ONOS with YANG & P4 Runtime

毛健炜 **Jianwei Mao**

ONOS Ambassador, China

Beijing University of Posts and Telecommunications (BUPT)

Future Network Laboratory (FNL)

MaoJianwei2012@126.com

Agenda

- ONOS Introduction & Architecture
- ONOS Newest Feature
  - YANG & Dynamic Configuration
  - P4 Runtime support
- ONOS Official Community & 中文社区

Agenda

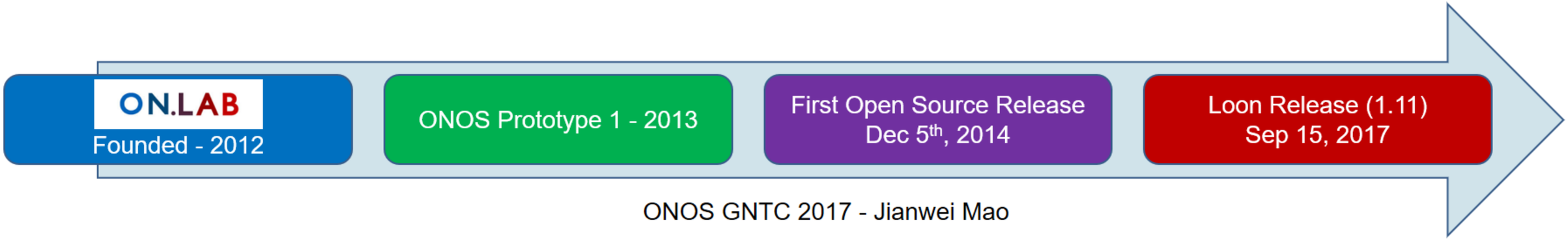
- ONOS Introduction & Architecture
- ONOS Newest Feature
  - **YANG & Dynamic Configuration**
  - **P4 Runtime support**
- ONOS Official Community & 中文社区

ONOS Overview

- ONOS: Open Network Operating System
  - Open source SDN network operating system
  - Objective: enable **Service Providers** to build real SDN/NFV solutions

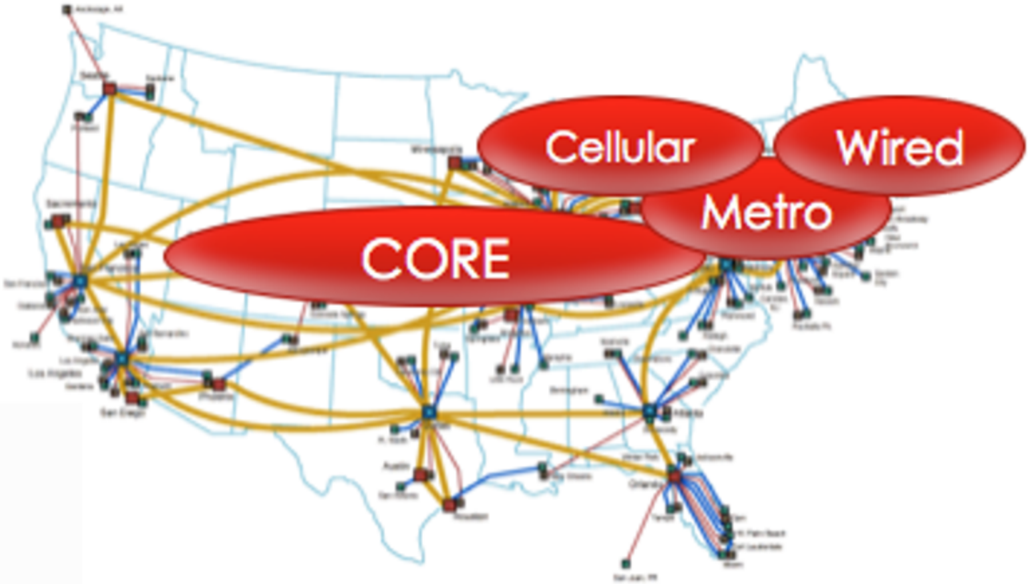
ONOS Overview

- ONOS: Open Network Operating System
  - Open source SDN network operating system
  - Objective: enable **Service Providers** to build real SDN/NFV solutions



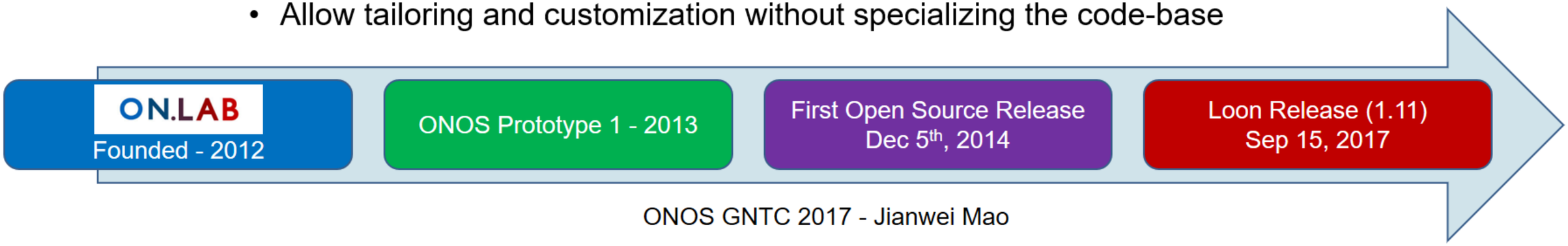
Service Provider Networks

- WAN core backbone
  - Multi-Protocol Label Switching (MPLS) with Traffic Engineering (TE)
  - *200-500 routers, 5-10K ports*
- Metro Networks
  - Metro cores for access networks
  - *10-50K routers, 2-3M ports*
- Cellular Access Networks
  - LTE for a metro area
  - *20-100K devices, 100K-100M ports*
- Wired access / aggregation
  - Access network for homes; DSL/Cable
  - *10-50K devices, 100K-1M ports*



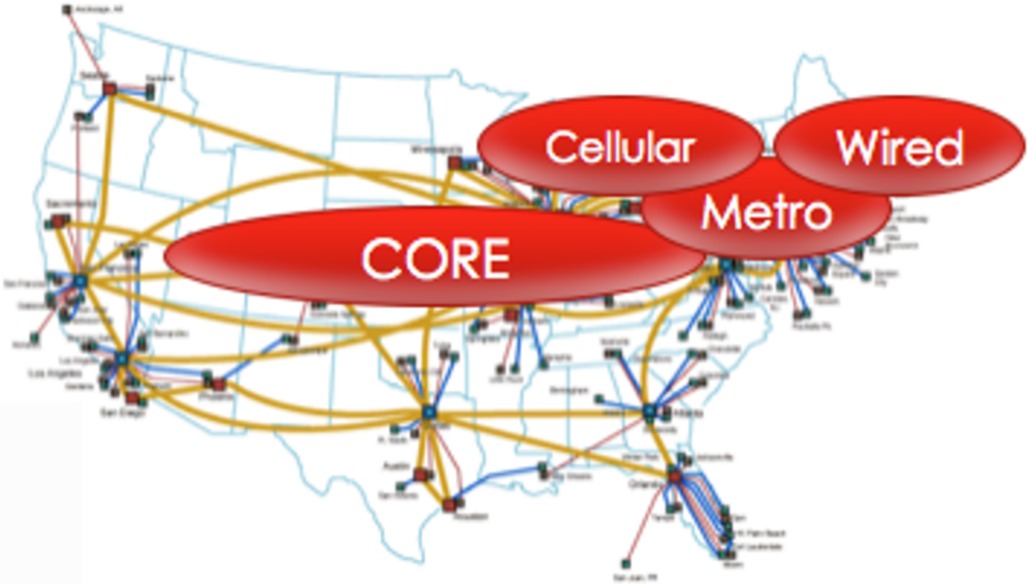
ONOS Overview

- ONOS: Open Network Operating System
  - Open source SDN network operating system
  - Objective: enable **Service Providers** to build real SDN/NFV solutions
  - Design Tenets:
    - **High -Availability, -Scalability and -Performance**
      - Required to sustain demands of service provider & enterprise networks
    - **Strong abstractions and simplicity**
      - Required for development of apps and solutions
    - **Protocol and device behavior independence**
      - Avoid contouring and deformation due to protocol specifics
    - **Separation of concerns and modularity**
      - Allow tailoring and customization without specializing the code-base



Service Provider Networks

- WAN core backbone
  - Multi-Protocol Label Switching (MPLS) with Traffic Engineering (TE)
  - *200-500 routers, 5-10K ports*
- Metro Networks
  - Metro cores for access networks
  - *10-50K routers, 2-3M ports*
- Cellular Access Networks
  - LTE for a metro area
  - *20-100K devices, 100K-100M ports*
- Wired access / aggregation
  - Access network for homes; DSL/Cable
  - *10-50K devices, 100K-1M ports*

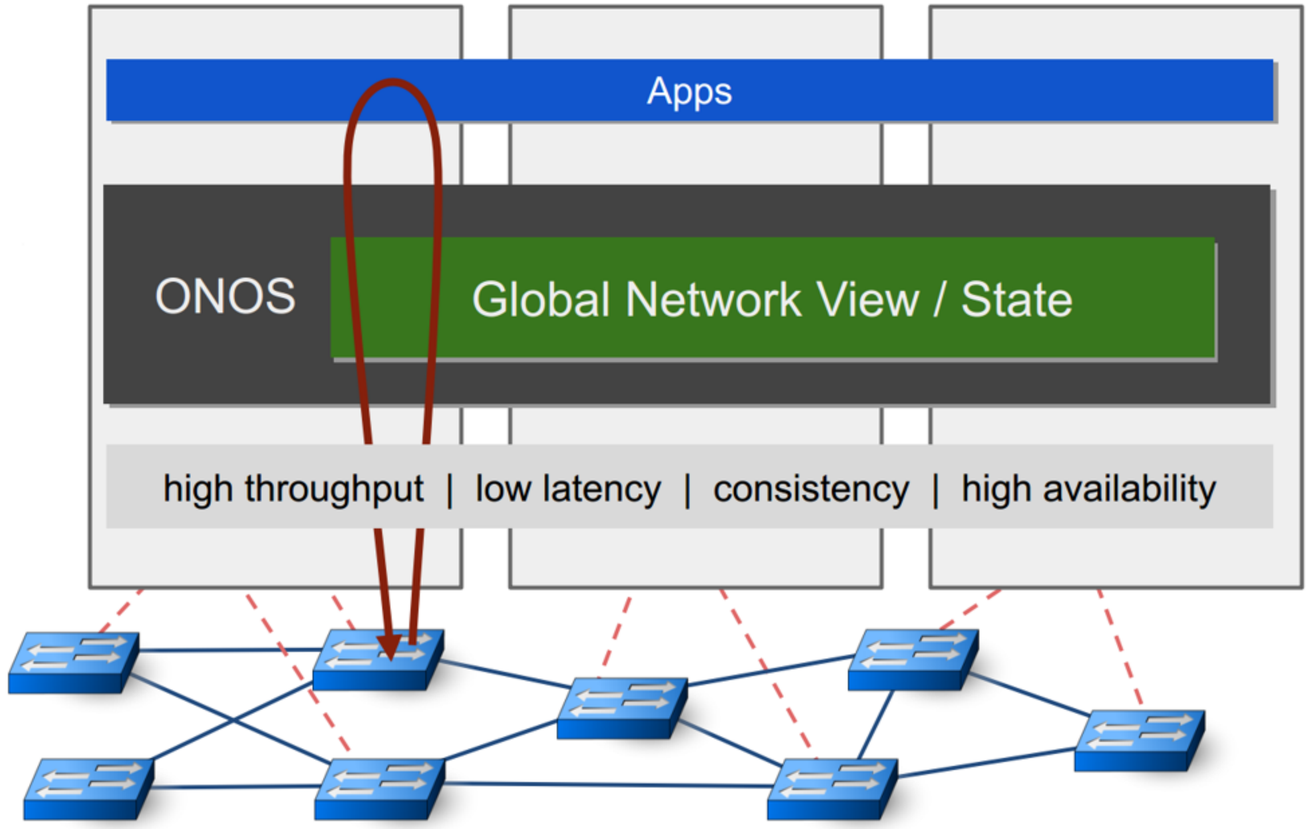


ONOS Architecture (1/6)

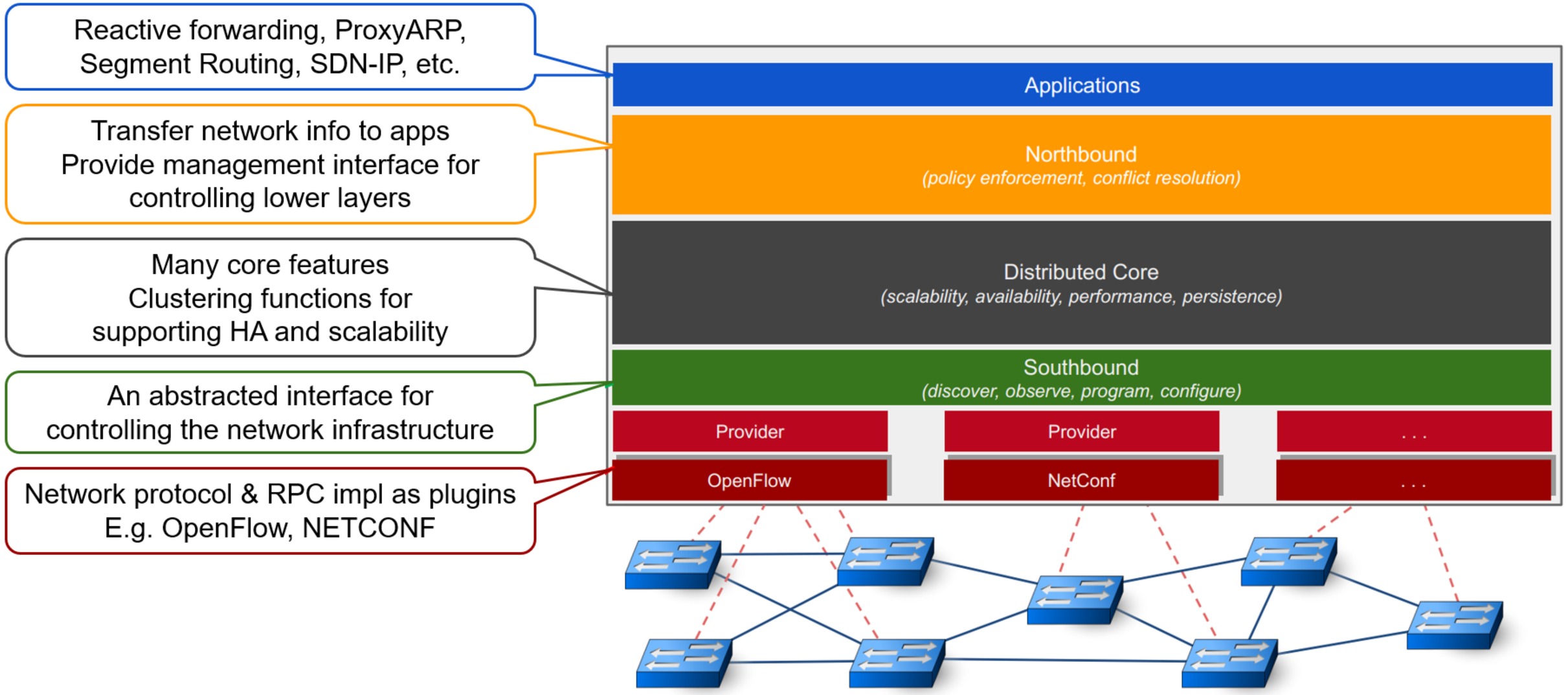
Key Performance Requirements

- High Throughput
  - 500K - 1M paths setups/s
  - 3 - 6M network state operations/s
- High Volume
  - 500GB - 1TB of network state data

Challenging !

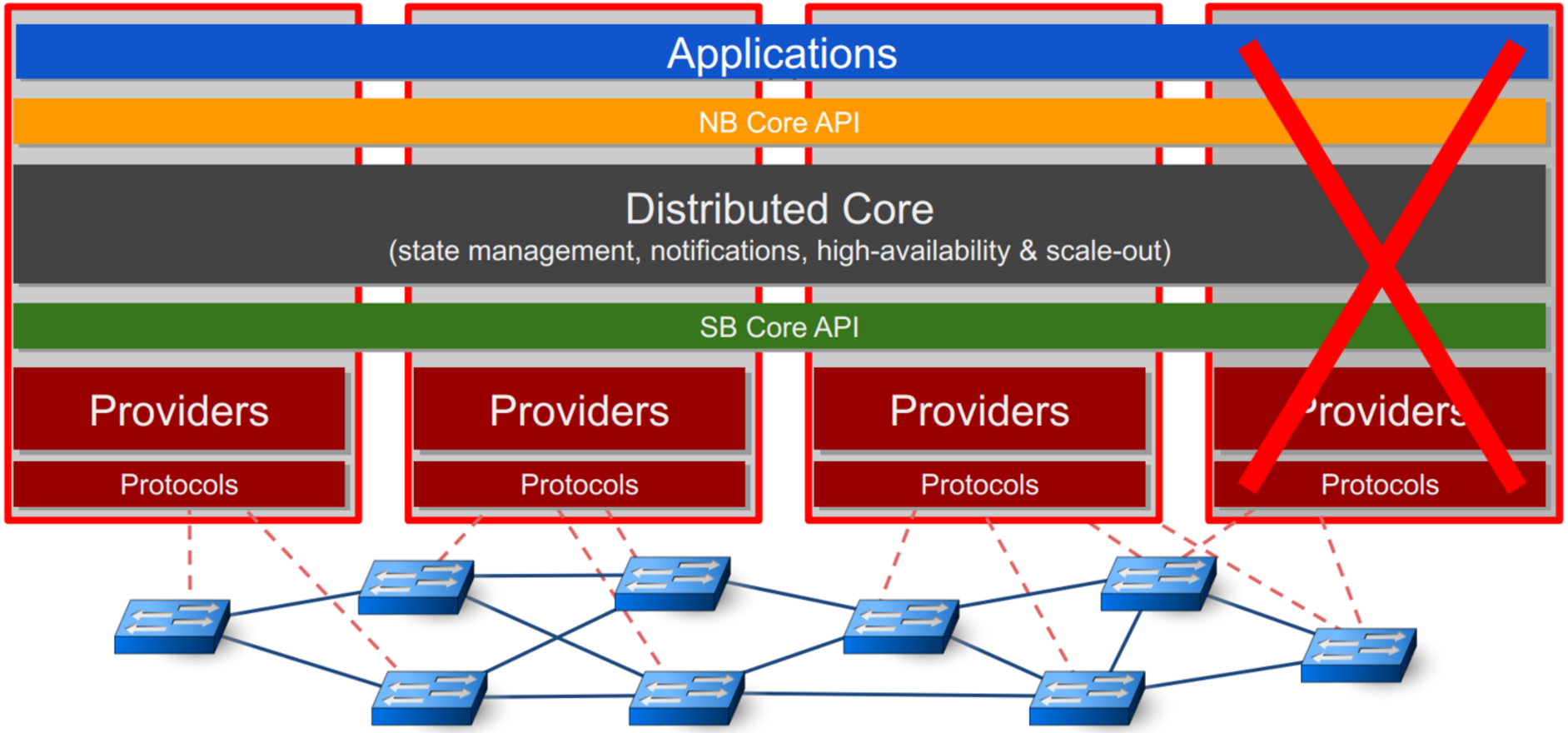


ONOS Architecture (2/6)

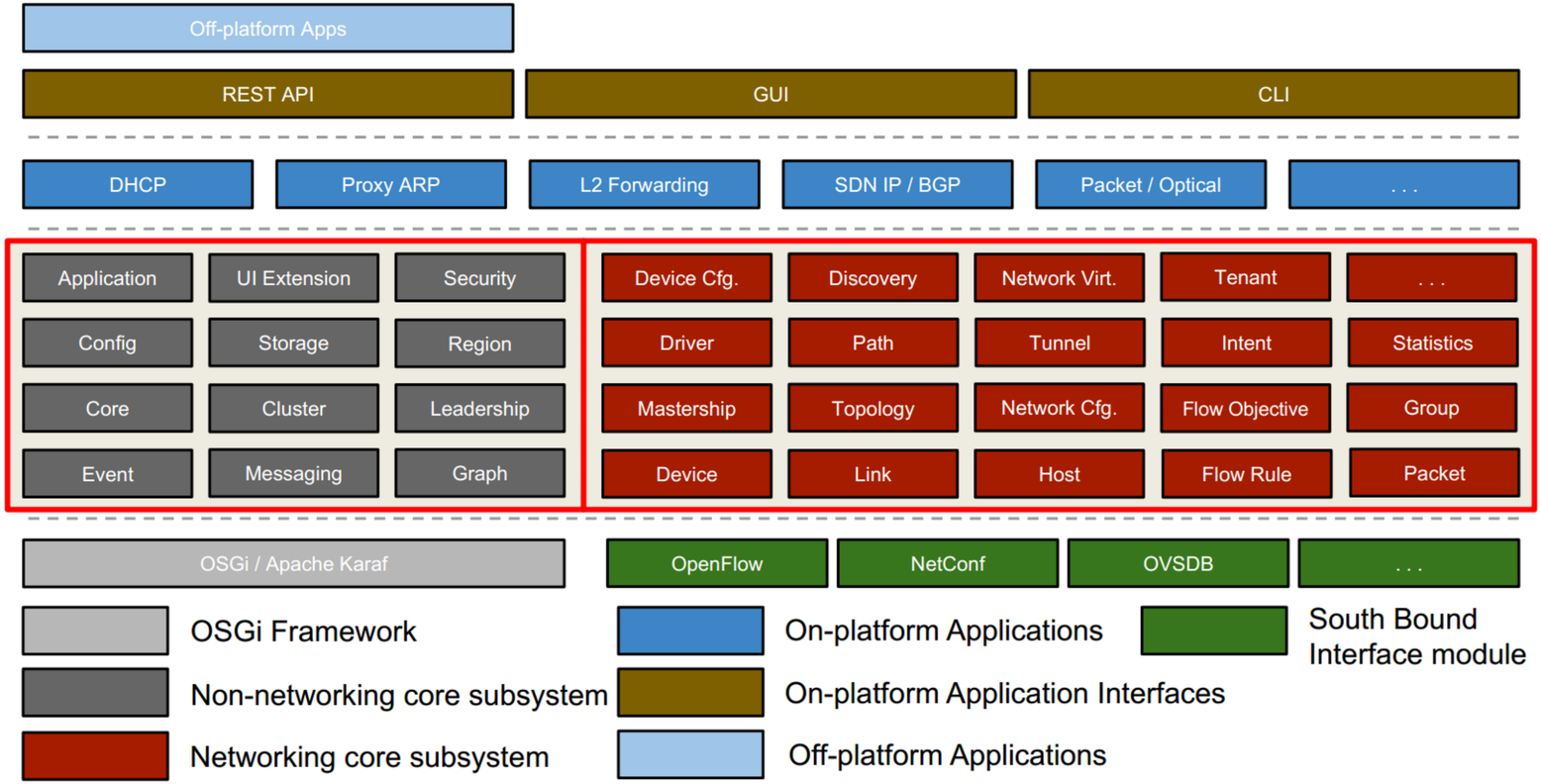


ONOS Architecture (3/6)

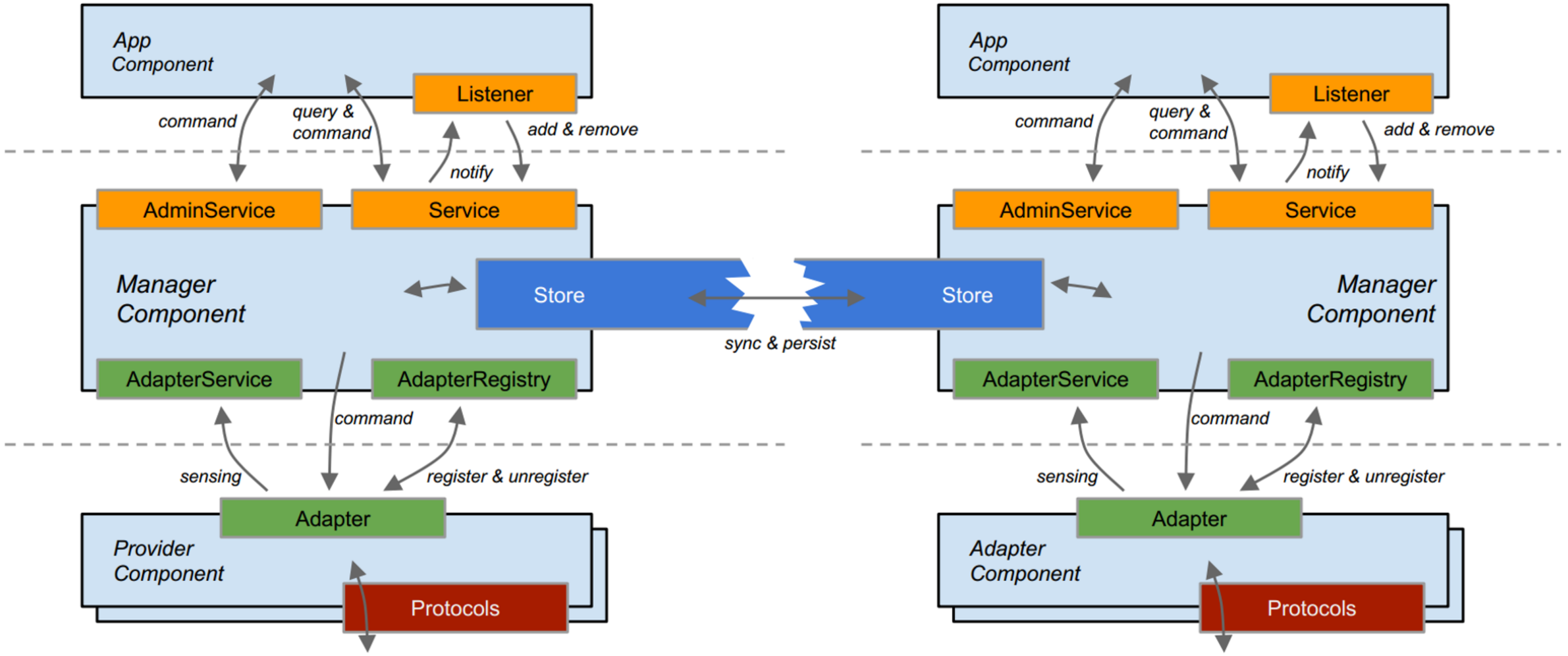
- Distributed Core Features
  - High Availability (HA)
  - Load Balancing (LB)



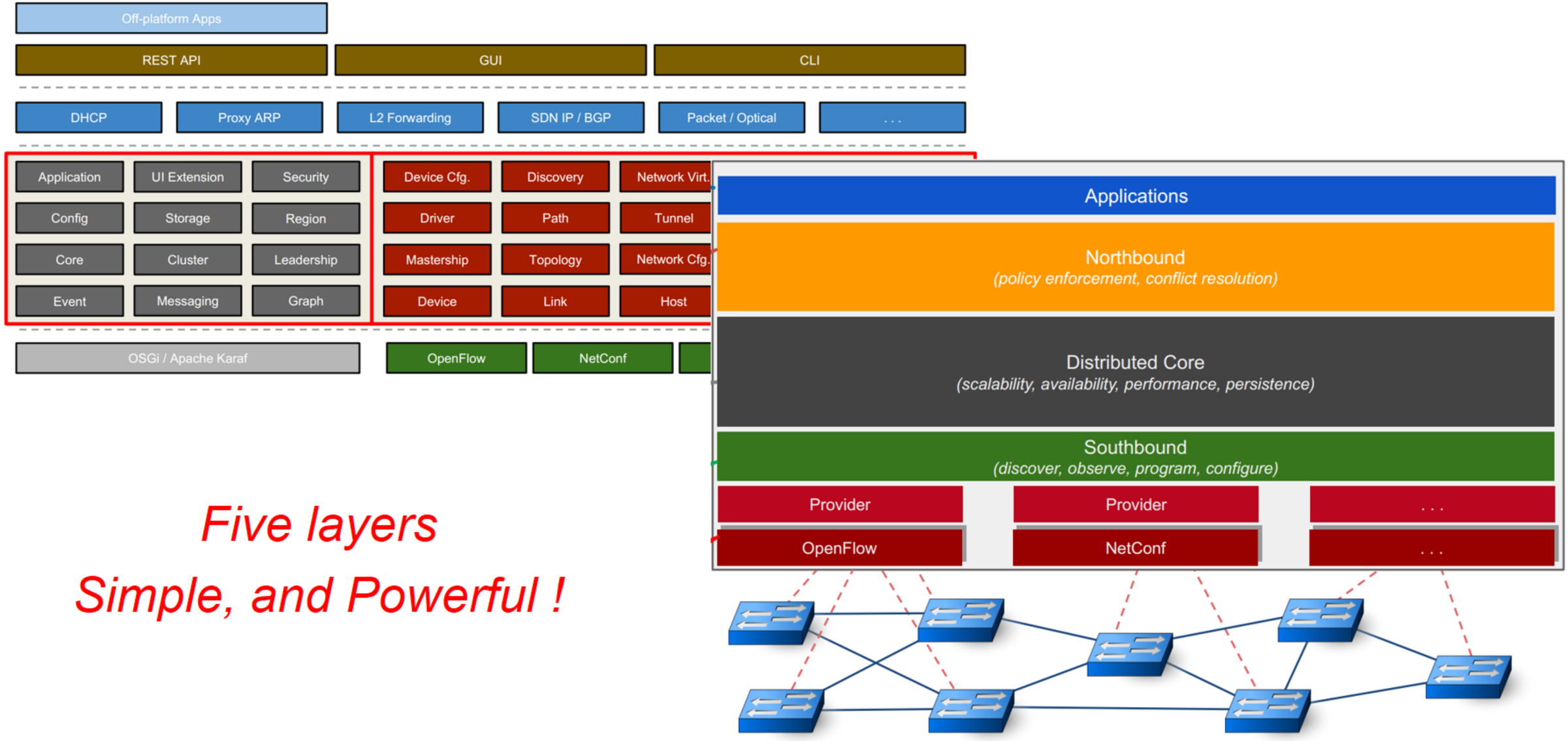
ONOS Architecture (4/6) - Subsystems



ONOS Architecture (5/6)



ONOS Architecture (6/6) - Subsystems



Five layers

Simple, and Powerful !

ONOS Newest Feature

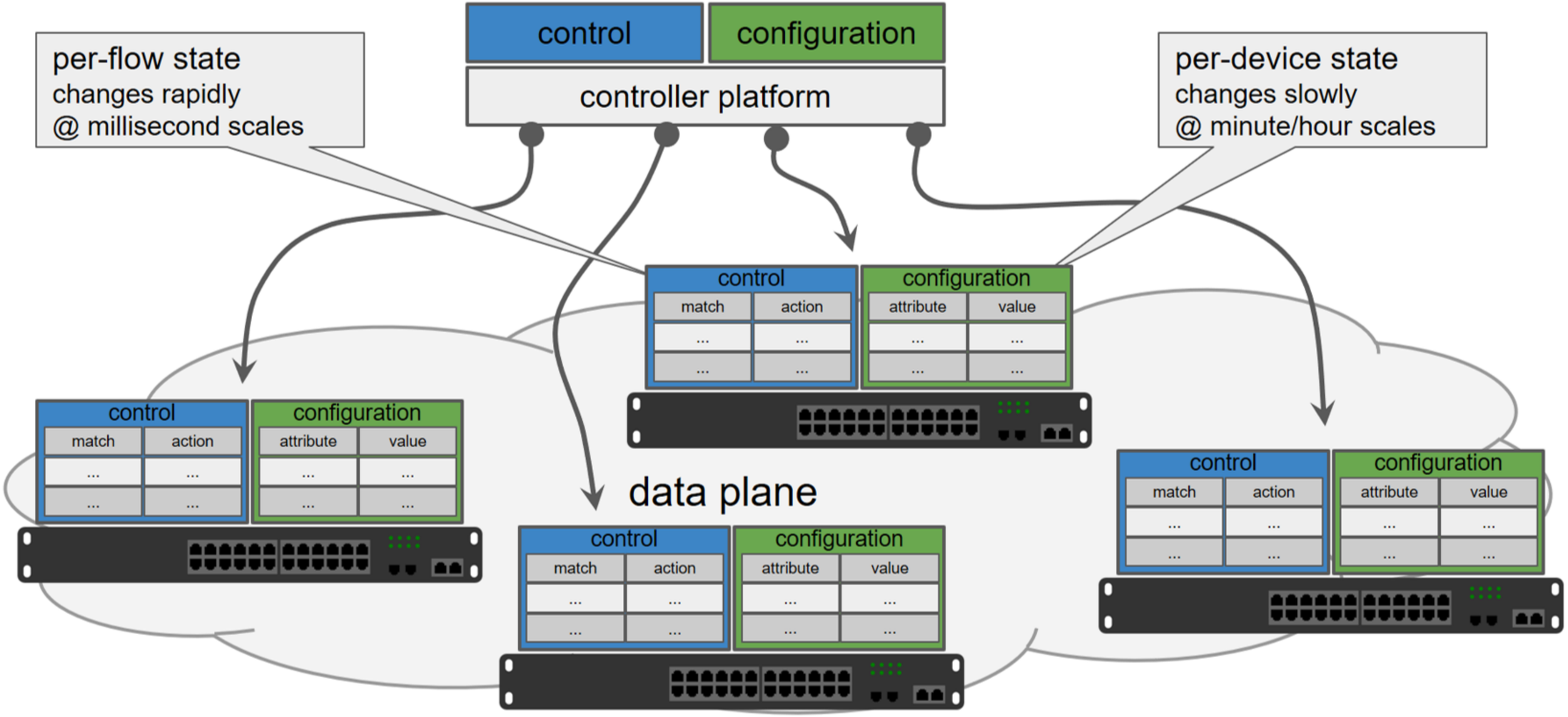
YANG Support

&

Dynamic Configuration Subsystem

Dynamic Configuration & YANG

- What is Dynamic Configuration?



Dynamic Configuration & YANG

Why we need it? ➔ ONOS can support legacy devices now!

- Before, ONOS regarded as a pure SDN solution for **white-box** switches
  - OpenFlow, OVSDB

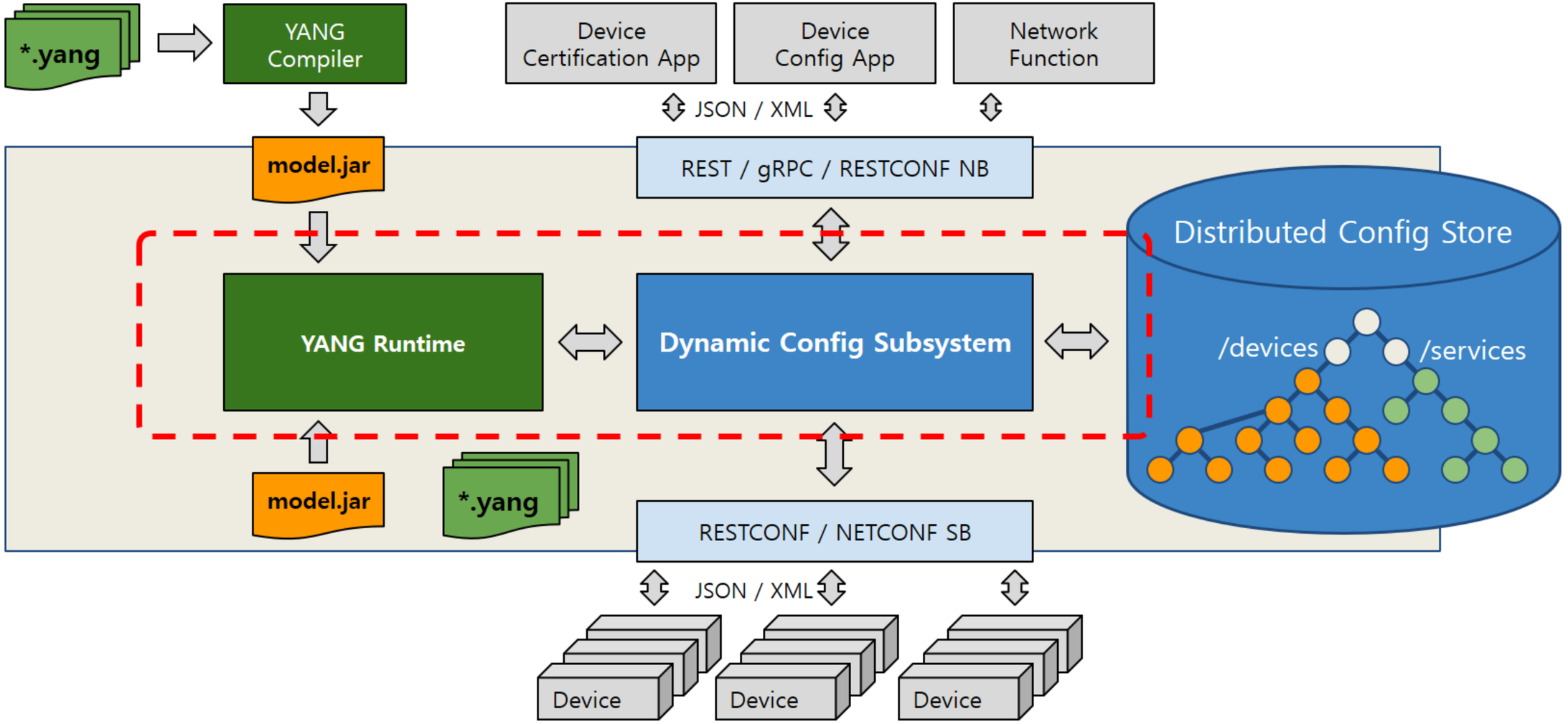
- Now, ONOS can cover **legacy devices**!
  - YANG, NETCONF, RestConf
  - Dynamic Configuration Subsystem

Dynamic Configuration & YANG

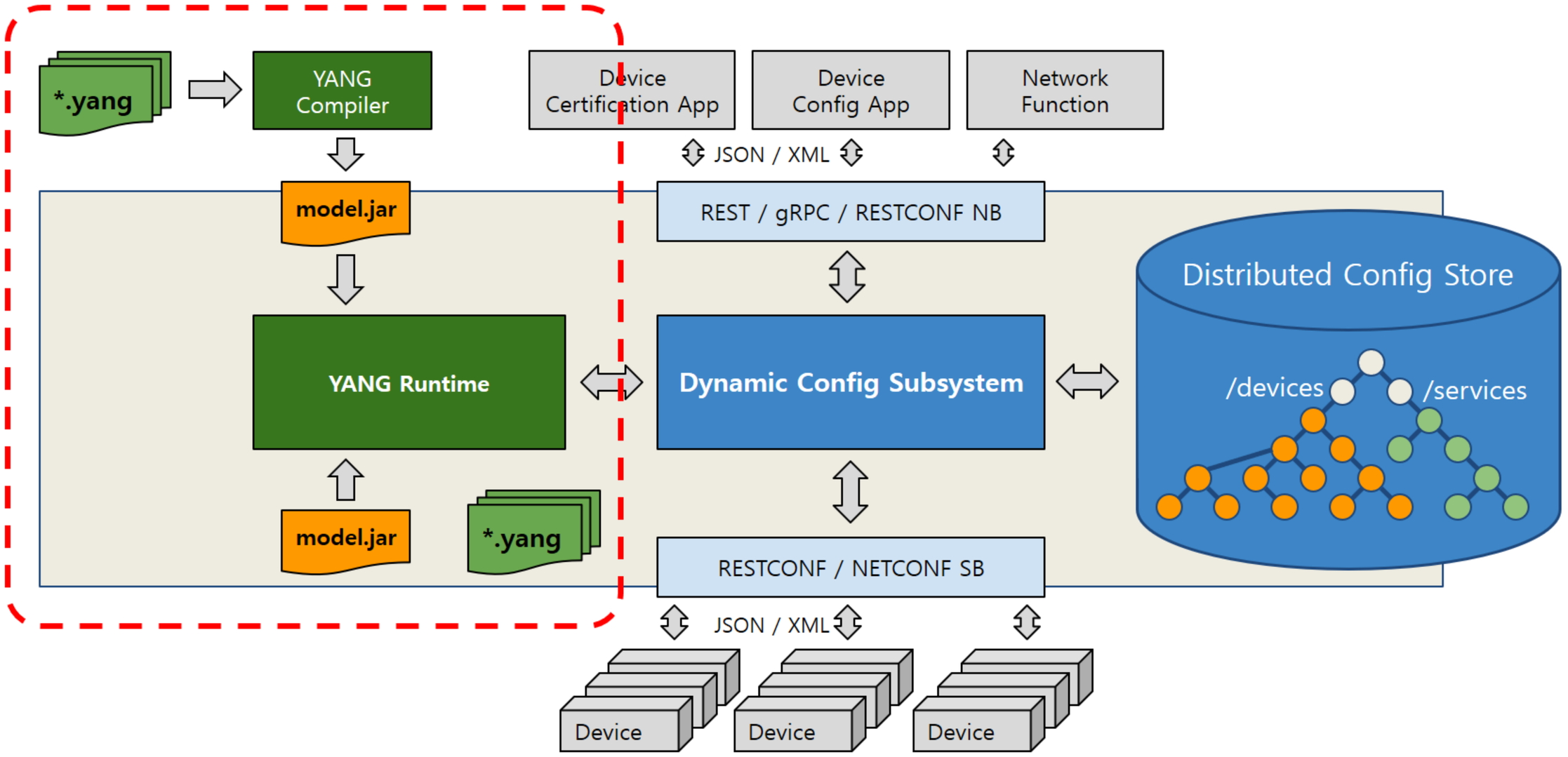
Why we need it? ➔ Configuration is still critical

- Dynamic (re)configuration continues to be critical
  - networks **still** need to be **managed and configured**
  - if nothing else, configured to be controlled
- Configuration even more important in brown-fields
  - devices may **expose** only **limited control capabilities**
- Operators want to **create & sell customized services**
  - do this with agility and **minimal human intervention**
  - create **automated ways** to instantiate such network services
  - services comprise both configuration & control

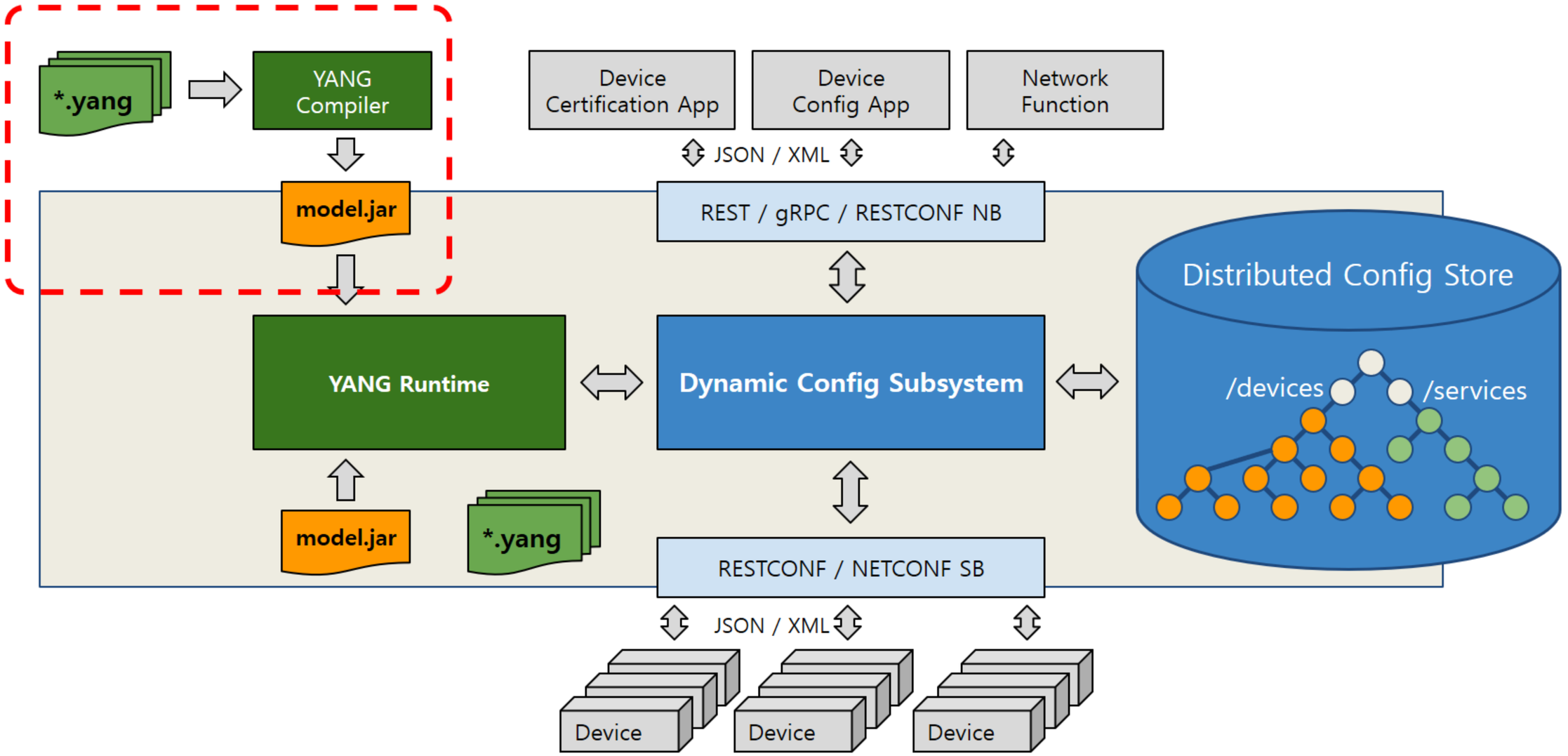
Dynamic Configuration & YANG



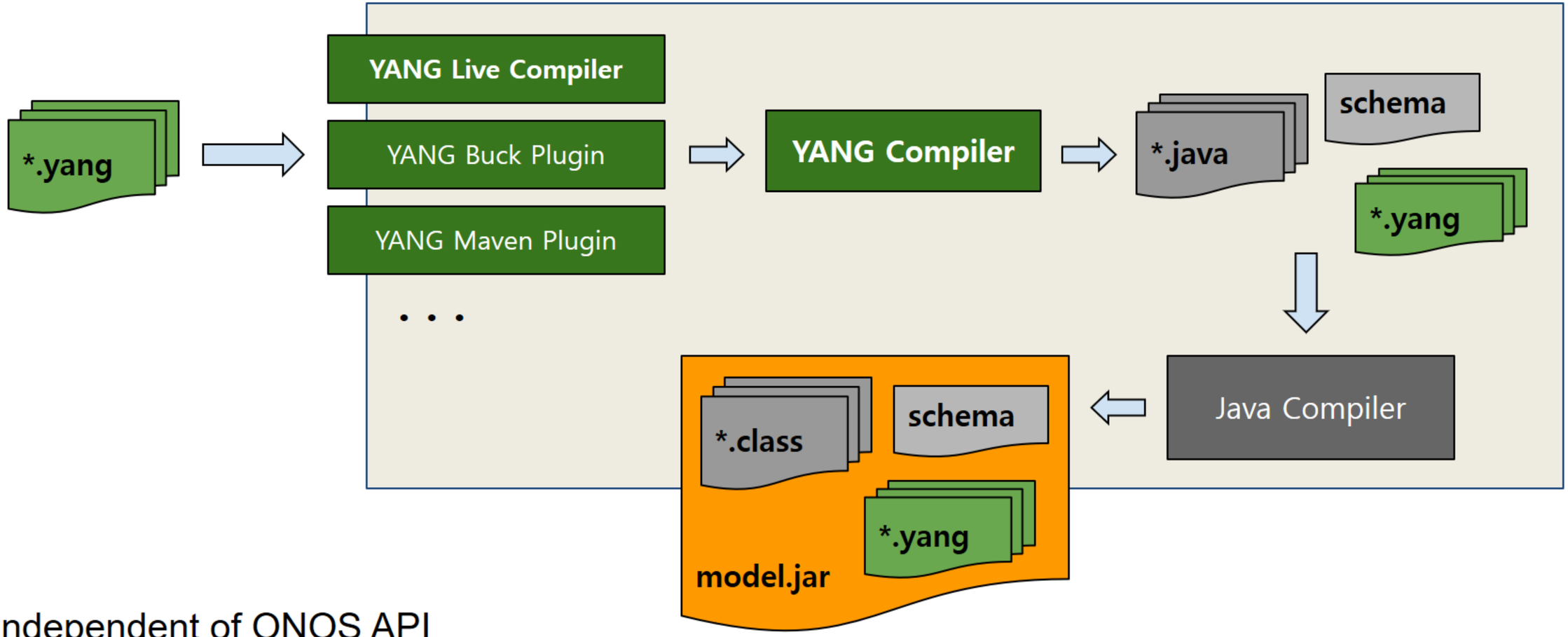
Dynamic Configuration & YANG



Dynamic Configuration & YANG

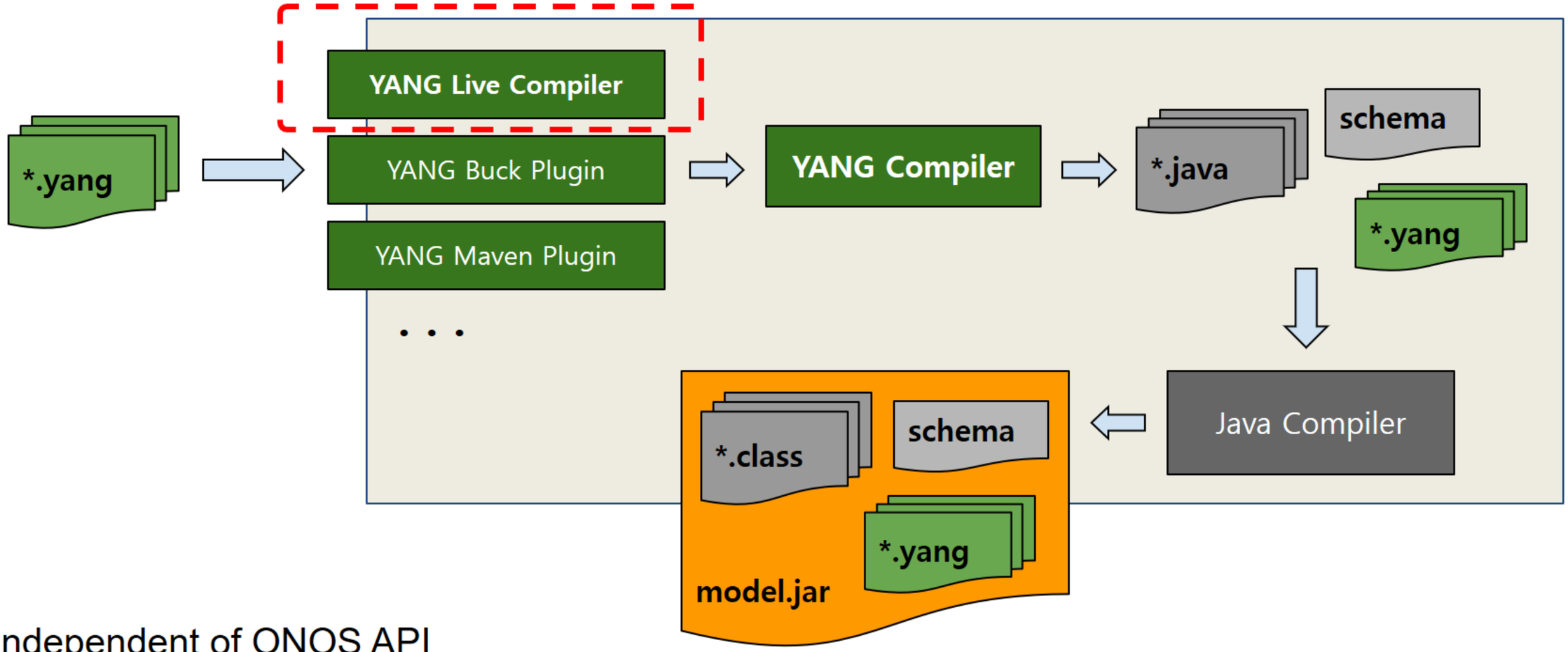


YANG Runtime - compile



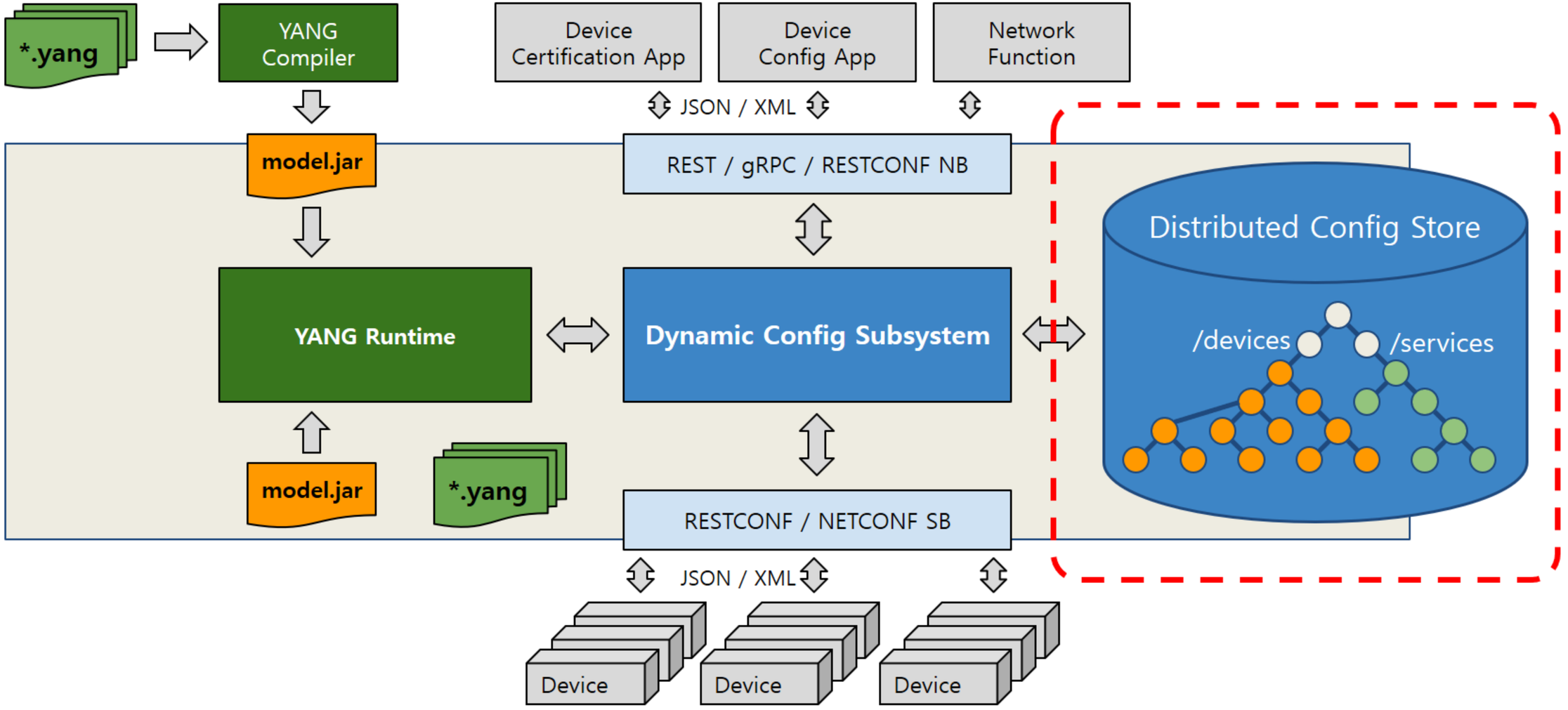
- ✓ Independent of ONOS API
- ✓ Supports model-agnostic data traversal
- ✓ Generates schema for run-time validation and encoding/decoding
- ✓ Generates model-specific rich data types

YANG Runtime - compile



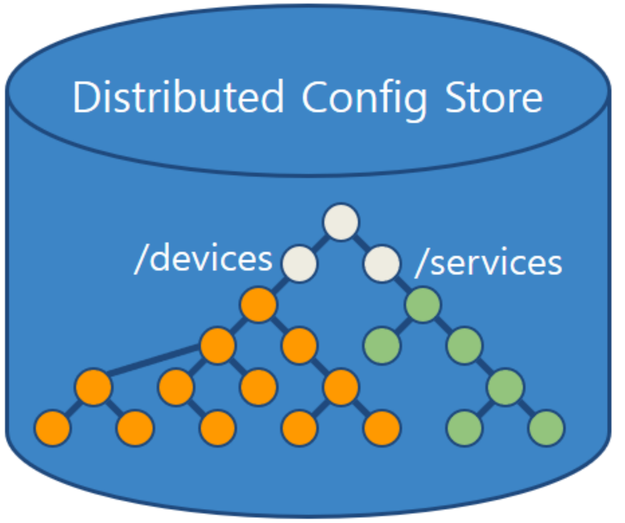
- ✓ Independent of ONOS API
- ✓ Supports model-agnostic data traversal
- ✓ Generates schema for run-time validation and encoding/decoding
- ✓ Generates model-specific rich data types

Dynamic Configuration & YANG



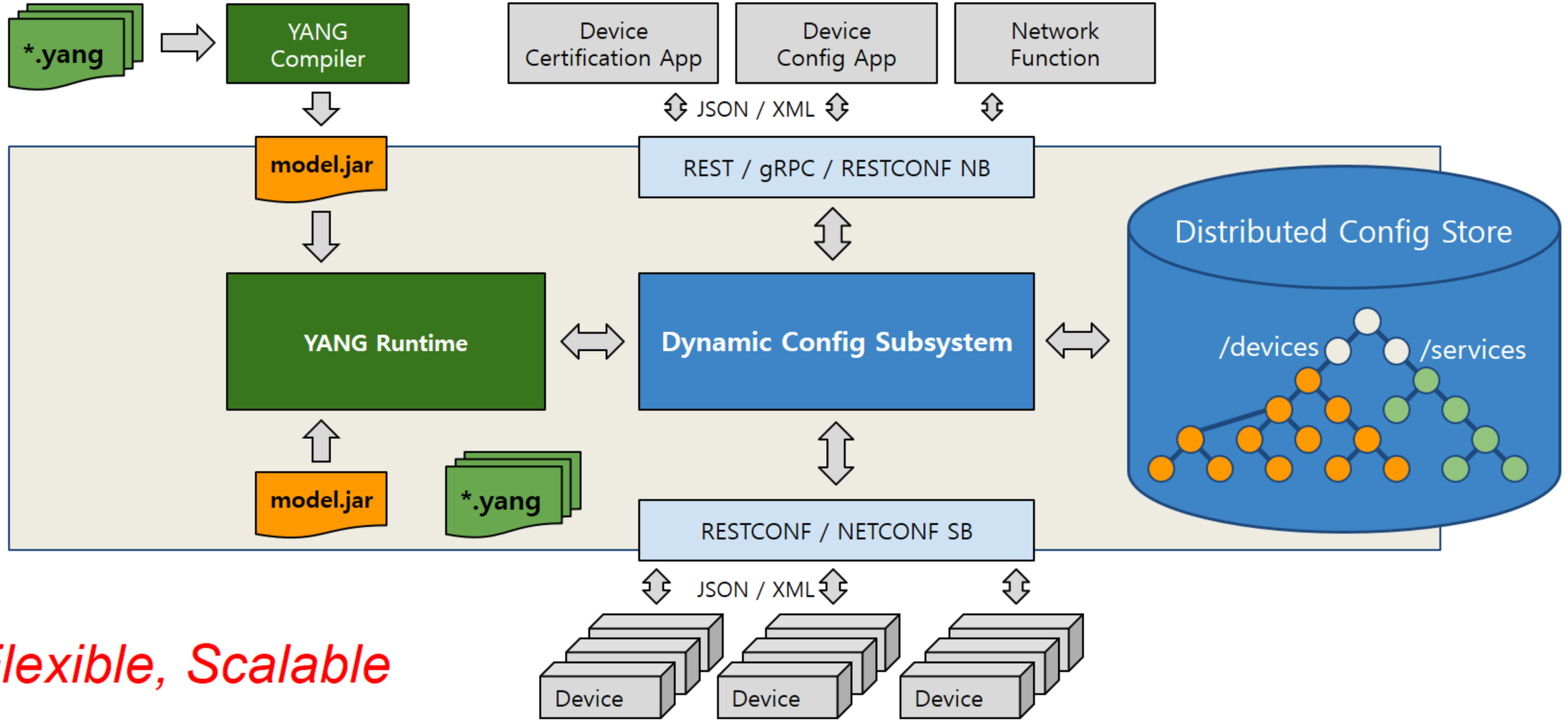
Dynamic Configuration & YANG

What does the Store look like? ➔ a Tree



- Now, ONOS Dynamic Configuration **Store**
  - implemented as **a fully-expanded tree**
  - holds both **configuration** data and **operational** state
  - holds both **service** and **device** configurations
- Scalability challenge for **large** networks
  - requires **partitioning** and **extensive optimizations** to scale
  - partitions **replicated** to maintain performance & high-availability
  - E.g. addressing meta-information is disproportionately sized
  - **high flexibility** from Tree carries a fairly **heavy tax**
- Future, **considering** alternate approach for the long-term solution
  - easier to scale, provide a reasonable level of flexibility and adaptability to arbitrary models.

Dynamic Configuration & YANG



Flexible, Scalable

PI Framework

P4 Runtime support in ONOS

P4 support in ONOS

- How can we control and configure **programmable** devices?

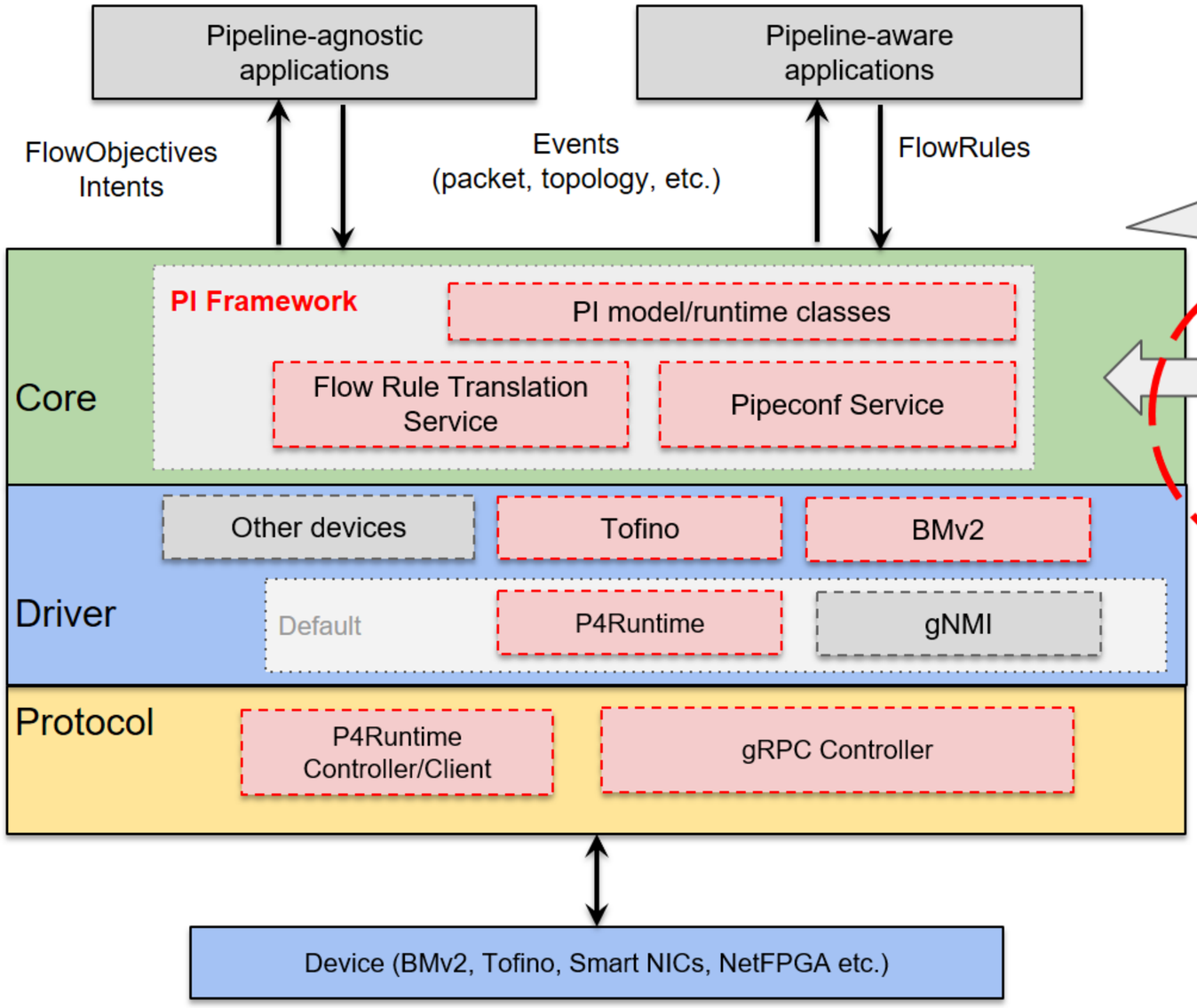
- ONOS initially designed around OpenFlow **fixed-function** dataplane
  - NB abstractions morphed around OpenFlow (e.g. **same** match/actions)
  - **Immutable** pipeline
    (e.g. In OF-DPA spec, Packet need to be processed by VLAN table **before** ACL table.)
- **Programmable,** such as P4:
  - Generalized forwarding abstraction (e.g. **arbitrary** match/actions)
  - **Mutable** pipeline (devices can support different pipelines in time)

- We want to combine *Fixed-function* with *Programmable* …

P4 support in ONOS - PI Framework

- PI = **protocol / program / pipeline** independent
- Modelled around P4 and PSA
  - Portable Switch Architecture(PSA), like OpenFlow Table Type Patterns(TTP)
- Include classes, services, and driver behaviours to model and control programmable data planes
  - Classes **starting with Pi*,** e.g. PiPipeconf, PiTableEntry, etc.
- Define **abstract** table entries, counters, etc.

P4 in ONOS - PI Architecture



Two modes of operations:

- Standard match, actions (i.e. OpenFlow ones)
- Pipeline-specific ones

Pipeconf (.oar)

new

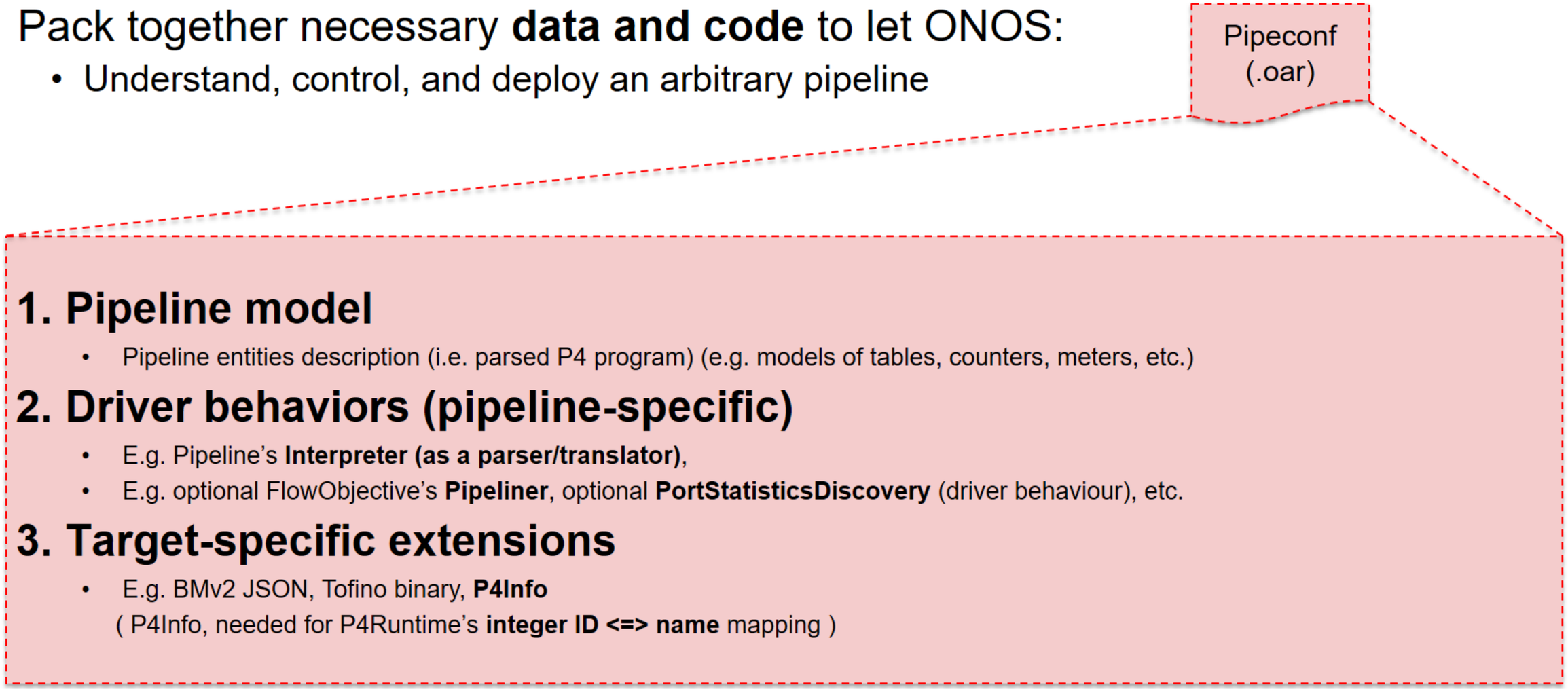
Future Work

Pack together everything needed to:

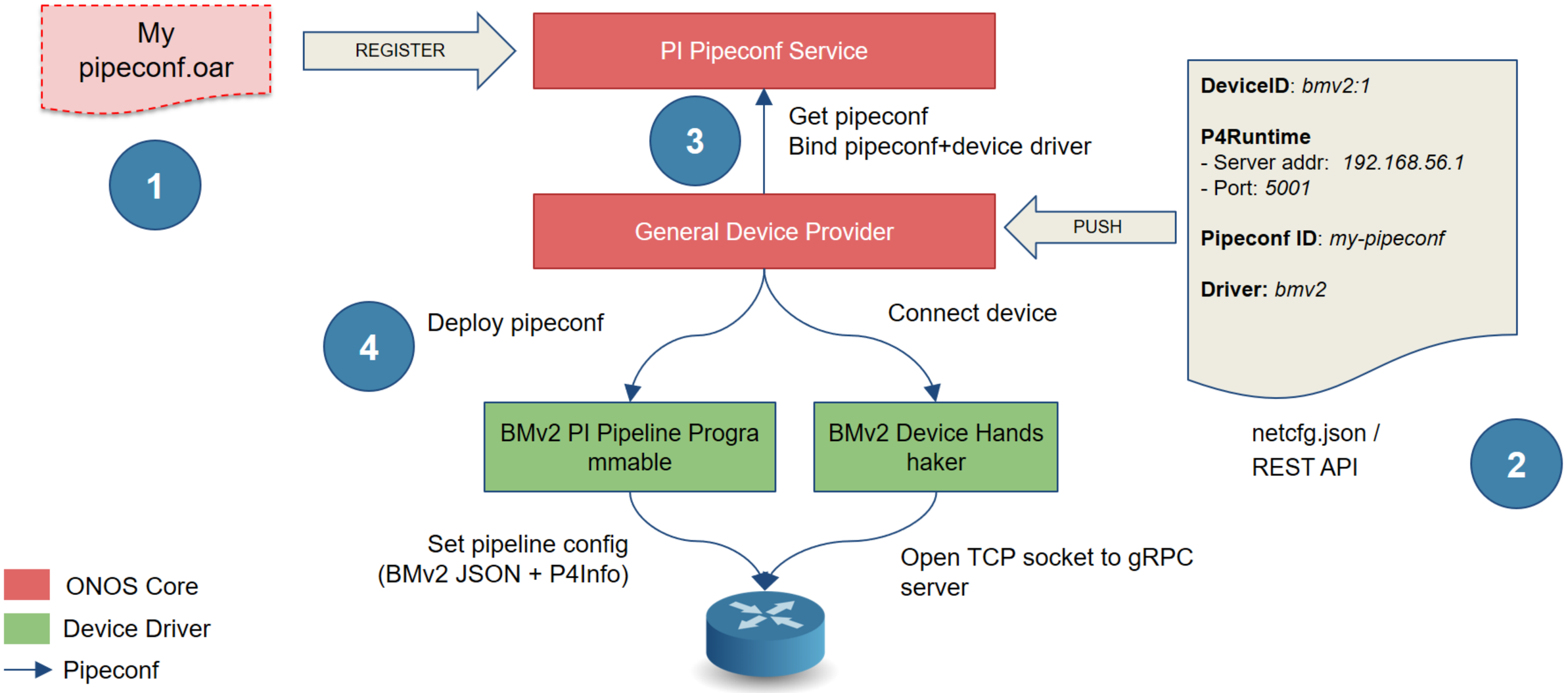
- Understand a P4 program/pipeline
- Control that pipeline (driver behaviour)
- Deploy P4 program to device

P4 in ONOS - Pipeconf

- Provided to ONOS as an application (.oar)
- Pack together necessary **data and code** to let ONOS:
  - Understand, control, and deploy an arbitrary pipeline



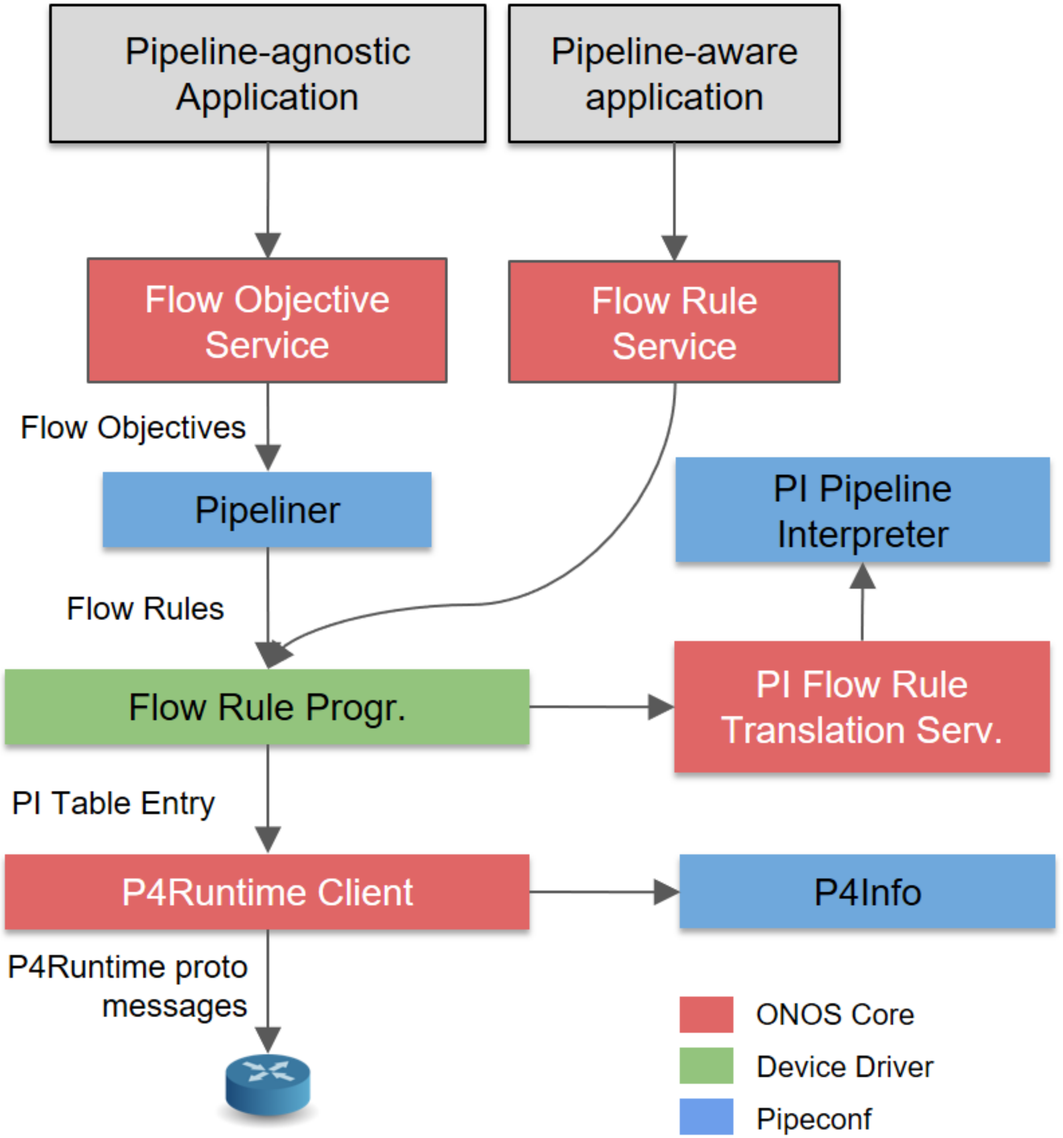
P4 in ONOS - Device Discovery



P4 in ONOS - Flow Operations

Pipeconf-based 3 phase translation:

- Pipeliner
  - Flow Objective ➔ Flow Rule
- **Interpreter**
  - **Flow Rule ➔ PI Table Entry**
- P4Info
  - PI Table Entry ➔ P4Runtime msg



P4 in ONOS - Packet In/Out Operations

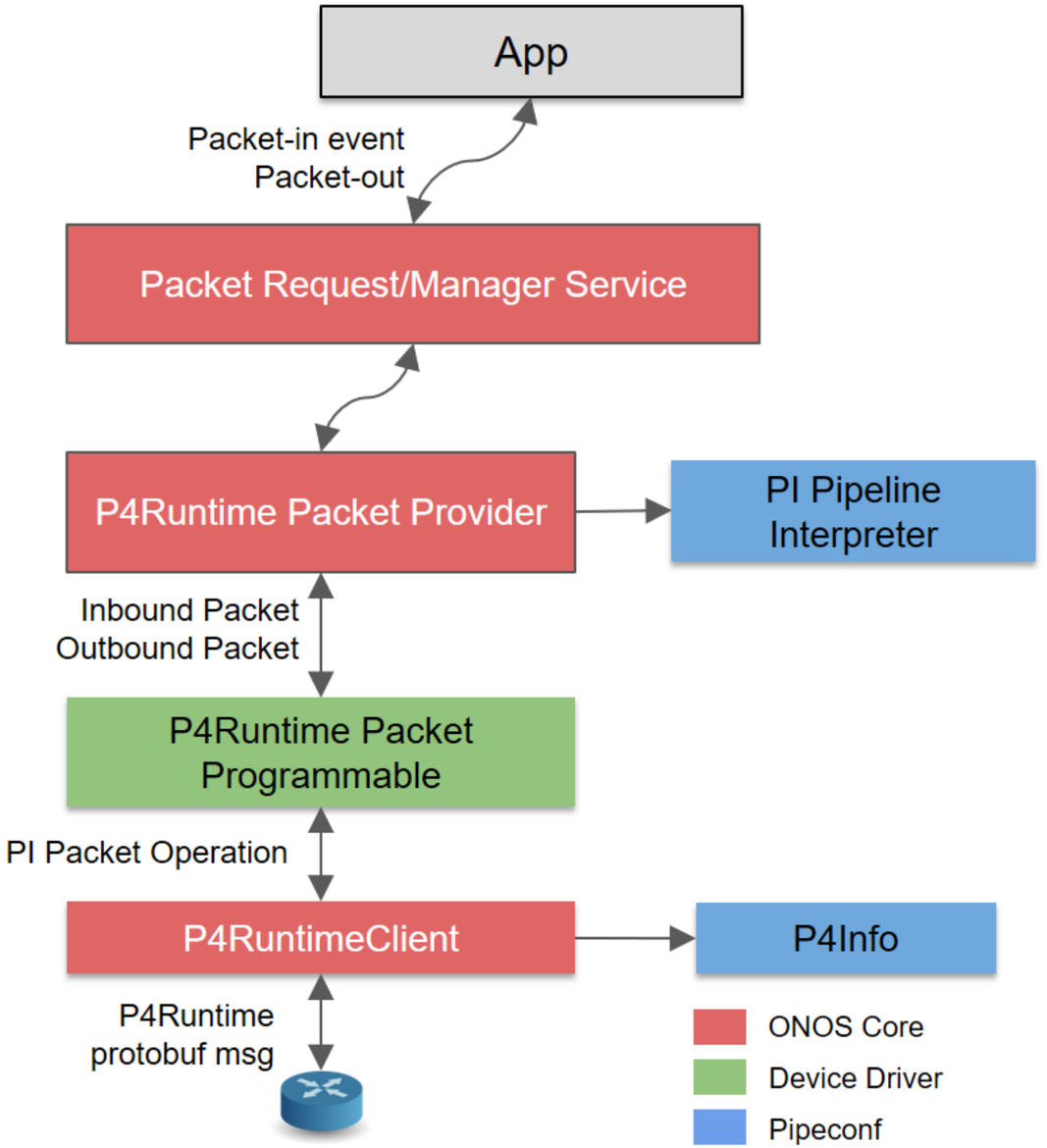
Similar to OpenFlow Packet-in/out Message

Packet-in:
packet received at a switch port encapsulated and sent to the controller.

Packet-out:
packet generated at the controller sent through a switch port.

With P4,
encapsulation format defined by programmer

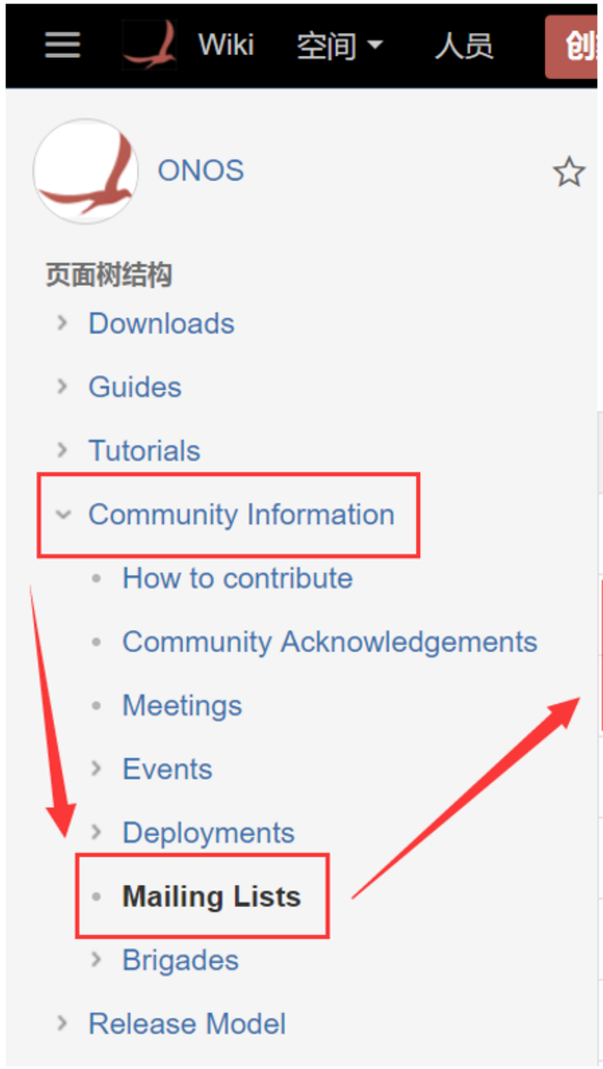
➔ ***Need Interpreter !***



P4 in ONOS - Workflow Review

1. **Write** P4 program
   - If you need, define SDN-like behaviours (packet-in/out, headers, tables, actions)
2. **Compile** it
   - Get P4Info, BMv2's JSON / Tofino's binary
3. Write & Compile & Assemble **Pipeconf** application
   - Pipeline model (e.g. BMv2 JSON)
   - Pipeline-specific driver behaviours:
     - **Interpreter**
     - **Pipeliner** (if you need Flow Objectives)
     - Any other behaviours that depends on the pipeline (e.g. **PortStatisticsDiscovery**)
   - Target-specific extensions
     - P4Info, BMv2 JSON, Tofino binary, etc.
4. Write your own **pipeline-aware application** or use **existing pipeline-agnostic** apps
5. Deploy & Enjoy!

ONOS Community



wiki . onosproject . org

Welcome to share your idea
&
contribute your code and project !

Address	Registration and archives	Description
onos-announce@onosproject.org	Link	General ONOS announcements.
onos-discuss@onosproject.org	Link	General ONOS discussion list.
onos-dev@onosproject.org	Link	ONOS developer discussions.
onos-tst@onosproject.org	Link	ONOS Technical Steering Team discussions.
onos-test@onosproject.org	Link	ONOS tester discussions.
collaborate@onosproject.org	Link	Linked used to send collaboration requests
ambassadors@onosproject.org	Link	ONOS Ambassadors program

ONOS 中文社区

ONOS 研究群

QQ Group：454644351

搜索、发表 ONOS 相关文章

www . sdnlab . com

ONOS with YANG & P4 Runtime

Thank you

毛健炜 **Jianwei Mao**

ONOS Ambassador, China

Beijing University of Posts and Telecommunications (BUPT)

Future Network Laboratory (FNL)

MaoJianwei2012@126.com