Software Defined Networking (1/2)

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Xuetao Wei

weixt@sustech.edu.cn

Outline

- **Motivation for SDN**
- SDN Overview
- OpenFlow
- Case Study

Computer Industry

App

—— Open Interface ——

Windows (OS) or Linux or Mac OS

—— Open Interface ——

Microprocessor

Traditional Computer Industry:
Veritically integrated, Closed
Slow innovation, Small Industry

PC era Industry:
Horizontal, Open interfaces
Rapid innovation, Huge industry

Networking Industry

南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Vertically integrated, with "mainframe" mind-set

Routing, management, mobility management, access control, VPNs, …

Feature

Feature

OS

Custom Hardware

Million of lines of source code

6,000 RFCs

Billions of gates

Bloated

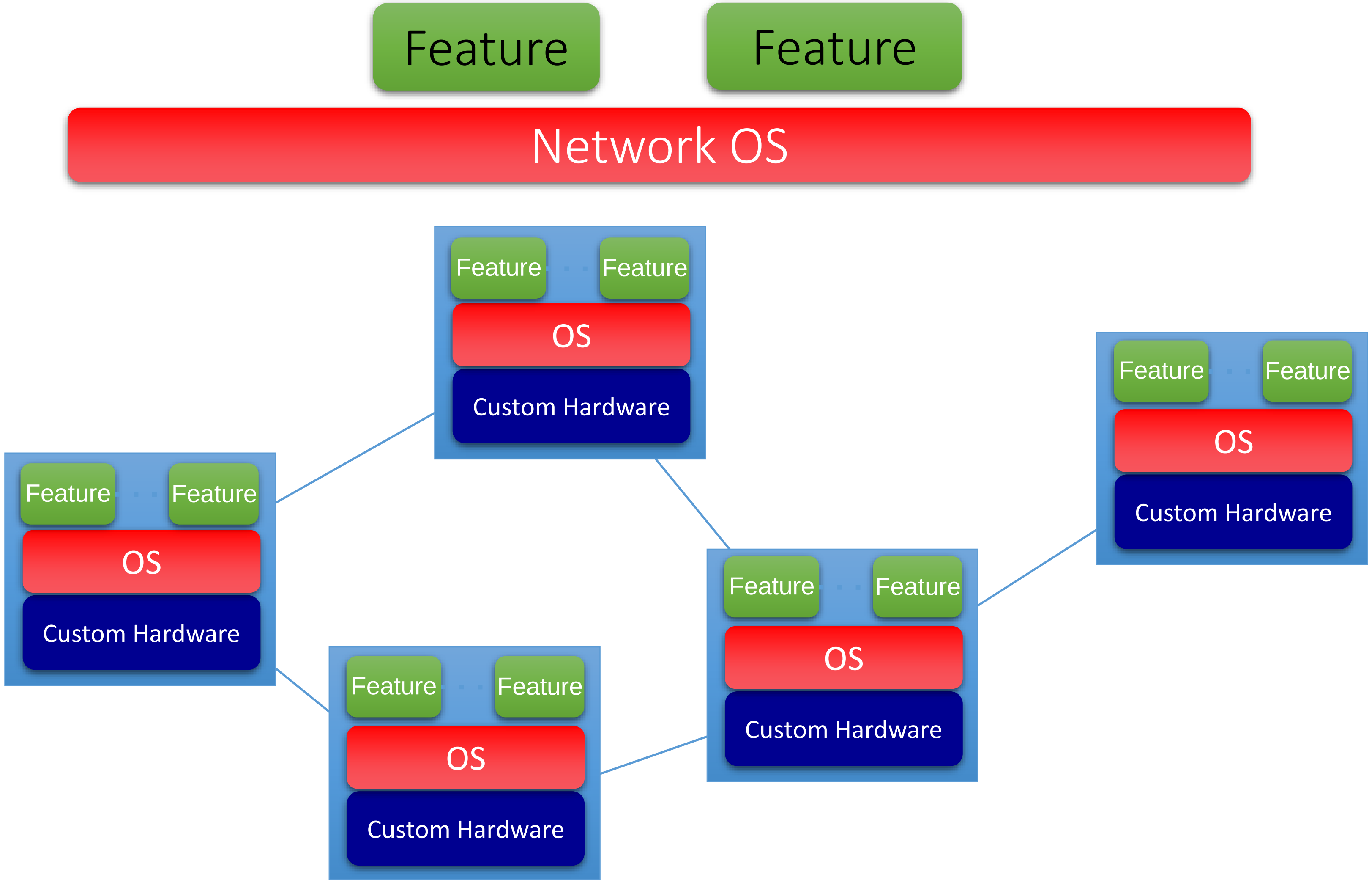
Power Hungry

Software Defined Network

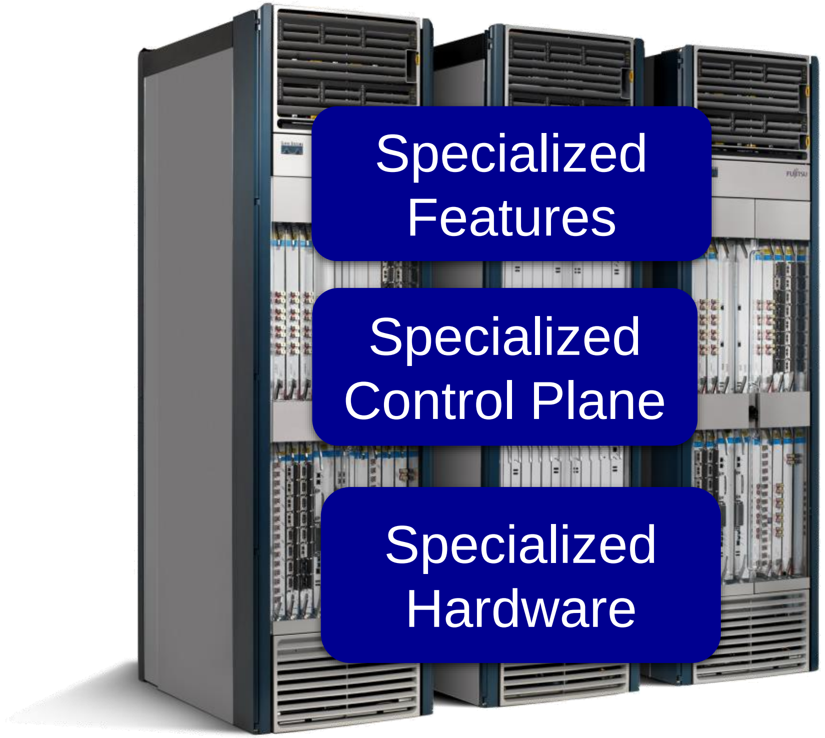
A network in which the **control plane** is physically separate from the **forwarding plane**.

and

A single **control plane** controls several **forwarding devices**.


Feature
Feature
Network OS
Feature
Feature
OS
Custom Hardware
Feature
Feature
OS
Custom Hardware
Feature
Feature
OS
Custom Hardware
Feature
Feature
OS
Custom Hardware
Feature
Feature
OS
Custom Hardware

Network Industry



App

Open Interface

Control Plane or Control Plane or Control Plane

Open Interface

Merchant Switching Chips

Vertically integrated
Closed, Slow innovation

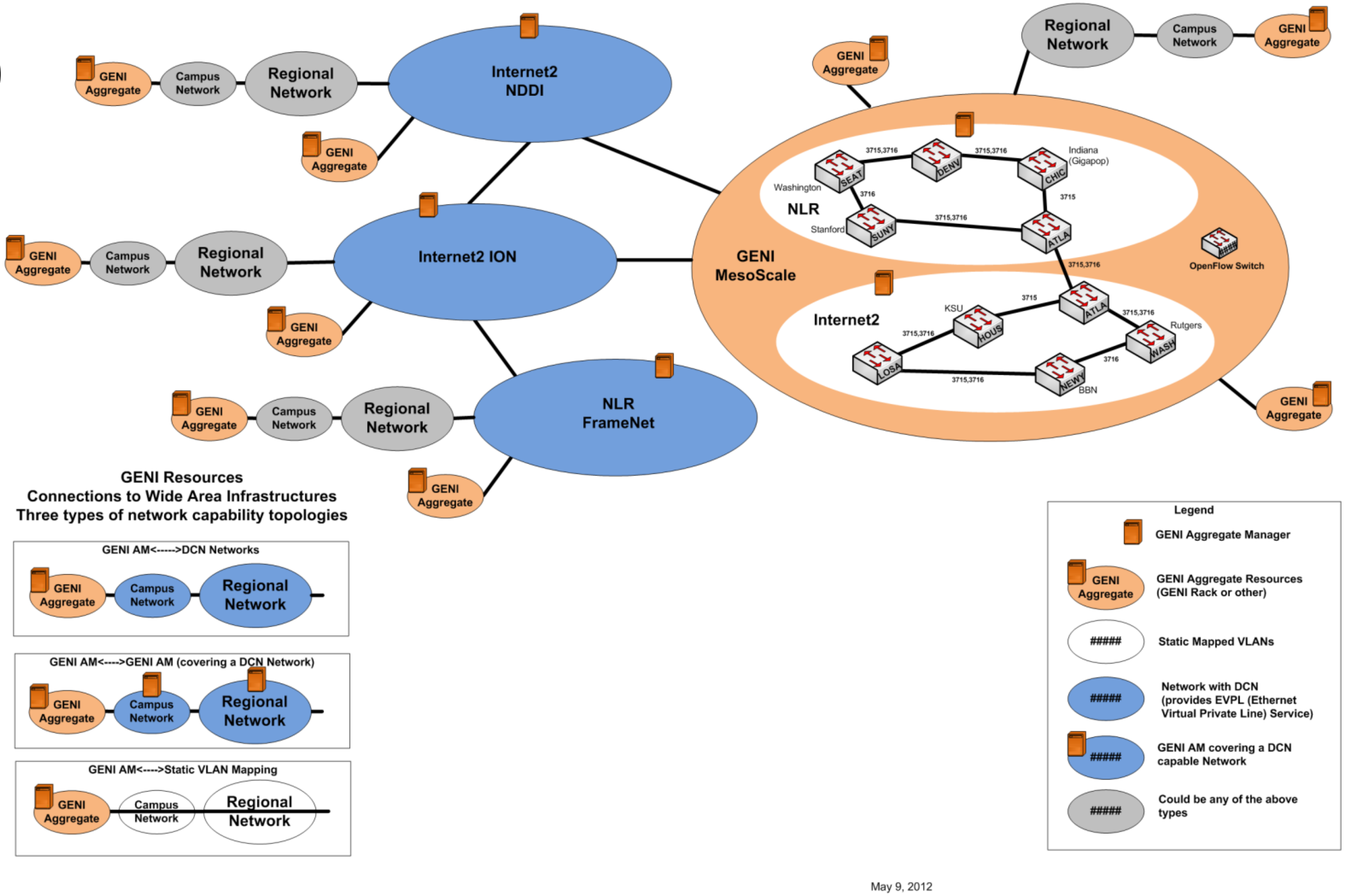
Horizontal
Open interfaces, Rapid innovation

Immediate Consequences

- Networks cost less: hardware is simple and streamlined again
- Networks cost less to operate: customized
- Networks are more reliable and more secure
- Puts network owners and operators in control of their destiny

History of SDN (1/3)

- **Background**: NGN (Next Generation Network Initiative)
  - Internet2
  - GENI (Global Environment for Network Innovations)



History of SDN (2/3)

1. **Clean Slate Project** (Nick McKeown @Standord), 2005~2007
   - Current network is bloated and convoluted by many things
   - Martin Casado: “Ethane: Taking Control of the Enterprise”, [*SIGCOMM 2007*]
   - Primary idea: central controller over multiple switches for a network

2. **Nicira startup** (Casado, McKeown, Scott Shenker@Berkely), 2007
   - NOX controller
   - OpenFlow interface
     (OpenFlow: Enabling Innovation in Campus Networks, [*SIGCOMM 2008*])
   - Acquired by VMware in 2012 for **1.26 billion** dollars!

History of SDN (3/3)

3. **OpenFlow 1.0** (Casado, Appenzeller@BigSwitch), 2009

4. ONF (Open Networking Foundation) founded, 2011
   - Deutshe Telecom, Facebook, Google, Microsoft, Verizon, Yahoo!
   - OpenFlow 1.1, 1.2, 1.3, 1.4

5. Many other organizations are founded, and many other SDN initiatives are launched since 2011...

6. Protocol-independent packet forwarding architecture is now being promoted by both academia (Stanford, Princeton, ...) and industry (Huawei)

Outline

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

- Motivation for SDN
- **SDN Overview**
- OpenFlow
- Case Study

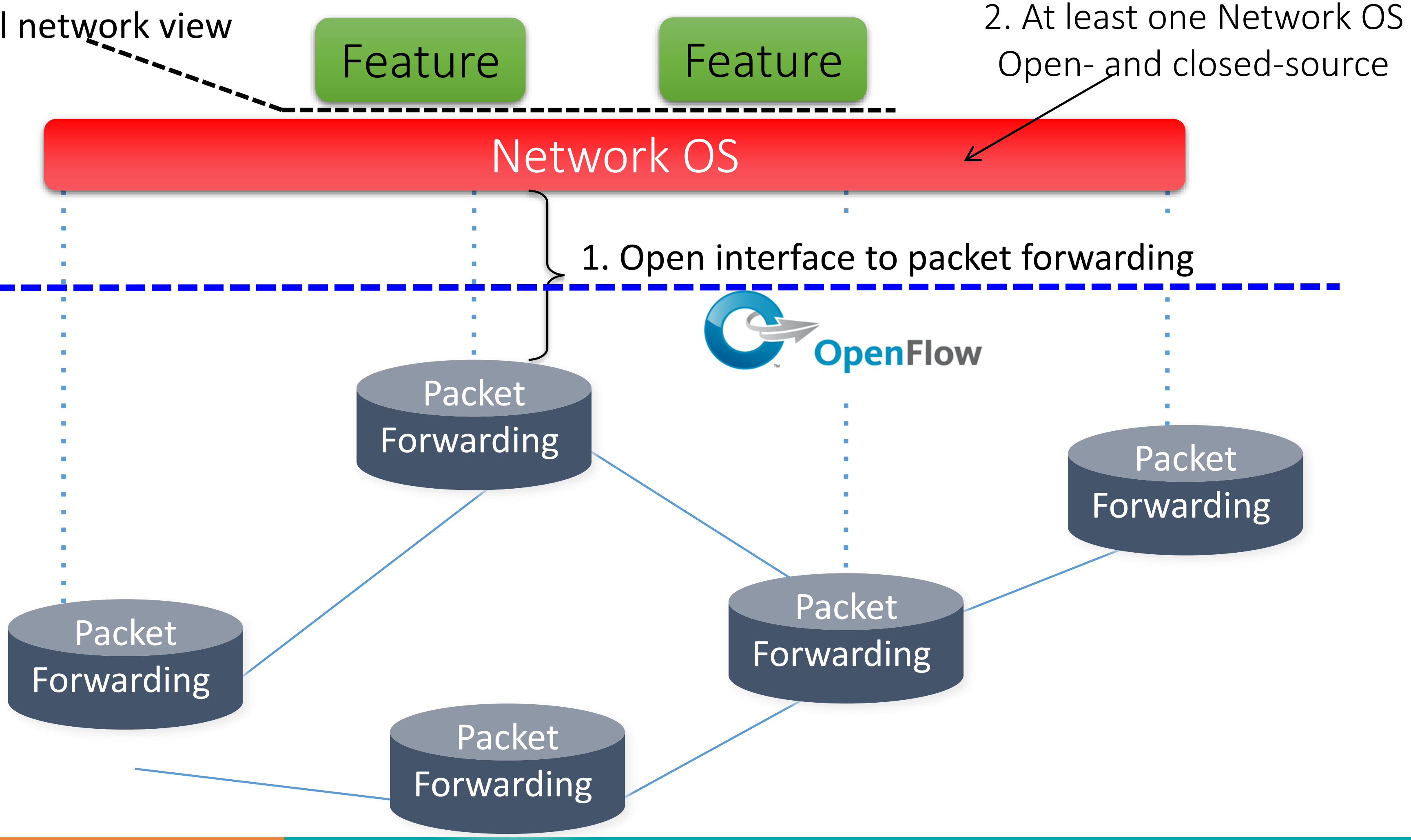
SDN Overview

SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

3. Consistent, up-to-date global network view

重要

Control plane:
Intelligence & Global view
Custom Software



Forwarding plane:
Simple & Dump & Efficient
Commodity Hardware

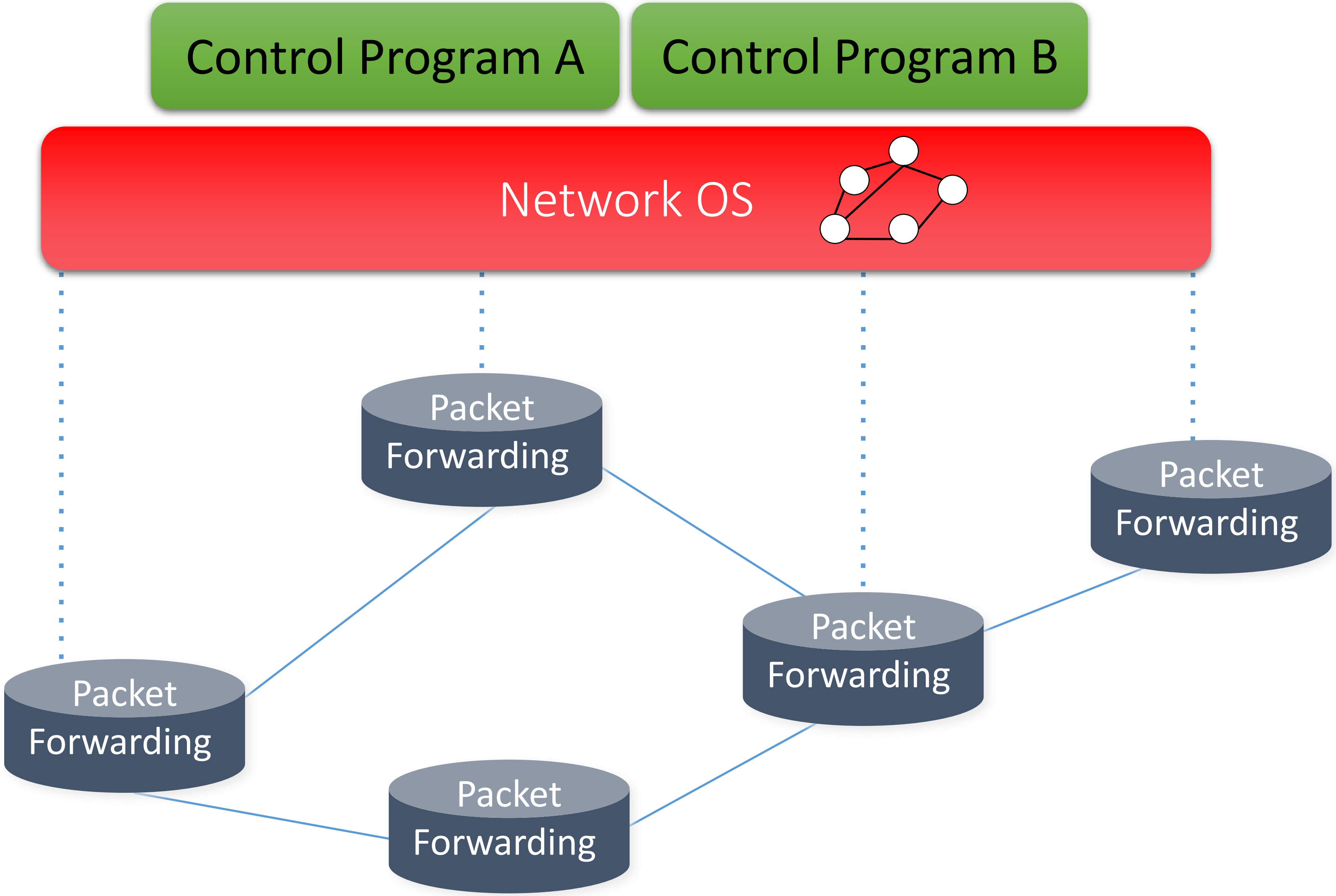
Network OS

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

- A distributed system that creates a consistent, up-to-date network view
  - Runs on servers (controllers) in the network
  - Examples: Floodlight, POX, Pyretic, Nettle ONIX, Beacon, …

- Uses forwarding abstraction to
  - Get state information **from** forwarding elements
  - Give control directives **to** forwarding elements

Control Program



Control Program

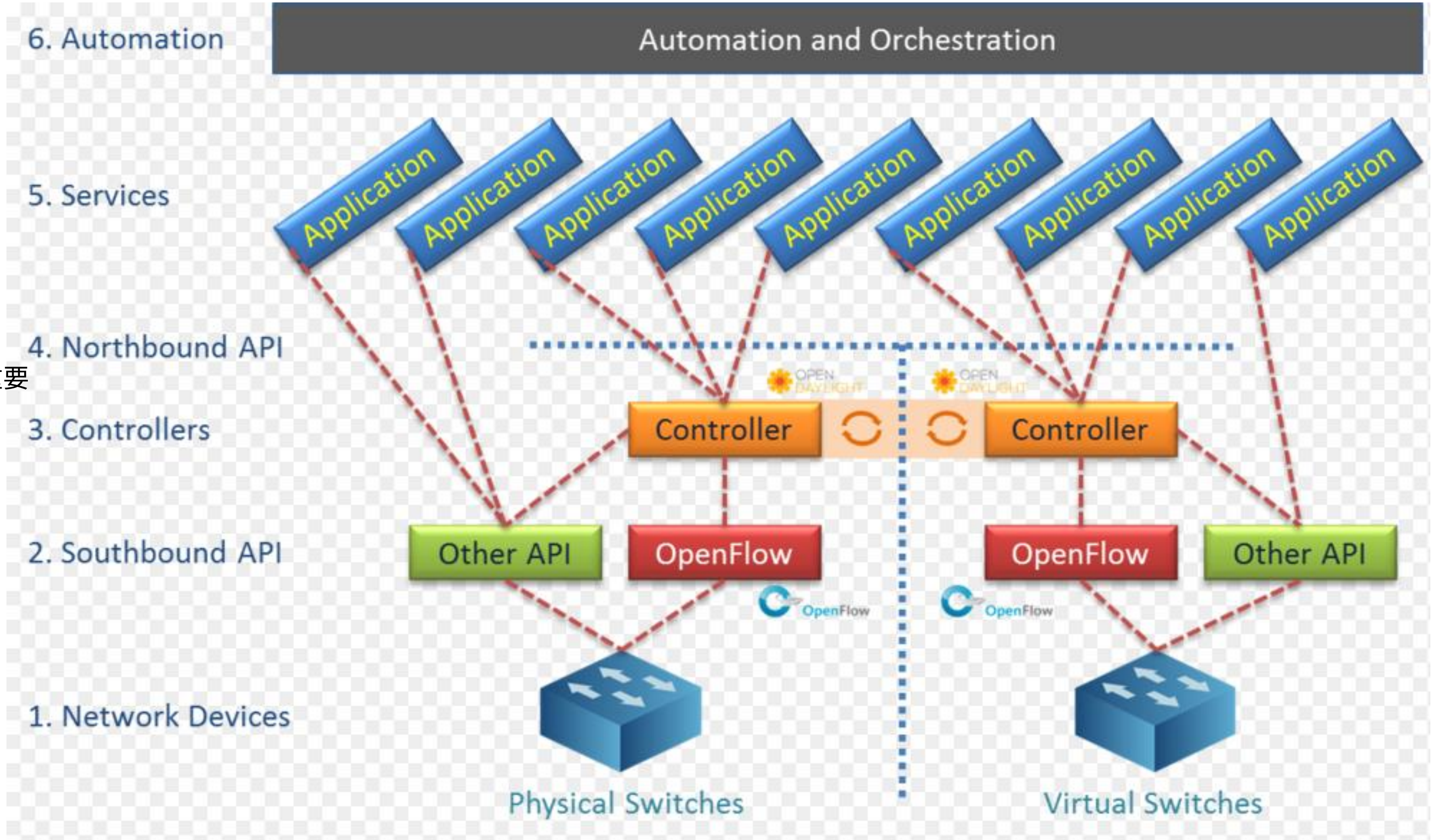
- Control program implements network features

- It operates on view of network
  - **Input**: global network view (graph/database)
  - **Output**: configuration of each network device

- Control program is not a distributed system
  - Abstraction hides details of distributed state

A Complete SDN Hierarchy

重要



1. Network Devices

- Abstracted as the forwarding plane or data plane, which forward packets based on flow tables
- Can be either hardware switches or software switches (OVS), can also be other devices like routers
- They receive instructions from the controller via the **Southbound Interface**, which will configure the flow table. They also report states and events to the controller through the **Southbound Interface**

2. Southbound Interface

南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

重要

- The interface between the control plane and data plane
- Traditional Southbound interfaces are proprietary, and the differences of Southbound interfaces among different vendors prevent managing heterogenous network devices as a whole within or across a network
- Standardization of the Southbound interface is urgently needed, and OpenFlow is a big step forward

3. Controller

- SDN controller configures the switches with flow tables, which specifies the rules for packet forwarding
- An SDN controller can be any x86 Linux servers or Windows servers, or other computers running networking software
- An SDN controller may control multiple switchs (and contralized control is promoted by SDN community), it is also possible that a switch is controlled by multiple controllers (one master, multiple slave)
- Controller decides the ecosystem, and is the kernel element in SDN network, therefore companies are competing in this area fiercely

4. Northbound Interface

重要

- Northbound Interface in traditional network refers to the interface between the switch control plane and the network management software (SNMP, TL1, ...)

- In SDN, Northbound Interface is the interface between the controller and the Apps/Services

- So far Northbound Interface is not standardized, and is much more complex to be standardized than the Southbound Interface

5. Apps/Services

重要

- With SDN, traditional network services can be better implemented, and new services can be praticed more easily
- **Load balancing**: global view and orchestration of the network simplifies load balancing
- **Security**: can be enforced with close collaboration between different switches coordinated by the centralized control plane
- **Monitoring and Performance Management**: can be implemented at the network level instead of at the switch level
- **LLDP**: topology discovery can be realized by the centralized controller easily and efficiently

Outline

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

- Motivation for SDN
- SDN Overview
- **OpenFlow**
- Case Study

Forwarding Abstraction

- Purpose: Abstract away forwarding hardware implementation details

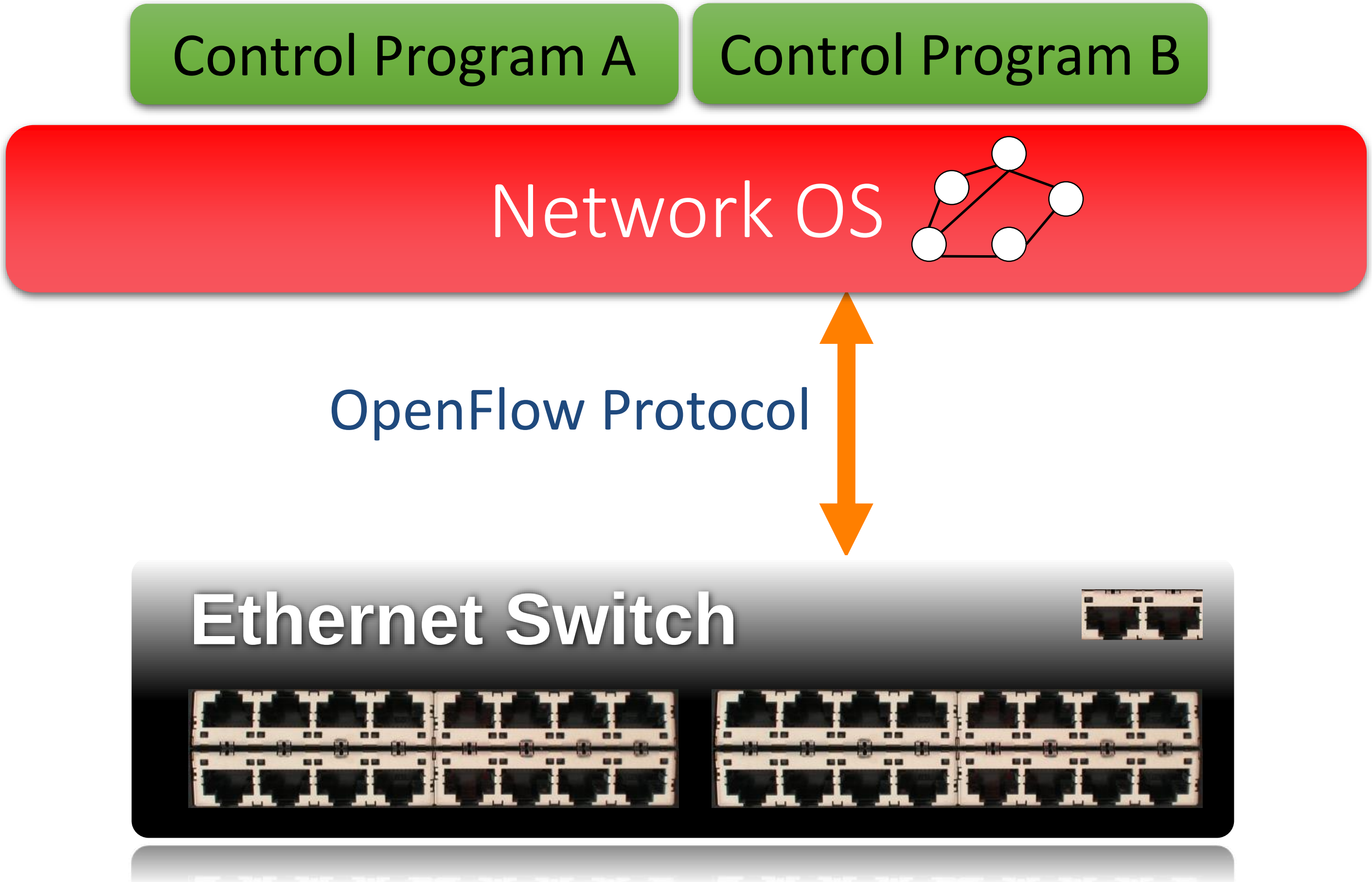
- Flexible
  - Behavior specified by control plane
  - Built from basic set of forwarding primitives

- Minimal
  - Streamlined for speed and low-power
  - Control program not vendor-specific

OpenFlow is an example of such an abstraction

OpenFlow

- OpenFlow: An API between the control plane and dataplane

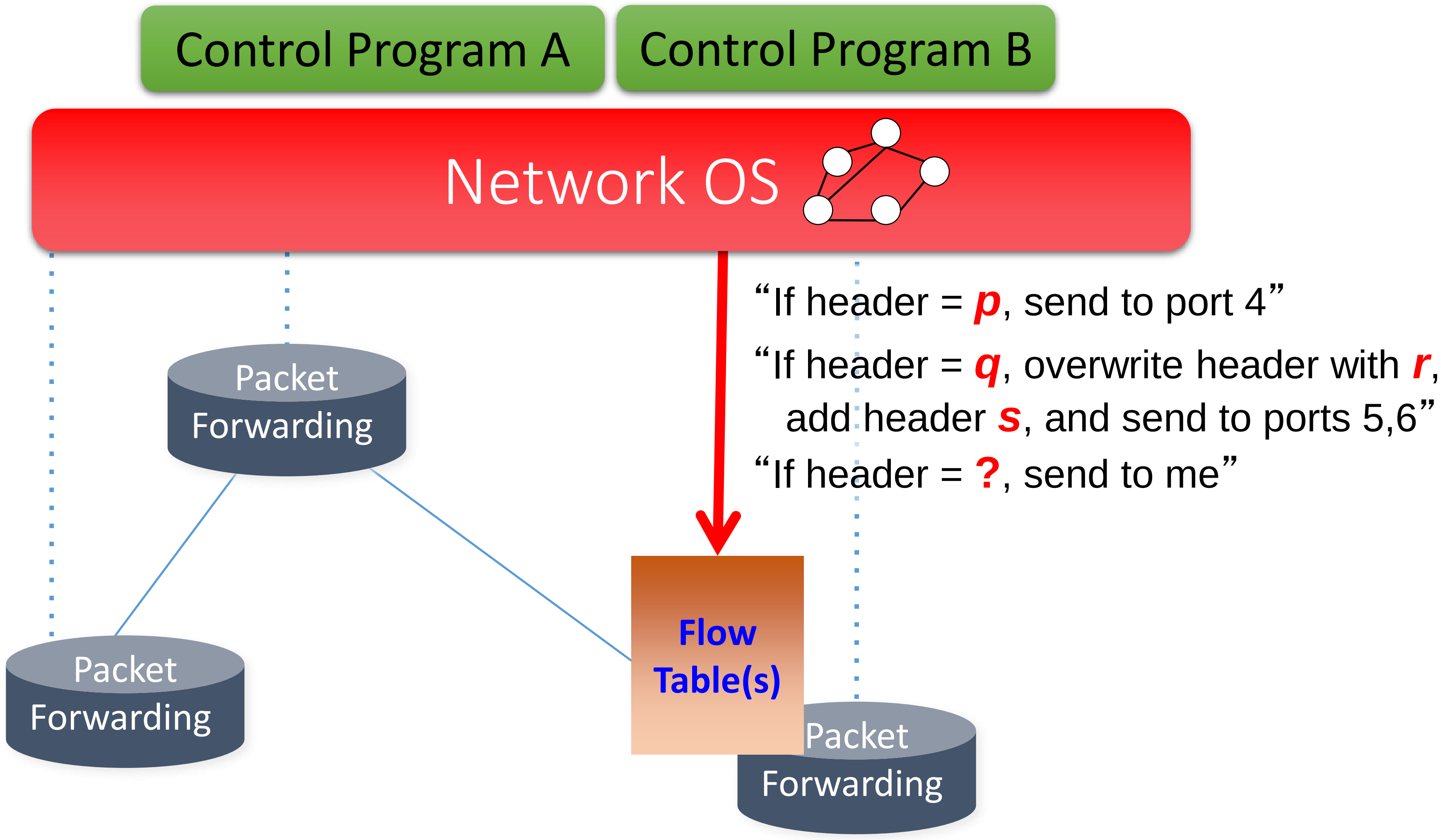


Like the **ISA** between
SW & HW in a computer

OpenFlow as An Interface

- OpenFlow is like an x86 instruction set for the network
- Provides open interface to “black box” networking node (ie. Routers, L2/L3 switch) to enable visibility and openness in network
- Separation of control plane and data plane by OpenFlow
  - The datapath of an OpenFlow Switch consists of a Flow Table, and an action associated with each flow entry
  - The control path consists of a controller which programs the flow entry in the flow table

Flow Tables



Control Program A
Control Program B
Network OS
“If header = p, send to port 4”
“If header = q, overwrite header with r, add header s, and send to ports 5,6”
“If header = ?, send to me”
Packet Forwarding
Packet Forwarding
Flow Table(s)
Packet Forwarding

<Match, Action> Primitives

南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

- ***Match*** arbitrary bits in headers:

Header	Data

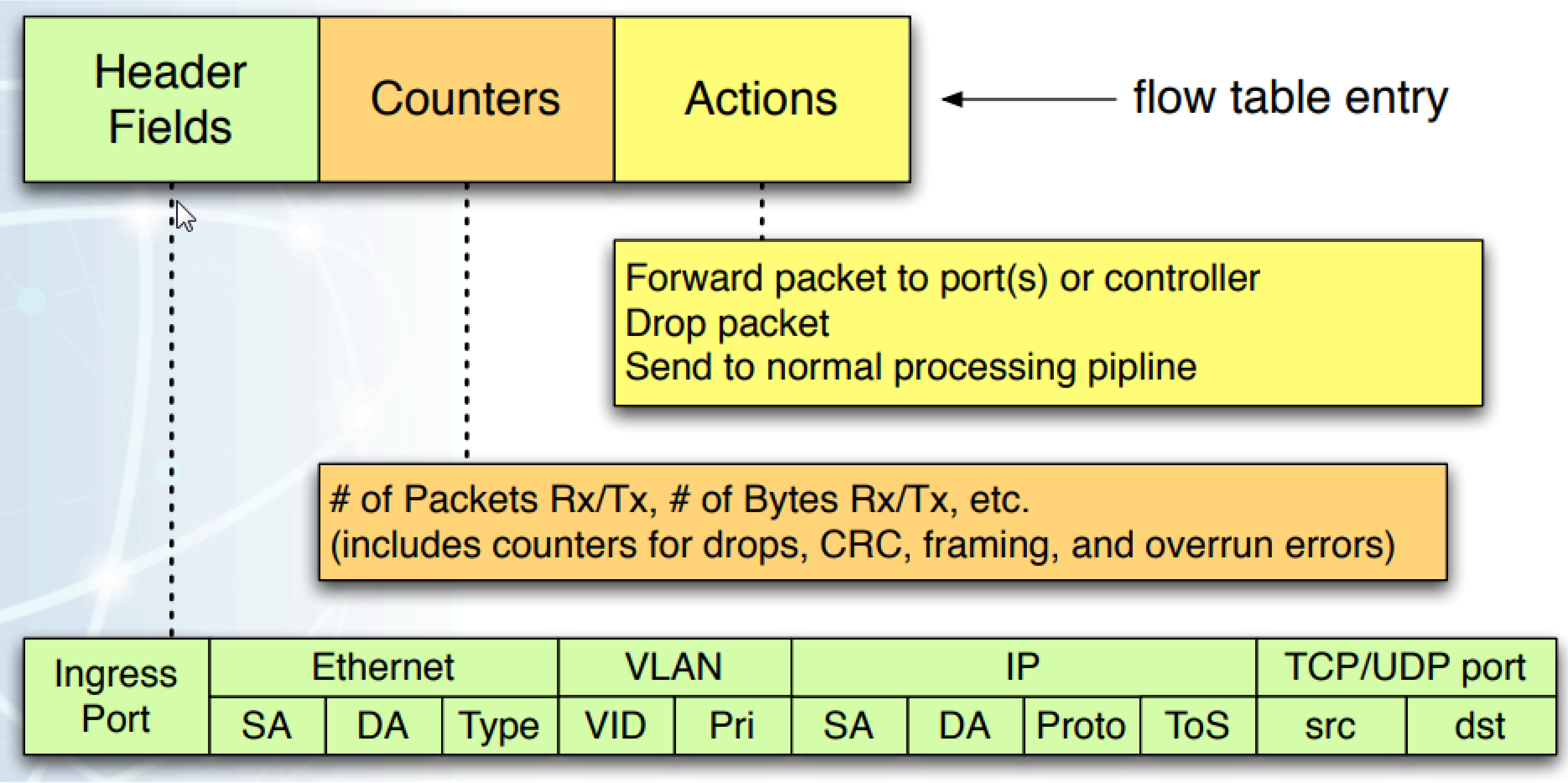
Match: 1000x01xx0101001x

  - Match on any header, or new header
  - Allows any flow granularity

- ***Action***
  - Forward to port(s), drop, send to controller
  - Overwrite header with mask, push or pop
  - Forward at specific bit-rate

Flow Table Entries

重要



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

Examples

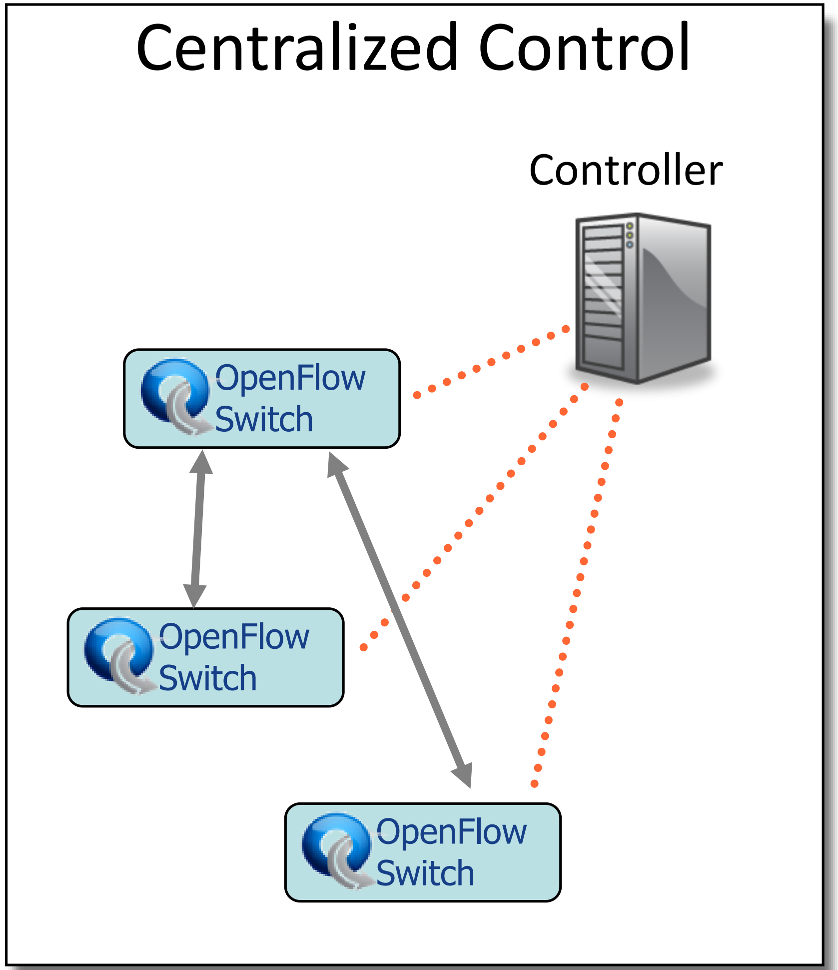
Routing

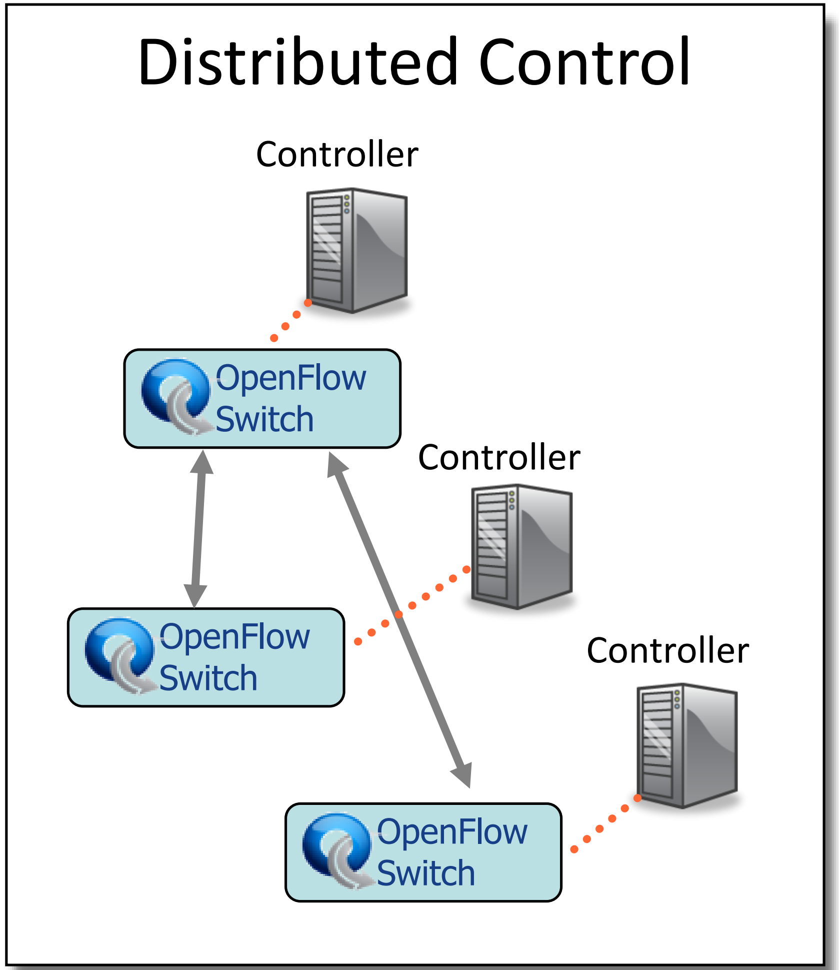
Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	5.6.7.8	*	*	*	port6

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

Centralized vs Distributed Control





Flow Routing vs. Aggregation

南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

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

Reactive vs. Proactive (pre-populated)

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

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

Controller-Switch Interactions

quiz不考

- **Controller-to-Switch Messages**
  - **Features**: query for functionalities supported by the switch
  - **Configuration**: configure the switch or query its configuration parameters
  - **Modify-State**: modify the flow table (add/delete/modify)
  - **Read-State**: access various state information of the switch (e.g., counter)
  - **Packet-out**: send out a packet matching a given flow entry
  - **Barrier**: enforce order between messages
  - **Role-request**: tell the switch its role when multiple controllers are connected to it
  - ...

Controller-Switch Interactions

- **Asynchronous Messages (Switch-to-Controller)**
  - **Packet-in**: a message to controller when a packet matches a flow entry with action "output to Controller-Port"
  - **Flow-removed**: when a flow entry is removed from the flow table (due to aging or instructed by the controller)
  - **Port-status**: a message to the controller when a port status is changed (link down/up)
  - **Error**: a message to the controller when the switch encounters errors

Controller-Switch Interactions

- **Symmetric Messages**
  - **Hello:** a message to inform the others when a controller or switch is booted up
  - **Echo:** an echo/reply handshake to make sure the connection is OK, or to measure the delay of the connection
  - **Experimenter**: a message for vendor-specific extensions
  -

Controller-Switch Workflow

1. Initialization
   - Switch: a default flow table with a single default flow entry that either discards all packets or forwarding them to the controller for processing

2. Business-driven flow table update by controller
   - According to business requirements, the controller adds new flow entries onto the flow table in the switch, updates existing entries, and deletes obsolete entries
   - Flow table update can be carried out by network operators, or automated by network Apps/Services. The switch itself can also retire aging flow entries

3. Packet processing in switch
   - An incoming packet searches the flow table for a match, then execute the corresponding actions

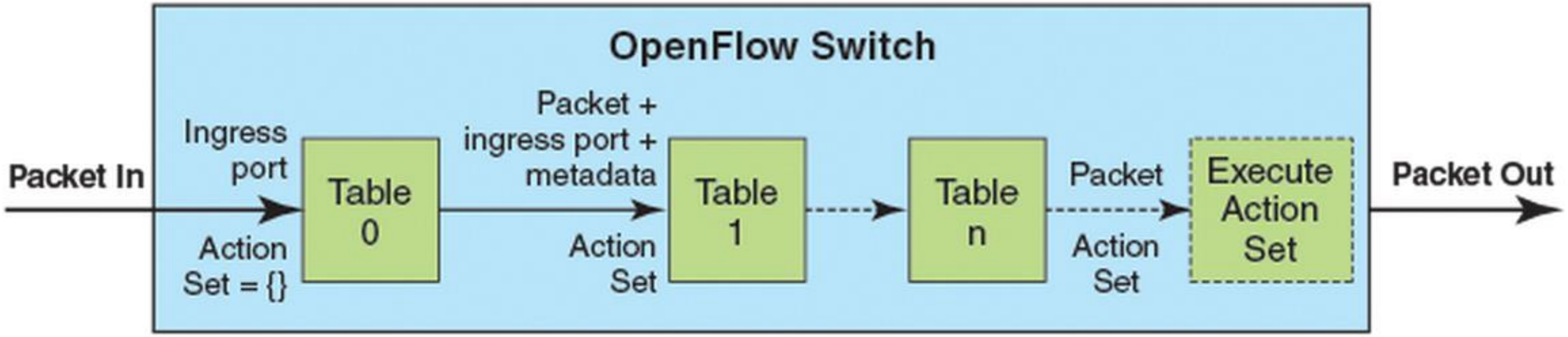
SDN Standardization on OpenFlow

- Open Network Foundation (ONF)
  - Dec. 2009: OpenFlow 1.0
  - Feb. 2011: OpenFlow 1.1
  - Dec. 2011: OpenFlow 1.2
  - Jun. 2012: OpenFlow 1.3
  - Oct. 2013: OpenFlow 1.4

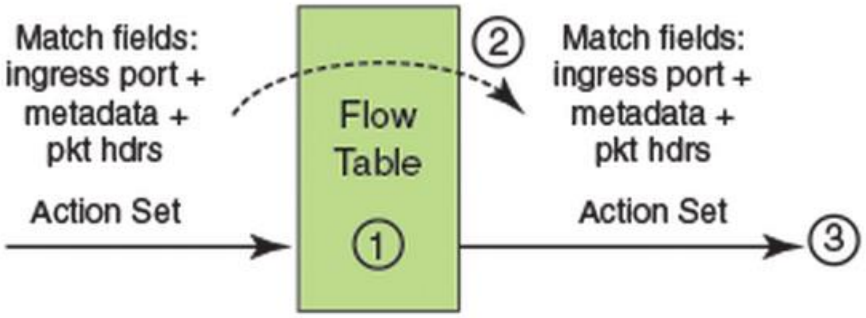
Where It's Going

- OF v1.3
  - multiple tables: leverage additional tables
  - tags and tunnels
  - multipath forwarding
  - per flow meters
- OF v2+
  - generalized matching and actions: protocol independent forwarding

Multiple Tables (OF v1.1+)



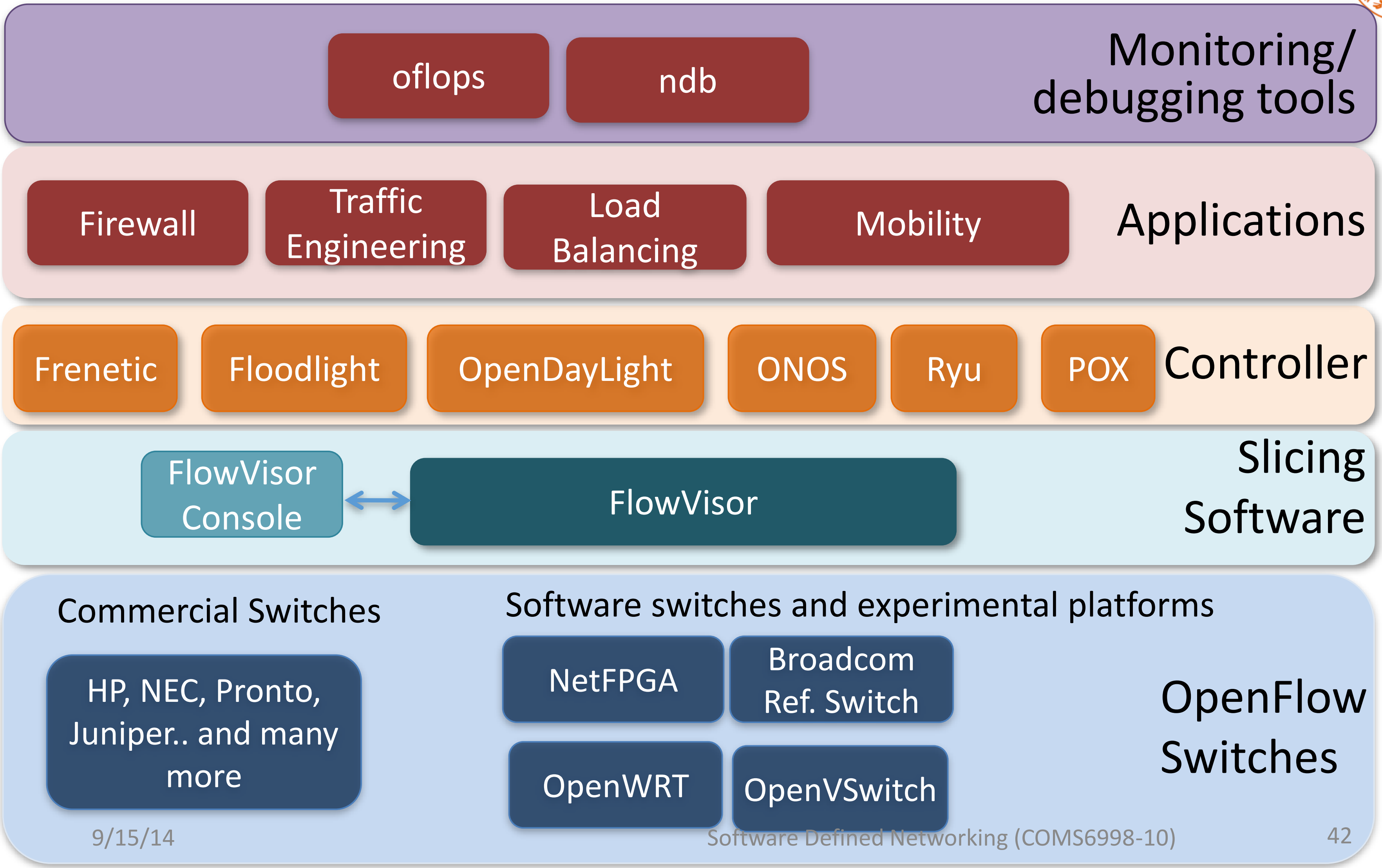
{a} Packets are matched against multiple tables in the pipeline



{b} Per-table packet processing

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

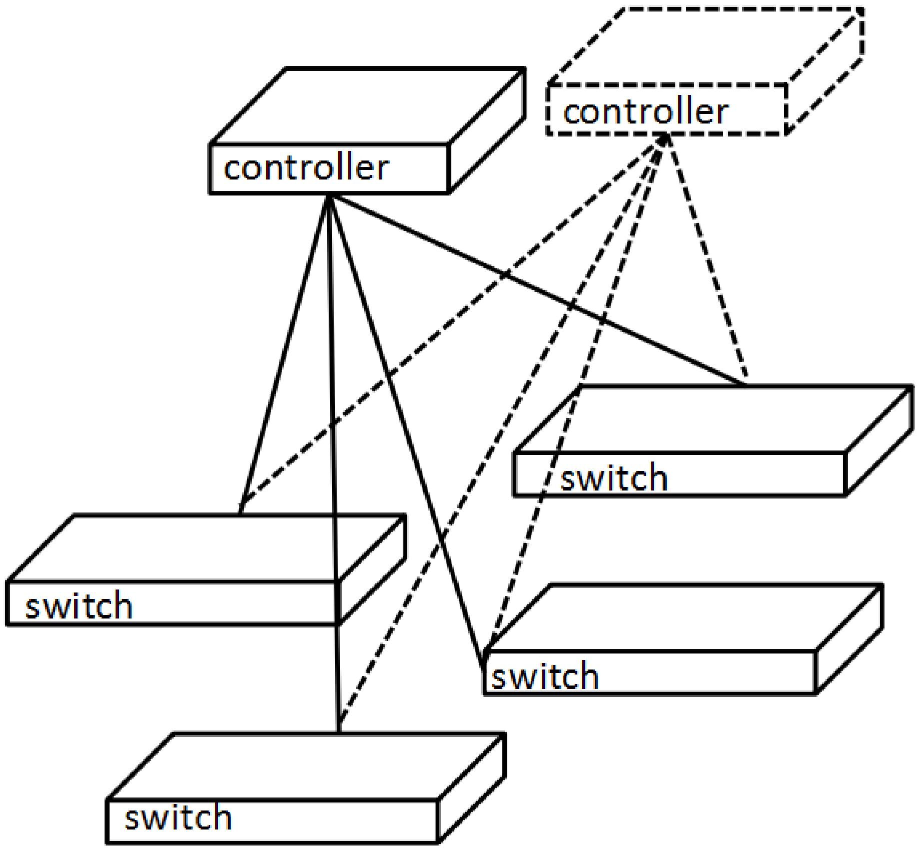
OpenFlow Building Blocks



OpenFlow Configuration and Management Protocol: OF-Config

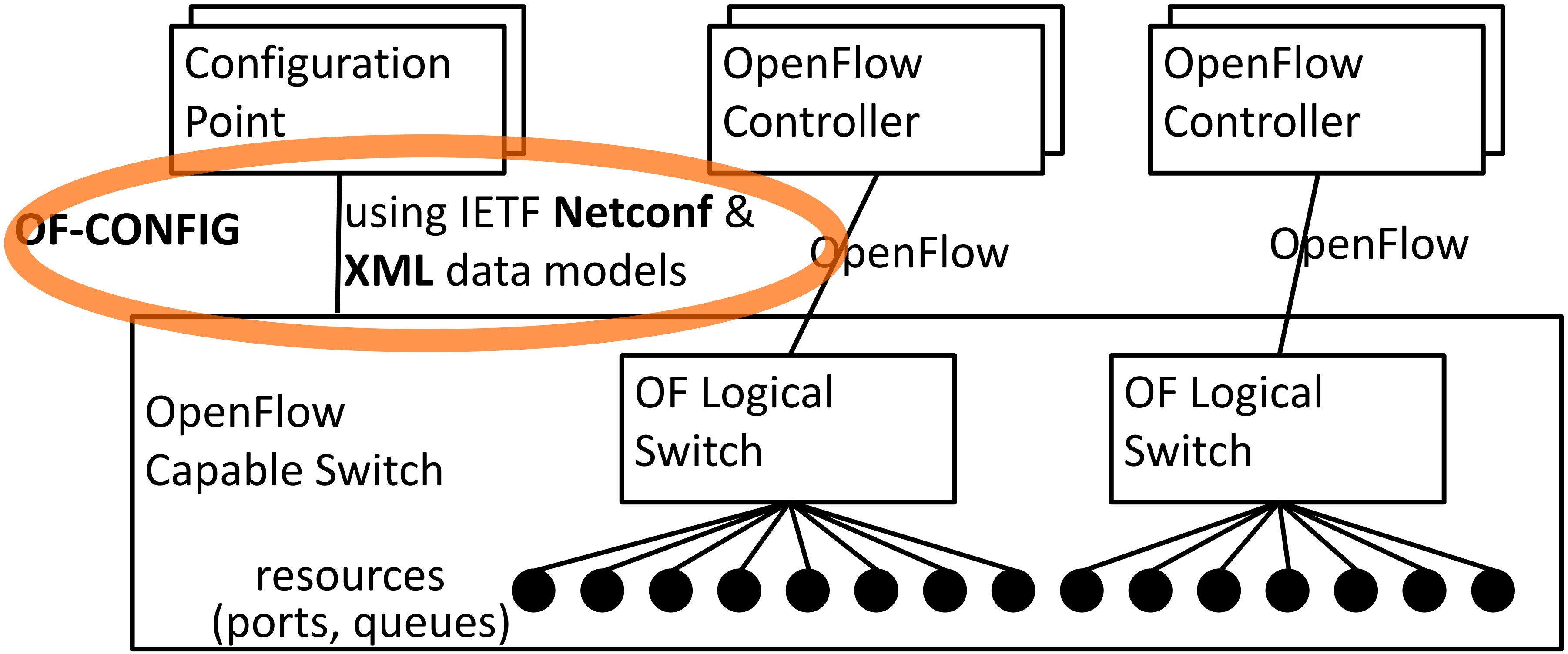
- Bootstrap OpenFlow network
  - Switch connects to controller
  - Controller(s) to connect to must be configured at switches

- Allocate resources within switches
  - Ports
  - Queues
  - . . .



OpenFlow Configuration and Management Protocol: Reference Model

- Configuration Point
  - Source of switch configuration
- OpenFlow Capable Switch
  - Hosts one or more logical switches
- OpenFlow Controller
- OpenFlow Logical Switch
  - instance of an OpenFlow Switch



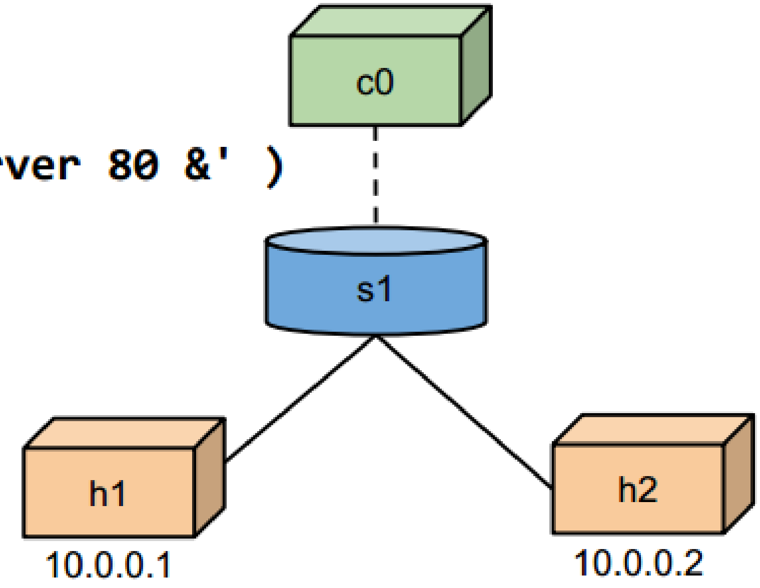
Mininet

- Mininet is an SDN network emulator

```
# mn
# mn --topo tree,depth=3,fanout=3 --
link=tc,bw=10
mininet> xterm h1 h2
h1# wireshark &
h2# python -m SimpleHTTPServer 80 &
h1# firefox &
# mn --topo linear,100
# mn --custom custom.py --topo mytopo
```

Mininet API Basics

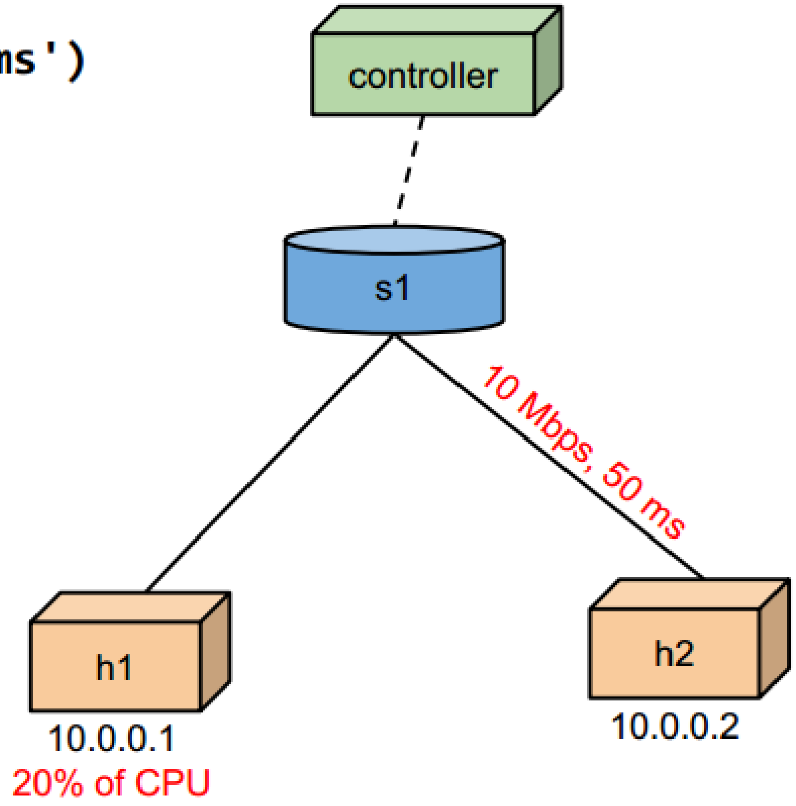
```
net = Mininet()                        # net is a Mininet() object
h1 = net.addHost( 'h1' )           # h1 is a Host() object
h2 = net.addHost( 'h2' )           # h2 is a Host()
s1 = net.addSwitch( 's1' )             # s1 is a Switch() object
c0 = net.addController( 'c0' )     # c0 is a Controller()
net.addLink( h1, s1 )                  # creates a Link() object
net.addLink( h2, s1 )
net.start()
h2.cmd( 'python -m SimpleHTTPServer 80 &' )
sleep( 2 )
h1.cmd( 'curl', h2.IP() )
CLI( net )
h2.cmd('kill %python')
net.stop()
```



Performance Modeling

SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

```
# Use performance-modeling link and host classes
net = Mininet(link=TCLink, host=CPULimitedHost)
# Limit link bandwidth and add delay
net.addLink(h2, s1, bw=10, delay='50ms')
# Limit CPU bandwidth
net.addHost('h1', cpu=.2)
```



Switch Vendors

Model	Virtualize	Notes	
HP Procurve 5400zl or 6600	1 OF instance per VLAN	-LACP, VLAN and STP processing before OpenFlow -Wildcard rules or non-IP pkts processed in s/w -Header rewriting in s/w -CPU protects mgmt during loop	
NEC IP8800	1 OF instance per VLAN	-OpenFlow takes precedence -Most actions processed in hardware -MAC header rewriting in h/w	
Pronto 3240 or 3290 with Pica8 or Indigo firmware	1 OF instance per switch	-No legacy protocols (like VLAN and STP) -Most actions processed in hardware -MAC header rewriting in h/w	

Controller Vendors

南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Vendor	Notes
Nicira's NOX	•Open-source GPL •C++ and Python •Researcher friendly
Nicira's ONIX	•Closed-source •Datacenter networks
Ryu	•Open-source GPL •Python

Vendor	Notes
Stanford's Beacon	•Open-source •Researcher friendly •Java-based
BigSwitch controller	•Ha open source version •Based on Beacon •Enterprise network
OpenDayLight	•Open-source •Based on Java
Frenetic	•Open-source •Written in functional programming languages

Outline

- Motivation for SDN
- SDN Overview
- OpenFlow
- **Case Study**

Case Study: Google B4

- B4: a private WAN connecting Google's data centers across the planet

Features and Goals

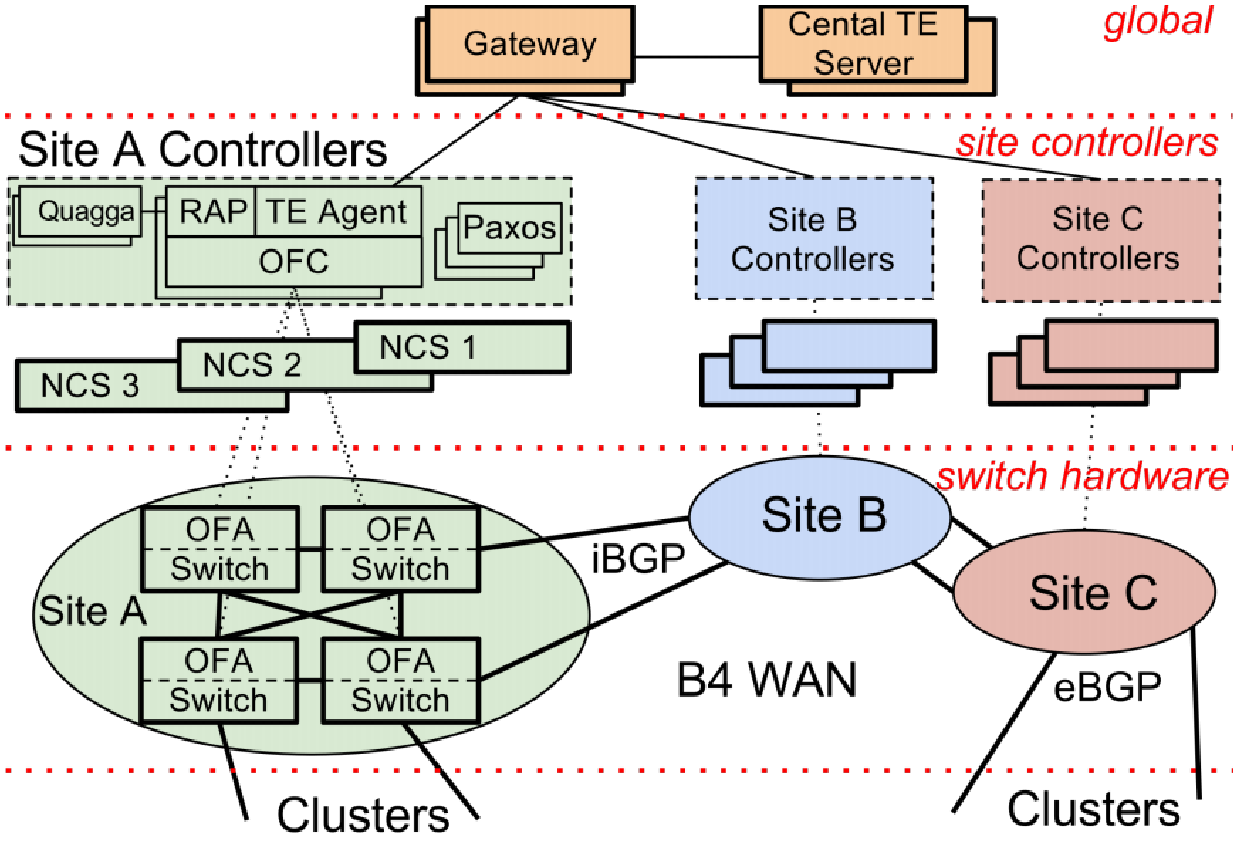
南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

- **Features of B4**
  - Massive bandwidth requirements deployed to a modest number of sites
  - Elastic traffic demand that seeks to maximize average bandwidth
  - Full control over the edge servers and network, which enables rate limiting and demand measurement at the edge

- **Goals**
  - TE: Centralized traffic engineering service that can drive links to near 100% utilization
  - Split application flows among multiple paths to balance capacity against application priority/demands

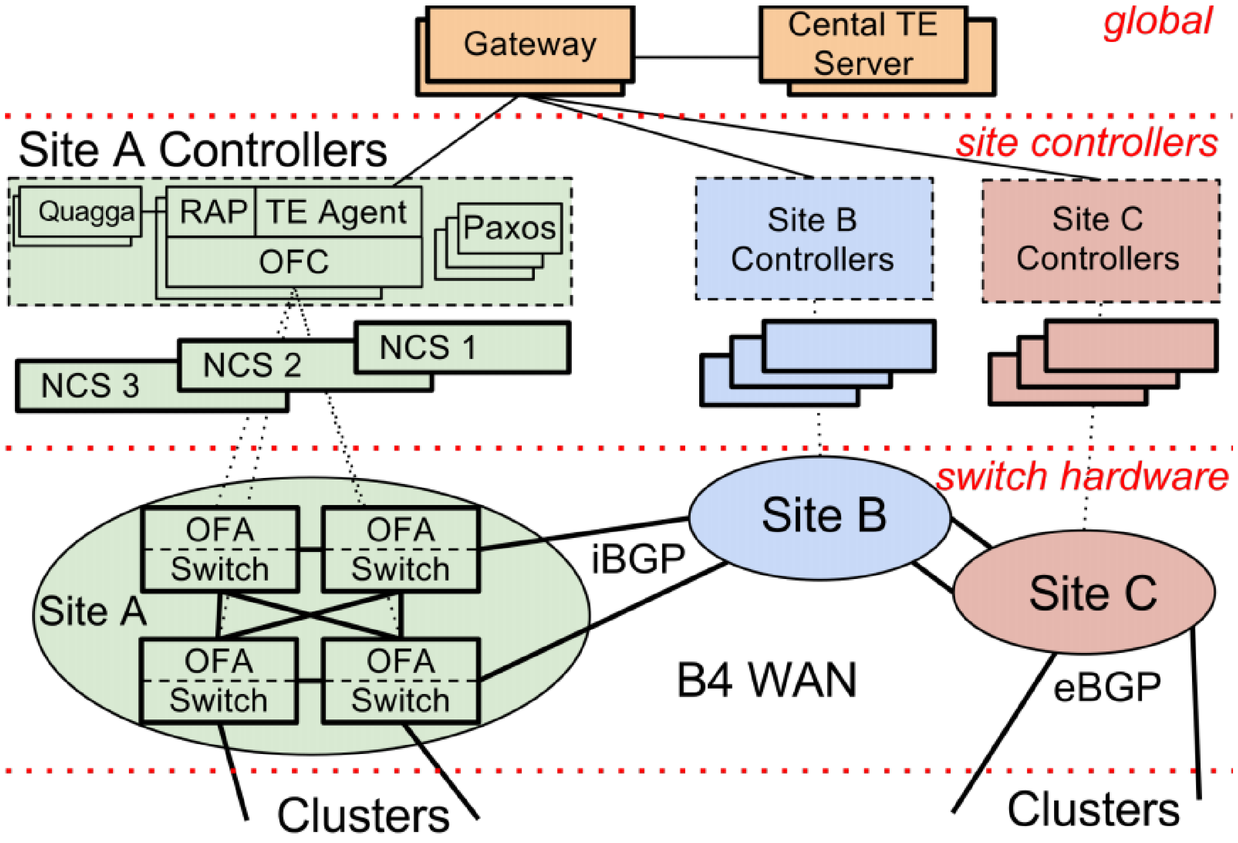
B4 Architecture Overview

- **Switch layer**
  - Primarily forwards traffic and does not run complex control software
- **Site controller layer**
  - Consists of Network Control Servers (NCS) hosting both OpenFlow controllers (OFC) and Network Control Applications (NCAs)
  - These servers enable distributed routing and central traffic engineering as a routing overlay



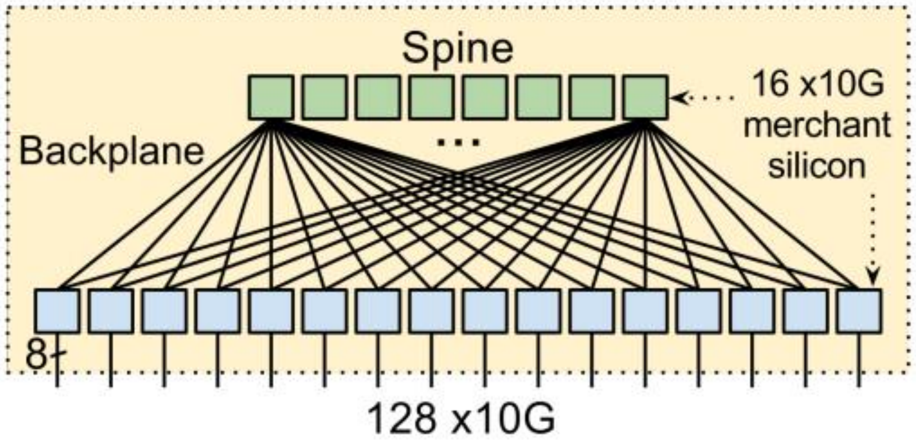
B4 Architecture Overview

- **Global layer**
  - Consists of logically centralized apps (e.g. an SDN Gateway and a central TE server) that enable the central control of the entire network via the site-level NCAs
  - The SDN Gateway abstracts details of OpenFlow and switch hardware from the central TE server



Switch Design

- Build B4 switches from multiple merchant silicon switch chips in a two-stage Clos topology with a copper backplane
  - OpenFlow Agent (OFA): a user-level process running on the switch hardware
  - OpenFlow Controller (OFC): OFA connects to a remote OFC, accepting OpenFlow (OF) commands and forwarding packets and link/switch events to the OFC



Switch Design

More details:
B4: Experience with a Globally-Deployed Software Defined WAN, [SIGCOMM 2013]

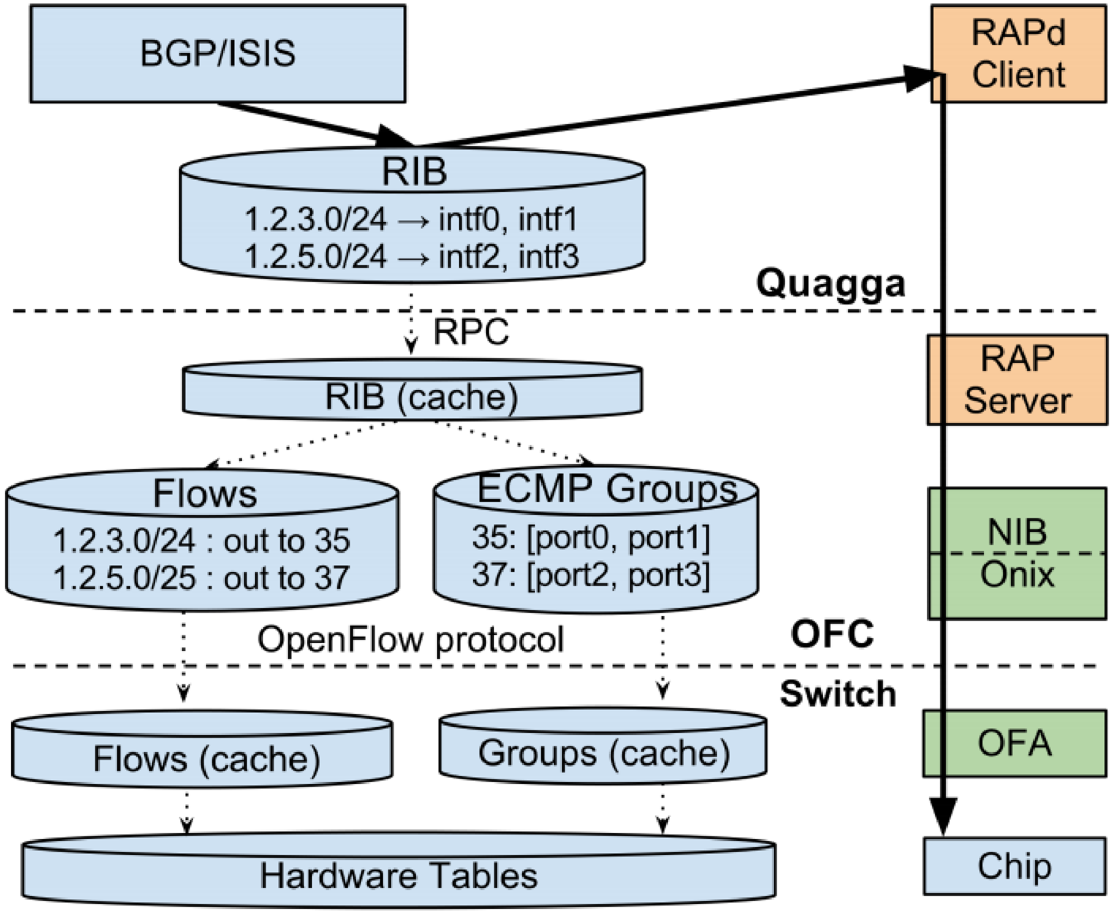


Figure 4: Integrating Routing with OpenFlow Control.