CMPE 209 Network Security

3. SDN Controllers

(Chapter 5. SDN Control Plane)

Dr. Younghee Park

SAN JOSÉ STATE UNIVERSITY

SDN Controllers

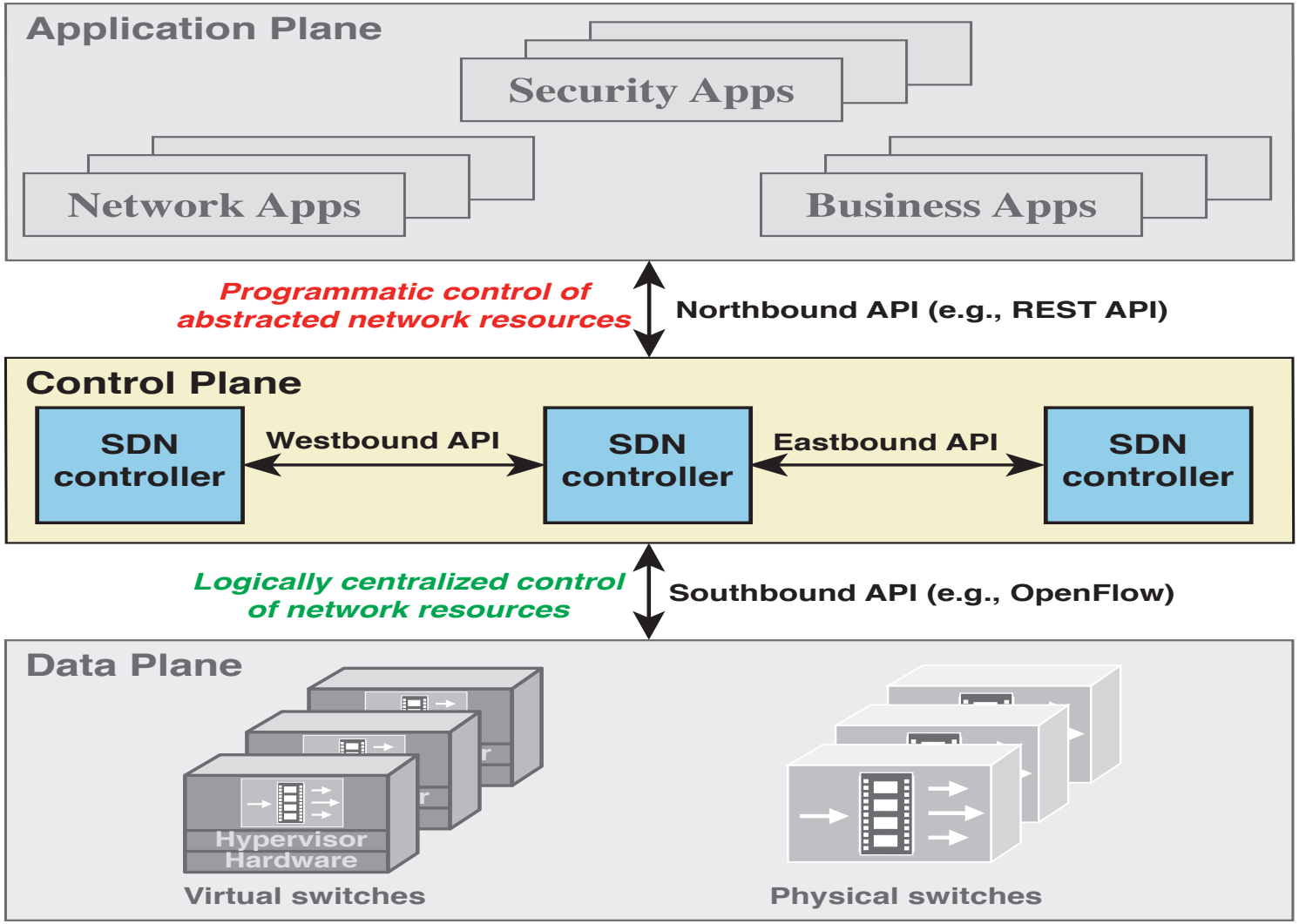
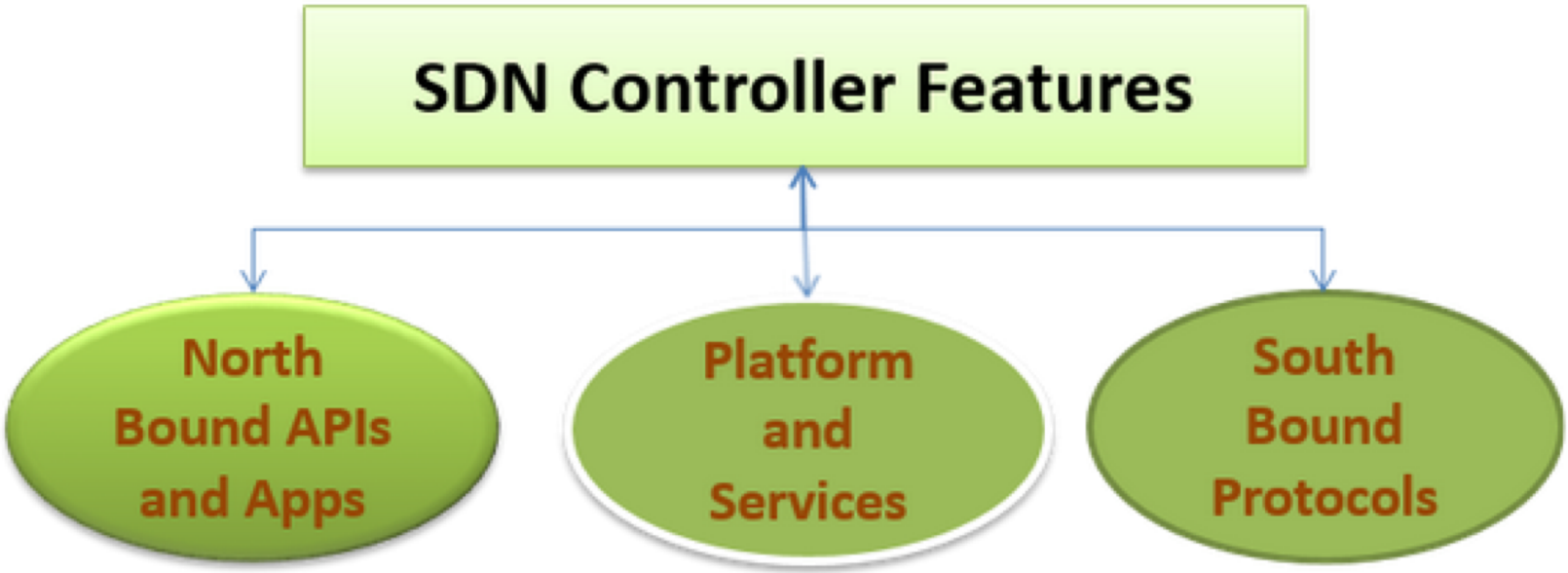


Figure 5.1 SDN Architecture

SDN Controllers



SDN Controllers

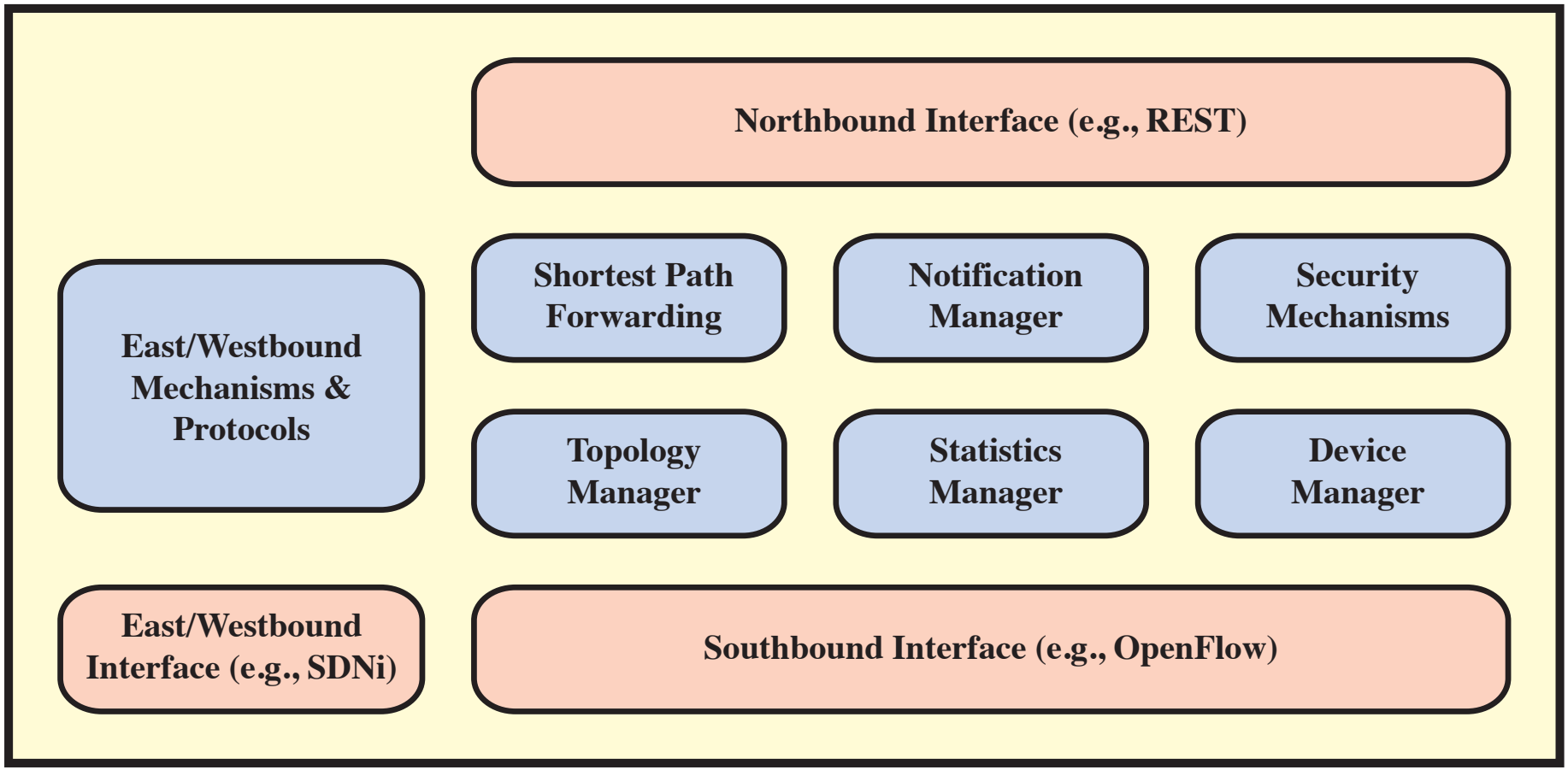


Figure 5.2 SDN Control Plane Functions and Interfaces

Network Operating System (NOS)

- The functionality provided by the SDN controller can be viewed as a network operating system
- A NOS provides essential services, common application programming interfaces (APIs), and an abstraction of lower-layer elements to developers
- The functions of an SDN NOS enable developers to define network policies and manage networks without concern for the details of the network device characteristics

Many SDN controllers on Markets

- A number of different initiatives, both commercial and open source, have resulted in SDN controller implementations:

OpenDaylight	**Open Network Operating System (ONOS)**	**POX**
Beacon	**Floodlight**	**Ryu**
	Onix	

SDN Controllers

- NOX
  - Python, C++ in Linux Platform under GPL license
  - Original author: Nicira
- Beacon
  - Java in Win, Mac, Android platform
  - GPL (cores), FOSS licenses for your code
  - Runtime modular, web UI framework, regression test framework
- Floodlight
  - Java in Win and Mac platform under Apache license
  - Original author: Big Switch

SDN Controllers

- ONOS
  - Avocet 1.0.0 version in 2014
  - Distributed architecture, high-availability, scale-out, performance, security-mode ONOS proposed for v2
- OpenDaylight
  - Distributed architecture, enterprise-grade, high-availability, performance, AAA services
- Ryu
  - NTT, Centralized multi-threaded, high quality controller for production environments

Interfaces

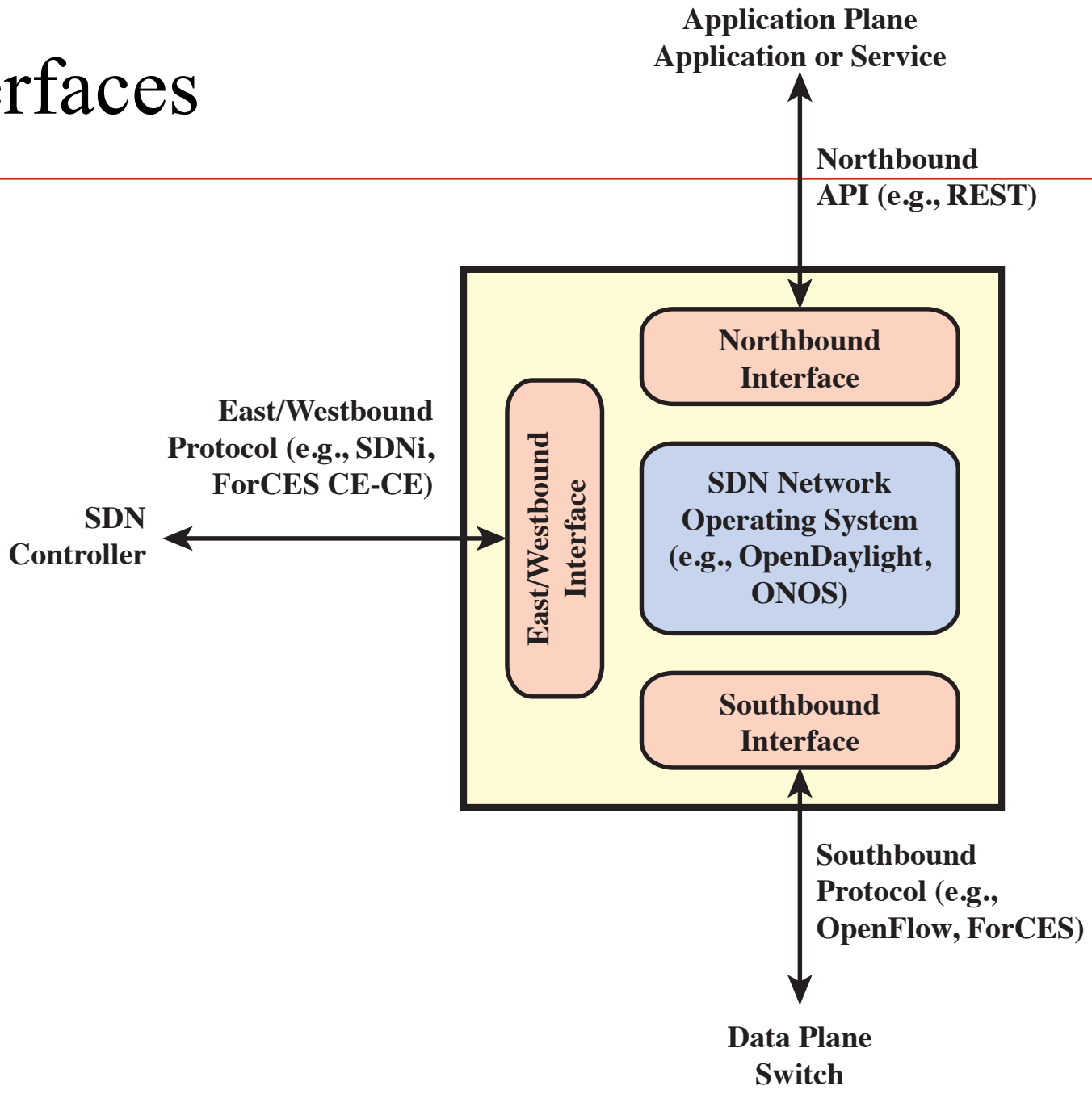


Figure 5.3 SDN Controller Interfaces

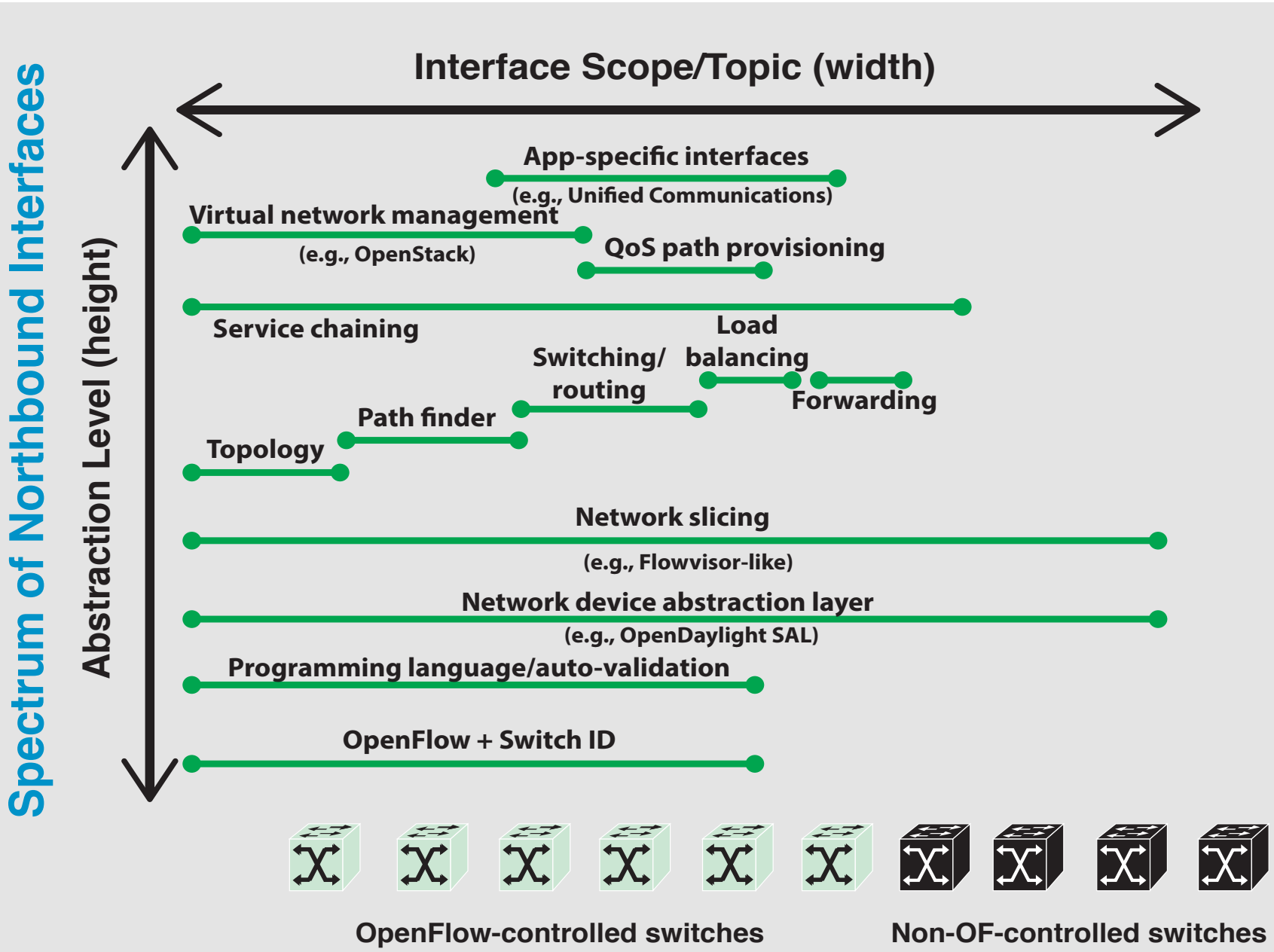
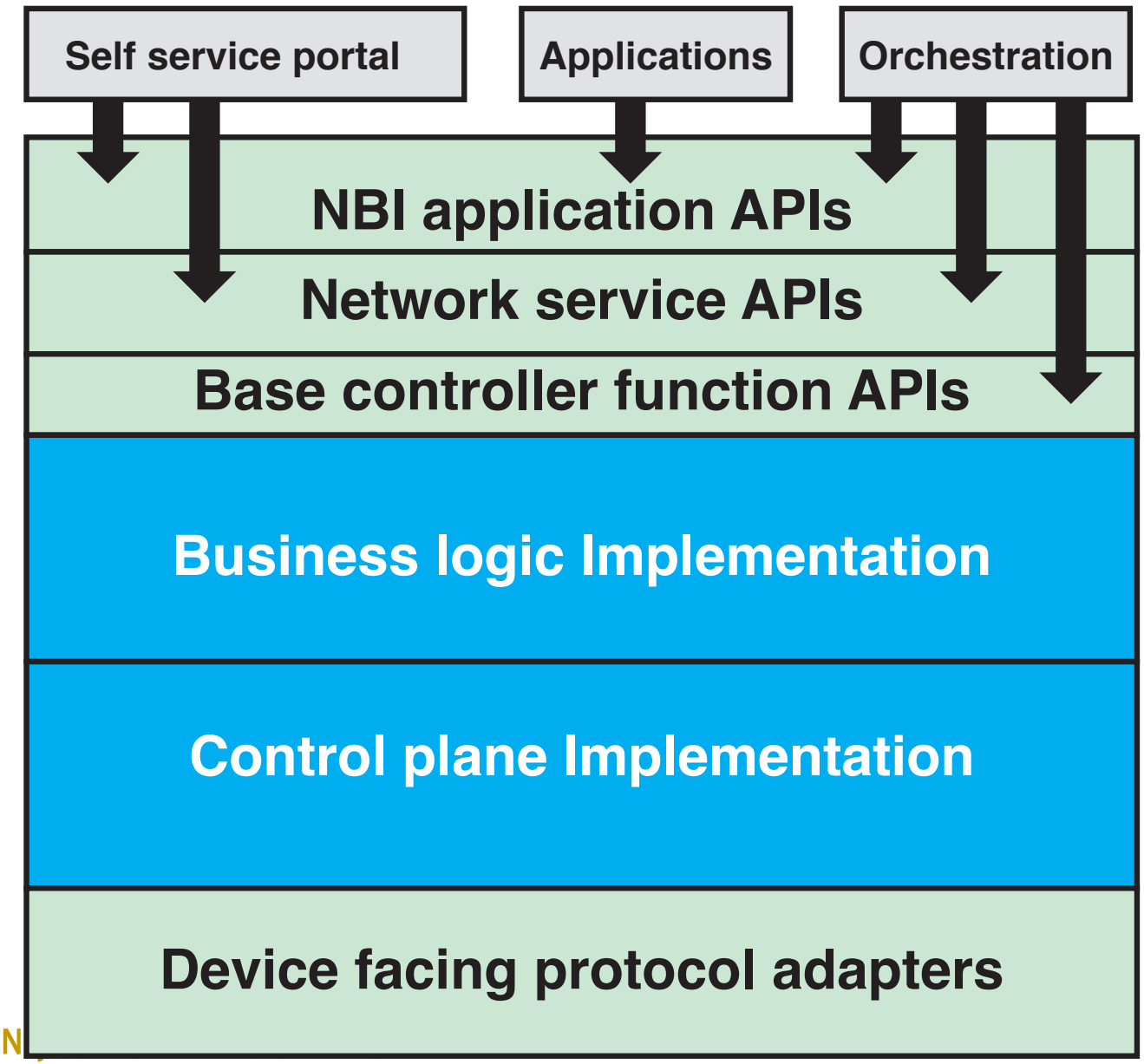
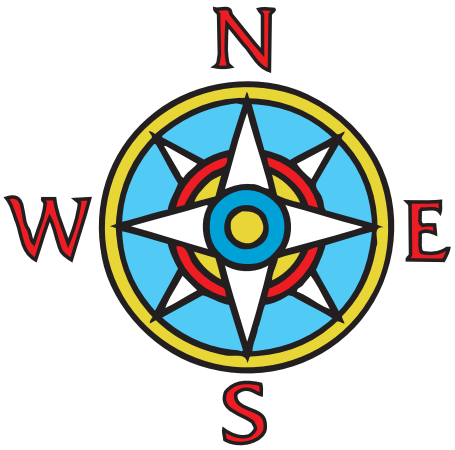


Figure 5.4 Latitude of Northbound Interfaces

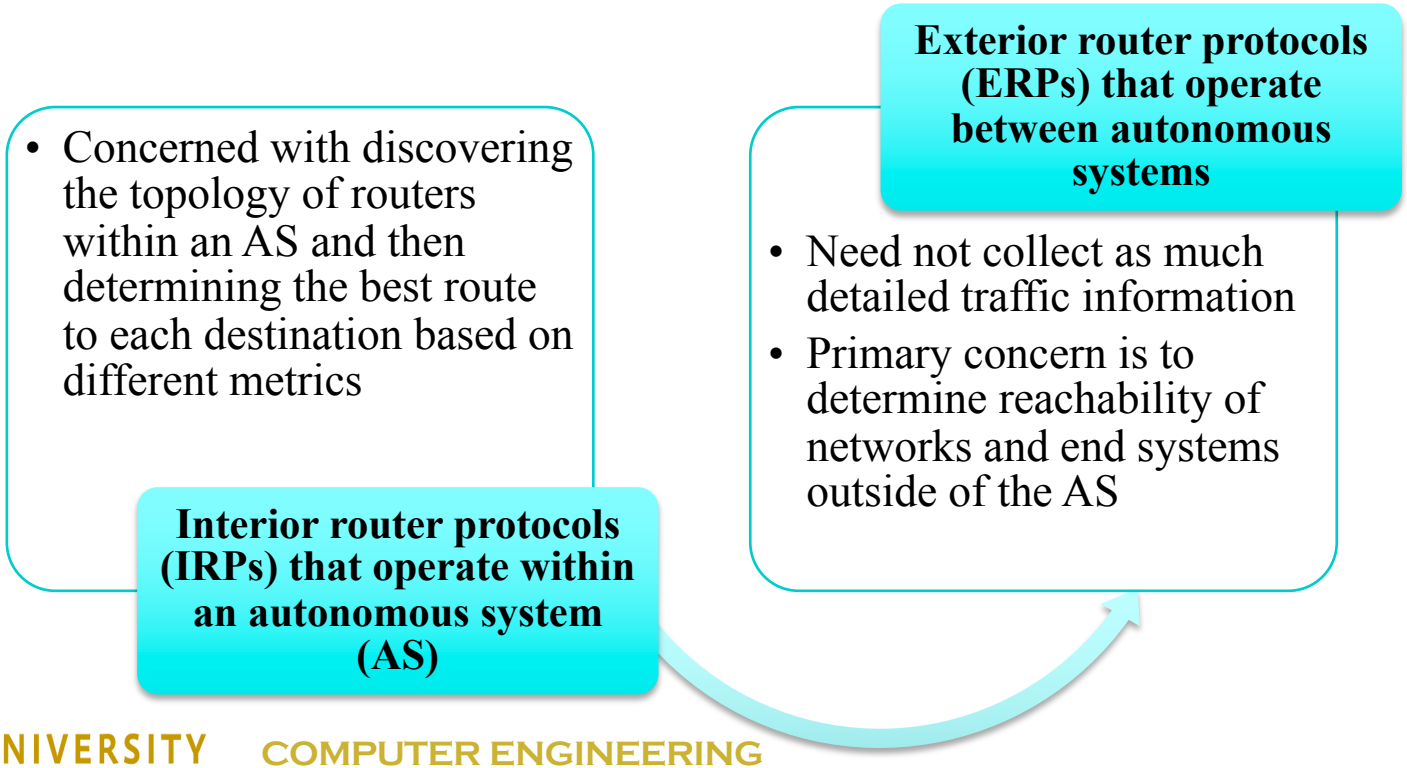
SDN Controller APIs





Routing

- The routing function comprises a protocol for collecting information about the topology and traffic conditions of the network, and an algorithm for designing routes through the network
- There are two categories of routing protocols:



Routing

- Traditionally the routing function is distributed among the routers in a network
- In an SDN controlled network, it makes sense to centralize the routing function within the SDN controller
- The controller can develop a consistent view of the network state for calculating shortest paths and can implement application aware routing policies
- The data plane switches are relieved of the processing and storage burden associated with routing, leading to improved performance

Routing

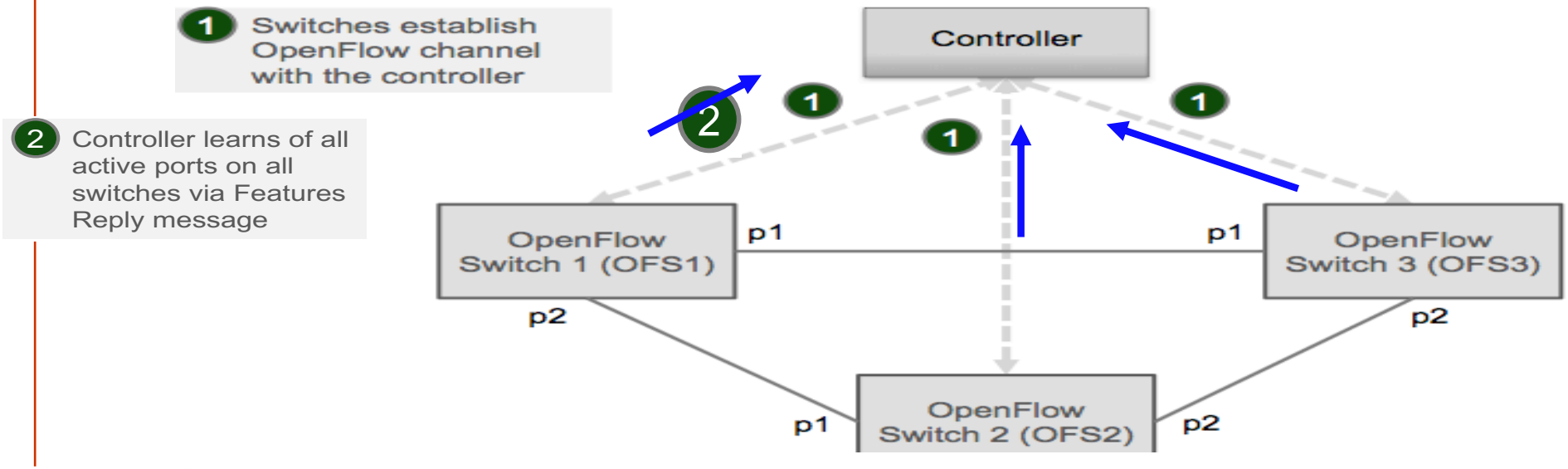
- The centralized routing application performs two distinct functions:
  - Link discovery
    - The routing function needs to be aware of links between data plane switches
    - Must be performed between a router and a host system and between a router in the domain of this controller and a router in a neighboring domain
    - Discovery is triggered by unknown traffic entering the controller's network domain either from an attached host or from a neighboring router

Routing

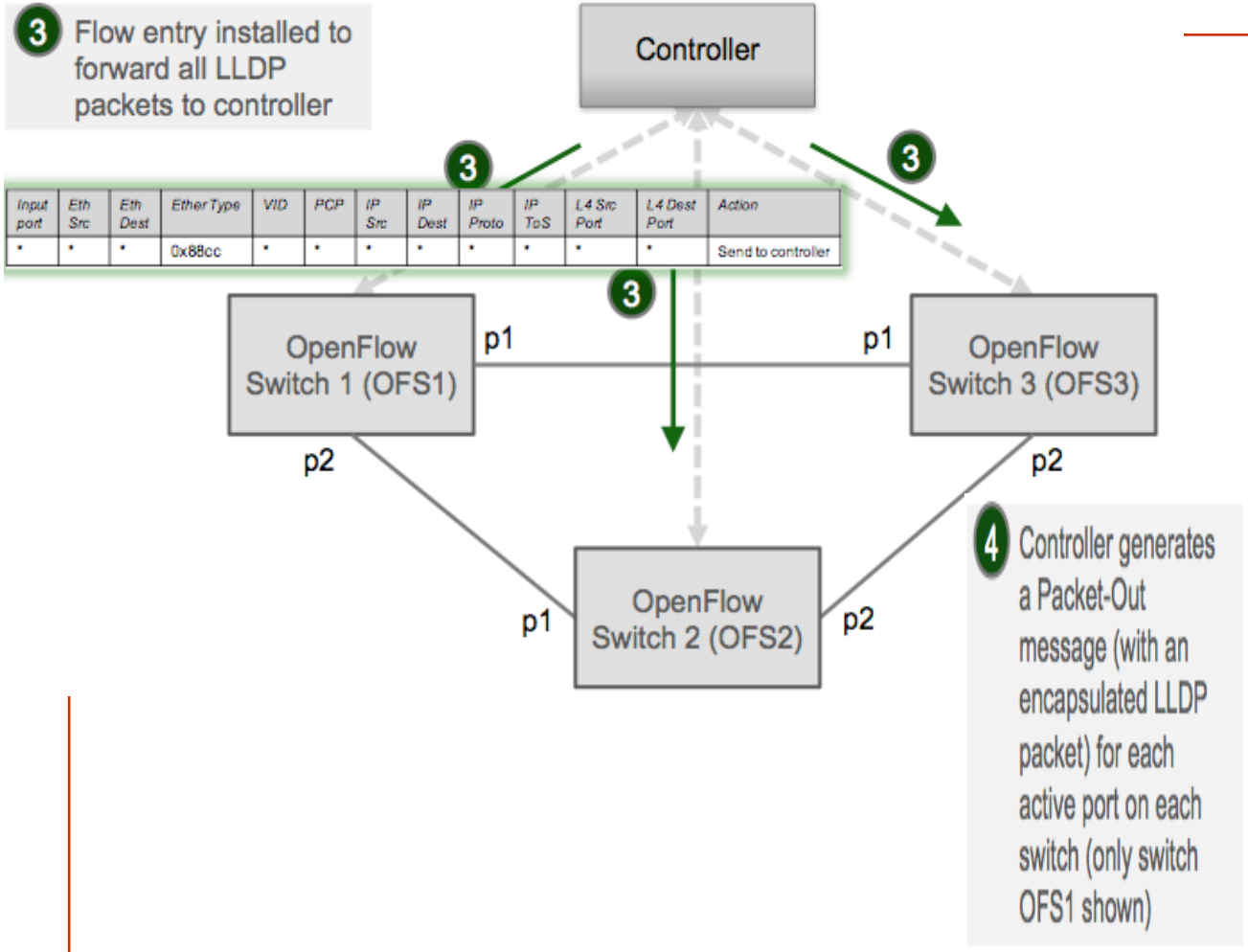
- Topology manager
  - Maintains the topology information for the network and calculates routes in the network.
  - Route calculation involves determining the shortest path between two data plane nodes or between a data plane node and a host.

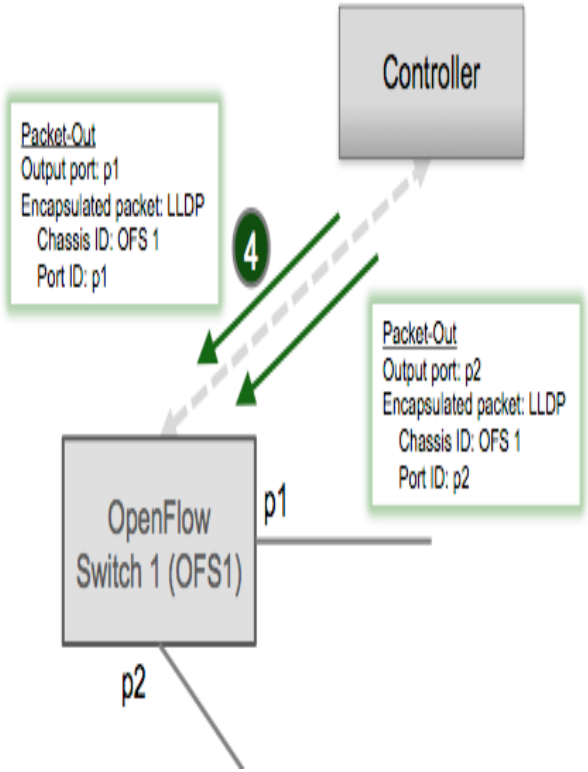
Topology Discovery Process

- Switches need to be bootstrapped as follows:
  - URI or IP address: <port> of OpenFloww controllers
  - LLDP packets (The EtherType field is set to 0x88cc)



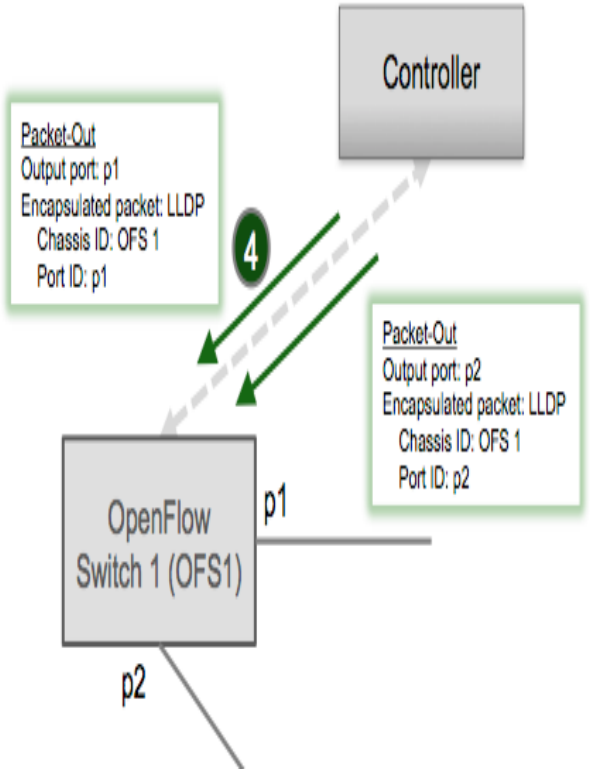
Topology Discovery Process



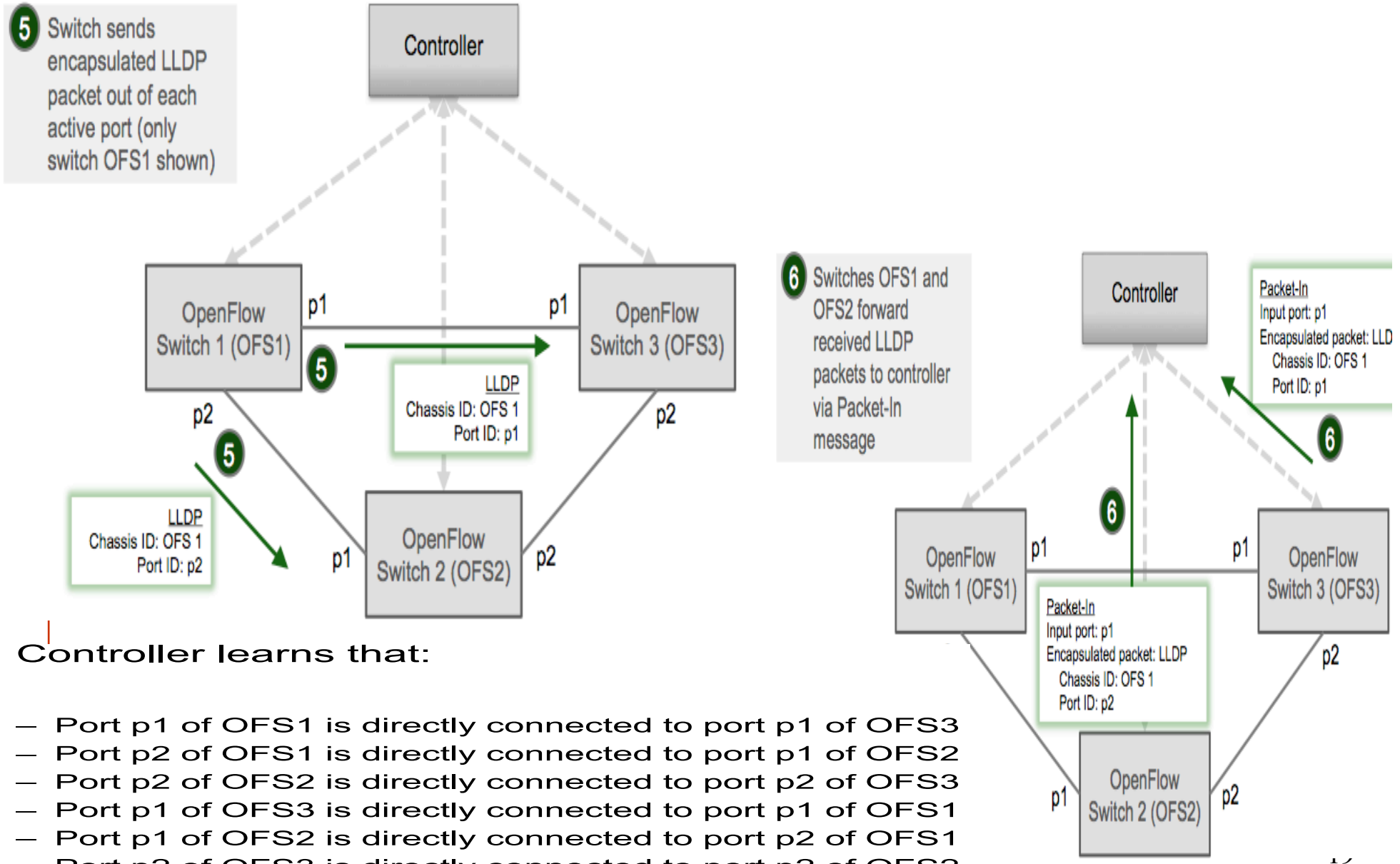


Topology Discovery Process

4 Controller generates a Packet-Out message (with an encapsulated LLDP packet) for each active port on each switch (only switch OFS1 shown)



Topology Discovery Process



- Controller learns that:
  - Port p1 of OFS1 is directly connected to port p1 of OFS3
  - Port p2 of OFS1 is directly connected to port p1 of OFS2
  - Port p2 of OFS2 is directly connected to port p2 of OFS3
  - Port p1 of OFS3 is directly connected to port p1 of OFS1
  - Port p1 of OFS2 is directly connected to port p2 of OFS1
  - Port p2 of OFS3 is directly connected to port p2 of OFS2

Reactive vs Proactive

- Reactive
  - First packets from flow triggers controller to insert flow table
  - If control connection lost, switch has limited utility.
  - Pros
    - Efficient use of flow table memory
  - Cons
    - Cause setup time
    - Hard dependency, connection must retain

Reactive vs Proactive

- Proactive
  - Controller pre-populate flow table in switches
  - Loss of control connection does not disrupt traffic
  - Essentially requires aggregated (wildcard) rules
  - Pros
    - Zero setup time
    - Soft dependency
  - Cons
    - Hard management

Proactive vs Reactive Flow Entries

- Entries in the flow table can be installed either a priori(proactive) or on demand (reactive):
  - It is also possible to have a combination of proative and reactive flow entries

Proactive	Reactive
• Applicable when flow patterns are known ahead of time • More suitable for aggregate traffic flows • May require larger tables to allow a complete set of flow entries • No delays with flow installation	• May be more applicable to dynamic flow patterns • Optimises flow table usage as inactive flows may be timed out • Delays may be experienced with flow installation as first packet needs to be sent to controller • Uninterrupted connection to controller is essential

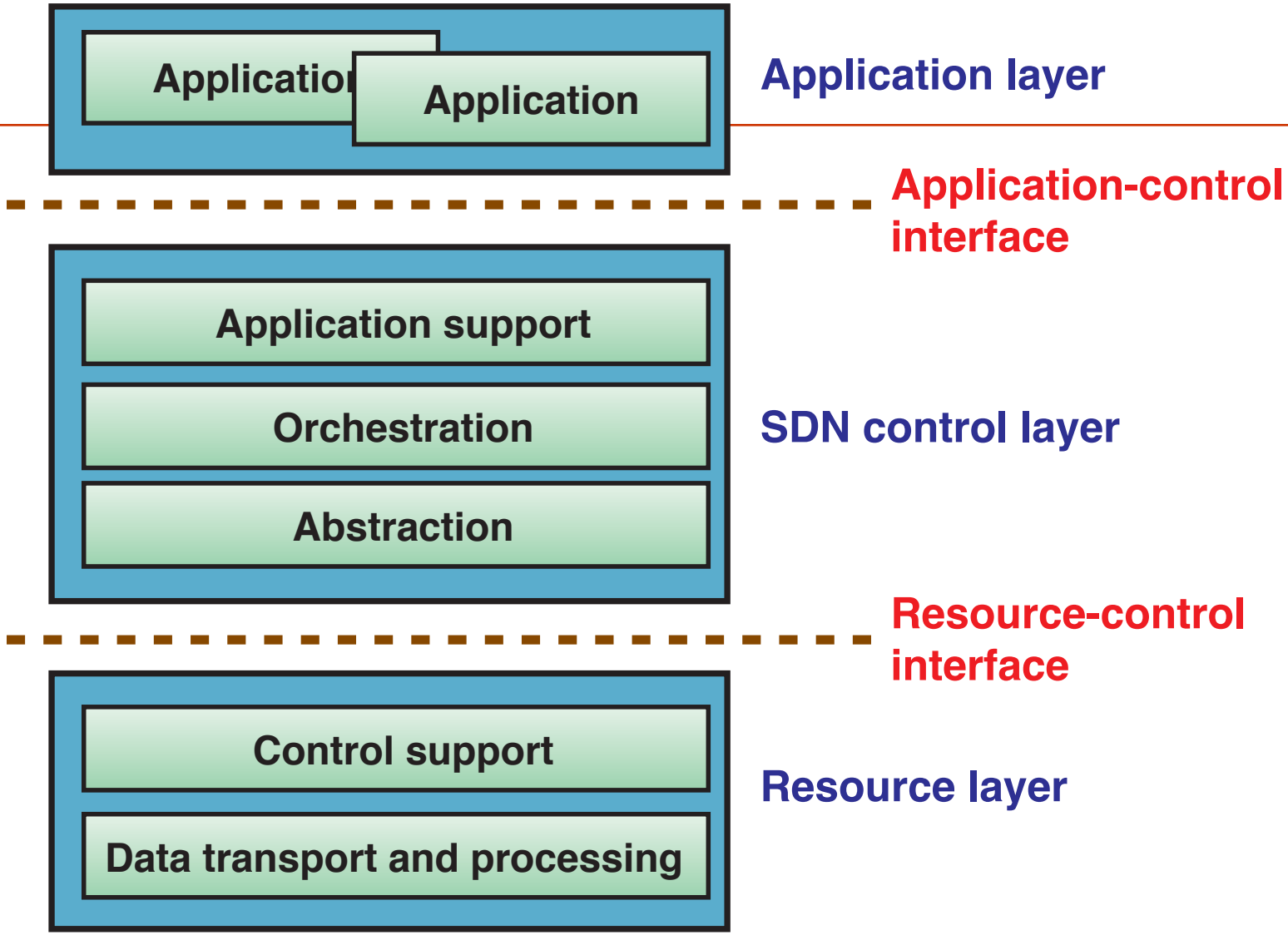
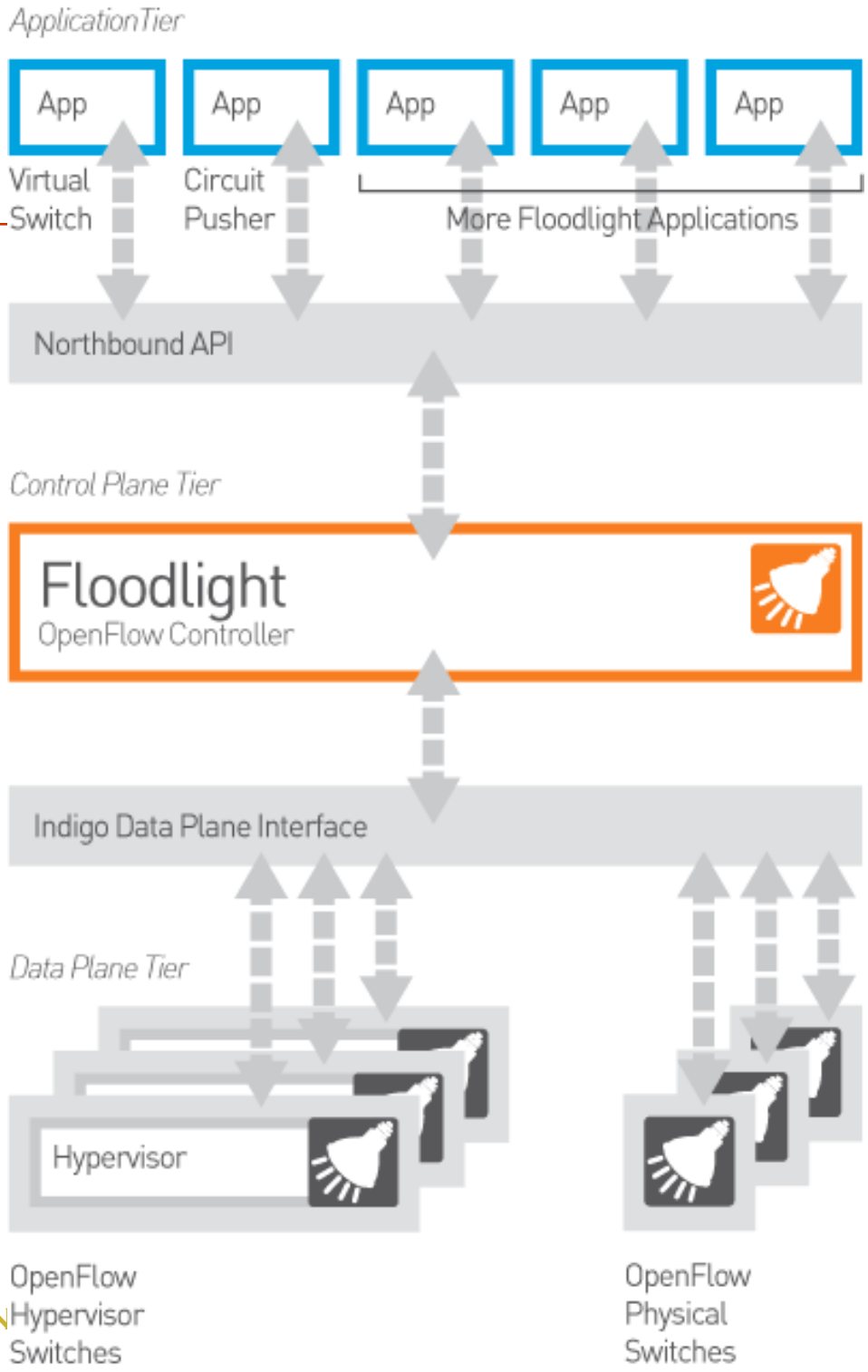


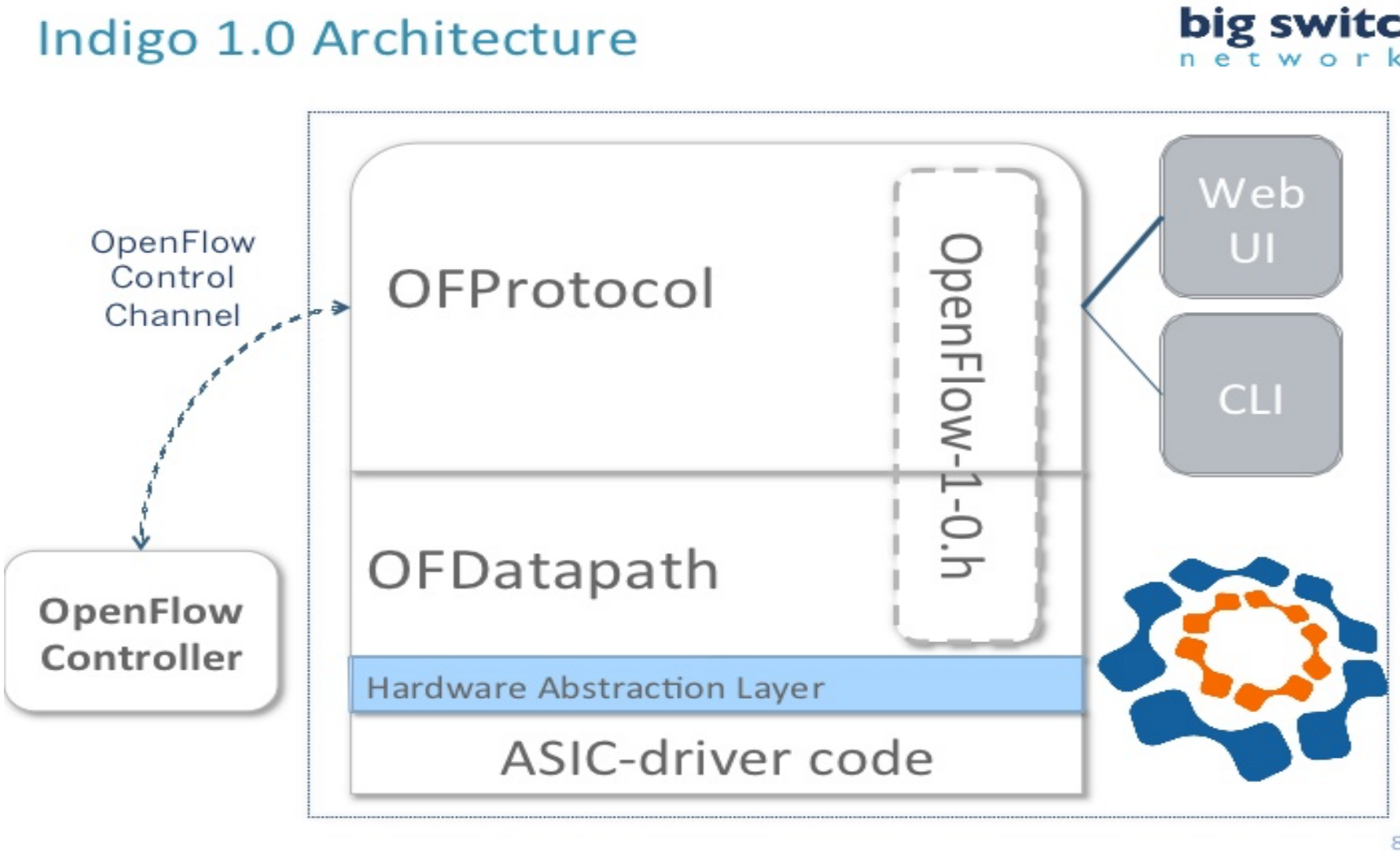
Figure 5.6 High-Level Architecture of SDN (ITU-T Y.3300)

Floodlight

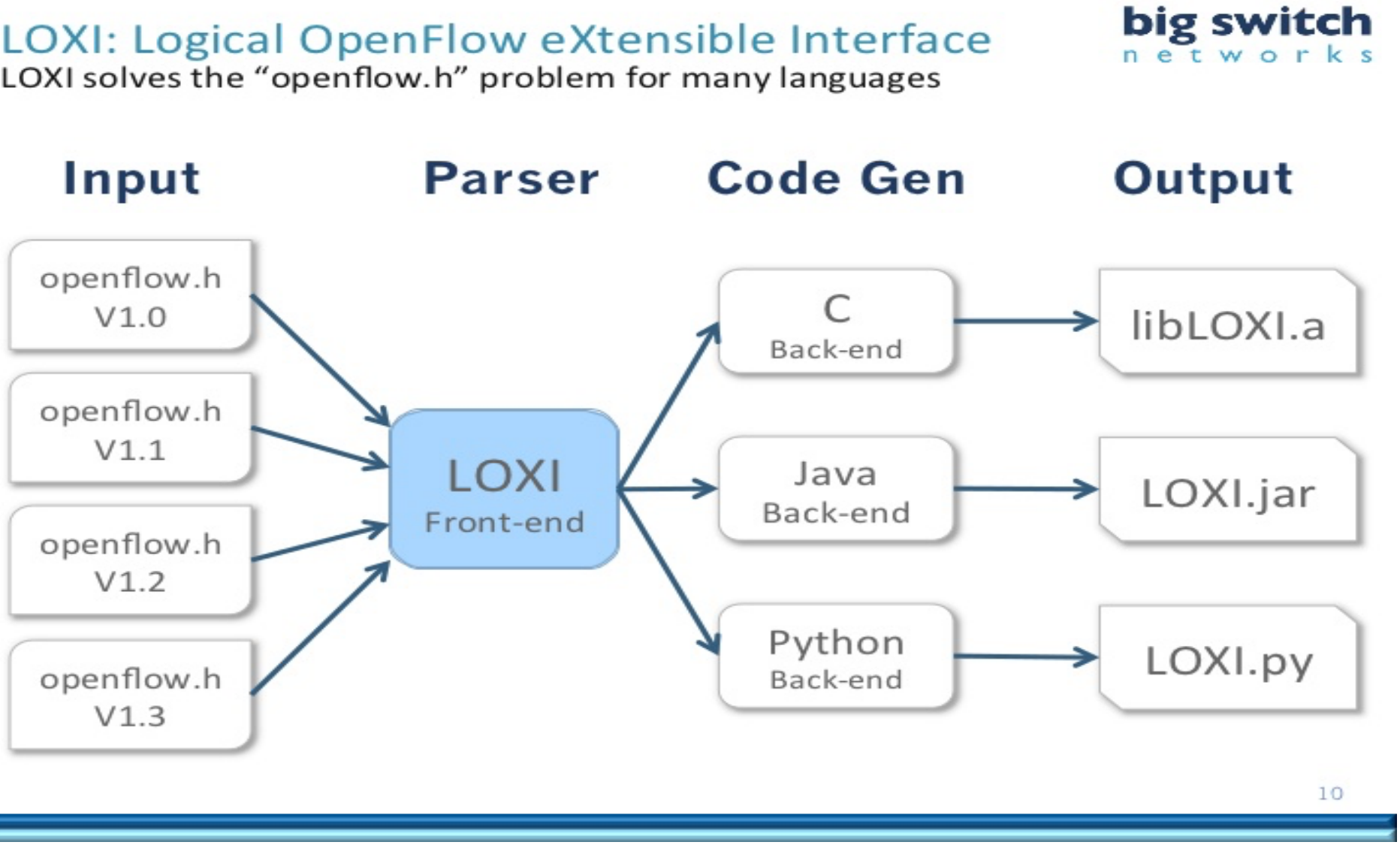
- **Openflow**
  - **Work with physical and virtual switches that speak openflow protocol**
- **Apache Licensed**
  - **Use floodlight for any purpose**
- **Open communication**
  - **Developed by open community**
- **Easy to use / Enterprise-class**
- **Application/Controller plane/Data plane Tier**
  - **Indigo Data plan interface (open source project/ aimed at enabling support for Openflow on physical and hypervisor switches**



Indigo 1.0 Architecture



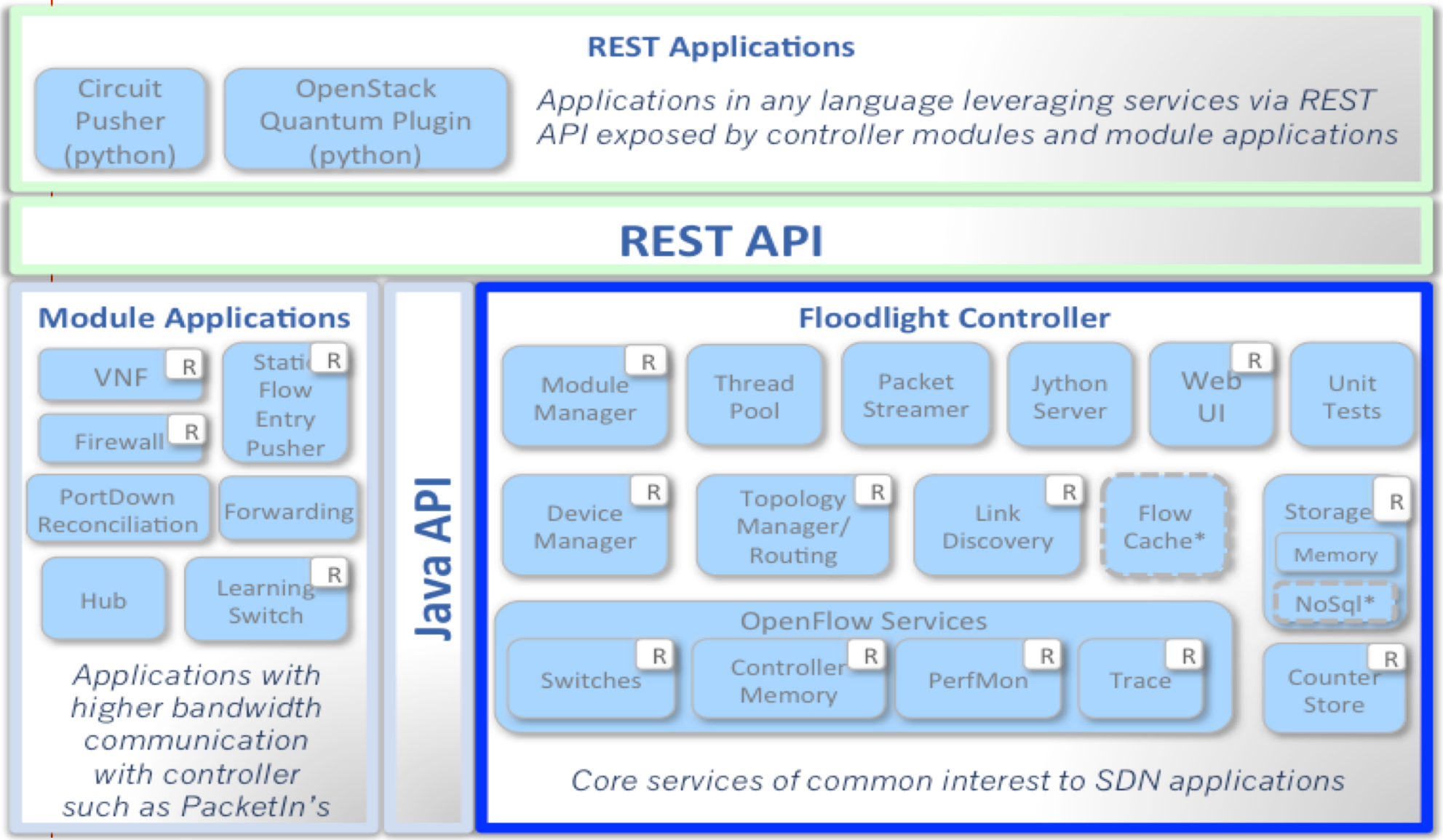
LOXI


LOXI: Logical OpenFlow eXtensible Interface
LOXI solves the "openflow.h" problem for many languages
big switch networks
Input
Parser
Code Gen
Output
openflow.h V1.0
openflow.h V1.1
openflow.h V1.2
openflow.h V1.3
LOXI Front-end
C Back-end
Java Back-end
Python Back-end
libLOXI.a
LOXI.jar
LOXI.py
10

Floodlight

- Floodlight controller
  - Core service of common interest to SDN applications
- Module application
  - Application with higher bandwidth communication with controller
- REST application
  - Application in any language leveraging services via REST API exposed by controller modules and module applications

Floodlight Architecture



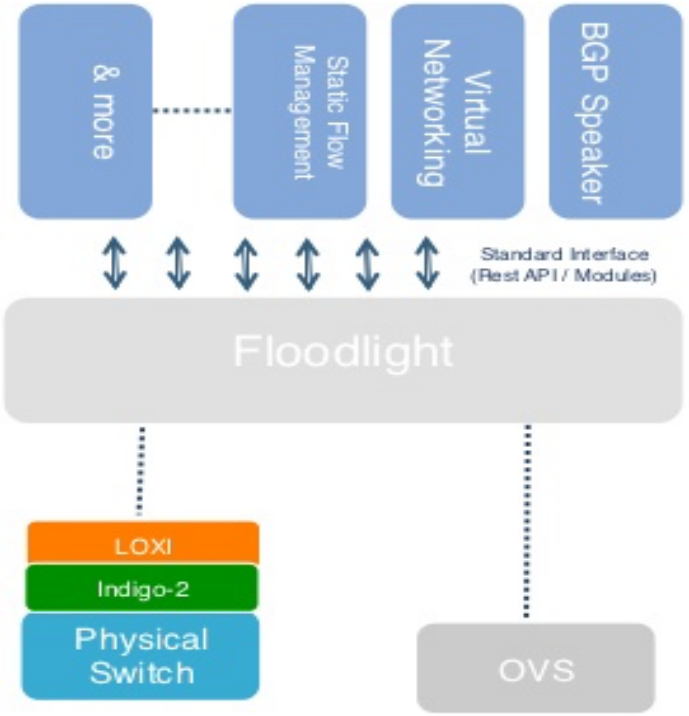
Floodlight Architecture

- Topology
  - Tracks links between host and switches
- Device Manager
  - Tracks devices in the network (MACs, IPs, etc.)
- Storage
  - Abstraction layer for storing controller storage. Memory is used.
- Counter Store
  - Openflow + Floodlight stats
- Routing/Forwarding
  - Core engine for storing, calculating paths and installing flows
- Web UI – rest APIs
- LearningSwitch – can replace Routing/forwarding
- Hub – can replace routing/forwarding

Complete SDN Stack

Complete SDN Stack
At Launch

big switch networks



Application Plane

- Contributed by Open Source community
- Compatible with Big Flow Controller

Control Plane

- Supported by largest community
- De facto northbound API

Data Plane

- OVS (Open vSwitch)
- LOXI (OpenFlow Library)
- Indigo (OF firmware agent)

Testing and Simulation Tools

- OFTest

20

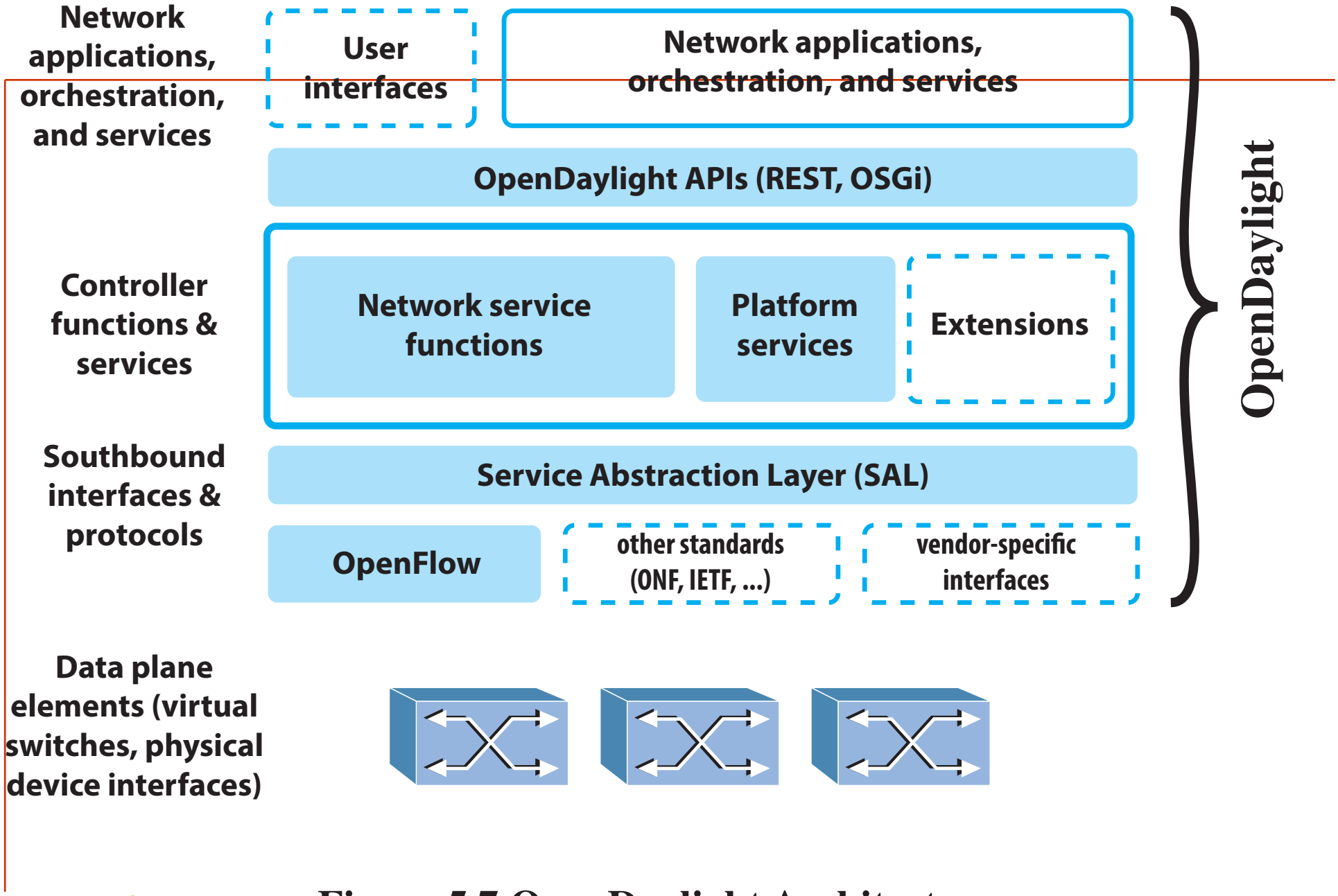


Figure 5.7 OpenDaylight Architecture

LEGEND

AAA	Authentication, Authorization & Accounting	OVSDB	Open vSwitch DataBase Protocol
BGP	Border Gateway Protocol	PCEP	Path Computation Element Communication Protocol
COPS	Common Open Policy Service	PCMM	Packet Cable MultiMedia
DLUX	OpenDaylight User Experience	Plugin2OC	Plugin To OpenControl
DDoS	Distributed Denial of Service	SDNi	SDN Interface
DOCSIS	Data Over Cable Service Interface Specification	SFC	Service Function Chaining
FRM	Forwarding Rules Manager	SNBi	Secure Network Bootstapping Infrasctructure
GBP	Group Based Policy	SNMP	Simple Network Management Protocol
LISP	Location/Identifier Separation Protocol	TTP	Table Type Patterns
OSGi	Open Service Gateway initiative	VTN	Virtual Tenant Network

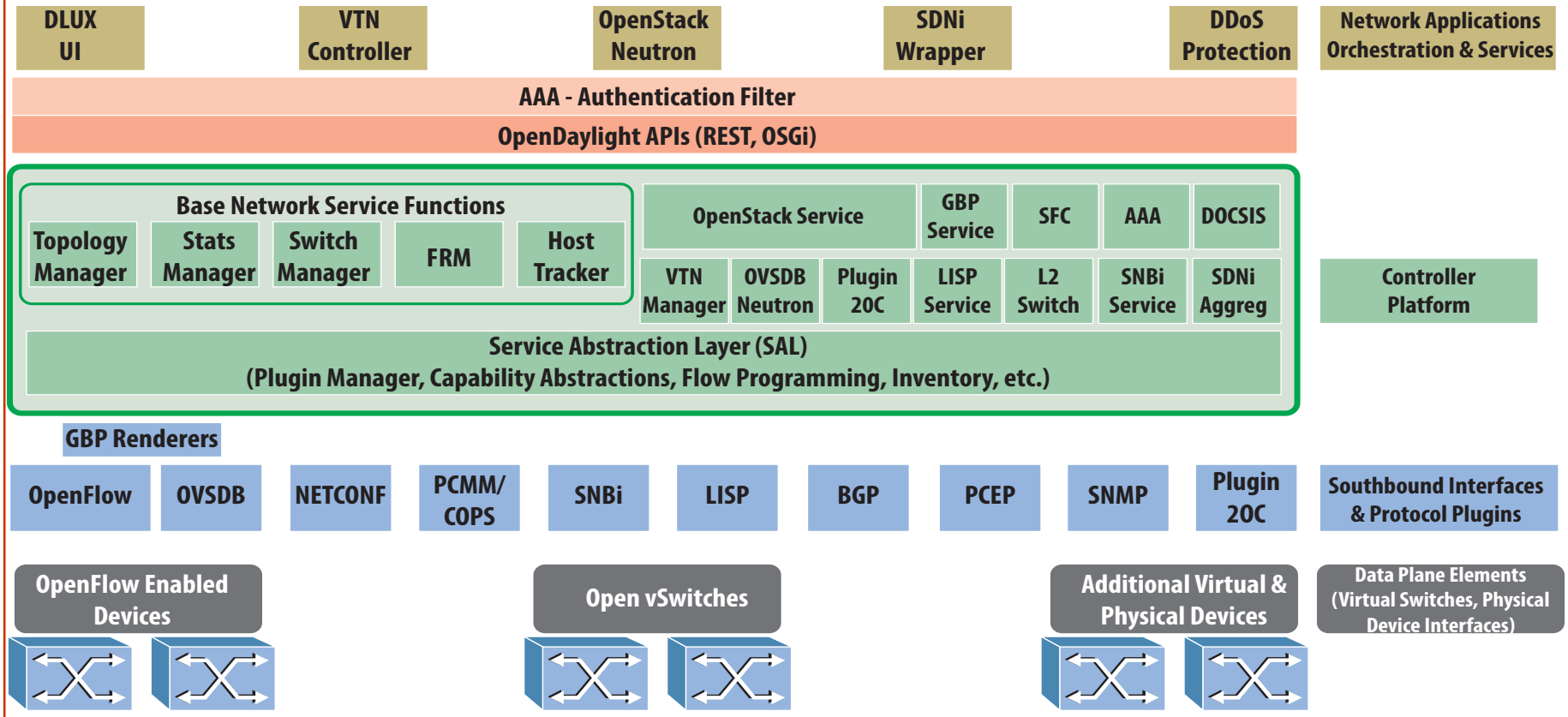


Figure 5.9 OpenDaylight Structure (Helium)

Five Base Network Service Functions

- **Topology manager**: A service for learning the network layout by subscribing to events of node addition and removal and their interconnection. Applications requiring network view can use this service.
- **Statistics manager**: Collects switch-related statistics, including flow statistics, node connector, and queue occupancy.

Five Base Network Service Functions

- **Switch manager**: Holds the details of the data plane devices. As a switch is discovered, its attributes (for example, what switch/router it is, software version, capabilities) are stored in a database by the switch manager.
- **Forwarding rules manager**: Installs routes and tracks next-hop information. Works in conjunction with switch manager and topology manager to register and maintain network flow state. Applications using this need not have visibility of network device specifics.
- **Host tracker**: Tracks and maintains information about connected hosts.

OpenDaylight Modules

Southbound Interfaces and Protocol Plugins

OpenFlow The OpenFlow protocol.

Open vSwitch DataBase Protocol (OVSDB) A network configuration protocol for virtual switching.

NETCONF A network management protocol developed by IETF. NETCONF provides mechanisms to install, manipulate, and delete the configuration of network devices.

Packet Cable MultiMedia (PCMM)/Common Open Policy Service (COPS) Packet Cable MultiMedia (PCMM) provides an interface to control and manage service flow for cable modem network elements.

Secure Network Bootstrapping Infrastructure (SNBi) Provides a secure channel that can be used by other applications to securely connect to various devices.

Location/Identifier Separation Protocol (LISP) An IETF proposed standard that separates the current IP address into two separate name spaces to show an IP location and identifier separately.

BGP For routing service across multi-carrier networks, the BGP-Link State (BGP-LS) protocol is run on the controller. BGP-LS learns the route information from adjacent ASs and builds a consolidated and centralized routing database.

Path Computation Element Communication Protocol (PCEP) Used to provision virtual private network (VPN) configuration information.

Simple Network Management Protocol (SNMP) A collection of specifications for network management that include the protocol itself, the definition of a database, and associated concepts. SNMP enables a management station to monitor and control a network of devices.

Plugin2OC A plugin to the OpenContrail platform. OpenContrail is an open source project whose focus is as a network platform for cloud infrastructure.

Controller Modules

OpenStack Service OpenStack is an open source project to develop a massively scalable cloud operating system platform. SDN architectures that enable virtual networks are a good fit for OpenStack.

Group Based Policy (GBP) Service GBP is an application-centric policy model for OpenDaylight that separates information about application connectivity requirements from information about the underlying details of the network infrastructure.

Service Function Chaining (SFC) An IETF proposed standard for combining service functions. In the SFC model, service functions, whether physical or virtualized, are not required to reside on the direct data path and traffic is instead steered through required service functions, wherever they are deployed.

Authentication, Authorization & Accounting (AAA) Provides three basic security services: authentication means to authenticate the identity of both human and machine users independent of choice of binding (direct or federated); authorization means to authorize human or machine user access to resources including RPCs, notification subscriptions, and subsets of the datatree; accounting means to record and access the records of human or machine user access to resources including RPCs, notifications, and subsets of the datatree.

Data Over Cable Service Interface Specification (DOCSIS) A standard protocol stack for cable modem interface to digital networks.

Virtual Tenant Network (VTN) Manager VTN is an application that provides multi-tenant virtual networks on an SDN controller.

Open vSwitch DataBase (OVSDB) Protocol Neutron Neutron interface to OVSDB.

Plugin2OC Service interface to the Plugin2OC plugin.

LISP Service Service interface to LISP plugin.

L2Switch This is an OSI layer 2 switch routing functionality. The basic concept is to use controller intelligence to design routes through an Ethernet switched network that avoid the use of broadcast when the route to the destination is known.

SNBi Service Service interface to the SNBi plugin.

SDN interface (SDNi) Aggregator The OpenDaylight- SDN Interface Application project aims at enabling inter-SDN controller communication by developing SDNi (Software Defined Networking interface) as an application(ODL-SDNi App). This service acts as an aggregator for collecting network information such as topology, statistics, and host identity and location.

AAA Authentication Filter intercept requests or responses to or from the controller to verify tokens

OpenDaylight Modues

Network Applications, Orchestration, and Services

DLUX UI A Javascript-based stateless user interface that provides a consistent and user-friendly interface to interact with OpenDaylight projects and base controller. DLUX is a web-based interface that provides easy access to a model of the network controlled by this OpenDaylight controller.

VTN Controller Provides API of VTN for users.

OpenStack Neutron Neutron is a subsystem of OpenStack that allows for model-based integration of the network via APIs (in support of the core IaaS capabilities).

SDNi Wrapper Responsible for sharing and collecting information to/from federated controllers.

Distributed Denial-of-Service (DDoS) Protection Instructs an OpenFlow controller to program virtual and physical switches to be OpenFlow counters that collect statistics on network traffic. The application learns baseline traffic patterns and then watches for anomalies indicative of a network-level DDoS attack. If the application detects an attack, it instructs the OpenFlow controller to send suspect flows to specialized mitigation appliances to filter out malicious traffic.

REpresentational State Transfer (REST)

- An architectural style used to define APIs
- This has become a standard way of constructing northbound APIs for SDN controllers
- A REST API, or an API that is RESTful is not a protocol, language, or established standard
- It is essentially six constraints that an API must follow to be RESTful
  - The objective of these constraints is to maximize the scalability and independence/interoperability of software interactions, and to provide for a simple means of constructing APIs
- REST Constraints between Applications and Services
  - Client/server, stateless, cache

Rest Constraints

- The six REST constraints are
  - Client-Server
  - Stateless
  - Cache
  - Uniform interface
  - Layered system

Client-Server

- This simple constraint dictates that interaction between application and server is in the client-server request/response style
- The principle defined for this constraint is the separation of user interface concerns from data storage concerns
- This separation allows client and server components to evolve independently and supports the portability of server-side functions to multiple platforms

Stateless Constraint

- Dictates that each request from a client to a server must contain all the information necessary to understand the request and cannot take advantage of any stored context on the server
- Similarly, each response from the server must contain all the desired information for that request
- One consequence is that any memory of a transaction is maintained in a session state kept entirely on the client
- REST typically runs over Hypertext Transfer Protocol (HTTP), which is a stateless protocol

Cache Constraint

- Requires that the data within a response to a request be implicitly or explicitly labeled as cacheable or non-cacheable
- If a response is cacheable, then a client cache is given the right to reuse that response data for later, equivalent requests
- Therefore, subsequent requests for the same data can be handled locally at the client, reducing communication overhead between client and server

Uniform Interface Constraint

- REST emphasizes a uniform interface between components, regardless of the specific client-server application API implemented using REST
- To obtain a uniform interface, REST defines four interface constraints:
  - Identification of resources (e.g. URI)
  - Manipulation of resources through representations (e.g. JSON, XML, or HTML)
  - Self-descriptive messages
  - Hypermedia as the engine of the application state
- The benefit of this constraint, for an SDN environment is that different applications can invoke the same controller service via a REST API

Layered System Constraint

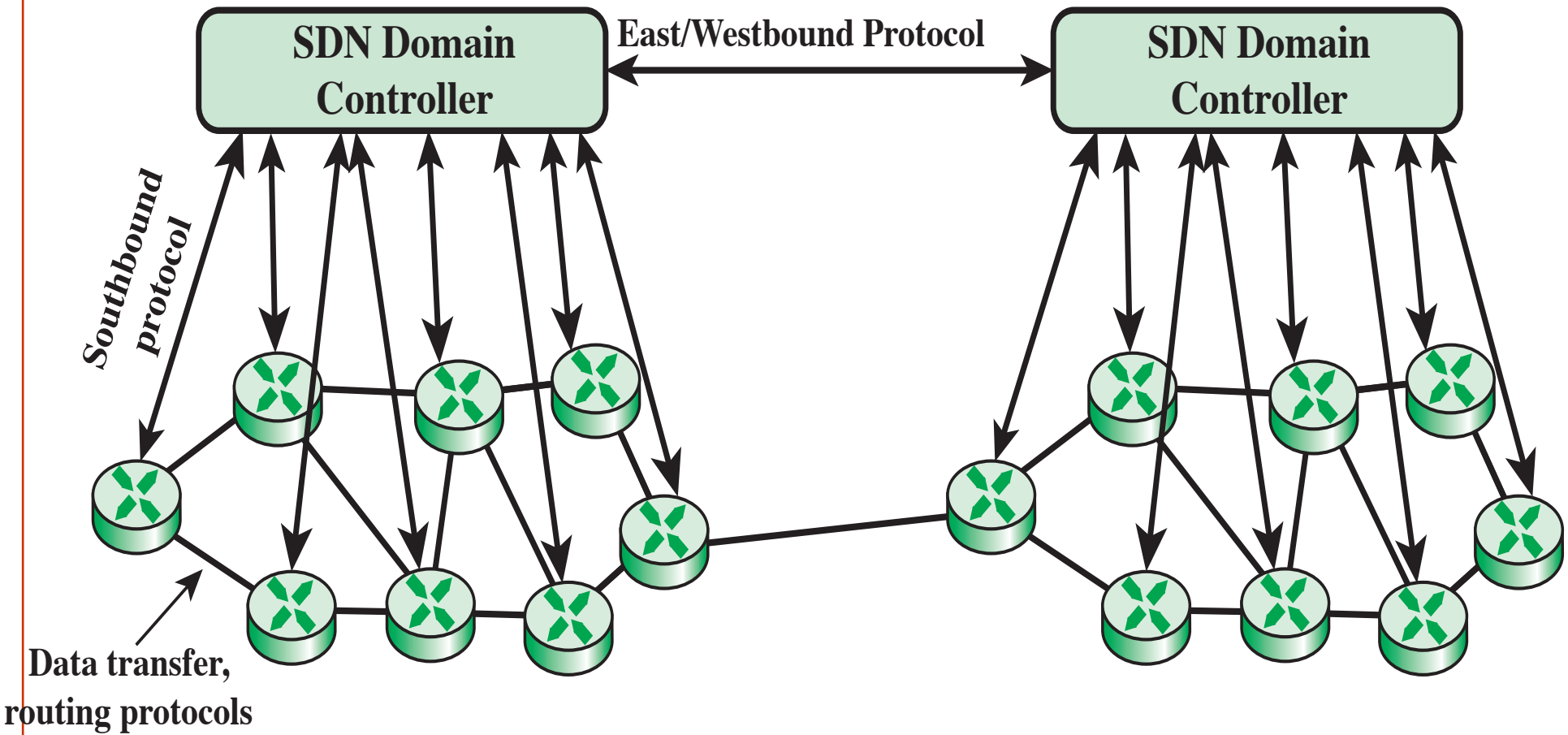
- A given function is organized in layers, with each layer only having direct interaction with the layers immediately above and below
- This is a fairly standard architecture approach for protocol architectures, OS design, and system services design

Switch Statistics using GET

Request type	Response message body attributes
Get all switches	datapath ID
Get switch description	datapath ID, manufacturer description, hardware description, software description, serial number, human readable description of datapath.
Get all flow stats of switch	datapath ID, length of this entry, table ID, time flow alive in seconds, time flow alive in nanoseconds, priority, number of seconds idle before expiration, number of seconds before expiration, flags, cookie, packet count, byte count, fields to match, actions
Get aggregate flow stats of switch	datapath ID, packet count, byte count, number of flows
Get port stats	receive packet count, transmit packet count, receive byte count, transmit byte count, dropped receive packet count, dropped transmit packet count, receive error count, transmit error count, frame alignment error count, receive packet overrun count, CRC error count, collision count, time port alive in seconds, time port alive in nanoseconds
Get ports description	datapath ID, port number, Ethernet address, port name. config flags, state flag, current features, advertised features, supported features, features advertised by peer, current bit rate, max bitrate
Get queues stats	datapath ID, port number, queue ID, transmit byte count, transmit packet count, packet overrun count, time queue alive in seconds, time queue alive in nanoseconds
Get groups stats	datapath ID, length of this entry, group ID, number of flows or groups that forward to this group, packet count, byte count
Get group description	datapath ID, type, group ID, buckets (weight, watch_port, watch_group, actions)
Get group features	datapath ID, types, capabilities, max number of groups, actions supported
Get meters stats	datapath ID, meter ID, length of this entry, number of flows, input packet count, input byte count, time meter alive in seconds, time meter alive in nanoseconds, meter band (packet count, byte count)
Get meter configuration	datapath ID, flags, meter ID, bands (type, rate, burst size)
Get meter features	datapath ID, max number of meters, band types, capabilities, max bands per meter, max color value

SDN Domain Structure

It is for scalability, reliability, privacy, incremental deployment(flexibility).



Border Gateway Protocol (BGP)

- Was developed for use in conjunction with internets that use the TCP/IP suite
- Has become the preferred exterior router protocol (ERP) for the Internet
- Enables routers, called gateways in the standard, in different autonomous systems to cooperate in the exchange or routing information
- The protocol operates in terms of messages, which are sent over TCP connections
- The current version of BGP is BGP-4

BGP

- Three functional procedures are involved in BGP

Neighbor acquisition

The term *neighbor* refers to two routers that share the same network

Occurs when two neighboring routers in different autonomous systems agree to exchange routing information regularly

Neighbor reachability

Once a neighbor relationship is established this procedure is used to maintain the relationship

Each partner needs to be assured that the other partner still exists and is still engaged in the neighbor relationship

Network reachability

Each router maintains a database of the networks that it can reach and the preferred route for reaching each network

Whenever a change is made to this database, the router issues an Update message that is broadcast to all other routers for which it has a neighbor relationship

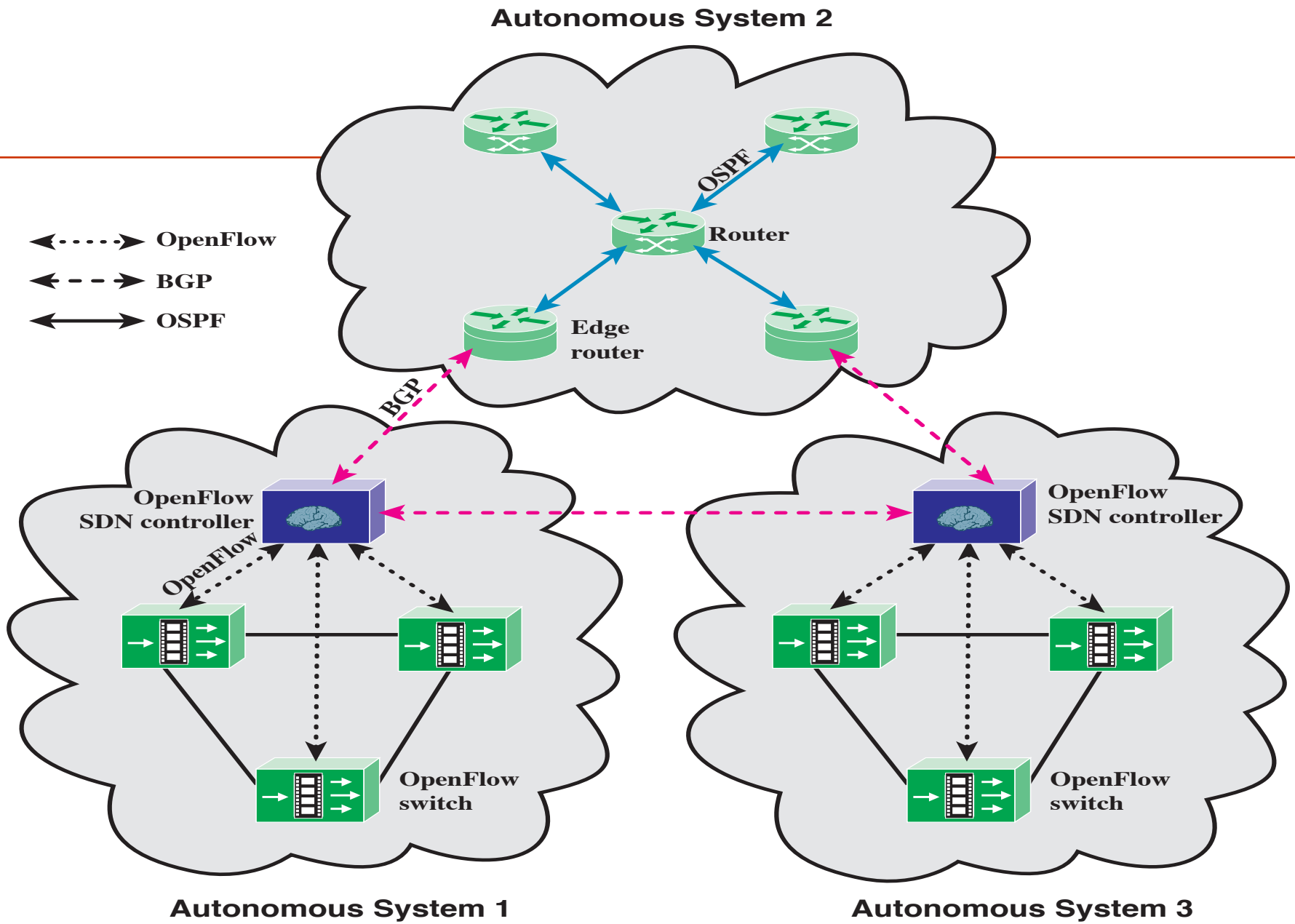


Figure 5.12 Heterogeneous Autonomous Systems with OpenFlow and non-OpenFlow Domains

IETF SDNi

- A draft specification that defines common requirements to coordinate flow setup and exchange reachability information across multiple domains
  - (SDNi: A Message Exchange Protocol for Software Defined Networks across Multiple Domains , draft-yin-sdn-sdni-00.txt, June 27, 2012).
- SDNi functionality, as defined in the document, includes:
  - Coordinate flow setup originated by application, containing information such as path requirement, QoS, and service level agreements across multiple SDN domains
  - Exchange reachability information to facilitate inter-SDN routing; this will allow a single flow to traverse multiple SDNs and have each controller select the most appropriate path when multiple such paths are available

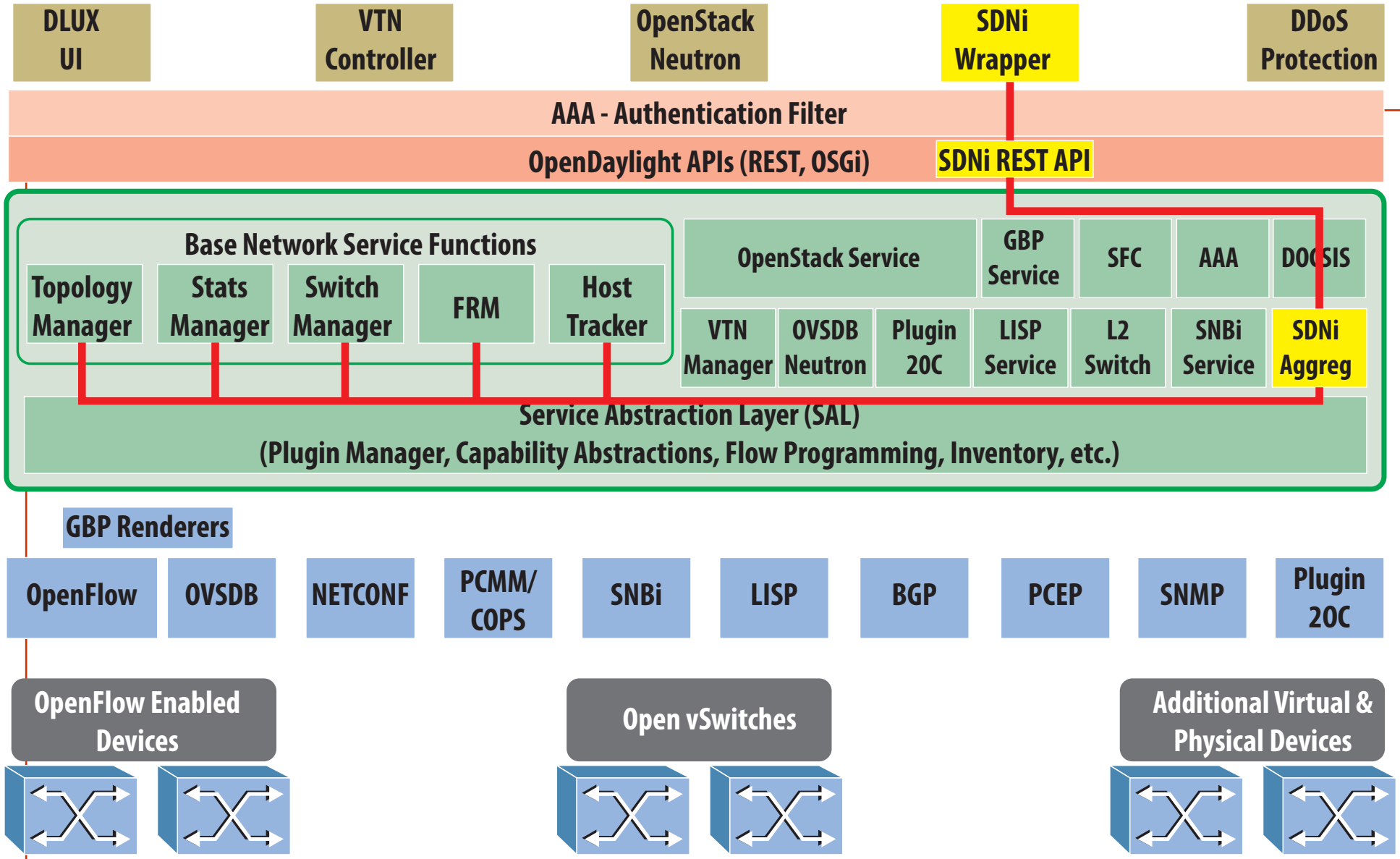


Figure 5.14 SDNi Components in OpenDaylight Structure (Helium)

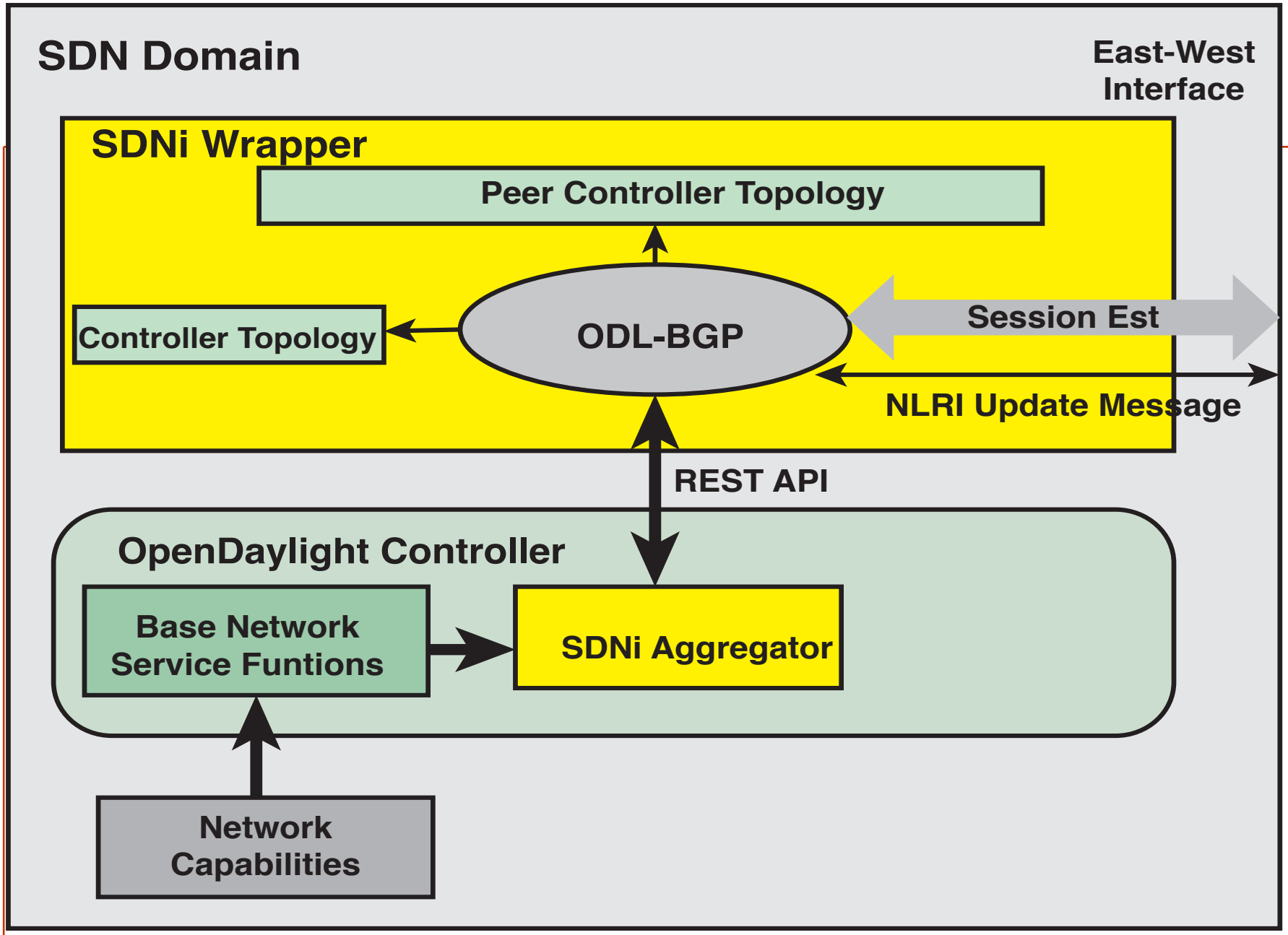
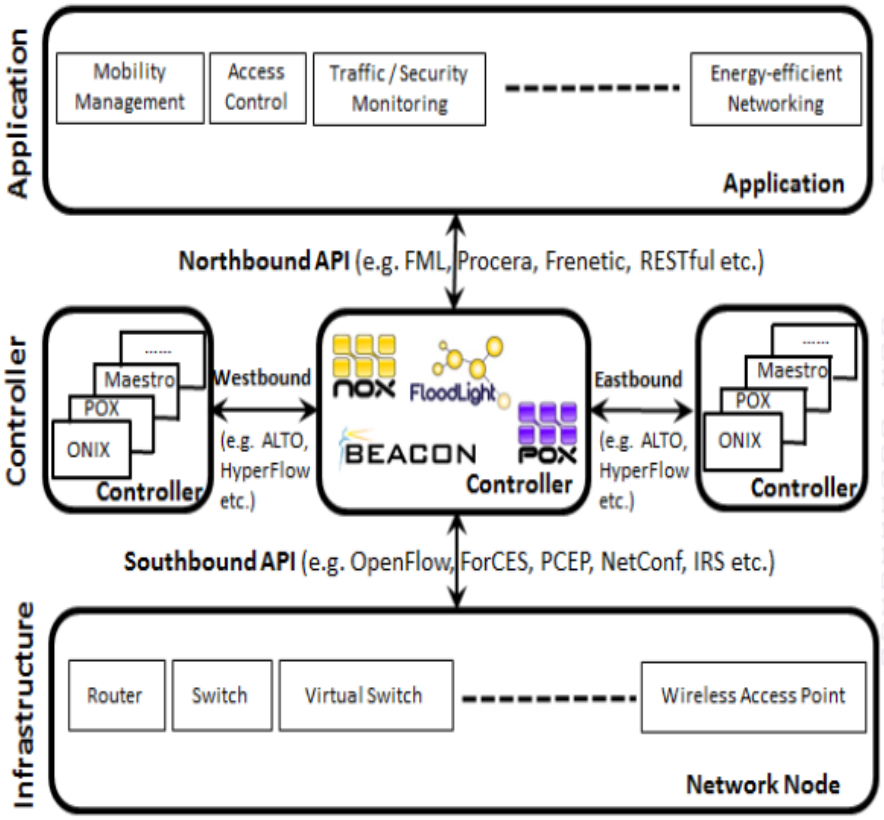


Figure 5.15 OpenDaylight SDNi Wrapper

Software Defined Networking …. and Security



Security Issue/Attack	SDN Layer Affected or Targeted				
	Application Layer	App-Ctl Interface	Control Layer	Ctl-Data Interface	Data Layer
Unauthorized Access e.g.					
Unauthorized Controller Access			✓	✓	✓
Unauthenticated Application	✓	✓	✓		
Data Leakage e.g.					
Flow Rule Discovery (Side Channel Attack on Input Buffer)					✓
Forwarding Policy Discovery (Packet Processing Timing Analysis)					✓
Data Modification e.g.					
Flow Rule Modification to Modify Packets			✓	✓	✓
Malicious Applications e.g.					
Fraudulent Rule Insertion	✓	✓	✓		
Controller Hijacking			✓	✓	✓
Denial of Service e.g.					
Controller-Switch Communication Flood			✓	✓	✓
Switch Flow Table Flooding					✓
Configuration Issues e.g.					
Lack of TLS (or other Authentication Technique) Adoption			✓	✓	✓
Policy Enforcement	✓	✓	✓		

Sezer, S., et al. "Are We Ready for SDN? Implementation Challenges for Software-Defined Networks" *IEEE Communications Magazine*, July 2013

Scott-Hayward, S.; Natarajan, S.; Sezer, S., "A Survey of Security in Software Defined Networks," *Communications Surveys & Tutorials, IEEE* , 10.1109/COMST.2015.2453114

SDN controller functionality: 10 key factors