Software Defined Networking
软件定义网络

Lecturer: CUI Lin

Department of Computer Science
Jinan University

Network Engineering

- LAN
  - Ethernet, 802.11
  - Switch Configuration
  - VLAN, STP/RSTP/MSTP, VRRP
- WAN
  - PPP, POS
  - Router Configuration
  - RIP, IGRP, OSPF, IS-IS, BGP

"Traditional" Networking:

- Network Management
  - Distributed control
  - SNMP
  - **CLI** (Command Line Interface)
- Technologies, protocols, commands…
- Certification
  - Cisco： CCNP, CCIE
  - Huawei: HCNP, HCIE

SDN has become an important trend…

OPEN NETWORKING FOUNDATION

Entire backbone

runs on SDN

Bought for **$1.2 x 10^9** (mostly cash)

Great impact on traditional network engineering

一名网络工程师尴尬的现状？

我是一名网络工程师

How SDN is going to affect your j as a network engineer

Published on May 28, 2014

Joachim Bauernberger
FOSS, Automation, Resilience, Security;

21 articles +

Remember the time when every l company had its own telecoms department? A group of experts w around installing cabling, terminal phones, ensuring central commur systems were in place.

This changed very quickly as Telecommunications moved to Vo the IT department then took over the tasks that the old telecommun group did.

For those who worked in these job meant learn new skills and adapt to the changing environment or die.

Why Network Engineers Are Sick Of SDN – And What Vendors Can Do About It

Ethan Banks

November 24, 2012

2012's dominant networking buzzword has been SDN: software defined networking. Packet Pushers has talked about SDN quite a bit because it represents a re-thinking of the way that the industry has built networks for the last two decades. SDN as a concept is both technically interesting and creatively inspiring, giving many pause for thought and cogitation along the lines of "what if."

However, far more network engineers are sick of hearing about SDN than are interested in it. Rather than thinking "what if", they are thinking "who cares?" The "who cares about SDN" attitude has been related to this Packet Pusher from all aspects of our social media community. Tweets, e-mails, blog comments, IRC chatter, and forum posts have seen a recurring theme of "shut up about SDN already and let's talk about something that really matters." In fairness, I empathize. I feel invective thoughts about SDN rising up in my own mind from time to time, and I'm a believer in the SDN idea.

Why are network engineers sick of hearing about SDN?

Engineers have a number of problems running their networks today. From a practical

Challenge vs. Opportunity

There are 2+1 acquisitions you may be insteresting…

Two acquisitions

- 2012: VMware buy Nicria (founded in 2007) for $1.26B
- 2019: Intel acquired Barefoot Networks (founded in 2013)
  - Amount is undisclosed. But Barefoot is raised more than $1.5B

Nick McKeown joined Intel in 2019 with the acquisition of Barefoot Networks

Intel Senior Vice President
General Manager, Network and Edge Group

A recent acquisition

- 2022.05, Broadcom acquires VMware, >$61B
- Broadcom is an American fabless semiconductor company that makes products for the wireless and broadband communication industry.
  - Major customers: Apple, HP, IBM, Dell, Asus, Lenovo, Cisco, …
- Products:
  - NIC
  - Trident+ASIC (e.g., Cisco Nexus 9000, EdgeCore AS6700)
  - WiFi chipsets

Broadcom produces the SoC for Raspberry Pi

Outline

- Why we need SDN
- Data and Control Plane
- SDN Controller
- OpenFlow
- SDN Next: P4?

Why we need SDN?

Question:

What make today's Internet a great success?

The Internet: A Remarkable Story

David Clark,
"Internet Design Philosophy",
SIGCOMM 1988

- Tremendous success: From research experiment to global infrastructure
- Openness
  - IP, "narrow waist"
- Brilliance of under-specifying
  - Network: best-effort packet delivery
  - Programmable hosts: arbitrary applications
- Enables innovation
  - Apps: Web, P2P, VoIP, social networks, …
  - Links: Ethernet, fiber optics, WiFi, cellular, …

Question:

Internet is so successful, why IPv6 promotion is so difficult?

Network Devices

Router/Switch

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

Feature | Feature

OS

Custom Hardware

Million of lines of source code

>8,000 RFCs

Billions of gates

Bloated

Power Hungry

- Vertically integrated, complex, closed, proprietary
- Networking industry with "mainframe" mind-set

Inside the Internet: A Different Story…

- Closed equipment
  - – Software bundled with hardware
  - – Vendor-specific interfaces
- Over specified
  - – Slow protocol standardization
- Few people can innovate
  - – Equipment vendors write the code
  - – Long delays to introduce new features

Research Stagnation (研究停滞)

- Lots of innovation in other areas, e.g., OS
  - Microsoft Windows: every 2~3 years
  - Linux kernel: 2~3 months
- Networks are largely the same as years ago
  - IPv4: RFC 791, 1981年
  - IPv6: RFC 2460, 1998年 (minor modification in RFC 8200, 2017)
  - DNS: RFC 1034/1035, 1987年
  - ARP: RFC 826, 1982年
  - …
- Rate of change of the network seems slower
  - Need better tools and abstractions to demonstrate and deploy

Computer System Evolution

Vertically integrated
Closed, proprietary
Slow innovation
Small industry

App

Open Interface

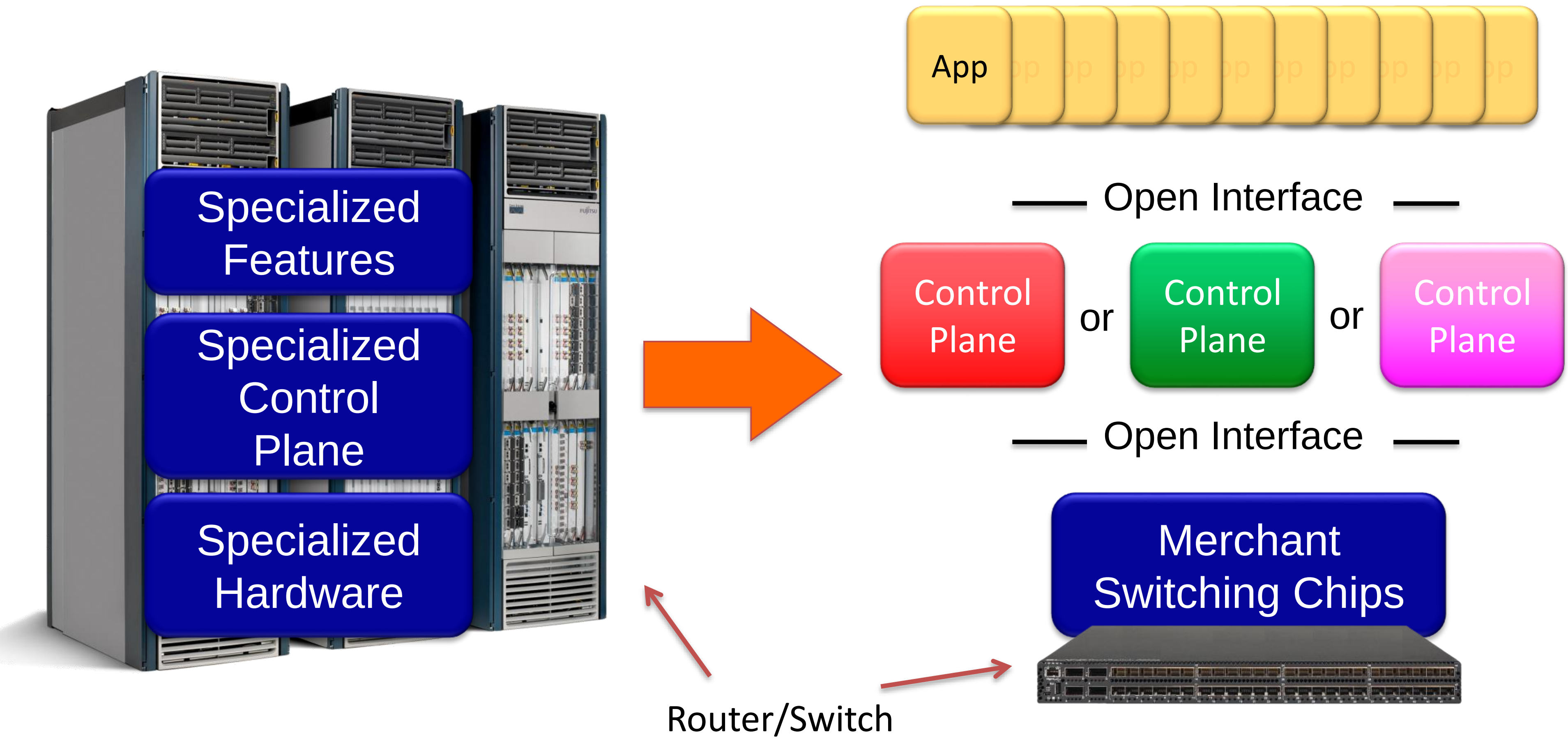
Windows (OS) or Linux or Mac OS

Open Interface

Microprocessor

Horizontal
Open interfaces
Rapid innovation
Huge industry

Network Devices



Vertically integrated
Closed, proprietary
Slow innovation

Horizontal
Open interfaces
Rapid innovation

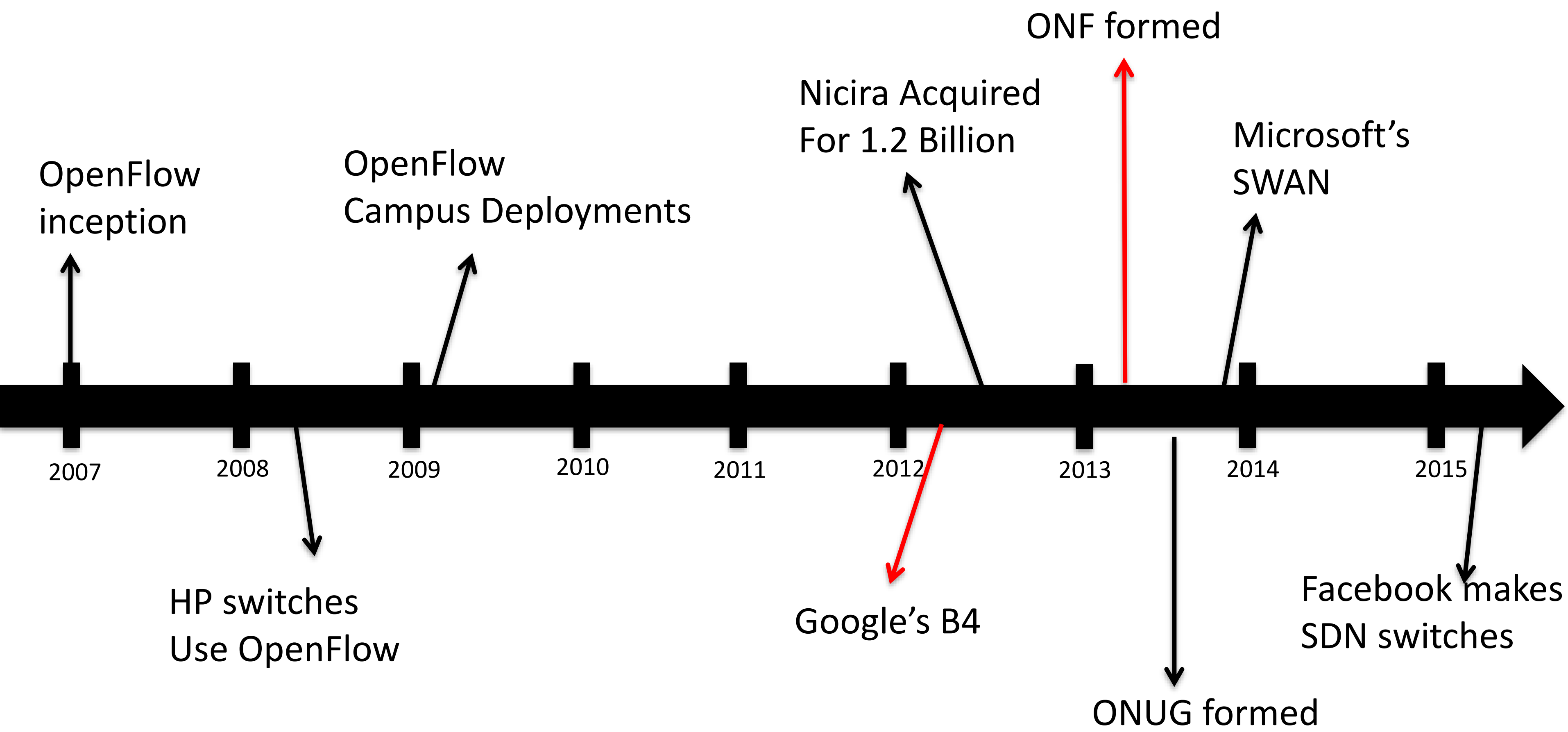
SDN Definition

SDN is a framework to allow network administrators to automatically and dynamically manage and control a large number of network devices, services, topology, traffic paths, and packet handling (QoS) policies using high-level languages and APIs.

--- Raj Jain

Network programmability

SDN Timeline



Researchers in SDN

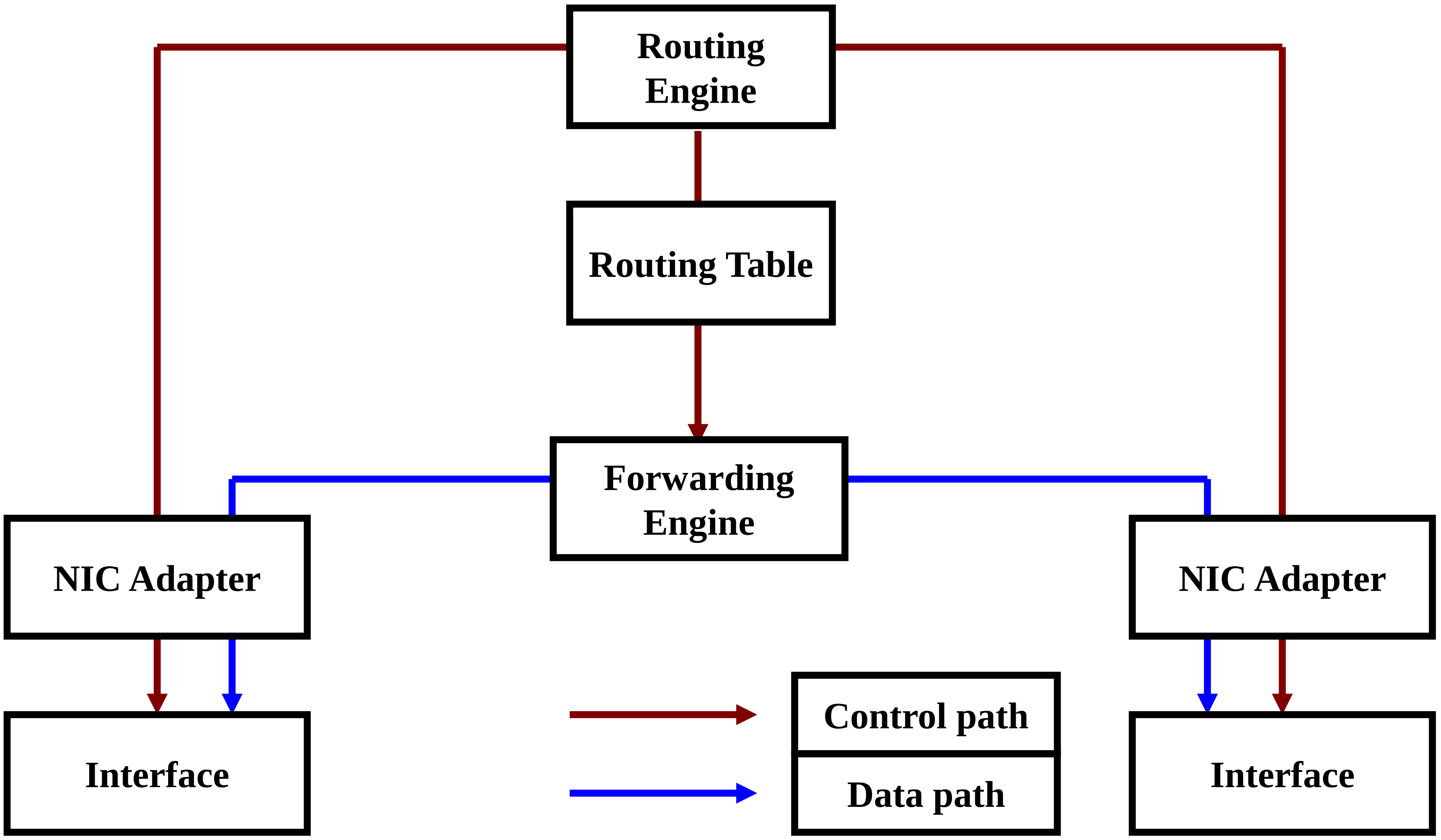
Jennifer Rexford

Nick Mckeown

Scott Shenker

Data Plane and Control Plane
数据平面和控制平面

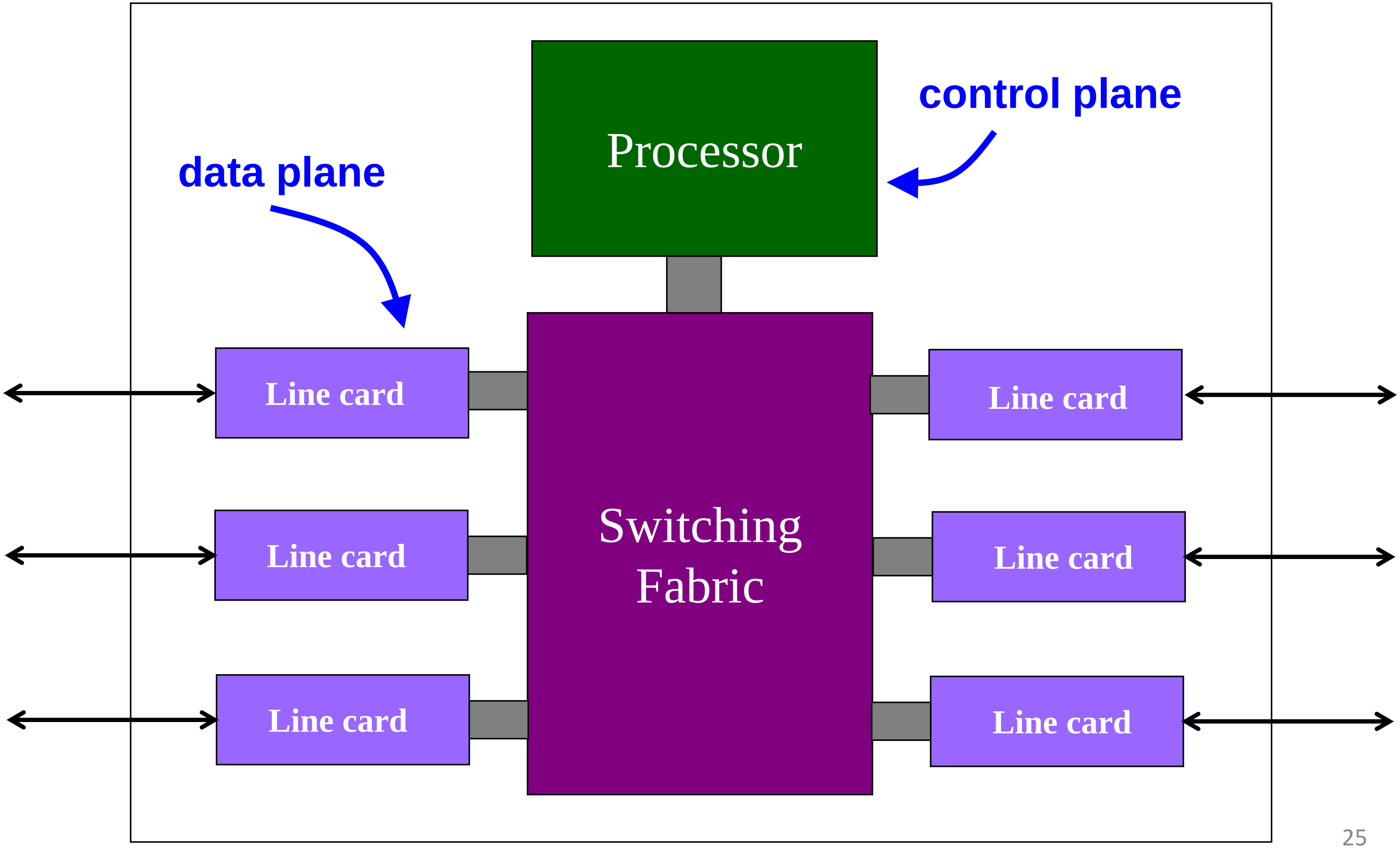
Logical Architecture of a Router/L3 Switch



What are the control and data planes?

- Control Plane: Logic for controlling forwarding behavior
  - Examples: routing protocols, network configuration.
- Data Plane: Forward traffic according to control plane logic
  - Examples: IP forwarding, Layer 2 switching

Data and Control Planes



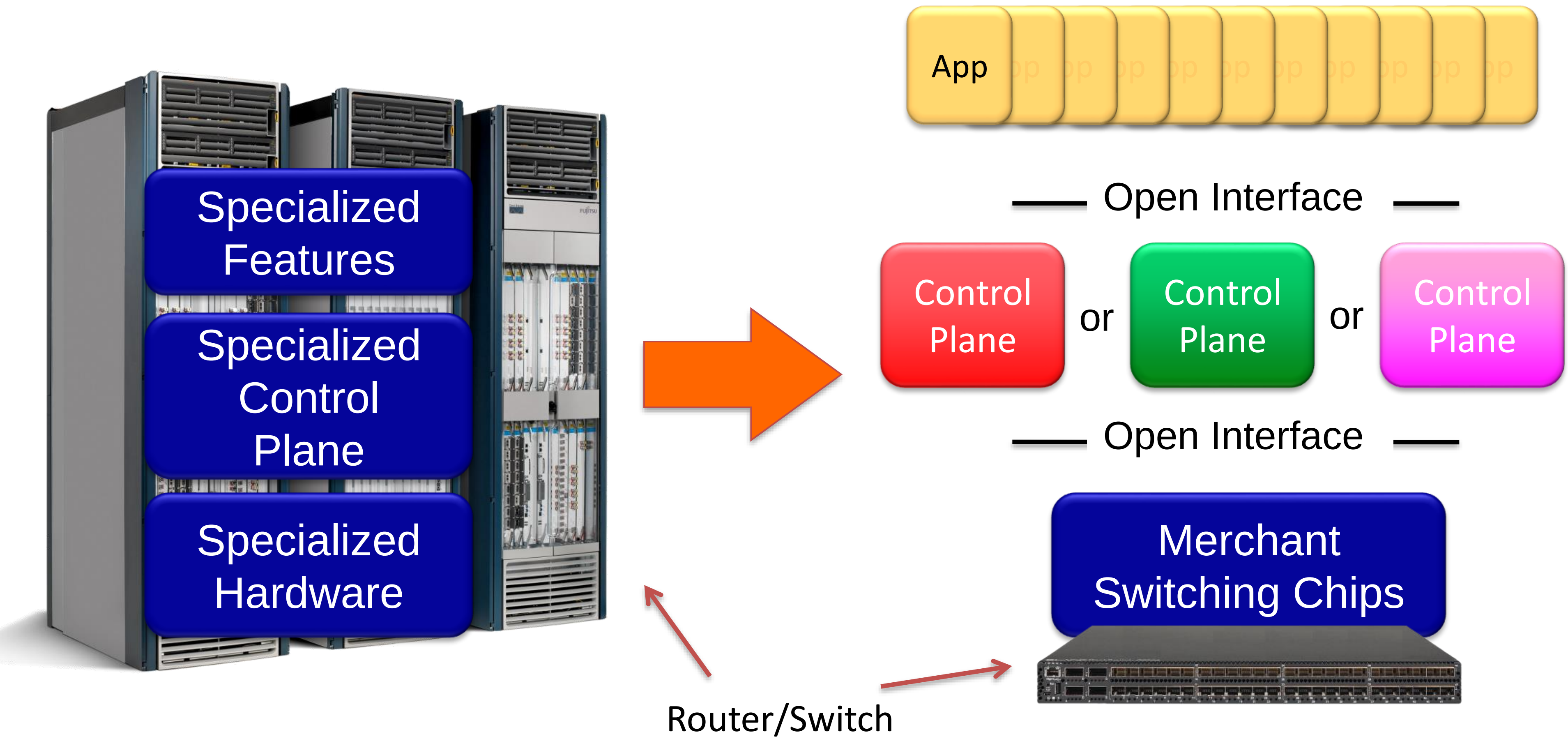
Timescales of different planes

	Data	Control
Time-scale	Packet (nsec)	Event (10 ms to sec)
Tasks	Forwarding, buffering, filtering, scheduling	Routing, circuit set-up
Location	Line-card hardware	Router software

Data and Control Plane Separation
数据平面与控制平面分离

Main approach of existing SDN solution

Network Devices



Vertically integrated
Closed, proprietary
Slow innovation

Horizontal
Open interfaces
Rapid innovation

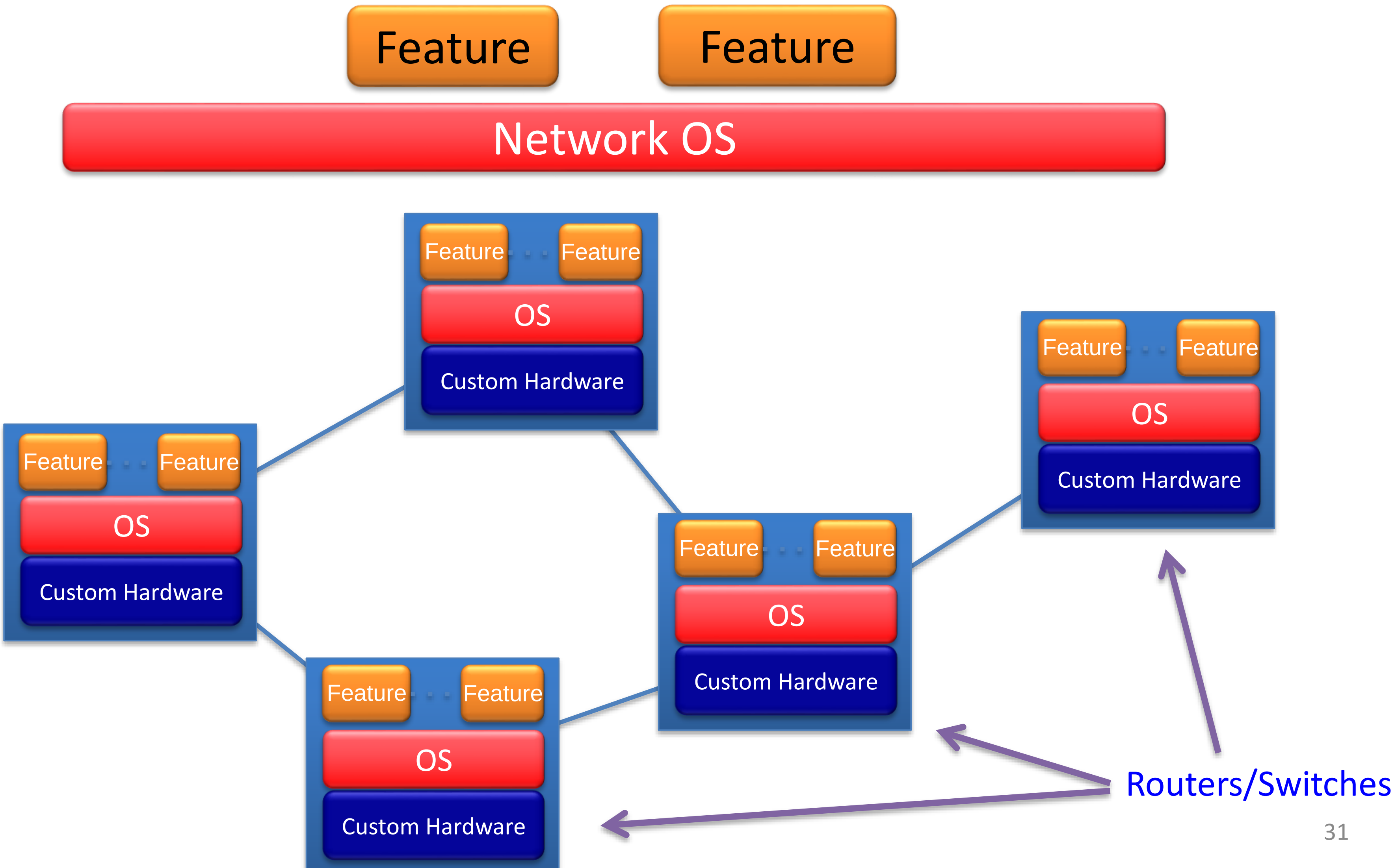
Application, Control and Data Plane

- Application plane
  - Applications/services
  - Examples: network topology discovery, network provisioning, path reservation, etc.
- Control Plane
  - Orchestrate network resources, and thereby offer better abstractions to application plane
  - Example: defines the traffic routing and maintain network topology
- Data Plane
  - Physically handles the traffic based on the configurations supplied from the Control Plane, e.g., packet forwarding, statistics and state collection

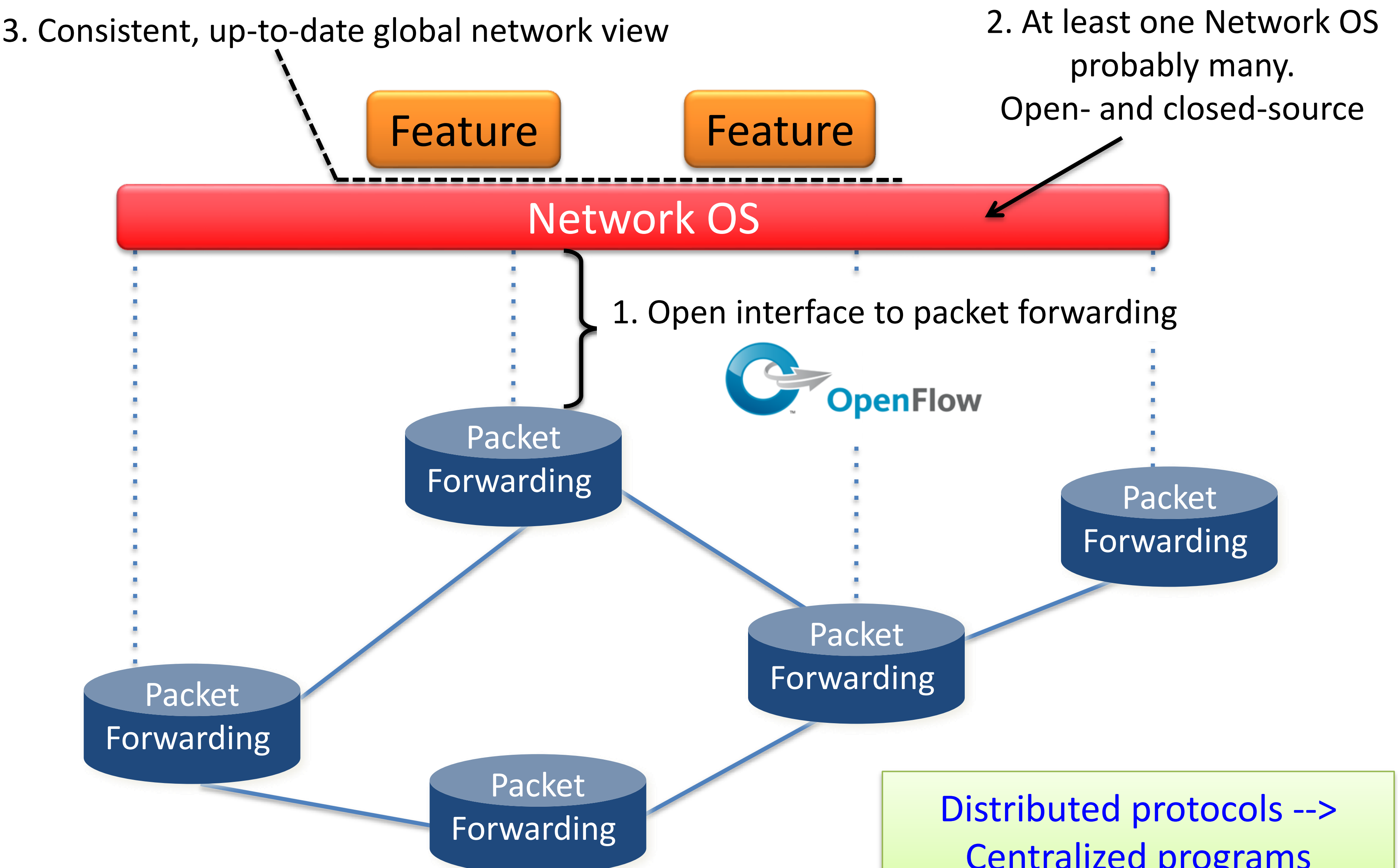
Why separate the control and data planes?

- Independent evolution and development
  - Control software and applications (e.g., routing) can evolve independently of the hardware
- Control from high-level software program
  - Control behavior using higher-level programs
  - Debug/check behavior more easily

Software Defined Network (SDN)

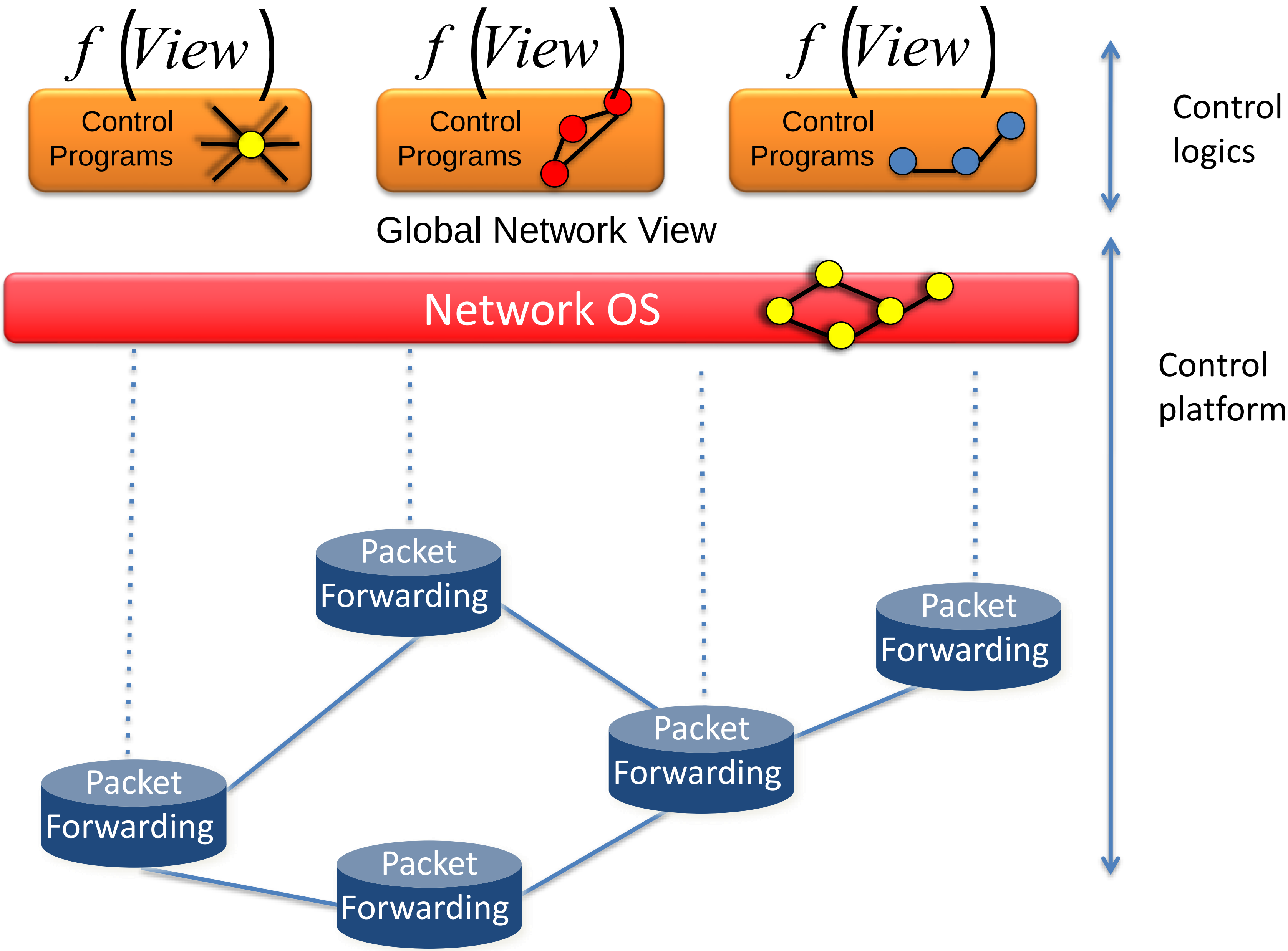


Software Defined Network (SDN)

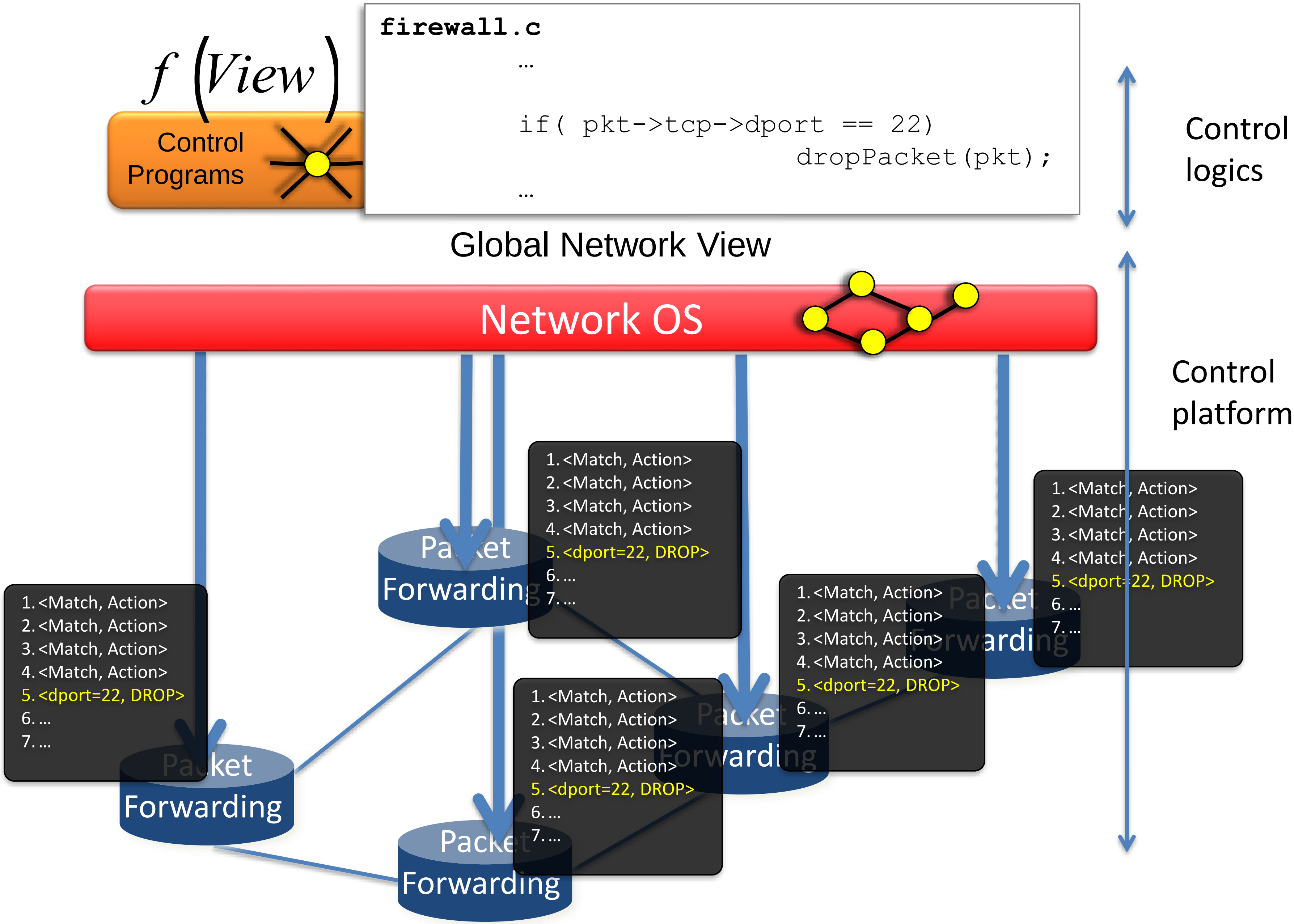


Distributed protocols -->
Centralized programs

Software Defined Network (SDN)



Software Defined Network (SDN)



We have achieved modularity!

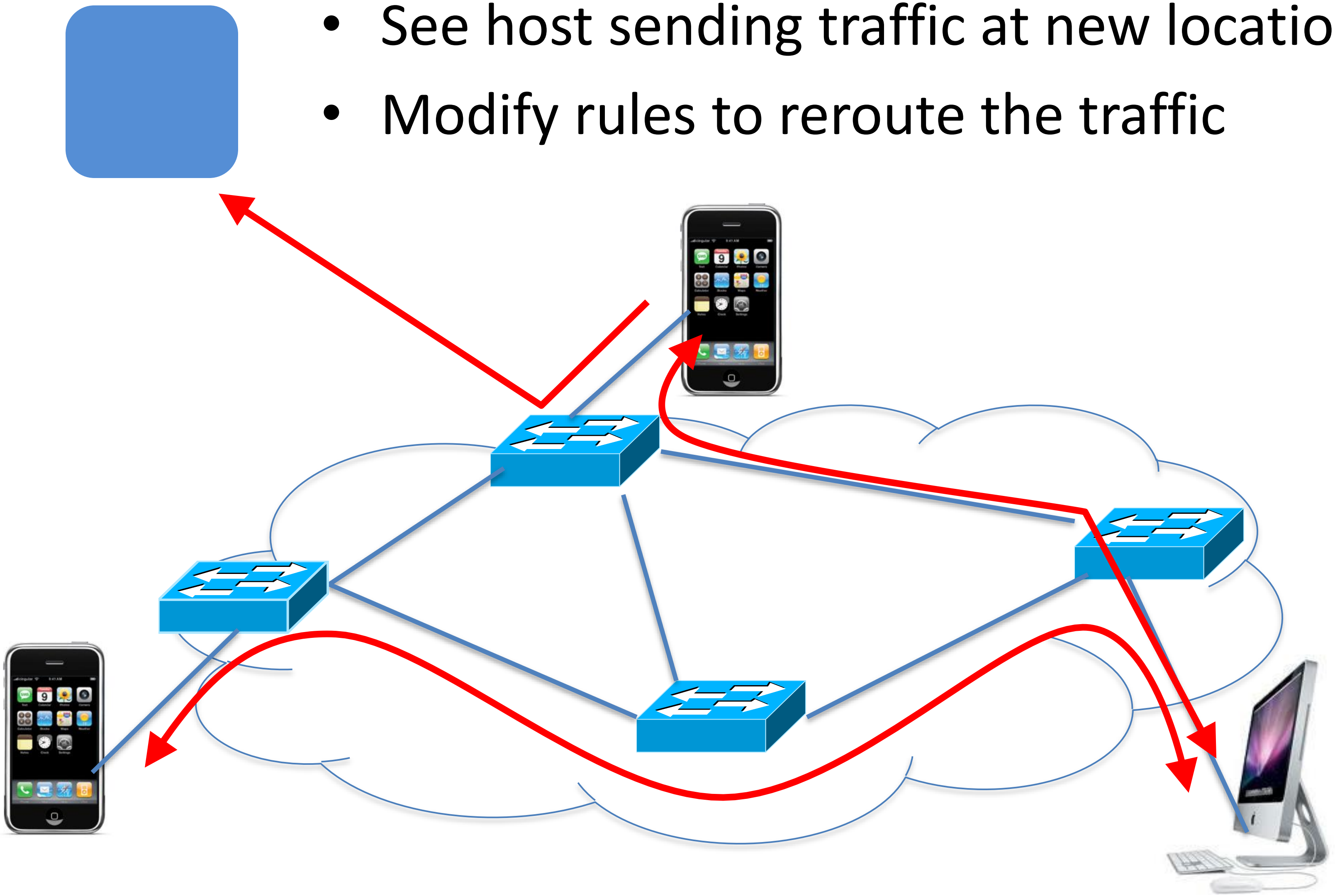
- Modularity (模块化) enables independent innovation
  - Gives rise to a thriving ecosystem
- Innovation is the true value proposition of SDN
  - SDN doesn't allow you to do the impossible
  - It just allows you to do the possible much more easily

Opportunities: where separation helps

- Cloud DC: VM migration, Layer 2 routing
- Routing: More control over decision logic
- Enterprise networks: Security applications
- Research networks: Coexistence with production

Example: Seamless Mobility

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

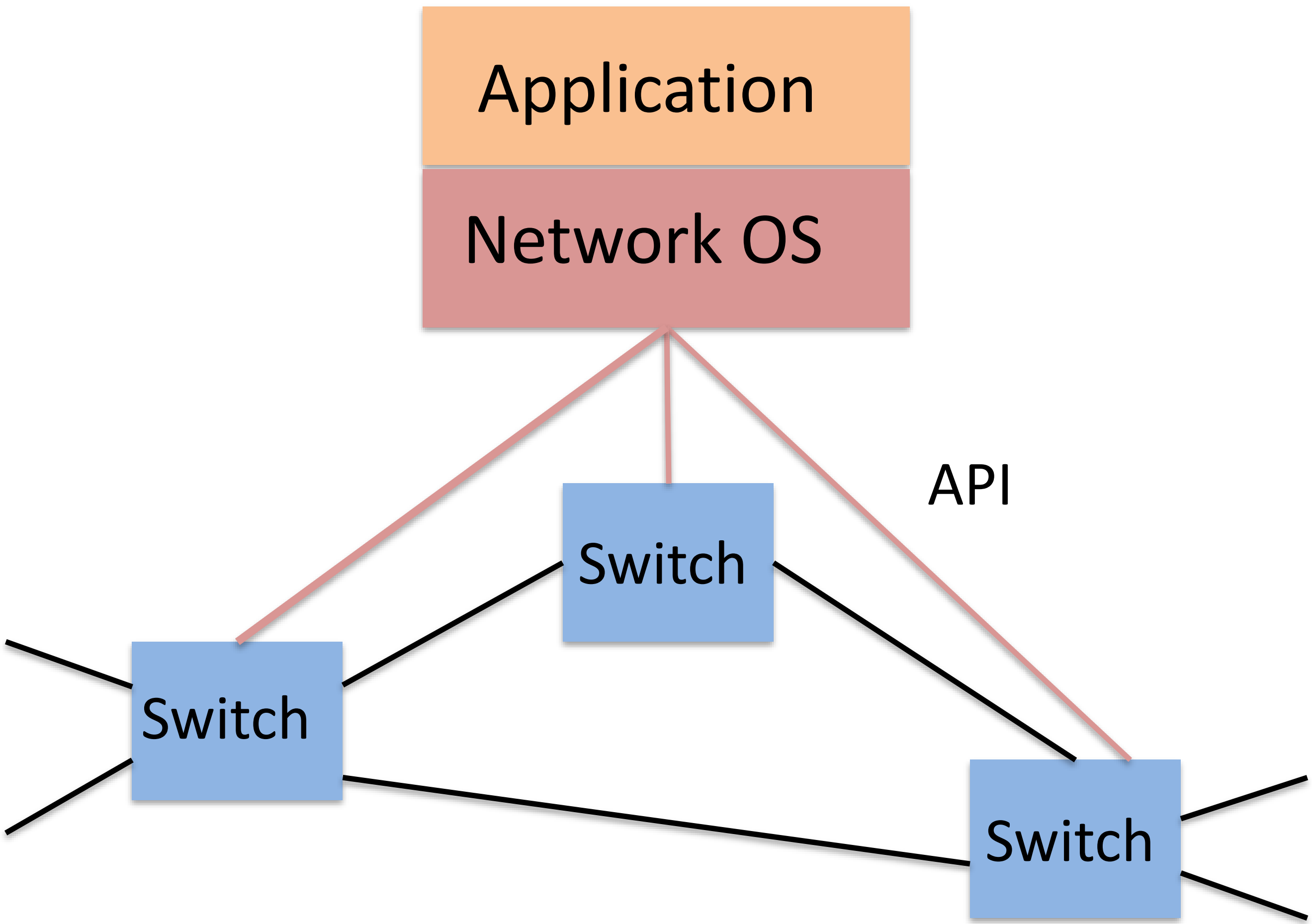


Example SDN Applications

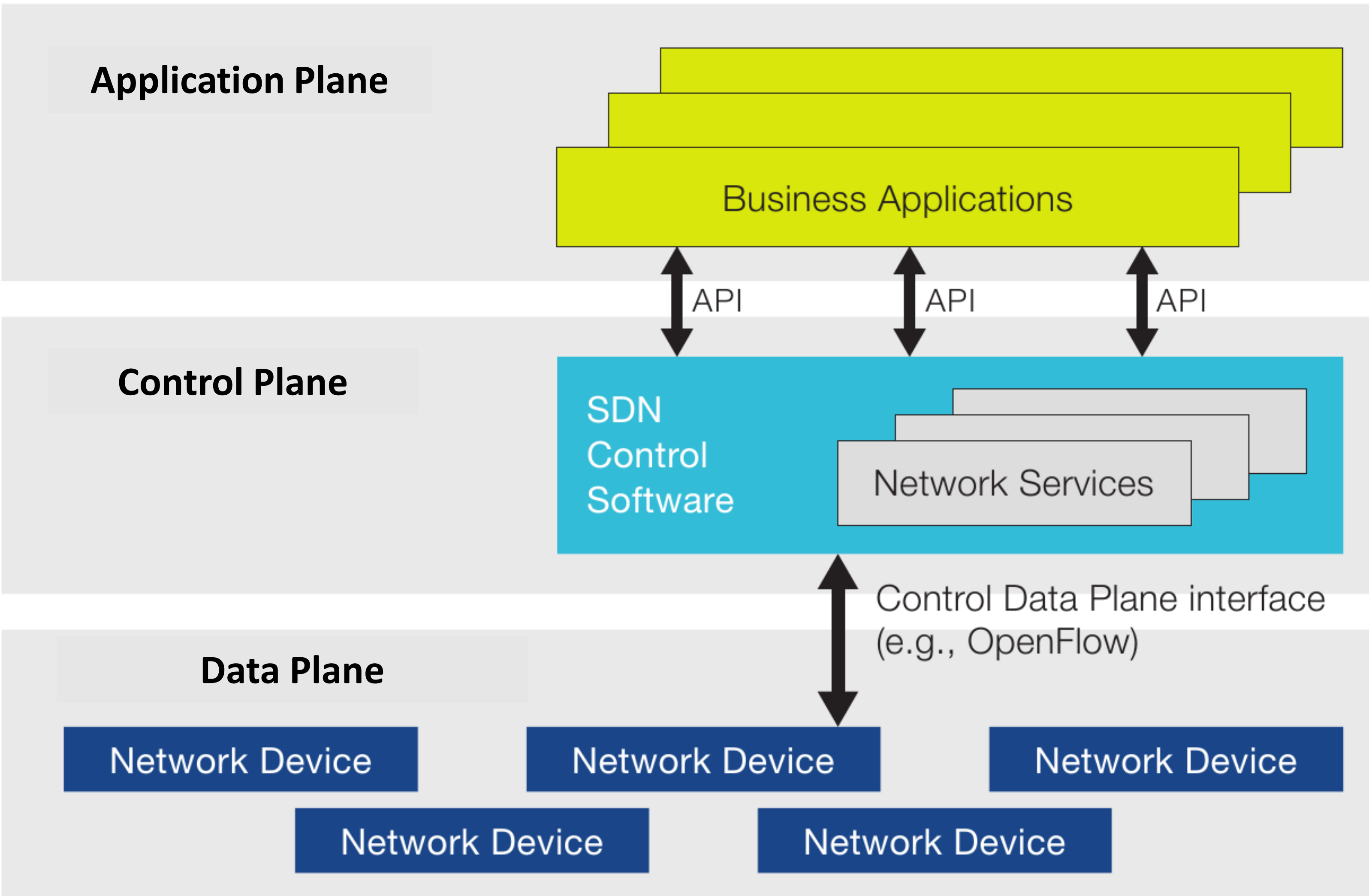
- Seamless mobility and migration
- Server load balancing
- Dynamic access control
- Using multiple wireless access points
- Energy-efficient networking
- Adaptive traffic monitoring
- Denial-of-Service attack detection
- Network virtualization

SDN控制器
SDN Controller

SDN Software Stack (Simple)



SDN Architecture



SDN Implementation

- Controller performs all of the functions
  – such as routing, policy implementation, and security
- This is the SDN control plane
- It is deployed in one or more SDN servers which are the controllers

SDN Implementation

- Controller defines and controls the data flows that occur in the SDN data plane
- Each flow through the network must first get permission from the controller
  - Controller verifies that the communication is permissible by the network policy

SDN Implementation

- If the controller allows a flow
  - it computes a route for the flow to take
  - adds an entry for that flow in each of the switches along the path
- Switches simply manage flow tables
  - entries can be populated only by the controller
  - all complex functions subsumed by the controller
- Communication between the controller and the switches uses a standardized protocol and API

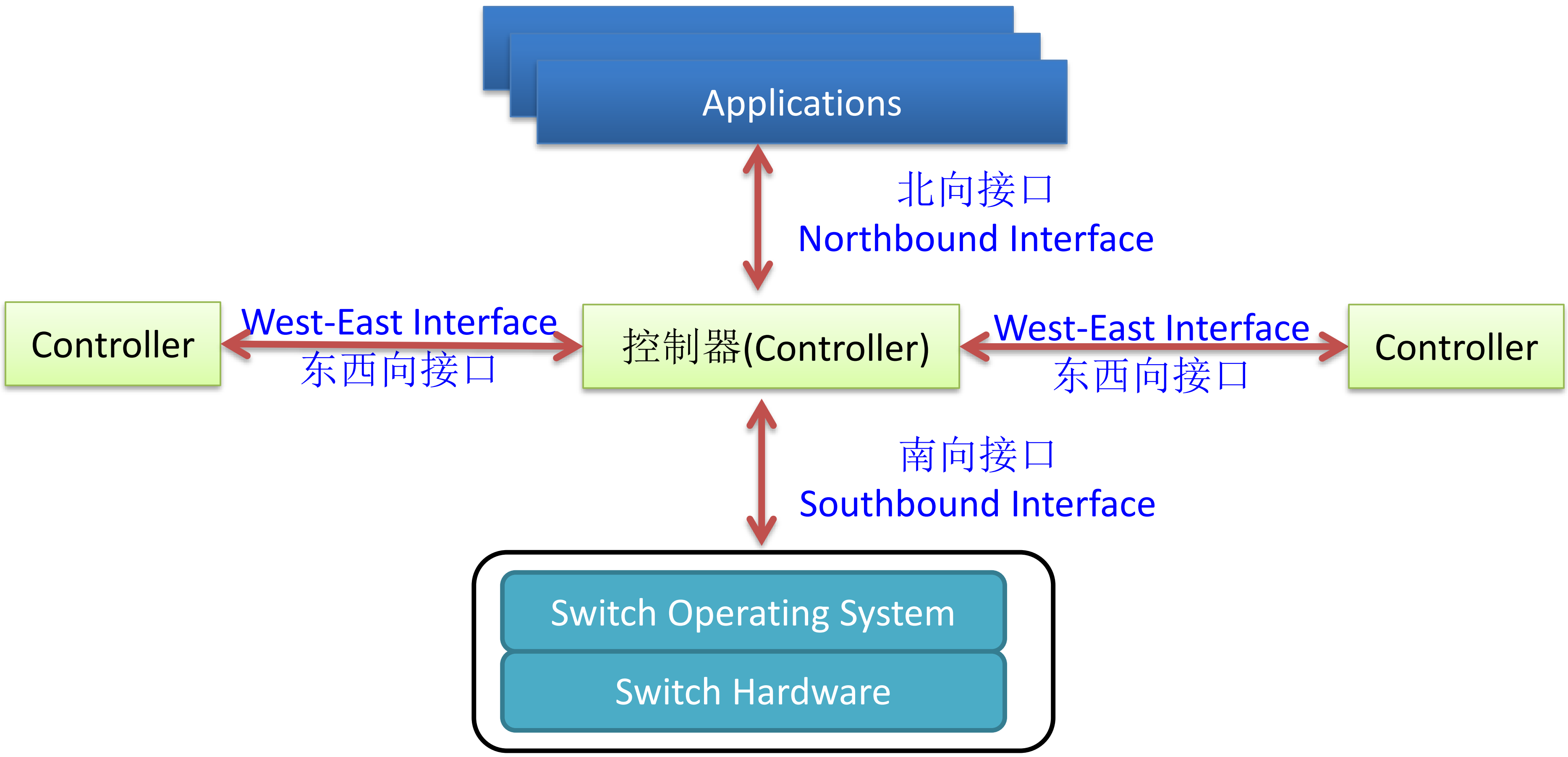
SDN Controller

- What controller do?
  - Provides a programmatic interface to the network, where applications can be written to perform control and management tasks, and offer new functionalities
  - This view assumes that the control is centralized and applications are written as if the network is a single system

SDN Controller

- While this simplifies policy enforcement and management tasks, the bindings must be closely maintained between the control and the network forwarding elements
- A controller that strives to act as a network operating system must implement at least two interfaces

SDN Controller Interfaces



- Southbound API: decouples the switch hardware from control function
  - Data plane from control plane
- Switch Operating System: exposes switch hardware primitives

Southbound Interface

- Allows switches to communicate with the controller
- The southbound interface is well defined
- It functions as a de facto standard for switch control
- We will discuss this later using an example of OpenFlow

SDN Controller

- Extract information about the underlying network
- Expose programmable API to network control and high-level policy applications and services
- Control an aspect of the network behavior or policy

Northbound Interface

- Unlike controller-switch communication, there is no currently accepted standard for the northbound interface
- They are more likely to be implemented on an ad-hoc basis for particular applications

OpenFlow

SDN Implementation

- To turn the SDN concept into practical, two requirements must be met:
  - A common logical architecture in all switches, routers, and other network devices to be managed by an SDN controller
  - A standard, secure protocol between the SDN controller and the network device

SDN Implementation: OpenFlow

- Both of these requirements are addressed by OpenFlow
  - a protocol between SDN controllers and network devices
  - a specification of the logical structure of the network switch functions

History of OpenFlow

OpenFlow Prototype

- Ethane Project (斯坦福大学项目)
  - By Martin Casado, Nick McKeown et al., 2004
  - Coupled simple flow based switches managed by a central controller which controlled the flows
- SIGCOMM 2008
  - “OpenFlow: enabling innovation in campus”, ACM Communications Review
- Nicira, 2007
  - Focused on SDN and network virtualization
  - Startup by Martin Casado, Nick McKeown, Scott Shenker

OpenFlow Progression

- OF v1.0, Dec. 2009, "Into the Campus"
  - OpenFlow is an open source research project of Stanford University and the UC Berkeley
- ONF - Open Network Foundation, March 2011
  - standard bearer for SDN
  - Led by its operator partners AT&T, China Unicom, Comcast, Deutsche Telekom, Google, and Turk Telekom.

OpenFlow Progression

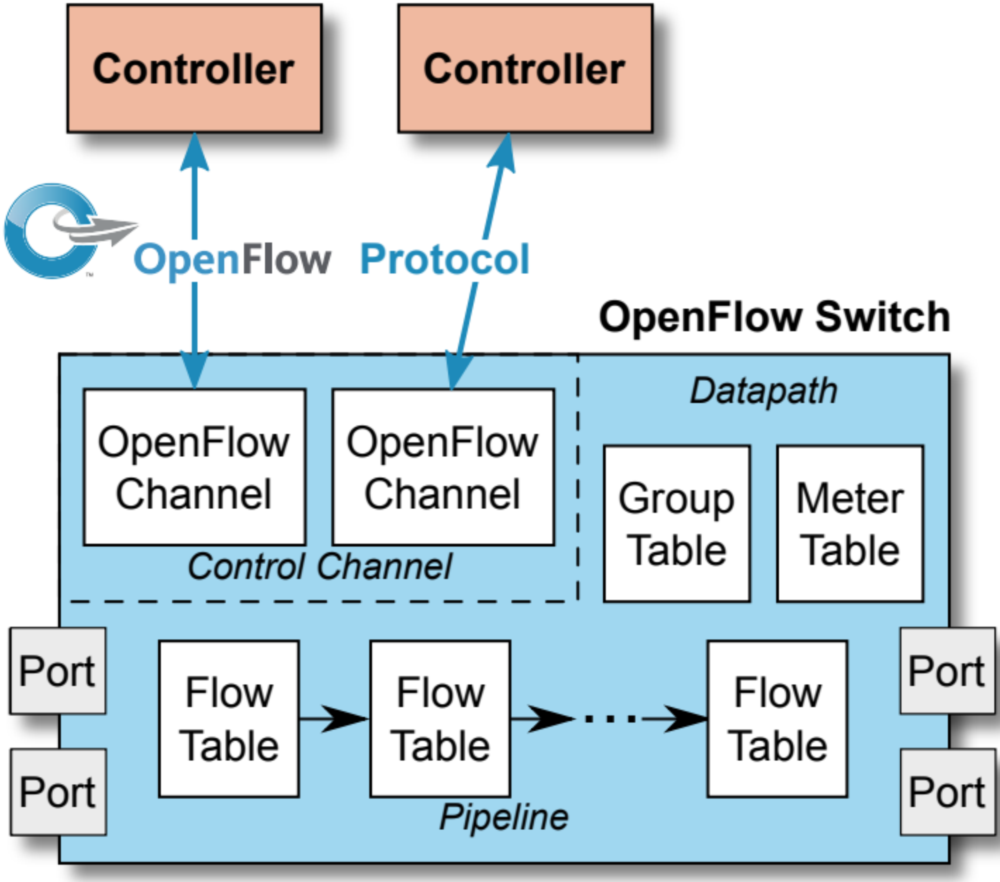
- OF v1.1: released March 1 2011: “Into the WAN”
  - multiple tables: leverage additional tables
  - tags and tunnels: MPLS, VLAN, virtual ports
  - multipath forwarding: ECMP, groups
- OF v1.2: approved Dec 8 2011: “Extensible Protocol”
  - extensible match
  - extensible actions
  - IPv6
  - multiple controllers
- Nicira acquired by VMware for $1.26 billion, July 2012

OpenFlow Progression

- OF v1.3, Jun. 2012
  - multiple tables: leverage additional tables
  - tags and tunnels
  - multipath forwarding
  - per flow meters
- OF v1.4, Oct. 2013
  - Synchronized Table
- OF v1.5, Jan. 2015
  - Egress Table

OpenFlow

- OpenFlow defines
  - OpenFlow Switch
  - Flow Table
  - OpenFlow Channel
  - OpenFlow Protocol



Communicate with other protocol stack: physical, logic or reserved ports

OpenFlow Channel

- OpenFlow channel is the interface that connects each switch to a controller
  - Configure and manage the switch.
  - Receive events from the switch.
  - Send packets out the switch
- OpenFlow channel is usually encrypted by using TLS.
- It can also be run directly over TCP.

OpenFlow protocol

Supports following message types:

- Controller-to-switch messages
  - Initiated by the controller
  - Used to directly manage or inspect the state of the switch.
  - Might or might not require a response from the switch.
- Asynchronous messages
  - Switches send asynchronous messages to controllers to inform a packet arrival or switch state change.
- Symmetric messages
  - Sent without solicitation, in either direction
  - Example: Hello, Echo, Error messages

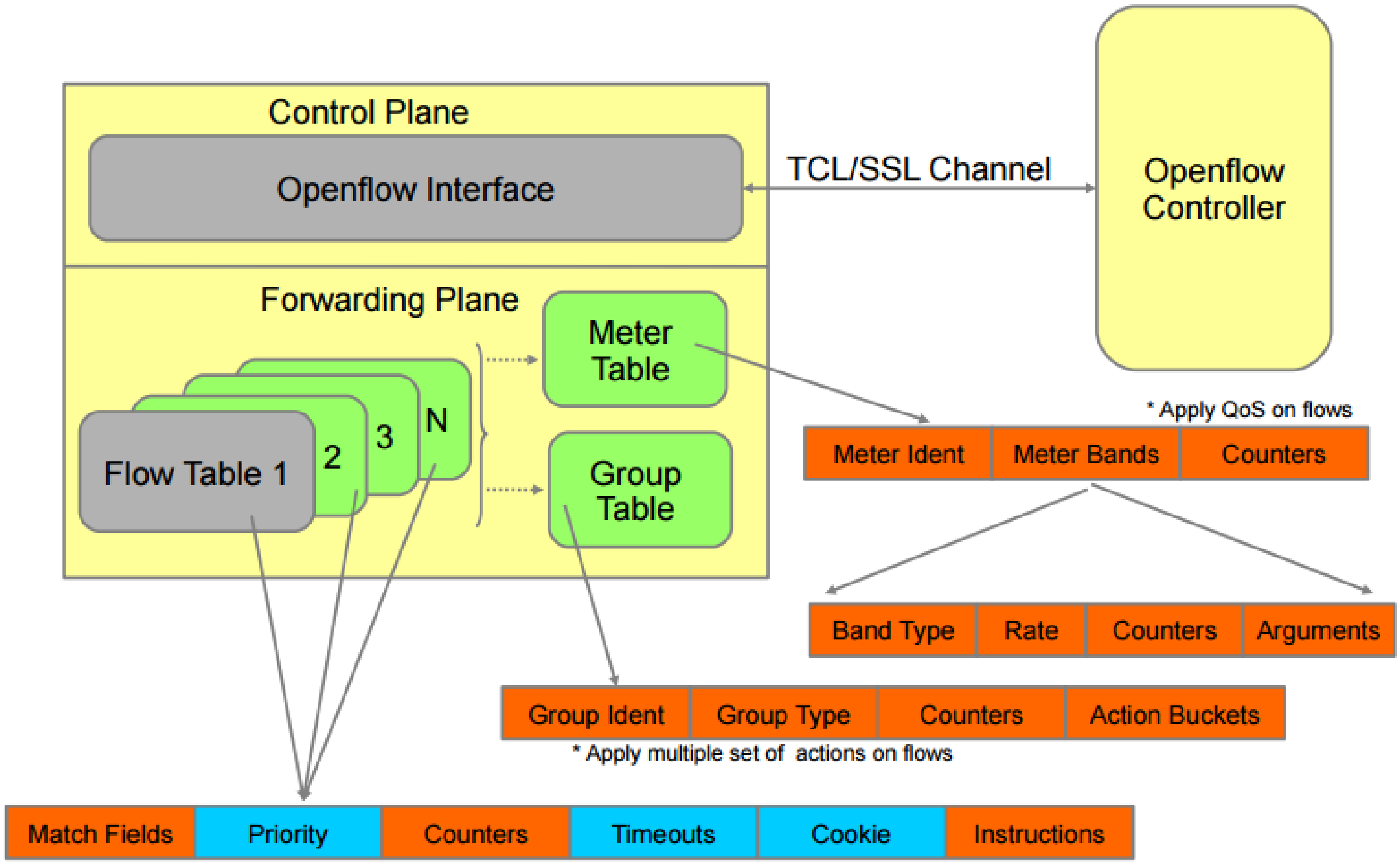
Tables Used by OpenFlow

- OpenFlow uses three types of tables in the logical switch
  - Flow Table
    - Directs incoming packets to the proper flow that then specifies the functions that are to be performed on the packets
    - There may be multiple flow tables that process in pipeline fashion

Tables Used by OpenFlow

- Group Table
  - This table may trigger a variety of actions that affect one or more flows
- Meter Table
  - The metering table controls any performance adjustments that may apply

OF 1.3.1: Switch Hardware



Flow Table

- The main table used is the flow table
- Every packet that comes into a switch goes through at least one flow table, and perhaps several

Flow Table

- Each flow table in the switch holds a set of flow entries that consists of
  - Header fields or match fields, with information found in the packet header, ingress port, metadata, used to match incoming packets
  - Counters, used to collect statistics for the particular flow, such as number of received packets, number of bytes, and duration of the flow
  - A set of instructions or actions to be applied after a match that dictates how to handle matching packets, such as forwarding a packet out to a specified port

Flow Table Entry

Header Fields	Counters	Actions	Priority

Ingress Port
Ethernet Source Addr
Ethernet Dest Addr
Ethernet Type
VLAN id
VLAN Priority
IP Source Addr
IP Dest Addr
IP Protocol
IP ToS
ICMP type
ICMP code

Per Flow Counters
Received Packets
Received Bytes
Duration seconds
Duration nanosecconds

Forward
(All, Controller, Local,
Table, IN_port, Port#
Normal, Flood)

Enqueue
Drop
Modify-Field

Flow Table Example

Header Fields	Counters	Actions	Priority
If ingress port == 2		Drop packet	32768
if IP_addr == 129.79.1.1		re-write to 10.0.1.1, forward port 3	32768
if Eth Addr == 00:45:23		add VLAN id 110, forward port 2	32768
if ingress port == 4		forward port 5, 6	32768
if Eth Type == ARP		forward CONTROLLER	32768
If ingress port == 2 && Eth Type == ARP		forward NORMAL	40000

OF 1.3.1 : Flow Entry

Match Fields	Priority	Counters	Timeouts	Cookie	Instructions

- Port
  - Ingress Port
  - Physical Input Port
- Metadata
- Header Fields
  - Ether Src
  - Ether dst
  - Ether type
  - VLAN Id
  - VLAN prio
  - IP Src
  - IP Dst
  - IP Proto
  - IP TOS (DSCP)
  - IP TOS (ECN)
  - TCP Src Port / TCP Dst Port
  - UDP Src Port / UDP Dst Port
  - SCTP Src Port / SCTP Dst Port
  - ICMP Type / ICMP Code
  - Arp Opcode
  - Arp Src IP / Arp Dst IP
  - Arp Ether Src / Arp Ether Dst
  - IPv6 Src / IPv6 Dst
  - IPv6 Flow Label
  - ICMPv6 Type / ICMPv6 Code
  - IPv6 ND Dst address
  - IPv6 ND Src Link Layer
  - IPv6 ND Dst Link Layer
  - MPLS Label
  - MPLS TC
  - MPLS BoS bit
  - PBB I-SID
  - Tunnel ID (Metadata)
  - Ipv6 Extension Header

- Per Flow Table
  - Active Entries
  - Packet Lookups
  - Packet Matches
- Per Flow
  - Received Packets
  - Received Bytes
  - Duration (Seconds)
  - Duration (Nano Secs)
- Per Port
  - Received Packets
  - Transmitted Packets
  - Received Bytes
  - Transmitted Bytes
  - Receive Drops
  - Transmit Drops
  - Receive Errors
  - Transmit Errors
  - Receive Frame Errors
  - Receive Overrun Errors
  - Receive CRC Errors
  - Collisions
  - Duration (Seconds)
  - Duration (Nano Sec)
- Per Queue
  - Transmit Packets
  - Transmit Bytes
  - Transmit Overrun Errors
  - Duration (Seconds)
  - Duration (Nano Secs)

- Meter <meter_id> – Direct packet to specified meter
- Apply-Actions <action(s)> - Apply action(s) immediately
- Clear-Actions – Clear all actions in action set
- Write-Metadata <metadata / mask > - Write masked metadata to metadata field
- Goto-Table <next-table-id> - Next table in processing

- Actions:
  - Output
    - ALL
    - CONTROLLER
    - LOCAL
    - TABLE
    - IN_PORT
    - NORMAL
    - FLOOD
  - Set-queue (Enqueue)
  - Drop
  - Group
  - Push Tag (VLAN, MPLS, PBB)
  - Pop Tag (VLAN, MPLS, PBB)
  - Set Field
    - All the Header Fields specified in Match Fields
  - Set TTL (MPLS TTL, IP TTL)
  - Decrement TTL (MPLS TTL, IP TTL)
  - Copy TTL outwards
  - Copy TTL inwards

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

First Packet Behavior

- The default behavior in OpenFlow 1.0 is to send all unknown packets to the controller
- This has been changed to drop the packet unless specific rules are there to send the packet to the controller

SDN Switch

- If the flow table look-up procedure does not result in a match, the action taken by the switch will depend on the instructions defined at the table-miss flow entry
- The flow table must contain a table-miss entry in order to handle table misses
- This particular entry specifies a set of actions to be performed when no match is found for an incoming packet

SDN Switch

- These actions include
  - dropping the packet
  - sending the packet out on all interfaces
  - or forwarding the packet to the controller over the secure channel

SDN Switch

- An SDN switch encapsulates and forwards the first packet of a flow to an SDN controller
- The controller decides what to do with a flow of packets
- If the packets are to be handled, a flow table is created for them

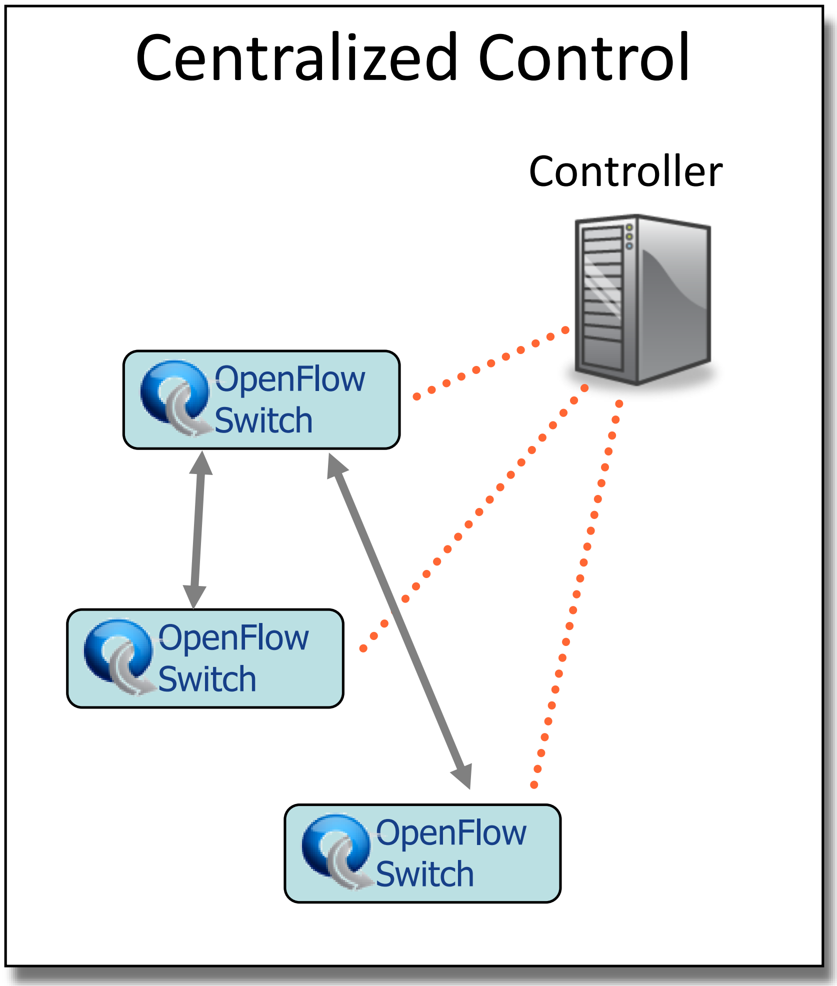
SDN Controller Operation

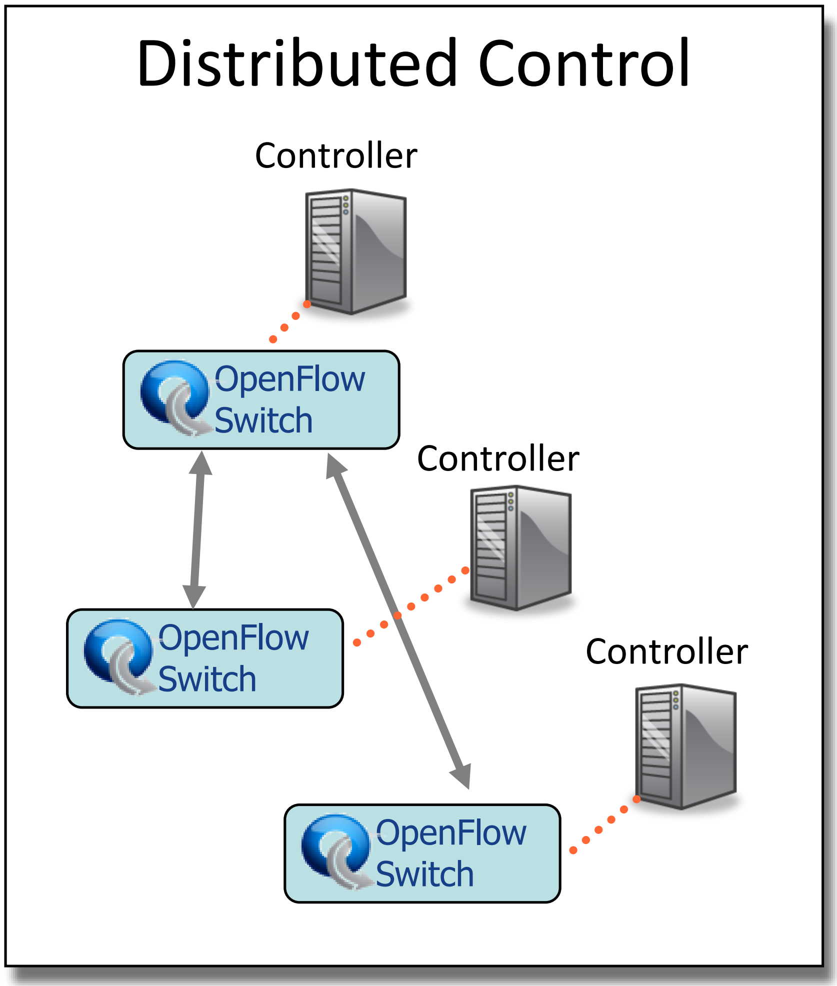
- The controller configures and manages the switch, receives events from the switch, and sends packets out to the switch through the interface
- Using OpenFlow protocol, a controller can add, update, or delete flow entries from the switch's flow table
- That can happen reactively - in response to a packet arrival - or proactively

OpenFlow Controllers and Hardwares

Centralized vs Distributed Control

Both models are possible with OpenFlow





Controllers

Name	Lang	Original Author	Notes
OpenFlow Reference	C	Stanford/Nicira	not designed for extensibility
NOX	**Python, C++**	Nicira	actively developed
Beacon	Java	David Erickson (Stanford)	runtime modular, web UI framework, regression test framework
Maestro	Java	Zheng Cai (Rice)	
Trema	Ruby, C	NEC	includes emulator, regression test framework
RouteFlow	?	CPqD (Brazil)	virtual IP routing as a service
POX	**Python**		
Floodlight	Java	BigSwitch, based on Beacon	

Switches

Vendor	Models	Virtualize?	Notes	Image
HP ProCurve	**5400zl, 6600, +**	1 OF instance per VLAN	-LACP, VLAN and STP processing before OF -Wildcard rules or non-IP pkts processed in s/w -Header rewriting in s/w -CPU protects mgmt during loop	
Pronto/ Pica8	3290, 3780, 3920, +	1 OF instance per switch	-No legacy protocols (like VLAN and STP) -Most actions processed in hardware -MAC header rewriting in h/w	

Name	Lang	Platform(s)	Original Author	Notes
OpenFlow Reference	C	Linux	Stanford/Nicira	not designed for extensibility
Open vSwitch	C/ Python	Linux/BSD?	Ben Pfaff/Nicira	In Linux kernel 3.3+
Indigo	C/Lua	Linux-based Hardware Switches	Dan Talayco/BigSwitch	Bare OpenFlow switch

Current SDN hardware

Juniper MX-series

NEC IP8800

WiMax (NEC)

HP Procurve 5400

Netgear 7324

PC Engines

Pronto 3240/3290

Ciena Coredirector

More coming soon...

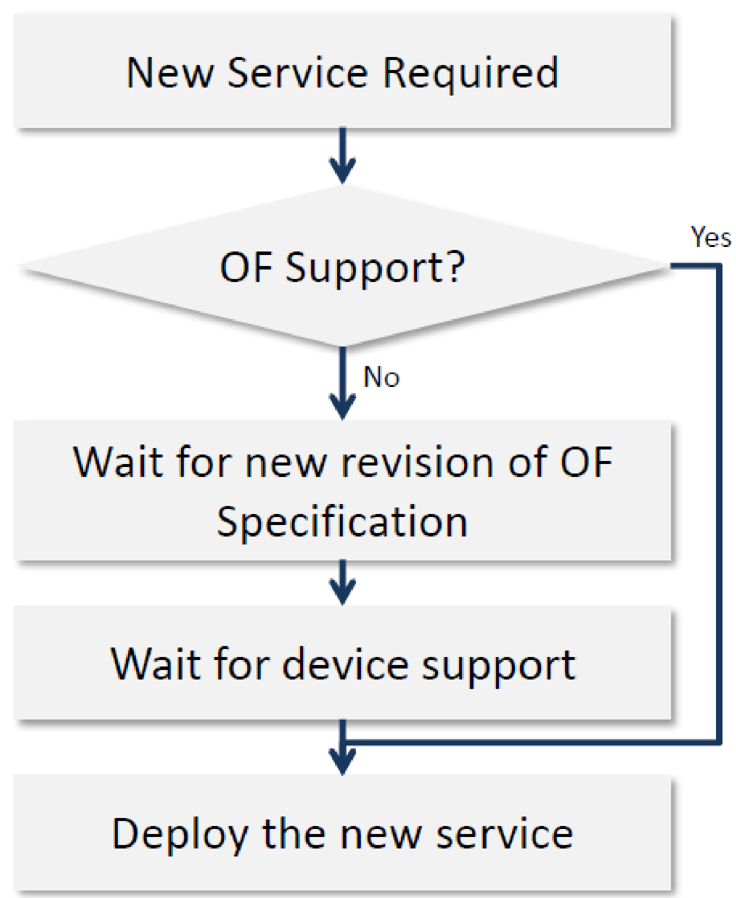
Off-class tutorials

- Mininet + OVS
- www.sdnlab.com

Evolution of Future Networking…

The Problem with OpenFlow 1.X

- New OF Specifications at high rate
- Hardware development cycle rate much lower
- Most OF Switches still run older OF versions
- OF table type pattern



Towards OpenFlow 2.0

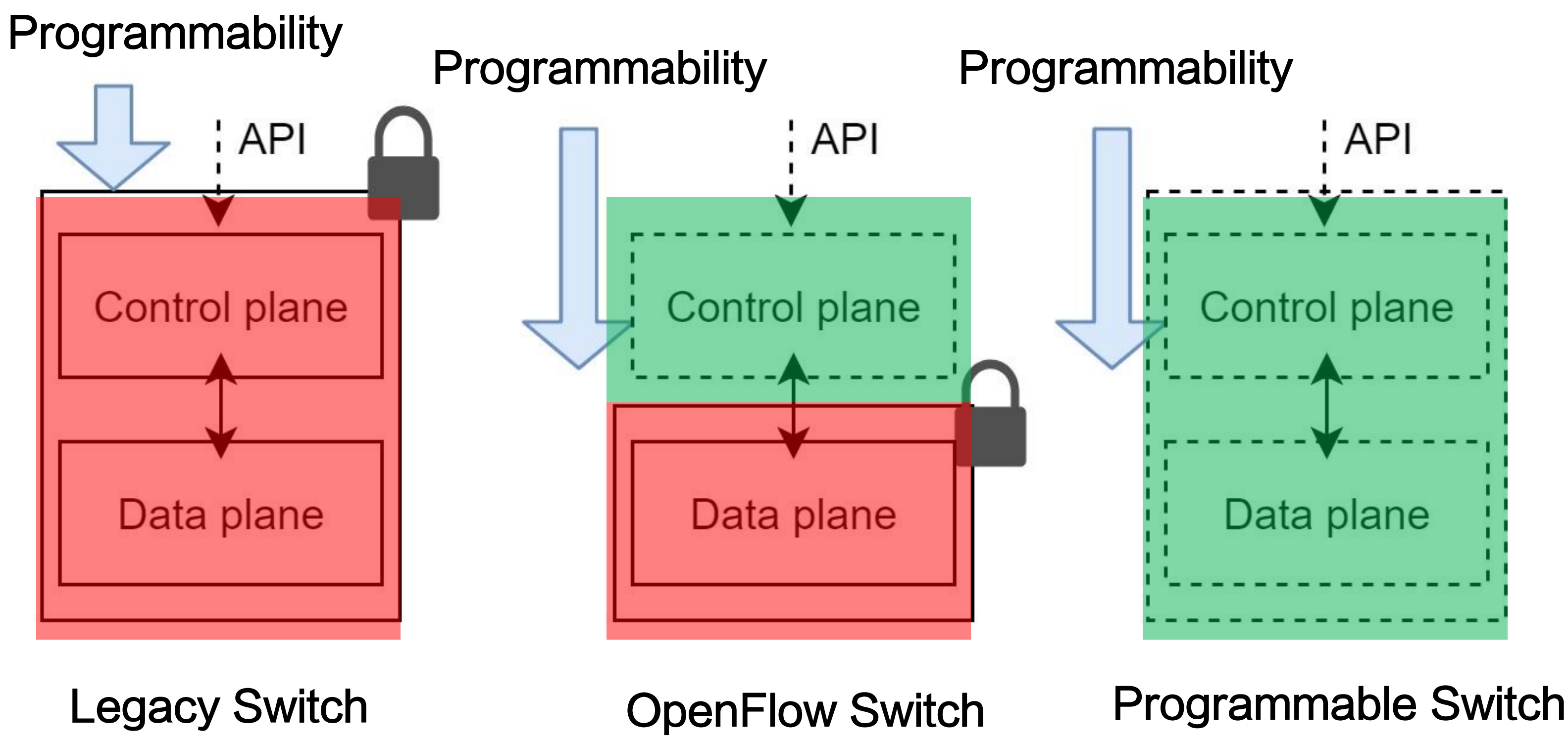
- New Services/Applications
  - Demand for new Matching Fields
  - Increasing Complexity of OF Instruction Set
- Trend to Programmable Switches
  - CPU/Software
  - Network Processors
  - FPGA
  - Flexible Match+Action ASICs

Protocol Independent Forwarding

- Protocol Independence (协议无关)
- Target Independence (目标设备无关)
- Reconfigurability (可重构性)
- Language Independence (编程语言无关)

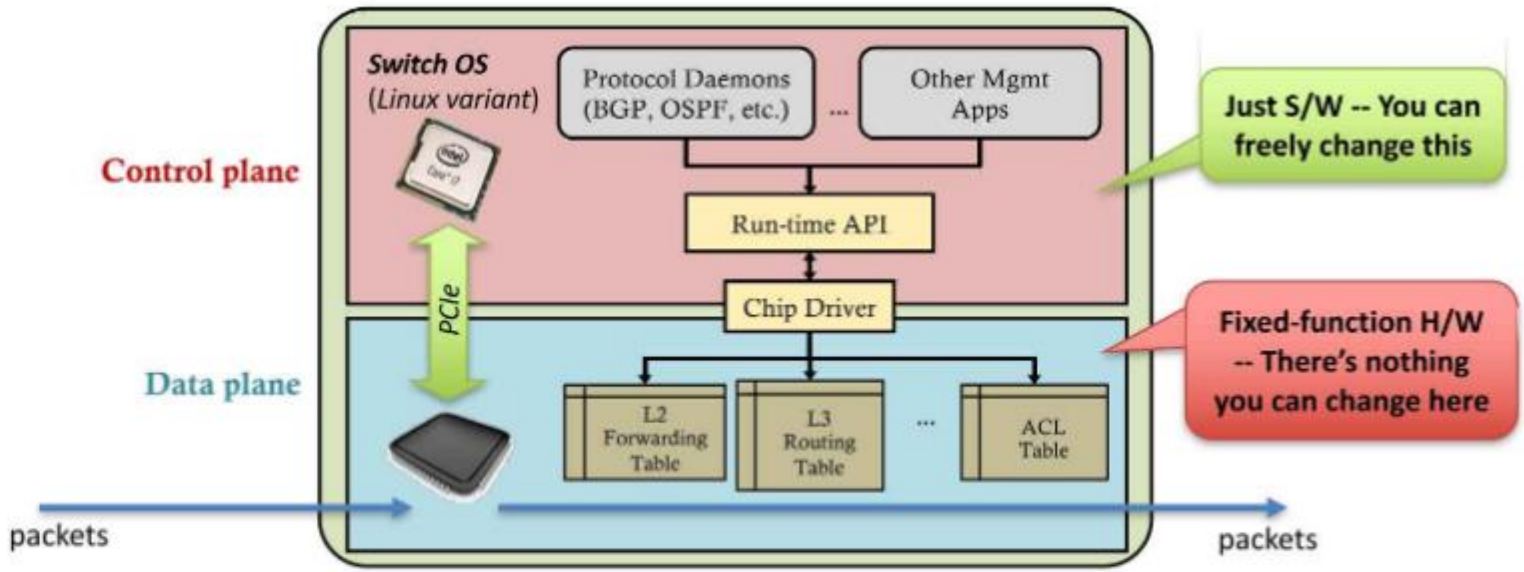
"We believe that future generations of OpenFlow should allow the controller to tell the switch how to operate rather than be constrained by a fixed switch design" – Nick McKeown

Programmability

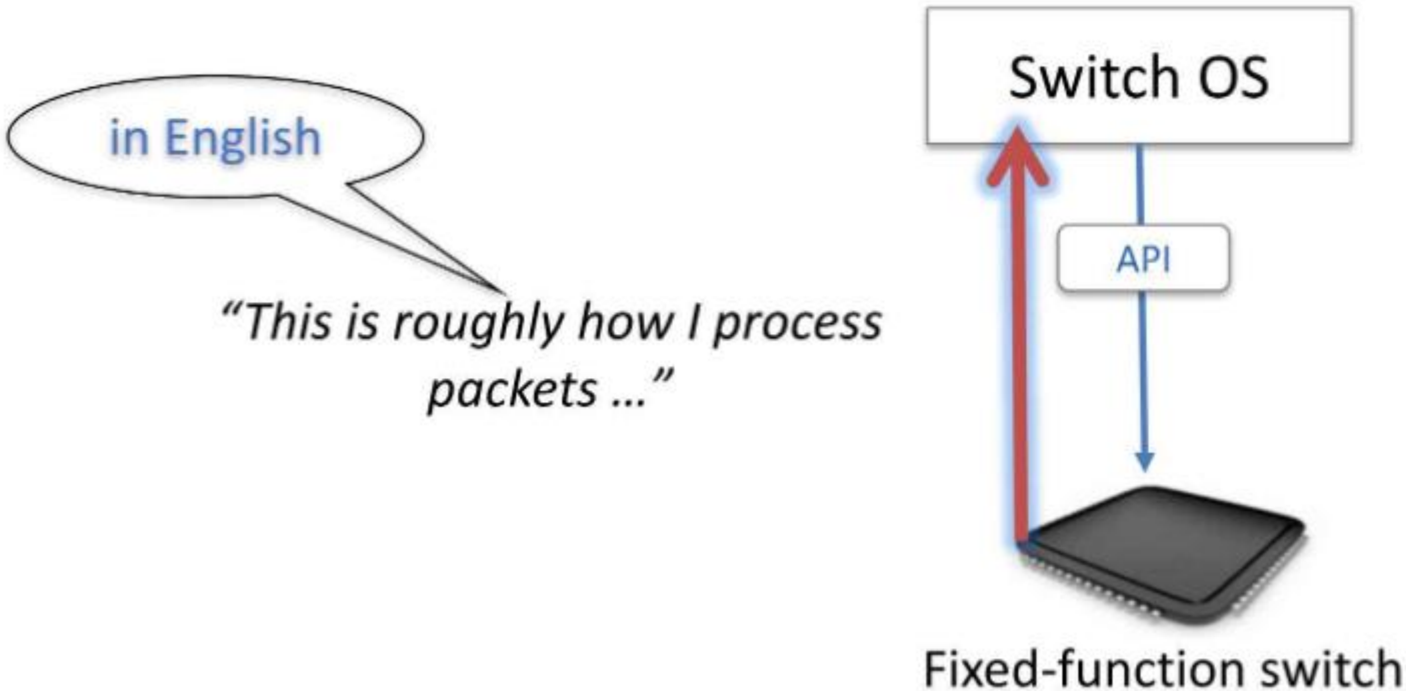


What does a typical switch look like?

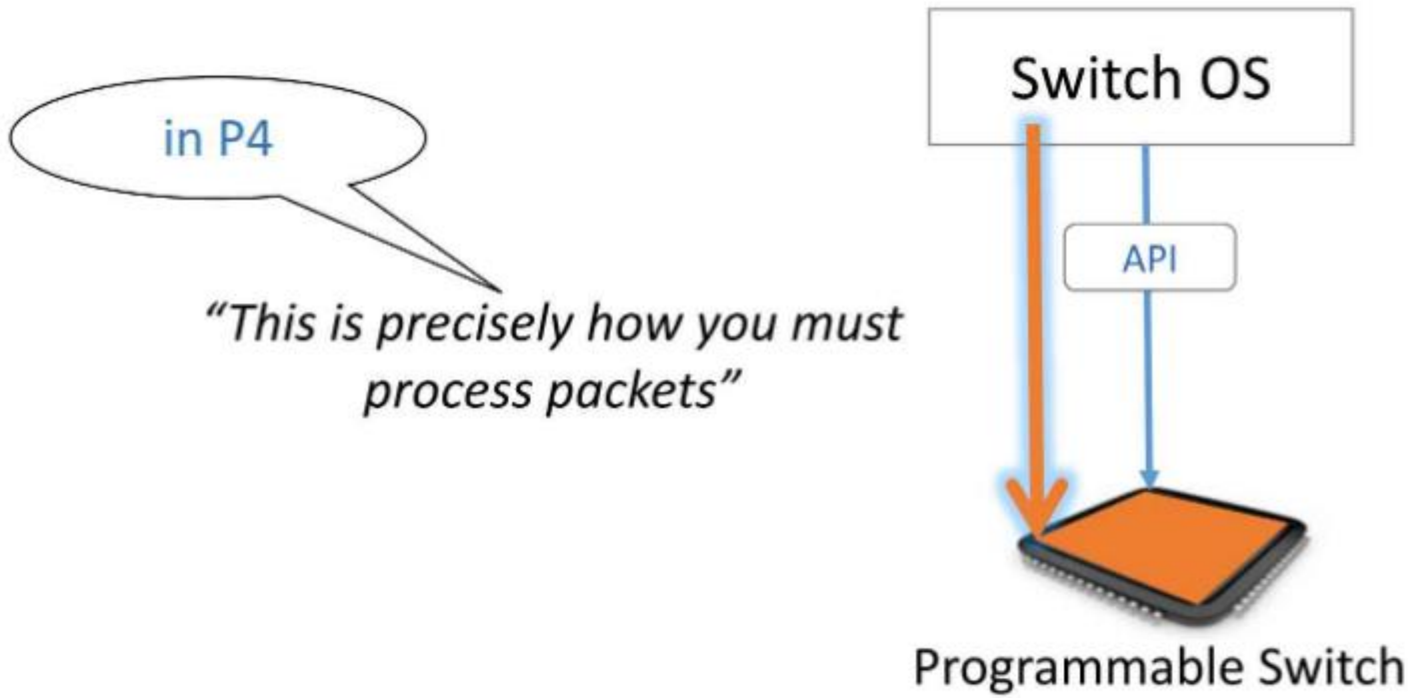
- A switch is just a Linux box with a high-speed switching chip



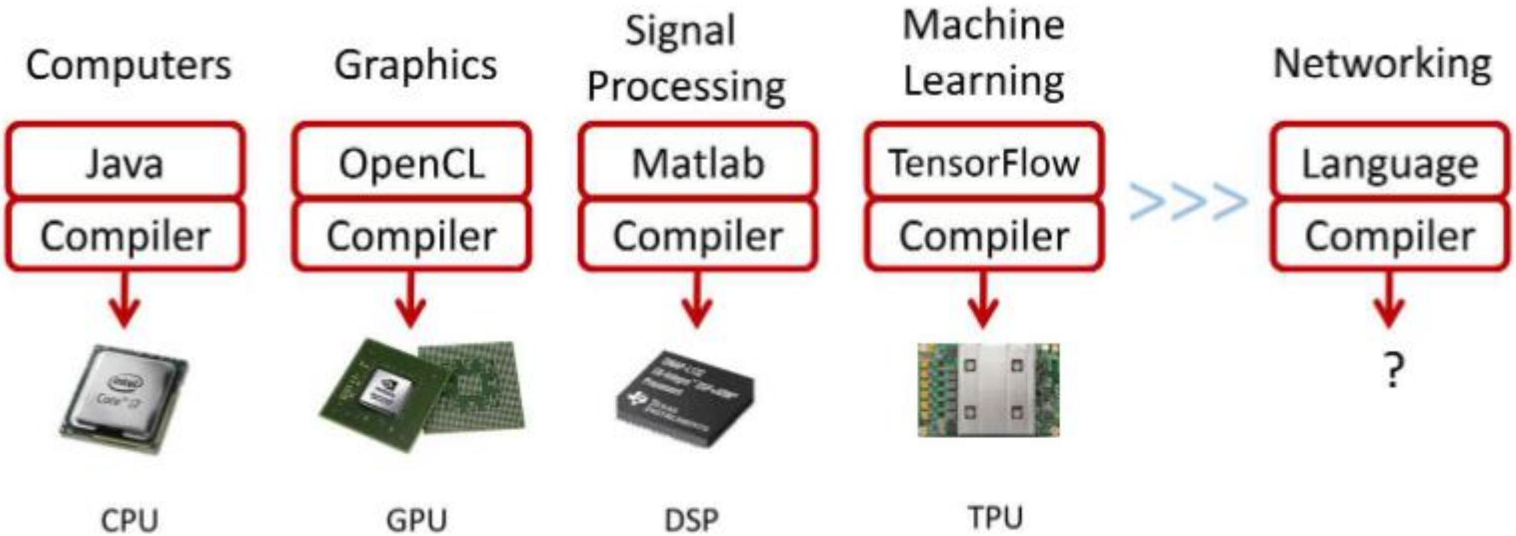
Networking systems have been built "bottoms-up"



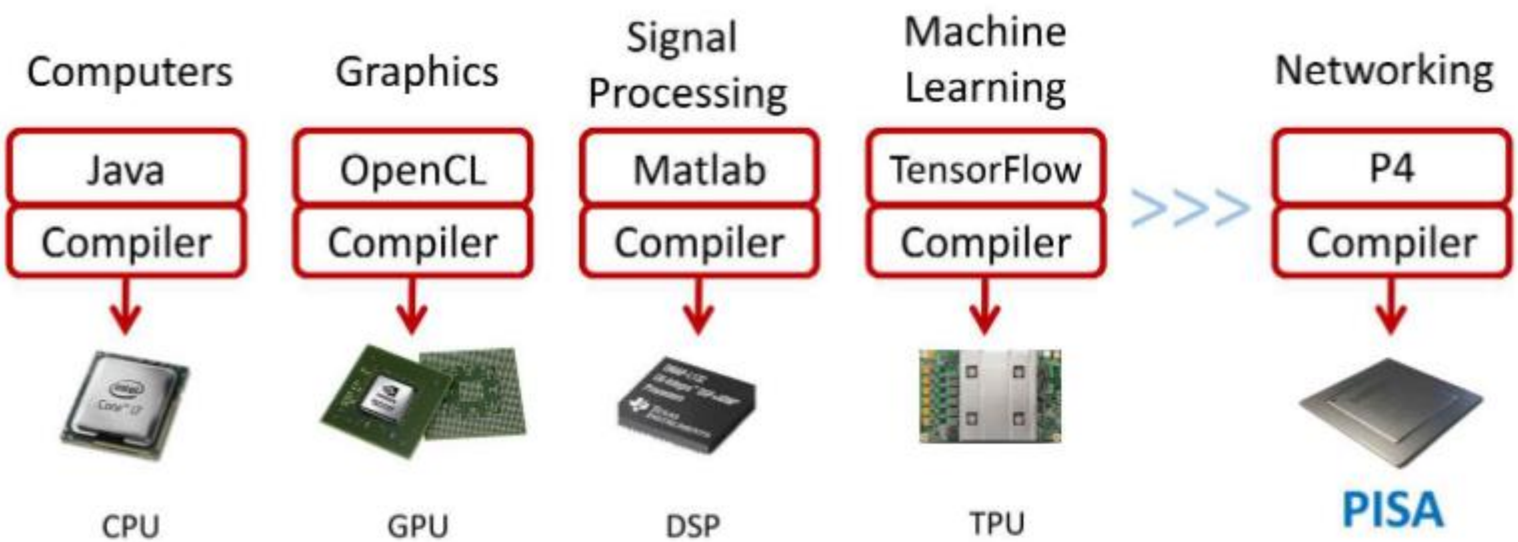
Turning the tables “top-down”



Domain-specific processors



Domain-specific processors



Programmable data plane

Category	Target	Throughput	Price	Latency
Programmable Switch	Intel Tofino ASIC, Juniper Networks’ MX480 router, Nvidia Spectrum-2 SN3000 series	10x Tbps	$10x K	1x µs
FPGA	NetFPGA	10x~100x Gbps	$1x K	10~100x µs
Smart NIC	Netronome Agilio CX, Netronomr NFP, Cavium Octeon II CN	10x~100x Gbps	$1x K	10~100x µs

Intel Infrastructure Processing Unit (IPU)

Intel Tofino ASIC

Three generations of Intel Tofino IFPs at-a-glance

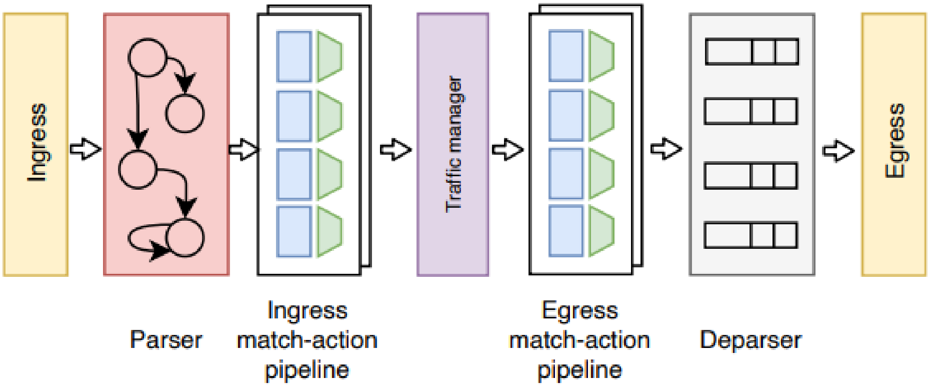
Parameter	Intel Tofino 1 (up to 6.4 Tb)	Intel Tofino 2 (up to 12.8 Tb)	Intel Tofino 3 (up to 25.6 Tb)
Process	16 nm	7 nm	7 nm
Num of MAU stages/pipe	12	20	20
Total SRAM/pipe	120 Mb	200 Mb	200 Mb
Total TCAM/pipe	6.2 Mb	10.3 Mb	10.3 Mb
Scheduler	1-level	2-level	2-level
Number of queues/port	32/100 Gb port	Up to 128/400 Gb port	Up to 128/400 Gb port
CPU port queues	32	Up to 128	Up to 128
Maximum SerDes speed	25 Gbps	56 Gbps	112 Gbps
Port speeds supported	100 Gb/50 Gb/40 Gb/ 25 Gb/10 Gb	400 Gb/200 Gb/100 Gb/ 50 Gb/40 Gb/25 Gb/10 Gb	400 Gb/200 Gb/100 Gb/ 50 Gb/40 Gb/25 Gb/10 Gb
Maximum port contexts	256	256	256

P4 – Programming Protocol-Independent Packet Processors

P4 is a domain-specific language for network devices, specifying how data plane devices (switches, NICs, routers, filters, etc.) process packets.

P4 Switch

V1Model



Websites

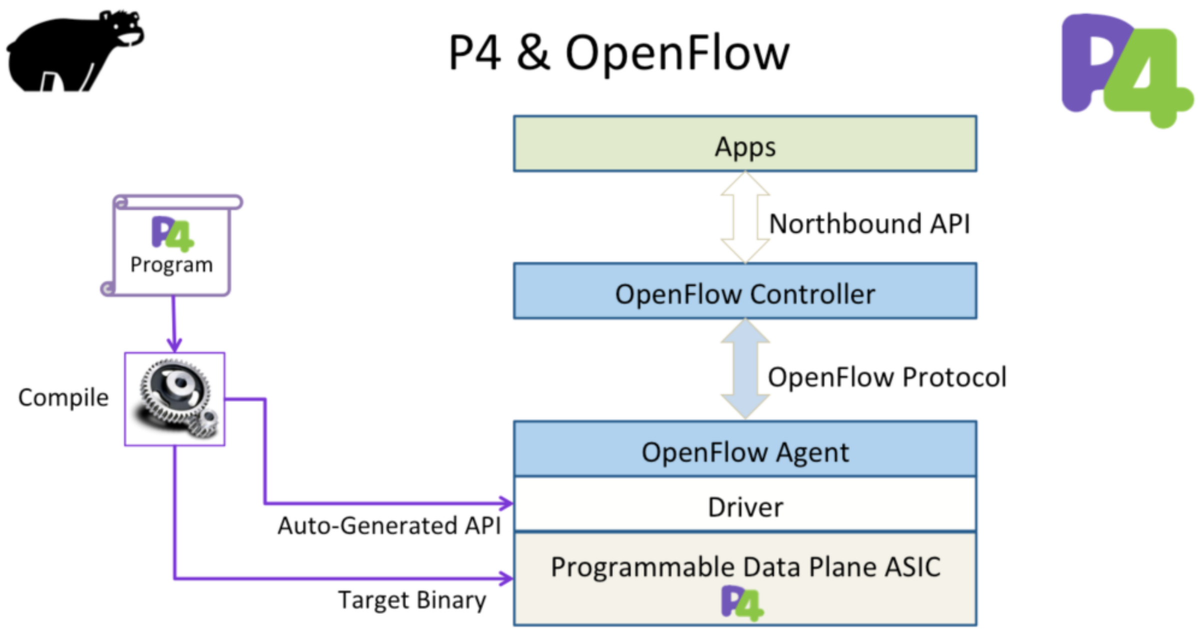
- **P4** – Programming Protocol-Independent Packet Processors
  - https://p4.org/
- **Intel Tofino** (Barefoot Networks)
  - https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch.html
- ONF – Open Networking Foundation
  - https://opennetworking.org/p4/

P4 and OpenFlow

- OpenFlow and P4 can work together for networks
  - OpenFlow is designed for SDN networks that separate the control plane from the forwarding plane
  - P4 is designed to program switches, which may be controlled locally from a switch OS, or remotely by an SDN controller.
- As more Table Type Patterns (TTP) are written for fixed-function switch chips, there is good reason that OpenFlow will be around for a while.

P4 and OpenFlow

- In P4, OpenFlow is one of many programs to describe what the forwarding plane does.
  - *openflow.p4*



About SDN

Innovation is the true value proposition of SDN.

SDN doesn't allow you to do the impossible.

It just allows you to do the possible much more easily.

--- Scott Shenker

Thank you!

Q & A

AntLab @ Jinan University

暨南大学 AntLab实验室

- 主要从事计算机网络方面的学术研究和系统应用等，包括
  - **可编程网络**：软件定义网络SDN、OpenFlow、可编程数据平面PDP…
  - **智能网络**：AI赋能网络(AI+Networking)
  - **网络协议优化**：网络拥塞控制、HTTP/3、TCP/QUIC、Multipath TCP/QUIC…
  - **云计算网络**：数据中心网络、网络功能虚拟化NFV…
  - **其他**：…

https://antlab.network

Prospective students

- We are looking for self-motivated graduates and PhD candidates. / 实验室长期招收硕士和博士研究生
- If you are interested in our researches, please apply for the master or PhD programs in AntLab
  如果您对实验室的研究方向感兴趣，<u>**欢迎报考AntLab实验室的硕士或博士**</u>

Website: https://antlab.jnu.edu.cn/cuilin/join
https://antlab.jnu.edu.cn

Email: tcuilin@jnu.edu.cn / cuilincui@gmail.com
Tel.: 15521031226
Office Venue: Room 435-4, Nanhai building, Main Campus (校本部南海楼435-4)
Lab Venue: Room 608, Nanhai building, Main Campus Campus