CS6045: Software Defined Networks

Prof. Krishna Moorthy Sivalingam
Dept. of CSE, IIT Madras
Chennai 600036
Krishna.sivalingam@gmail.com,
skrishnam@iitm.ac.in

Sep. 2015

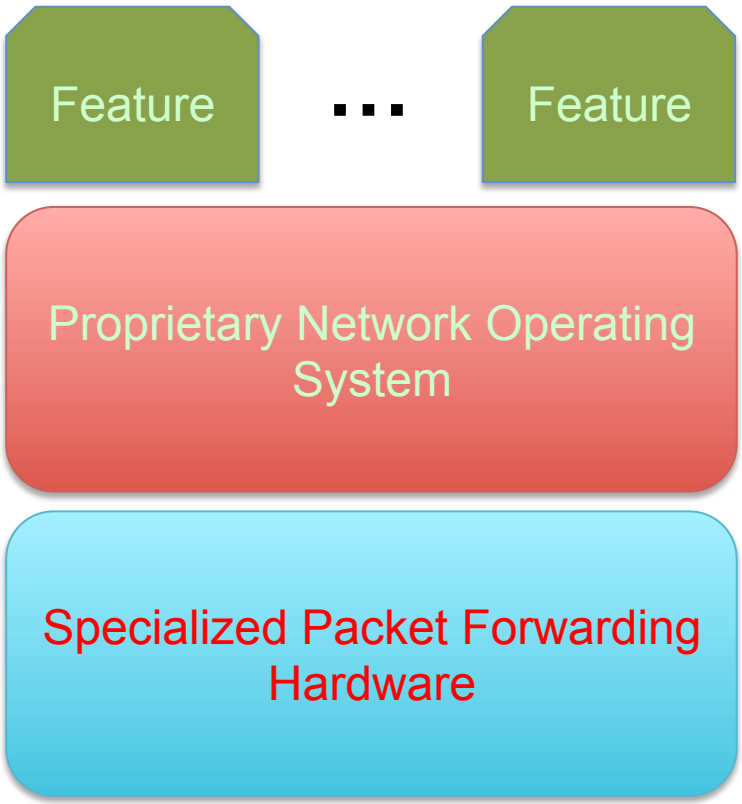
Overview

- Software Defined Networking
  - Highlights and Challenges Ahead

Ossification of the Internet

- Difficult to make significant changes to protocols/mechanisms in the core Internet
- Most major changes are done at end-host protocols/mechanisms
- Programmatic interfaces not available
- Policy/configuration consistency is a pain
- Vendors reluctant to use disruptive technologies
- Cost of upgrade is significant, post-investments
- **Innovation in Internet is slowing down?**

Ossified Network



- Features: Routing, Management, Access Control, VPNs
- OS/Features: MLOC, 5000+ RFCs, Interoperability Issues
- Hardware: ASICs, Power consuming
- "Mainframe" Days of Networking

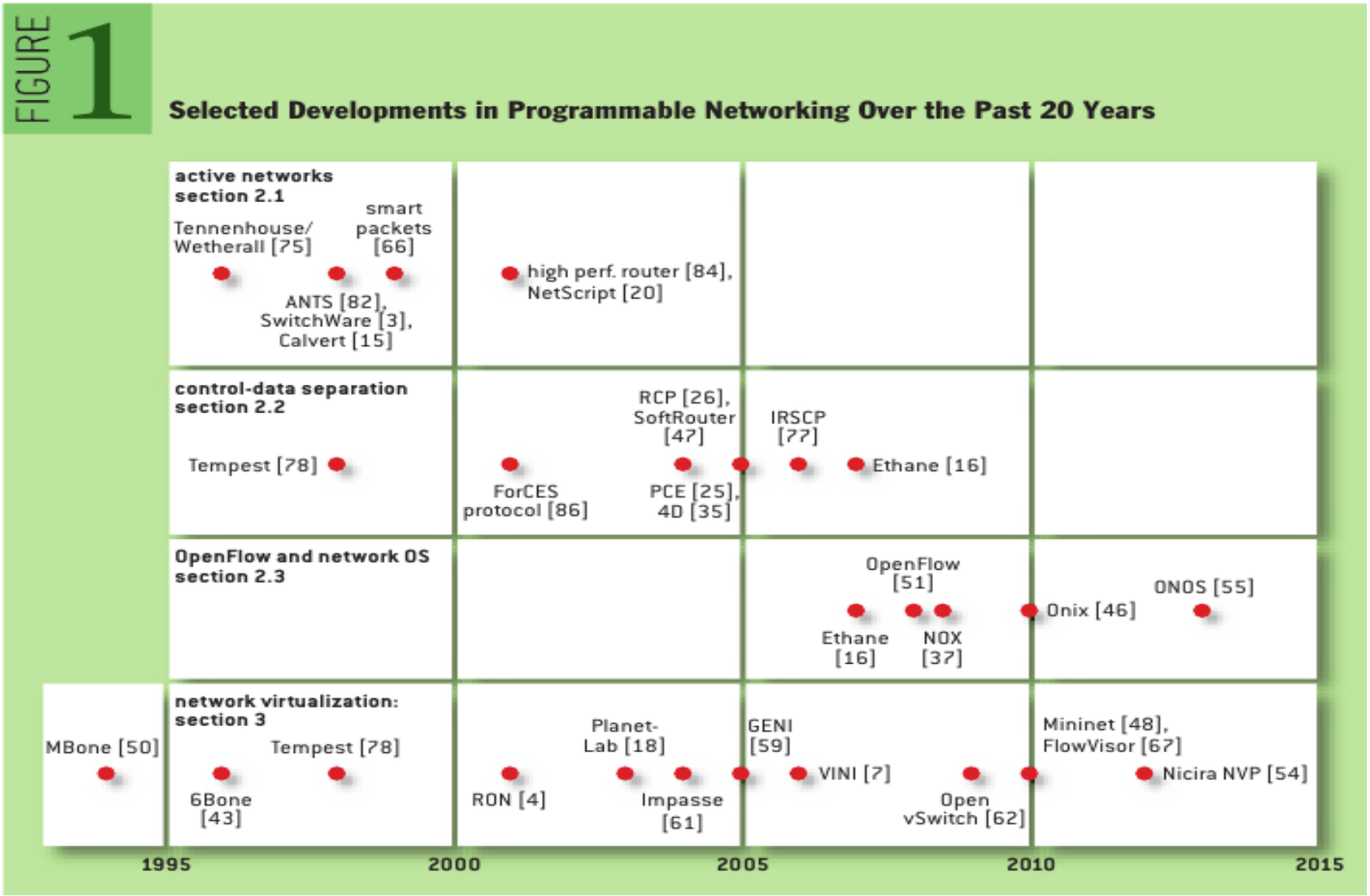
Next Generation Network Design

- Future Internet Design (FIND): 2006-2009
- European FIRE TestBed
- Future Internet Architecture: 2010 –
  - **Named Data Networking (CCN or ICN)**
  - Mobility First
  - Nebula
  - eXpressive Internet Architecture (XIA)
  - ChoiceNet
- Global Environment for Network Innovations (GENI) Testbed
- **Software Defined Networking (SDN)**

Software Defined Networking

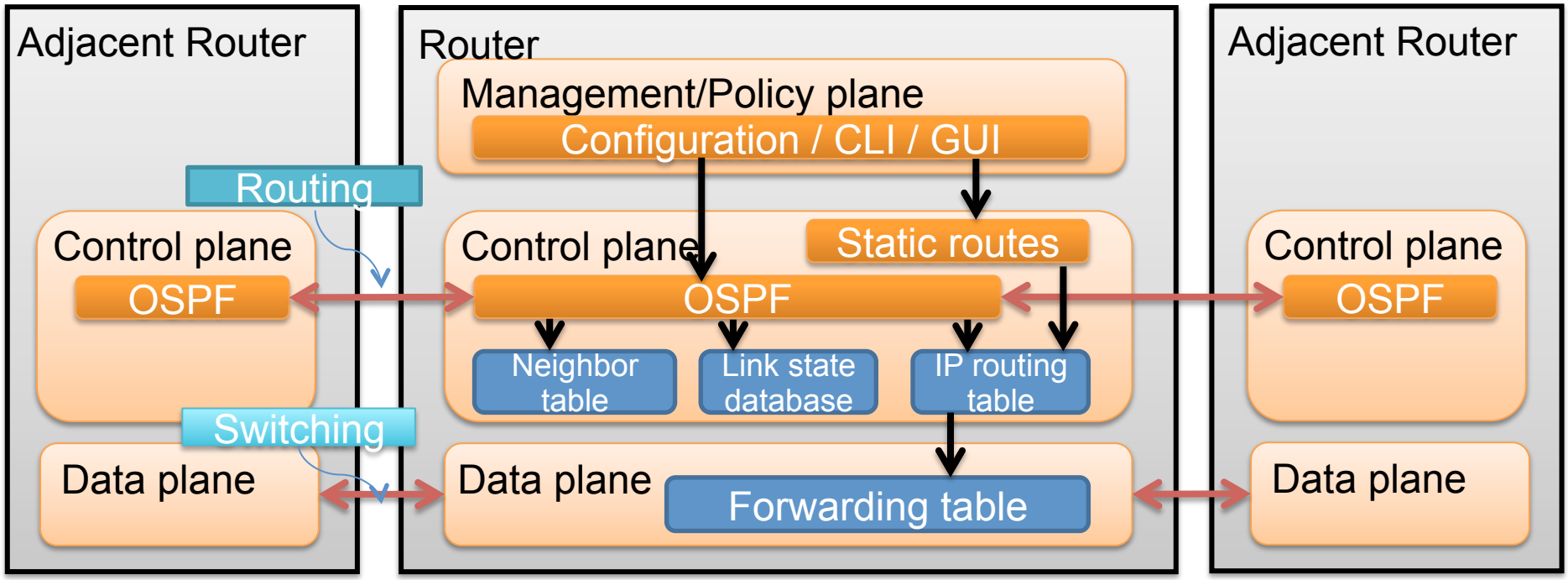
- SDN Architecture
  - OpenFlow
- SDN Tools/Products
- SDN Applications
- SDN Research Challenges

History of Programmable Networks



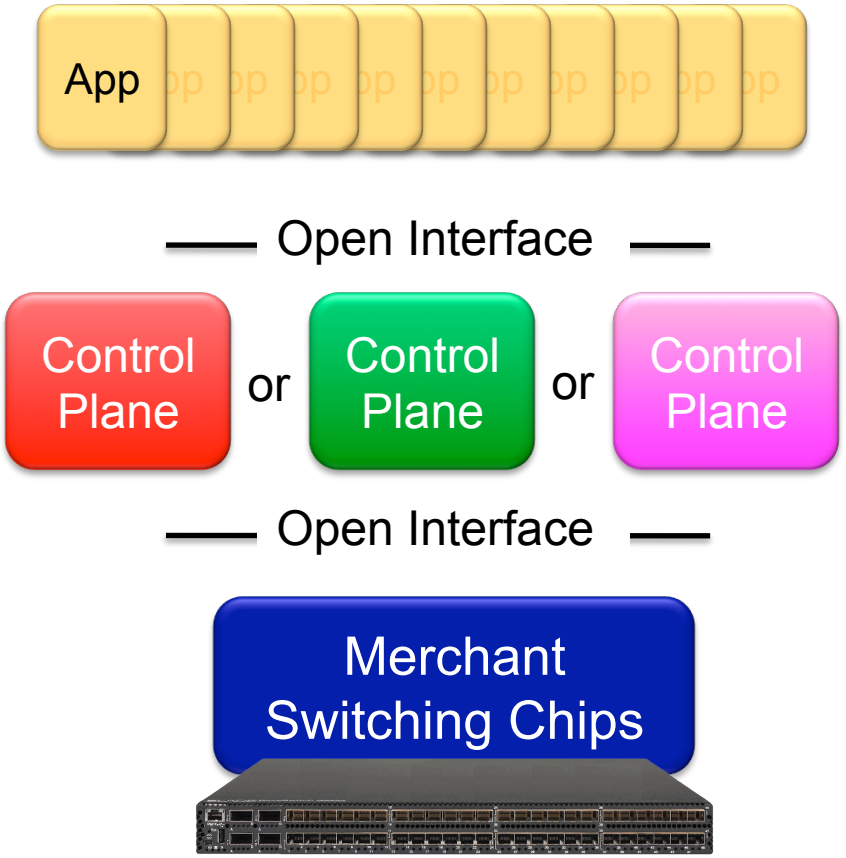
Source: "ACM Queue - The Road to SDN, An intellectual history of programmable networks "

Traditional Network Router



Source: www.cs.nthu.edu.tw/~ychung/slides

Source: Nick McKeown's ONS-2012 Talk



Vertically integrated
Closed, proprietary
Slow innovation

Horizontal
Open interfaces
Rapid innovation

Definition of SDN

"Centralization of network control via":

- Moving of control logic from network elements to a (logically) centralized controller, that
- "Enables automation and orchestration of network services via"
- **"Open programmatic interfaces"**

Advantages:

- Efficiency
- Scale
- Innovation

From SDNCentral.com (http://www.slideshare.net/SDNCentral/sdnu-101-final)

Vision (Shenker, McKeown et al.)

Hardware

- Cheap, Commodity Items
- Implement Flow Handling

Software

- Logically Centralized Controller
- Frequent Updates for added Functionality
- Decoupled from Hardware

Functionality

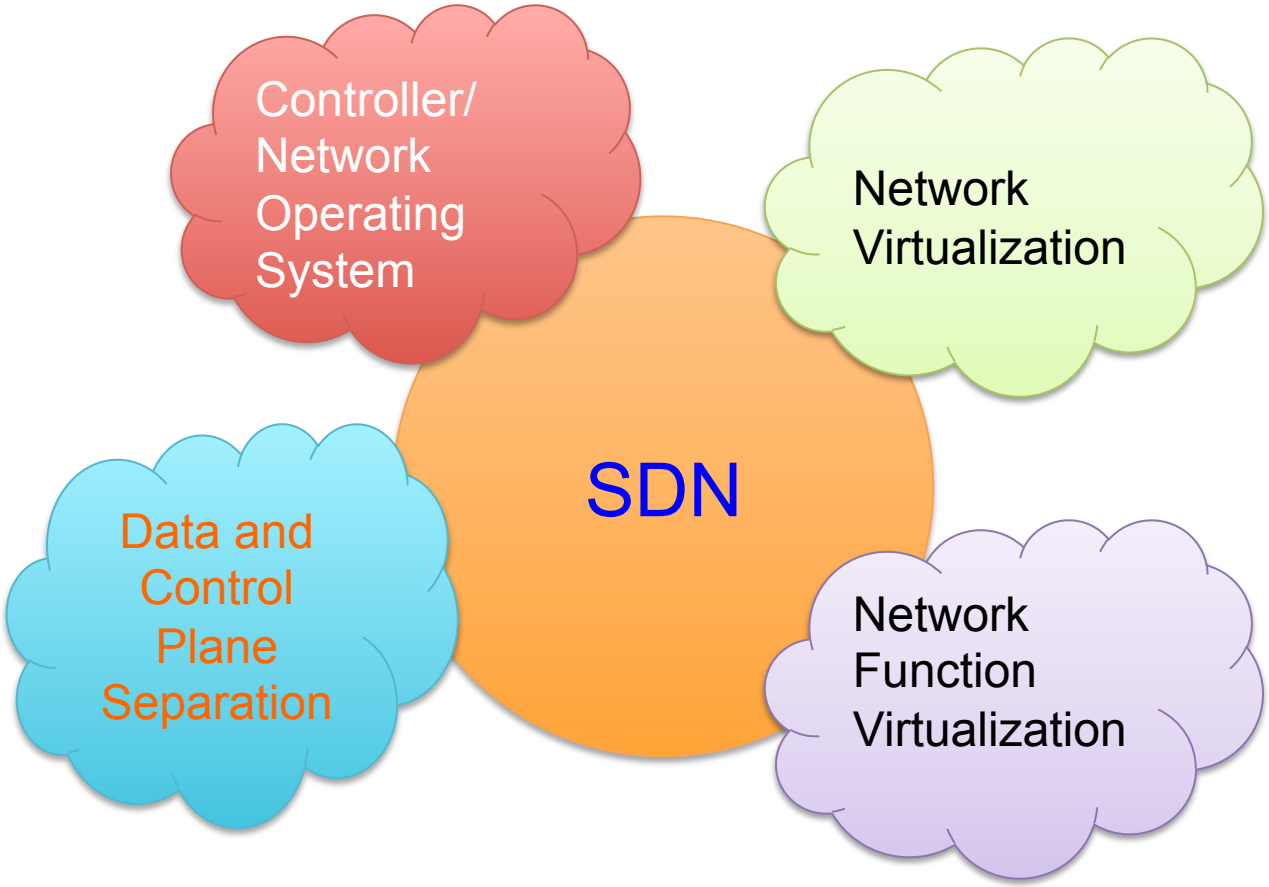
- Mostly driven by Software
- Edge Devices are more important
- Control Program

SDN's Primary Goals

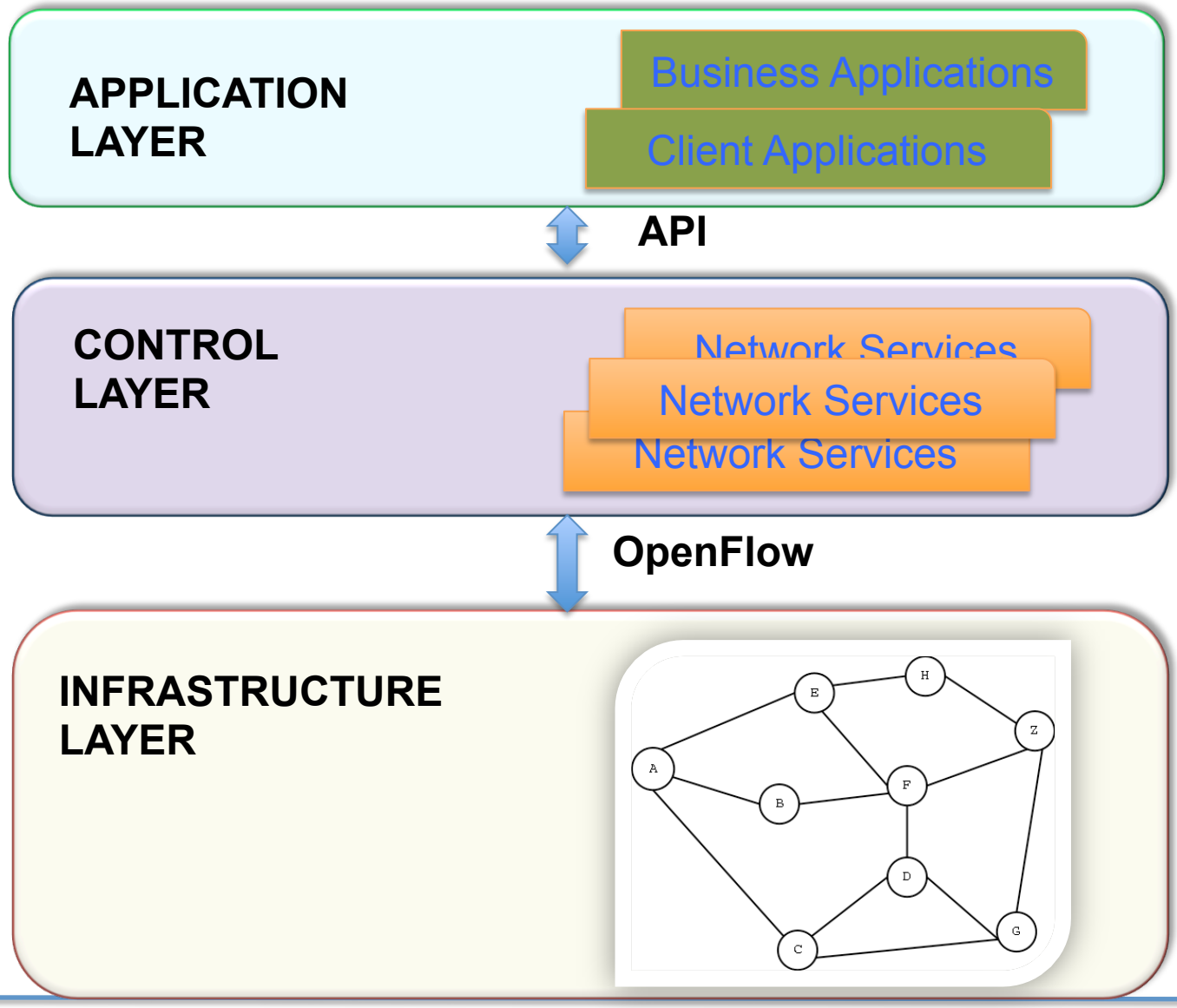
1. Formally verify that our networks are behaving correctly.
2. Identify bugs, then systematically track down their root cause.

Source: Nick McKeown's ONS-2012 Talk

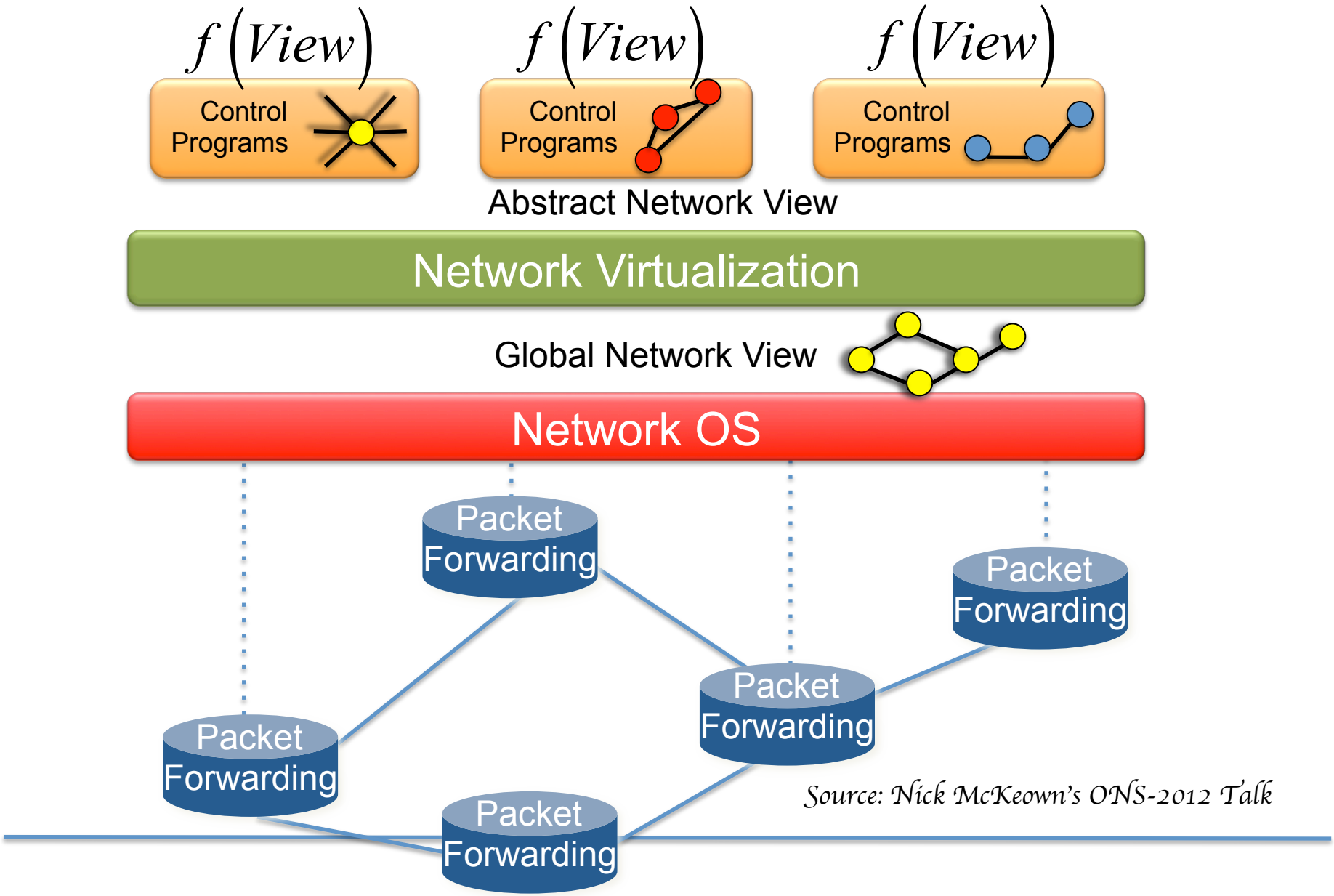
SDN Concepts



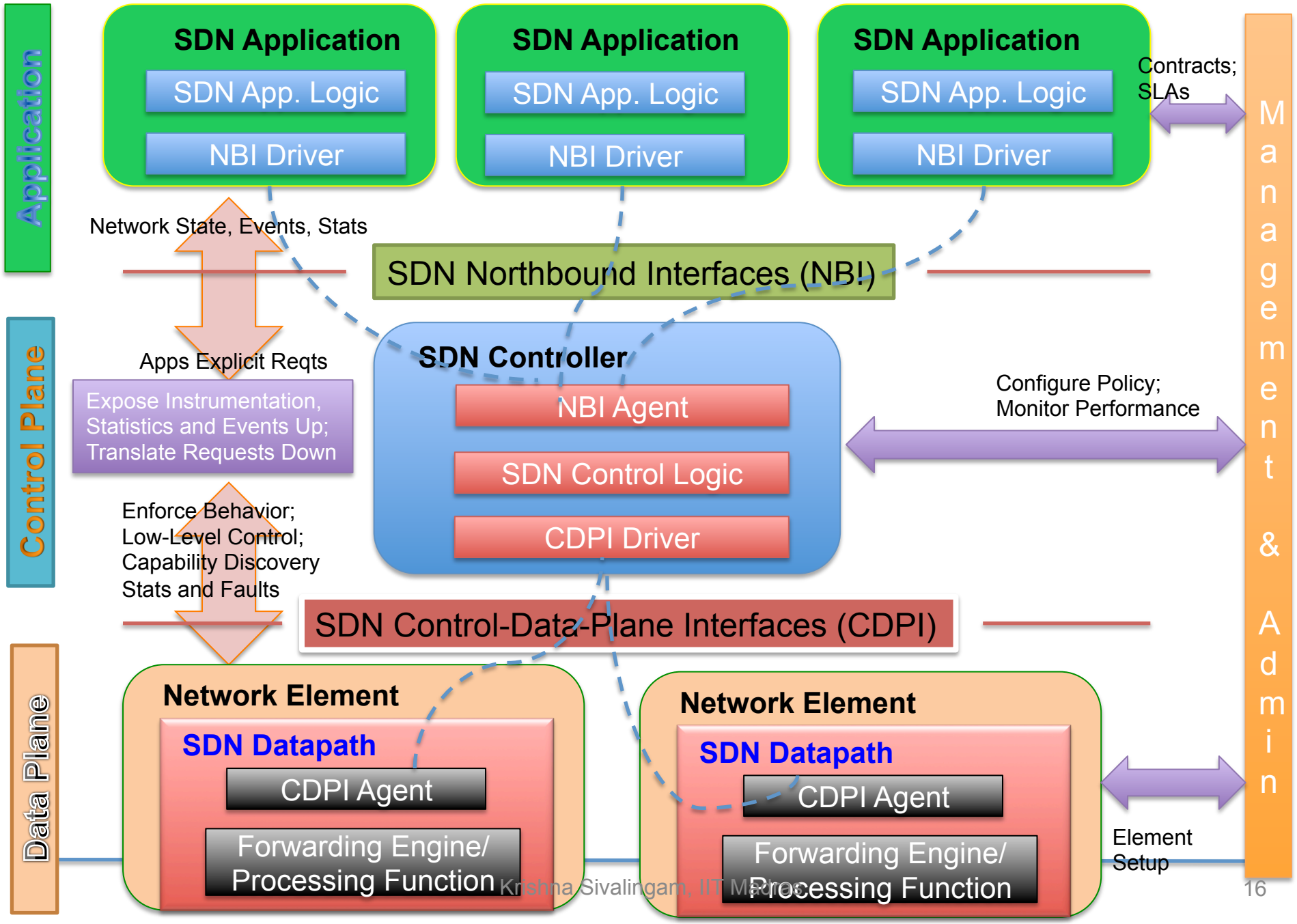
SDN Simplified View



Software Defined Network (SDN)



Source: ONF, SDN Arch. Overview, Dec 2013



Applications of SDN

Data Center Networks	• Improved Traffic Management, Policy Enforcement and Energy Management • Google's success (B4) with WAN Data Centers
Enterprise Networks	• Easier to implement different policies • Provide for testbeds in University settings
Wireless Access Networks	• Cellular (Softcell) and WiFi (Odin)
Optical Networks	• Improve Transport Network Control • Deploy new services using Virtualization
Home and Small Business Networks	• Home Network Recorder to generate logs • Remote Monitoring and Control to ensure improved security

CONTROL AND DATA PLANE SEPARATION

SDN: Abstractions in Control Plane

Forwarding Abstraction

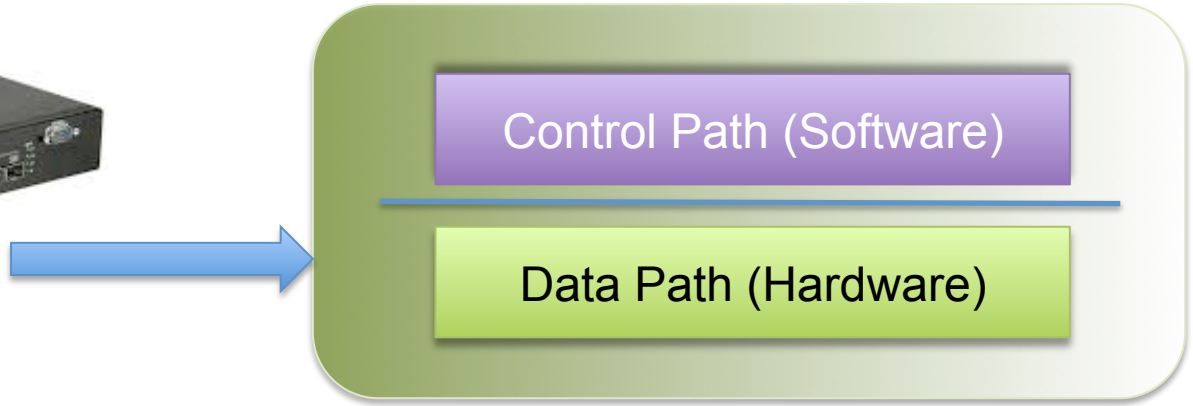
- Compute Forwarding State
  - Based on entire network topology
  - For all router/switches
  - OpenFlow for forwarding: <header, action>

Network State Abstraction

- Compute Global Network View
  - Computed by Network Operating System (Controller), running on servers
  - "View" obtained from Routers/Switches
  - Compute Device Configuration = Fn(View)
  - Send Configuration information to R/S

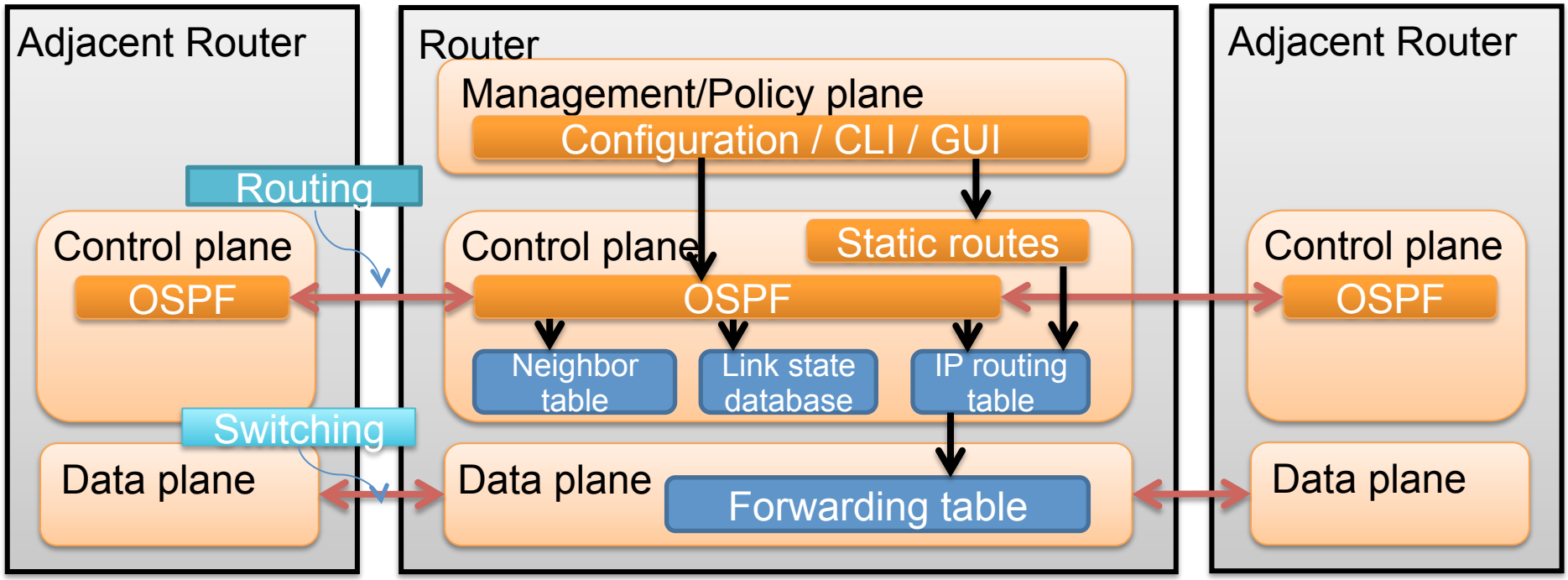
Traditional Network L2 Switch

- Control Plane
- Data Plane
- Management Plane



Traditional Network L3 Router

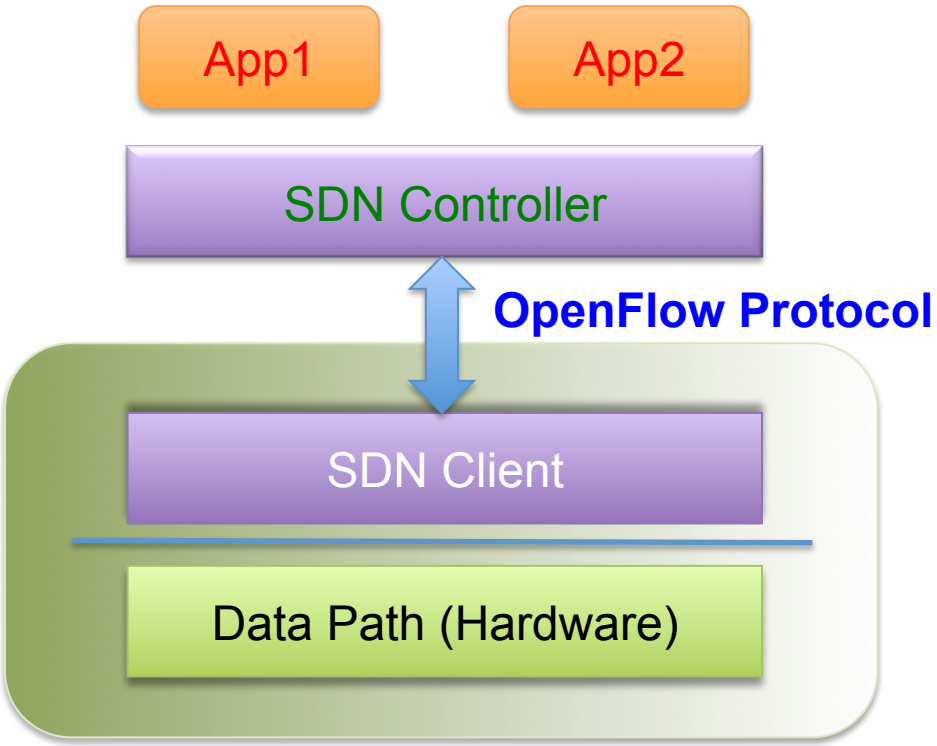
- Management plane / configuration
- Control plane / Decision: OSPF (Open Shortest Path First)
- Data plane / Forwarding



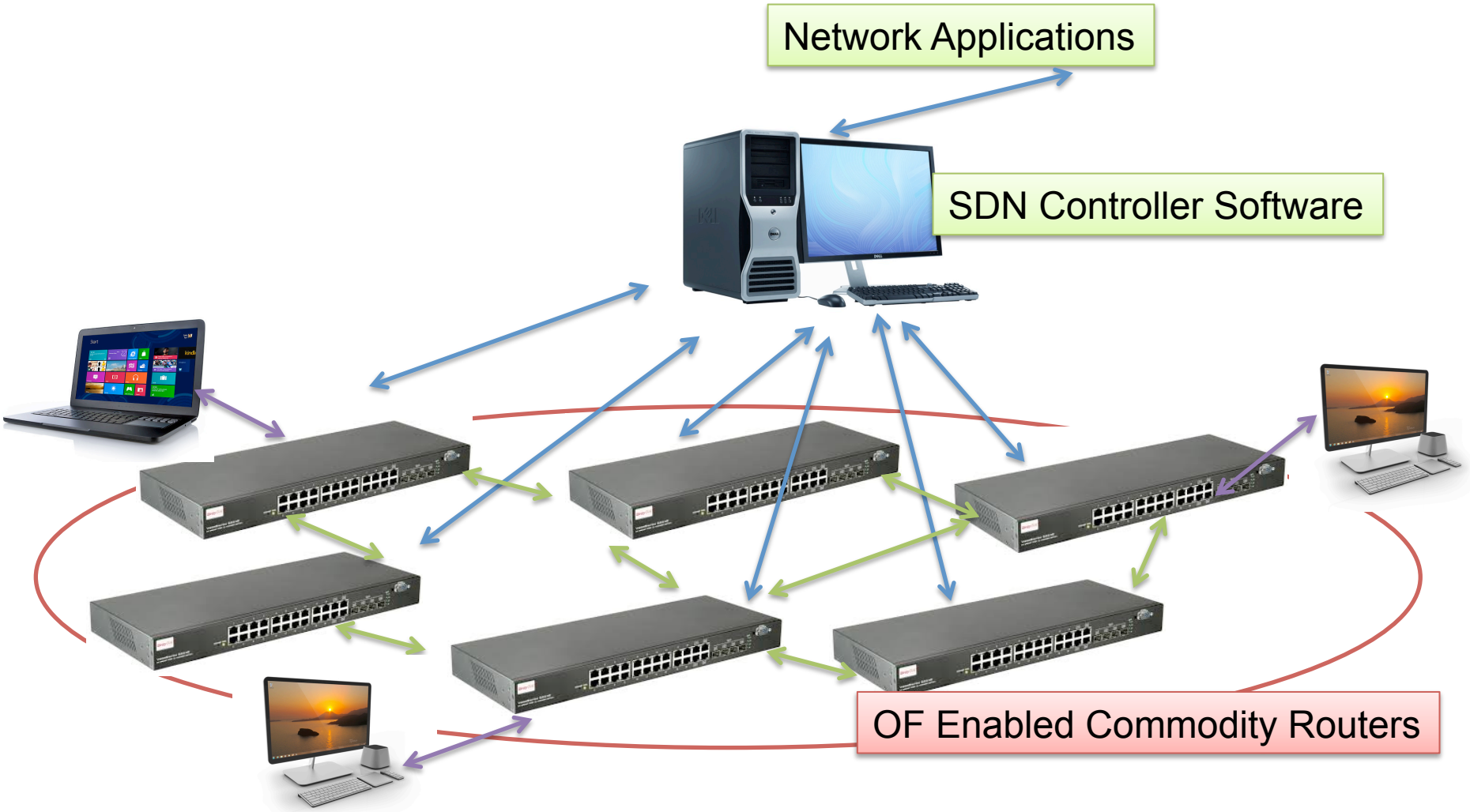
Source: www.cs.nthu.edu.tw/~ychung/slides

OpenFlow based Switch

- Separation of Control and Data Planes
- Control Plane implemented in a separate Controller (in Software)

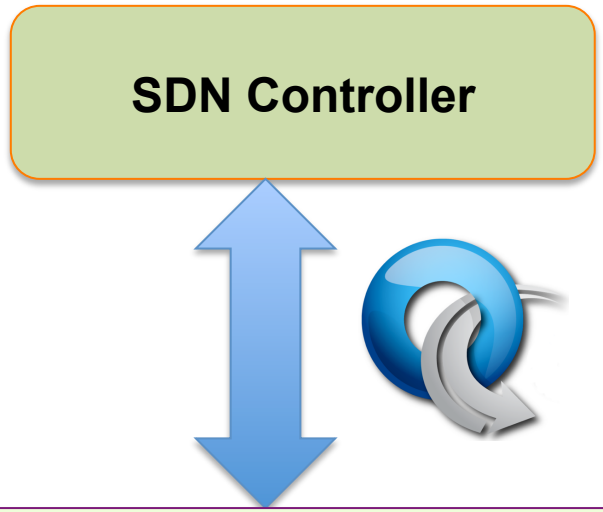


OpenFlow Architecture



OpenFlow Protocol

- Enables interface to the Switch's (Physical or Virtual) Forwarding Engine (FE)
- Uses "Flows" to identify network traffic
- Flow Table in Switch filled by Controller
  - Low-level granularity based flow handling
- FE implements "CPU"-like instructions
- Initially used for Ethernet switches; extensible to other switches/routers
- Current version: 1.4 (v1.3 commonly used)



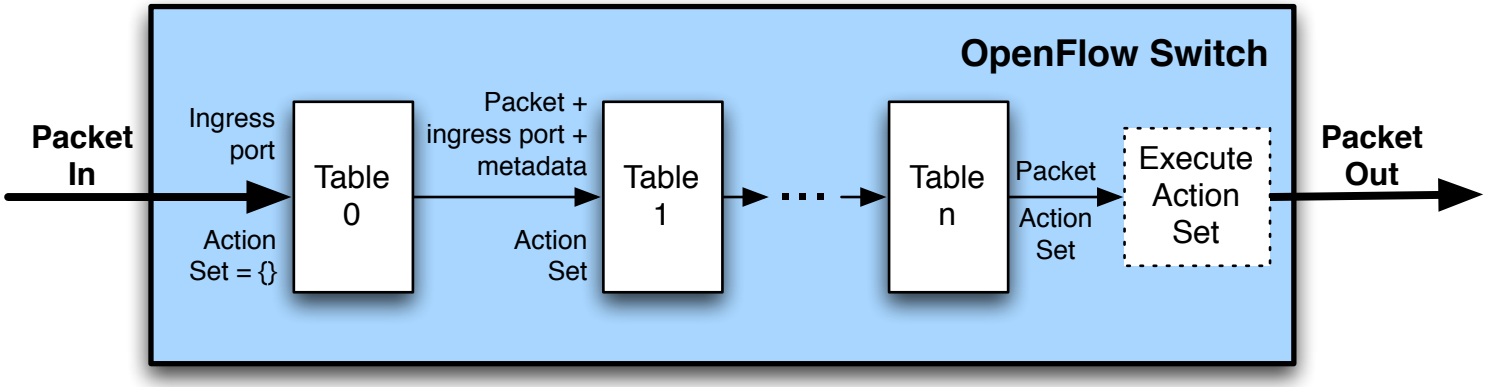
- Individual Flows;
- Aggregated Flows;
- Flow-to-entry mapping
  - one-to-one
  - many-to-many

OpenFlow Enabled Network Element

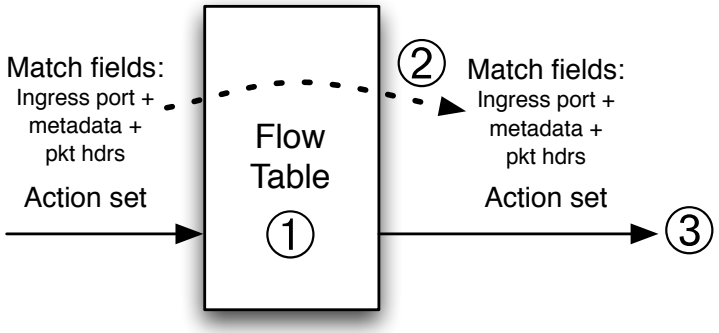
Flow Table

MAC src	MAC dst	IP src	IP dst	TCP dport	…	Action	Stats.
*	10:20:*	*	5.6.7.8	25		Port 1	250
						Drop	380
						Local	100
						Ctrler.	1

OpenFlow Switch Processing



(a) Packets are matched against multiple tables in the pipeline



① Find highest-priority matching flow entry

② Apply instructions:
- i. Modify packet & update match fields (apply actions instruction)
- ii. Update action set (clear actions and/or write actions instructions)
- iii. Update metadata

③ Send match data and action set to next table

(b) Per-table packet processing

Source: ONF's OpenFlow Specification Document

OpenFlow v1.0

- Released in 2009
- **Single Flow Table** per Switch/Router
- **Twelve** fields for matching a FT entry
  - Switch input port
  - VLAN ID and VLAN priority
  - Ethernet Source and Destination Address
  - Ethernet frame type
  - IP Source and Destination Address
  - IP Protocol
  - IP Type of Service (TOS) Bits
  - TCP/UDP Source and Destination Ports

OpenFlow v1.0 ACTIONS

- Required Actions: Forward (Output) or Drop
- Physical port on Switch/Router
- Special Virtual Ports
  - LOCAL: send to switch's Local OF control software, e.g. messages from controller to be processed
  - ALL: flood a packet on all but ingress port of packet
  - CONTROLLER: send to controller
  - IN_PORT: send packet back on input port
  - TABLE: packets sent by controller forwarded either on specified output port in PACKET_OUT message or to the Flow Table
  - FLOOD: Similar to ALL, except that it sends only on links of STP tree
  - NORMAL: Send to legacy forwarding logic (Hybrid switch)

OpenFlow v1.0 ACTIONS, contd.

- Optional Actions
  - Enqueue: Selects a specific queue of output port
  - Modify: Update some fields of packet headers
- A flow table entry can specify multiple Actions for a given packet
  - Processed in the order specified in the entry

OpenFlow Controller Apps

- Controller provides programming interface
- Network Manager writes programs to update flow rules
- Same datapath can operate as broadcast medium, VLAN switch, router, firewall
  - Simultaneously, for different traffic
- Coordination among multiple paths needed for network-wide design
  - Single point of application and Agile

OpenFlow Controllers, contd.

Name	Lang	Platform(s)	License	Original Author	Notes
OpenFlow Reference	C	Linux	OpenFlow License	Stanford/ Nicira	not designed for extensibility
NOX/POX	Python, C++	Linux	GPL	Nicira	actively developed
Beacon	Java	Win, Mac, Linux, Android	GPL (core), FOSS	David Erickson (Stanford)	runtime modular, web UI framework, regression test framework; Basis for Floodlight
Floodlight	Java	Mac, Linux	Apache	BigSwitch	Core of commercial product from BigSwitch
Maestro	Java	Win, Mac, Linux	LGPL	Zheng Cai (Rice)	
Trema	Ruby, C	Linux	GPL	NEC	includes emulator, regression test framework
OpenDayLight	Java	Linux and others	EPCL	Consortium	SDN Controller: Helium released in 2015

SDN Controllers

- **NOX/POX**
- **Ryu**
- **Floodlight**
- **OpenDaylight**
- **ONOS**
- Pyretic
- Frenetic
- Procera
- RouteFlow
- Trema

NOX

- First generation OpenFlow controller
  - open-source, mature, stable (noxrepo.org)
  - Developed by Nicira/Stanford/ICSI and donated in 2008
- Flavors of NOX
  - NOX-Classic: C++/Python. Not supported.
  - NOX (Current): C++ only
- Users write controller logic in C++
- Supports OF v1.0, 1.1, 1.2, and 1.3
- Controller registers for events (from switch)
  - Event handlers take care of events (e.g. PacketIn)

POX

- "Cousin" of NOX, designed for academia and research
- NOX in Python
  - Supports only OF 1.0
- Well maintained and supported
- Performance is poor compared to NOX
- Useful for rapid prototyping and experiments

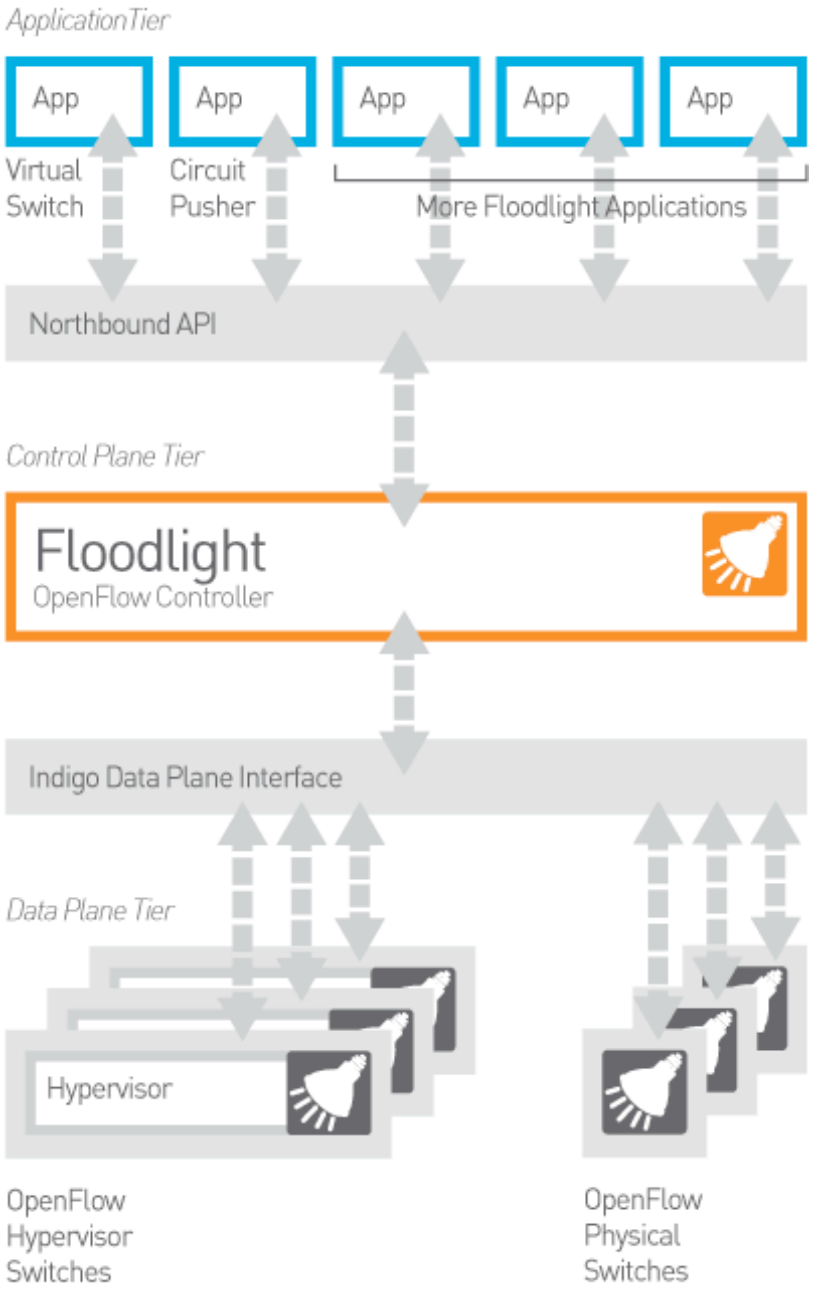
Ryu

- http://osrg.github.io/ryu/
- Open source under Apache 2.0 license
- Ryu (ree-yooh) means “flow” in Japanese
- Component-based SDN Framework
- Supports 1.0, 1.2, 1.3, 1.4, 1.5 and Nicira Extensions
- Also supports other SBIs such as NETCONF

Floodlight

- Java-based open-source controller software
- Forked from Stanford's Beacon Switch
  - Maintained by Big Switch networks
- Works with Physical and Virtual Switches that support OpenFlow
- Can handle mixed OpenFlow and non-OpenFlow networks
- Module loading system for extensions and enhancements
- Support for **production-level OpenStack/Multi-tenant** cloud orchestration platform

BigSwitch's Floodlight (Evolved from Stanford's Beacon) SDN Controller: http://www.projectfloodlight.org/floodlight/

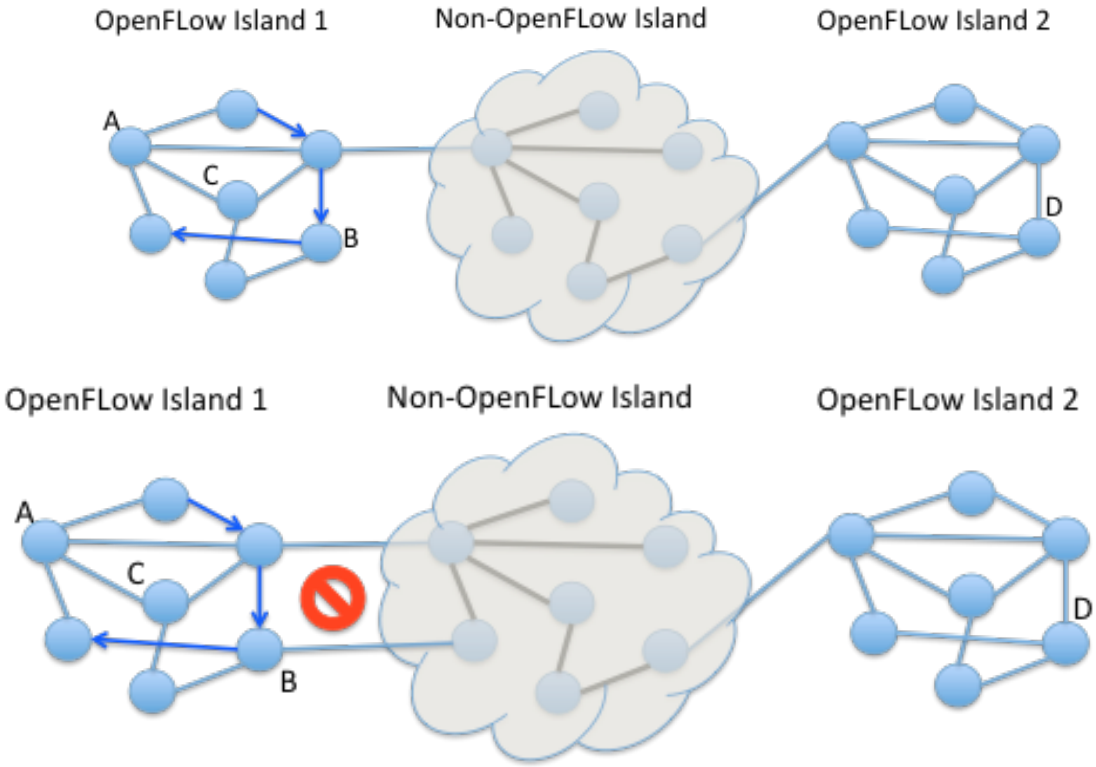


Floodlight Compatible Switches

- As of Dec. 2014
- **Virtual Switches**
  - Open vSwitch (OVS)
  - ofsoftswitch
- **Hardware Switches**
  - Arista 7050
  - Brocade MLXe, Brocade CER, Brocade CES
  - Dell S4810, Dell Z9000
  - Extreme Summit x440, x460, x670
  - HP 3500, 3500yl, 5400zl, 6200yl, 6600, and 8200zl (the old-style L3 hardware match platform)
  - HP V2 line cards in the 5400zl and 8200zl (the newer L2 hardware match platform)
  - Huawei openflow-capable router platforms
  - IBM 8264
  - Juniper (MX, EX)
  - NEC IP8800, NEC PF5240, NEC PF5820
  - NetGear 7328SO, NetGear 7352SO
  - Pronto (3290, 3295, 3780) - runs the shipping pica8 software

Floodlight Supported Topologies

- Supports two default reactive packet forwarding applications

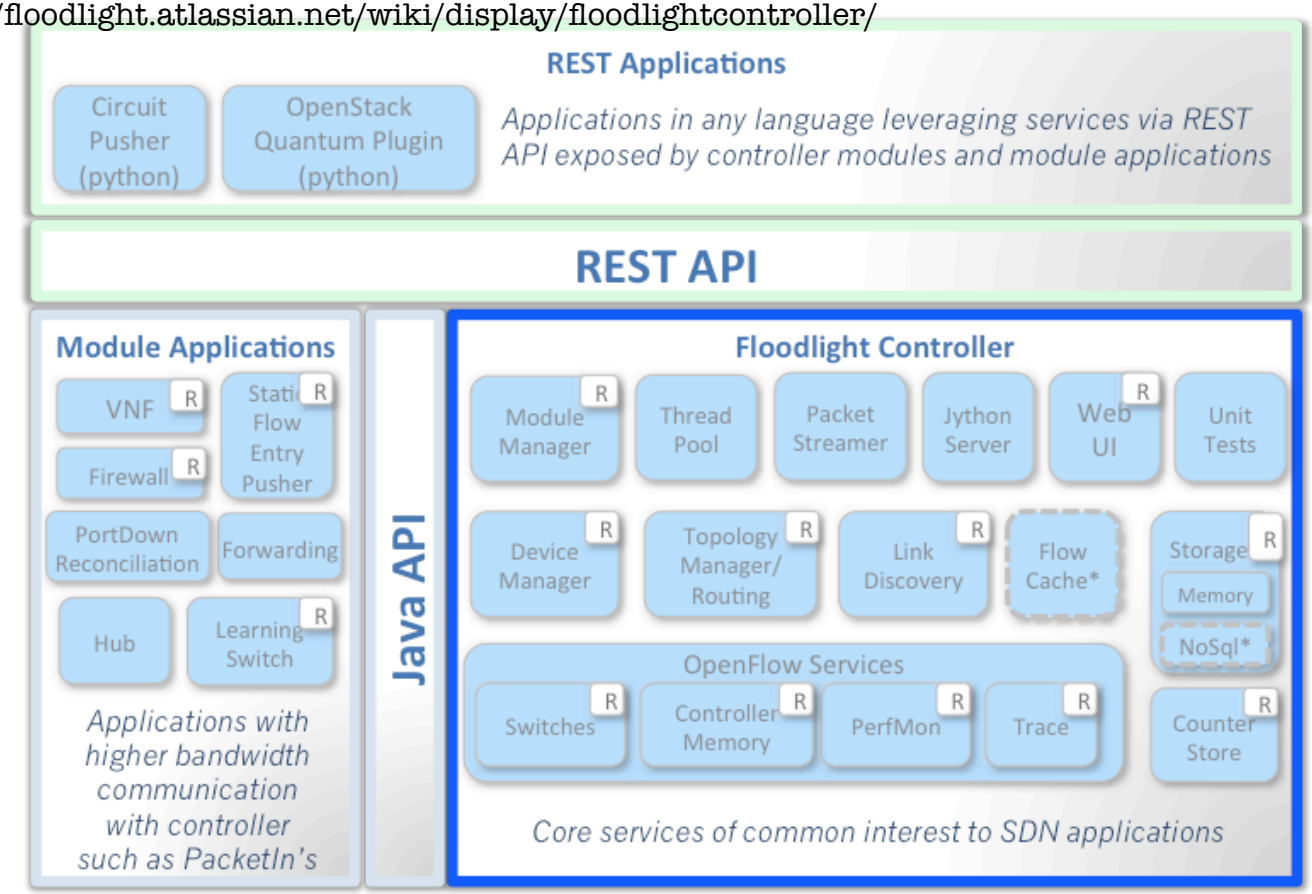


Source: https://floodlight.atlassian.net/wiki/display/floodlightcontroller/Supported+Topologies

Floodlight Controller

- FL contains Controller and few default apps
- REST: Representational State Transfer
- http://www.restapitutorial.com/lessons/whatisrest.html

Source: https://floodlight.atlassian.net/wiki/display/floodlightcontroller/



* Interfaces defined only & not implemented: FlowCache, NoSql

RESTful API

- Software Architecture style for Web applications
- REST based Client-Servers communicate typically using HTTP GET/POST/PUT/DELETE
- REST interfaces deal with collection of resources identified by URIs
  - Resource: /people/abcd
  - Action: DELETE /people/abcd
- Data represented using Java Script Object Notation (JSON)/XML etc.

ACL REST API Example

- Add rule:
  - curl -X POST -d '{"src-ip":"10.0.0.1/32","dst-ip":"10.0.0.2/32","action":"deny"}' http://<controller_ip>:8080/wm/acl/rules/json
- List rules:
  - curl http://<controller_ip>:8080/wm/acl/rules/json | python -mjson.tool
- Delete rule:
  - curl -X DELETE -d '{"ruleid":"1" }' http://<controller_ip>:8080/wm/acl/rules/json
- Remove all rules:
  - curl http://<controller_ip>:8080/wm/acl/clear/json

Firewall and other REST APIs

- Firewall
- State Flow Pusher

Applications on FloodLight

- OpenStack Quantum Plugin
  - Quantum exposes networking-as-a-service model via REST
- Virtual Switch
  - Network virtualization application
  - Provides Layer 2 (MAC) based network virtualization, creation of multiple logical layer 2 networks in a single layer 2 domain
- ACL (stateless FW)
  - Enforces ACL rules (Access Control List) on OpenFlow enabled switches
- Circuit Pusher
  - Creates a bidirectional circuit, i.e., permanent flow entry, on all switches in route between two devices based on IP addresses with specified priority
- Source: http://www.projectfloodlight.org/applications/

Building New Applications

- REST API
- Module Applications
- OpenStack Quantum based

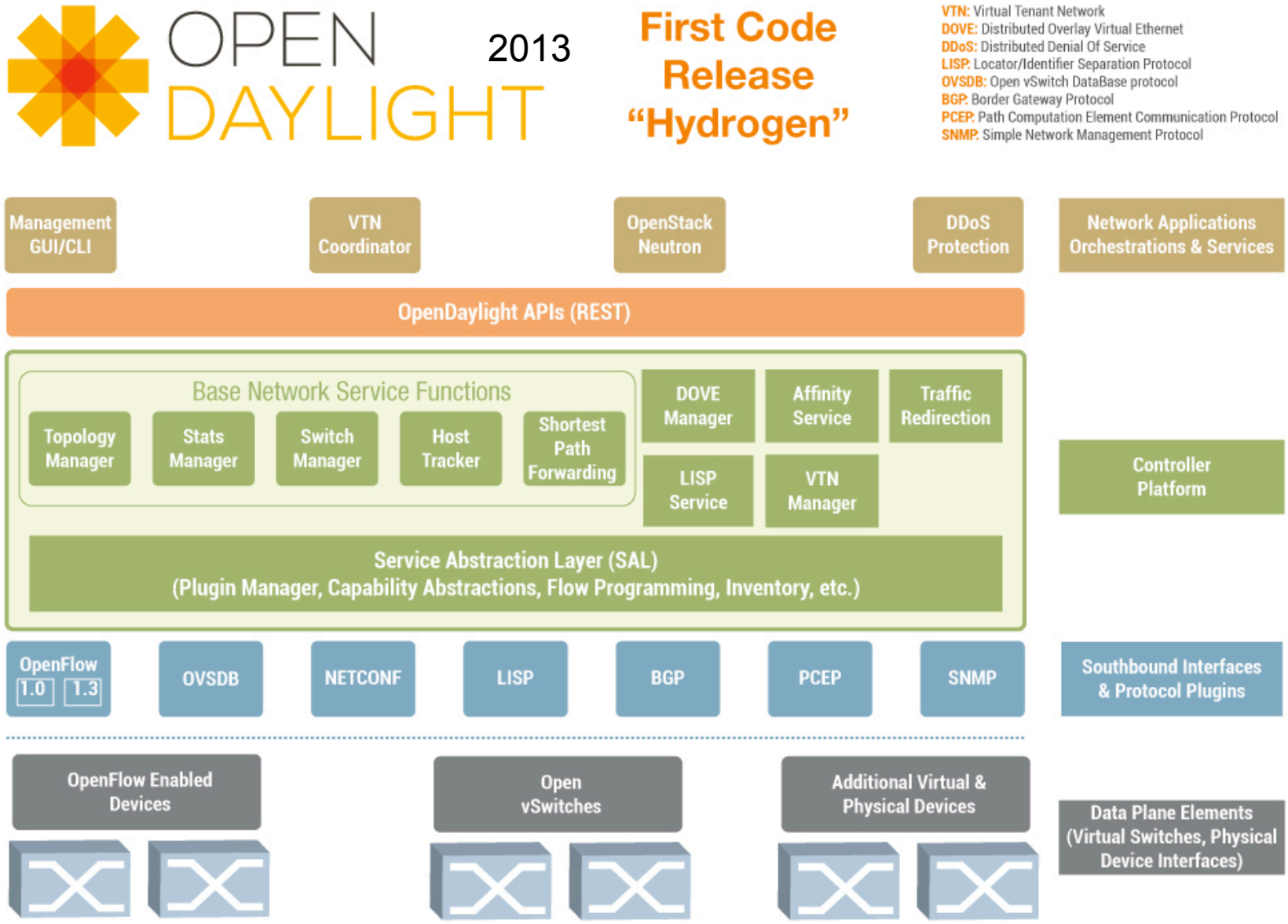
Projects on top of FL

- AD - Anomaly Detection
- Android - Floodlight Android UI
- Avior - Network management and testing GUI written in Java
- CLI - Command Line Interface (CLI) to Floodlight.
- Custom Costs Balancer - To setup custom costs on OpenFlow links cached by the Floodlight Topology module.
- DHCP - Floodlight DHCP server
- GreenMST - Implements a MST over a network.
- HAND - Controller that makes decisions using Ganglia metrics.
- KHopMetric - Floodlight module to speed up the network convergence time after a fault.
- Proxy ARP – Offers MAC address directly to a requesting host.
- QOS - Floodlight with QoS module and tools to manage QoS state in an OF network

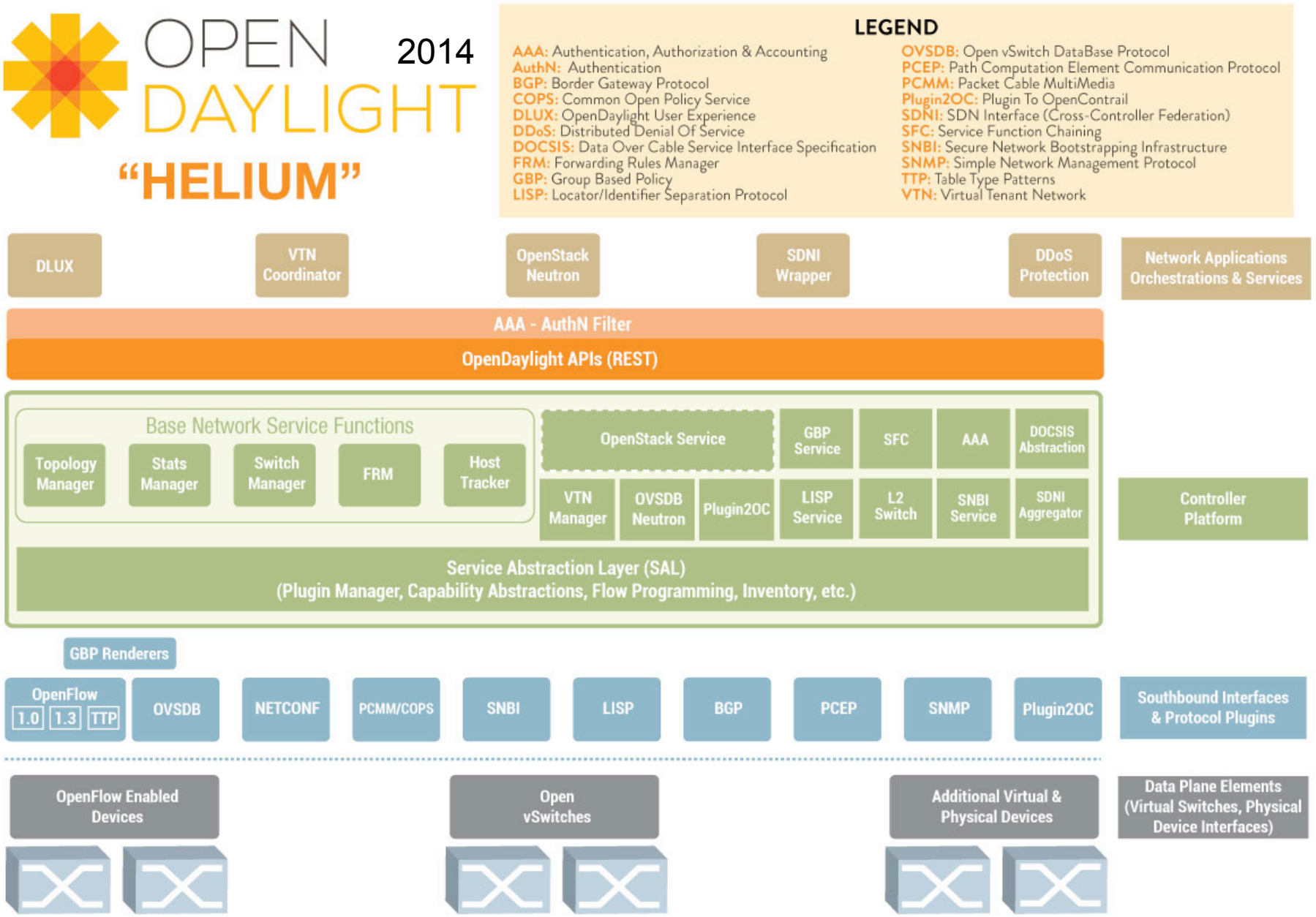
https://floodlight.atlassian.net/wiki/display/floodlightcontroller/Floodlight+Projects

OpenDayLight project

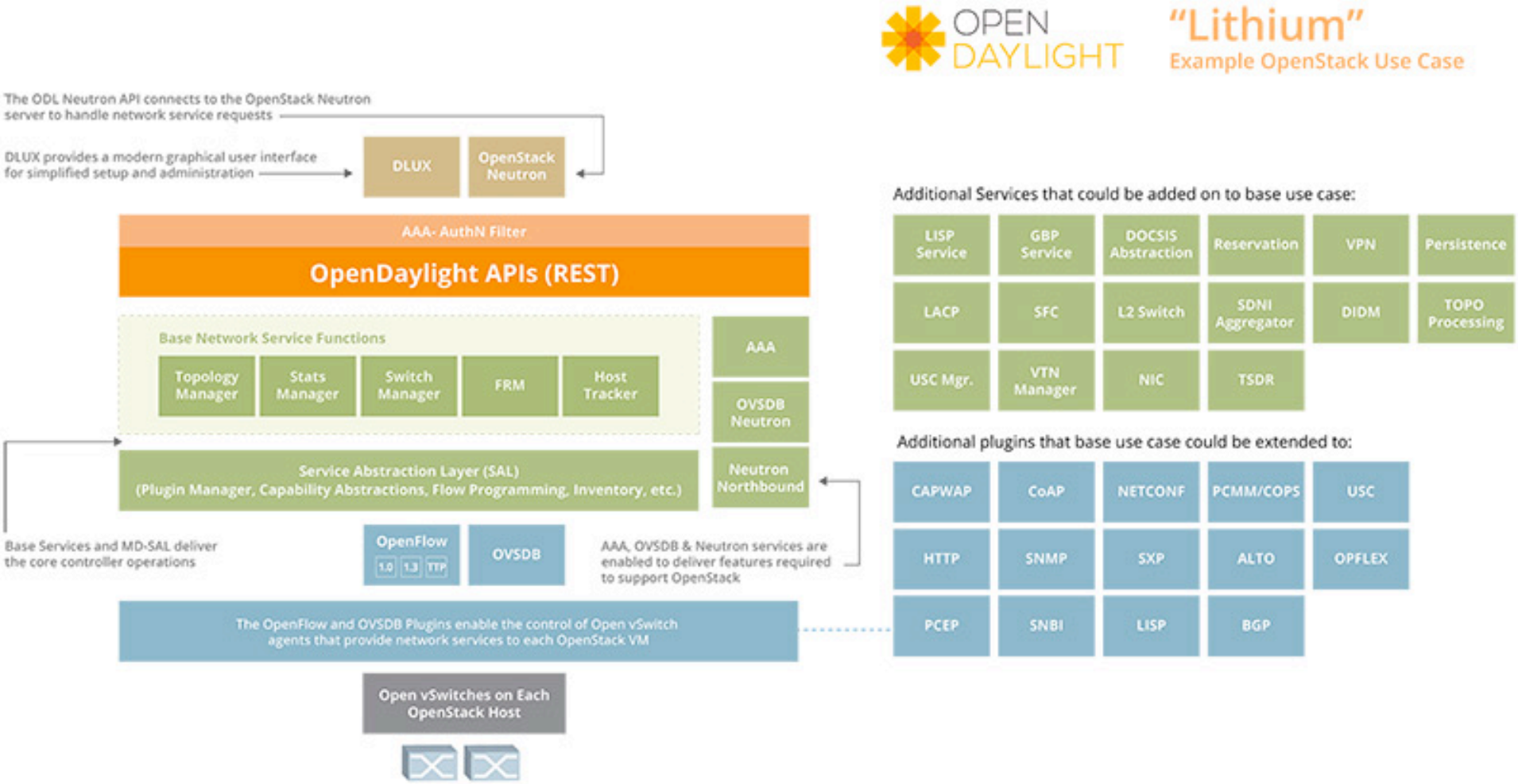
- Linux Foundation project
- Large consortium of companies including Cisco, HP, Brocade, Dell, RedHat, Intel, Citrix, etc.
- Open source license
- Three versions: Hydrogen, Helium, Lithium
- Java based interface
- Advantages:
  - Industry acceptance
  - Production-level performance
  - integration with OpenStack, cloud, etc.
- Disadvantages: Steep learning curve



Source: http://www.opendaylight.org/software/downloads


OPEN DAYLIGHT
2014
"HELIUM"
LEGEND
AAA: Authentication, Authorization & Accounting
AuthN: Authentication
BGP: Border Gateway Protocol
COPS: Common Open Policy Service
DLUX: OpenDaylight User Experience
DDoS: Distributed Denial Of Service
DOCSIS: Data Over Cable Service Interface Specification
FRM: Forwarding Rules Manager
GBP: Group Based Policy
LISP: Locator/Identifier Separation Protocol
OVSDB: Open vSwitch DataBase Protocol
PCEP: Path Computation Element Communication Protocol
PCMM: Packet Cable MultiMedia
Plugin2OC: Plugin To OpenContrail
SDNI: SDN Interface (Cross-Controller Federation)
SFC: Service Function Chaining
SNBI: Secure Network Bootstrapping Infrastructure
SNMP: Simple Network Management Protocol
TTP: Table Type Patterns
VTN: Virtual Tenant Network
DLUX
VTN Coordinator
OpenStack Neutron
SDNI Wrapper
DDoS Protection
Network Applications Orchestrations & Services
AAA - AuthN Filter
OpenDaylight APIs (REST)
Base Network Service Functions
Topology Manager
Stats Manager
Switch Manager
FRM
Host Tracker
OpenStack Service
GBP Service
SFC
AAA
DOCSIS Abstraction
VTN Manager
OVSDB Neutron
Plugin2OC
LISP Service
L2 Switch
SNBI Service
SDNI Aggregator
Controller Platform
Service Abstraction Layer (SAL)
(Plugin Manager, Capability Abstractions, Flow Programming, Inventory, etc.)
GBP Renderers
OpenFlow 1.0 1.3 TTP
OVSDB
NETCONF
PCMM/COPS
SNBI
LISP
BGP
PCEP
SNMP
Plugin2OC
Southbound Interfaces & Protocol Plugins
OpenFlow Enabled Devices
Open vSwitches
Additional Virtual & Physical Devices
Data Plane Elements (Virtual Switches, Physical Device Interfaces)

2015

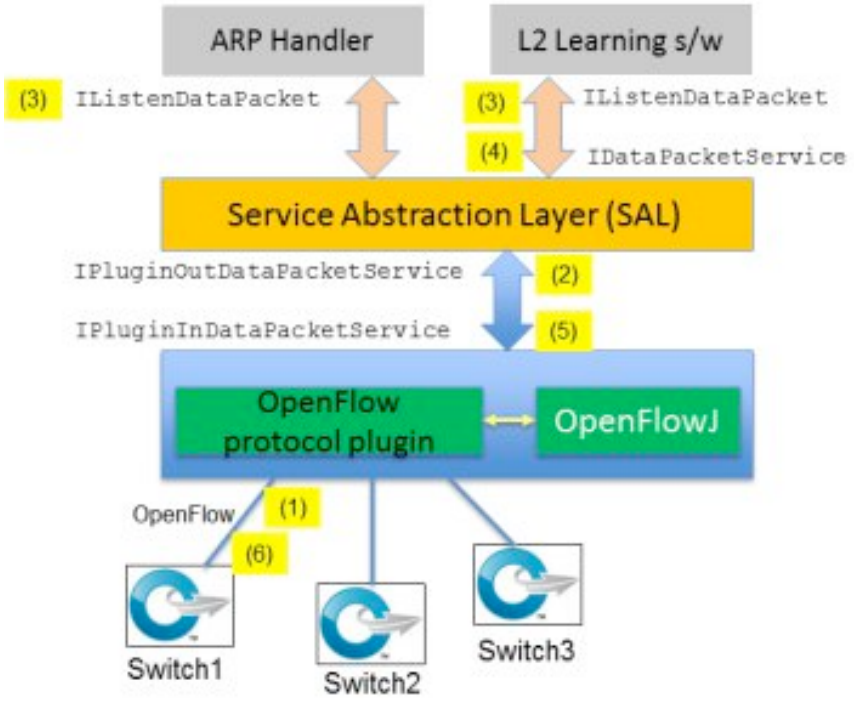

OPEN DAYLIGHT
"Lithium"
Example OpenStack Use Case
The ODL Neutron API connects to the OpenStack Neutron server to handle network service requests
DLUX provides a modern graphical user interface for simplified setup and administration
DLUX
OpenStack Neutron
AAA- AuthN Filter
OpenDaylight APIs (REST)
Base Network Service Functions
Topology Manager
Stats Manager
Switch Manager
FRM
Host Tracker
AAA
OVSDB Neutron
Service Abstraction Layer (SAL)
(Plugin Manager, Capability Abstractions, Flow Programming, Inventory, etc.)
Neutron Northbound
Base Services and MD-SAL deliver the core controller operations
OpenFlow
1.0 1.3 TTP
OVSDB
AAA, OVSDB & Neutron services are enabled to deliver features required to support OpenStack
The OpenFlow and OVSDB Plugins enable the control of Open vSwitch agents that provide network services to each OpenStack VM
Open vSwitches on Each OpenStack Host
Additional Services that could be added on to base use case:
LISP Service
GBP Service
DOCSIS Abstraction
Reservation
VPN
Persistence
LACP
SFC
L2 Switch
SDNI Aggregator
DIDM
TOPO Processing
USC Mgr.
VTN Manager
NIC
TSDR
Additional plugins that base use case could be extended to:
CAPWAP
CoAP
NETCONF
PCMM/COPS
USC
HTTP
SNMP
SXP
ALTO
OPFLEX
PCEP
SNBI
LISP
BGP

ODL Interfaces

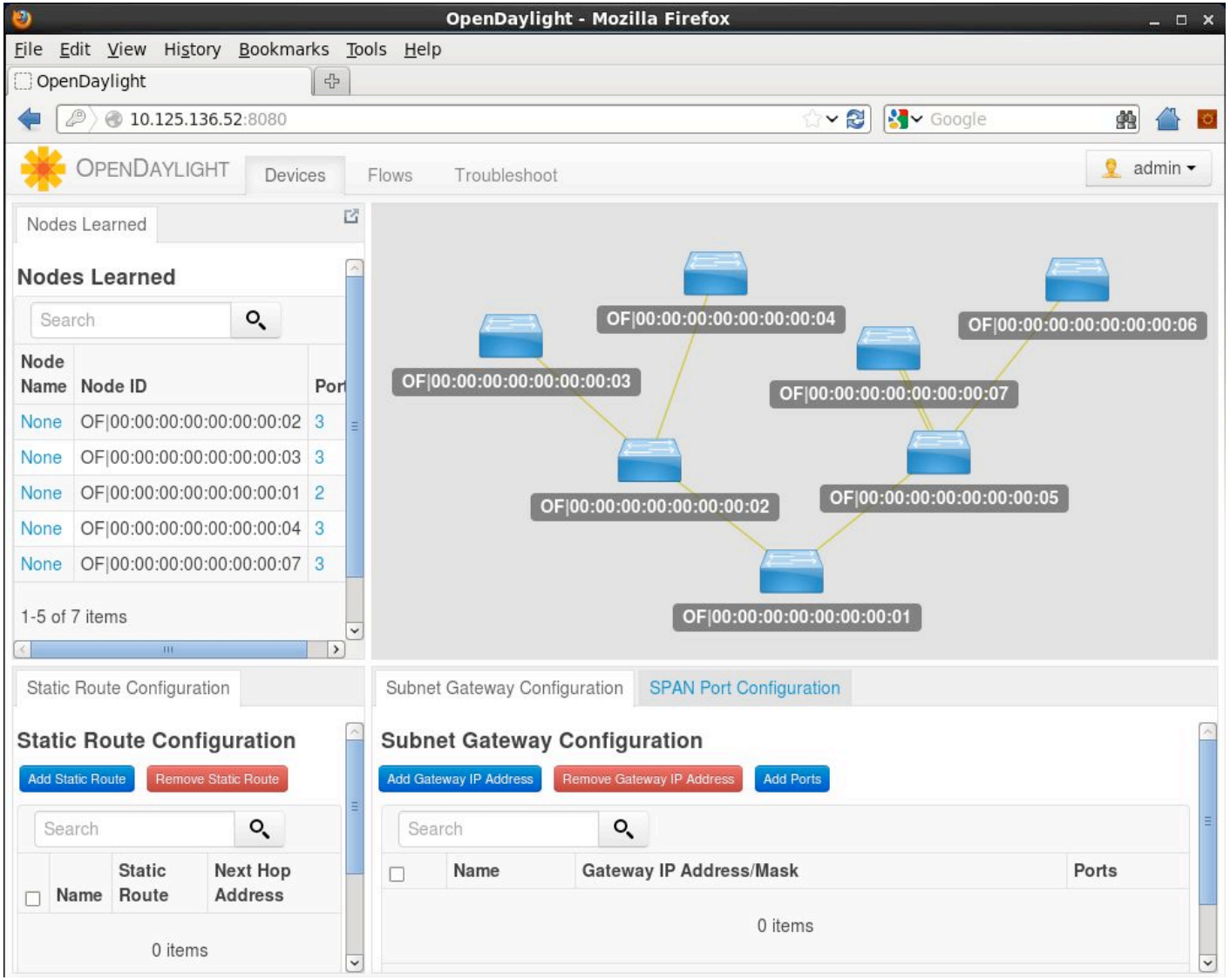
- Java: entreprise-grade, cross-platform language
  - Used for event listening, specs., forming patterns
- Maven: a build system for Java
- OSGi (Open Sources Gateway Initiative):
  - Modular system and service platform for Java
  - Implements dynamic &complete component model
  - Allows dynamically loading bundles (jar files)
  - Allows registering dependencies and services exported

Life of a packet

1. Pkt a S1 send to plugin
2. Plugin parses packet; generates event for SAL
3. SAL dispatches packet to modules listening
4. Module handles packet and sends packet_out
5. SAL dispatches packet to modules below listening
6. OpenFlow message sent to switch



ODL Web interface



Essential Code Constructs

Task	Construct

Packet_In handling	public class XX implements IListenDataPacket
	{
	public PacketResult receiveDataPacket(RawPacket inPkt)
	{ ... }
	}

Packet parsing	Ethernet ethHdr = (Ethernet) this.dataPacketService.decodeDataPacket(inPkt); IPv4 ipv4Hdr = (IPv4) ethHdr.getPayload();

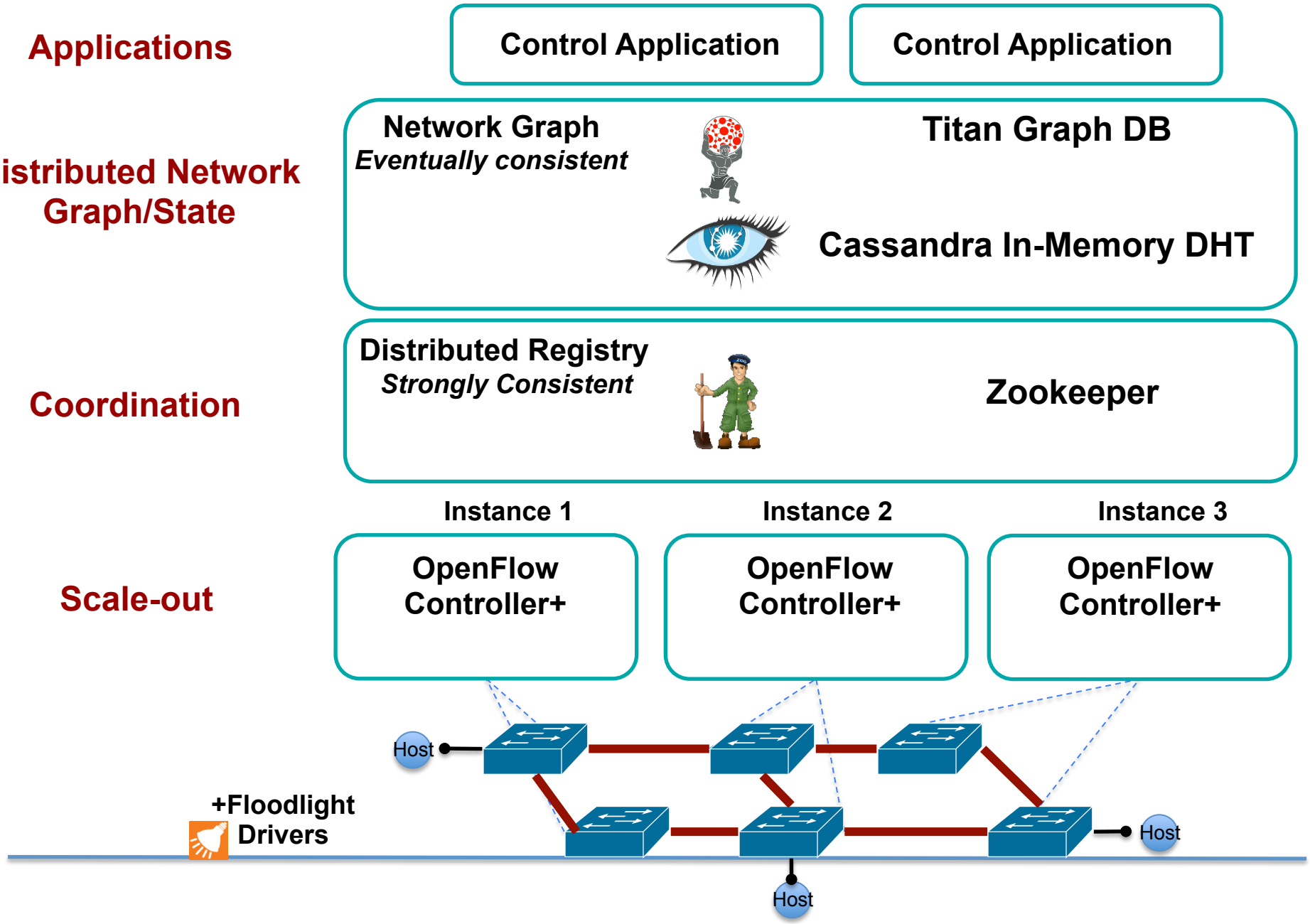
Send msg to switch	RawPacket destPkt = new RawPacket(inPkt);
	destPkt.setOutgoingNodeConnector(p);
	this.dataPacketService.transmitDataPacket(destPkt);

Example: https://github.com/sdnhub/SDNHub_Opendaylight_Tutorial/blob/master/learning-switch/implementation/src/main/java/org/sdnhub/odl/tutorial/learningswitch/impl/TutorialL2Forwarding.single_switch.solution

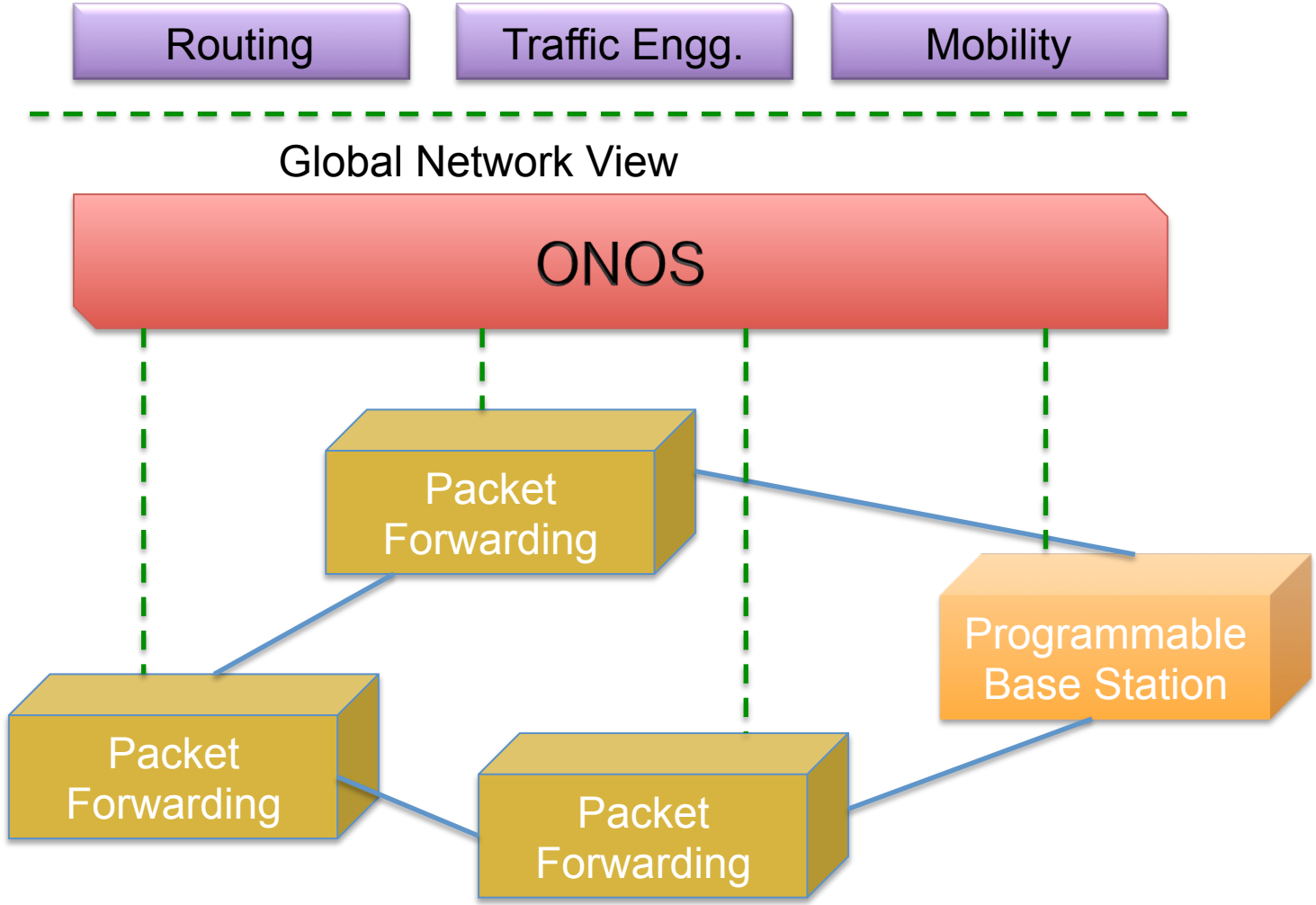
ONOS: Open Network OS

- Open source distributed network operating system for service provider networks
- Currently v1.2.1 (Cardinal): June 2015
  - Controller: OpenFlow
  - Distributed Key-Value Store for Network State: Cassandra
  - Network Graph Abstraction: Titan
  - Inter-ONOS coordination: Zookeeper
- http://tools.onlab.us/onos.html
- http://onosproject.org
- https://wiki.onosproject.org/display/ONOS/Tutorials+and+Walkthroughs

ONOS High Level Architecture



ONOS Architecture



Source: http://tools.onlab.us/onos.html

Controller Comparison

	NOX	POX	Ryu	Floodlight	OpenDaylight	ONOS
Language	C++	Python	Python	Java	Java	Java
Performance	Fast	Slow	Slow	Fast	Fast	Fast
Distributed	No	No	Yes	Yes	Yes	Yes
OpenFlow	V1.0	1.0	1.0-1.4	1.0	1.0, 1.3	
Multi-tenant Clouds	No	No	Yes	Yes	Yes	
Learning Curve	Medium	Easy	Medium	Steep	Steep	