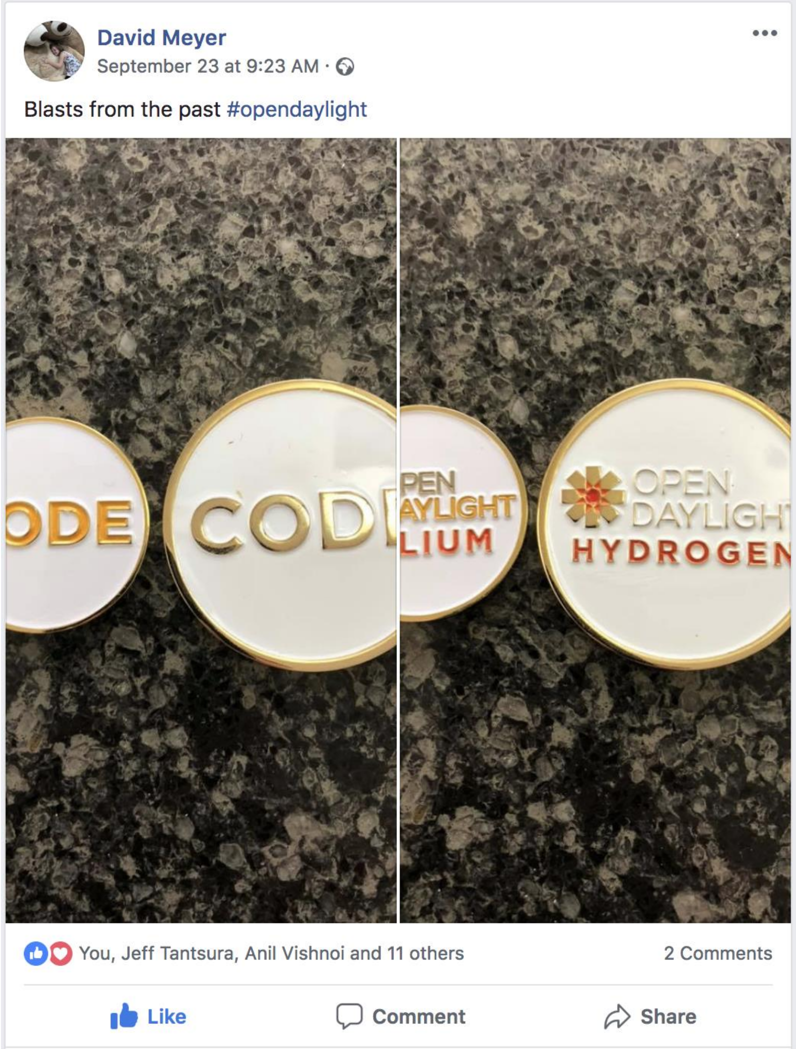
Agenda

- OpenDaylight Overview and Architecture
- OpenDaylight Use Cases (Partial List)
    - I. Network Abstraction
    - II. ONAP
    - III. AI/ML with OpenDaylight
    - IV. Network Virtualization
    - V. ODL in OSS
- OpenDaylight: Getting Involved
- Acknowledgements
- Q & A

OpenDaylight Overview and Architecture

A month ago …

- Dinner Discussion with Phil Robb, VP of Operations, Networking & orchestration, Linux Foundation at the ONS Europe
  - Topic: our first OpenDaylight Meetings
    - November 2012



David Meyer

September 23 at 9:23 AM ·

Blasts from the past #opendaylight

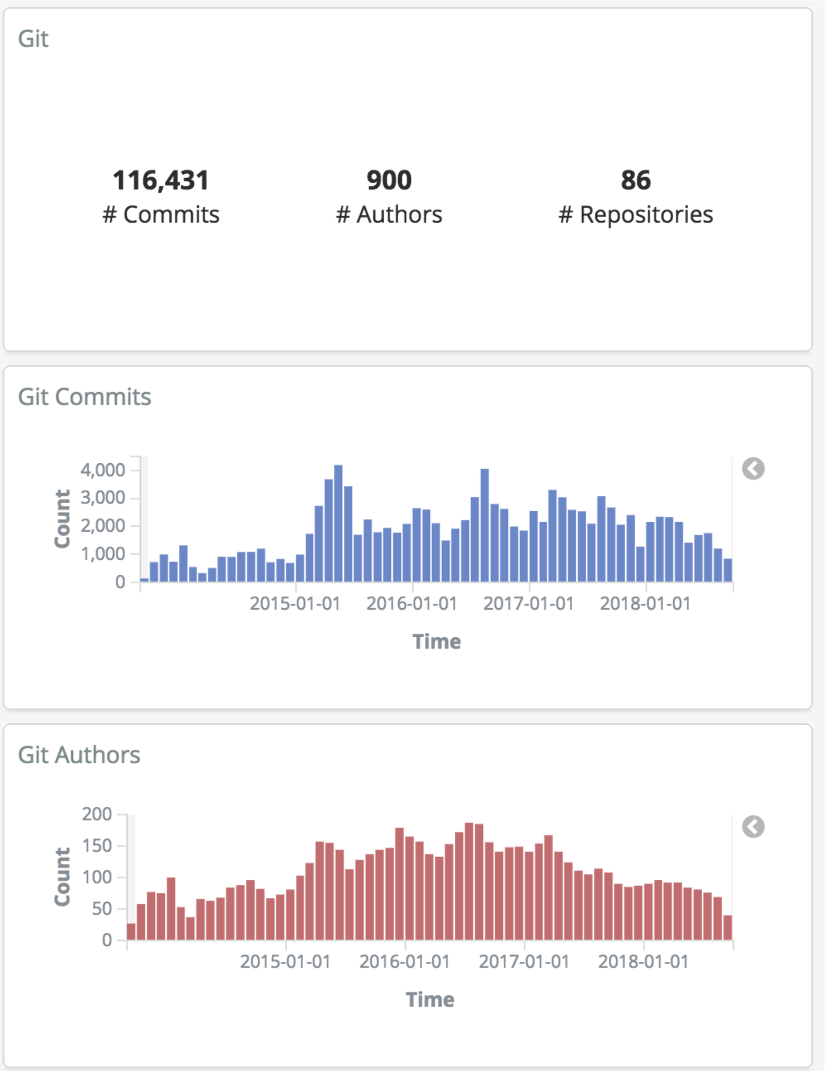
Nostalgic post by Dave Meyer, first ODL TSC chair on Facebook about first release Hydrogen in Jan 2014

OpenDaylight Project Goals

- **Code:** To create a robust, extensible, open source code base that covers the major common components required to build an SDN solution and create a solid foundation for Network Functions Virtualization (NFV)
- **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 and around.

OpenDaylight Now

- Mature, Open Governance
- 900 Contributors
- Over 100 deployments
- Multiple use cases
- Dozens of ODL-based solutions
- Mature code base – continued robust contributions even after 5+ years
- Focus on performance, scale and extensibility



Git

116,431	**900**	**86**
# Commits	# Authors	# Repositories

Git Commits

Git Authors

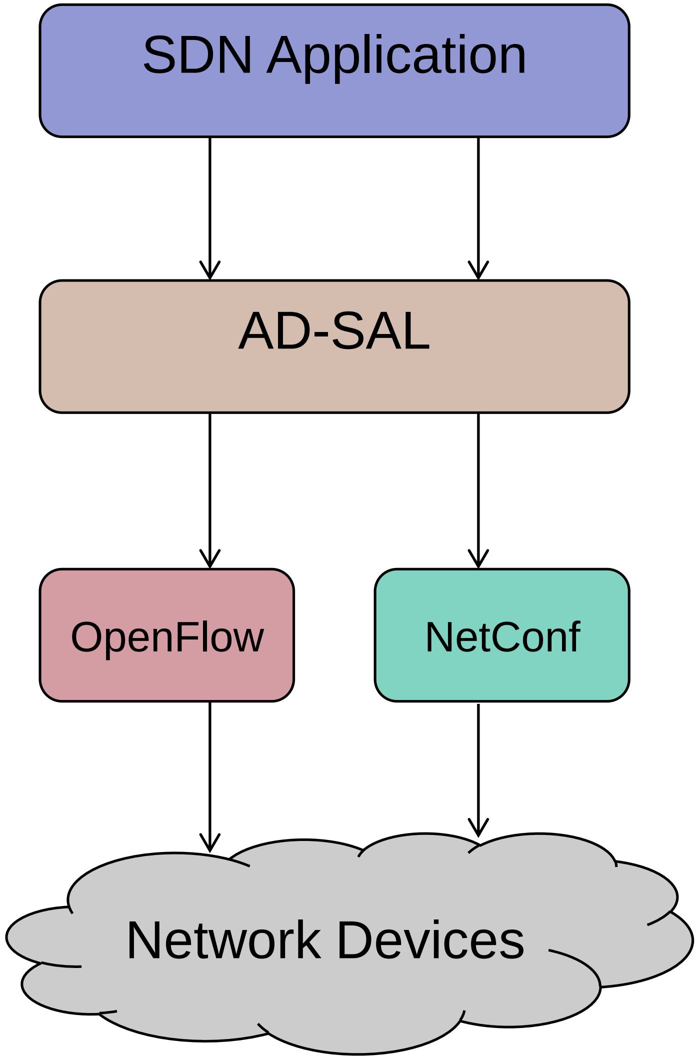
https://opendaylight.biterg.io/

Service Abstraction Layer

- Initial SDN controllers
  - Controller application APIs strongly tied to OpenFlow
  - Hence applications developed limited to a single southbound protocol
- OpenDaylight Goal
  - Decouple the application API from the southbound protocol plugins - be that Openflow, NETCONF, OVSDB, PCEP, BGP, SNMP, or whatever.
- How to achieve the goal?
  - Use an abstraction layer – or what is called by OpenDaylight as Service Abstraction Layer or SAL

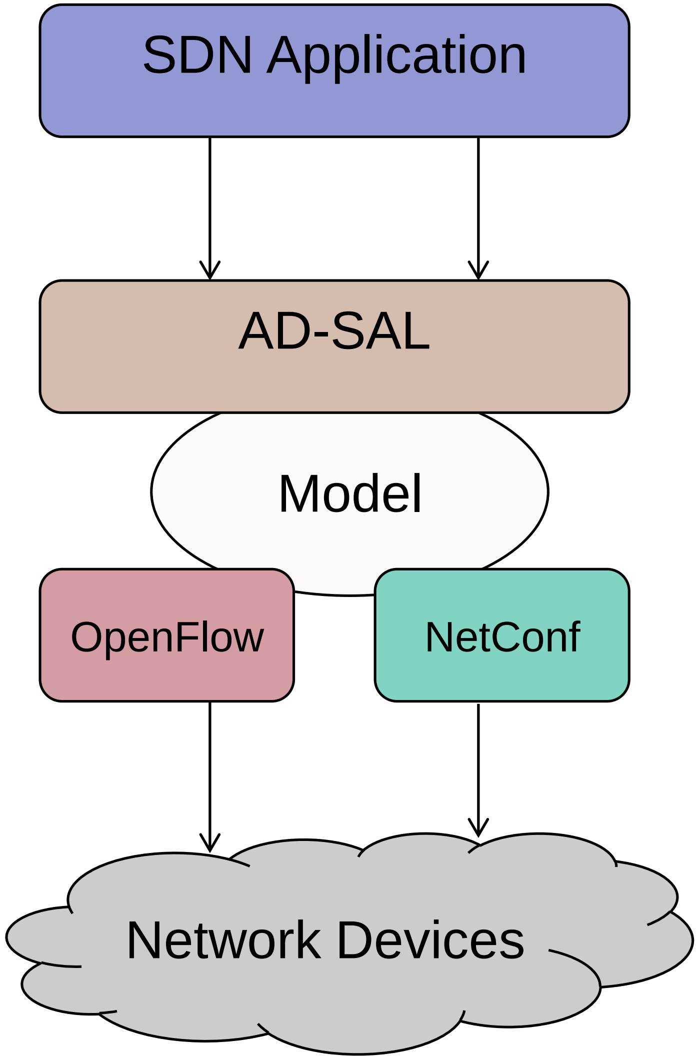
API Driven SAL (AD-SAL)

- Initial attempt at abstraction
  - API-Driven SAL, for communicating more directly with devices, using protocol(s) associated with the specific API.
- However abstraction difficult to realize in practice than it was in theory
    - AD-SAL became a collection of independent and discrete APIs, with one set of APIs for each and every southbound protocol
- AD-SAL was soon deprecated in OpenDaylight.



So how to achieve true abstraction?

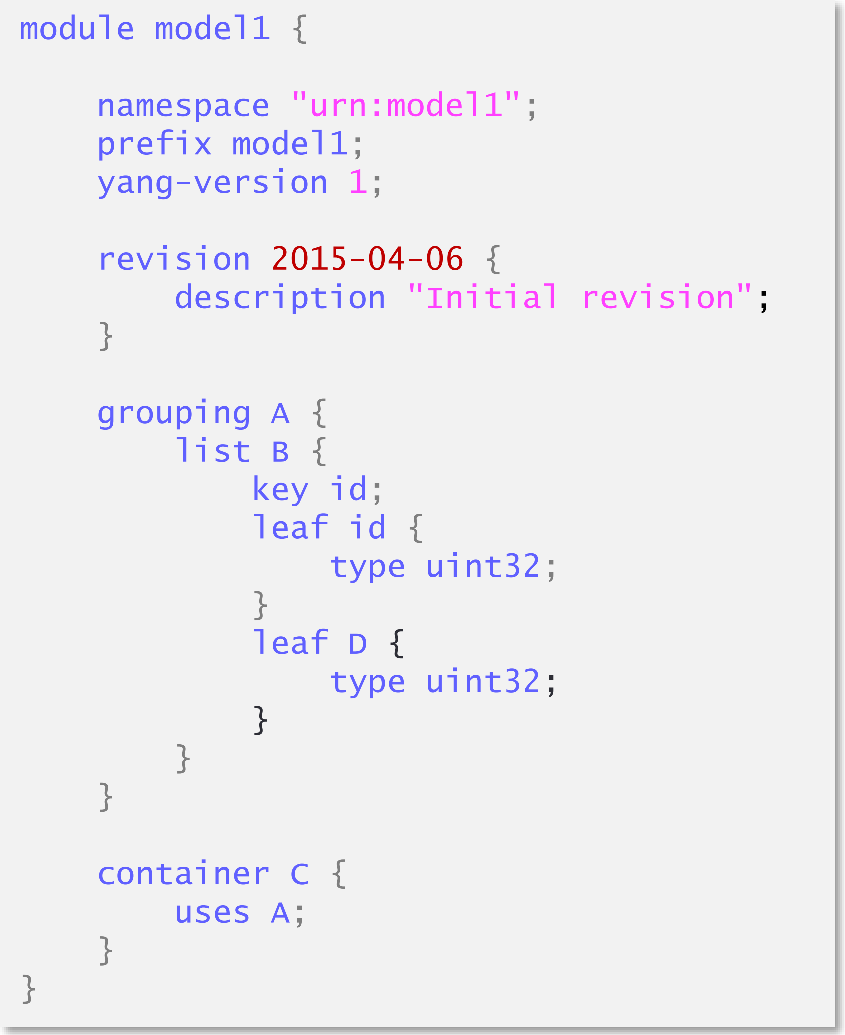
- Alternatives
  - Build a better SAL
    - Take the existing APIs for the different plugins, and attempt to come up with an API abstraction that meets all of their needs
  - Use models
    - Implement a model layer within the SAL which has SDN applications dealing with software models of network devices, rather than directly with the devices themselves.
    - This was the approach taken by OpenDaylight – to develop a **Model Driven SAL** or the **MD-SAL** built around **Yang models**



YANG

- Data modeling language that is also the preferred configuration language for NETCONF protocol

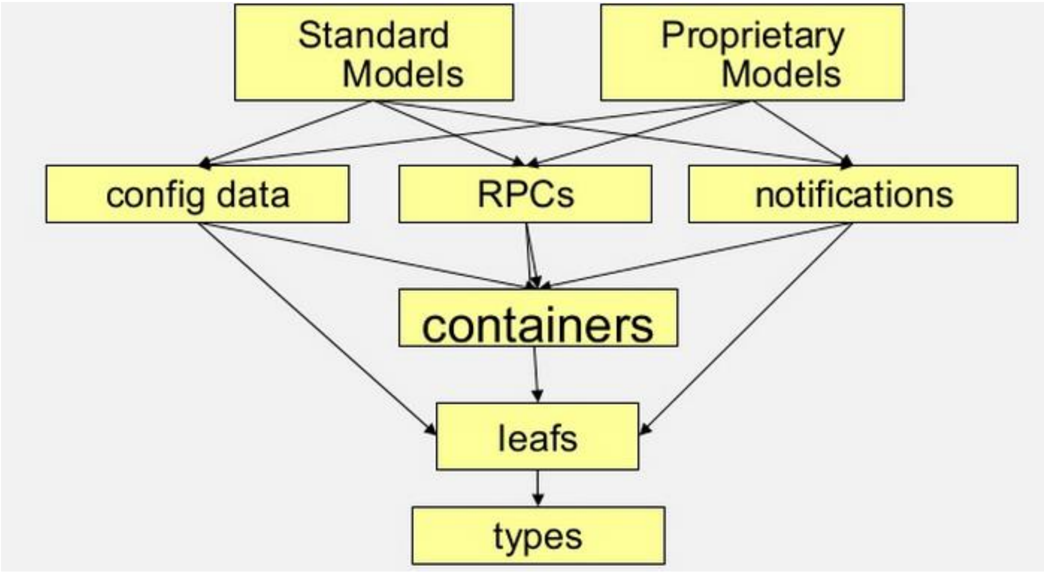
- Further reads:
    - [YANG introductory tutorial]
    - [RFC 6020 - YANG - A data modeling language for NETCONF]
    - [RFC 7950 – The YANG 1.1 Data Modeling Language]



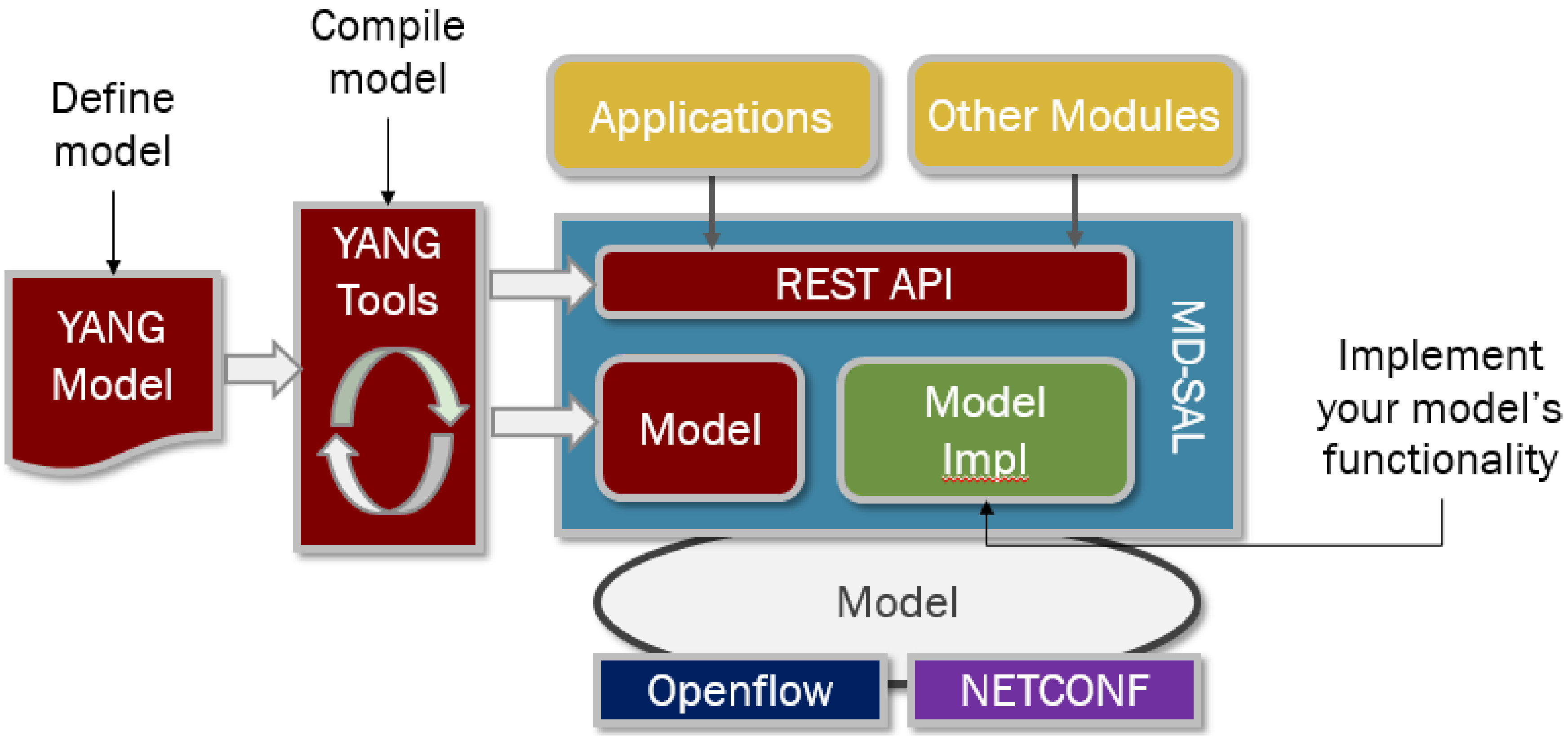
```
module model1 {

    namespace "urn:model1";
    prefix model1;
    yang-version 1;

    revision 2015-04-06 {
        description "Initial revision";
    }

    grouping A {
        list B {
            key id;
            leaf id {
                type uint32;
            }
            leaf D {
                type uint32;
            }
        }
    }

    container C {
        uses A;
    }
}
```

What can YANG model?

- Data
  - Config data
  - Operational
- RPCs:
  - Perform procedure call with input/output,
    without worrying about actual provider for that procedure
- Notifications:
  - Publish one or more notifications to registered listeners



MD-SAL Application Creation Process

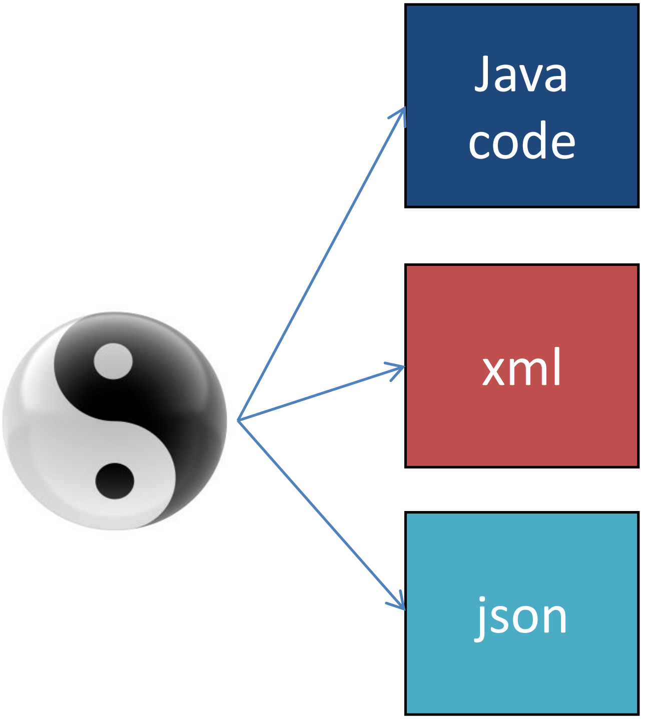


- Applications built defining models
- YANG used for defining models
- Compilation results in the skeleton of application: model, RESTCONF API, etc.

- Elements in red color above is the app skeleton
- The model implementation (green) is where you will write code to do whatever it is that your application or the model within your application does

Yangtools – What does Yangtools do?

- Generates Java code from Yang
- Provides ‘Codecs’ to convert
  - Generated Java classes to Document Object Model (DOM)
  - DOM to various formats
    - XML
    - JSON
    - Etc
- ‘Codecs’ make possible automatic:
  - RESTCONF
  - Netconf
  - Other bindings



Yang to Java benefits

- Consistent Data Transfer Objects (DTOs) everywhere

  - **Automated Bindings**:
    - restconf
    - netconf

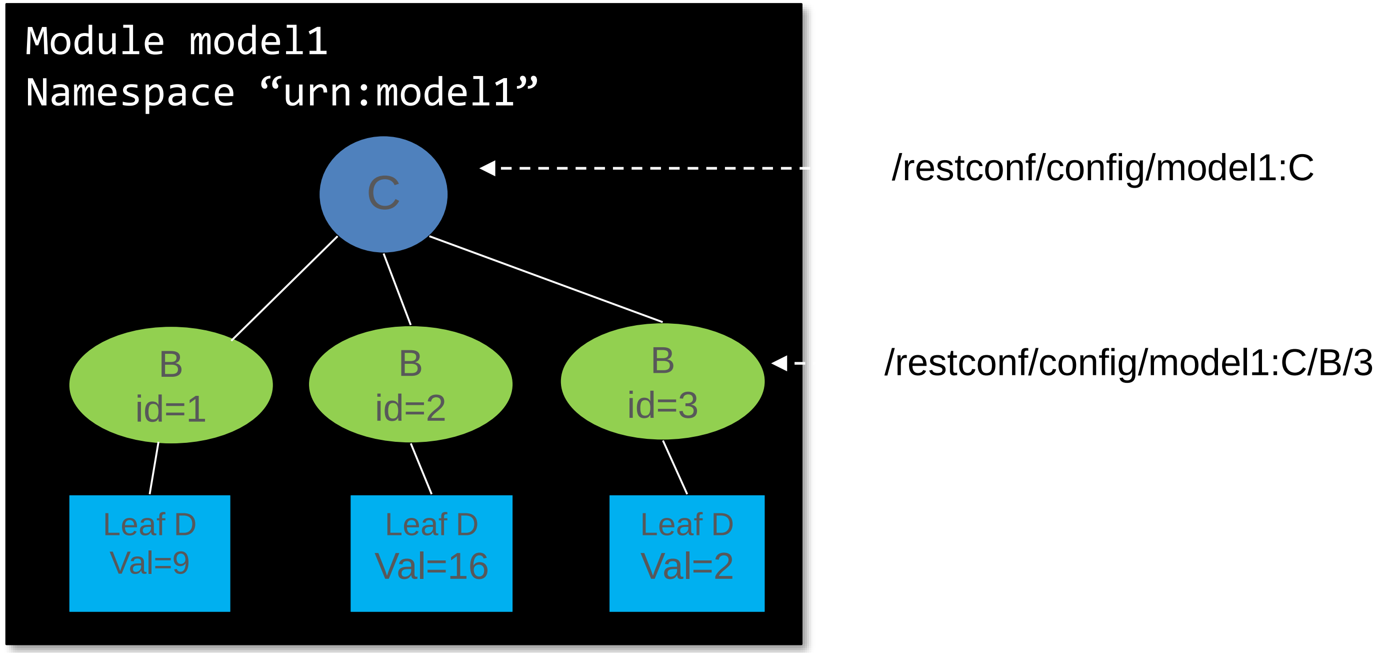
  - **Consistent**:
    - reduce learning curve

- **Immutable**: to avoid thread contention

- **Improvable** – generation can be improved and all DTOs get those improvements immediately system wide

MD-SAL

- Model-driven SAL is the kernel of the OpenDaylight controller
- It manages the contracts and state exchanges between every application. It does this adaptation by managing centralized state
- Takes in the YANG model at runtime and constructs the tree in the data store



```
Module model1
Namespace “urn:model1”
```

C

B
id=1

B
id=2

B
id=3

Leaf D
Val=9

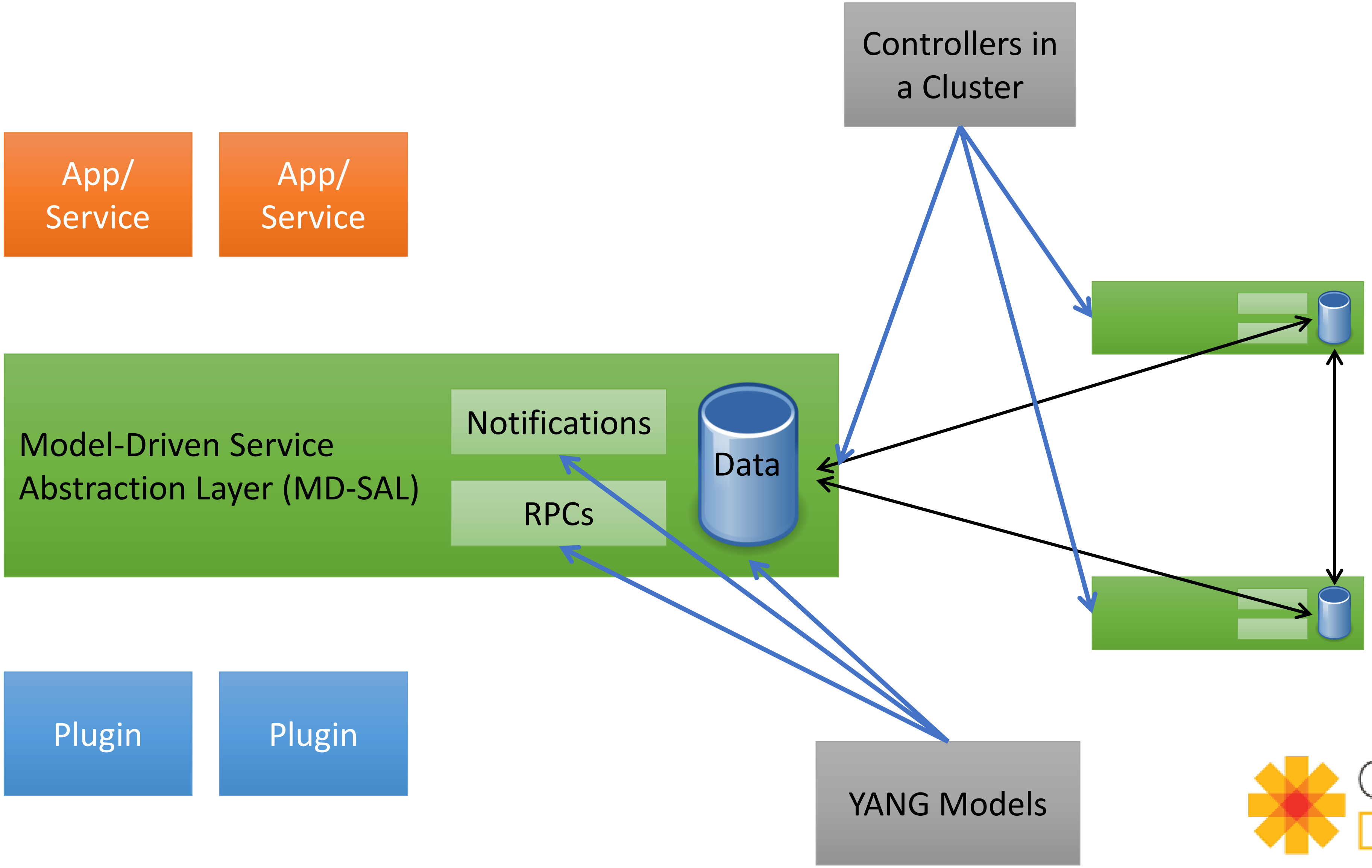
Leaf D
Val=16

Leaf D
Val=2

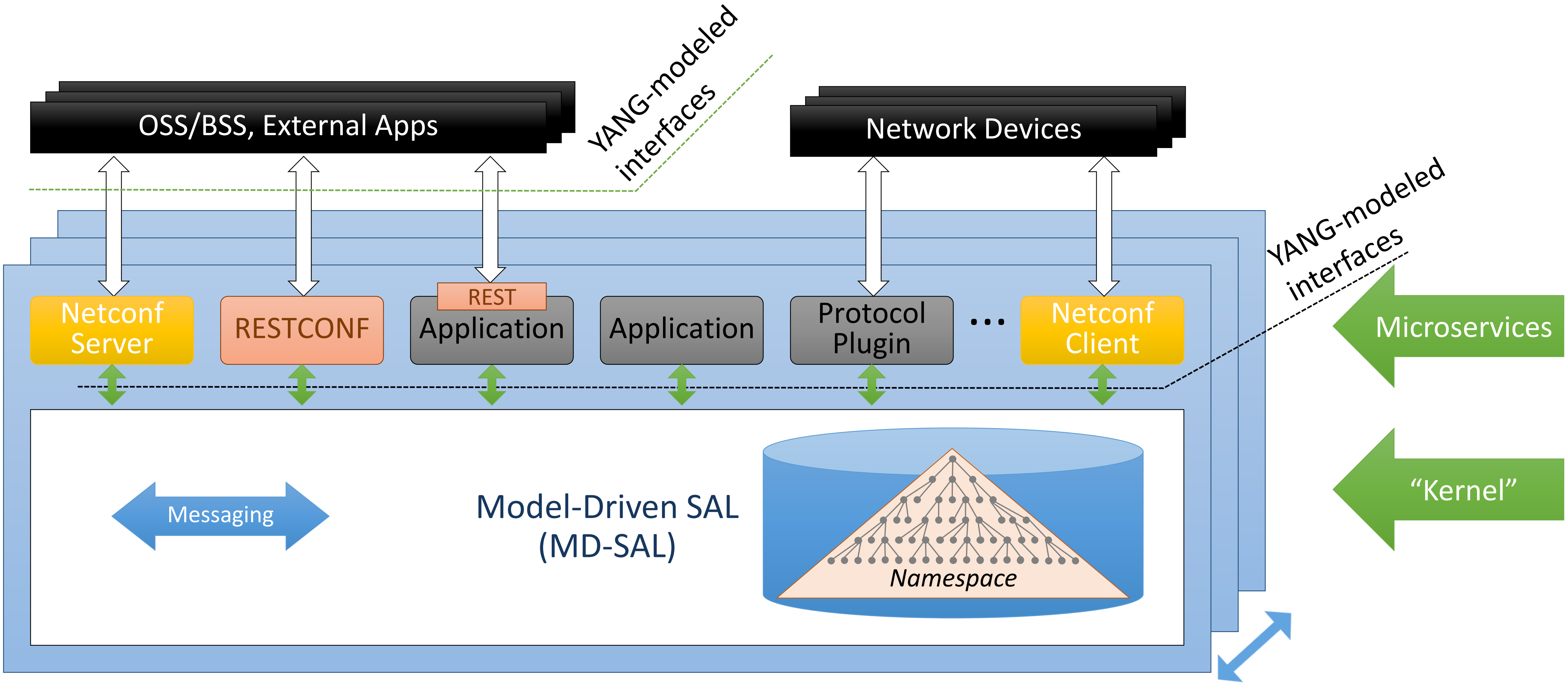
/restconf/config/model1:C

/restconf/config/model1:C/B/3

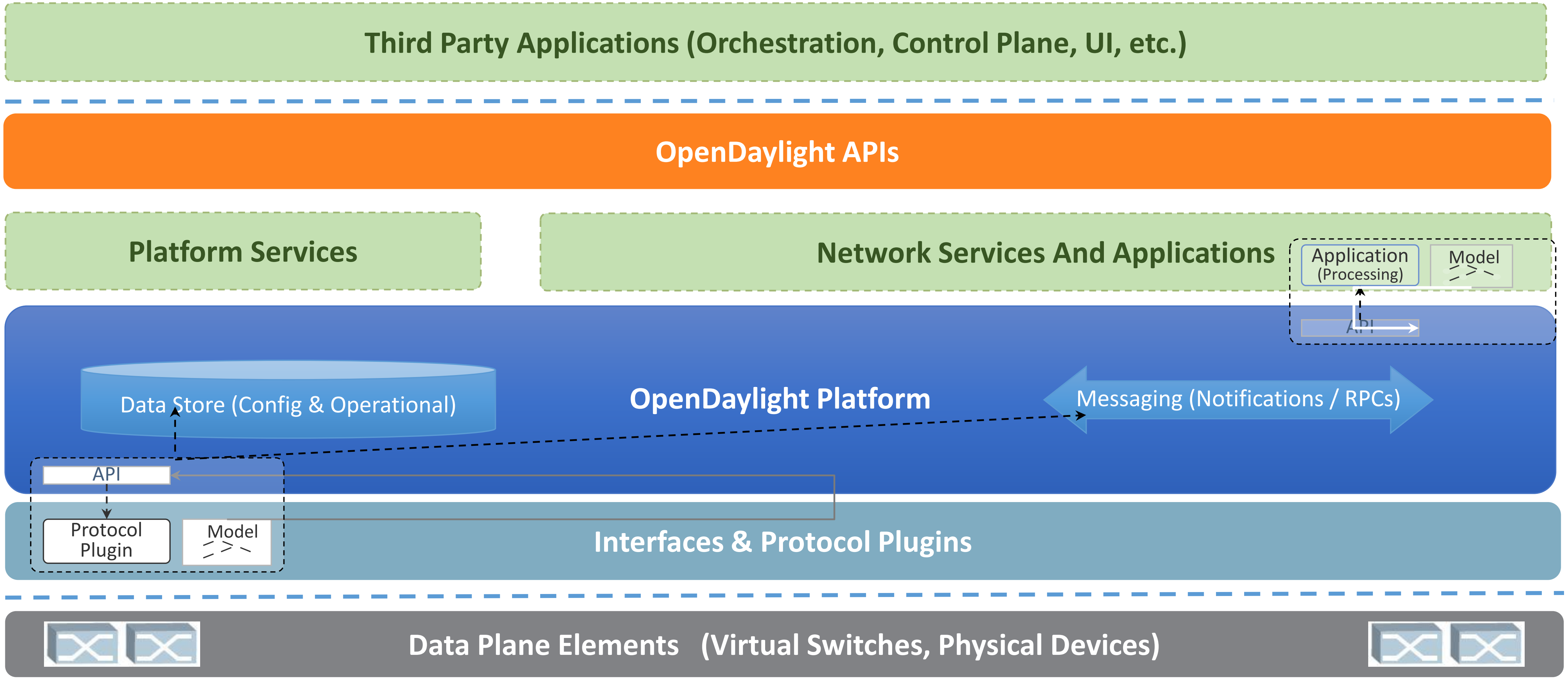
OpenDaylight Architecture - Simplified View



An Aspect of the architecture: ODL is a µ-services platform



OpenDaylight Architecture - Operational View



Third Party Applications (Orchestration, Control Plane, UI, etc.)

OpenDaylight APIs

Platform Services

Network Services And Applications

OpenDaylight Platform

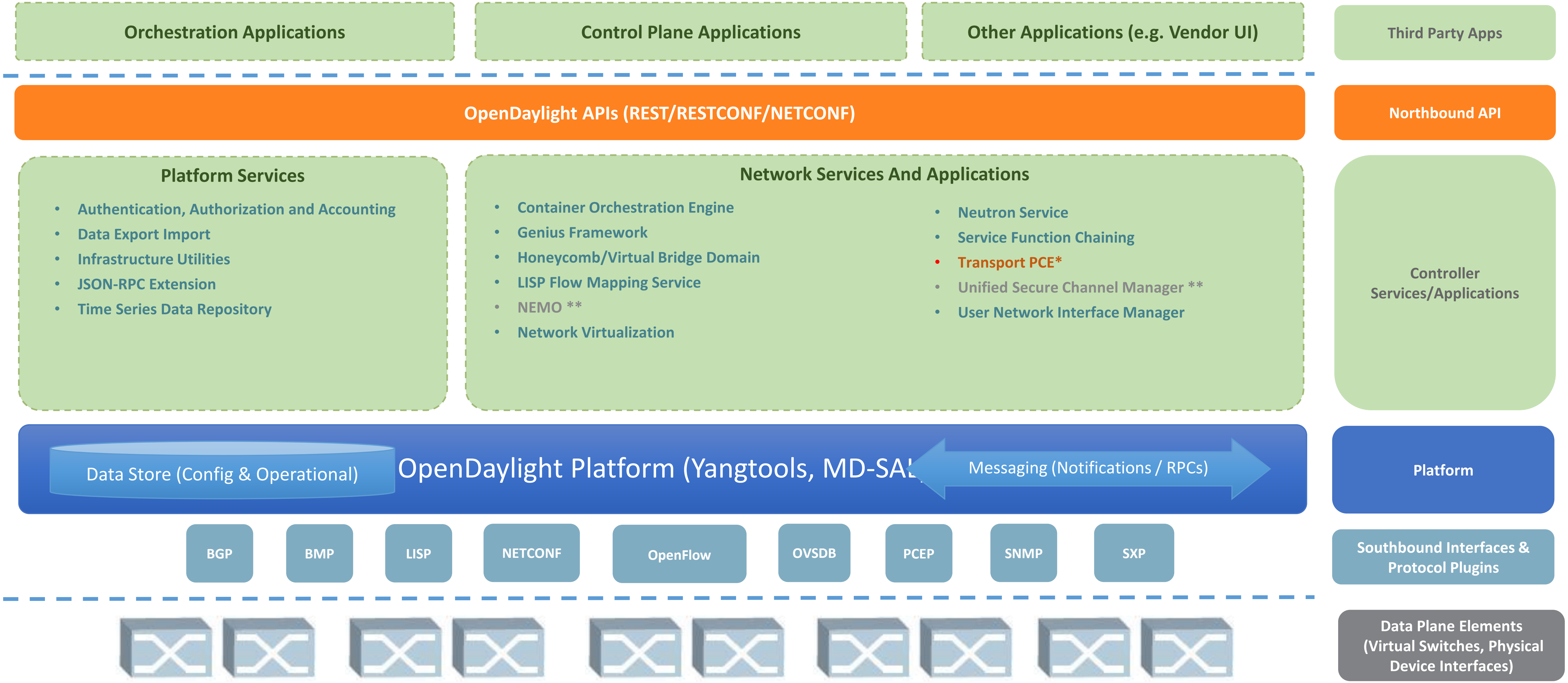
Data Store (Config & Operational)

Messaging (Notifications / RPCs)

Interfaces & Protocol Plugins

Data Plane Elements (Virtual Switches, Physical Devices)

OpenDaylight Fluorine Release



Orchestration Applications

Control Plane Applications

Other Applications (e.g. Vendor UI)

Third Party Apps

OpenDaylight APIs (REST/RESTCONF/NETCONF)

Northbound API

Platform Services

- Authentication, Authorization and Accounting
- Data Export Import
- Infrastructure Utilities
- JSON-RPC Extension
- Time Series Data Repository

Network Services And Applications

- Container Orchestration Engine
- Genius Framework
- Honeycomb/Virtual Bridge Domain
- LISP Flow Mapping Service
- NEMO **
- Network Virtualization
- Neutron Service
- Service Function Chaining
- Transport PCE*
- Unified Secure Channel Manager **
- User Network Interface Manager

Controller Services/Applications

Data Store (Config & Operational)

OpenDaylight Platform (Yangtools, MD-SAL

Messaging (Notifications / RPCs)

Platform

BGP

BMP

LISP

NETCONF

OpenFlow

OVSDB

PCEP

SNMP

SXP

Southbound Interfaces & Protocol Plugins

Data Plane Elements (Virtual Switches, Physical Device Interfaces)

* First release for the project

** Not included in Fluorine distribution - separate download

OpenDaylight Architecture: Key Takeaway

- OpenDaylight architecture is amenable to be applied to a variety of use cases as:
  - Not tied to a particular protocol
  - Modular, Extensible
  - Has built-in tools to simplify application development

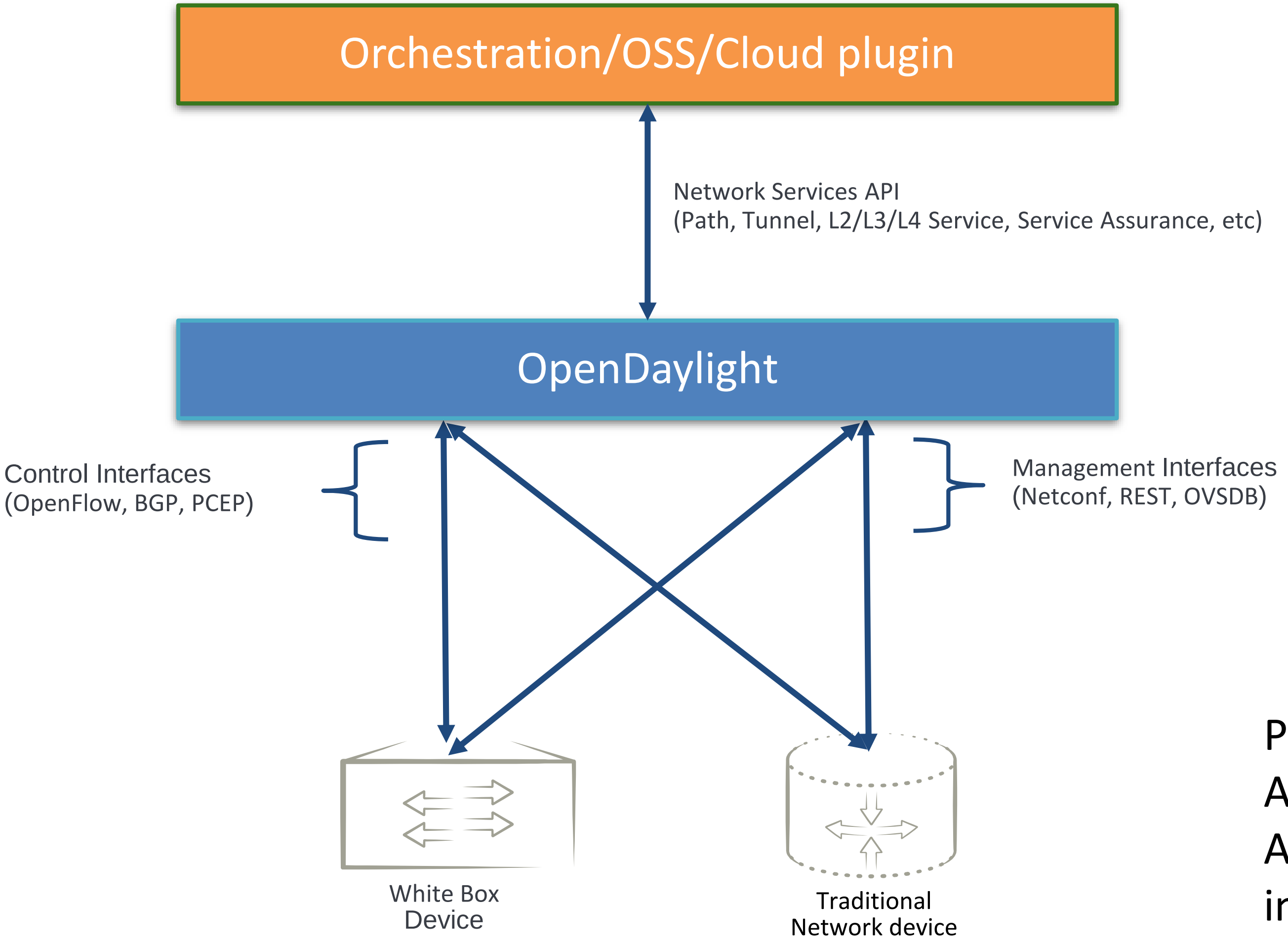
OpenDaylight Use Cases (Partial List)

Note

OpenDaylight architecture has been used in many use cases – not all covered here

Use Case I

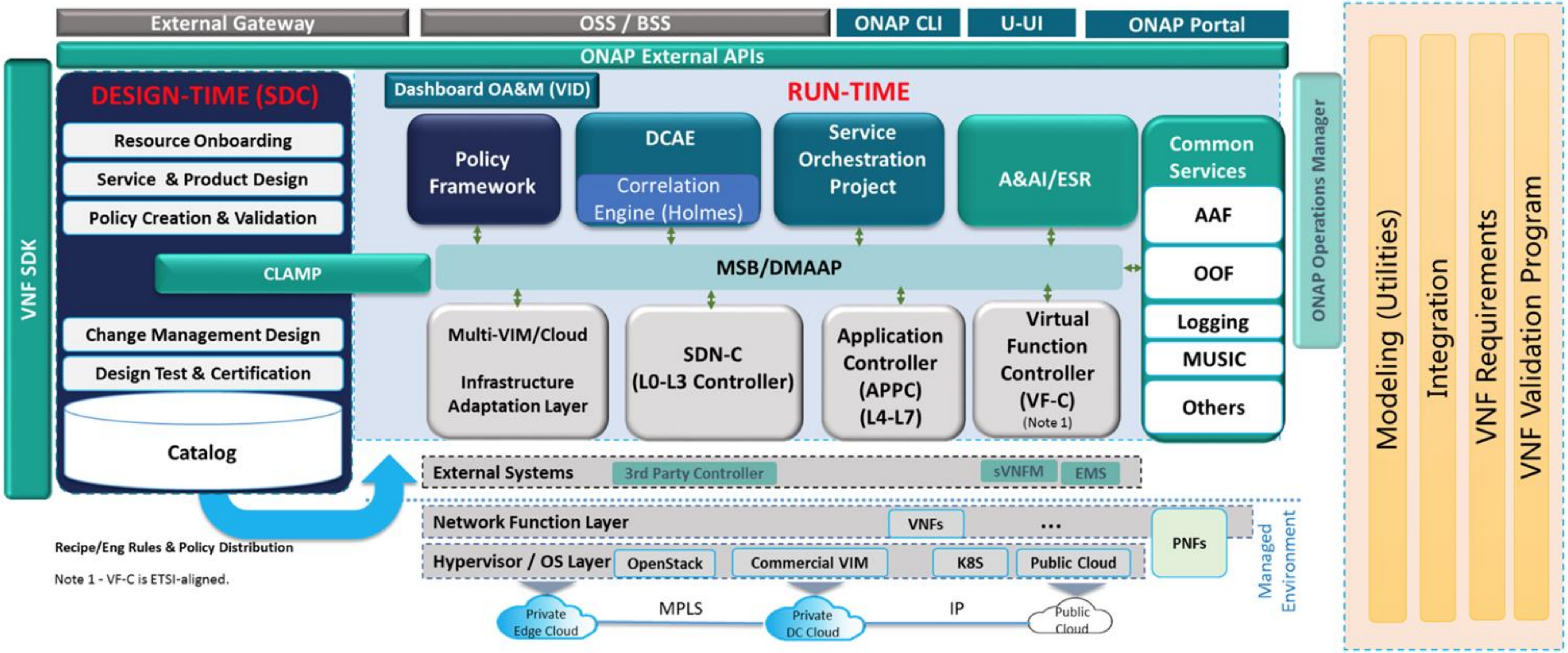
Network Abstraction



Provides Network Services API for Network Automation
in a Multi Vendor Network

Use Case II

ONAP Project

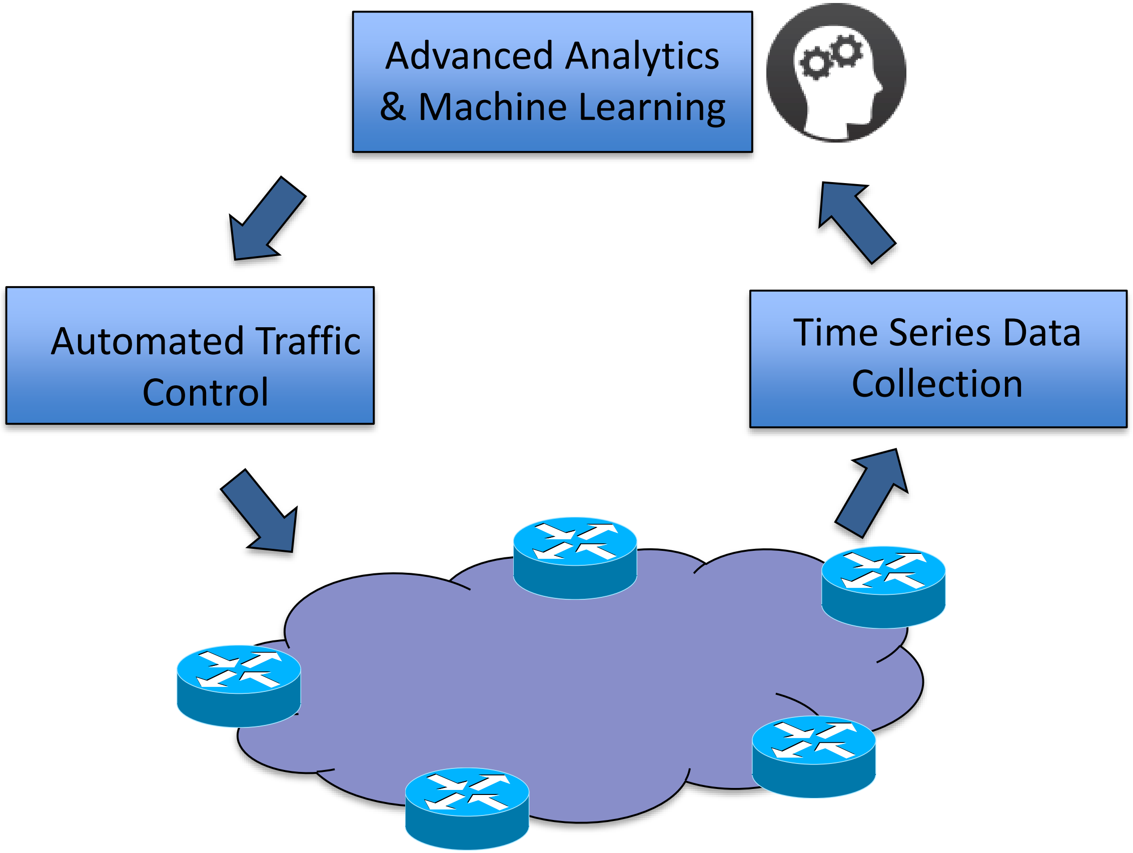


SDN-C & App-C based on OpenDaylight code

Use Case III (future)

AI/ML with OpenDaylight

Smart SDN Controller


Advanced Analytics & Machine Learning
Time Series Data Collection
Automated Traffic Control

- Network status awareness
  - ➢ Rely on time series data collected from the network
- Traffic Control Policy Change decision making
  - ➢ Based on the advanced analytics and machine learning.
- Dynamic change of Control policies
  - ➢ Automatically change the traffic control policies based on the analytics results.

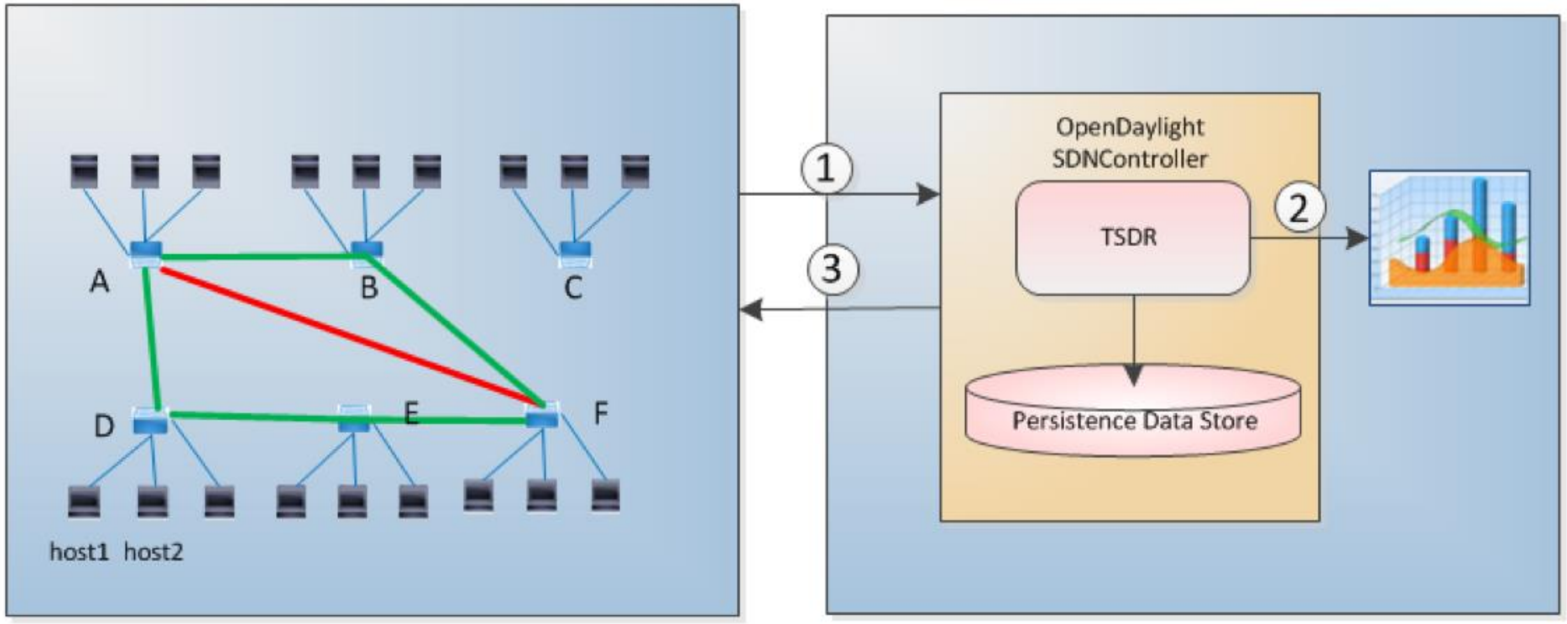
Why we need Machine Learning in SDN

- Software Defined Networks needs to be intelligent.
  - To be aware of the runtime status of the network.
  - To make the right decisions that adjust the policies for traffic classification and traffic shaping.
  - To dynamically change the policies according to the analytics results.
    - AI / MI can be used to establish normalized profiles and dynamically update the profiles based on a set of predetermined or dynamically learned rules.

Use Cases of a smart and intelligent SDN controller

- Traffic Control and Routing Optimization
  - Congestion Control
  - Traffic Pattern Prediction
  - Routing Optimization
- Resource optimization
  - Networking resource allocation optimization
  - Cloud resource management optimization
- Security and Anomaly Detection
  - DDoS attack detection and mitigation
- Troubleshooting and Self-healing

AI/ML Example Use Case
Traffic congestion prediction with automated control



SDN controlled network

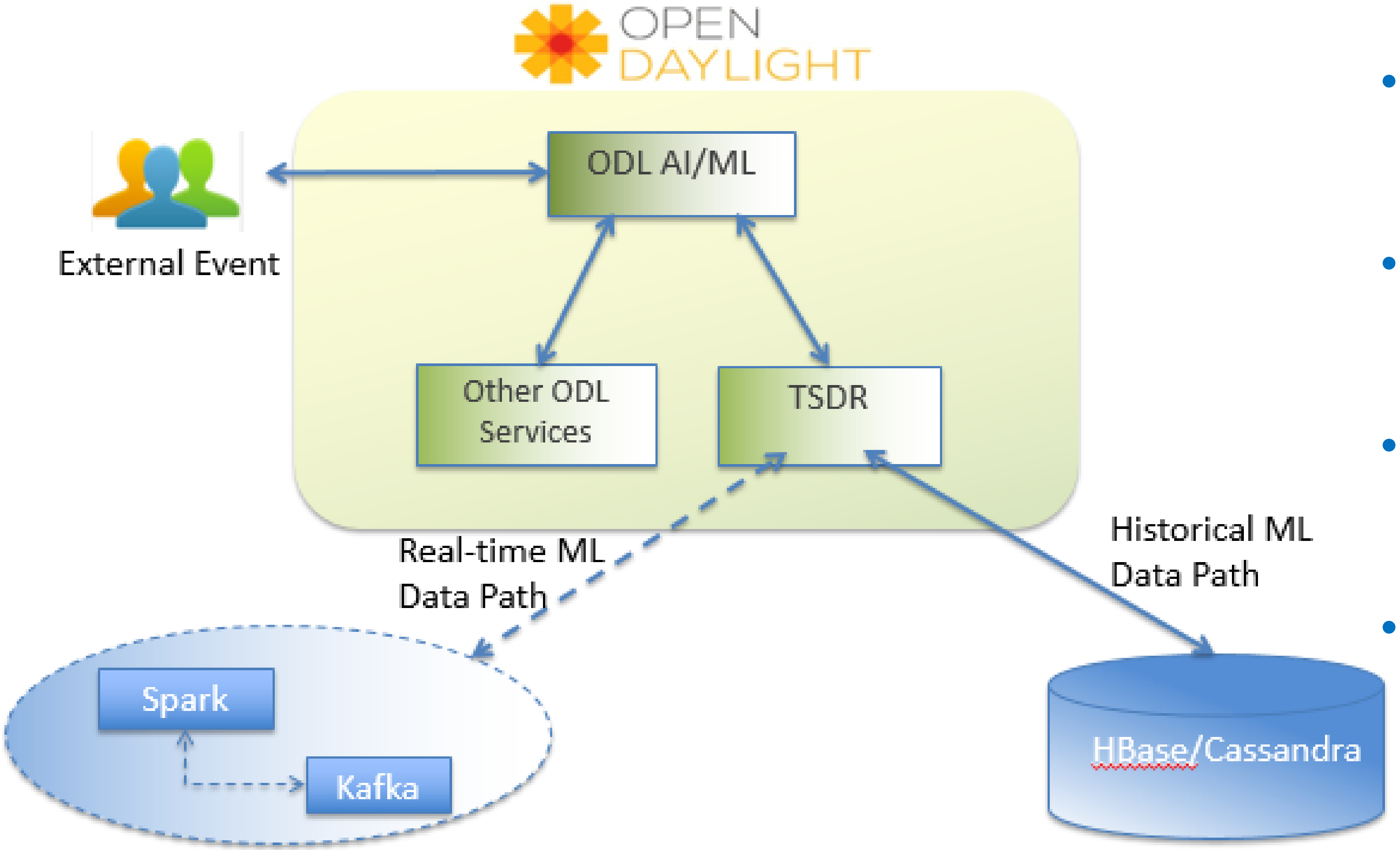
OpenDaylight + TSDR

1. Collect stats from the network and store into TSDR
2. Data analysis through data analytics engines integration
3. Traffic flow redirection from A->F to A->B->F and A->D->E->F

🟥 Predicted congestion path in the next 24 hours

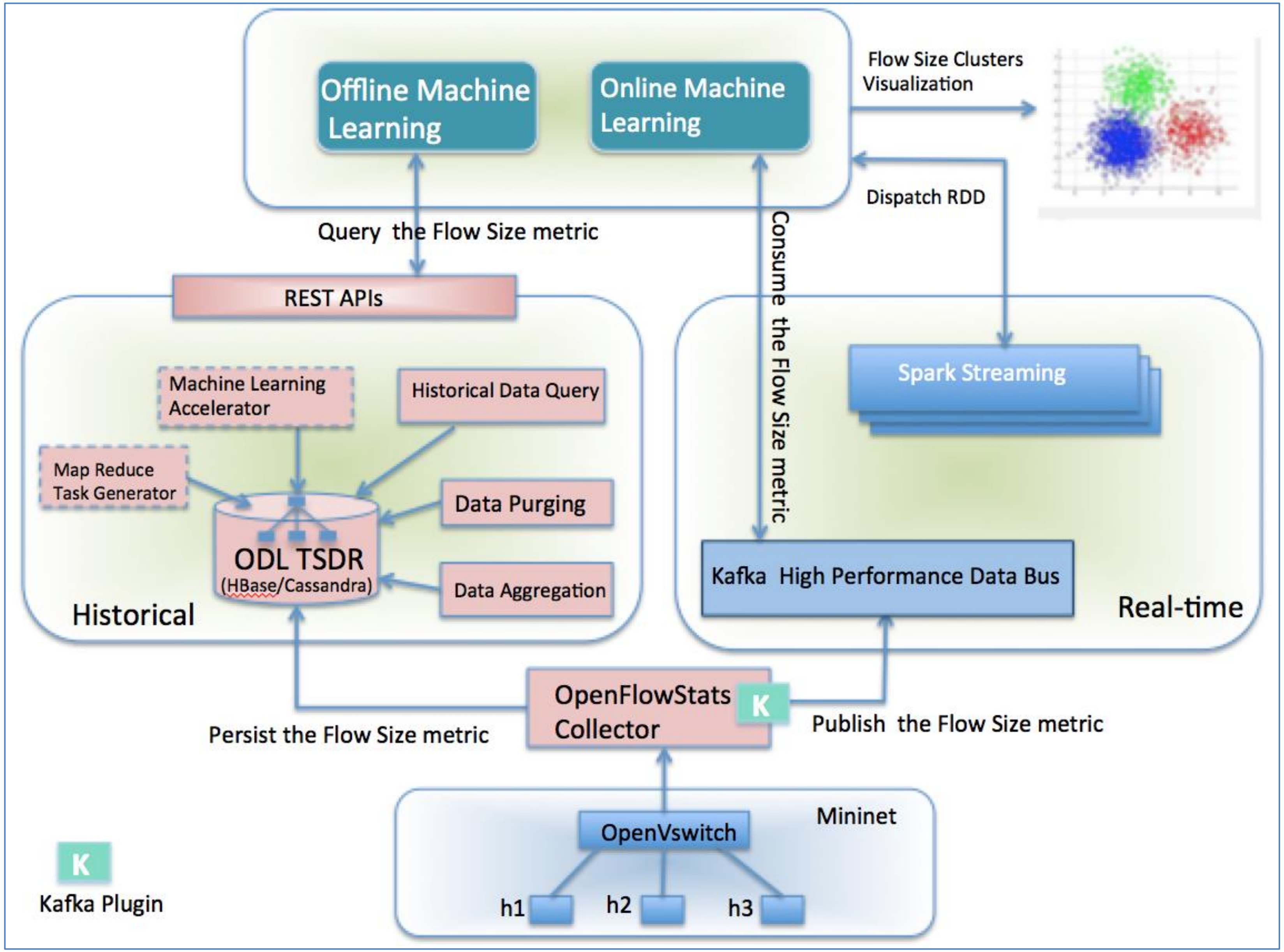
🟩 Healthy path in the next 24 hours

ODL AI/ML framework in the ODL ecosystem



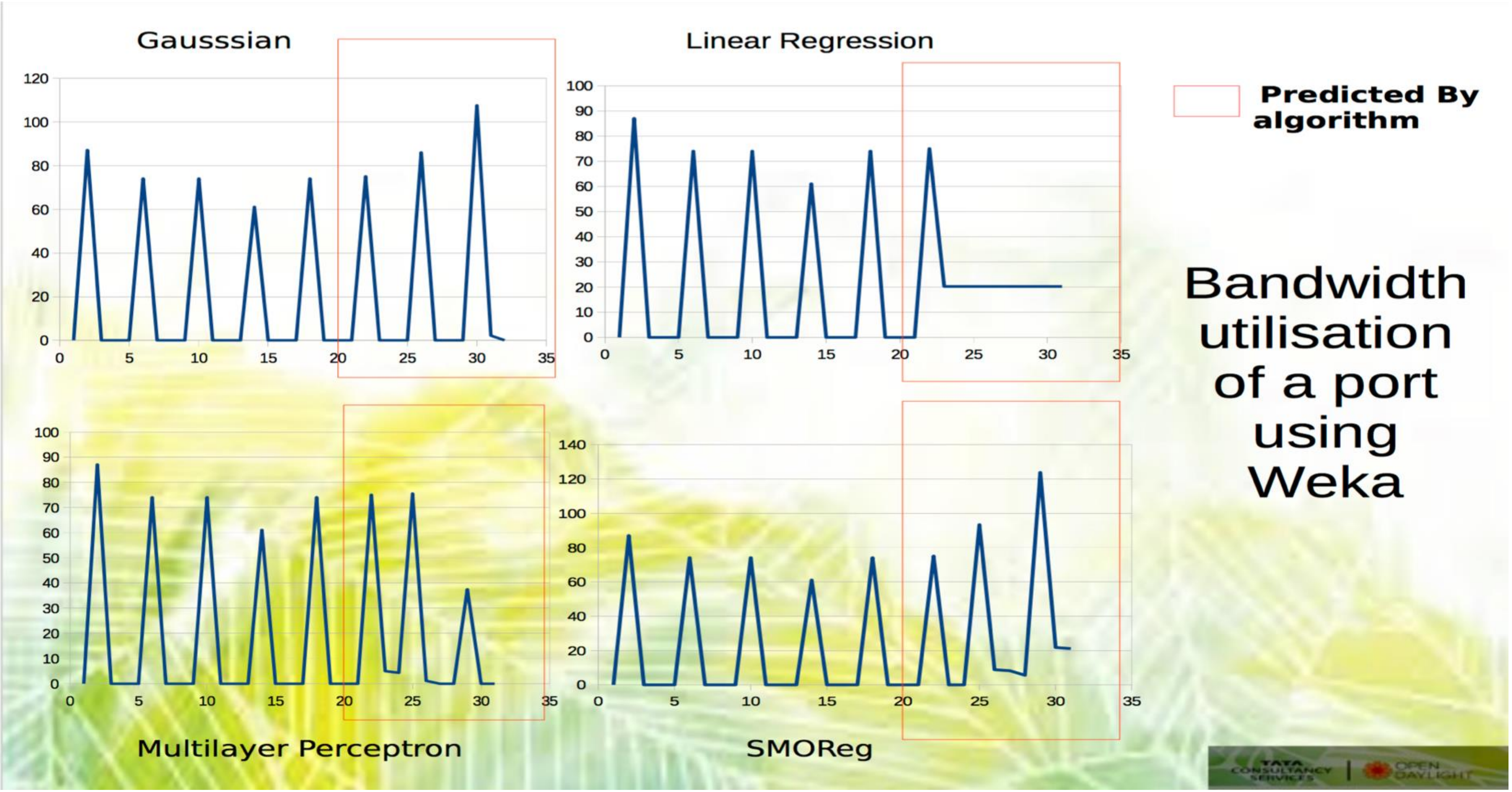
- Enable AI/ML on both historical and real-time data paths.
- Many use cases would require both offline and online ML on the time series data.
- External events could be additional input for accurate machine learning results.
- Feed back the results to SDN control path for automatic traffic steering and policy placement.
- Well-defined interface among the components towards future standardization of advanced analytics in SDN.

ODL AI/ML framework PoC Architecture



- PoC of both historical offline machine learning and real-time online machine learning
  - ➢ Collect the time series data
  - ➢ Persist into scalable data storage
  - ➢ Publish to high performance data bus
- Integrate with external machine learning libraries
  - ➢ Spark MLlib
  - ➢ DeepLearning4J
  - ➢ Future: TensorFlow?
- Collect OpenFlow Stats and apply machine learning algorithms
  - ➢ *k*-means clustering

Prediction using Weka leveraging data collected in TSDR



□ **Predicted By algorithm**

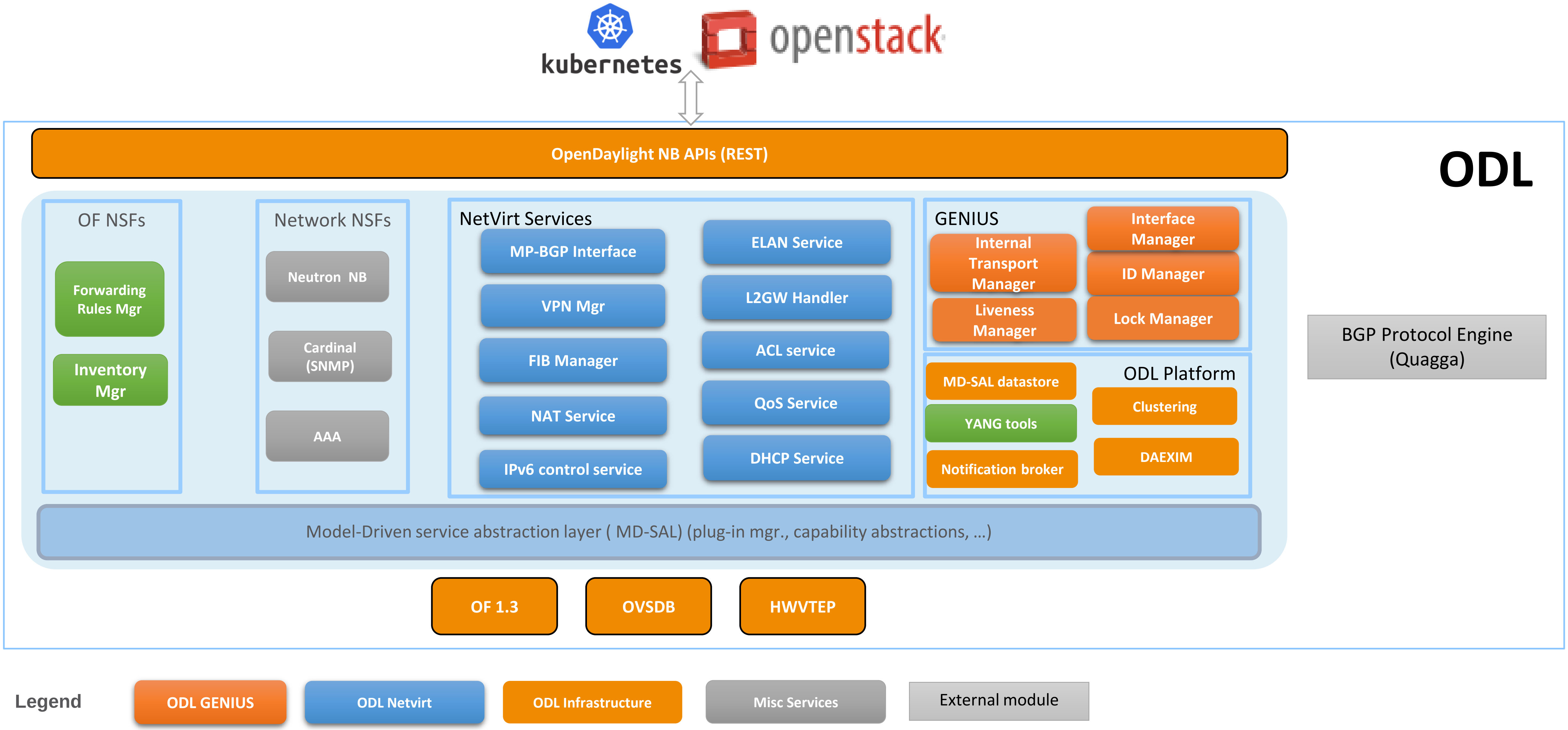
Bandwidth utilisation of a port using Weka

Use Case IV

Network Virtualization

- A set of projects working in tandem to provide network virtualization (overlay connectivity) inside and between data centers for Cloud SDN use case
  - VxLAN within the data center
  - L3 VPN across data centers
- Integration with OpenStack Neutron and Kubernetes (in-progress)
- Uses Open vSwitch and hardware VTEPs (ToR) as the datapath

Network Virtualization: OpenDaylight Components



kubernetes

openstack

ODL

OpenDaylight NB APIs (REST)

OF NSFs

- Forwarding Rules Mgr
- Inventory Mgr

Network NSFs

- Neutron NB
- Cardinal (SNMP)
- AAA

NetVirt Services

- MP-BGP Interface
- VPN Mgr
- FIB Manager
- NAT Service
- IPv6 control service
- ELAN Service
- L2GW Handler
- ACL service
- QoS Service
- DHCP Service

GENIUS

- Internal Transport Manager
- Liveness Manager
- Interface Manager
- ID Manager
- Lock Manager

ODL Platform

- MD-SAL datastore
- YANG tools
- Notification broker
- Clustering
- DAEXIM

BGP Protocol Engine (Quagga)

Model-Driven service abstraction layer (MD-SAL) (plug-in mgr., capability abstractions, …)

OF 1.3

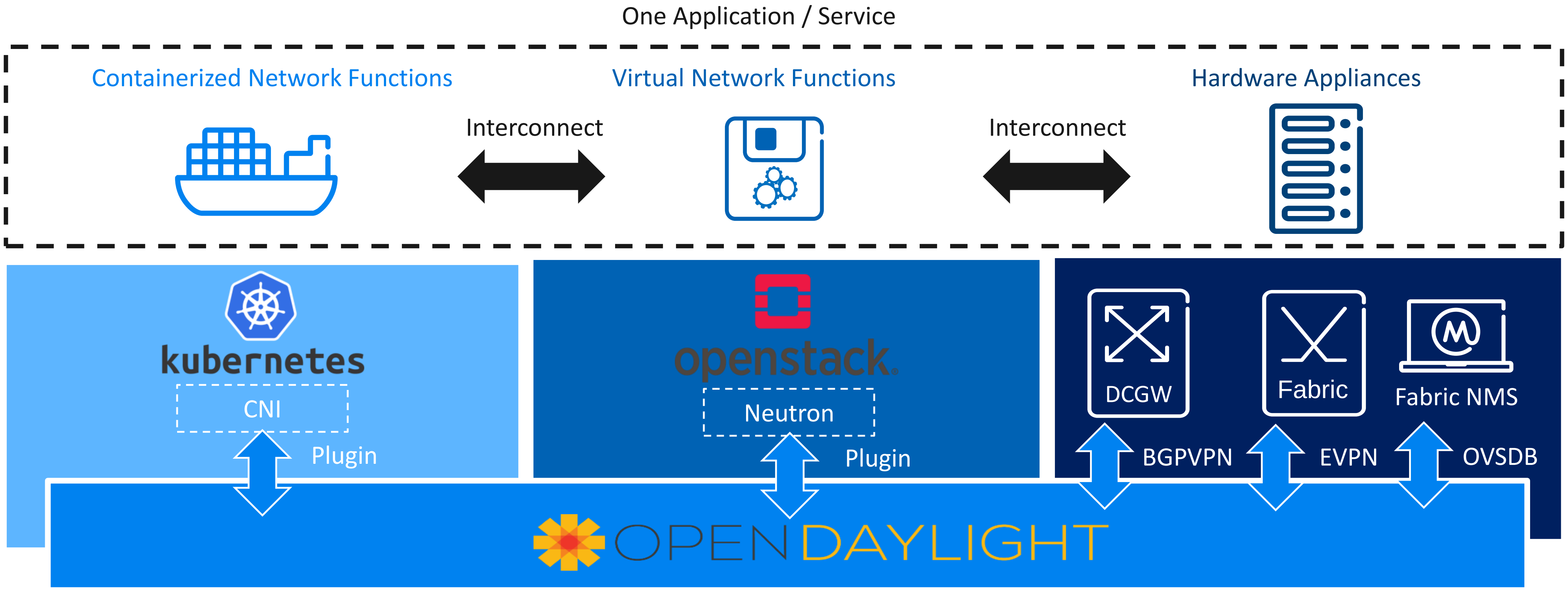
OVSDB

HWVTEP

Legend

- ODL GENIUS
- ODL Netvirt
- ODL Infrastructure
- Misc Services
- External module

A common controller platform



One Application / Service

Containerized Network Functions

Interconnect

Virtual Network Functions

Interconnect

Hardware Appliances

kubernetes

CNI

Plugin

openstack

Neutron

Plugin

DCGW

Fabric

Fabric NMS

BGPVPN

EVPN

OVSDB

OPENDAYLIGHT

Uniform service capabilities

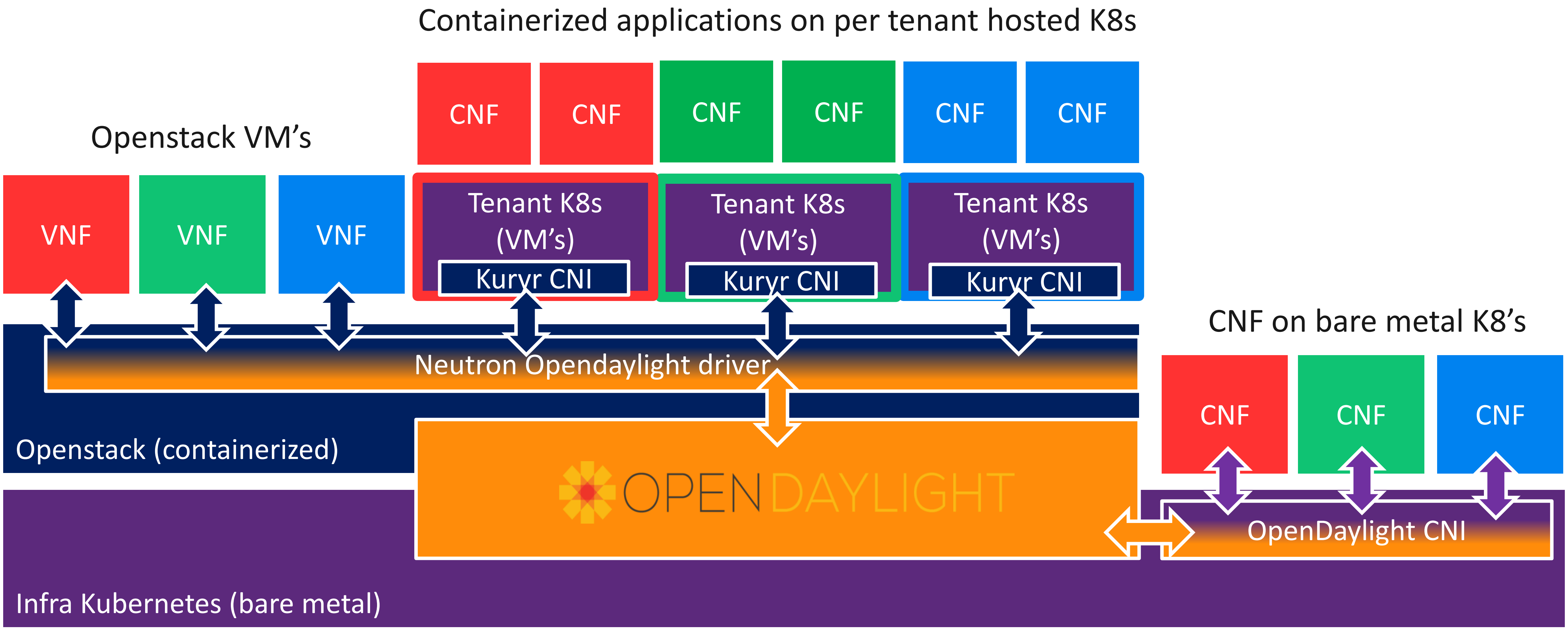
Simplified interworking

Reduced training And validation

Simplified troubleshooting

Common dashboard

OpenDaylight multi-instance controller



OpenDaylight Container Orchestration Engine

- Current Status
  - Hybrid scenario:
    - Openstack and Kubernetes side by side
      - Integration with ODL via Openstack Kuryr
      - Supports Multinode environment
      - Supports container in a VM scenario
  - Baremetal scenario
    - Kubernetes only
      - Tight integration with ODL NetVirt
      - Supports Pod 2 Pod networking L2/L3

- Future Scenarios
  - Support for non-OF southbound
    - NetConf
  - Testing with L3VPN for multi-tenant scenarios
  - Scale testing & improvement

Use Case V

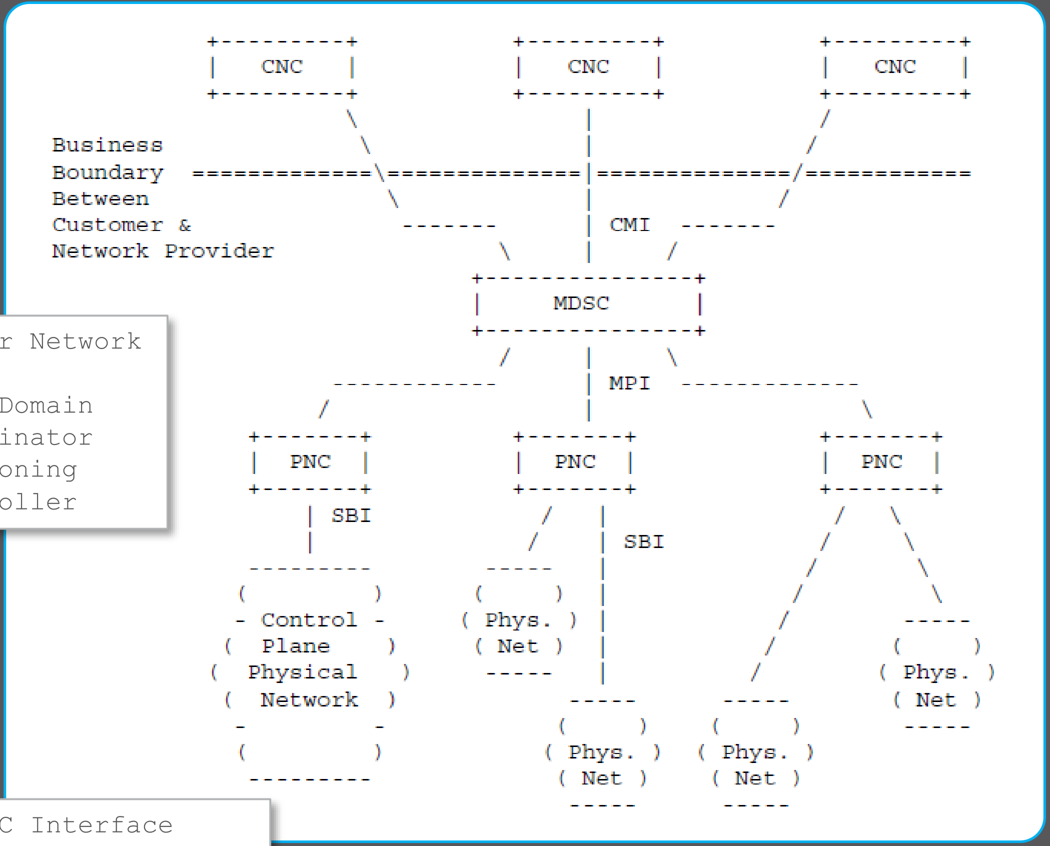
OpenDaylight in OSS (future)

WAN Transport Orchestrator (WAN-O)

- Based on ACTN (Abstraction of Control of Traffic Engineered Network) IETF Standard for realizing hierarchical SDN architecture
  - Yang Based (NetConf/RESTCONF) Models

SDN Hierarchical architecture based on ACTN

- Coordination of resources across multiple independent networks and multiple technology layers to provide end-to-end services
- Layered operational model:
  - *Customer:* issuing a service request from catalog
  - *Service Provider:* dealing w/ Customer and providing the service (may or may not own the network(s) as such)
  - *Network Provider:* infrastructure providers owning the physical network(s) and building the infrastructure



```
             +----------+          +----------+          +----------+
             |   CNC    |          |   CNC    |          |   CNC    |
             +----------+          +----------+          +----------+
                        \                |                /
Business                 \               |               /
Boundary  ============\==============|==============/============
Between                    \             |             /
Customer &                  -------      | CMI  -------
Network Provider                   \     |     /
                               +---------------+
                               |     MDSC      |
                               +---------------+
                                  /      |      \
                  ------------           | MPI   -------------
                 /                       |                    \
          +--------+               +--------+             +--------+
          |  PNC   |               |  PNC   |             |  PNC   |
          +--------+               +--------+             +--------+
              | SBI                  /    |                 /   \
              |                     /     | SBI            /     \
          ---------             -----     |               /       \
         (         )           (     )    |              /         \
         - Control -          ( Phys. )   |             /         -----
        (  Plane    )          ( Net )    |            /         (     )
       (  Physical   )          -----     |           /         ( Phys. )
        (  Network  )                   -----       -----        ( Net )
         -         -                   (     )     (     )        -----
         (         )                  ( Phys. )   ( Phys. )
          ---------                    ( Net )     ( Net )
                                        -----       -----
```

CNC - Customer Network Controller
MDSC - Multi Domain Service Coordinator
PNC - Provisioning Network Controller

CMI - CNC-MDSC Interface
MPI - MDSC-PNC Interface
SBI - South Bound Interface

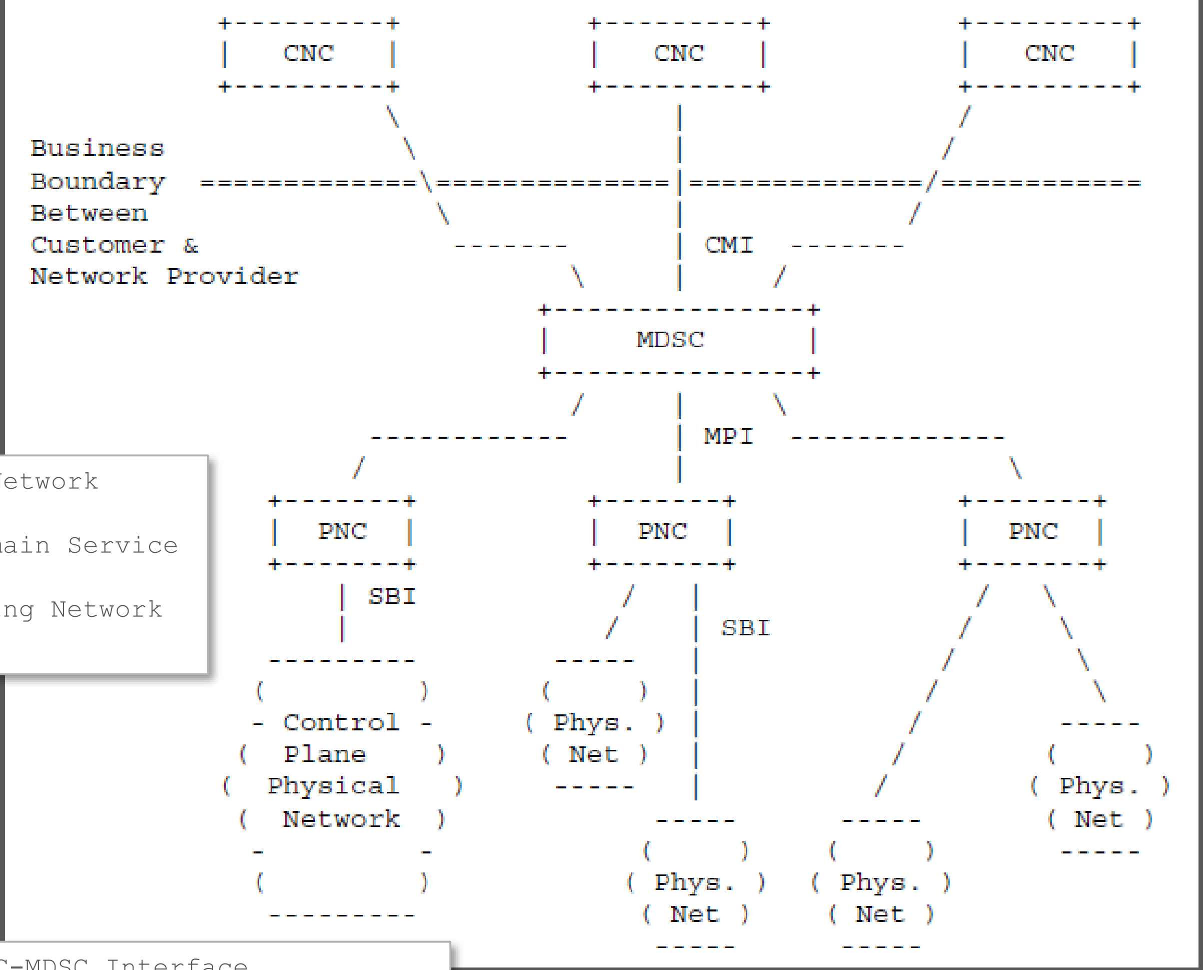
WAN-O as MDSC, interfaces

MDSC NBI:

- CMI: CNC to MDSC interface
- YANG based (Netconf/Restconf)
- End to end Virtual Network concept
- Unified end to end topology

MDSC SBI:

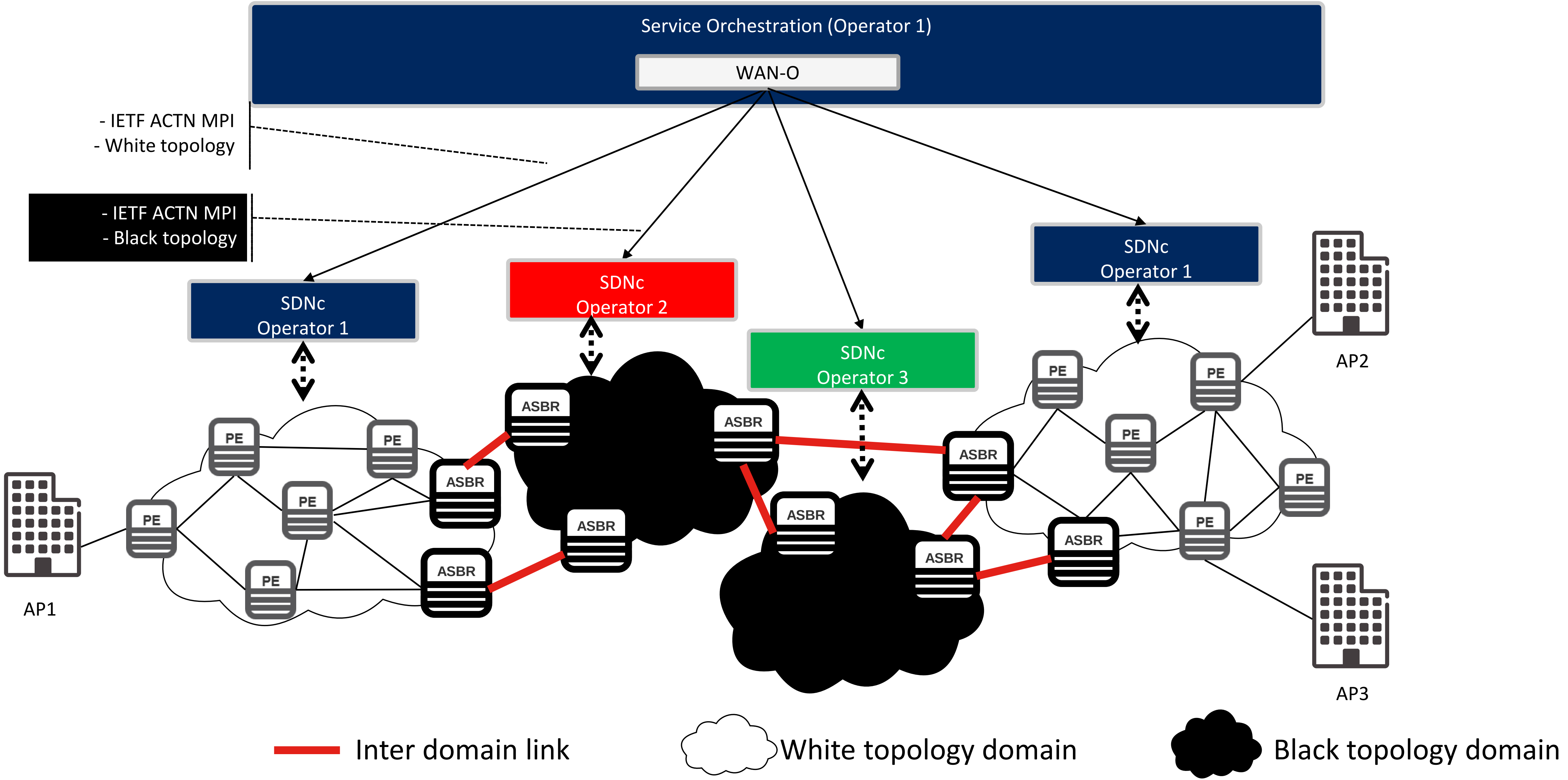
- MPI: MDSC to PNC interface
- YANG based (Netconf/Restconf)
- Per domain TE-Tunnels
- White or Black Domain topology



CNC - Customer Network Controller
MDSC - Multi Domain Service Coordinator
PNC - Provisioning Network Controller

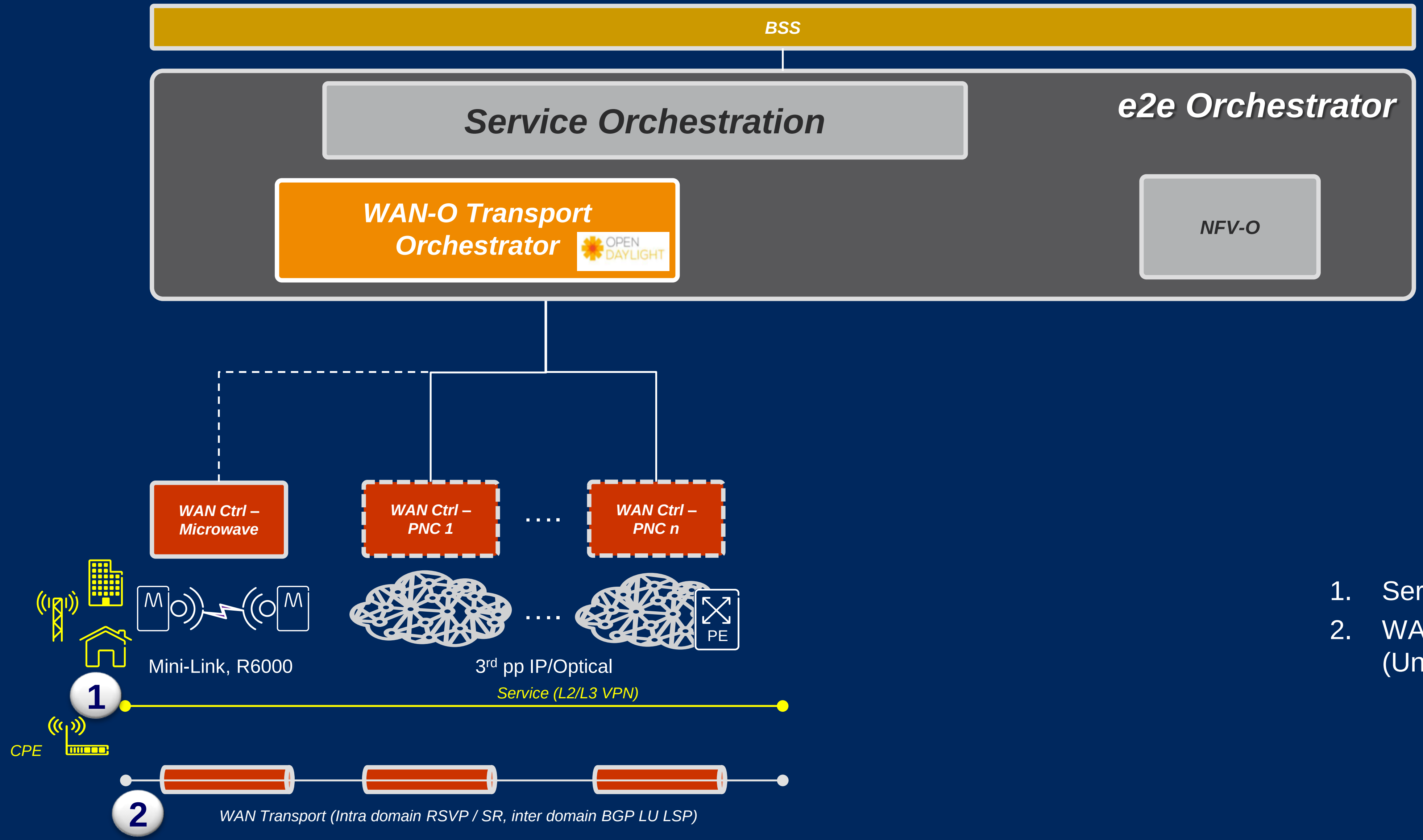
CMI - CNC-MDSC Interface
MPI - MDSC-PNC Interface
SBI - South Bound Interface

Transport Network architecture



END to END service orchestration

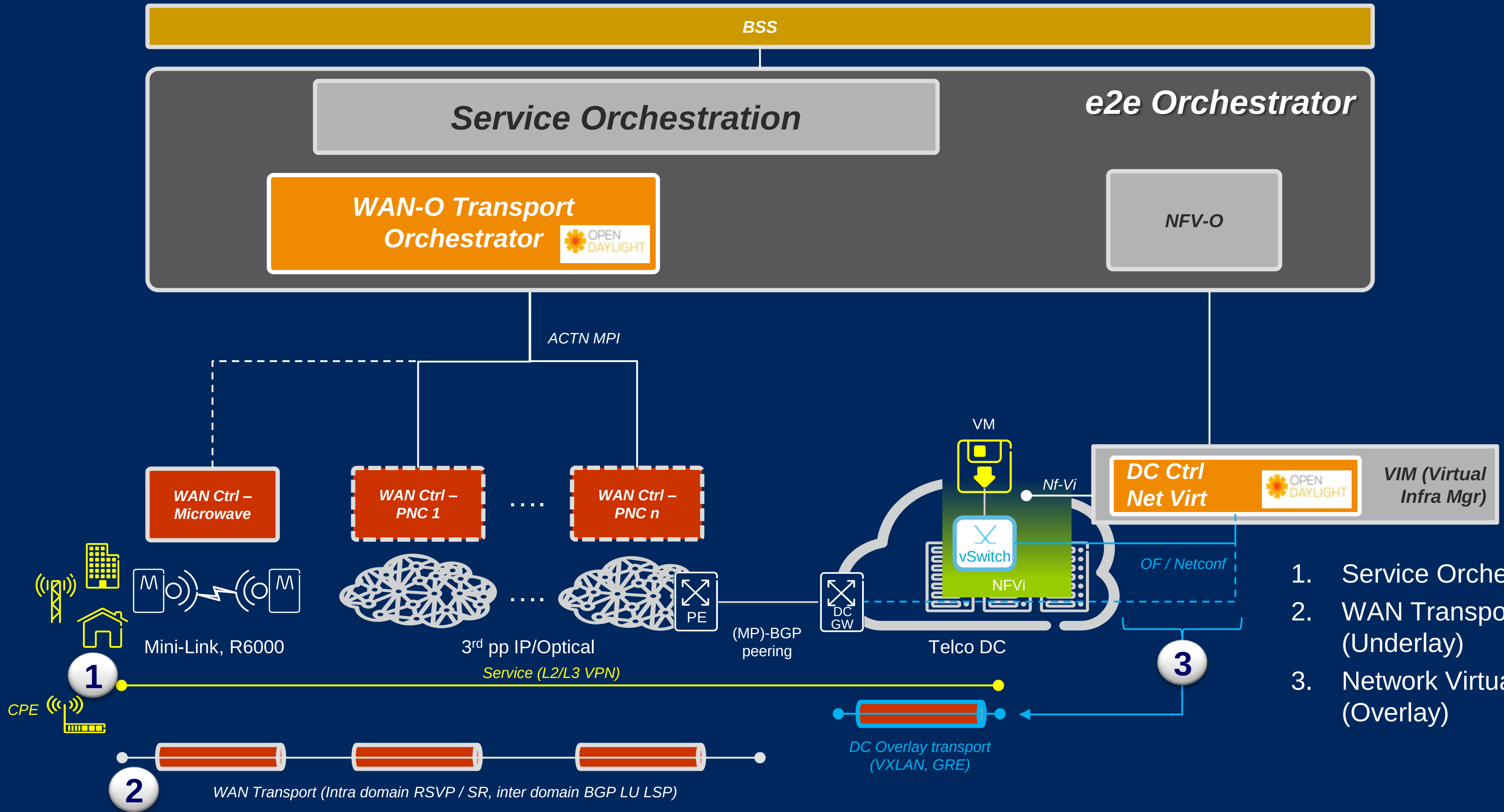
Connectivity services



1. Service Orchestration
2. WAN Transport SDN (Underlay)

END to END service orchestration

VNF services



1. Service Orchestration
2. WAN Transport SDN (Underlay)
3. Network Virtualization (Overlay)

OpenDaylight: Getting Involved

Avenues for getting involved

- OpenDaylight Wiki: https://wiki.opendaylight.org
- Mailing Lists:
  - Central / Cross Project: https://wiki.opendaylight.org/view/Mailing_Lists
  - Complete List including individual projects: https://lists.opendaylight.org/mailman/listinfo
- Chat with developers via IRC: https://wiki.opendaylight.org/view/IRC
- Meetings:
  - Technical Steering Committee: https://wiki.opendaylight.org/view/TSC:Meeting
  - Technical Work Stream: https://wiki.opendaylight.org/view/Tech_Work_Stream:Main
  - Complete List including individual projects: https://wiki.opendaylight.org/view/Meetings

Areas to getting involved in

- OpenDaylight Documentation Project
- Project of your interest
  - https://wiki.opendaylight.org/view/Project_list
  - Code Reviews
  - Bug Fixing
- MD-SAL & Clustering (Distributed Systems)
  - Experts
  - Enthusiasts: Improve your skills in these hot & in-demand area
- Scale & Performance
- Testing
- Architecture Improvements
  - Example: Scalable and Robust Data Replication using etcd.

Acknowledgements

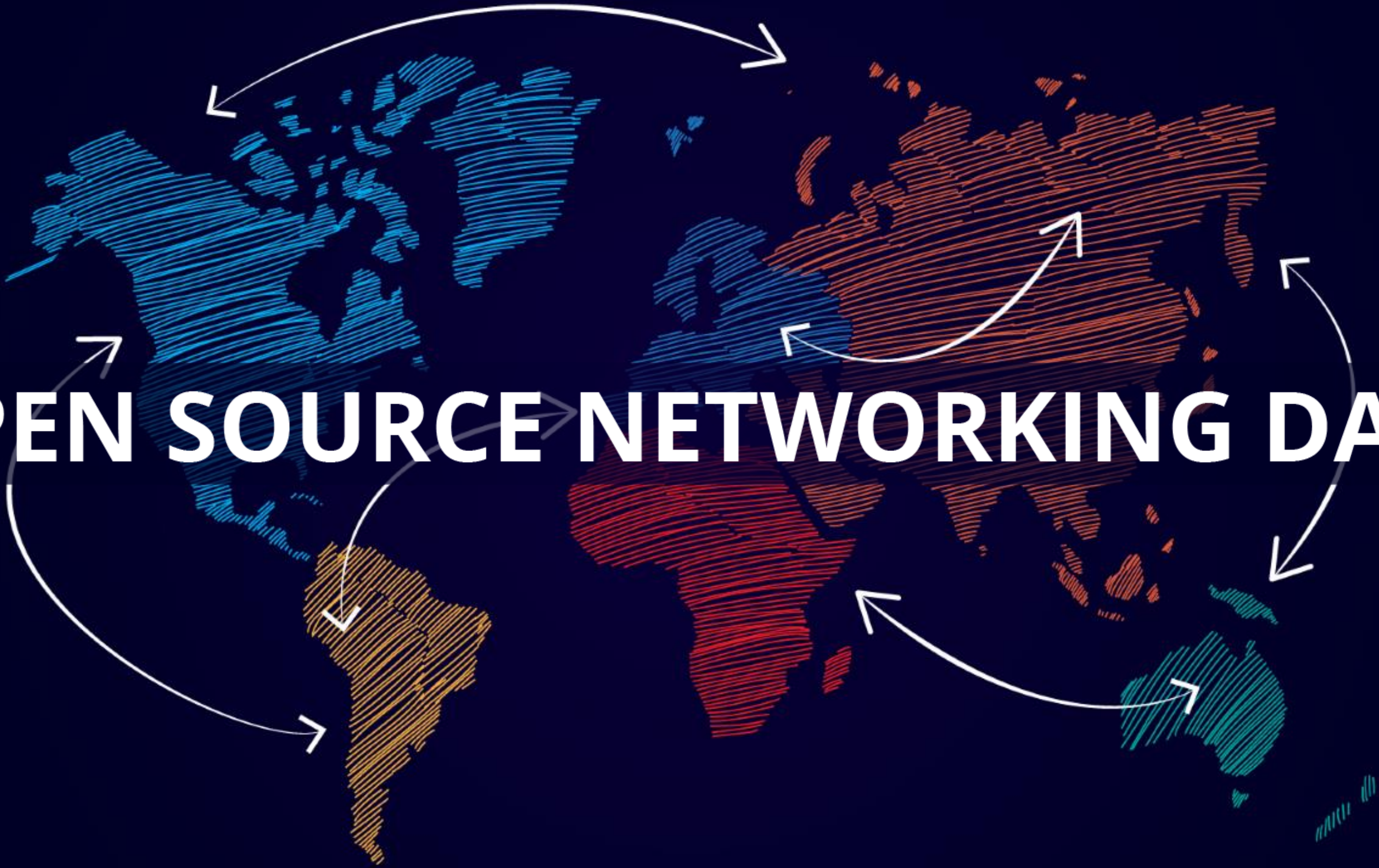
Contributors to slides

- Antonio De Gregorio
- Colin Dixon
- Daniele Ceccarelli
- Dayavanti Kamath
- Francois Lemarchand
- Frederick Kautz

- Jan Medved
- Luis Gomez
- Prem Sankar Gopanan
- Scott Melton
- Srini Seetharaman
- YuLing Chen

- Reference
  - https://github.com/BRCDcomm/BVC/wiki/MD-SAL

Q & A



OPEN SOURCE NETWORKING DAYS