Open Networking Foundation

# ODTN
# An Open Controller for the Disaggregated Optical Network

Andrea Campanella
andrea@opennetworking.org

An Operator Led Consortium

# Outline

- Clear ask from Operators
- ONOS as a Platform
- Incremental steps
  - Phase 1.0
  - Phase 1.5
  - Phase 2.0
- Trials
- Next Steps
- Takeaways

# Clear ask from operators

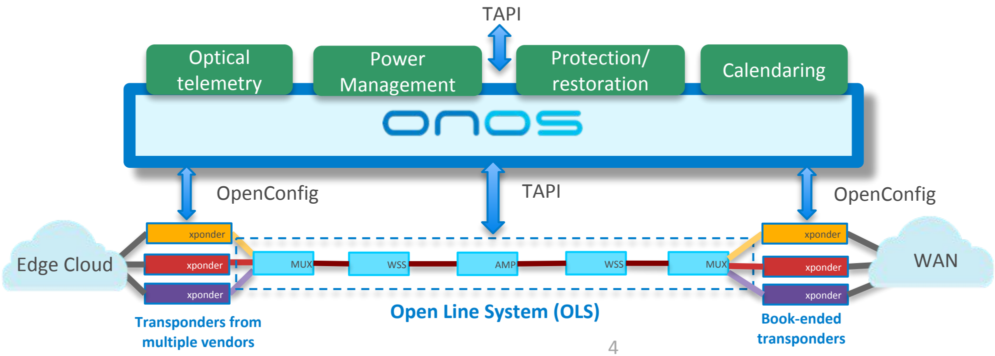
**Open Source Data Center Interconnect (DCI) Solution**

1. **Open and Standard APIs** to be vendor neutral and modular.
2. **Rapid cycle of innovations** can happen in terminal equipment (Transponders)
3. Clear separation of the behavior of the transponder and the line system (OLS)
4. Enable **Services** to be rapidly created, prototyped, tested
5. Support OLS that transport any kind of signal (**Alien Wavelengths**)
6. Modular and **production ready platform**
7. CI/CD pipeline for DevOps environment

# Disaggregating Transponders from OLS

## Business Benefits

- Rapid adoption of innovations in terminal equipment
  - Enable vendors to innovate: speed, reach, QoT, …
  - Let operators reap benefits through simple bookending
- Rapid introduction of new services in production network
  - Realize DevOps model through SDN-enabled optical network
  - Build CI/CD pipeline between operator, vendors, and open source software stack



# Why ONOS ?

- **Modular Architecture**
  - Support for multiple protocols
  - Support for multiple device models
  - ease of extensibility
- **Resiliency** in case of failures
  - Multi instance
  - Device Mastership handling
- **Dynamic Configuration Subsystem (DCS)**
- **Performance**
- **Production ready** and proven code

Open Network Operating System

# Southbound Protocols

**ODTN Southbound protocol needs**

- NETCONF + YANG → Yang tools and Dynamic Configuration Subsystem
- REST and RESTCONF
- gRPC → gNMI

**Support Current Networks but also look ahead to future deployments**

# Drivers

- Device specific driver
  - collection of behaviors
  - on-demand activation
  - encapsulate device specific logic and code
    - ports,controller,flowrule,power…
    - models

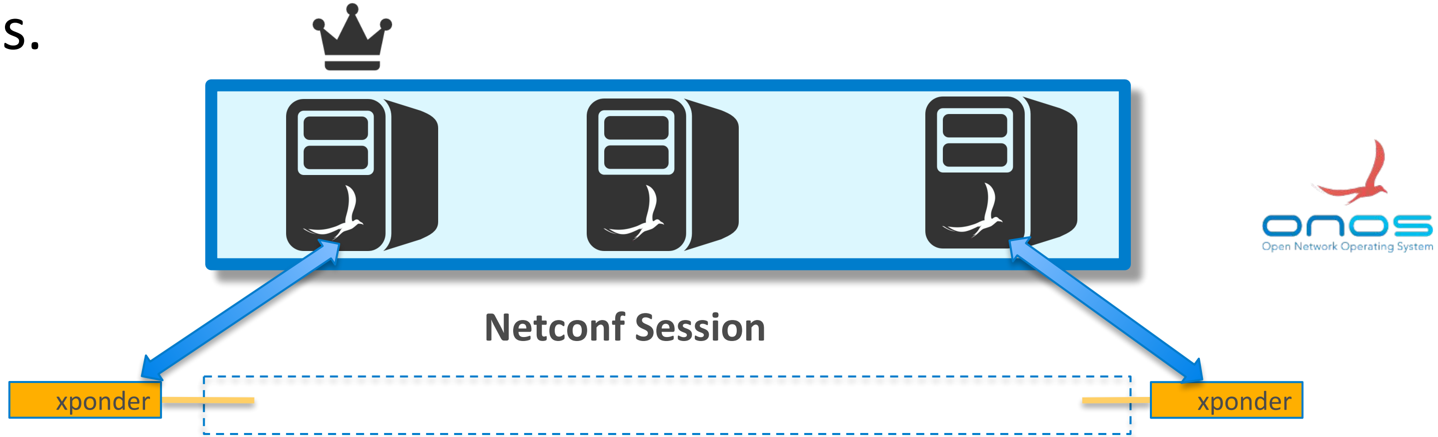
**<company>-drivers.xml e.g microsemi**

```
<driver name="microsemi-netconf"
extends="netconf" manufacturer="Microsemi"
  hwVersion="EA1000">
        <behaviour api=InterfacePath
                  impl=ImpementationPath />
</driver>
```

**Integrate different devices with different Yang models with no change to the ONOS core or Northbound API**

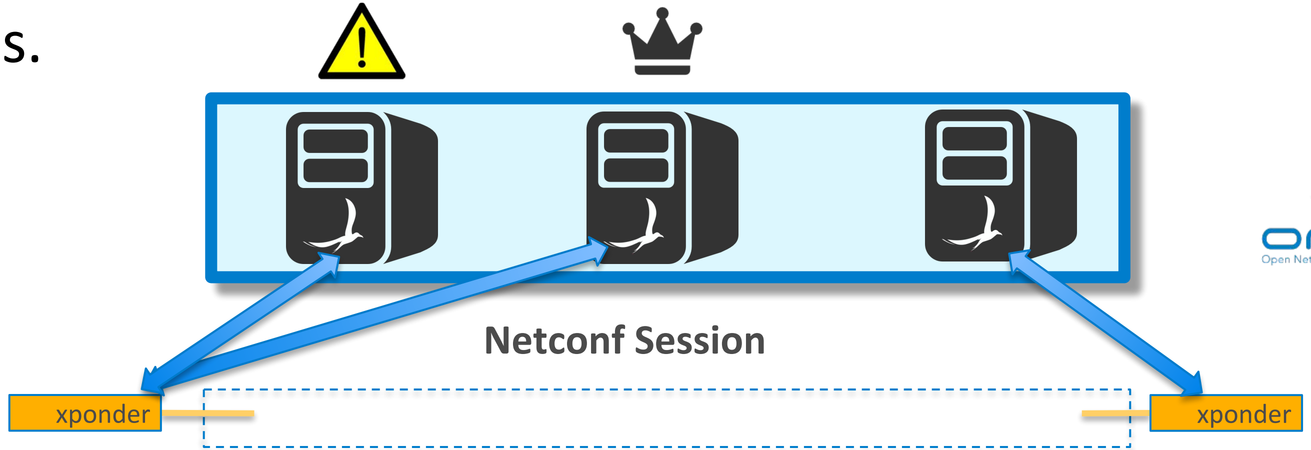
# Mastership handling 1/2

Handle ONOS instance failure even with mastership un-aware devices.



# Mastership handling 2/2

Handle ONOS instance failure even with mastership un-aware devices.

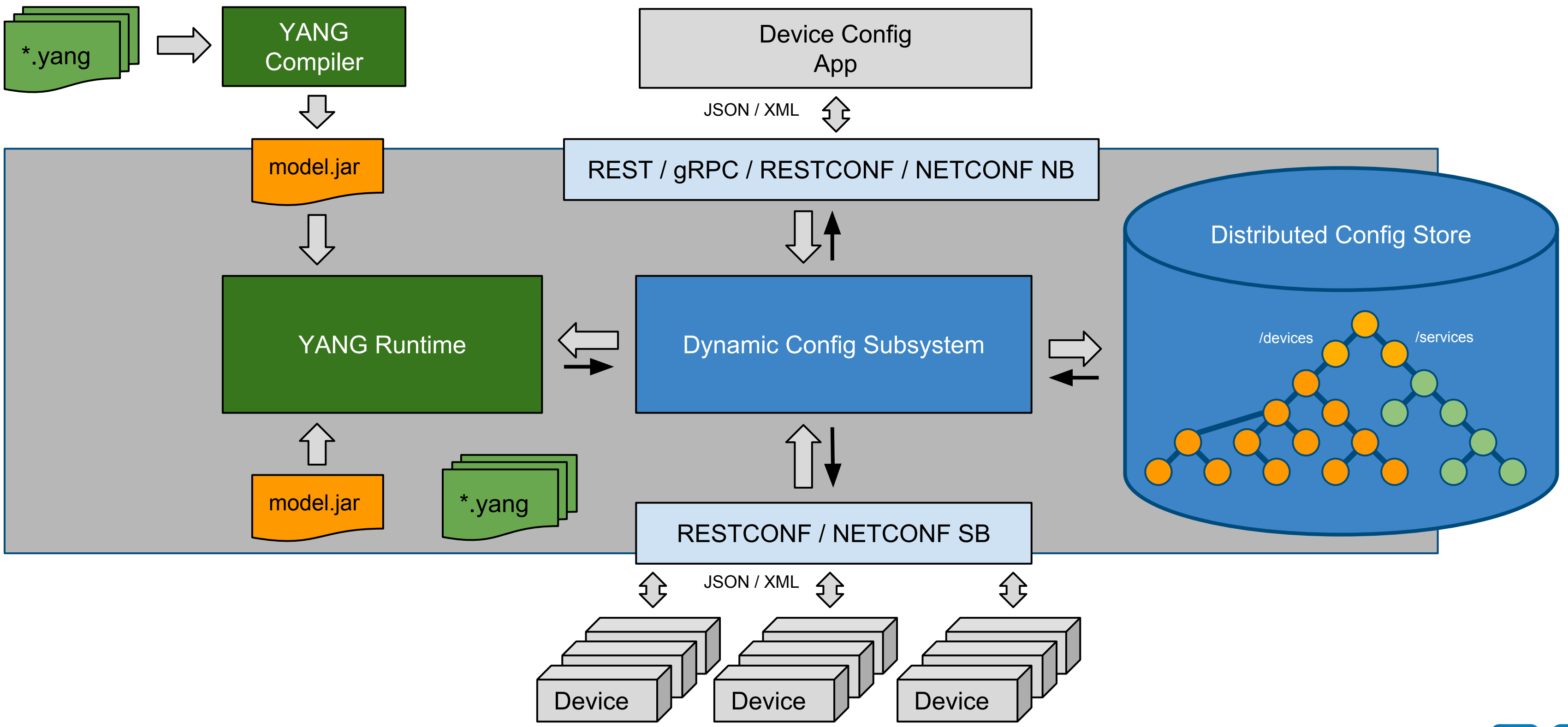



**No downtime of device control and management**

# Dynamic Configuration Subsystem(DCS)

- YANG Compiler
  - processes YANG models to understand structure of data
  - generates model APIs and code that carries and conveys data
- YANG Runtime
  - transforms data between external and internal representations
- Protocol Adapters
  - ingest & emit data using various protocols, NETCONF, gRPC
- Information Store
  - persist and distribute data throughout the cluster of nodes
  - retain NB-to-SB edicts and SB-to-NB operational state

# Major DCS System Components



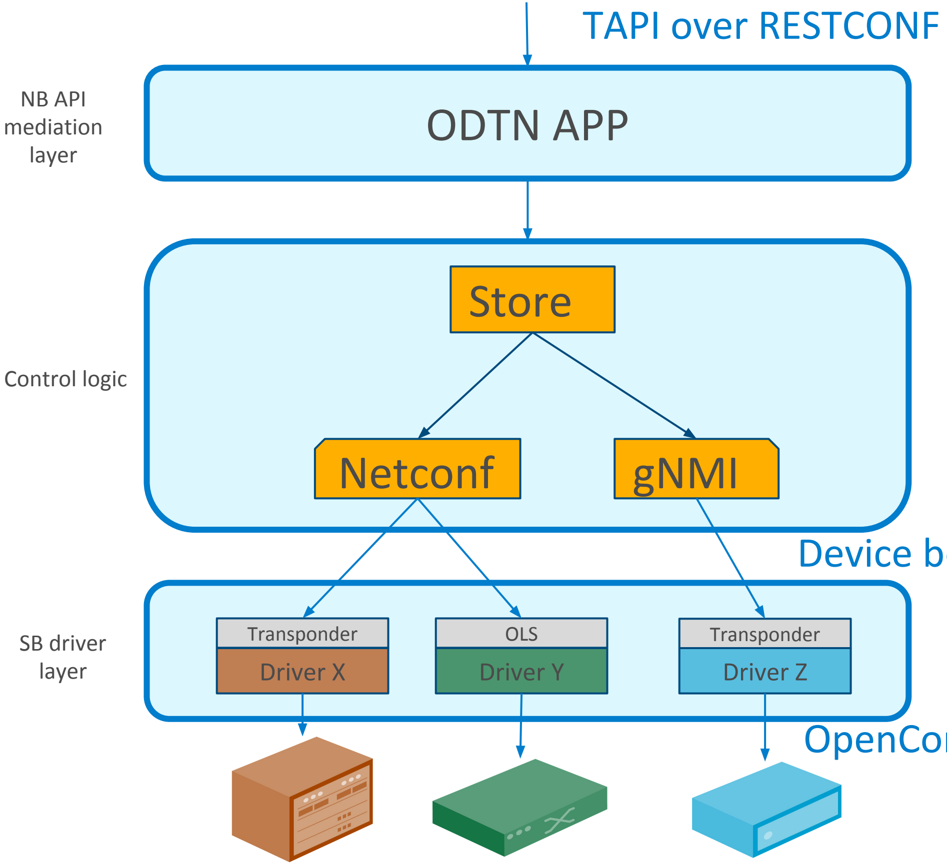
# Incremental Approach

ODTN gets developed one step at a time through:

- definition of use-case
- choice of common API(s) to achieve given use-case
- implementation in ONOS
- test, debug and trials

**Each phase builds on top of the previous one with new and further enhancements**

# High Level Design




Protocol handling

Understand semantics of request

Solves constrained path

Manage resources

Generate device control and configuration

Device behavior calls

Map imposed semantics to commands that device understands

OpenConfig v.nn over NETCONF or TAPI over REST/Resconf

# ODTN Phase 1.0

# ODTN Phase 1.0 - Use Case and APIs

**Use Case**

- Point to point connection made of 2 transponders and an optional Open Line system
- Directly connected transponders, or OLS configured out-of-band
- Enable cross-connection between line-side and client side ports of the transponder

**APIs**

- Northbound Transport API (TAPI) through RESTCONF
- Transponders configuration: OpenConfig models over NETCONF

# Why OpenConfig for TX

- **Well know API**
- **Supported** already by many vendors
- **Proper abstraction** model for transponder devices capabilities and information
- Defines capabilities at **correct level for programmability** but also abstraction from physical details
- Capability and Flexibility to **support vendor specific features**
- Can represent both **multi-layer** w/ and w/o OTN
- **Extensible and Open Source**

# Why TAPI for ONOS Northbound and OLS ?

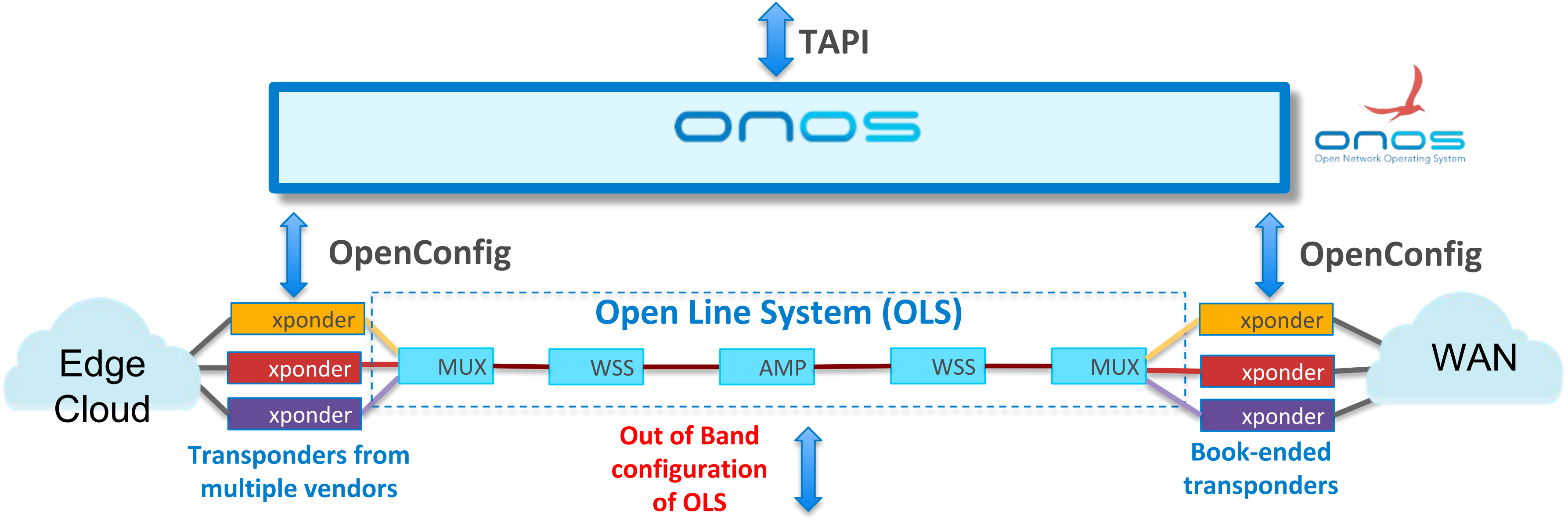
- **Well know API**
- **Extensible and Open Source**
- **Tested and deployed** (See Interop Testing)
- **Proper abstraction** for high level optical domain programming
- Can represent both **multi-layer** end to end provisioning with optical parameters
- Great community of vendors and Service Providers

# ODTN Phase 1.0 - Topology

Transponders on either side of one p2p connection must be of same vendor

OLS, if present, is configured out of band to carry alien wavelengths across

**Transponders** → Infinera XT3300, NOKIA 1830PSI-2T, NEC, Edge-core CASSINI

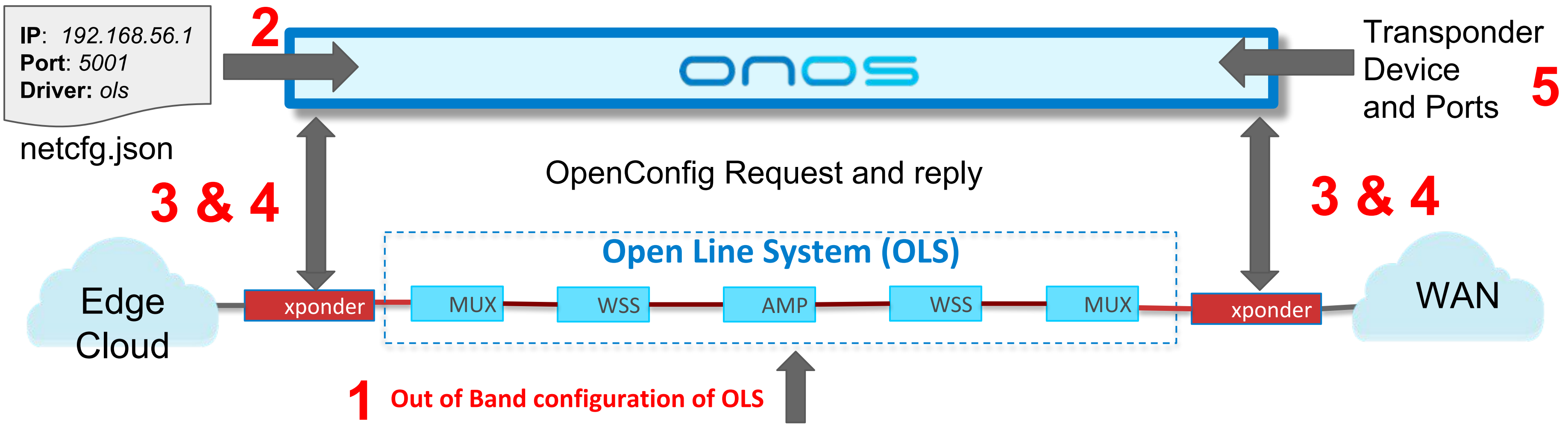


# ODTN Phase 1.0 - Implementation

- Auto-generated **RESTCONF ONOS northbound based on TAPI** yang models through DCS
- **ODTN Application** for end to end control with TAPI model integration
- Implementation of an **Openconfig ONOS driver** supporting standard version of Openconfig
- **Specific device drivers** were developed when needed (Infinera XT-3300) due to deviances from the model

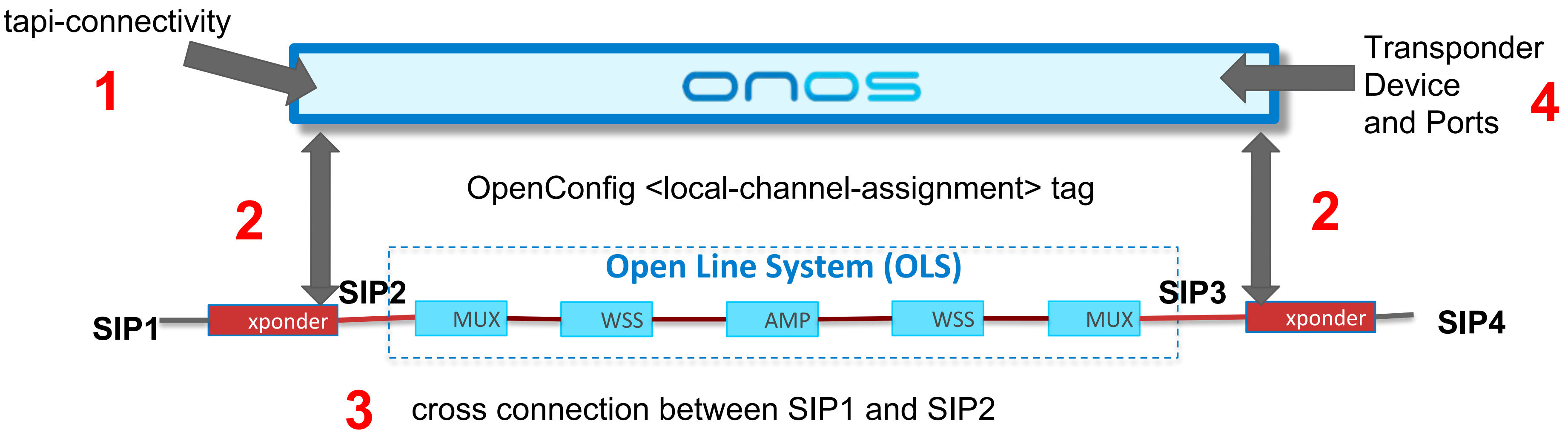
# ODTN Phase 1.0 - Transponder discovery

1. Pre-Provision of OLS
2. OSS/BSS or Operator send Json with OLS endpoint to ONOS
3. ONOS Initial reach out and OpenConfig request topology request
4. Transponder returns device information and ports
5. ONOS exposes ports it as Service Interface Points (SIPs)
6. ONOS Stores Transponders device and ports in distributed store

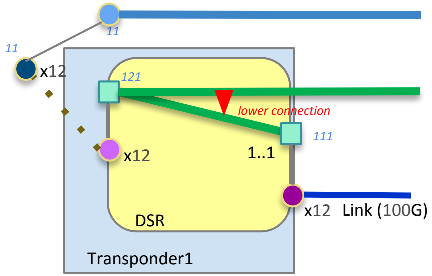


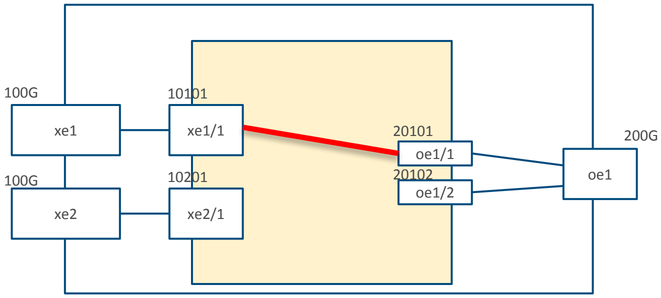
# ODTN Phase 1.0 - Transponder provisioning

1. OSS/BSS send TAPI connectivity Request to ONOS with two SIPs (SIP1, SIP4)
2. ONOS computes OpenConfig Payload to create cross-connect in each device (e.g. SIP1-SIP2) and sends it to devices
3. Transponder creates cross connection
4. ONOS Stores configuration of Transponders and can return it via TAPI NB



# Mapping from TAPI to OpenConfig




tapi-sample-step2-intermediate.xml

```
<connection xmlns="urn:onf:otcc:yang:tapi-connectivity">
  <uuid>00000000-0000-3000-0001-111000000000</uuid>
  <connection-end-point>
    <topology-id>...-100000000000</topology-id>
    <node-id>...-100000000000</node-id>
    <owned-node-edge-point-id>...-121000000000</owned-node-edge-point-id>
    <connection-end-point-id>...-121000000000</connection-end-point-id>
  </connection-end-point>
  <connection-end-point>
    <topology-id>...-100000000000</topology-id>
    <node-id>...-100000000000</node-id>
    <owned-node-edge-point-id>...-111000000000</owned-node-edge-point-id>
    <connection-end-point-id>...-111000000000</connection-end-point-id>
  </connection-end-point>
  <layer-protocol-name>DSR</layer-protocol-name>
</connection>
```

*client side*

*line side*

sbi-openconfig-sample-infinera.xml

```
<logical-channels>
  <channel>
    <logical-channel-assignments>
      <assignment>
        <index>10101</index>
        <config>
          <index>10101</index>
          <assignment-type>LOGICAL_CHANNEL</assignment-type>
          <logical-channel>20101</logical-channel>
          <allocation>100.0</allocation>
        </config>
      </assignment>
    </logical-channel-assignments>
  </channel>
```

# CASSINI white-box TX Integration

ODTN

OpenConfig

OcNOS

| TAI | TAI |
|---|---|
| libtai.so (for vendor A) | libtai.so (for vendor B) |
| Transponder A | Transponder B |

onos Open Network Operating System

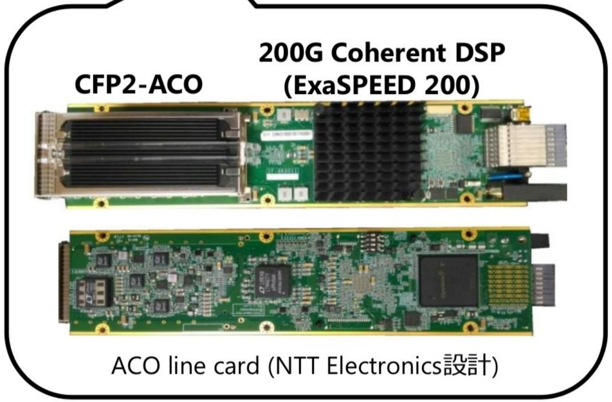
OPENCONFIG

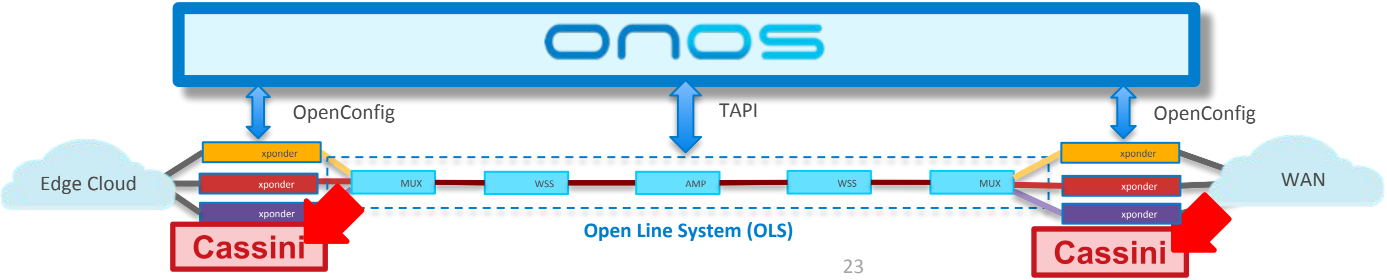
ipinfusion

**Transponder Abstraction Interface**

Edge-core NETWORKS

NTT

Open Networking Foundation

# ODTN Phase 1.5

# ODTN Phase 1.5 - Use Case and APIs

## Use Case

- Point to point connection made of 2 transponders and an Open Line system
- Enable end to end path provisioning with Transponder and OLS control

## APIs

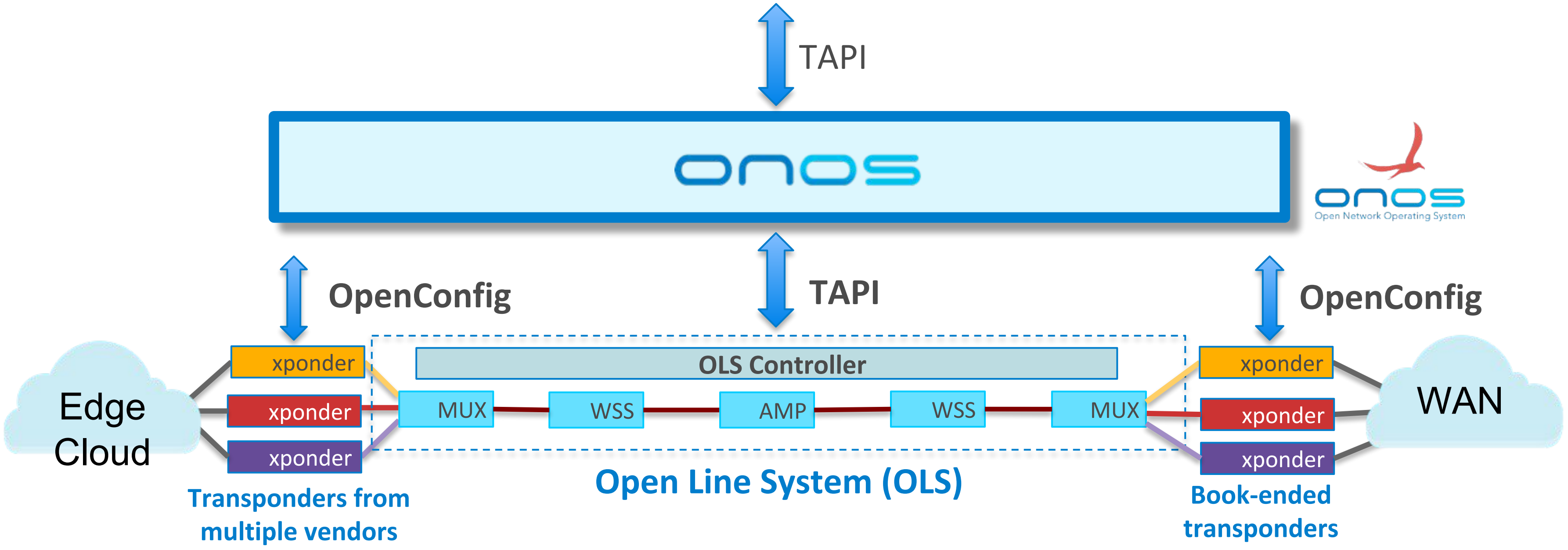
- Northbound: Transport API (TAPI) through RESTCONF
- Transponders configuration: OpenConfig models over NETCONF
- **OLS configuration: T-API 2.1 models over REST**

# ODTN Phase 1.5 - Topology

Same as Phase 1.0 but OLS discovered and controlled by ONOS

Open Line System is exposed as a single device (big-switch)

**OLS Vendors** → **ADVA**, Coriant/Infinera, Nokia, Juniper



# ODTN Phase 1.5 - Implementation

**Done:**

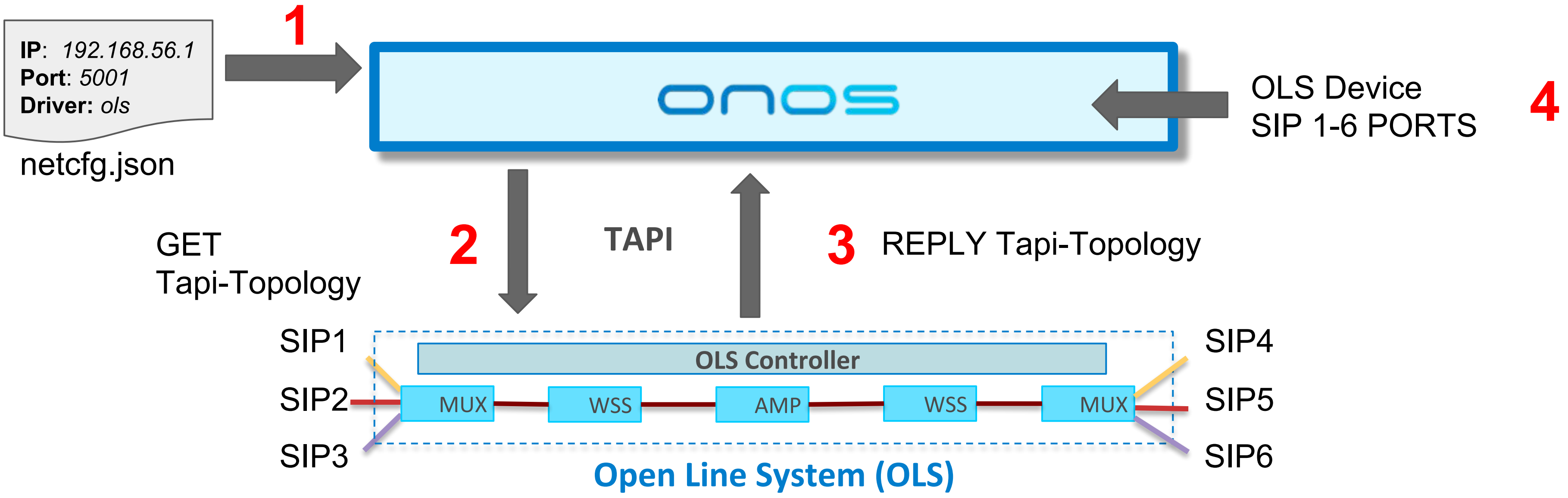
- **Augmented transponder drivers** with Line Side port configuration for wavelength trough OpenConfig
- Extend Northbound **TAPI** to **2.1**
- Driver for **discovery of OLS device and Ports** as SIPs (Service Interface Points) through TAPI 2.1 on Southbound (Working with ADVA OLS)

**In Progress:**

- **connectivity request for OLS** through TAPI in SB
- **Power** negotiation and configuration
- **Other OLS integration**

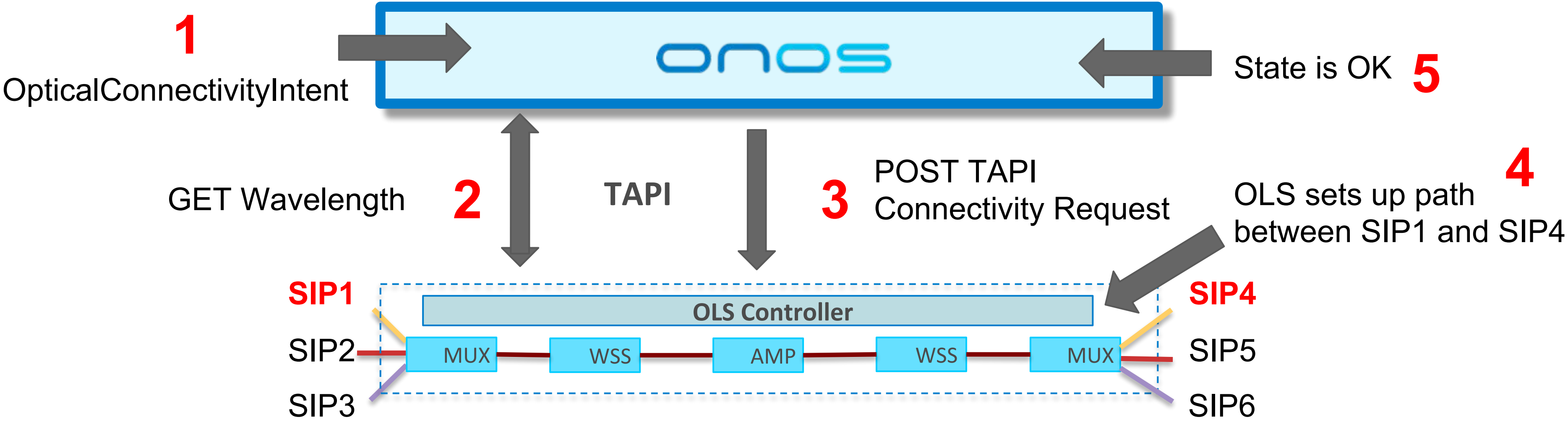
# ODTN Phase 1.5 - OLS Discovery

1. OSS/BSS or Operator send Json with OLS endpoint to ONOS
2. ONOS Initial reach out and TAPI topology request
3. OLS returns basic device information and Service Interface Points (SIPs)
4. ONOS Stores device and SIPs as Ports in distributed store



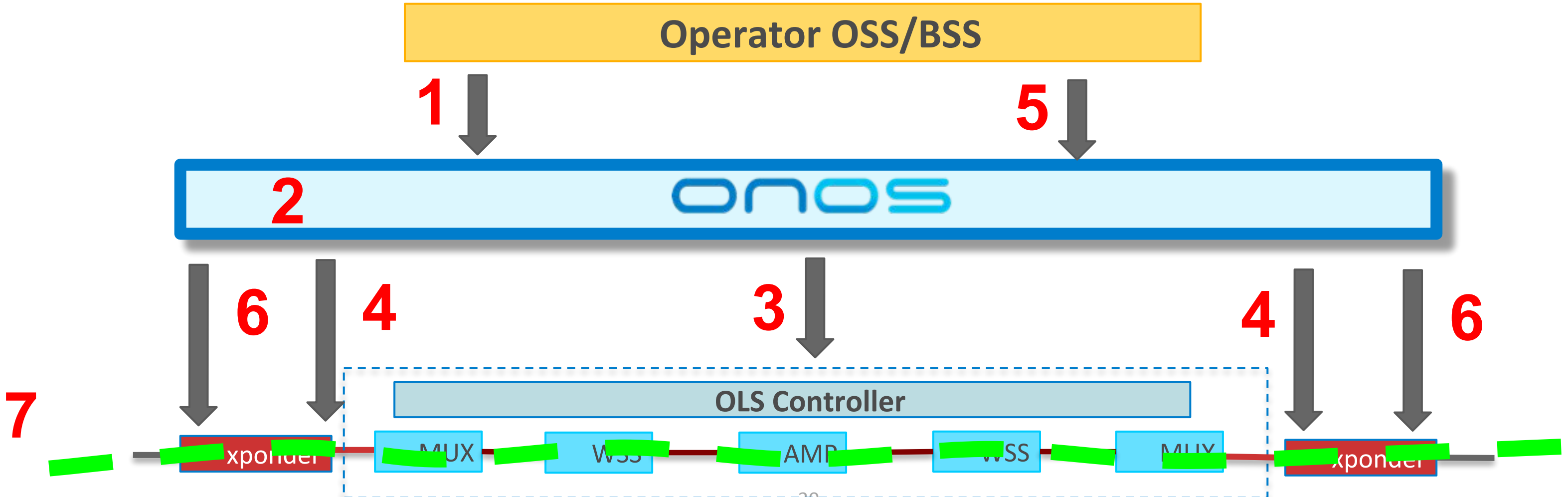
# ODTN Phase 1.5 - OLS Provisioning

1. ONOS creates an Optical Connectivity Intent and Identifies two SIPs (1,4) as ports required to pass through the OLS
2. (Optional) wavelength request on given ports to OLS
3. TAPI Connectivity request between SIP 1 and 4 on wavelength (if needed)
4. OLS sets up internal path and returns OK
5. Intent is installed and ONOS know of the OLS properly provisioned



# ODTN Phase 1.5 - end to end provisioning

1. OSS/BSS requests optical layer provisioning through TAPI
2. ONOS creates OpticalConnectivityIntent
3. OLS is provisioned through TAPI
4. Line side of the transponder is provisioned through OpenConfig
5. OSS/BSS request end to end L3 connectivity
6. Cross-connect line side to client side is setup through OpenConfig
7. End to end path is provisioned



# Lab Trial Plans

Telefonica ↔ NOKIA

NTT Communications ↔ infinera

NTT Communications ↔ Edge-corE NETWORKS, ipinfusion™

**Transponders**: NOKIA, infinera, Edge-corE NETWORKS, ipinfusion™

**Open Line System**: ADVA™ Optical Networking

TBD: ADVA, INFINERA, OTHERS ?



onf

Open Networking Foundation

# ODTN Phase 2.0

# ODTN Phase 2.0 - Use Case and APIs

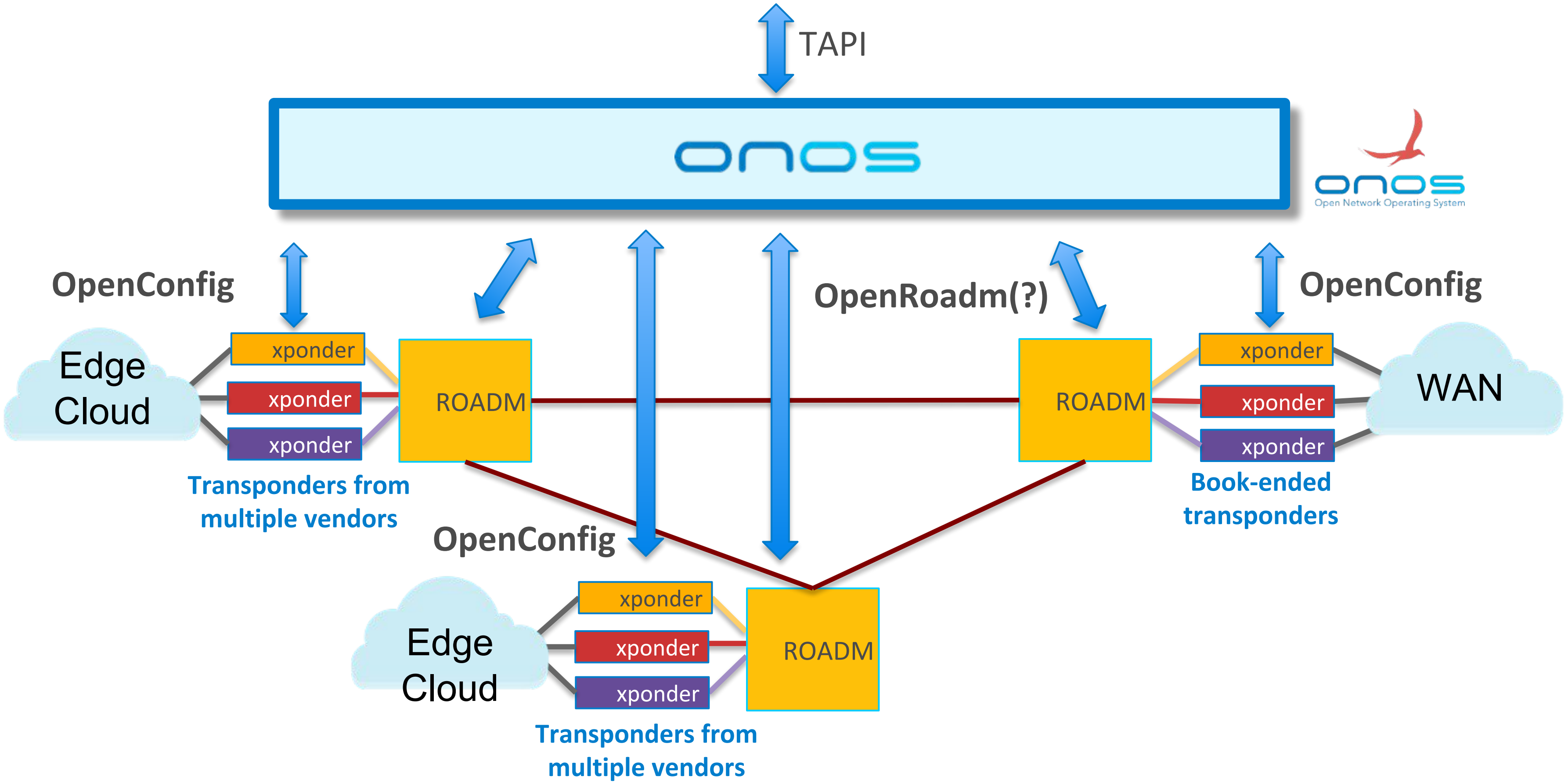
## Use Case

- **Mesh ROADM network** made of N ROADMS and N transponders (N>=2)
- Enable end to end path provisioning with Transponder and ROADM control

## APIs

- Northbound: Transport API (TAPI) through RESTCONF
- Transponders configuration: OpenConfig models over NETCONF
- **ROADM configuration: openROADM (?), others (?)**

# ODTN Phase 2.0



# Phase 2.0 Lab Trial Plans

**Transponders** — **Coriant ?**

**Open Line System** — **Lumentum**

# ODTN Phase 1.5 - Implementation

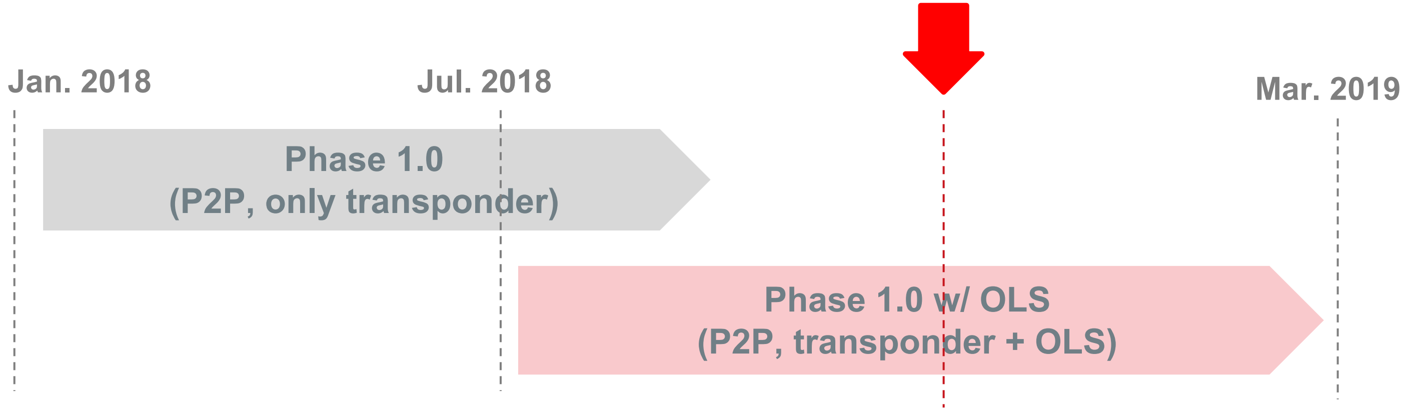
- **leveraging existing ROADM effort in ONOS**
- **drivers for different roadms**
- **openRoadm API**

Open Networking Foundation

# Next Steps

# Next Steps

- Complete OLS Integration
- Lab Trial phase 1.5 solution
- Expand Dynamic Config features (Dry-run, startup config, backup)
- Multi vendor Transponder and OLS Trial
- Code and platform hardening.
- Define scope and API for phase 2.0



# Takeaways

# Takeaways

- ODTN is the **first (and only)** project to build **open source software stack for** control and management of **optical networks**
- ODTN Uses **standard and open device APIS** (OpenConfig for Transponders, TAPI for OLS)
- ODTN uses **TAPI** as a standard and open API on the northbound
- ODTN leverages architecture, performance e scalability of **ONOS**
- ODTN integrates a **wide variety of vendors** for network equipment.
- **Incremental** approach towards production readiness
- **Lab trials** with major operators → **feedback loop** of requirements and enhancements

# Takeaways

## Great Community, Thanks you!

## Still lots to do, come and join us!

## odtn@opennetworking.org

# Useful Info

ODTN Wiki: https://wiki.onosproject.org/display/ODTN/ODTN

Technical Weekly Meeting: **Every Tuesday at 8 AM PST**

# Questions ?

andrea@opennetworking.org

# Phase 1.0

Phase 1.0 Blogpost

https://www.opennetworking.org/news-and-events/blog/odtn_phase1_results/

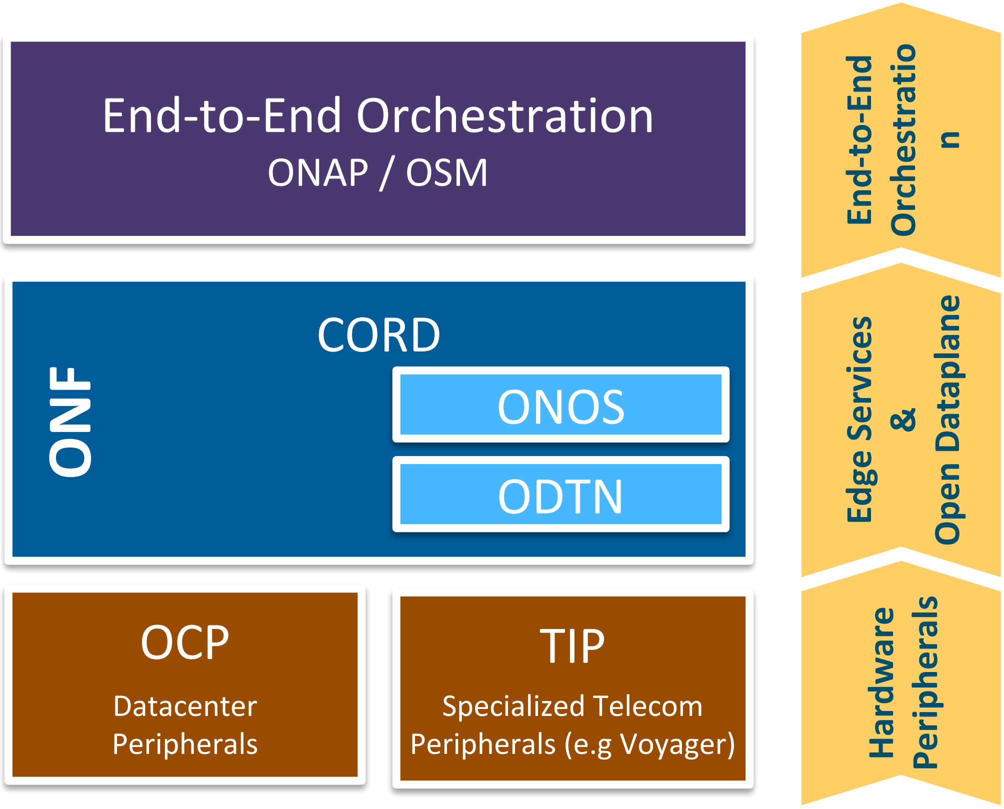
Phase 1.0 Demo with NTT and Infinera

https://wiki.onosproject.org/pages/viewpage.action?pageId=23335851

Phase 1.0 Demo with Telefonica and NOKIA

https://wiki.onosproject.org/pages/viewpage.action?pageId=27590874 https://www.opennetworking.org

# Where ODTN Fits into Open Source Ecosystem




**ODTN is the only optical transport open source project**

**First project to build open source software stack for control and management of optical networks**

This ecosystem is poised to deliver robust solutions over time, from white box peripherals to orchestrated end-to-end solutions

# Relationship to Other Standards & Optical Organizations

- ONF Transport API
  - Wide industry support and growing acceptance
  - ODTN using TAPI for service provisioning, topology, …
- OpenConfig
  - Develops common data models for network management
  - ODTN using OpenConfig models for transponders, MUX, WSS, AMP
- Telecom Infra Project (TIP)
  - Open Optical Packet Transport group
  - ODTN to consume TIP's network planning tools and open APIs
  - ODTN software stack can be used with TIP hardware building blocks (e.g. CASSINI)
- OpenROADM MSA
  - Develops open models for optical devices, networks and services
  - Focus on transponder compatibility (eliminating need for bookending)
  - Models may be incorporated if ODTN community puts focus on data plane interoperability

**ODTN is the only optical transport open source project**

**First project to build open source software stack for control and management of optical networks**

# Phase 3: Full Disaggregated ROADM with Open APIs

## Goal

- Integrate ONOS and disaggregated optical components by using open APIs
- Verify the reference implementation that works certainly for disaggregated ROADM use case
- Identify problems to be solved toward production

## Device Components

- Transponder, WSS, AMP, AOS, etc. (details TBD)

## Term

- Q4 2019 (?)