下世代Network Slicing模組設計

課程單元：SDN與Openflow

國立中山大學 資訊工程系
授課教師：李宗南教授
教材編撰：吳承祐

目錄
CONTENTS

01 Course Objectives

Course Objectives

- To understand SDN,OpenFlow and SDN Controller.
- Learn how to use OpenFlow to SDN Controller to achieve the concept of SDN.

02 SDN Introduction

SDN Introduction

- SDN Background
- Software Define Networking
- SDN Concept

SDN Background

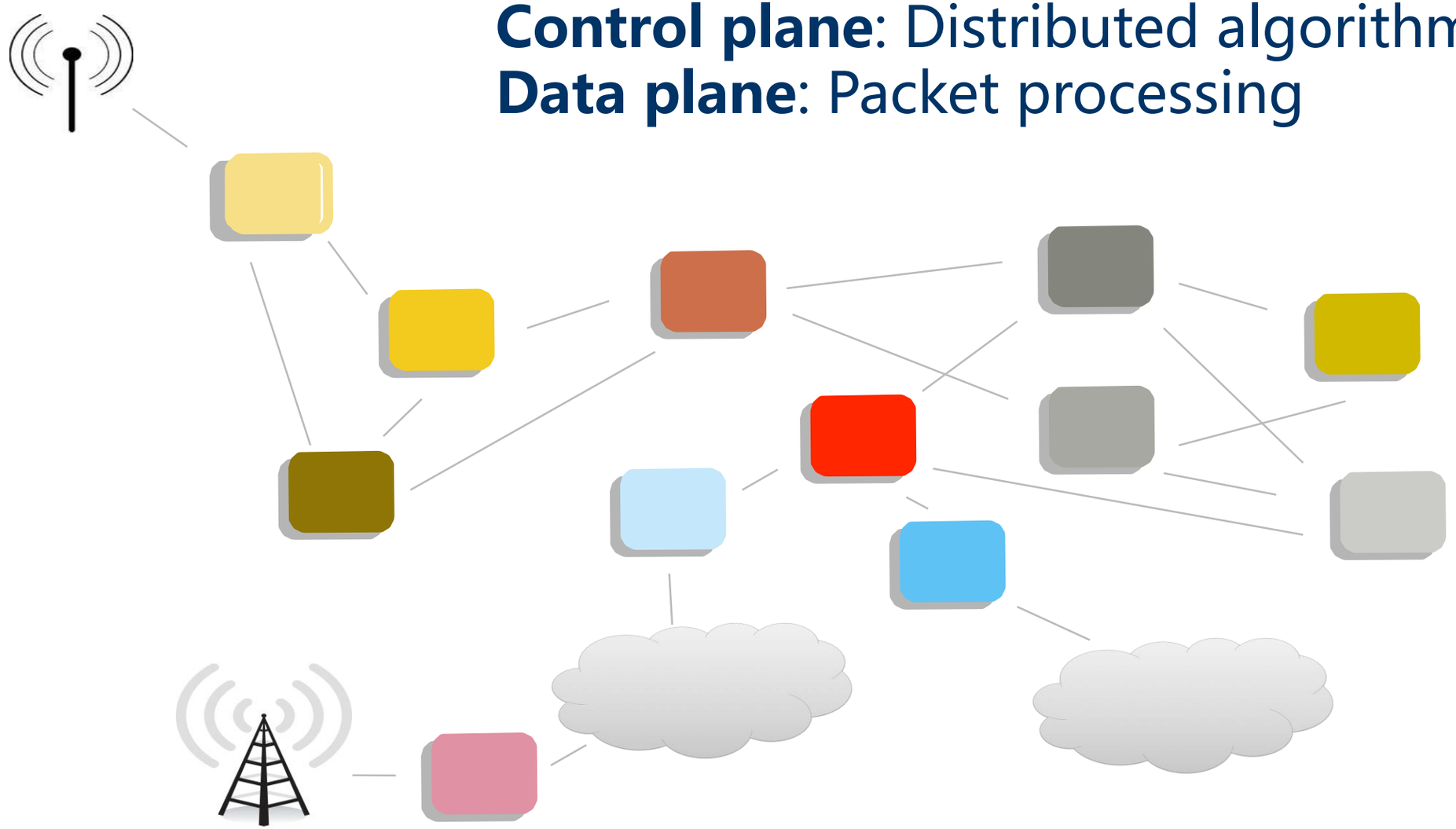
- To solve the limitations faced by the traditional physical network environment operation architecture.
- Software Defined Networking (SDN) architecture proposed to significantly improve the flexibility, efficiency and cost reduction of network operations.
- SDN has become the focus of next-generation network technology development.
- Many index companies have been actively involved

Software Defined Networking

- A new network architecture. Using the OpenFlow protocol, the control plane of the router is separated from the data plane and implemented in software.
- This architecture allows network administrators to re-plan the network in a centrally controlled manner without changing the hardware.
- It provides a new way to control network traffic and provides a good platform for core network and application innovation.
- Three major factors that make SDN important to the enterprise: **automation**, **rapid deployment**, and **simple network management**.

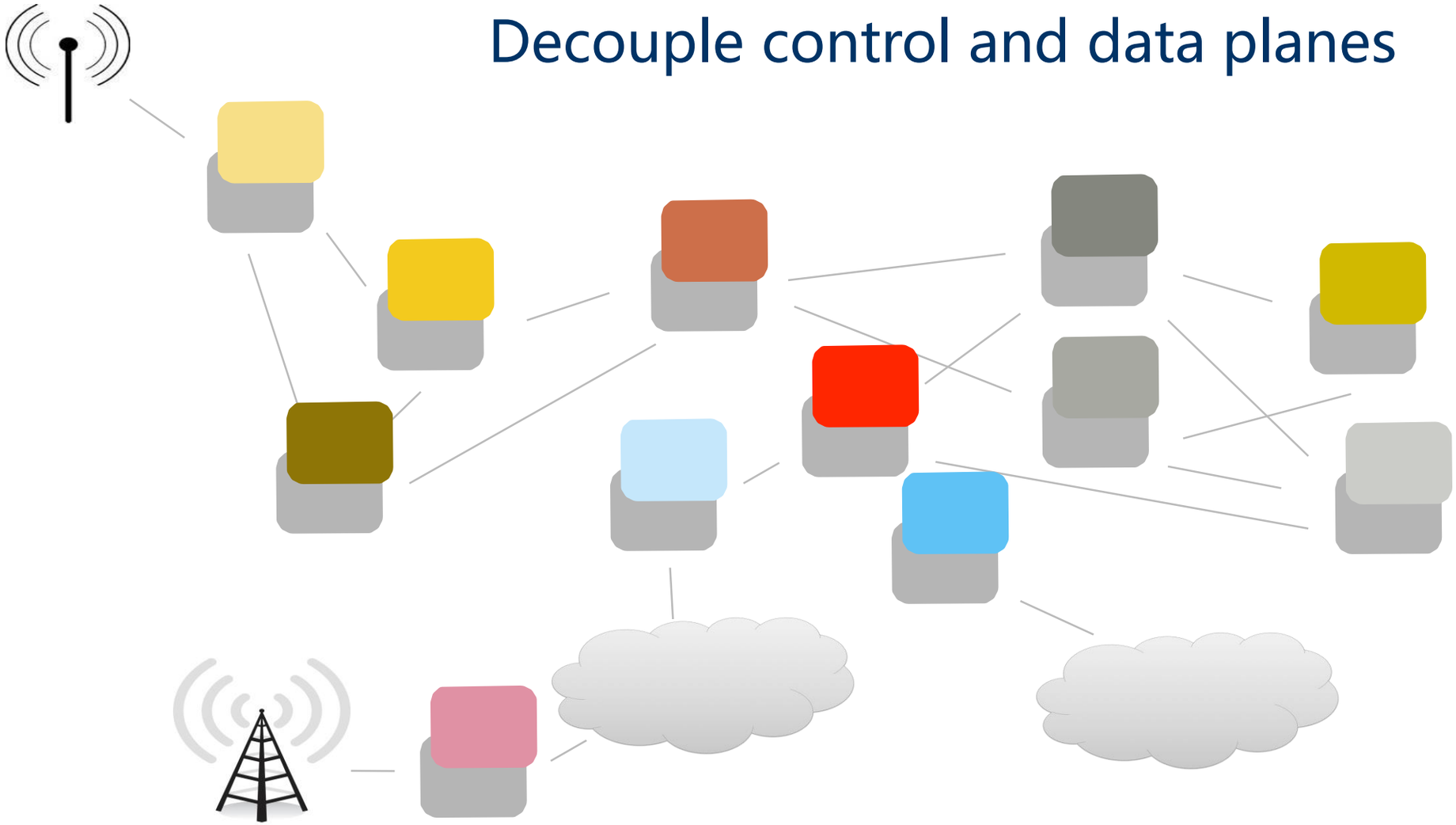
Software Defined Networking

Control plane: Distributed algorithms
Data plane: Packet processing



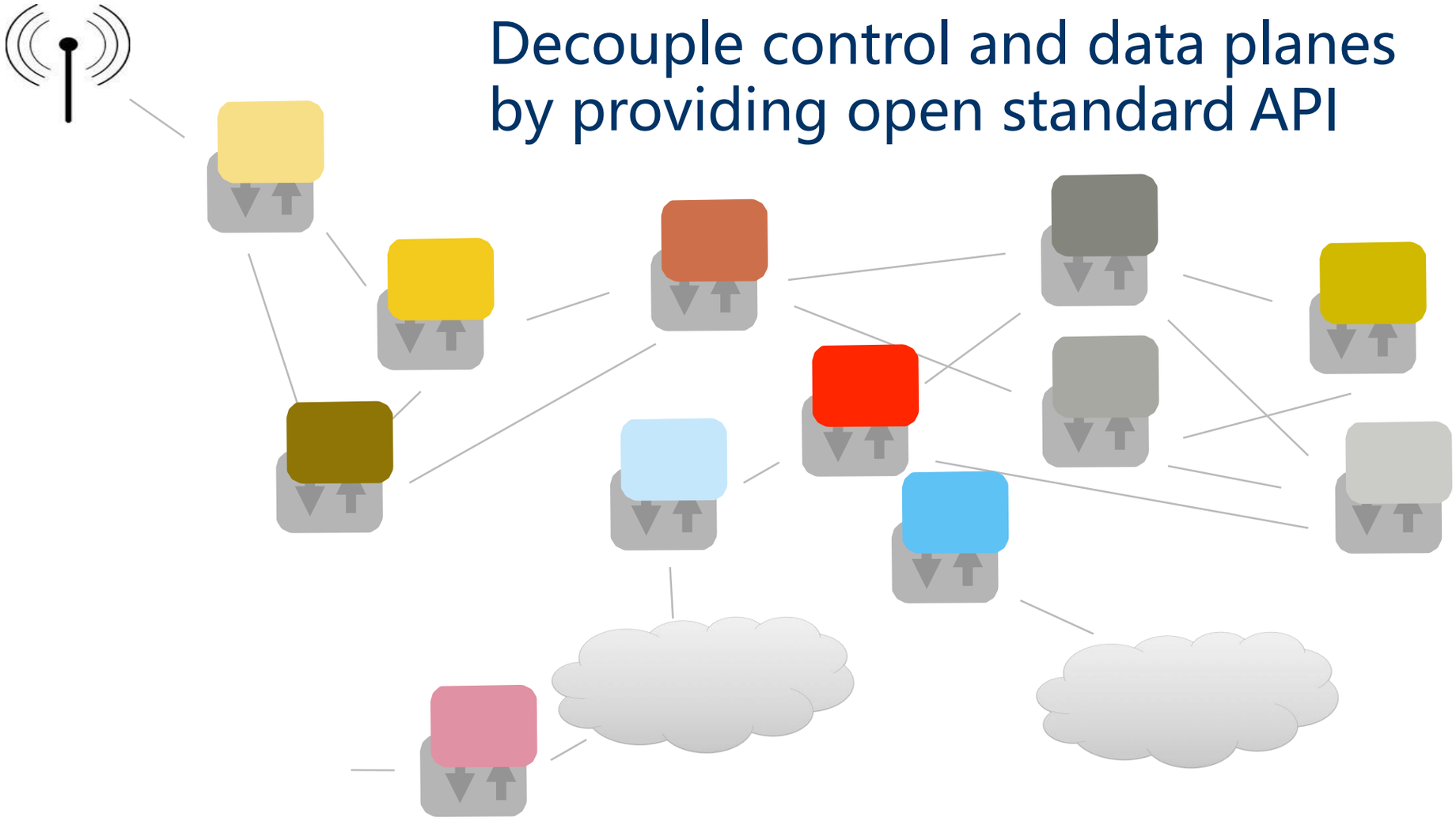
資料來源：蔡孟勳教授SDN/NDV教材—The Road to SDN

Software Defined Networking



資料來源：蔡孟勳教授SDN/NDV教材—The Road to SDN

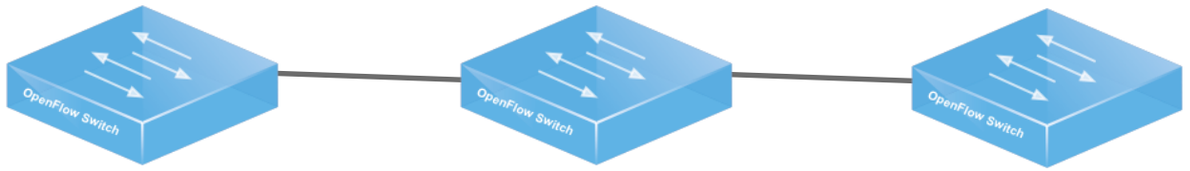
Software Defined Networking



資料來源：蔡孟勳教授SDN/NDV教材—The Road to SDN

Simple, Open Data – Plane API

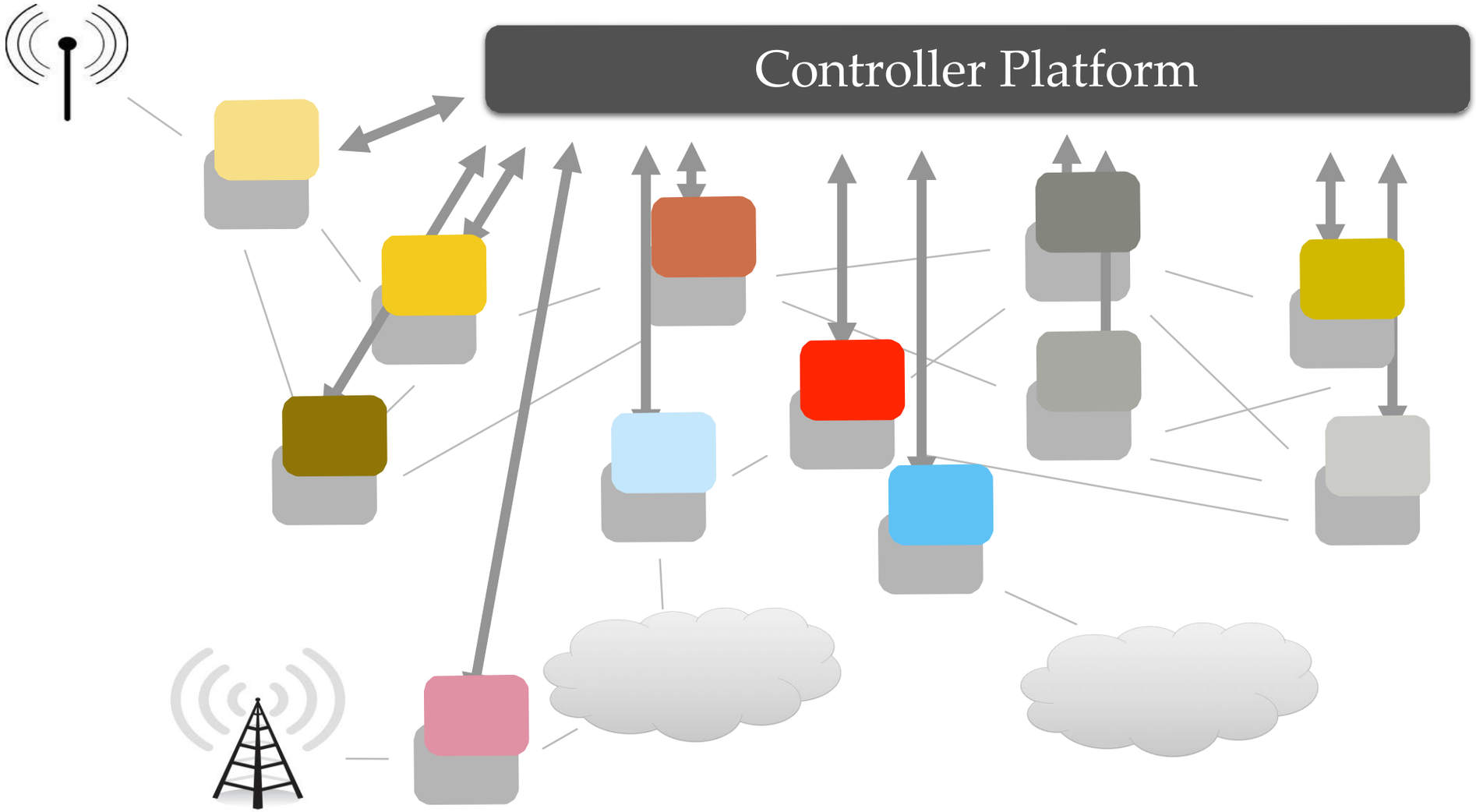
- Prioritized list of rules
  - Pattern: match packet header bits
  - Actions: drop, forward, modify, send to controller
  - Priority: disambiguate overlapping patterns
  - Counters: #bytes and #packets



1. src=1.2.*.*, dest=3.4.5.* → drop
2. src = *.*.*.*, dest=3.4.*.* → forward(2)
3. src=10.1.2.3, dest=*.*.*.* →send to controller

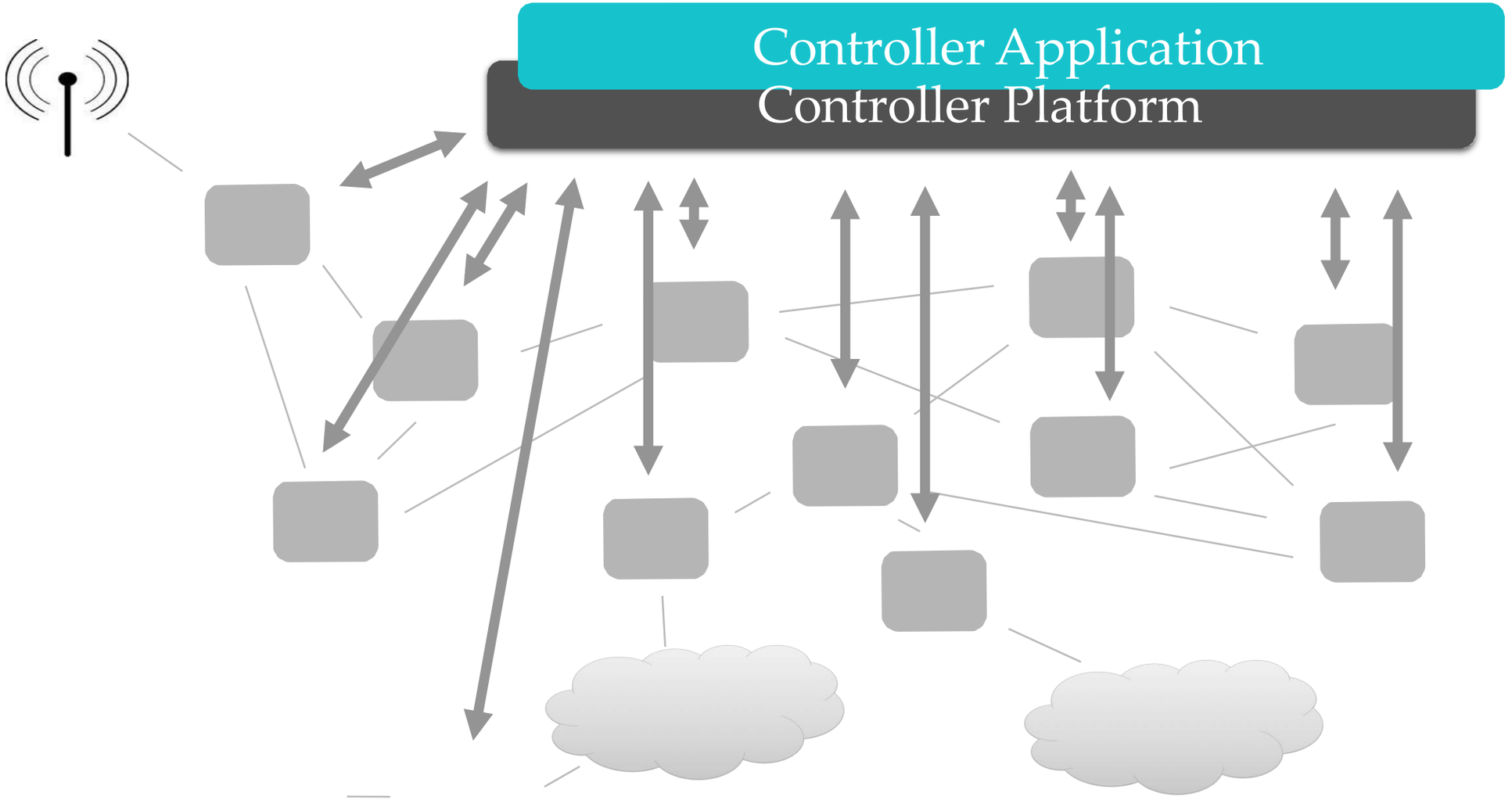
資料來源：蔡孟勳教授SDN/NDV教材—The Road to SDN

Centralized Controller(Logically)



資料來源：蔡孟勳教授SDN/NDV教材—The Road to SDN

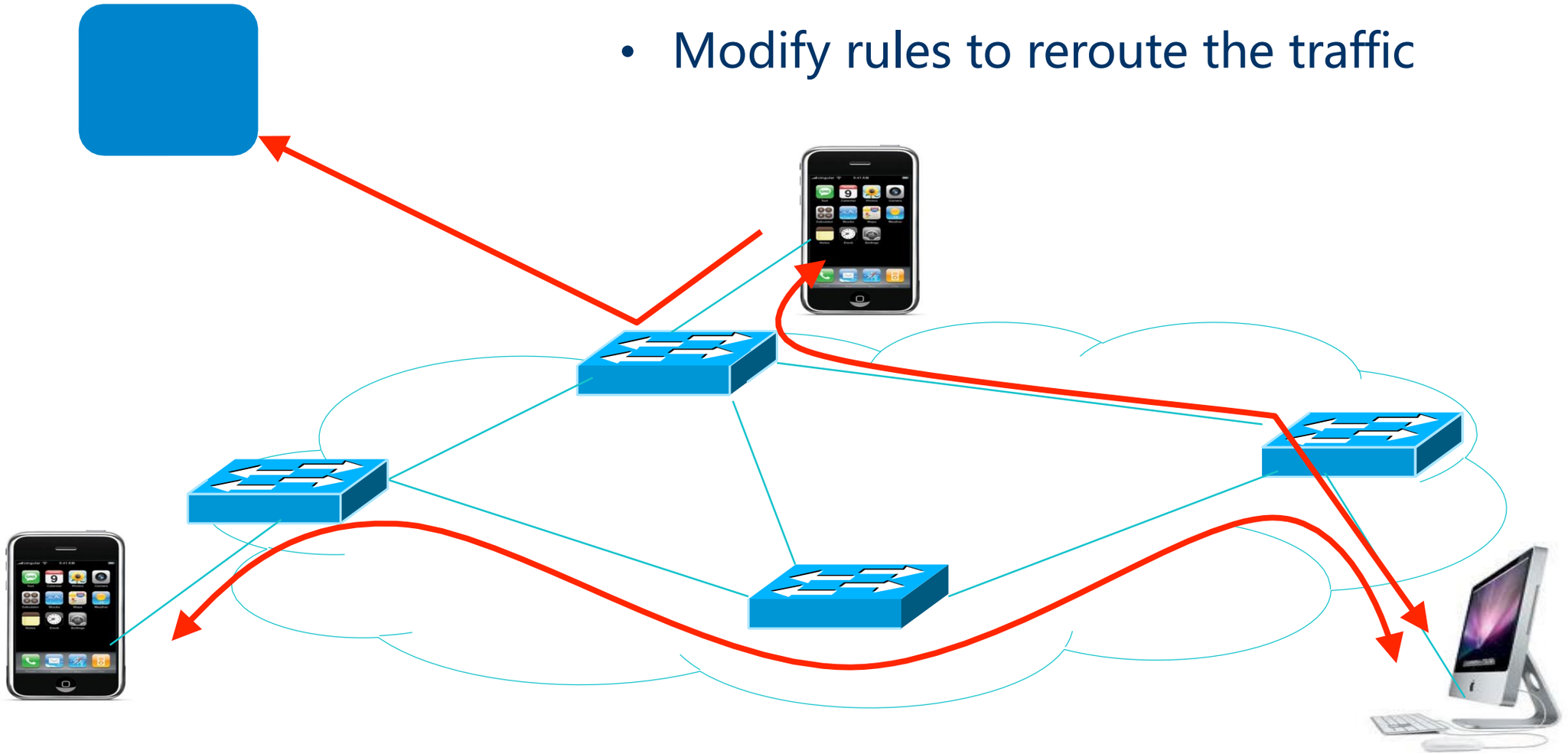
Protocols → Applications



資料來源：蔡孟勳教授SDN/NDV教材—The Road to SDN

Seamless Mobility

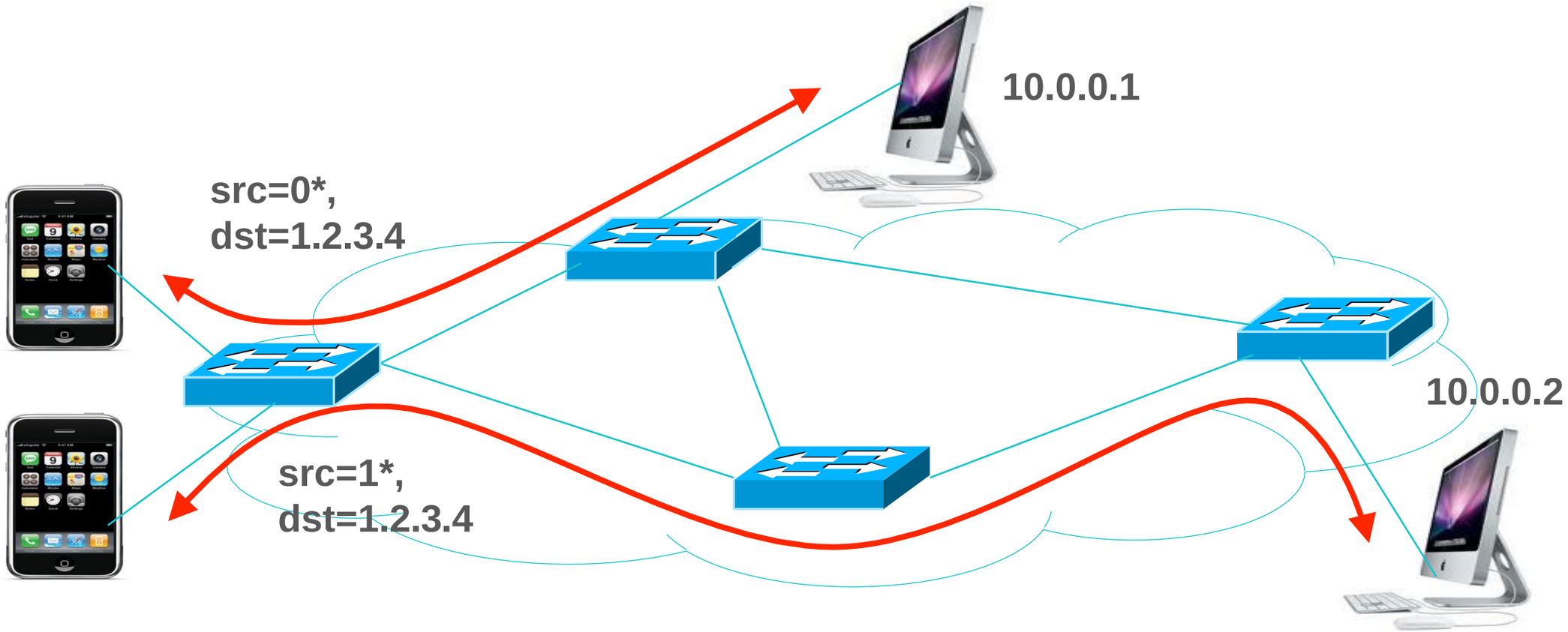
- See host sending traffic at new location
- Modify rules to reroute the traffic



資料來源：蔡孟勳教授SDN/NDV教材—The Road to SDN

Server Load Balancing

- Pre-install load-balancing policy
- Split traffic based on source IP



資料來源：蔡孟勳教授SDN/NDV教材—The Road to SDN

Example SDN Applications

Top Apps and Service that can benefit from SDN are:

- Security services
- Network Monitoring and Intelligence
- Bandwidth Management
- Content Availability
- Regulation and Compliance-Bound Applications
- Distributed Application Control and Cloud Integration
- High –Performance Applications

https://lavellenetworks.com/sdn-applications/

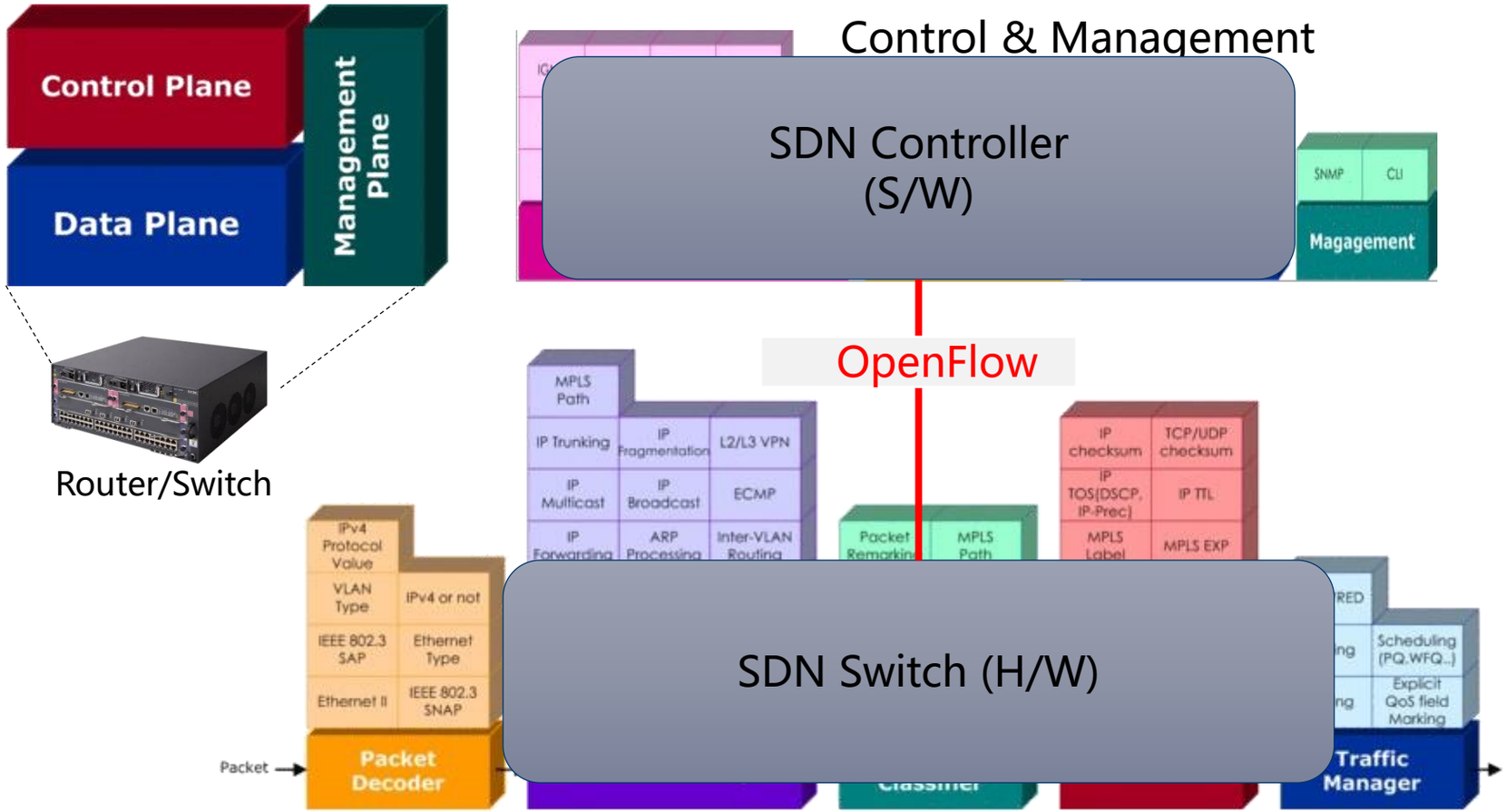
Example SDN Applications

- Seamless mobility and migration
- Server load balancing
- Dynamic access control
- Using multiple wireless access points
- Energy-efficient networking
- Adaptive traffic monitoring
- Denial-of-Service attack detection
- Network virtualization

https://lavellenetworks.com/sdn-applications/

SDN Concept

- SDN separates Control and Data plane functions



(source "Understanding L3 Switch", Netmanias Talk, 2011/11/09)

資料來源: Korea,Postech,Department of Computer Science and Engineering,James Won-Ki Hong: Software Defined Networking — Introduction to SDN&Openflow

SDN Concept

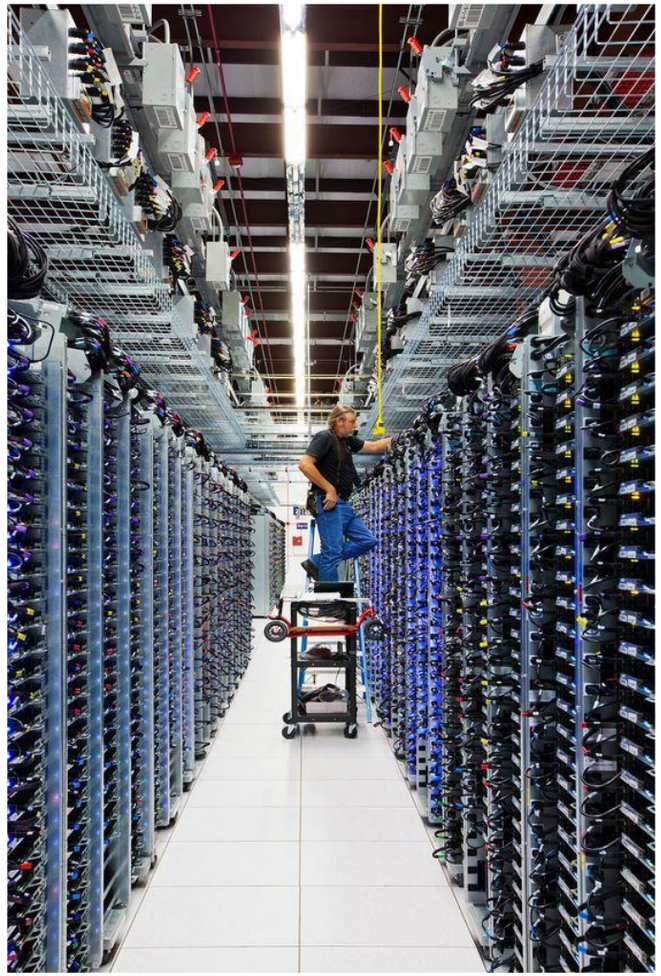
- SDN Concept
  - Separates control plane and data plane entities
    - Network intelligence and state are logically centralized
    - The underlying network infrastructure is abstracted from the applications
  - Execute or run control plane software on general purpose hardware
    - De-couple from specific networking hardware
    - Use commodity computers
  - Have programmable data planes
    - Maintain, control and program data plane state from a central entity
  - An architecture to control not only a networking device but an entire network
    - Similar to existing Network Management System (NMS), but more powerful
- Control Software (SW)
  - Control SW operates on view of network
  - Control SW is not a distributed system
    - Abstraction hides details of distributed states

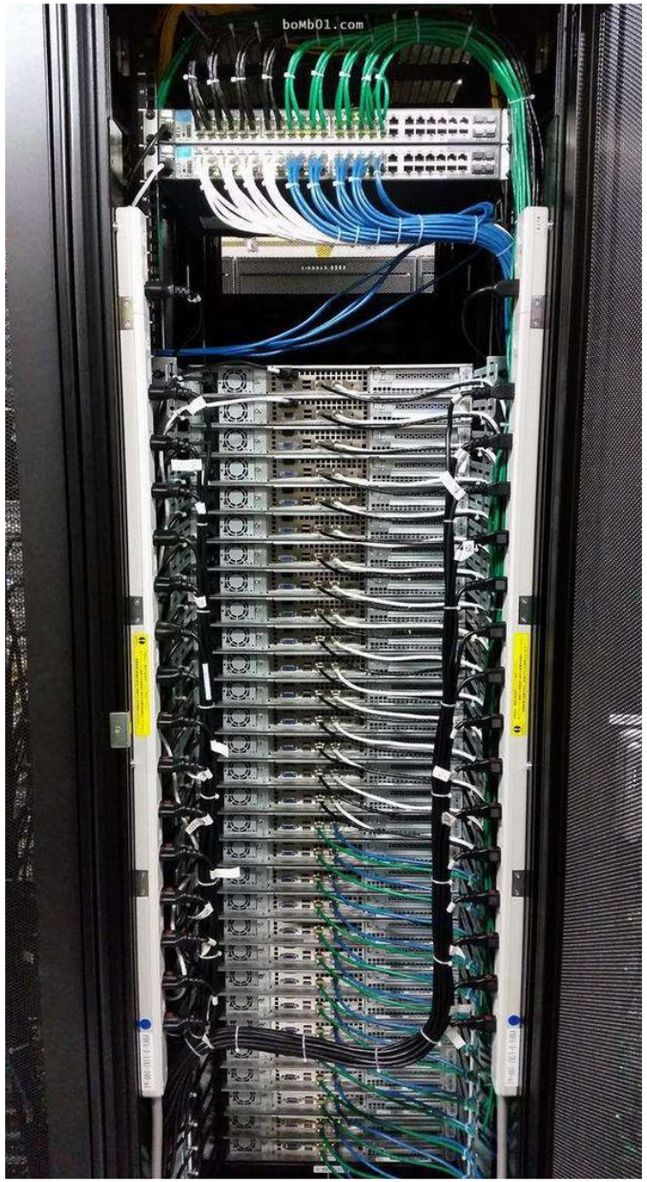
03 Traditional network vs SDN

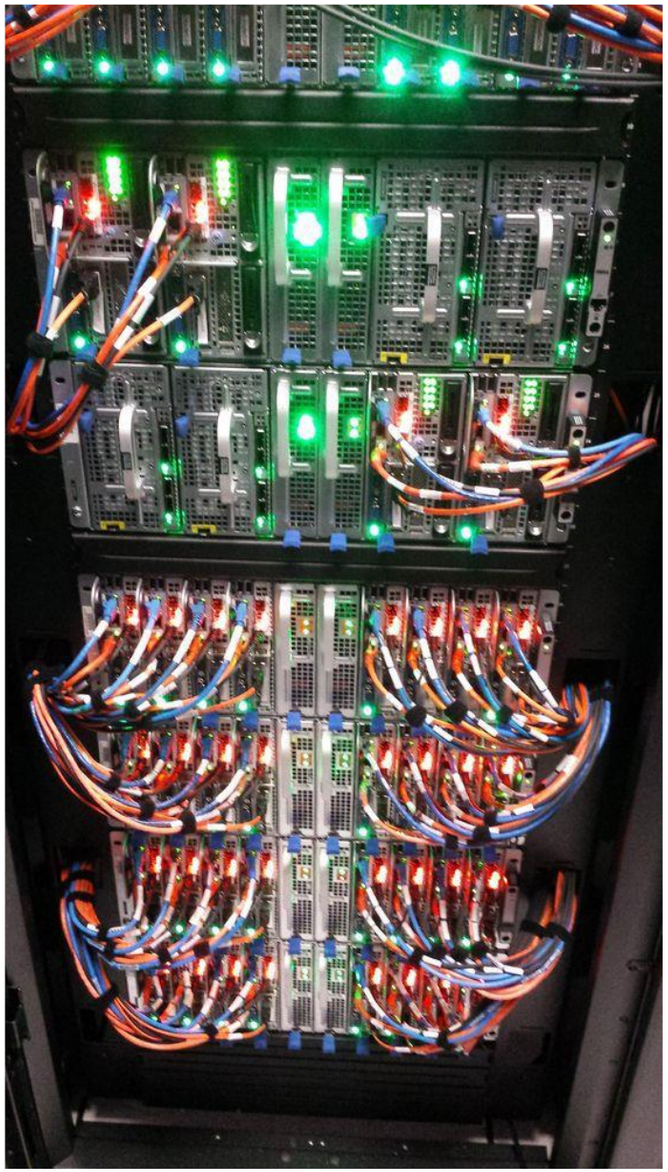
Traditional Network vs SDN

- Traditional Network
- Traditional Network vs SDN
- SDN Architecture
- SDN Scheme
  - Advantage
  - Benefit
  - Misunderstanding
- SDN Commanded by the Controller

Traditional Network Data Center

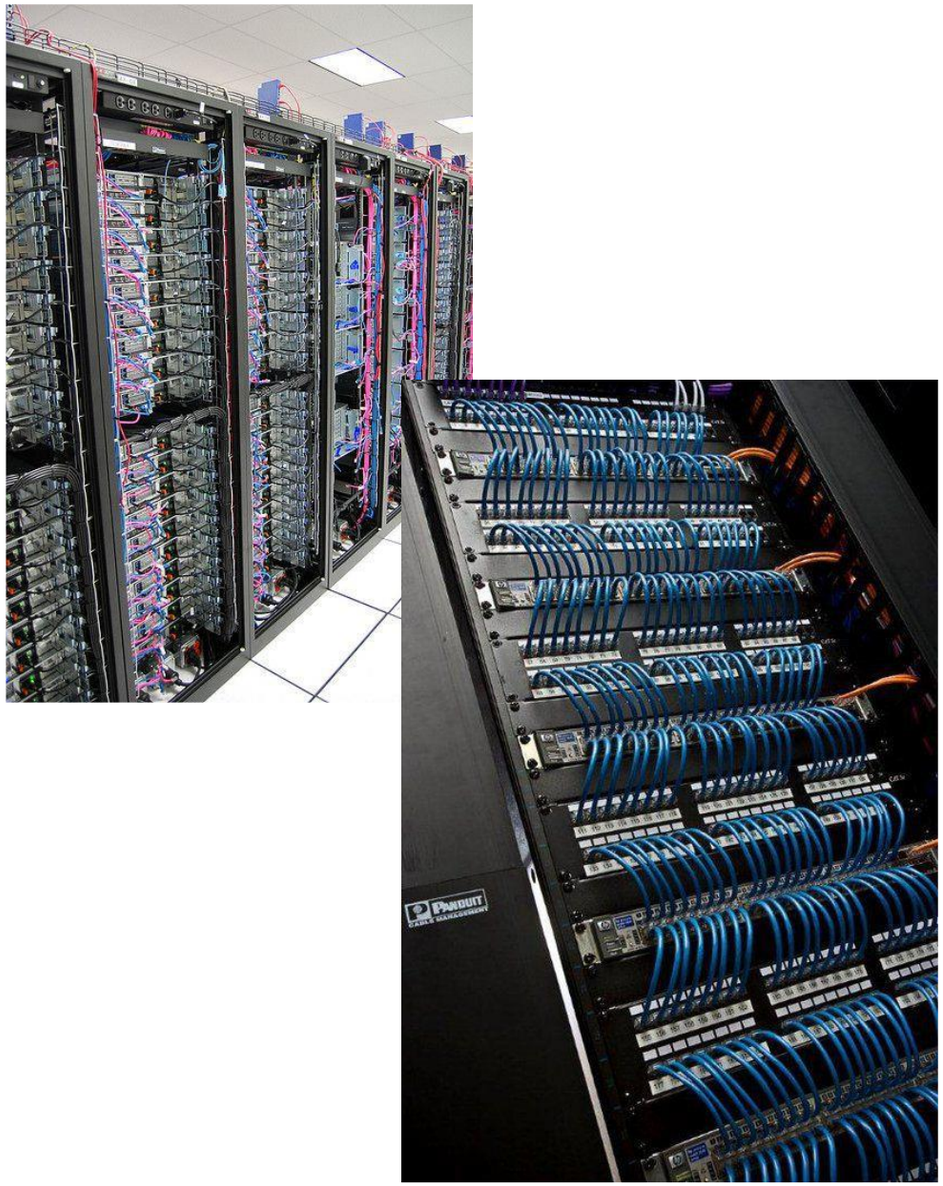


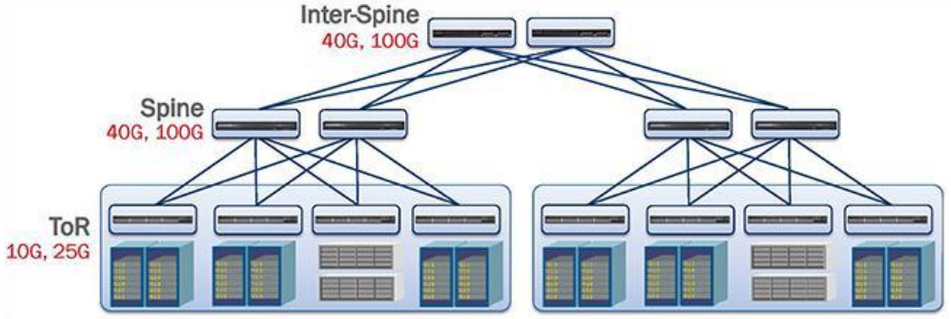


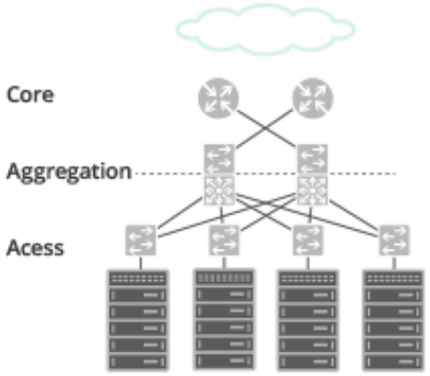


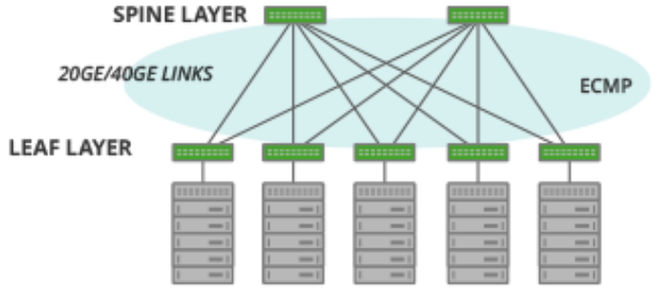
資料來源：數位活氧科技，高銘聰—軟體定義網路(SDN)簡介與發展

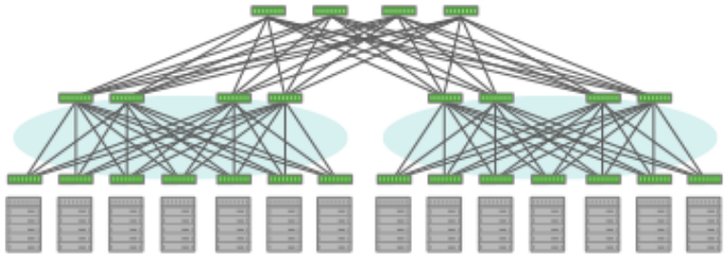
The topology of a traditional network data center

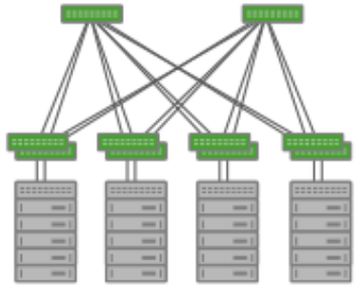






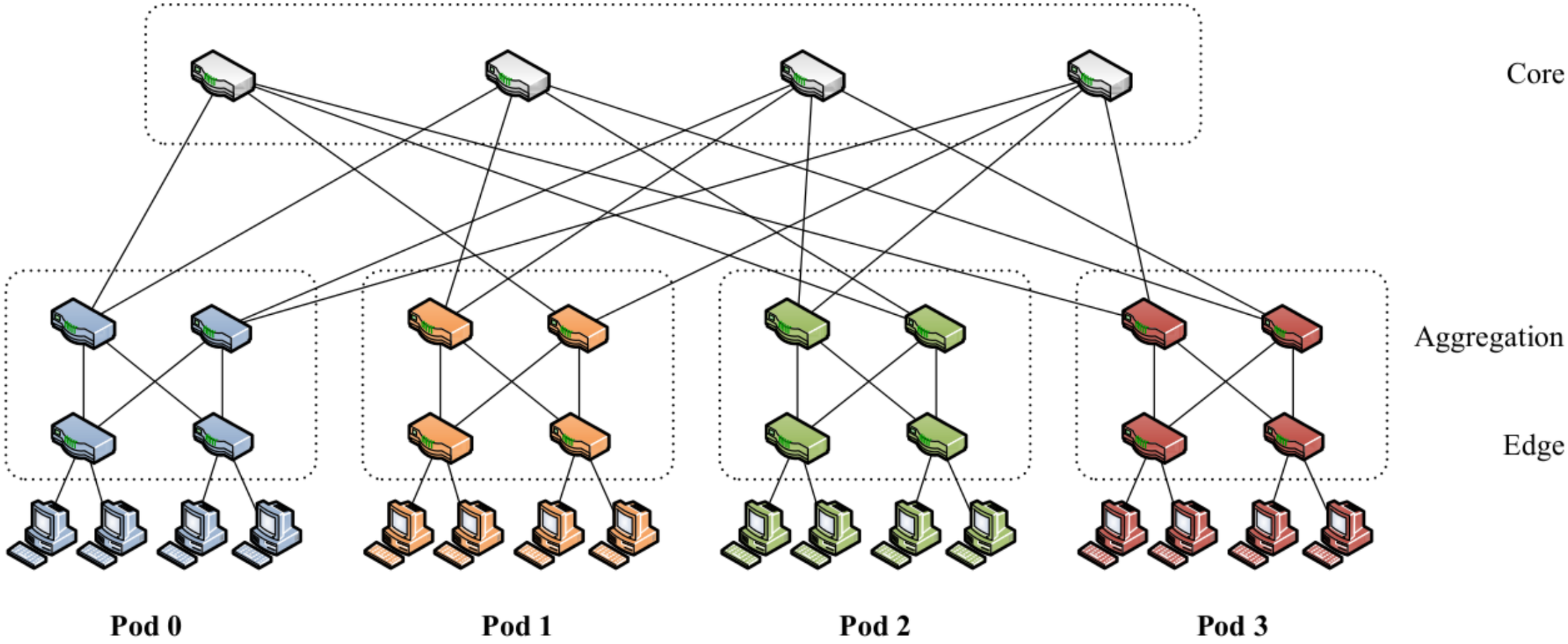






資料來源：數位活氧科技，高銘聰—軟體定義網路(SDN)簡介與發展

The Fat tree topology of Traditional Network



資料來源：數位活氧科技，高銘聰—軟體定義網路(SDN)簡介與發展

The current network architecture is inadequate

- Today's network architecture is a three-tier architecture built on the Spanning Tree Protocol (STP) that delivers packets over a variety of transport protocols.

- However, with the increasing demand for cloud application services and huge amounts of data, the routing tables of the Internet have become more and more complex, which has caused many problems in the current network architecture and is becoming more and more inadequate.

- In order to implement various network protocols, switches or routers must constantly split and reassemble packets, resulting in poor transmission efficiency and ineffective network bandwidth.

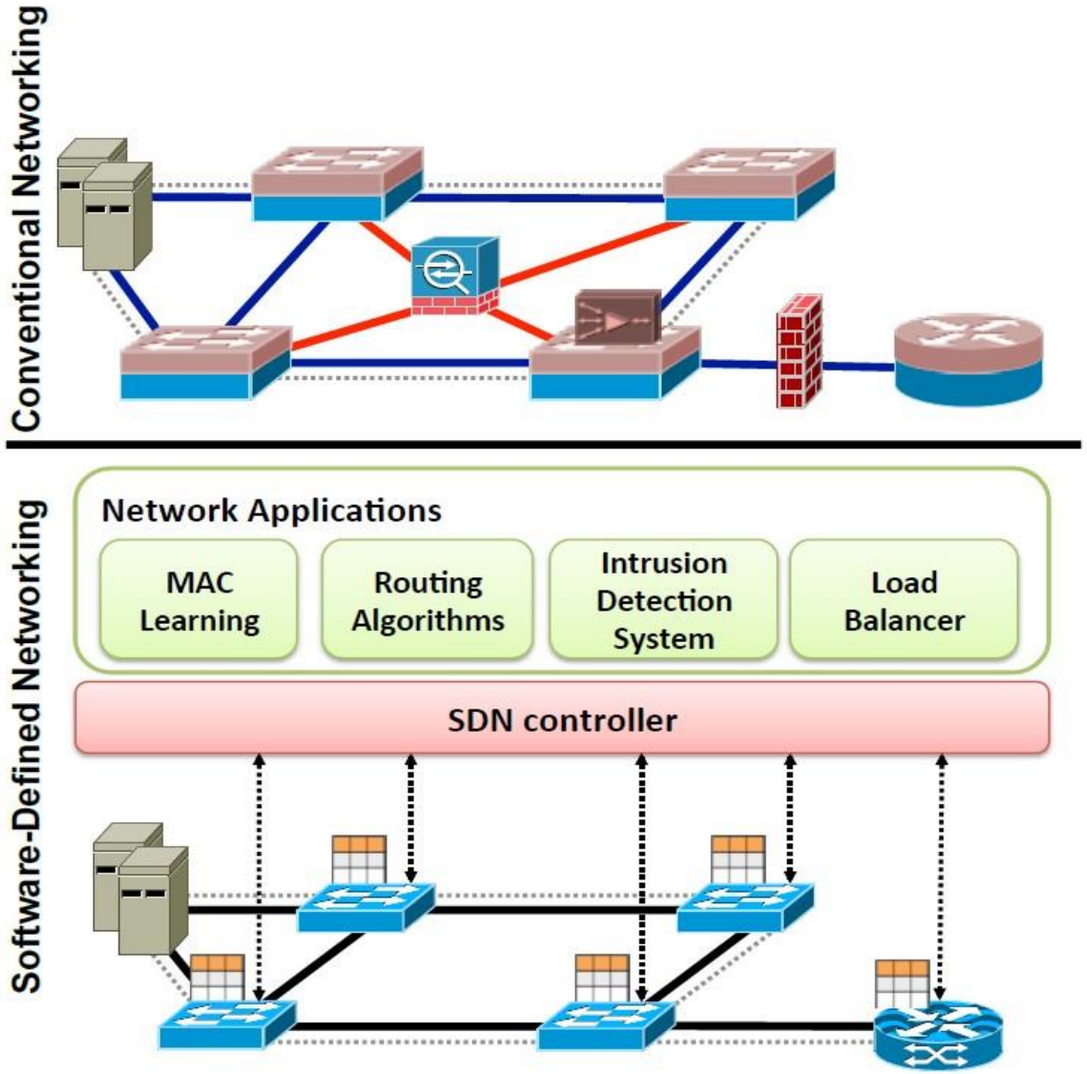
The current network architecture is inadequate

- When network administrators needed, use the command-line interface (CLI) settings for each switch or router.

  Troublesome, high risk in manually setting one by one, easy to cause network service failure.

- Network management software is difficult to be compatible with each other.

Traditional Network vs SDN

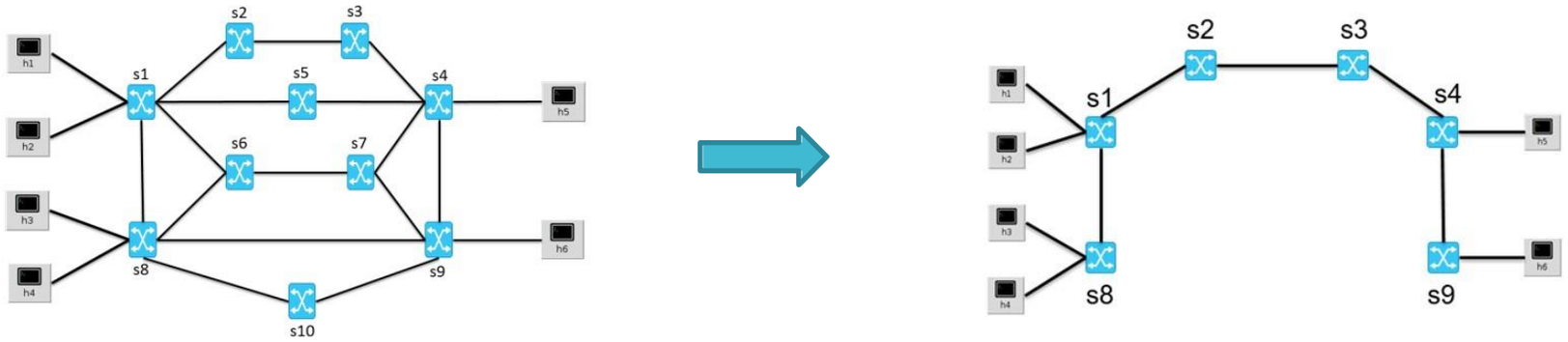


資料來源：數位活氧科技，高銘聰—軟體定義網路(SDN)簡介與發展

Traditional Network vs SDN

- Ethernet network use Spanning Tree Protocol(STP) to forward frame.
  - IEEE 802.1D
  - Avoid loop and ARP storm
  - Analysis and decide which port can transmit
  - Single path routing

- STP



資料來源：數位活氧科技，高銘聰—軟體定義網路(SDN)簡介與發展

Traditional Network vs SDN

- Load balancing
  - Achieve higher bandwidth utilization
  - Balancing the traffic load
  - Static load balancing
  - Dynamic load balancing

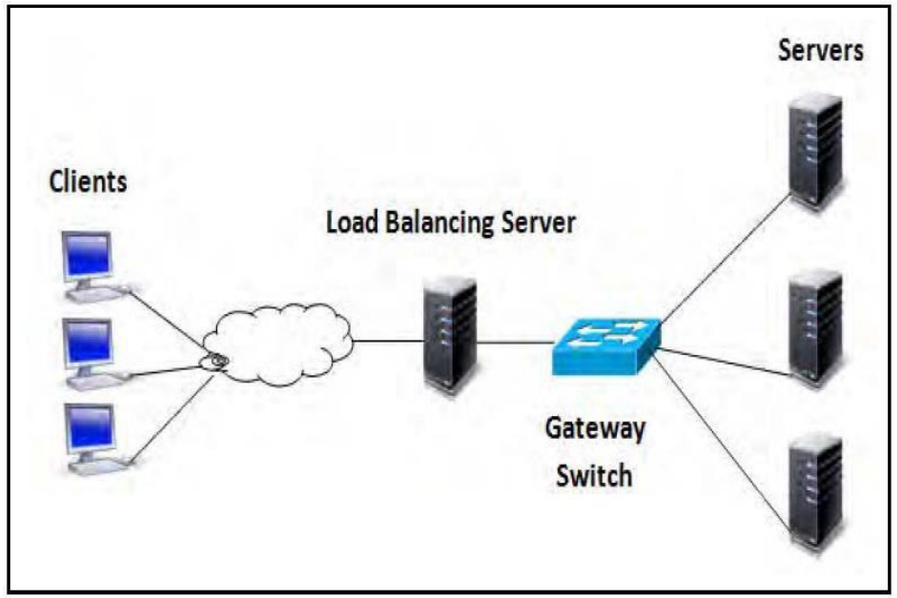


Figure 1. Ttraditional load balancing model

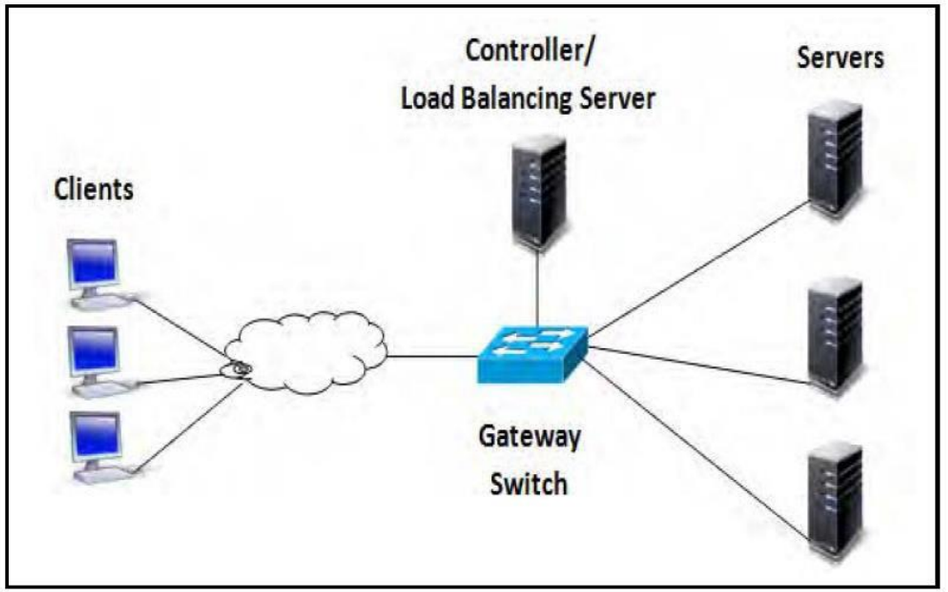


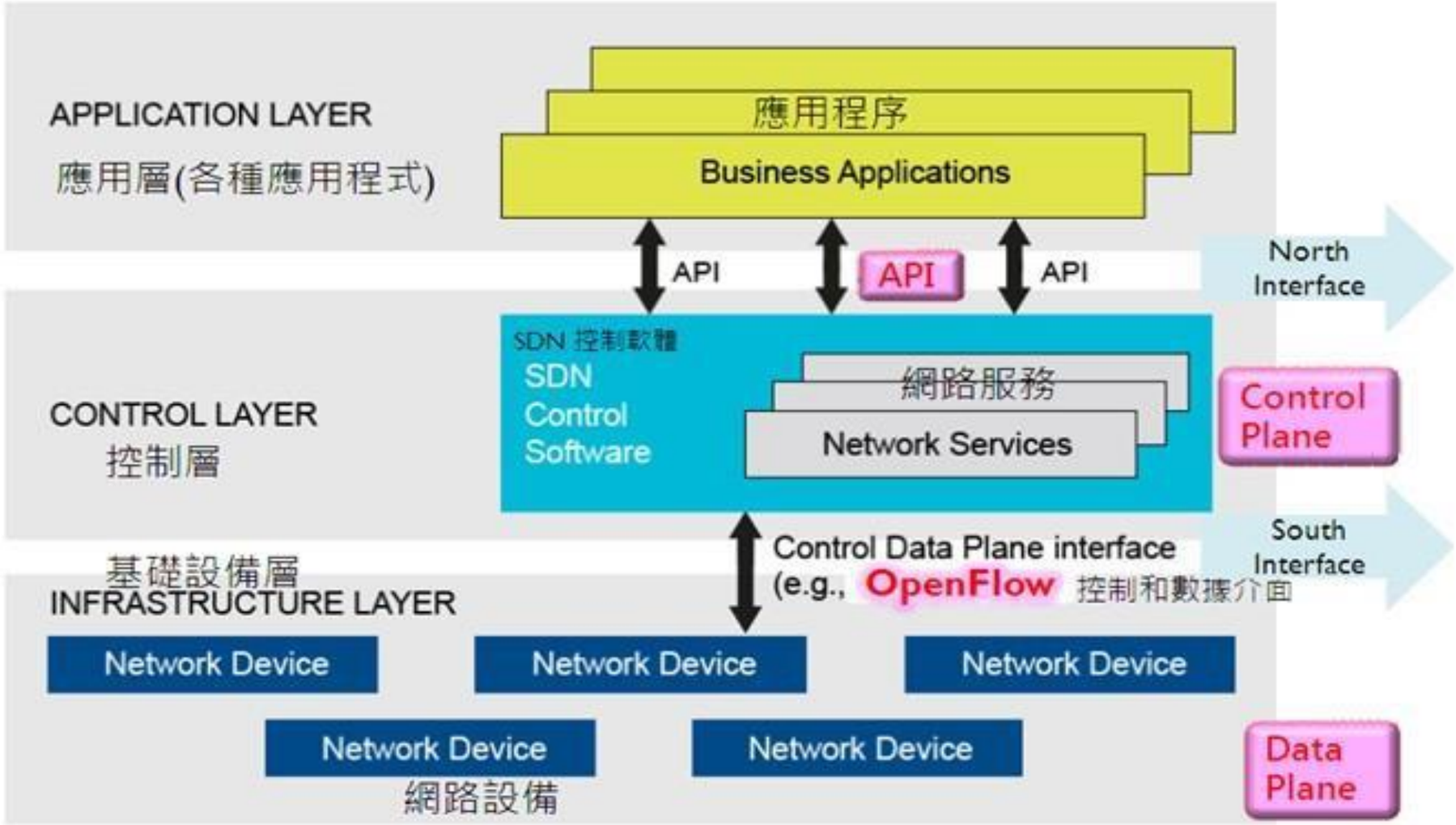
Figure 2. SDN based load balancing model

資料來源：數位活氧科技，高銘聰—軟體定義網路(SDN)簡介與發展

SDN is a new generation network concept and architecture

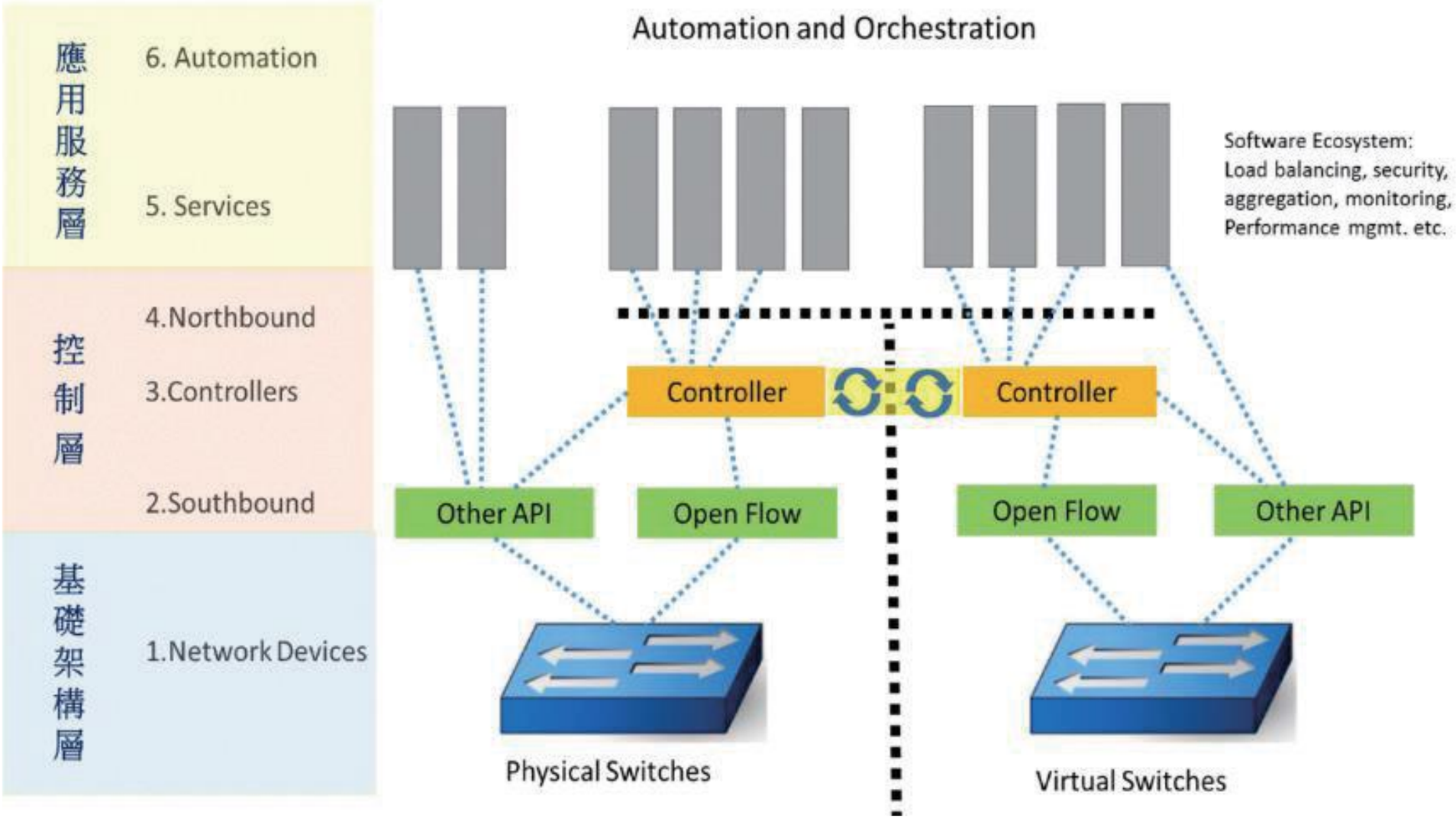
- The control plane of the network is completely independent of the Data plane.
- Centrally manage data traffic behavior across the network through the Controller software.
- The Controller software provides a application programming interface (API) for further integration with other upper-layer devices such as VMs.
- With the API, users can develop a variety of additional services on the Controller, such as Firewall, IDP.
- DPI (Deep Packet Inspection), LB (Load Balance), Schedule ..., etc., can be deployed in a unified manner to provide more diversified service projects for enterprises.

SDN Architecture (1/4)



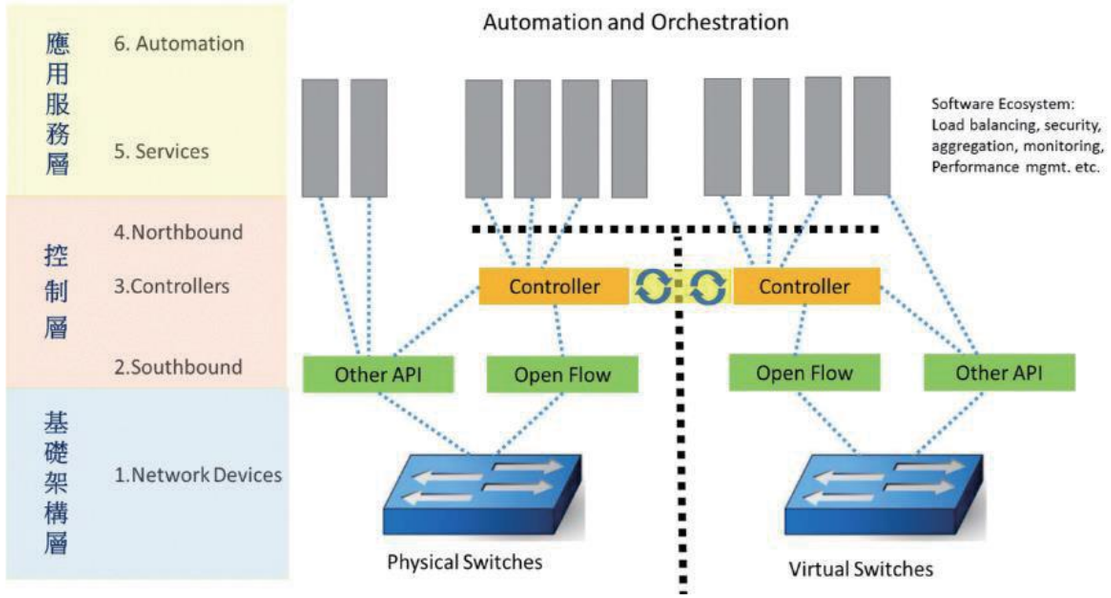
資料來源：數位活氧科技，高銘聰—軟體定義網路(SDN)簡介與發展

SDN Architecture (2/4)



資料來源：台灣期貨雙月刊 2019年4月號，關鍵看法—軟體定義網路(SDN)架構之應用與探討

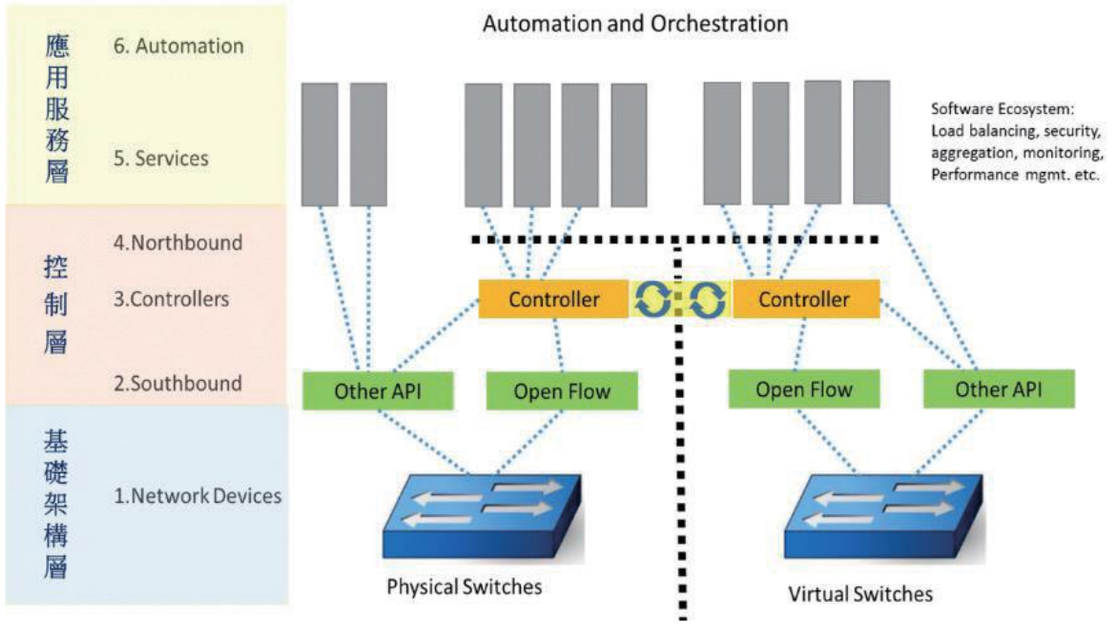
SDN Architecture (3/4)



- **Network Devices:** switch, router, virtual switch, or abstract forwarding plane (Forwarding/Data Plane). All forwarding rules are stored in the network device, and the user data packets are processed and forwarded here. The network device receives the command sent by the controller through the southbound interface, and also actively reports the event to the controller through the southbound interface.
- **Southbound Interface:** between the control plane and the data forwarding layer. The traditional network exists in the private code of each device vendor and is not standardized. In SDN the southbound interface is standardized, such as the Openflow standard interface.
- **Controller:** The core elements of the SDN network provide up to the application's programming interface and down control of the hardware. Usually run on a separate server, such as an x86 Linux server or Windows server.

資料來源：台灣期貨雙月刊 2019年4月號，關鍵看法—軟體定義網路(SDN)架構之應用與探討

SDN Architecture (4/4)



- **Northbound Interface:** In the traditional network, the northbound interface refers to the interface between the switch control plane and the network management software. In the SDN architecture, it refers to the interface between the controller and the application.
- **Service:** Control and manage the network in the form of software applications, such as: Load Balancing, Security, Monitoring (including congestion and latency, network performance management and detection), LLDP (topology detection) and other functions.
- **Automation:** Automation is the packaging and integration of applications. It usually comes with Orchestration, such as including multiple applications and services in a system management framework, and regularly collecting device line load through the controller.

資料來源：台灣期貨雙月刊 2019年4月號，關鍵看法—軟體定義網路(SDN)架構之應用與探討

SDN Scheme — Advantages

- Higher automation and reduces the misconfiguration of enterprises caused by humans.
- With SDN, customers only need to select the applications and necessary resources they want to run in the cloud, and the control plane intuitively deploys services using the optimal configuration of compute, storage, and network resources.
- Quickly deploy and scale your application can make a business or ruin a business.
- In addition to making employees easy to access, SDN can quickly respond to changing business and reduce the time it takes for new products to enter the market.
- SDN will greatly change the way the network infrastructure is configured and managed. By separating the control functions from the rest of the network, SDN allows IT teams to manage the network environment in a bird's eye view of the business so that each business do not operate in isolation

SDN Scheme — benefit

- Developable applications make network data traffic more flexible and bandwidth usage more efficient.
- Equivalent to traffic engineering and know the status of real-time traffic.
- Dynamically change the traffic path based on bandwidth usage to increase network usage
- Can be added to the schedule for flexible use.
- Reduce the cost of maintenance manpower or equipment.
- Uniform control, easy to operate and manage.
- Improve the speed of obstacle removal.
- Centralized management, single inspection.
- Unlimited equipment brand, unified operation mode.
- The same standard, across the label restrictions.
- Flexible and variable value-added development space.
- It can integrate future FW, IDP (Intrusion Detection and Defense), DPI, VM, LB, Schedule., etc. to provide diversified services.

SDN Scheme — Misunderstanding (1/2)

Like any new technology,as long as SDN exists,there must be people who argue the toss.For any business, you want to understand the truth behind the biggest misunderstandings before deploying an SDN solution.

- **SDN is not suitable for small data centers**

  People tend to think that it is only suitable for large data centers (that is,data centers that provide public,private,and hybrid cloud services) when SDN is mentioned. Although these larger providers are early adopters, in fact, SDN is beneficial for all levels of data centers. Not only does it make configuration, management, and monitoring tasks simple, it also greatly reduces the burden on the IT department, which is the perfect choice for small companies with a lean team.

- **SDN means that many IT jobs will disappear**

  An SDN-enabled environment requires less manual work to maintain normal operation than traditional network environments. This statement is true, but that does not mean that traditional network management positions will disappear. As enterprises transition to SDN mode, networking skills evolve, so the demand for network skills also increased. In fact, the type of skills needed for the new era of IP will continue to change. Business and IT professionals should be aware of this, and accordingly tailor their own training and development programs.

SDN Scheme — Misunderstanding(2/2)

- **If the server is already virtualized, you don’ t need SDN.**
  This is not true. Extending the principles of server virtualization to the network by replacing traditional hardware with a more flexible virtualized network infrastructure will bring more of the same important benefits.SDN can also play a greater role, particularly it allows to extend the network to the server is provided and more efficient management of traffic between the servers can be visualized.

- **To Implement SDN,the entire data center network must be replaced.**
  “Dismantling the existing system " is not a necessary condition for successful implementation of SDN. The more scientific method is to gradually migrate from traditional network infrastructure to SDN.In fact, Implementing SDN is very simple: use SDN devices as a default choice for network components, as part of an existing hardware update plan; or deploy SDN when new projects or expansions need to add new devices.

SDN Commanded by the Controller

- The management authority of the network is transferred to the controller (Controller) software of the control layer, and the centralized control is adopted.

- The controller software is like a human brain, and the instructions are given to the network device. The network device is dedicated to the transmission of the packet, just like the human limbs are responsible for performing various actions. This concept allows network administrators to configure network resources more flexibly. In the future, network administrators can set up automation automatically by simply issuing commands to the controller. They do not need to log in to the network device one by one to make individual settings.

04 Openflow Introduction

Openflow Introduction

- Openflow
  - Introduction
  - Standardization
  - Overview
  - Opneflow Switch
  - Openflow Controller
  - Openflow Building blocks
  - Components of Openflow Network
- How does Openflow work
  - Usage
  - Flow table and flow table entries
  - Example

Openflow Introduction

- OpenFlow technology is a communication protocol used to establish a transmission channel between the control layer and the data layer.
- OpenFlow technology regards the path of packet transmission as a “Flow”, just like a dedicated transmission path. The network administrator can determine the packet transmission mode by setting various network management functions on the controller software and pre-establishing the logical network according to the enterprise policy or service level agreement (SLA).

Openflow Introduction(cont.)

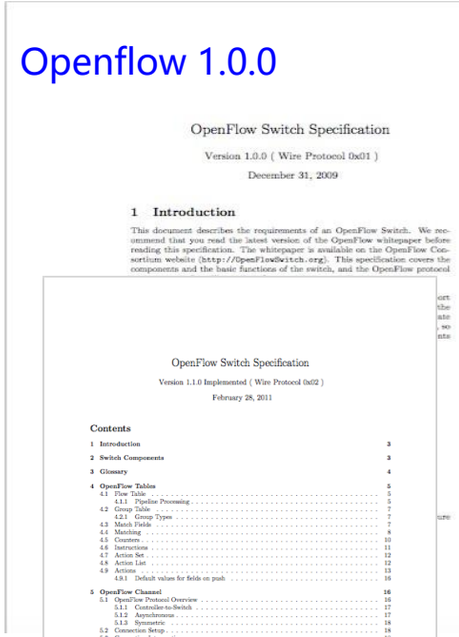
- Then, a secure transmission channel is established between the control layer and the data layer by using SSL encryption technology, and the controller transmits the set OpenFlow routing table to the network device of the data layer through the transmission channel for packet delivery. Because the transmission path is pre-set, the switch does not need to continuously learn to find the path of the packet transmission, which can greatly improve the transmission efficiency and reduce the delay time.

- In the future, enterprises only need to update their OpenFlow firmware provided by the manufacturer. In other words, no matter which manufacturer purchases the network equipment that supports OpenFlow technology, it will be managed by the controller, and the problem of being bound by a single network communication vendor can be solved.

Standardization of Openflow

- The nonprofit Internet organization openflow.org was created in 2008 as a mooring to promote and support OpenFlow. The physical organization was really just a group of people that met informally at Stanford University.
- The first release, OpenFlow 1.0.0, appeared on Dec. 31, 2009. Later, OpenFlow 1.1.0 was released on Feb. 28, 2011.
- On March 21, 2011, the Open Network Foundation (ONF) was created for the express purpose of accelerating the delivery and commercialization of SDN.

Openflow 1.0.0



OpenFlow Switch Specification

Version 1.0.0 (Wire Protocol 0x01)

December 31, 2009

1 Introduction

This document describes the requirements of an OpenFlow Switch. We recommend that you read the latest version of the OpenFlow whitepaper before reading this specification. The whitepaper is available on the OpenFlow Consortium website (http://OpenFlowSwitch.org). This specification covers the components and the basic functions of the switch, and the OpenFlow protocol

OpenFlow Switch Specification

Version 1.1.0 Implemented (Wire Protocol 0x02)

February 28, 2011

Contents

1

Openflow 1.1.0

資料來源：蔡孟勳教授SDN/NDV教材—Openflow

Openflow Switch

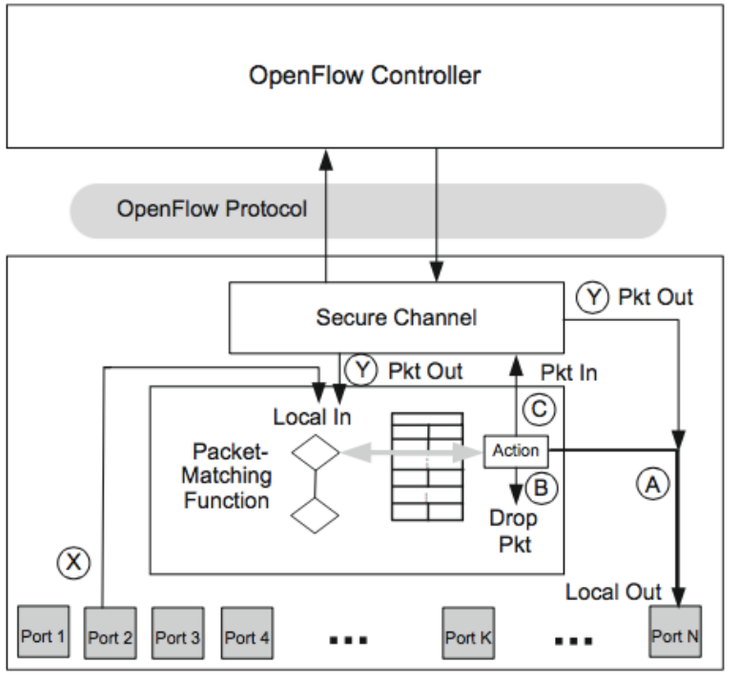
- The packet-matching function tries to match the incoming packet (X) with an entry in flow table, and then directs the packet to an action box.

- The action box has three fundamental options:

(A)Forward the packet out , possibly modifying certain header fields first.

(B)Drop the packet.
Pass the packet to the controller

(C)Through a OpenFlow PACKET_IN message.



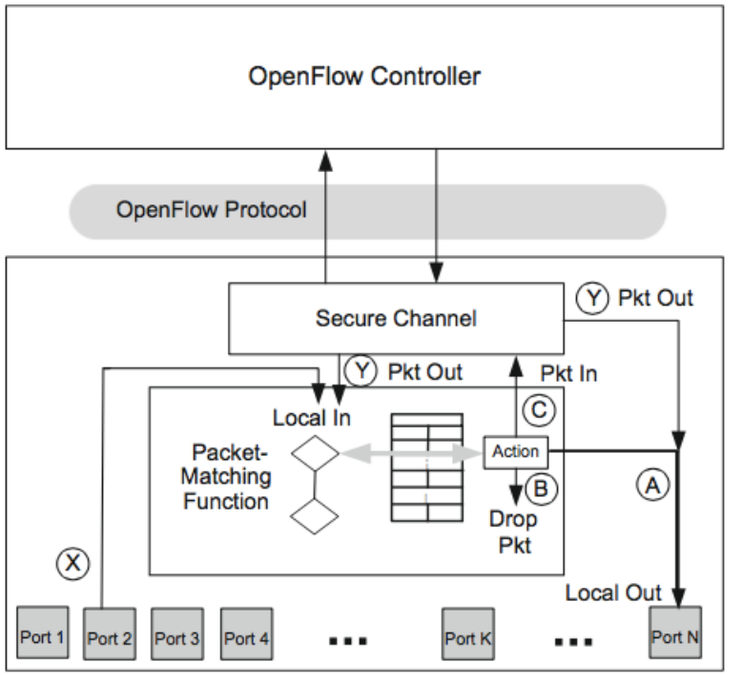
資料來源：蔡孟勳教授SDN/NDV教材—Openflow

Openflow Switch(cont.)

- The packets are transferred between the controller and the switch through secure channel.

- When the controller has a data packet to forward out through the switch, it uses the OpenFlow PACKET_OUT message. Two paths are possible:

  (1)Controller directly specifies the output port.
  (2)Controller defer the forwarding decision to the packet-matching logic.



資料來源：蔡孟勳教授SDN/NDV教材—Openflow

Openflow Controller

- The OpenFlow control plane differs from the legacy control plane in three key ways:
  - It can program different data plane elements with a common and standard language, OpenFlow.
  - It exists on a separate hardware device than the forwarding plane.
  - The controller can program multiple data plane elements from a single control plane instance.

Openflow Controllers

	POX	Ryu	Trema	FloodLight	OpenDaylight
Interfaces	SB (OpenFlow)	SB (OpenFlow) +SB Management (OVSDB JSON)	SB (OpenFlow)	SB (OpenFlow) NB (Java & REST)	SB (OpenFlow & Others SB Protocols) NB (REST & Java RPC)
Virtualization	Mininet & Open vSwitch	Mininet & Open vSwitch	Built-in Emulation Virtual Tool	Mininet & Open vSwitch	Mininet & Open vSwitch
GUI	Yes	Yes (Initial Phase)	No	Web UI (Using REST)	Yes
REST API	No	Yes (For SB Interface only)	No	Yes	Yes
Productivity	Medium	Medium	High	Medium	Medium
Open Source	Yes	Yes	Yes	Yes	Yes
Documentation	Poor	Medium	Medium	Good	Medium
Language Support	Python	Python-Specific + Message Passing Reference	C/Ruby	Java + Any language that uses REST	Java
Modularity	Medium	Medium	Medium	High	High
Platform Support	Linux, Mac OS, and Windows	Most Supported on Linux	Linux Only	Linux, Mac & Windows	Linux
TLS Support	Yes	Yes	Yes	Yes	Yes
Age	1 year	1 year	2 years	2 years	2 Month
OpenFlow Support	OF v1.0	OF v1.0 v2.0 v3.0 & Nicira Extensions	OF v1.0	OF v1.0	OF v1.0
OpenStack Networking (Quantum)	NO	Strong	Weak	Medium	Medium

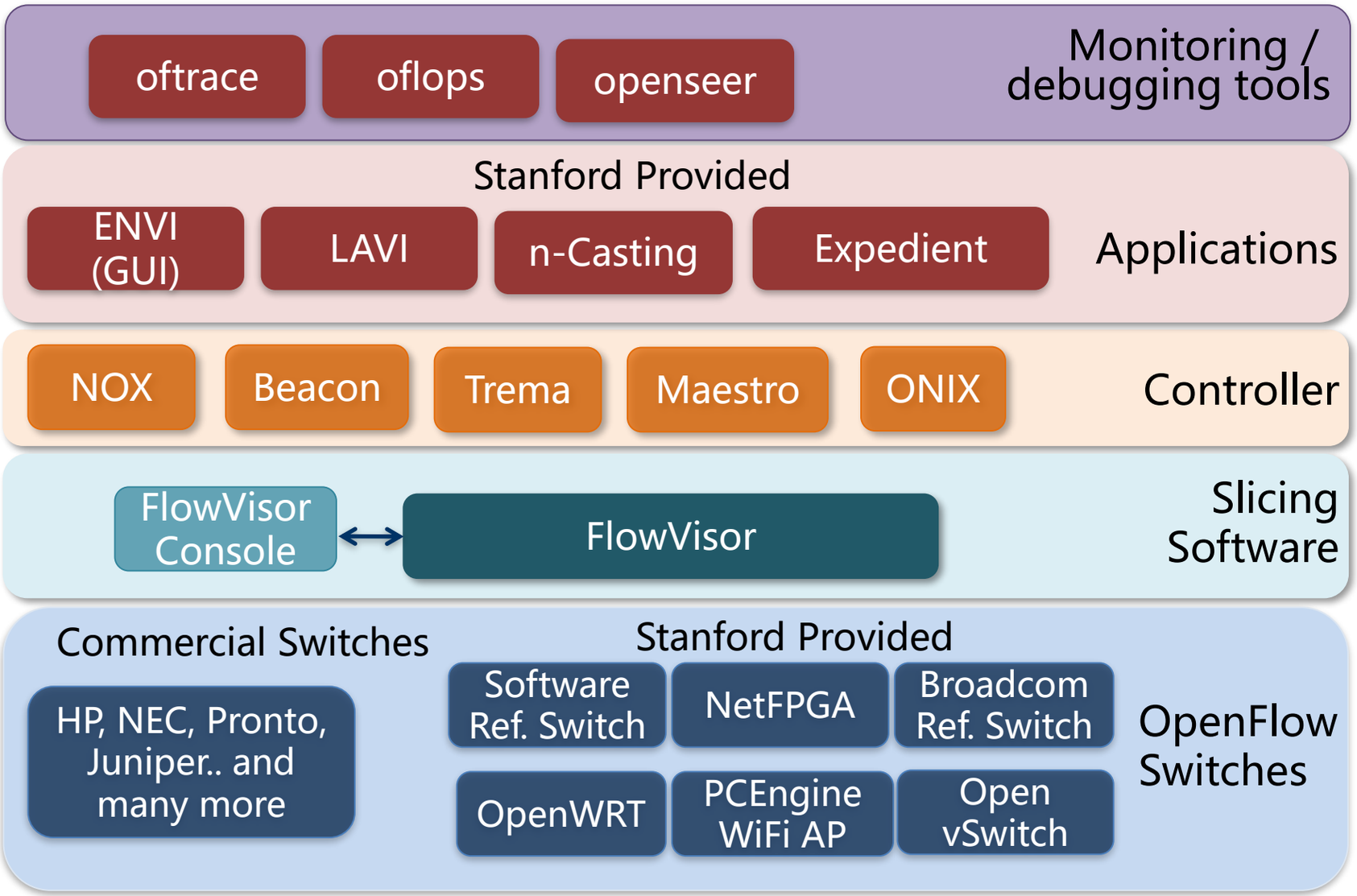
https://www.researchgate.net/figure/Comparison-among-SDN-controllers_fig5_281979574

Openflow Controllers

Use-Cases \ Controllers	Trema	Nox/Pox	RYU	Floodlight	ODL	ONOS***
Network Virtualizaiton by Virtual Overlays	YES	YES	YES	PARTIAL	YES	NO
Hop-by-hop Network Virtualization	NO	NO	NO	YES	YES	YES
OpenStack Neutron Support	NO	NO	YES	YES	YES	NO
Legacy Network Interoperability	NO	NO	NO	NO	YES	PARTIAL
Service Insertion and Chaining	NO	NO	PARTIAL	NO	YES	PARTIAL
Network Monitoring	PARTIAL	PARTIAL	YES	YES	YES	YES
Policy Enforcement	NO	NO	NO	PARTIAL	YES	PARTIAL
Load Balancing	NO	NO	NO	NO	YES	NO
Traffic Engineering	PARTIAL	PARTIAL	PARTIAL	PARTIAL	YES	PARTIAL
Dynamic Network Taps	NO	NO	YES	YES	YES	NO
Multi-Layer Network Optimization	NO	NO	NO	NO	PARTIAL	PARTIAL
Transport Networks - NV, Traffic-Rerouting, Interconnecting DCs, etc.	NO	NO	PARTIAL	NO	PARTIAL	PARTIAL
Campus Networks	PARTIAL	PARTIAL	PARTIAL	PARTIAL	PARTIAL	NO
Routing	YES	NO	YES	YES	YES	YES

https://www.researchgate.net/figure/Comparison-among-SDN-controllers_fig5_281979574

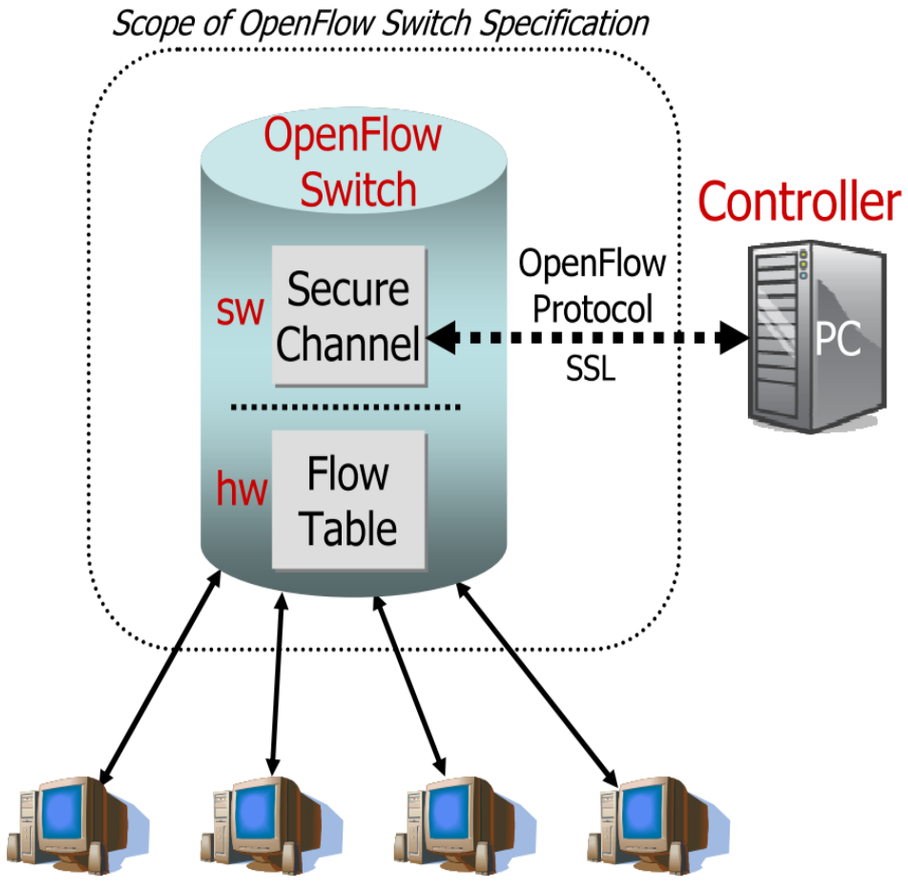
Openflow building blocks



https://www.slideshare.net/openflow/openflow-tutorial

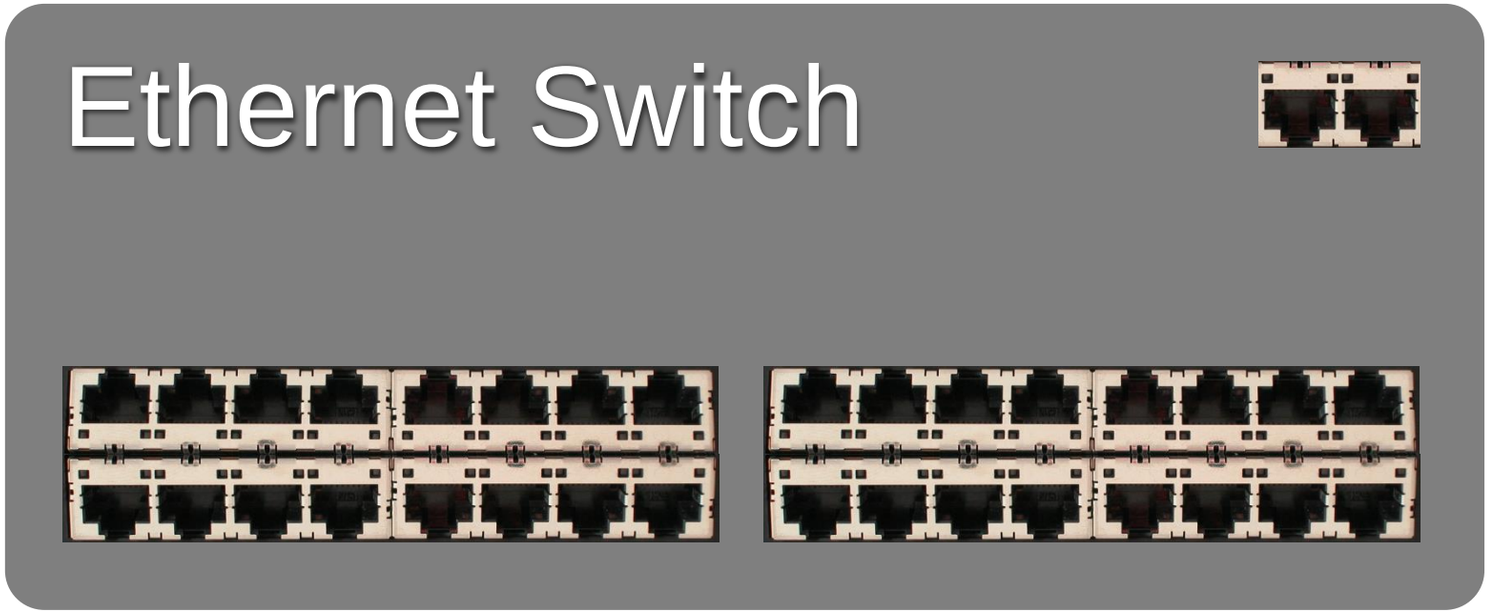
Components of OpenFlow Network

- Controller
  - OpenFlow protocol messages
  - Controlled channel
  - Processing
    - Pipeline Processing
    - Packet Matching
    - Instructions & Action Set
- OpenFlow switch
  - Secure Channel (SC)
  - Flow Table
    - Flow entry



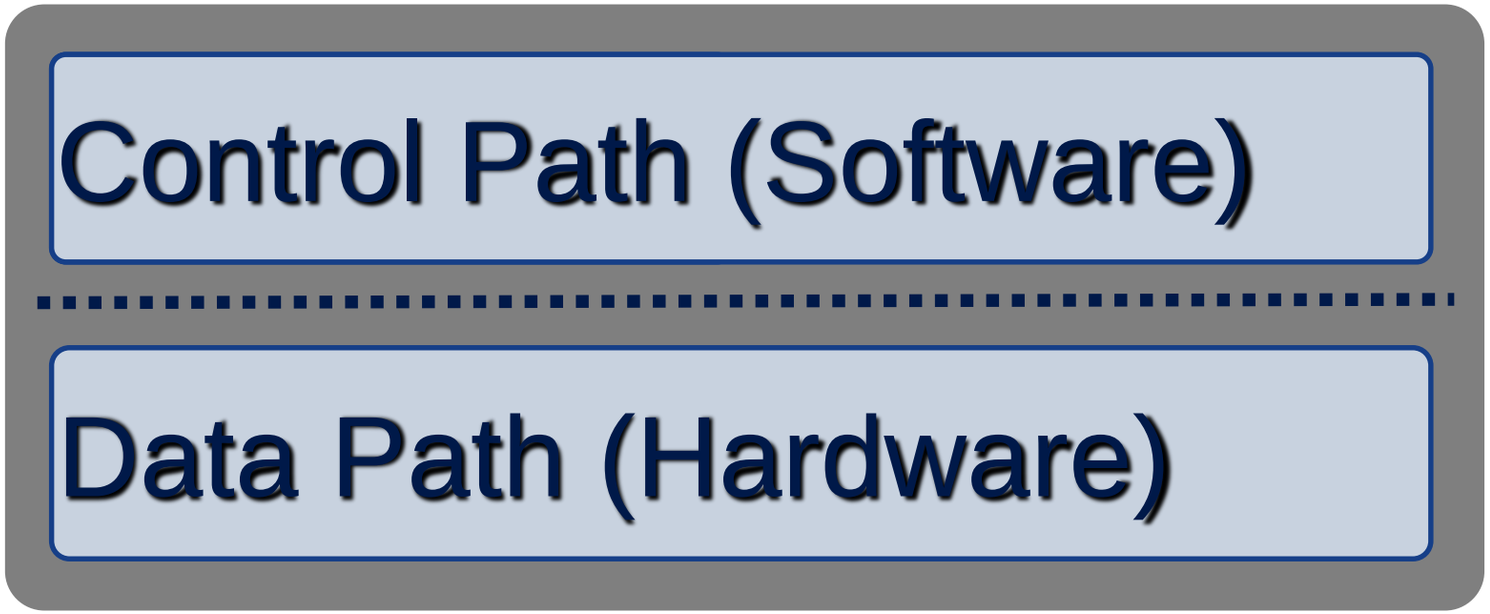
資料來源：國立清華大學資工系，鍾葉青教授—虛擬化技術:Network Virtualization : Software Defined Network

How does OpenFlow work?



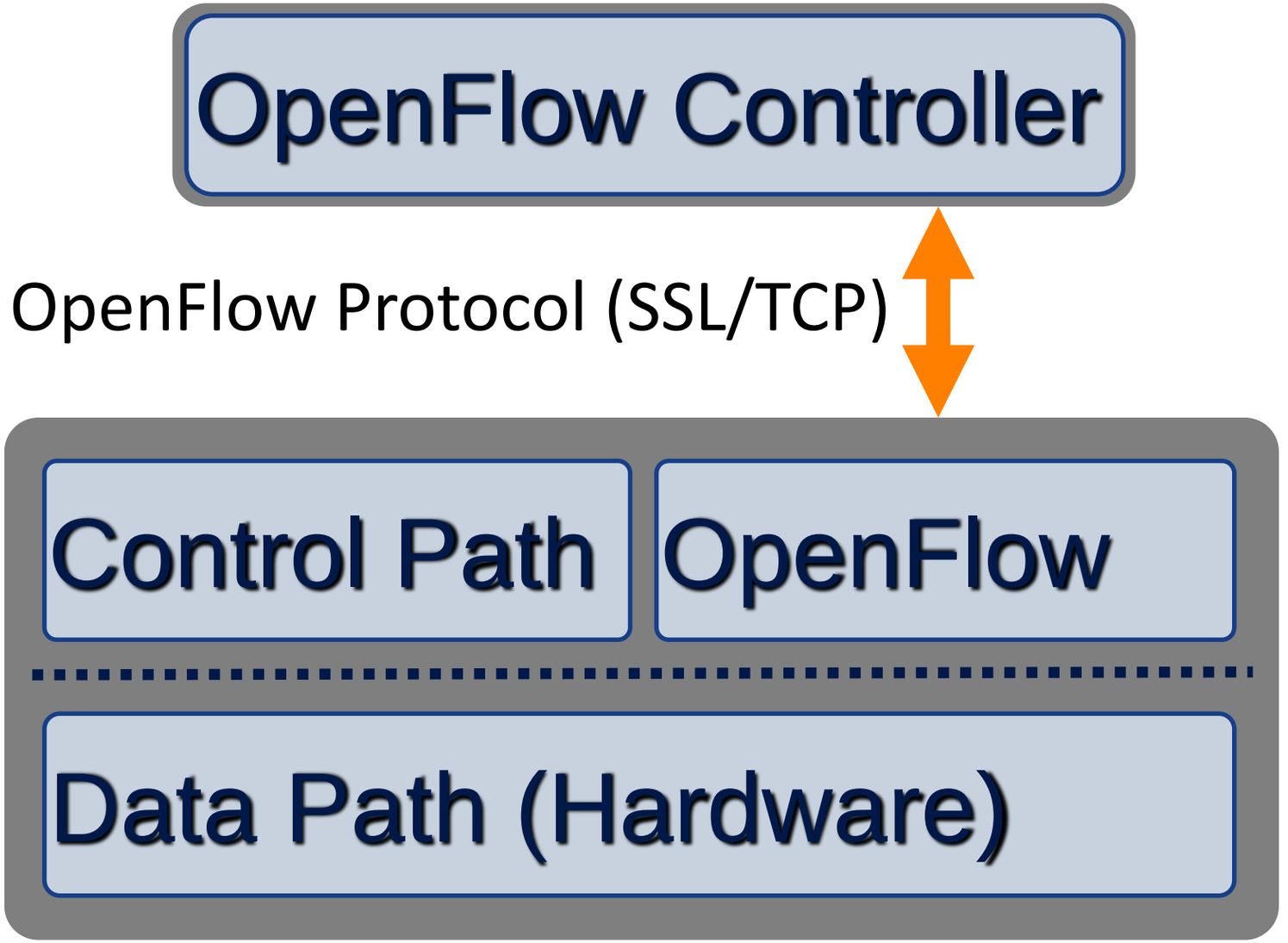
資料來源： OpenFlow/SDN tutorial OFC/NFOEC March,2012

How does OpenFlow work?



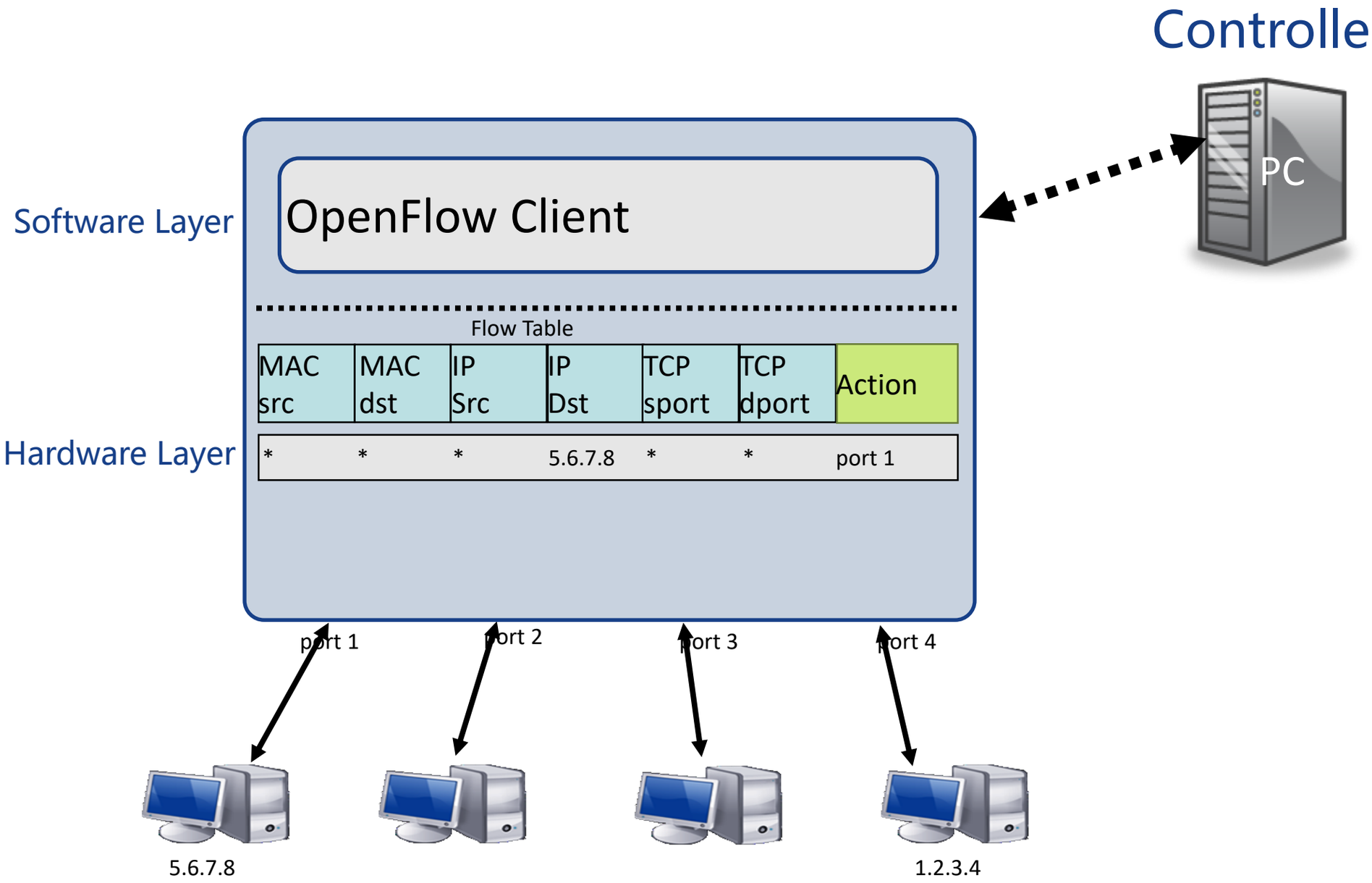
資料來源： OpenFlow/SDN tutorial OFC/NFOEC March,2012

How does OpenFlow work?



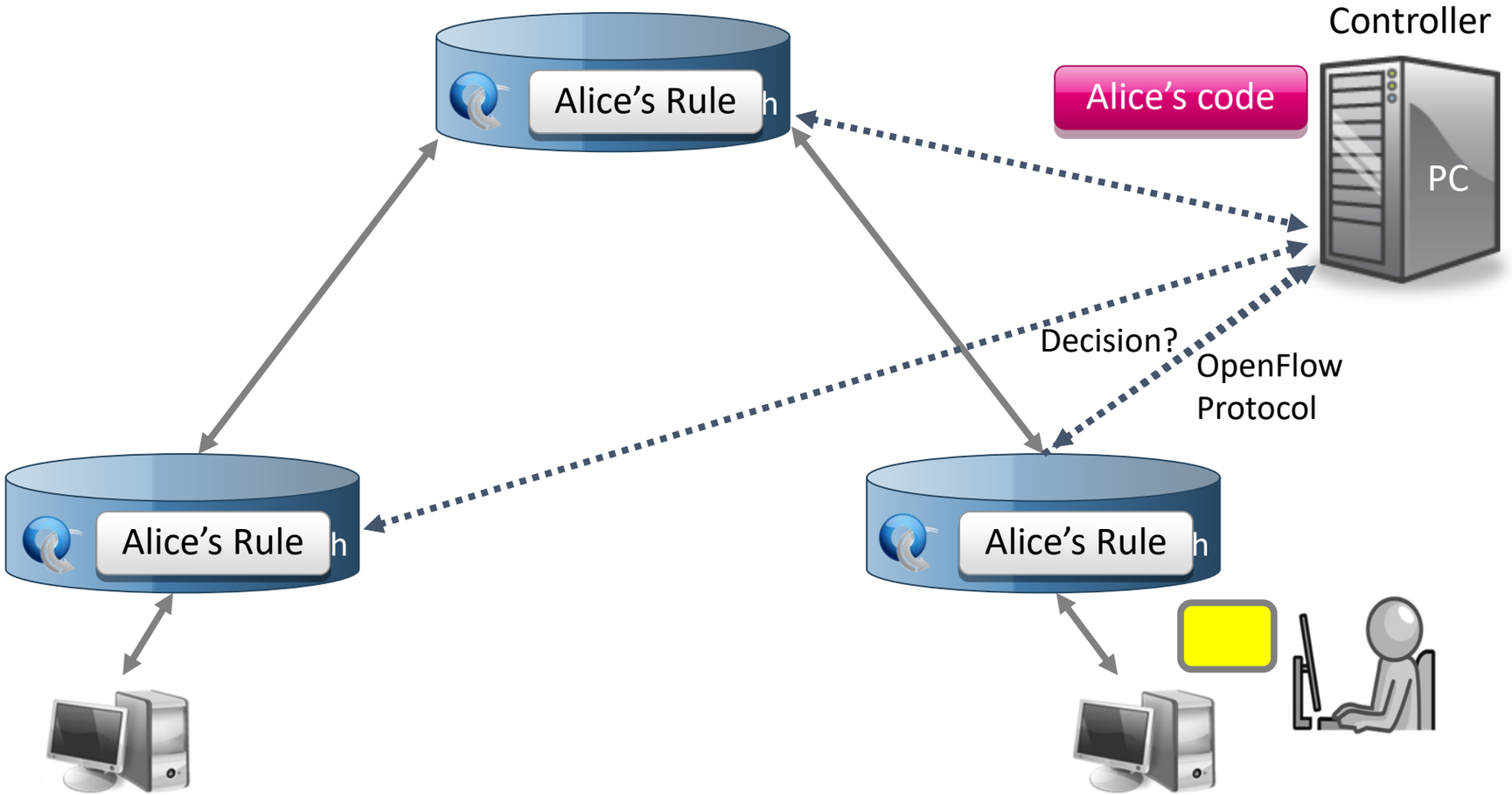
資料來源： OpenFlow/SDN tutorial OFC/NFOEC March,2012

OpenFlow Example



資料來源： OpenFlow/SDN tutorial OFC/NFOEC March,2012

OpenFlow usage



OpenFlow offloads control intelligence to a remote software

資料來源： OpenFlow/SDN tutorial OFC/NFOEC March,2012

OpenFlow usage(cont.)

Alice’ s code:

- Simple learning switch
- Per Flow switching
- Network access control/firewall
- StaAc “VLANs”
- Her own new rouAng protocol: unicast, mulAcast, mulApath
- Home network manager
- Packet processor (in controller)
- IPvAlice

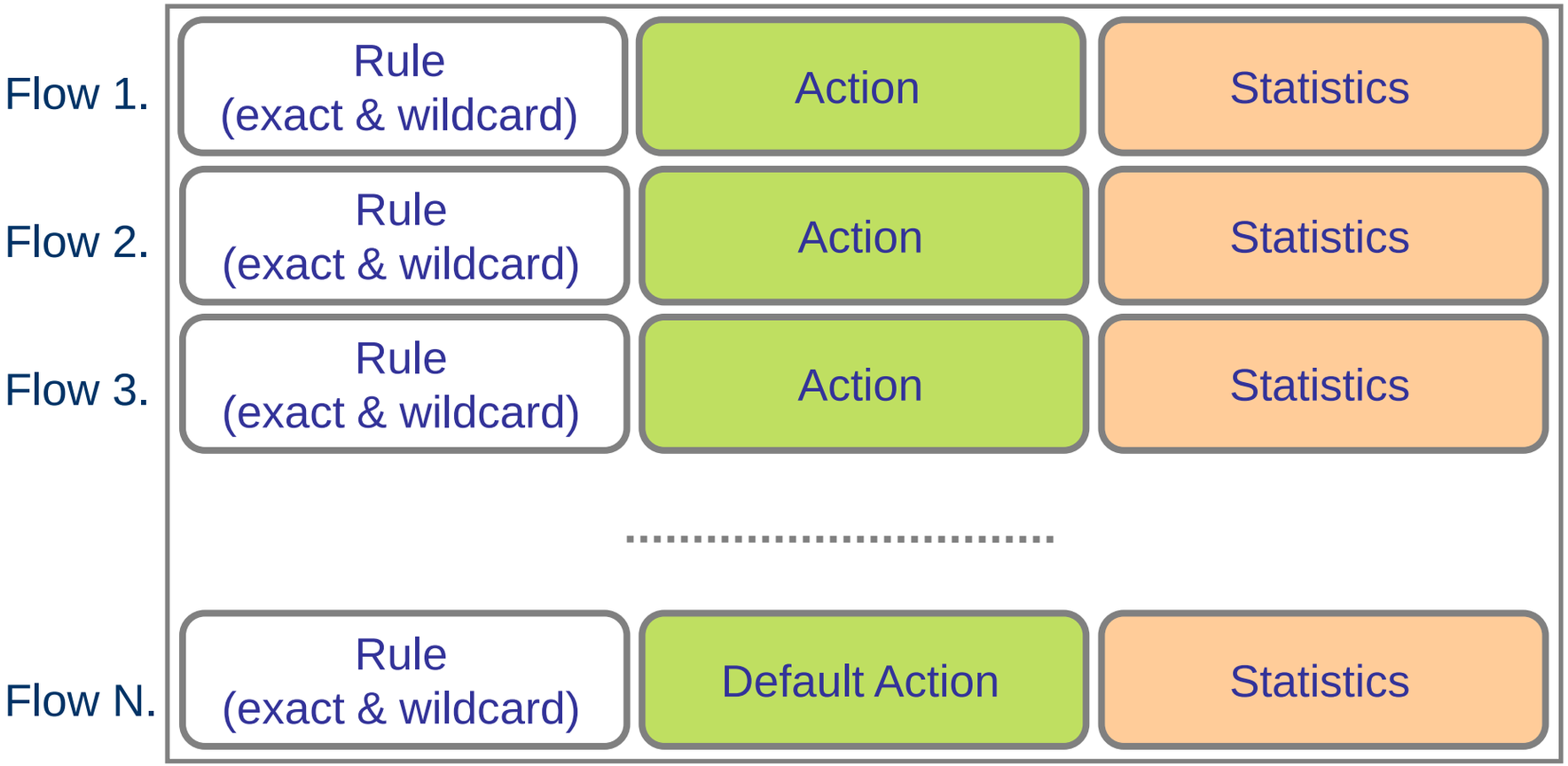
- VM migration
- Server Load balancing
- Mobility manager
- Power management
- Network monitoring and visualizaAon
- Network debugging
- Network slicing

… and much more you can create!

資料來源： OpenFlow/SDN tutorial OFC/NFOEC March,2012

Flow Table

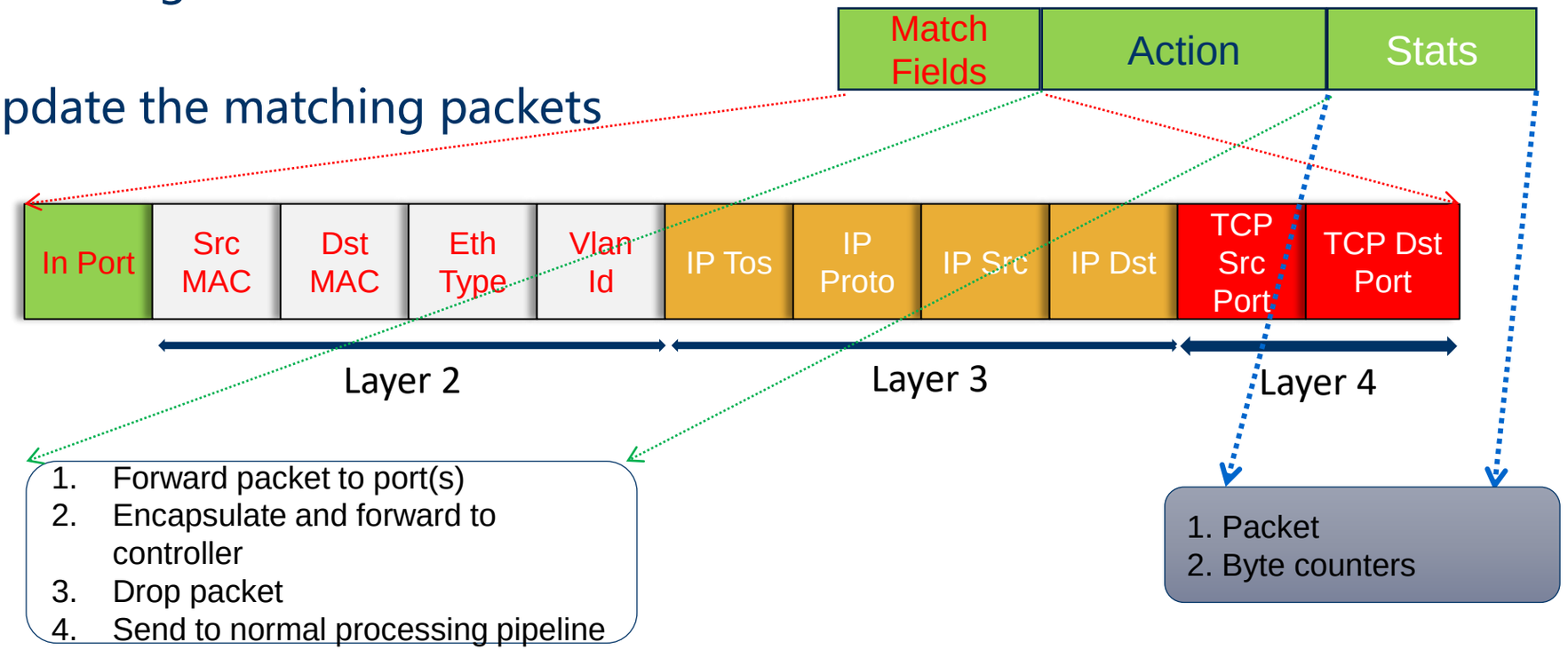
- Flow table in switches, routers, and chipsets



資料來源：國立清華大學資工系，鍾葉青教授—虛擬化技術:Network Virtualization Software Defined Network

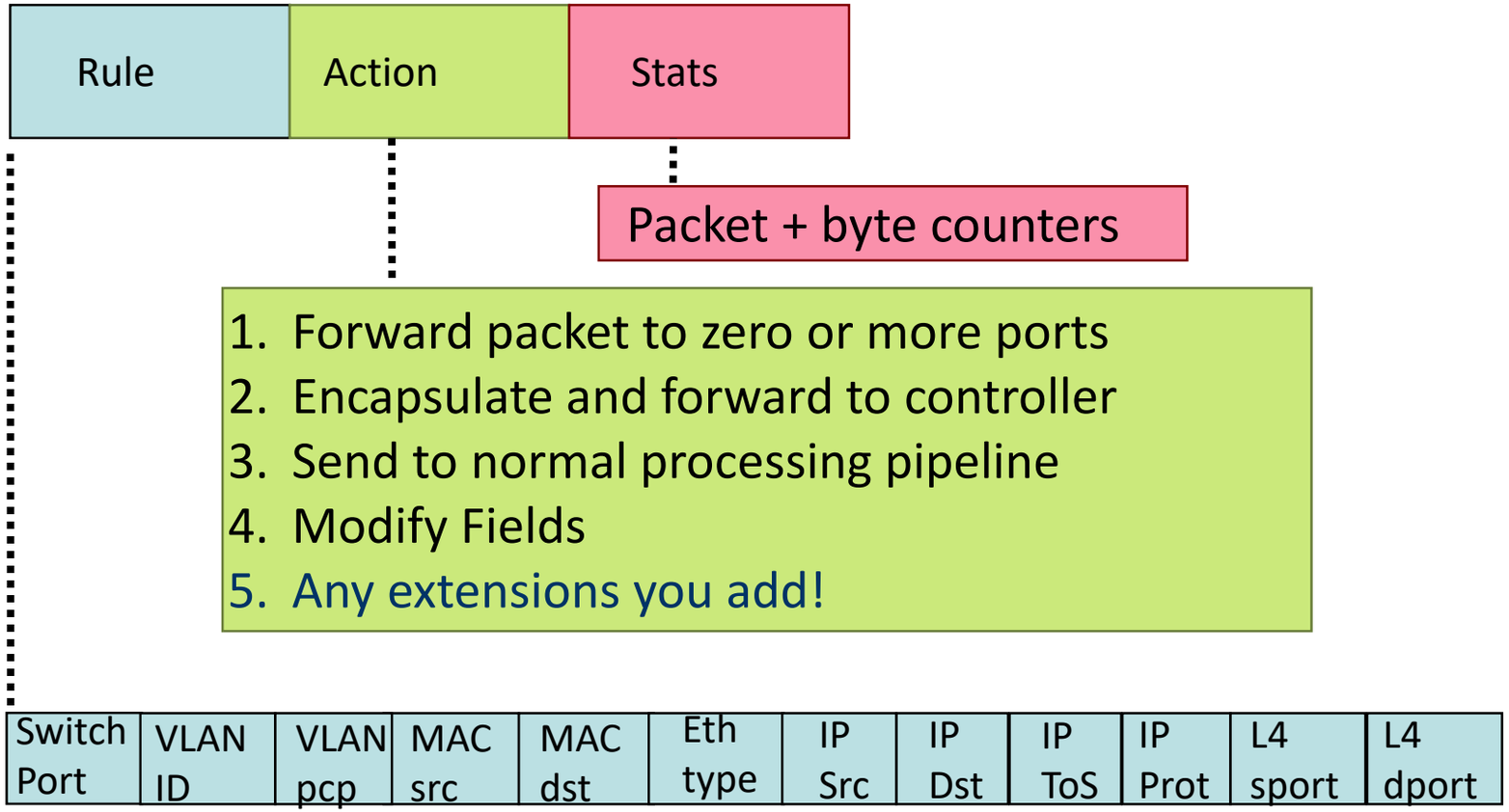
Flow Table

- A flow entry consists of
  - Match fields
    - Match against packets
  - Action
    - Modify the action set or pipeline processing
  - Stats
    - Update the matching packets



資料來源：國立清華大學資工系，鍾葉青教授—虛擬化技術:Network Virtualization Software Defined Network

Flow Table Entries



+ mask what fields to match

資料來源： OpenFlow/SDN tutorial OFC/NFOEC March,2012

Examples

Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f:..	*	*	*	*	*	*	*	port6

Flow Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
port3	00:20..	00:1f..	0800	vlan1	1.2.3.4	5.6.7.8	4	17264	80	port6

Firewall

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	*	*	*	22	drop

資料來源： OpenFlow/SDN tutorial OFC/NFOEC March,2012

Examples

Routing

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	5.6.7.8	*	*	*	port6

VLAN Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f..	*	vlan1	*	*	*	*	*	port6, port7, port9

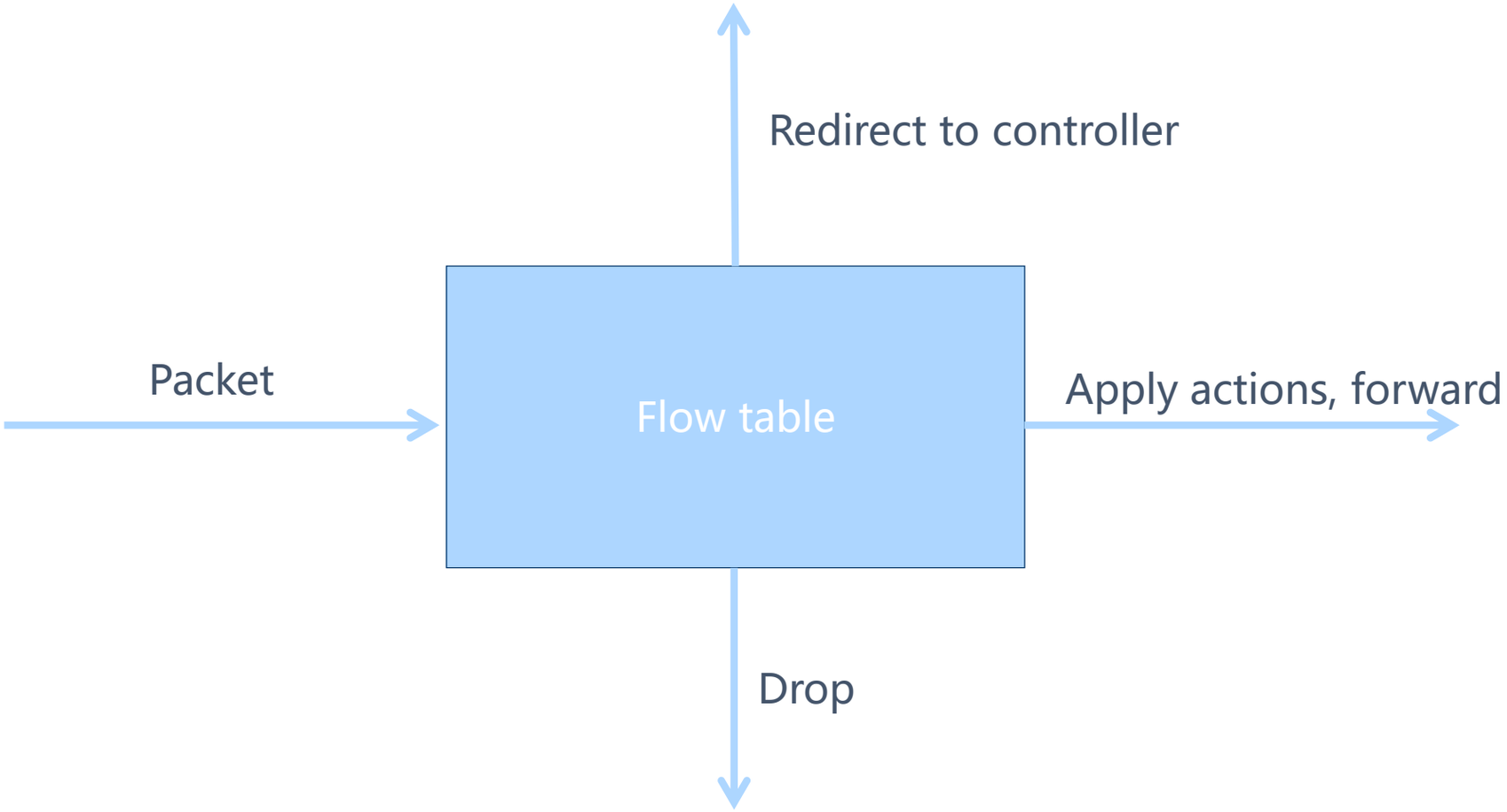
資料來源： OpenFlow/SDN tutorial OFC/NFOEC March,2012

05 OpenFlow(v1.0-1.3)

OpenFlow(v1.0-1.3)

- OpenFlow 1.0
- OpenFlow 1.1
- OpenFlow 1.2
- OpenFlow 1.3
- OpenFlow 1.4 and 1.5
- Version Comparison

OpenFlow recap



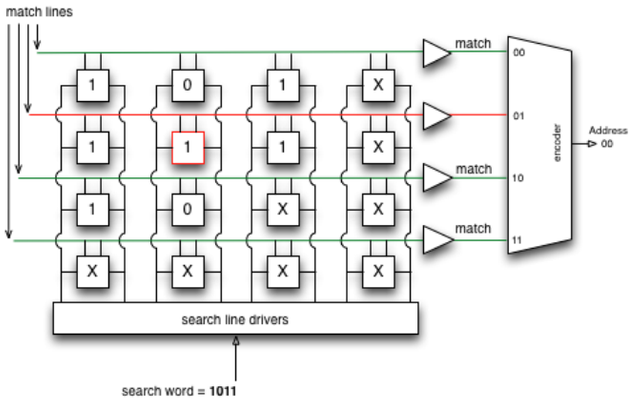
資料來源： IEEE CAMAD 2014 — From dumb to smarter switches in software defined networks : an overview of data plane evolution

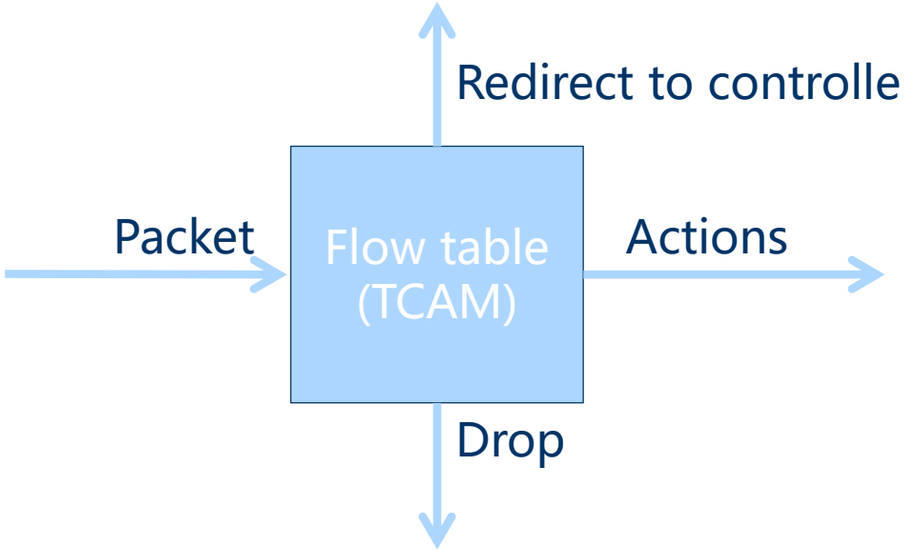
Models can be perfect and clean, reality is dirty!

- The match/action model can ideally be used to program any network behavior and to get rid of protocol limitations at any level
- But unfortunately, with OF:
  - Matches can be done only on a set of predefined header fields (Ethernet, IPv4, MPLS, VLAN tag, etc.)
  - Actions are limited to a rather small set
  - Header manipulation (like adding label/tags, rewriting of fields, etc.) is limited to standard schemes
- As a result, OF is not really protocol independent and standards (including OF standards) are still necessary

Where do OF limitations come from?

- OpenFlow has been designed having in mind current specialized HW architecture for switches
- Specialized HW is still fundamental in networking
  - General purpose HW (CPU) and soft-switches are still 2 order of magnitude slower
  - Architectures based network processors are also at least 1 order of magnitude slower
- The reference HW model for OF flow tables is TCAM (Ternary Content Addressable Memory)

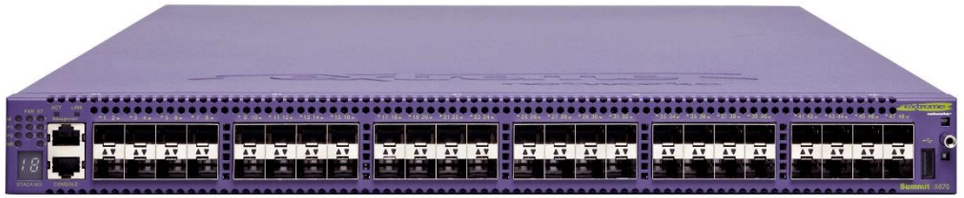




資料來源： IEEE CAMAD 2014 — From dumb to smarter switches in software defined networks : an overview of data plane evolution

Where do OF limitations come from?

- TCAMs however are typically expensive components that are used by manufacturers only when strictly necessary
- Less expensive memory components based on predefined search keys are often used for most of the common functions of a switch
- OF success depends on its “vendor neutral” approach where implementations issues are completely opaque (including reuse of standard modules for e.g. MAC and IP forwarding)
- Specialized ASICs are typically complex with a number of hard limitations on table types, sizes, and match depth



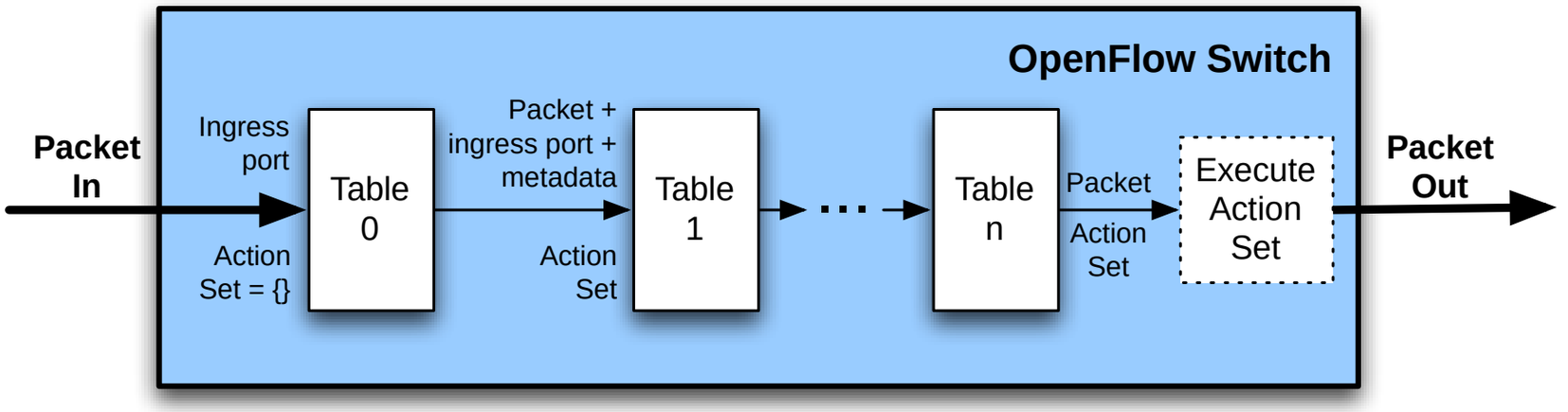
資料來源： IEEE CAMAD 2014 — From dumb to smarter switches in software defined networks : an overview of data plane evolution

Switches cannot remain dumb: Starting the process of data plane evolution

- One man alone can be pretty **dumb** sometimes, but for real bona fide stupidity, there ain't nothin' can beat **teamwork**. [Edward Abbey]

Evolution of the AL in OpenFlow:OF 1.1

- Single tables are costly : all possible combinations of header values in a single long table
- Solution: **Multiple Match Tables (MMT)**
- New actions of **MMT**:
  - **Add metadata**: parameters added and passed to next table
  - **Goto table**: possibility to go to specific tables for further processing



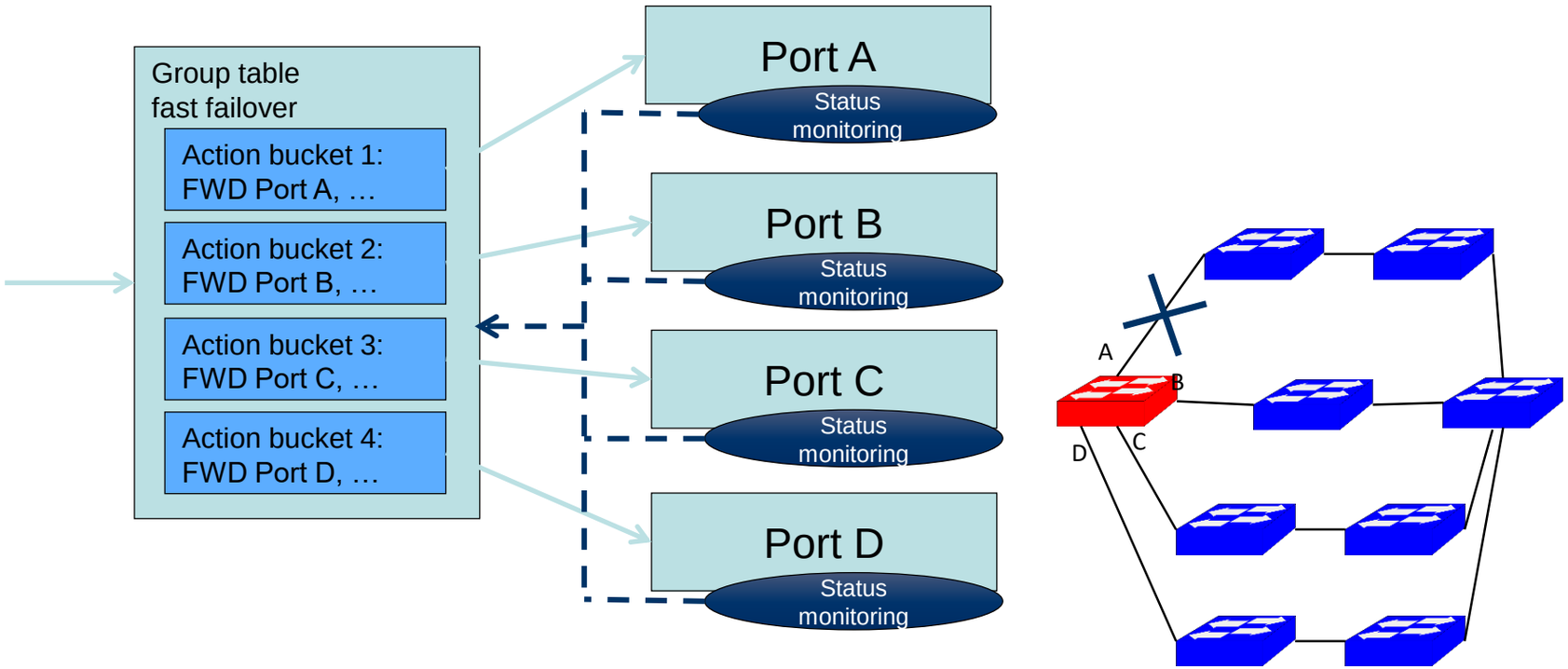
資料來源： IEEE CAMAD 2014 — From dumb to smarter switches in software defined networks : an overview of data plane evolution

Evolution of the AL in OpenFlow:OF 1.1

- Packets of the same flow are applied the same actions unless the table entry is modified by the controller
- Not good for some common and important cases (e.g. multicast, multipath load balancing, failure reaction, etc.)
- Solution: **Group tables**
  - Goto table “group table n”
  - List of buckets of actions
  - All or some of the buckets are executed depending on the following types of Group tables :
    - All (multicast)
    - Select (multipath)
    - Fast-failover (protection switching)

Evolution of the AL in OpenFlow:OF 1.1

- **Fast failover**
- Note that this is the first “stateful” behavior in the data plane introduced in OF !!!



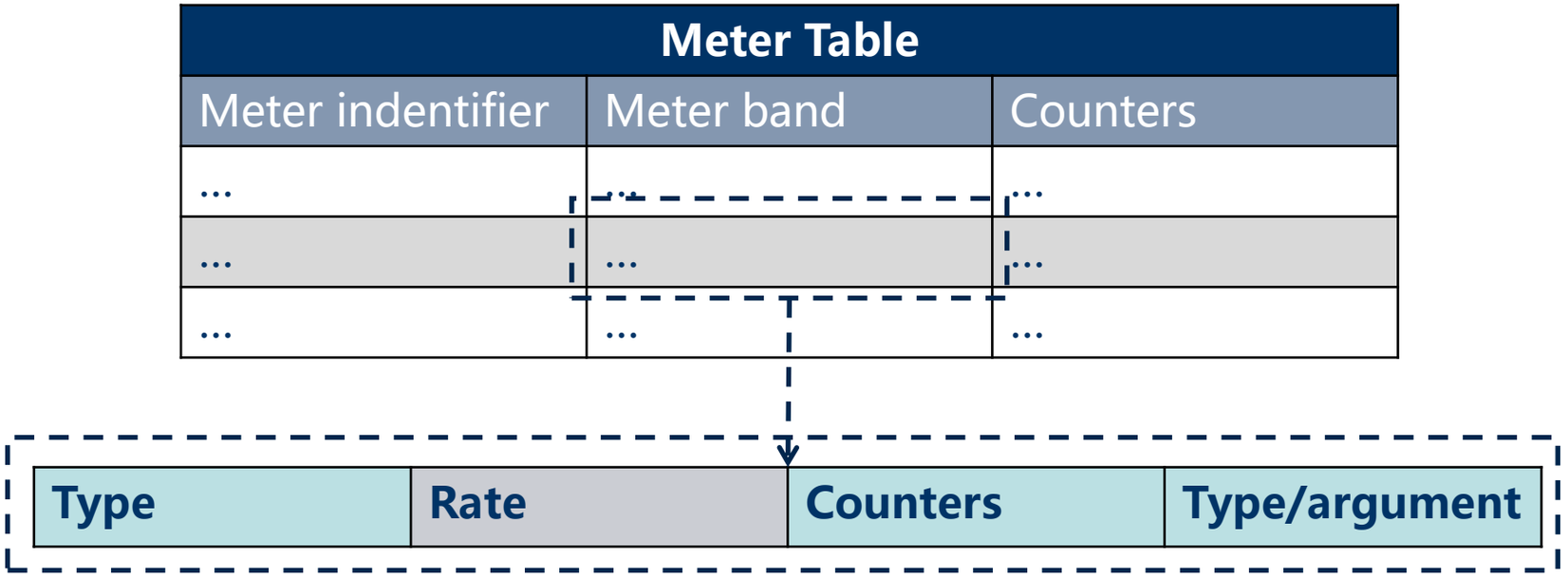
資料來源： IEEE CAMAD 2014 — From dumb to smarter switches in software defined networks : an overview of data plane evolution

Evolution of the AL in OpenFlow:OF 1.2

- Support for IPv6, new match fields:
  - source address, destination address, protocol number, traffic class, ICMPv6 type, ICMPv6 code, IPv6 neighbor discovery header fields, and IPv6 flow labels
- Extensible match (Type Length Value)
- Experimenter extensions
- Full VLAN and MPLS support
- **Multiple controllers**

Evolution of the AL in OpenFlow:OF 1.3

- Initial traffic shaping and QoS support
  - **Meters:** tables (accessed as usual with “goto table”) for collecting statistics on traffic flows and applying rate-limiters



資料來源： IEEE CAMAD 2014 — From dumb to smarter switches in software defined networks : an overview of data plane evolution

Evolution of the AL in OpenFlow:OF v1.4 and v1.5

- **OpenFlow v1.4**
  - More extensible wire protocol
  - **Synchronized tables**
    - tables with synchronized flow entries
  - **Bundles**
    - similar to transactional updates in DB
  - Support for optical ports

- **OpenFlow v1.5**
  - Packet type identification process (Ethernet packet,PPP packet) egress table.

Comparison with OpenFlow Version

Protocol Version	Main Function
OpenFlow 1.0	Single-table , IPv4
OpenFlow 1.1	Multi-table , Group-table , MPLS , VLAN
OpenFlow 1.2	Multi-Controller , IPv6
OpenFlow 1.3 (mainstream , long-term support , stable)	Meter Table , Version negotiation capability
OpenFlow 1.4	Synchronized tables , Protocol message perfection
OpenFlow 1.5	Packet type identification process (Ethernet packet,PPP packet) egress table.

資料來源： https://blog.csdn.net/qq_29229567/article/details/88797395

06 SDN Controller

SDN Controller

- Background
- SDN Controllers
  - NOX
  - POX
  - Ryu
  - Floodlight
  - Opendaylight
  - Onos
- Summary

Background

- Networks have so far been managed and configured using lower level, device-specific instruction sets and mostly closed proprietary NOSs (e.g., Cisco IOS and Juniper JunOS).

- SDN is promised to facilitate network management and ease the burden of solving networking problems by means of the logically centralized control offered by a NOS.

- With NOSs, to define network policies a developer no longer needs to care about the low-level details of data distribution among routing elements.

How many flows exist in real network/data centers

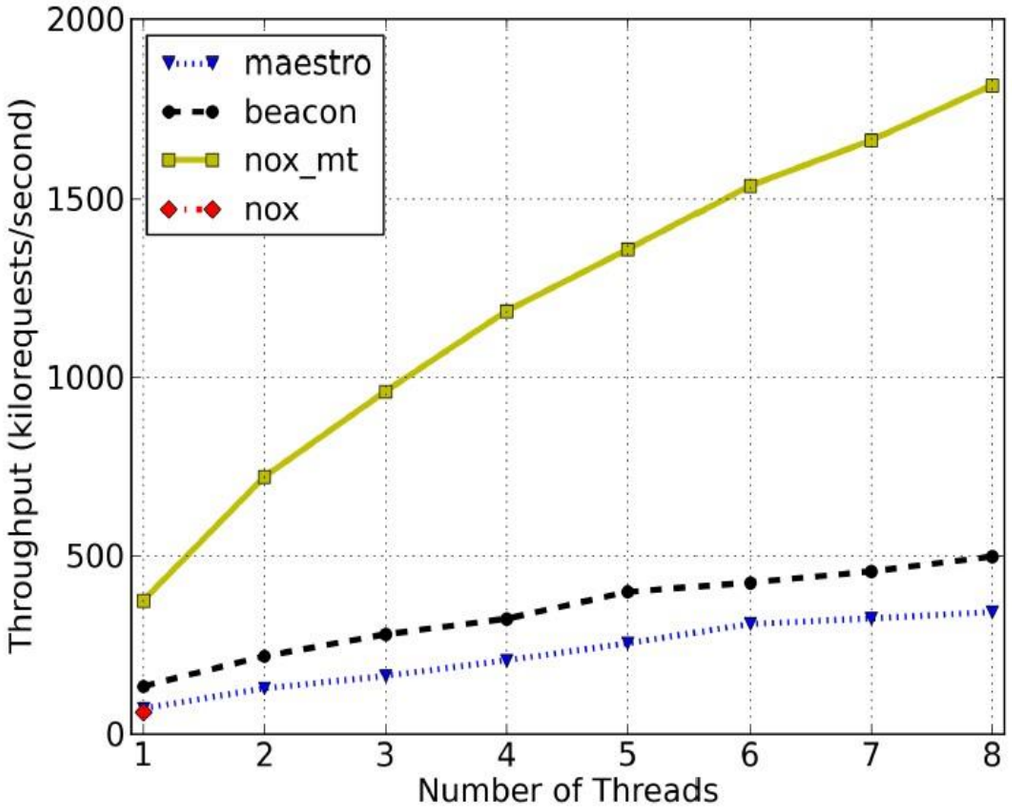
- NOX handles around 30k flow initiation events per second while maintaining a sub-10ms flow install time.
- Kandula et al. found that a 1500-server cluster has a median flow arrival rate of 100k flows per second.
- Benson et al. show that a network with 100 switches can have spikes of 10M flows arrivals per second in the worst case.

資料來源：On Controller Performance in Software-Defined Networks,2012

Centralized Controllers

- A centralized controller is a single entity that manages all forwarding devices of the network.
- Naturally, it represents a single point of failure and may have scaling limitations.
- Centralized controllers are designed as highly concurrent systems (i.e., multithreaded design for multicore computer) to achieve required throughput.
- Beacon can deal with more than 12 million flows per second by using Amazon cloud service.
- List of centralized controllers:
  NOX-MT, Maestro, Beacon, Floodlight, Trema, Ryu, Meridian, ProgrammableFlow, Rosemary

Effect of Multi-threading on Throughput



資料來源： A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood. On controller performance in software-defined networks. In USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE), 2012.

Distributed Controllers

- A distributed NOS can be scaled up to meet the requirements of potentially any environment.
- Most distributed controllers offer weak consistency semantics, which implies that there is a period of time in which distinct nodes may read different values.
- Another common property is fault tolerance. However, SDN resiliency as a whole is an open challenge.
- List of distributed controllers:
  Onix, HyperFlow, HP VAN SDN, ONOS, DISCO, yanc, PANE, SMaRt-Light, Fleet

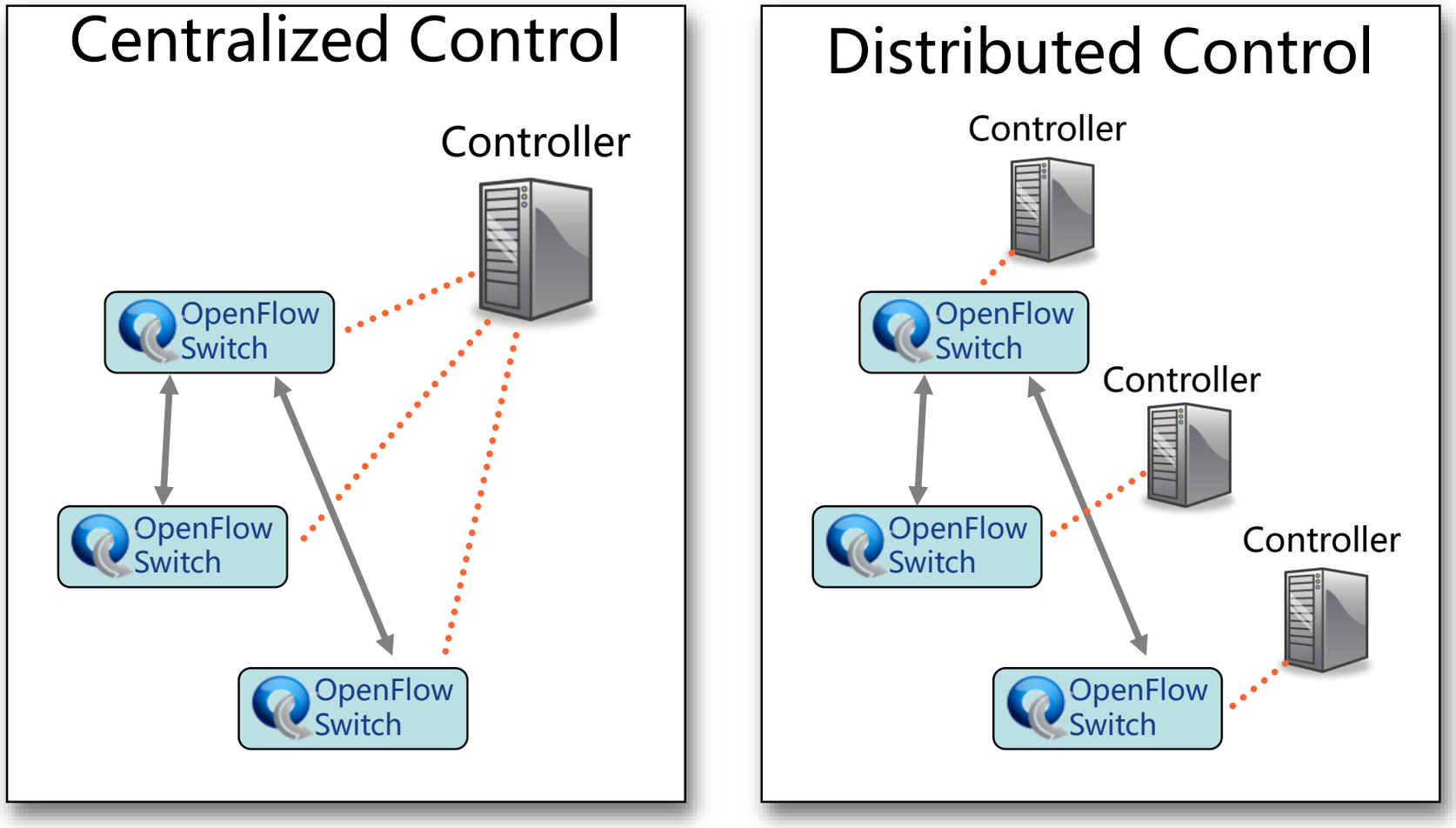
Architectural and Design Elements of SDN Controllers

Component	OpenDaylight	OpenContrail	HP VAN SDN	Onix	Beacon
Base network services	Topology/Stats/Switch Manager, Host Tracker, Shortest Path Forwarding	Routing, Tenant Isolation	Audit Log, Alerts, Topology, Discovery	Discovery, Multi-consistency Storage, Read State, Register for updates	Topology, device manager, and routing
East/Westbound APIs	—	Control Node (XMPP-like control channel)	Sync API	Distribution I/O module	*Not present*
Integration Plug-ins	OpenStack Neutron	CloudStack, OpenStack	OpenStack	—	—
Management Interfaces	GUI/CLI, REST API	GUI/CLI	REST API Shell / GUI Shell	—	Web
Northbound APIs	REST, REST-CONF [200], Java APIs	REST APIs (configu-ration, operational, and analytic)	REST API, GUI Shell	Onix API (general purpose)	API (based on OpenFlow events)
Service abstraction layers	Service Abstraction Layer (SAL)	—	Device Abstraction API	Network Information Base (NIB) Graph with Import/Export Functions	—
Southbound APIs or connectors	OpenFlow, OVSDB, SNMP, PCEP, BGP, NETCONF	—	OpenFlow, L3 Agent, L2 Agent	OpenFlow, OVSDB	OpenFlow

資料來源：蔡孟勳教授SDN/NDV教材—Controller

Centralized vs Distributed Control

Both models are possible with OpenFlow



資料來源： OpenFlow/SDN tutorial OFC/NFOEC March,2012

Flow Routing vs Aggregation

Both models are possible with OpenFlow

Flow-Based

- Every flow is individually set up by controller
- Exact-match flow entries
- Flow table contains one entry per flow
- Good for fine grain control, e.g. campus networks

Aggregated

- One flow entry covers large groups of flows
- Wildcard flow entries
- Flow table contains one entry per category of flows
- Good for large number of flows, e.g. backbone

資料來源： OpenFlow/SDN tutorial OFC/NFOEC March,2012

Reactive vs. Proactive(pre-populated)

Both models are possible with OpenFlow

Reactive

- First packet of flow triggers controller to insert flow entries
- Efficient use of flow table
- Every flow incurs small additional flow setup time
- If control connection lost, switch has limited utility

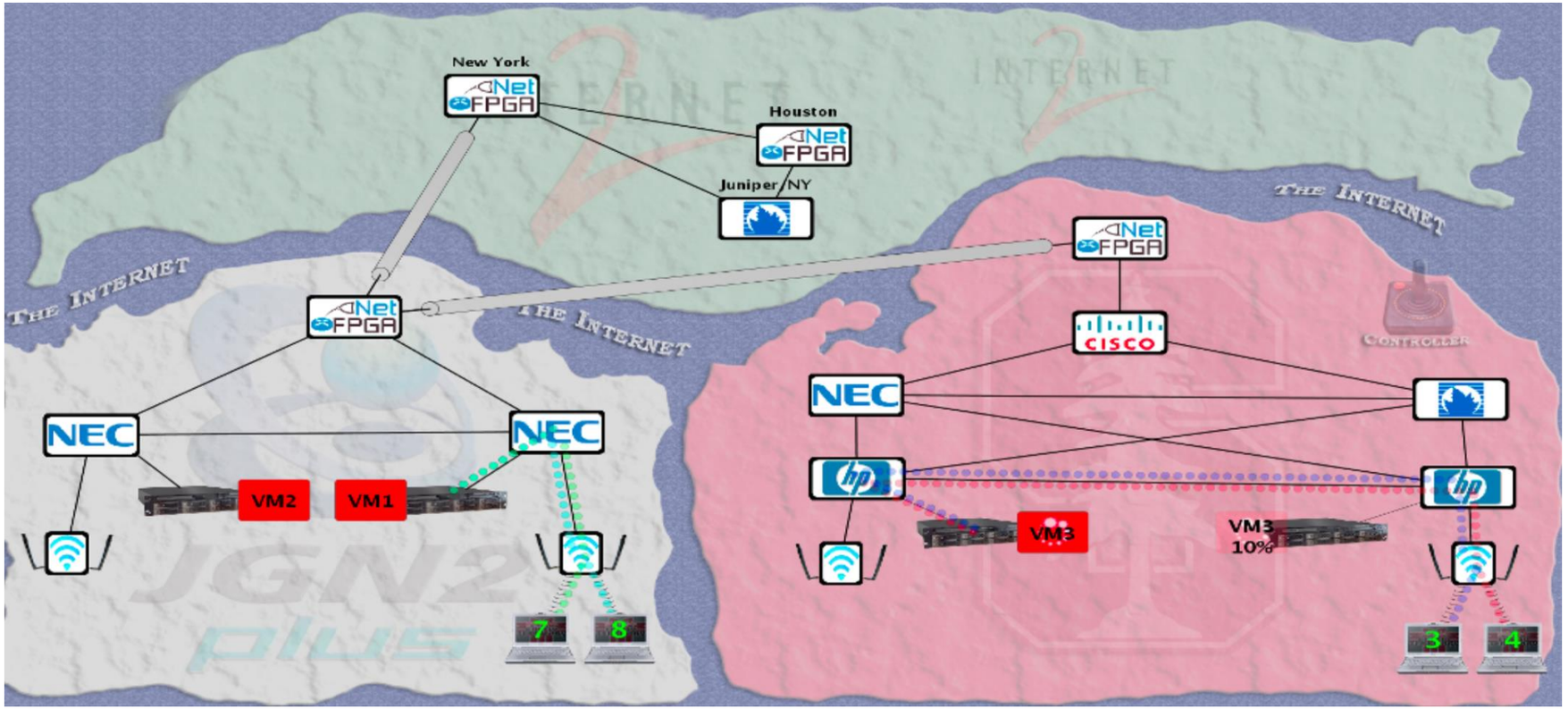
Proactive

- Controller pre-populates flow table in switch
- Zero additional flow setup time
- Loss of control connection does not disrupt traffic
- Essentially requires aggregated (wildcard) rules

資料來源： OpenFlow/SDN tutorial OFC/NFOEC March,2012

Intercontinental VM migration

- Moved a VM from Stanford to Japan without changing its IP.
- VM hosted a video game server with active network connections.



資料來源：蔡孟勳教授SDN/NDV教材— How SDN Works

Many Different SDN Controllers

- NOX/POX
- Ryu
- Floodlight
- OpenDaylight

Controller	routing method	Language	Multithread	framework
Ryu	SPF	Python	YES	-
NOX	OSPF	C/python	YES	Boost library
POX	OSPF	Python	NO	-
Floodlight	SPF	Java	YES	Netty
OpenDayLight	SPF	Java	YES	OSGi
ONOS	SPF	Java	YES	Intent

NOX:Overview

- First-generation OpenFlow controller
  - Open source, stable, widely used

http://www.noxrepo.org/

- Two “flavor” of NOX
  - NOX-Classic: C++/Python. No longer supported.
  - NOX (the “new NOX”)
    - ✓ C++ only
    - ✓ Fast, clean codebase
    - ✓ Well maintained and supported

NOX:Characteristics

- Users implement control in C++
- Supports OpenFlow v.1.0
  - A fork (CPqD) supports 1.1, 1.2, and 1.3
- Programming model
  - Controller registers for events
  - Programmer writes event handler

When to Use NOX

- You know C++
- You are willing to use low-level facilities and semantics of OpenFlow
- You need good performance

POX:Overview

- NOX in Python
  - Supports OpenFlow v. 1.0 only
- Advantages
  - Widely used, maintained, supported
  - Relatively easy to read and write code
- Disadvantages: Performance

When to Use POX

- You know Python
- You are are not concerned about controller performance
- Rapid prototyping and experimentation

Ryu

- Open source Python controller
  - Supports OpenFlow 1.0, 1.2, 1.3, 1.4, 1.5, Nicira extensions
  - Works with OpenStack
- Aims to be an “Operating System” for SDN
- Advantages
  - OpenStack integration
  - OpenFlow 1.2, 1.3, 1.4, 1.5
  - Good documentation
- Disadvantages: Performance

http://osrg.github.io/ryu/

Ryu means "flow" in Japanese. Ryu is pronounced "ree-yooh".

Floodlight

- Open-source Java controller
  - Supports OpenFlow v. 1.0 and v. 1.3
  - Fork from the Beacon Java OpenFlow controller
  - Maintained by Big Switch Networks

- Advantages
  - Good documentation
  - Integration with REST API
  - Production-level, OpenStack/Multi-Tenant Clouds

- Disadvantages: Steep learning curve

http://www.projecEloodlight.org/floodlight/

OpenDaylight:Overview

- Consortium
- Architecture
- Demonstration
  - Life of a packet, Web interface
  - Essential ODL functions
- More information
  - http://sdnhub.org/
  - http://www.slideshare.net/sdnhub/opendaylight-app-development-tutorial

OpenDaylight : Consortium

- Heavy industry involvement and backing

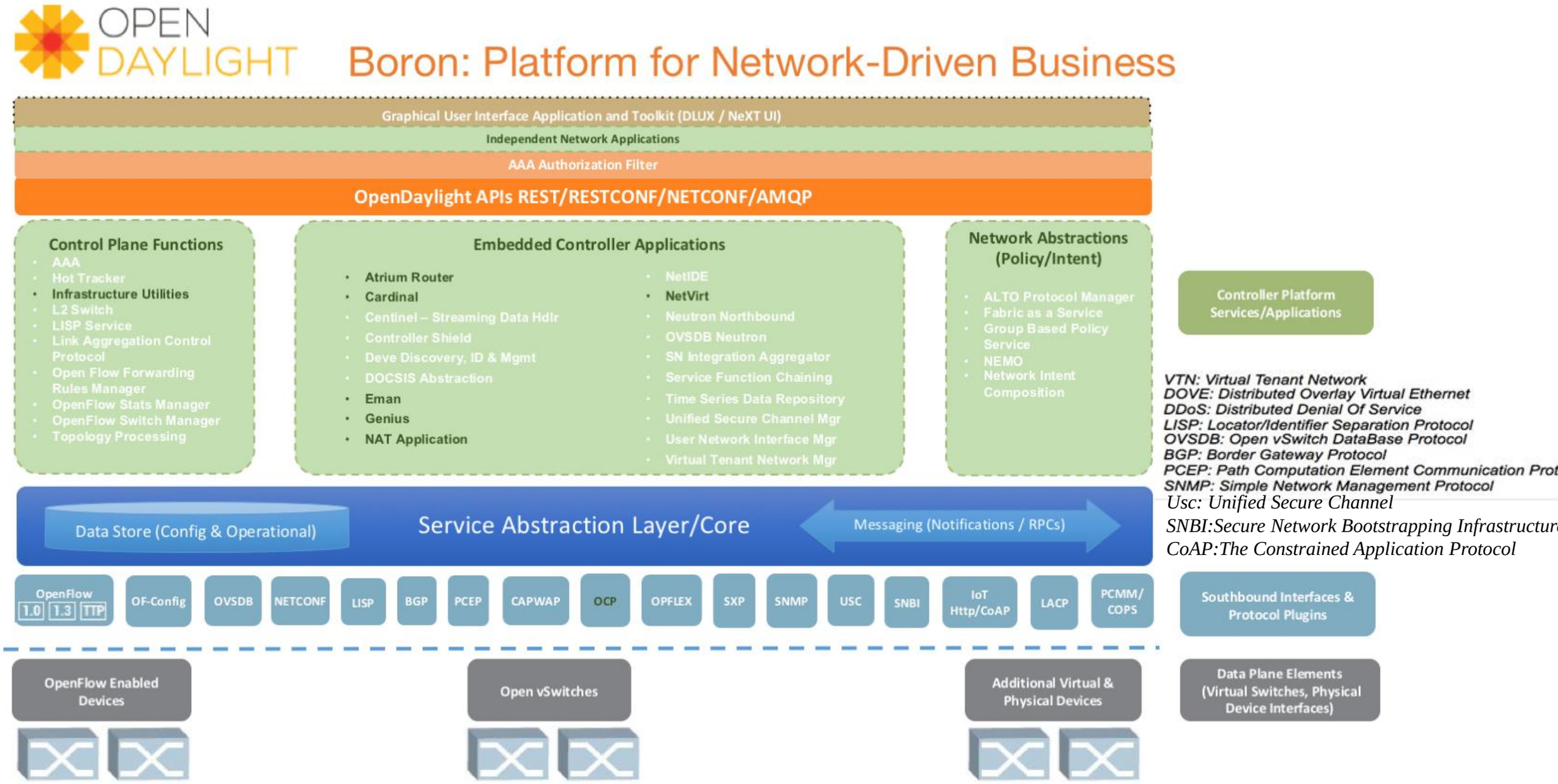
- Focused on having an open framework for building upon SDN/NFV innovations
- Not limited to OpenFlow innovations

OpenDaylight : Advantage and Disvantage

- Advantage :
  - OpenDaylight supports all southbound interfaces. Therefore, it will work with a huge range of network devices and existing deployments.
  - Resiliency and scalability
  - Modularity and extensibility

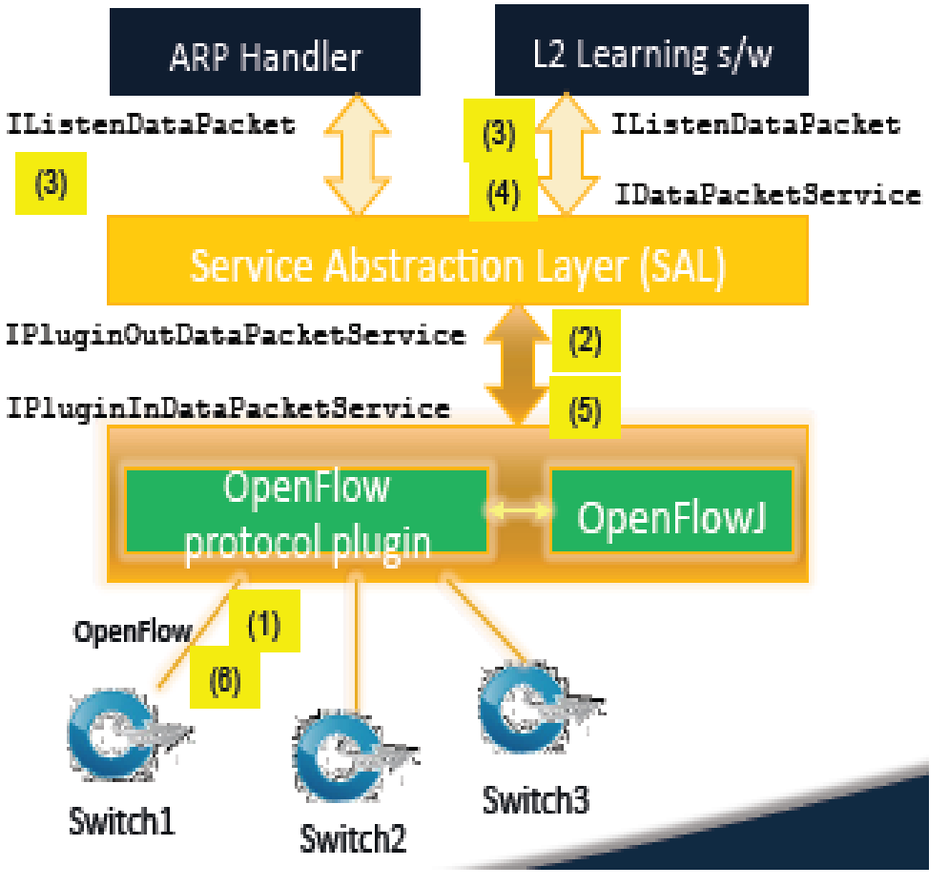
- Disadvantage :
  - Complex to install and maintain.

Boron Release



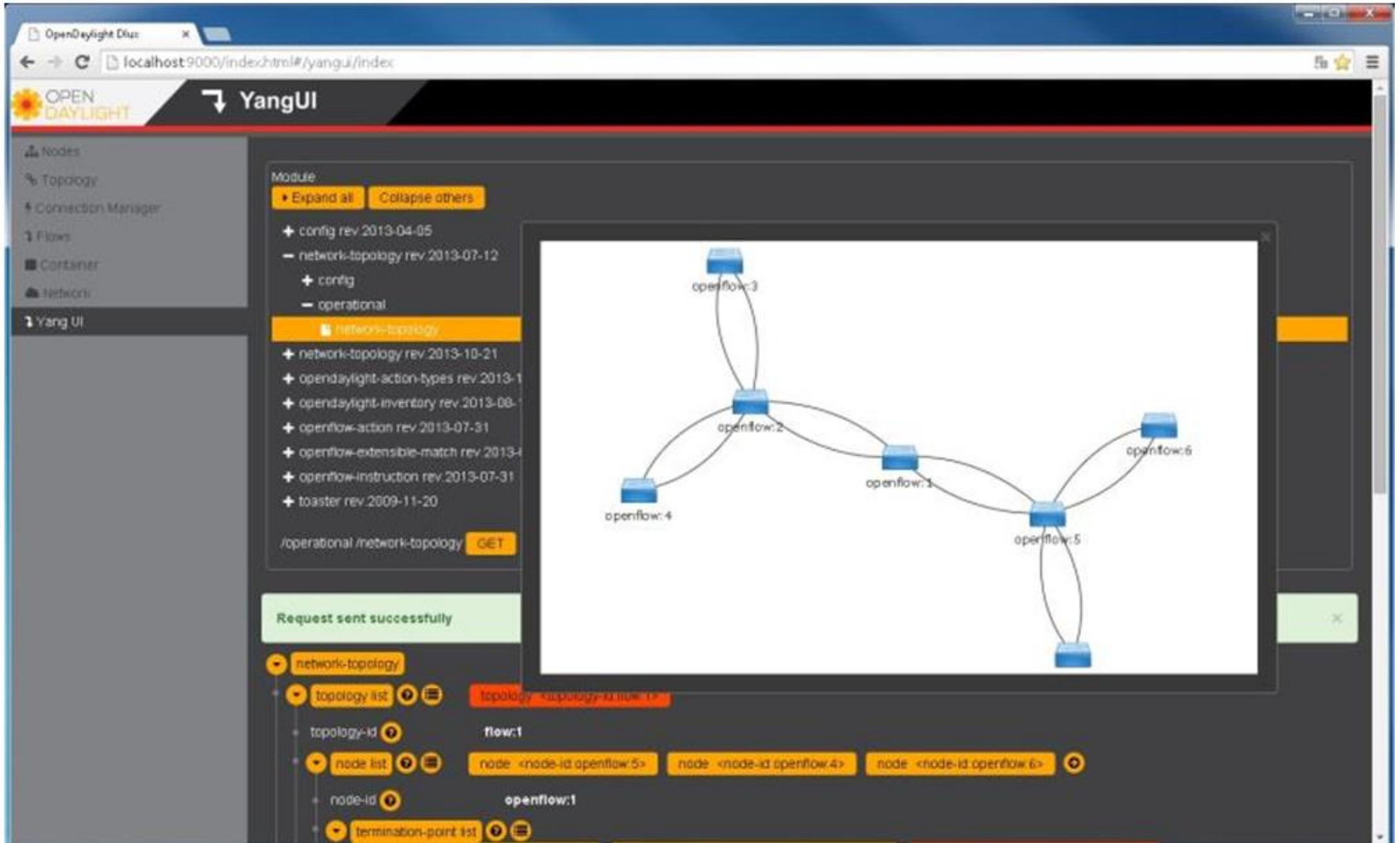
Life of a Packet

- A packet arriving at Switch1 will be sent to the appropriate plugin managing the switch
- The plugin will parse the packet, generate an event for SAL
- SAL will dispatch the packet to the modules listening for DataPacket
- Module handles packet and sends packet_out through IDataPacketService
- SAL dispatches the packet to the modules listening for DataPacket
- OpenFlow message sent to appropriate switch

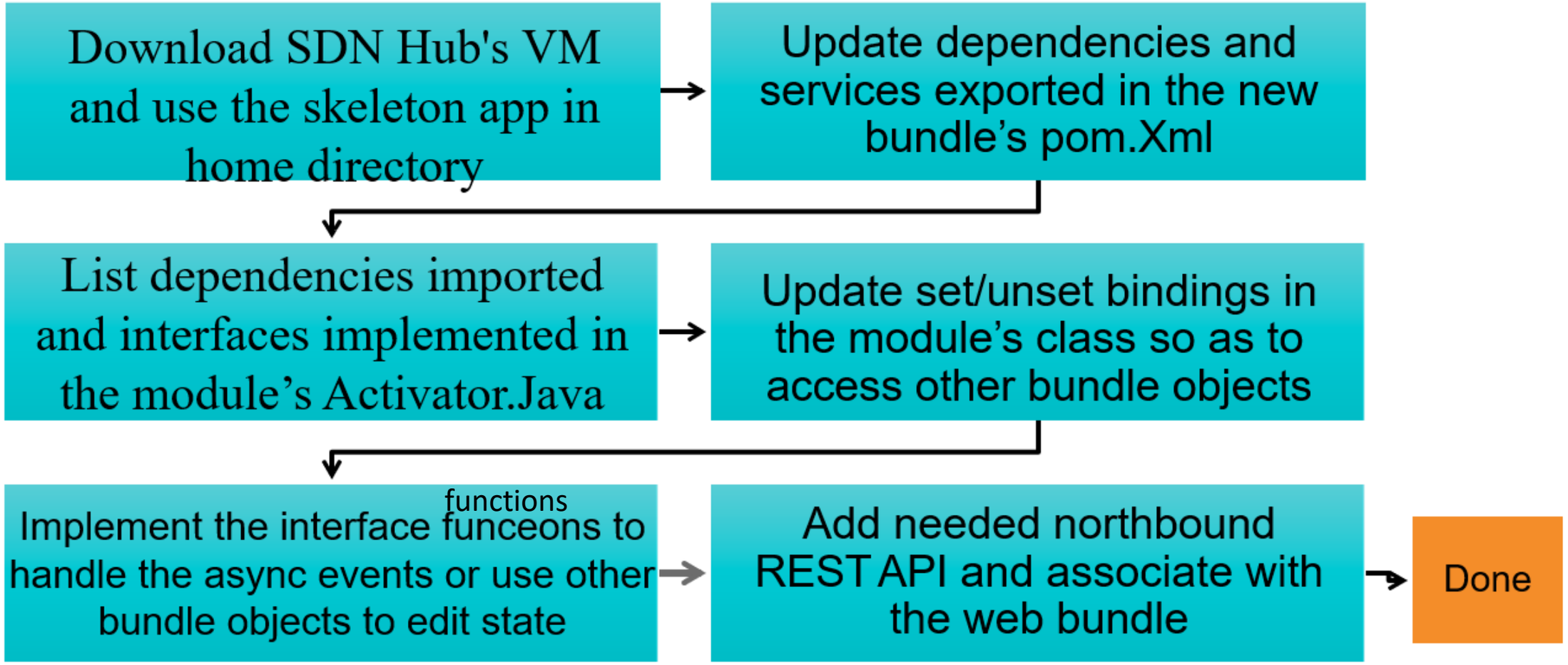


資料來源：蔡孟勳教授SDN/NDV教材—Controller

OpenDaylight Web Interface



Steps for Writing a new application



資料來源：蔡孟勳教授SDN/NDV教材—Controller

Main Constructs

A. Packet in event handling:

- public class TutorialL2Forwarding implements IListenDataPacket
  - Indicates that the class will handle any packet_in events
- public PacketResult receiveDataPacket(RawPacket inPkt) { ... }
  - Call-back function to implement in the class for receiving packets

B. Packet parsing

- Packet formattedPak = this.dataPacketService.decodeDataPacket(inPkt);
- byte[] srcMAC = ((Ethernet)formattedPak).getSourceMACAddress();
- long srcMAC_val = BitBufferHelper.toNumber(srcMAC);

C. Send message (packet_out or flow_mod) to switch

- RawPacket destPkt = new RawPacket(inPkt);
- destPkt.setOutgoingNodeConnector(p);
- this.dataPacketService.transmitDataPacket(destPkt);

Useful Interfaces and Bundles

Bundle	Exported interface	Description
arphandler	IHostFinder	Component responsible for learning about host location by handling ARP.
hosttracker	IfIptoHost	Track the location of the host relatively to the SDN network.
switchmanager	ISwitchManager	Component holding the inventory information for all the known nodes (i.e., switches) in the controller.
topologymanager	ITopologyManager	Component holding the whole network graph.
usermanager	IUserManager	Component taking care of user management.
statisticsmanager	IStatisticsManager	Component in charge of using the SAL ReadService to collect several statistics from the SDN network.

資料來源：蔡孟勳教授SDN/NDV教材—Controller

Useful Interfaces and Bundles

Bundle	Exported interface	Description
sal	IReadService	Interface for retrieving the network node's flow/port/queue hardware view
sal	ITopologyService	Topology methods provided by SAL toward the applications
sal	IFlowProgrammerService	Interface for installing/modifying/ removing flows on a network node
sal	IDataPacketService	Data Packet Services SAL provides to the applications
web	IDaylightWeb	Component tracking the several pieces of the UI depending on bundles installed on the system.

資料來源：蔡孟勳教授SDN/NDV教材—Controller

Summary

- OpenDaylight is an industry-backed effort to develop broader set of SDN solutions

- SDN is no longer just OpenFlow!
  - Possible to integrate a broad set of cloud-based applications
  - Set of functions is similar to other controllers

- Learning curve is significant.

  SDN Hub is a good starter kit!

參考資料

[1]台灣期貨雙月刊 2019年4月號，關鍵看法—軟體定義網路(SDN)架構之應用與探討

[2]國立成功大學資工系，蔡孟勳教授— SDN/NFV教材

[3]Korea,Postech,Department of Computer Science and Engineering,James Won-Ki Hong — Software Defined Networking:Introduction to SDN&Openflow

[4] 數位活氧科技，高銘聰—軟體定義網路(SDN)簡介與發展

[5]國立清華大學資工系，鍾葉青教授—虛擬化技術:Network Virtualization Software Defined Network

參考資料

[6]OpenFlow/SDN tutorial OFC/NFOEC March,2012

[7]The IEEE International Workshop on CAMAD(Computer-Aided Modeling,Analysis and Design of Communication Links and Networks) 2014 ,Athens,1-3 December — From dumb to smarter switches in software defined networks：an overview of data plane evolution

[8]A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood — On controller performance in software-defined networks. In USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE), 2012.

參考資料

[9] CSDN : https://blog.csdn.net/qq_29229567/article/details/88797395