SDN Controllers

Professor Chien-Chao Tseng

Department of Computer Science
National Yang Ming Chiao Tung University
cctseng@cs.nctu.edu.tw

Credited to
1. Open Networking Foundation, LF
2. OpenDaylight Project, LF
3. Elisa Rojas Sánchez, Telcaria
4. Michael Vannest, CloudSmartz

Open-source SDN Controllers

https://github.com/noxrepo/nox

- NOX, developed by Nicira Networks and now owned by VMware,
  - NOX-Classic: first OF controller, in C++, API for Python too, 2008
  - NOX “new NOX”: C++ only, 2010
  - POX: Python only, 2010
- Trema: Ruby & C, 2011
- Beacon: Java, 2011
- Floodlight: mostly contributed by Big Switch Networks, Java, 2011
- Ryu: Python, 2012
- OpenDaylight: LF, Java + OSGi, 2013
- ONOS: ONF (now Part of LF), Java + OSGi, 2014
- …

List of SDN controller software
https://en.wikipedia.org/wiki/List_of_SDN_controller_software

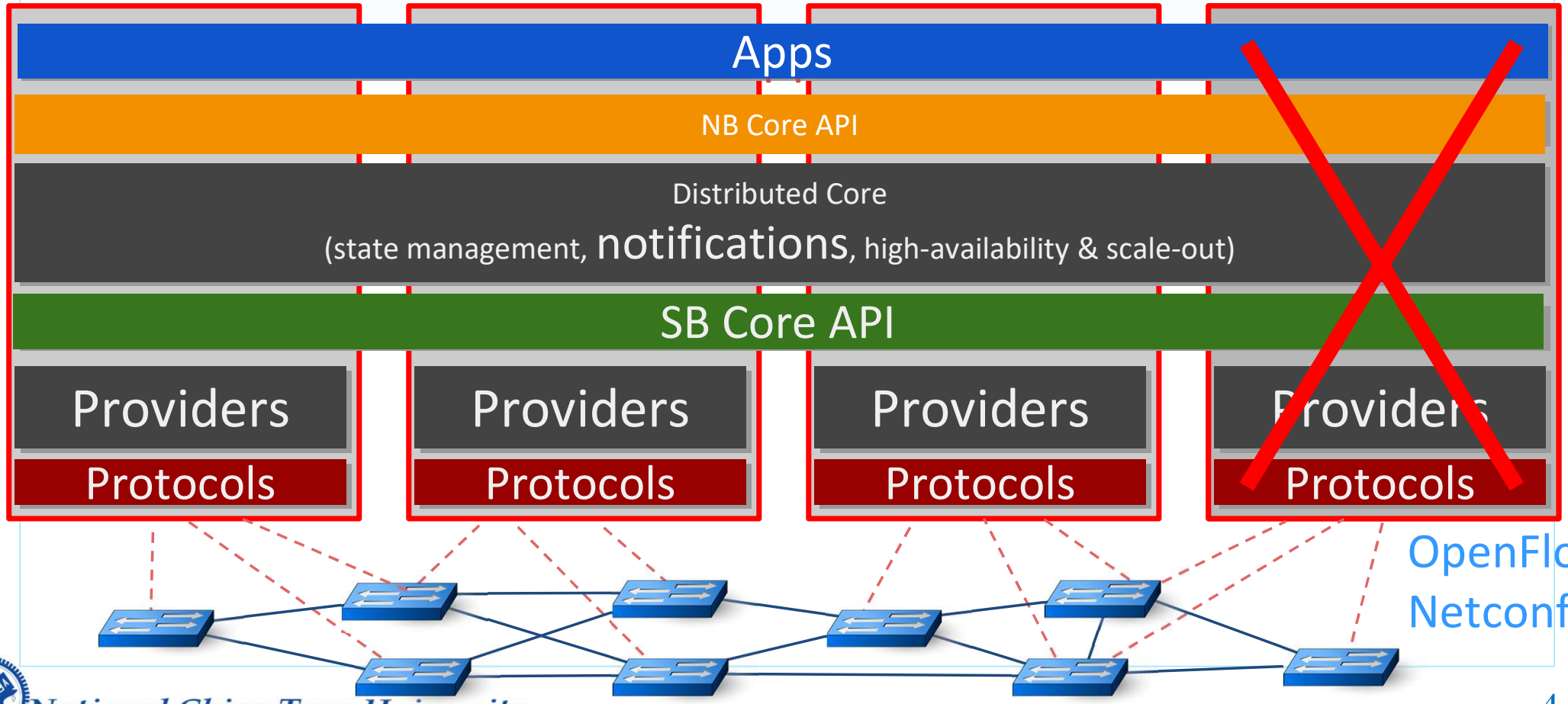
Commercial SDN Solutions

- **Big Switch** Big Cloud Fabric
- **Plexxi** Big Data Fabric
- **Brocade** Vyatta Controller
- **HP** Virtual Application Networks (VAN) SDN Controller/Virtual Cloud Networking (VCN)
- **Juniper** Contrail (was OpenContrail)
- **Cisco** Application Centric Infrastructure (ACI)/Application Policy Infrastructure Controller (APIC)
- **Ericsson** SDN controller (based on ODL)
- …

Elisa Rojas Sánchez, Telcaria
https://fdocuments.in/reader/full/clash-of-titans-in-sdn-opendaylight-vs-onos-elisa-rojas

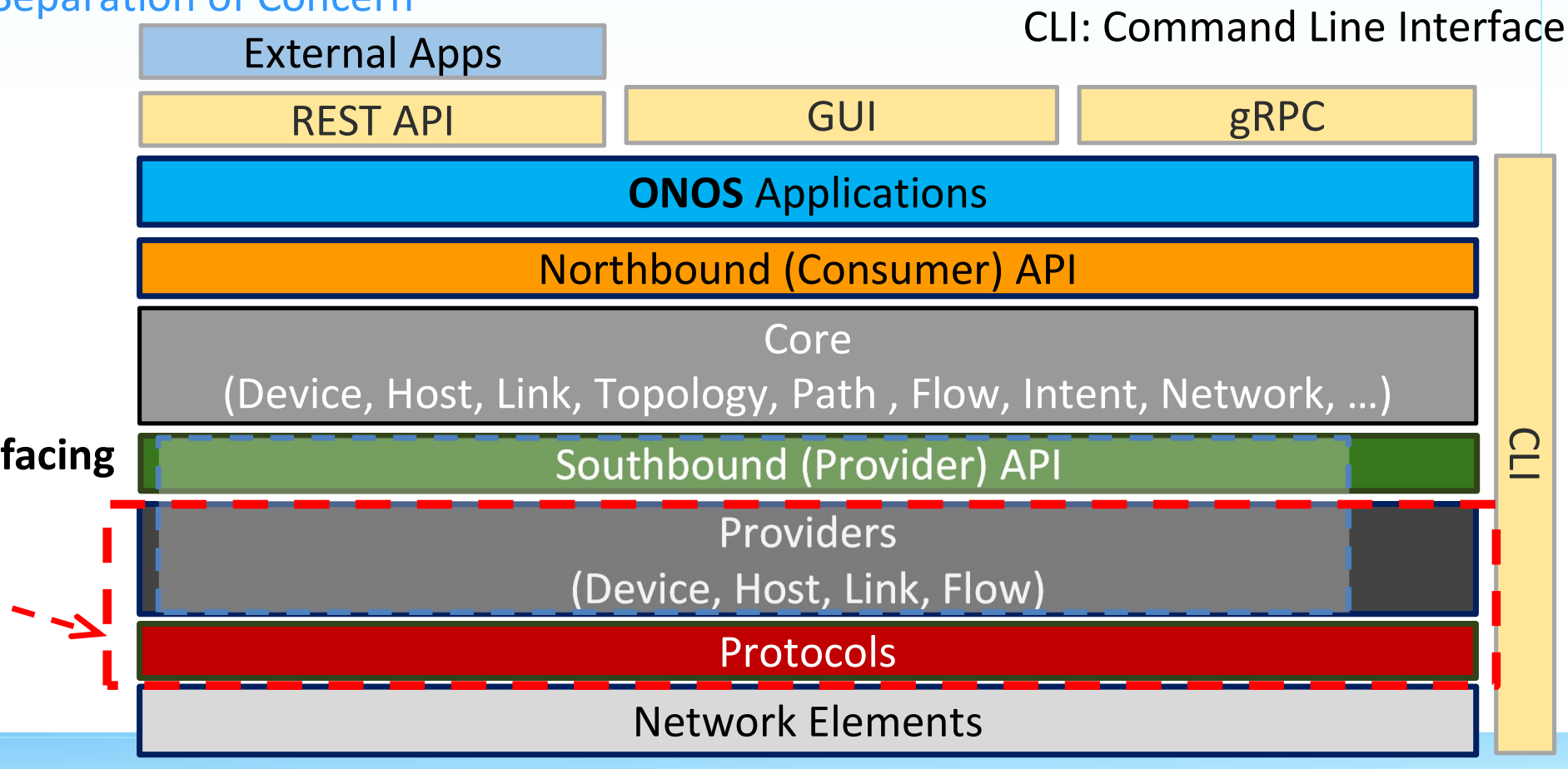
ONOS Distributed Architecture

- Scalable Distributed Core for Scalability, HA, performance



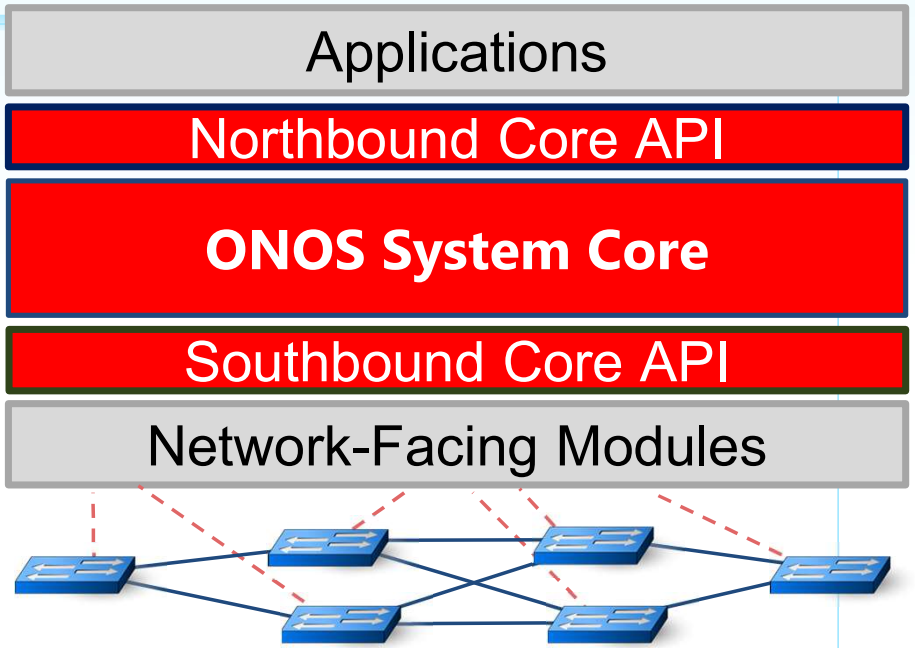
ONOS Architecture

- Layered architected with tiers of functionality
  - Separation of Concern



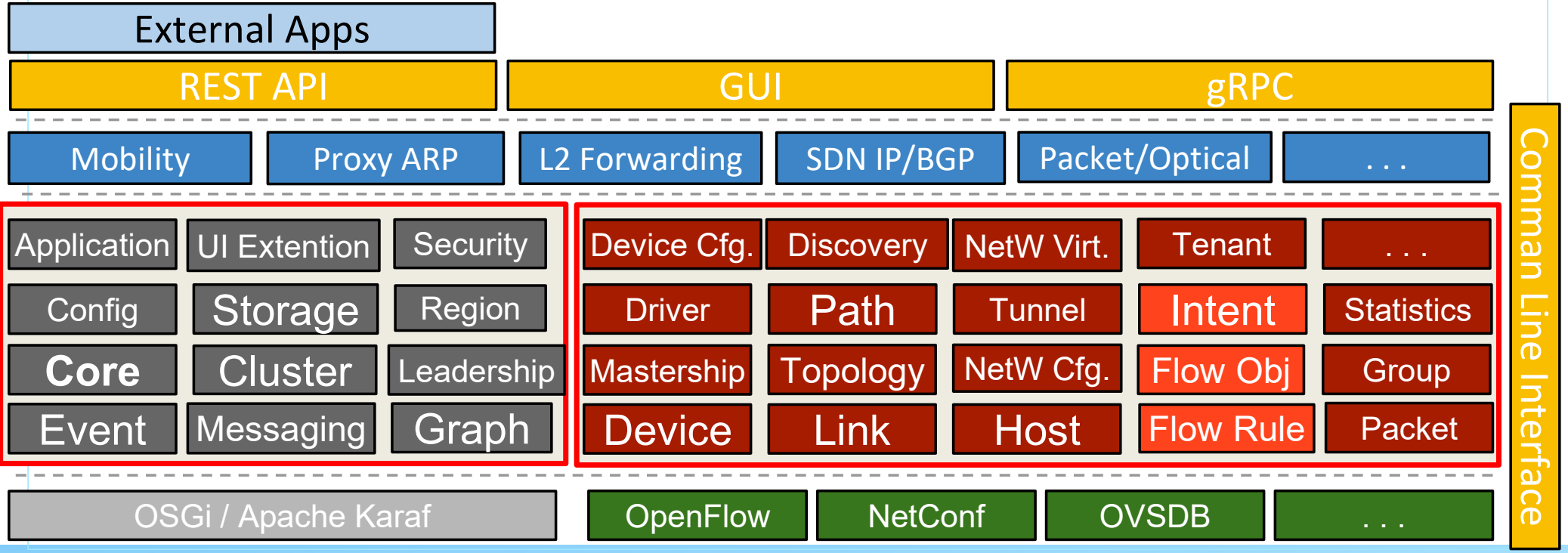
Network Element and State Representations in ONOS

- **Model** Objects: protocol-agnostic
  - Constructs of Core:
  - Exposed to applications
  - ➢ **Shield applications from protocol-specifics**
- **Protocol-specific** Objects
  - Constructs of appropriate **providers**
- **Types of Model Objects**
  - **Network Topology** (directed graphs)
    - *Device, Port, Host, Link, EdgeLink, Path,Topology*
  - **Network Control** (high-level flow rules: match+action)
    - *FlowRule (!=OF), Intent, RoleValue (clusters: NONE, SLAVE, MASTER)*
  - **Network Packets** (protocol agnostic)
    - *OutboundPacket , InboundPacket*



ONOS Core Subsystems (Services)

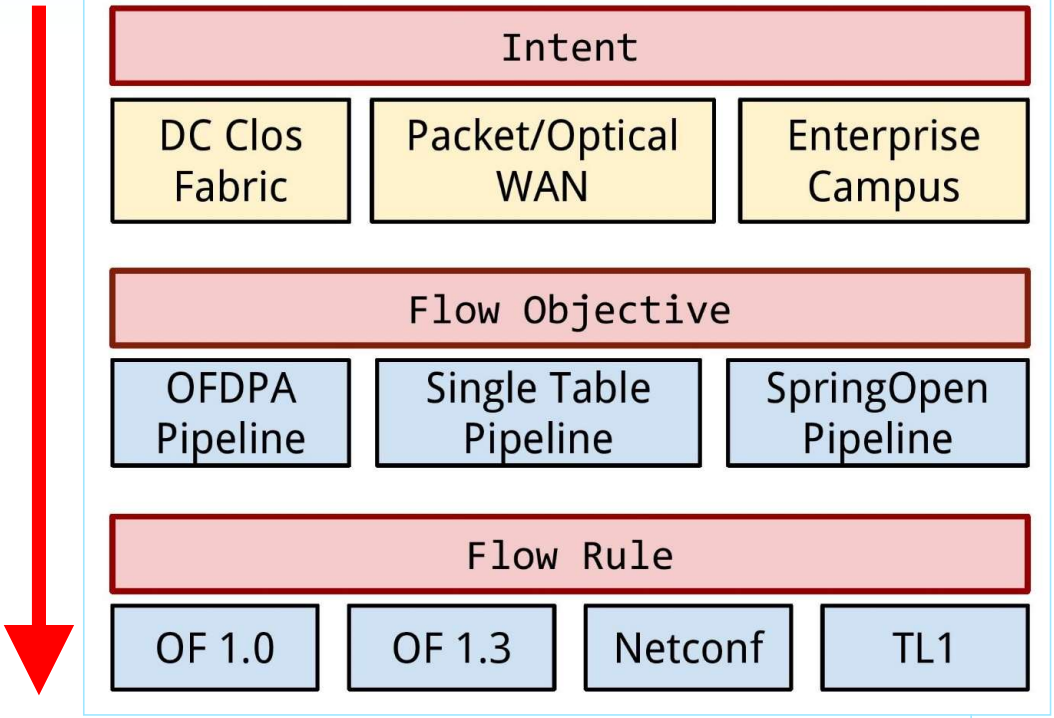
- ***Service***: a unit of functionality comprised of multiple components that create a **vertical slice** (through the tiers as a software stack.)
  - ***Subsystem:*** collection of components making up a service
    - Use 'service' and 'subsystem' interchangeably in this introduction.



Abstraction for Data-Plane Programming

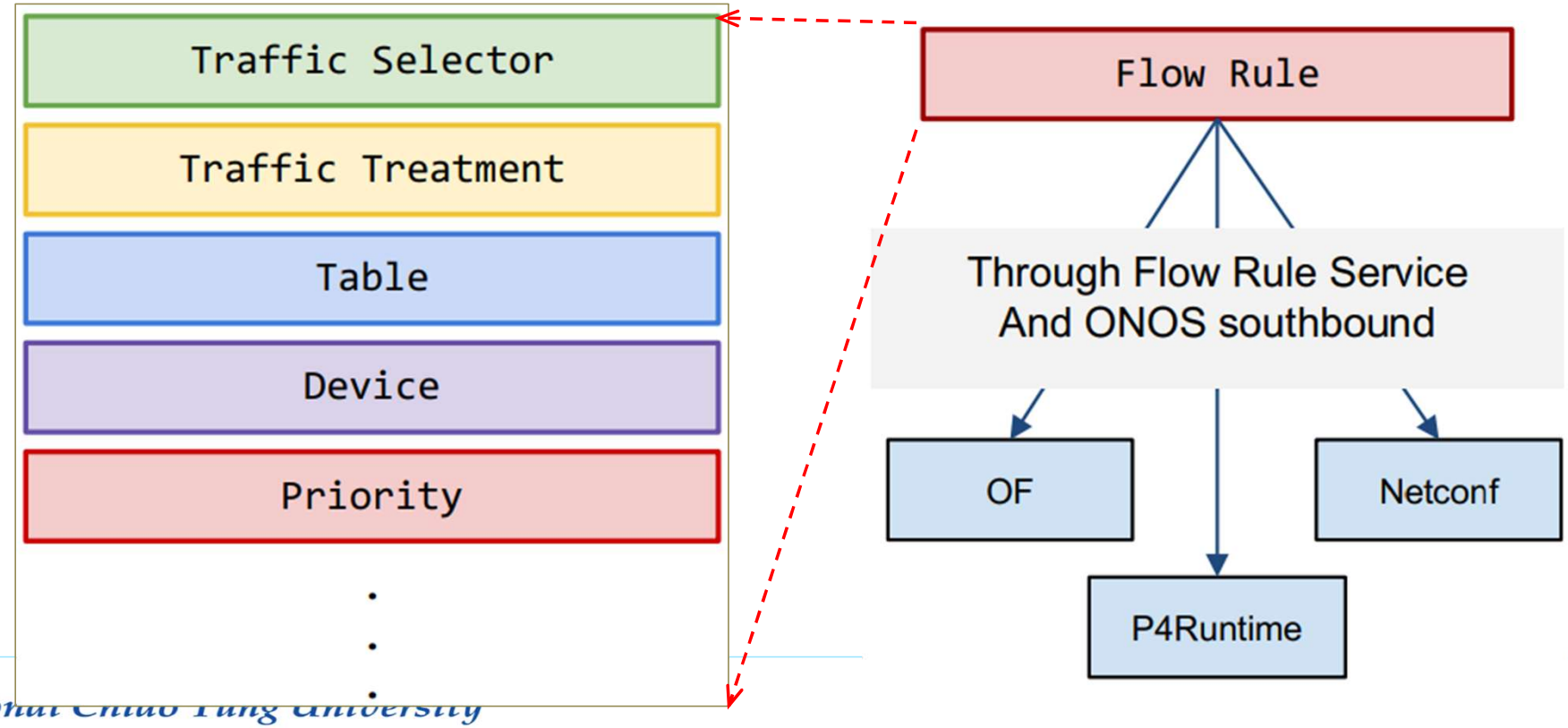
- **Intent**
  - **Network-centric** abstraction
  - Topology-independent
    - Provision 10G path from DC1 to DC 2 optimized for cost
- **Flow Objective**
  - **Device-centric** abstraction
  - Table pipeline-independent and flow rule agnostic
- **Flow Rule**
  - Match/Action abstraction
  - **Protocol-independent**

Abstract to Concrete



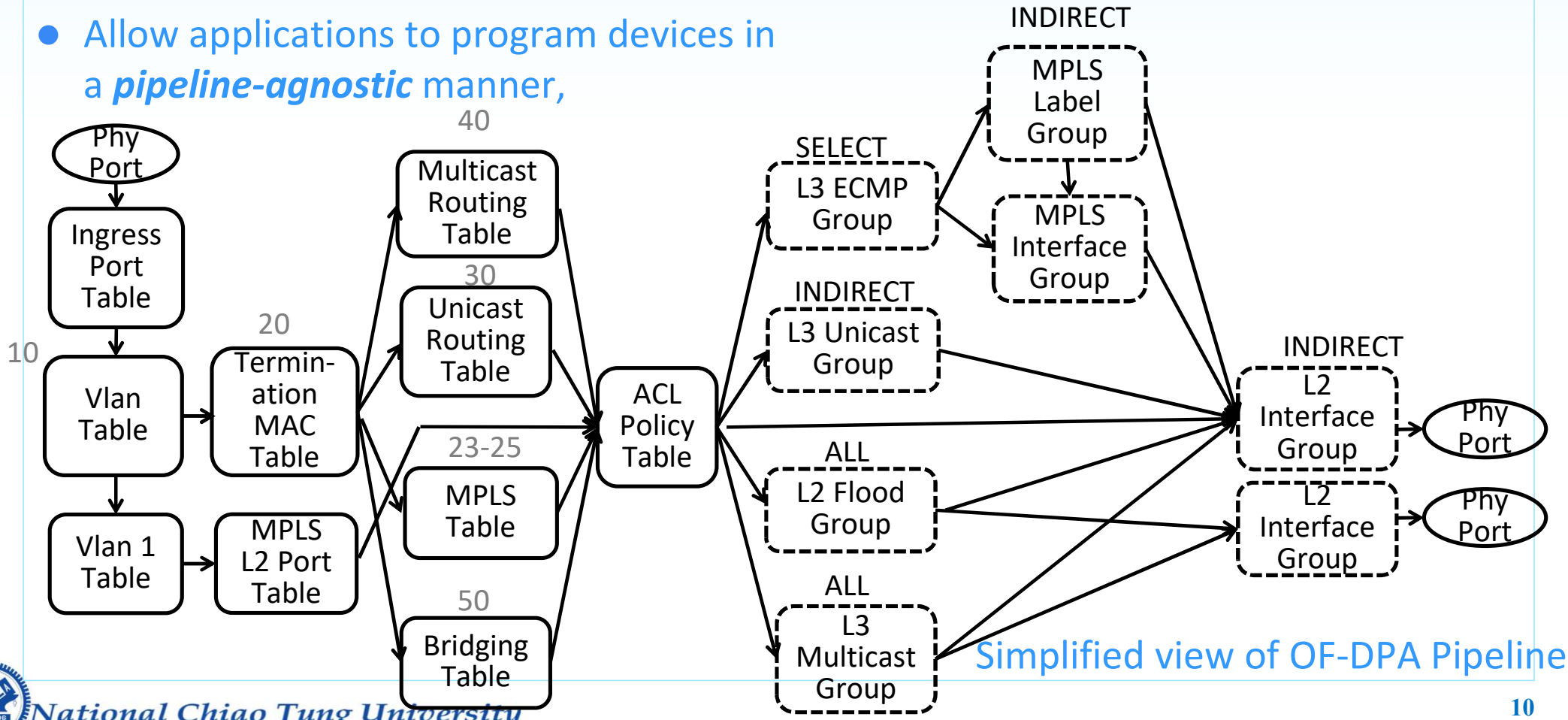
Flow Rule Service

- ONOS core provides FlowRuleService
  - Protocol Independent, Pipeline Specific
    - Applications needs to specify Table and Flow Rule information



Flow Objective Service

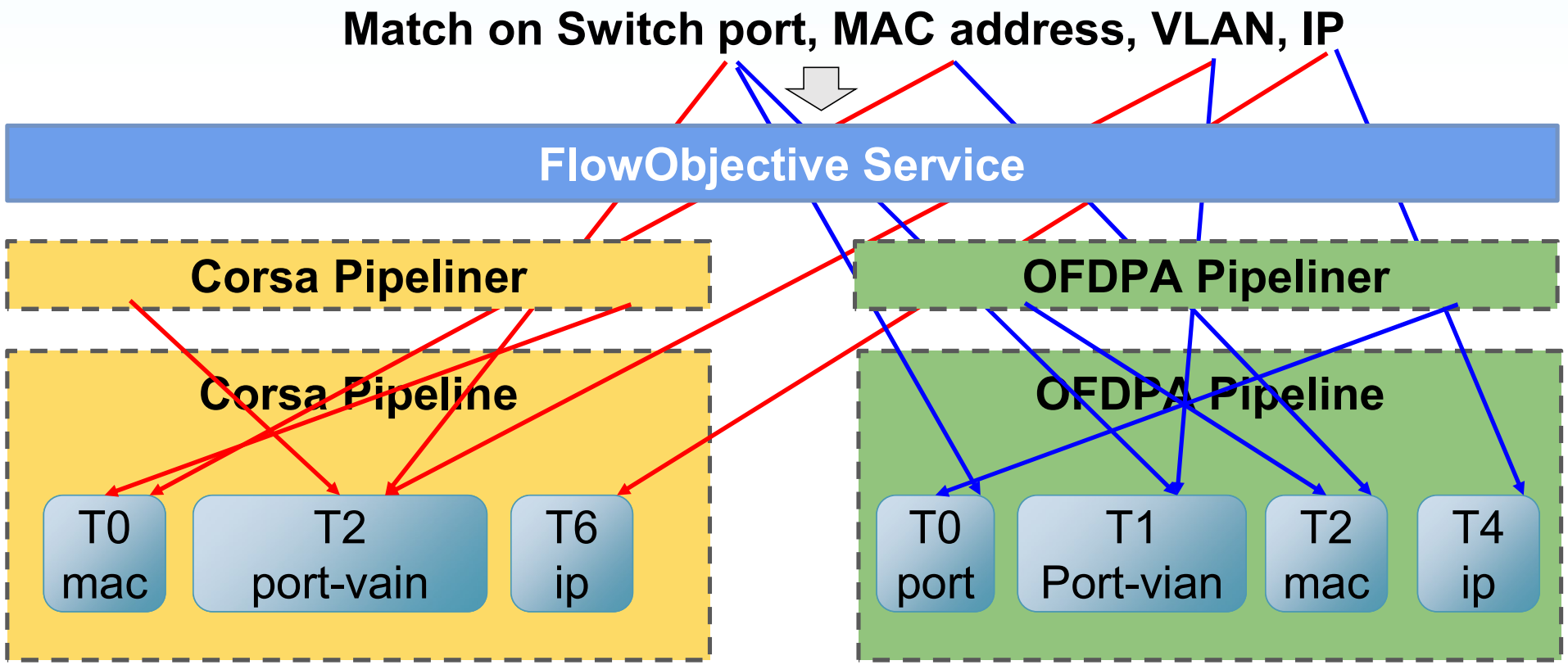
- **Device-centric *abstraction*** (Pipeline Agnostic)
- Allow applications to program devices in a ***pipeline-agnostic*** manner,



Simplified view of OF-DPA Pipeline

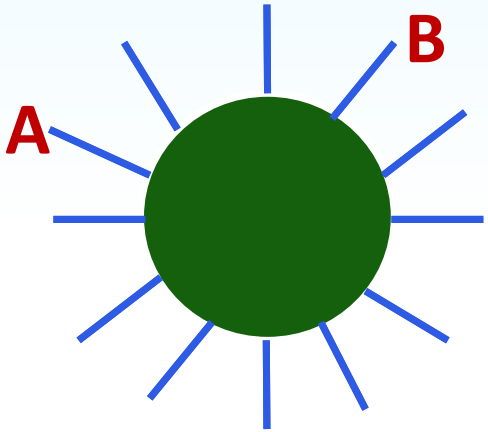
Pipeline Agnostic

- Enable developers to write applications once for all pipelines !?



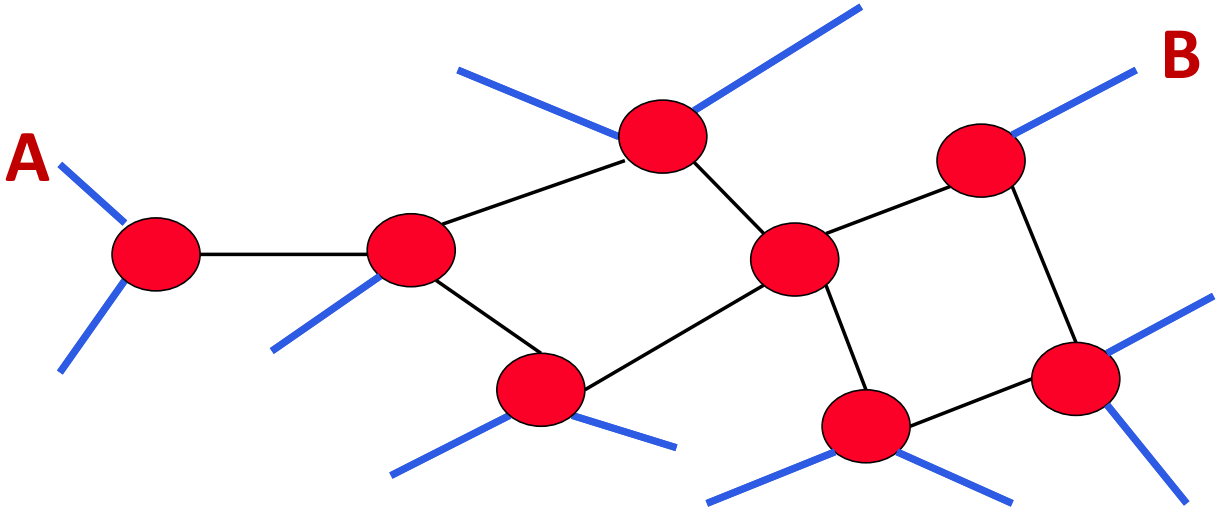
Specification Abstraction

- Simple Example: Access Control



What

Abstract
Network
Model



How

Global
Network
View

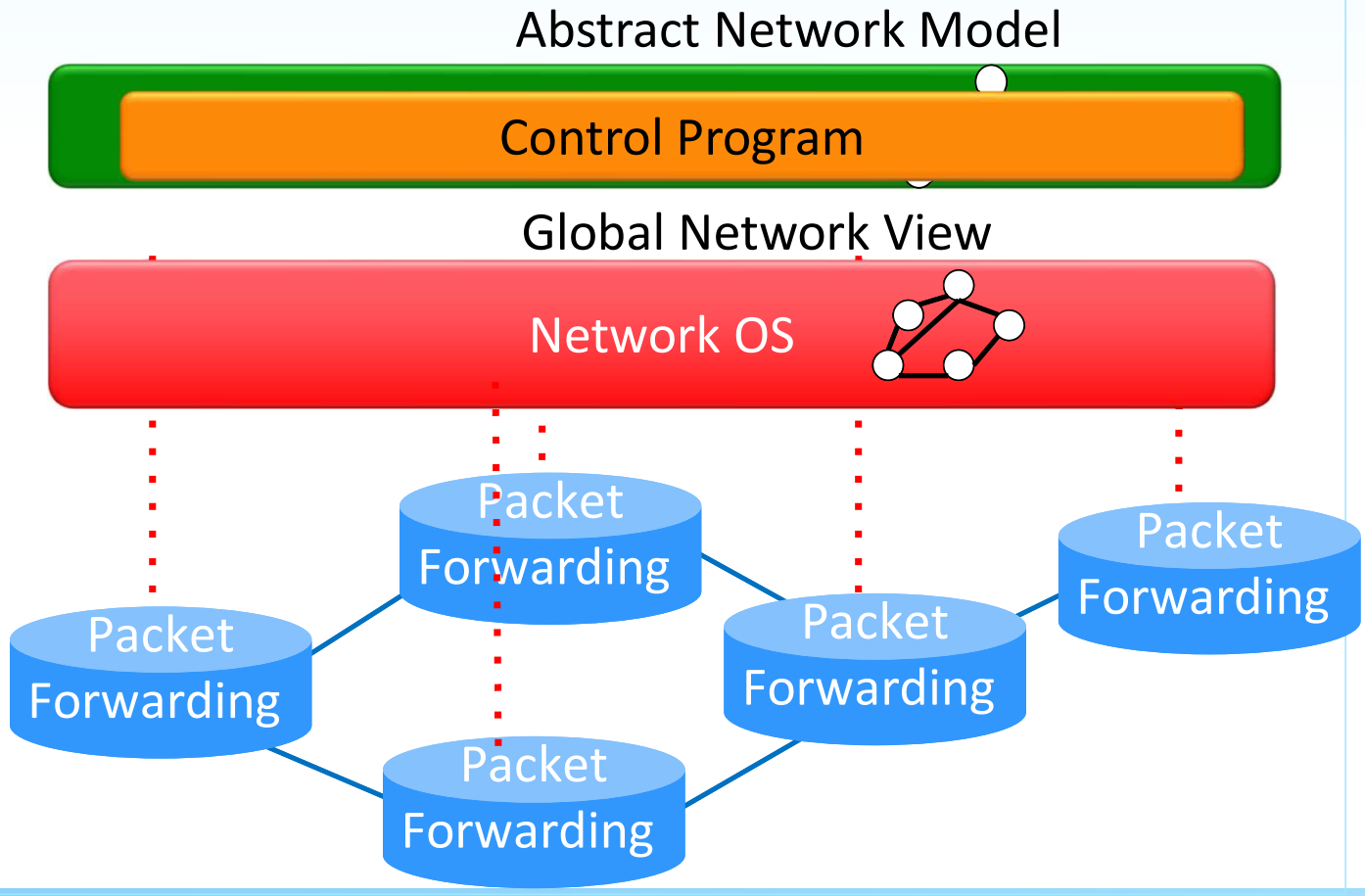
Credited to Prof. Scott Shenker (UC, Berkeley) and Prof.Nick Mckeown (Stanford University)

Software Defined Network – Network Abstraction

Specifies Behavior

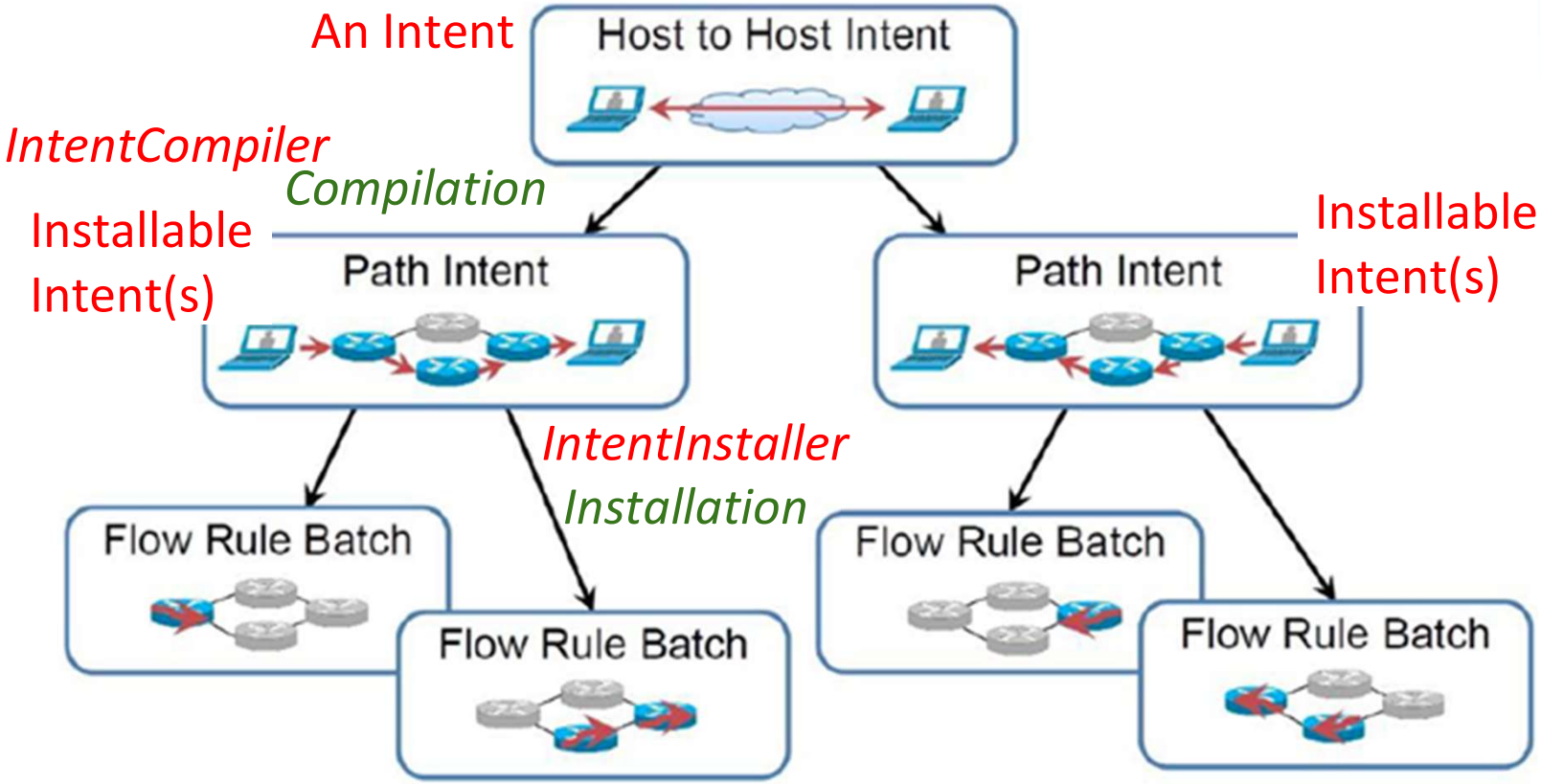
Compiles to Topology

Transmits to Switches



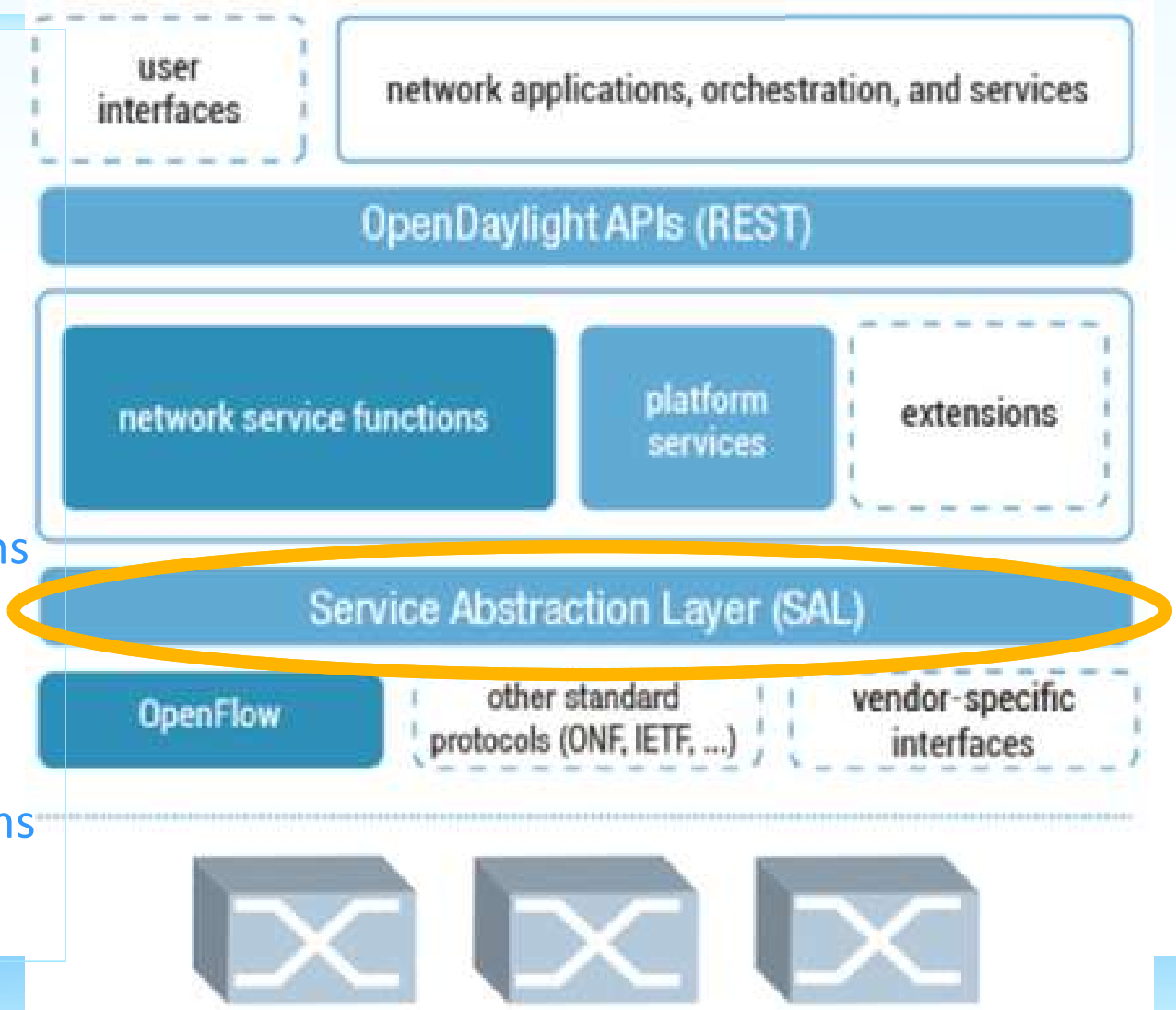
Intent Compilation and Flow Rule Installation

- Intents are ultimately compiled down into a set of **Flow Rules:**



OpenDaylight: A Model-View-Control platform

- Core of OpenDaylight platform
  - Model-Driven Service Abstraction Layer (MD-SAL).
- Objects (or Models): represent
  - Underlying network devices
  - Network applications
- **Model: YANG**
  Model data, RPC and notifications
  - Processed within SAL.
- **View: REST API**
- **Control (SAL): Java Core**
  handle data changes, notifications and RPC callbacks



Summary: ODL vs. ONOS

- Both written in Java
- Both designed for modular use
  - with a customizable infrastructure
- Every ONOS partner is also an ODL member

- Some Key Differences:
  - Cloud Provider vs. Carrier-grade networks
  - Legacy vs. “Pure” SDN
  - Corporate initiated vs. Academic initiated.

- AT&T: using
  - ONOS as a local controller
  - ODL as the basis for global SDN controller.