CS 5114
Network Programming Languages
Software-Defined Networking

http://www.flickr.com/photos/rofi/2097239111/

Nate Foster
Cornell University

Spring 2013

Based on lecture notes by Aditya Akella (and transitively, Aaron Gember, and Nick McKeown)

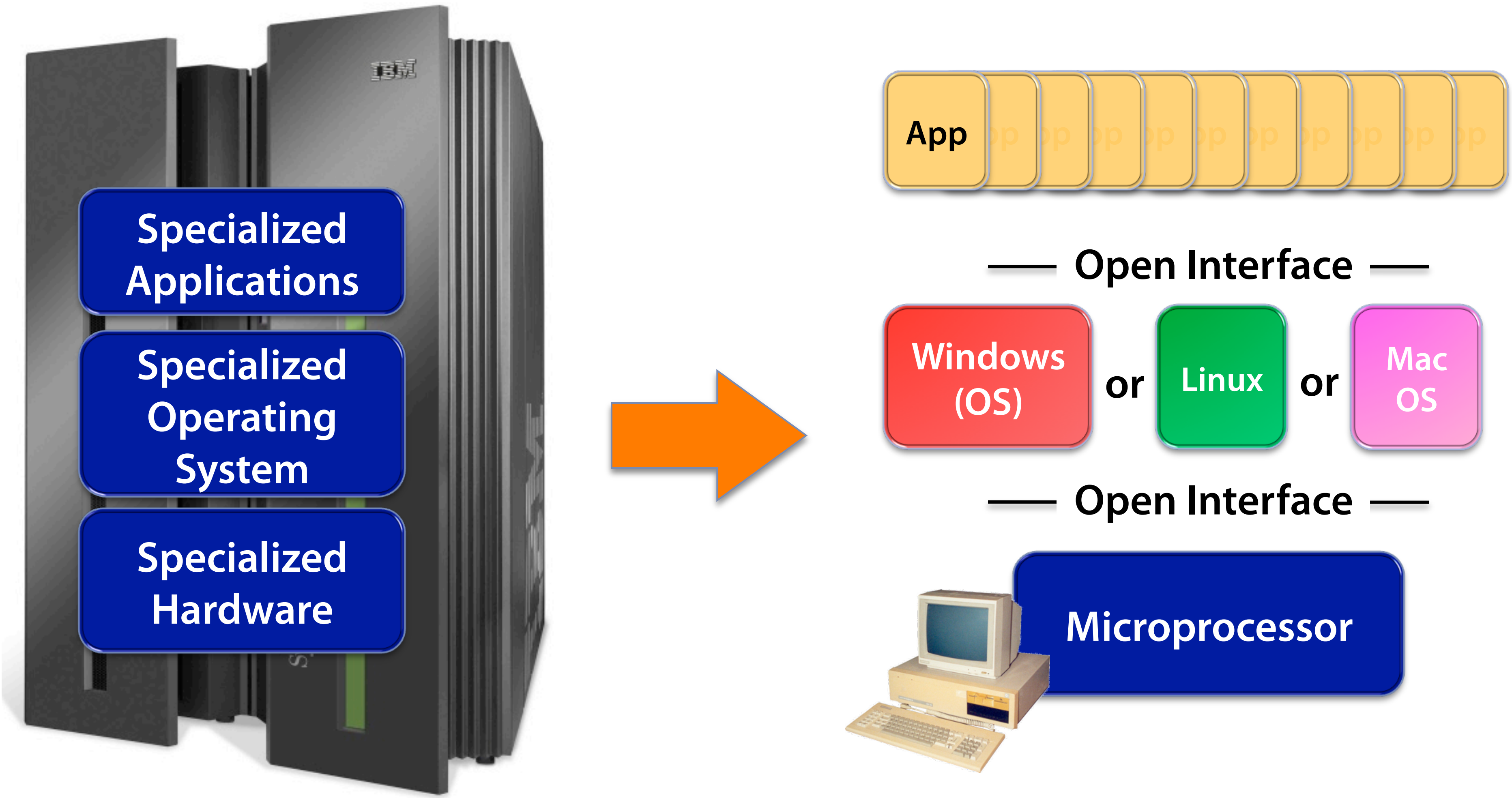
Announcements

Office Hours

3-4pm today in Upson 4137

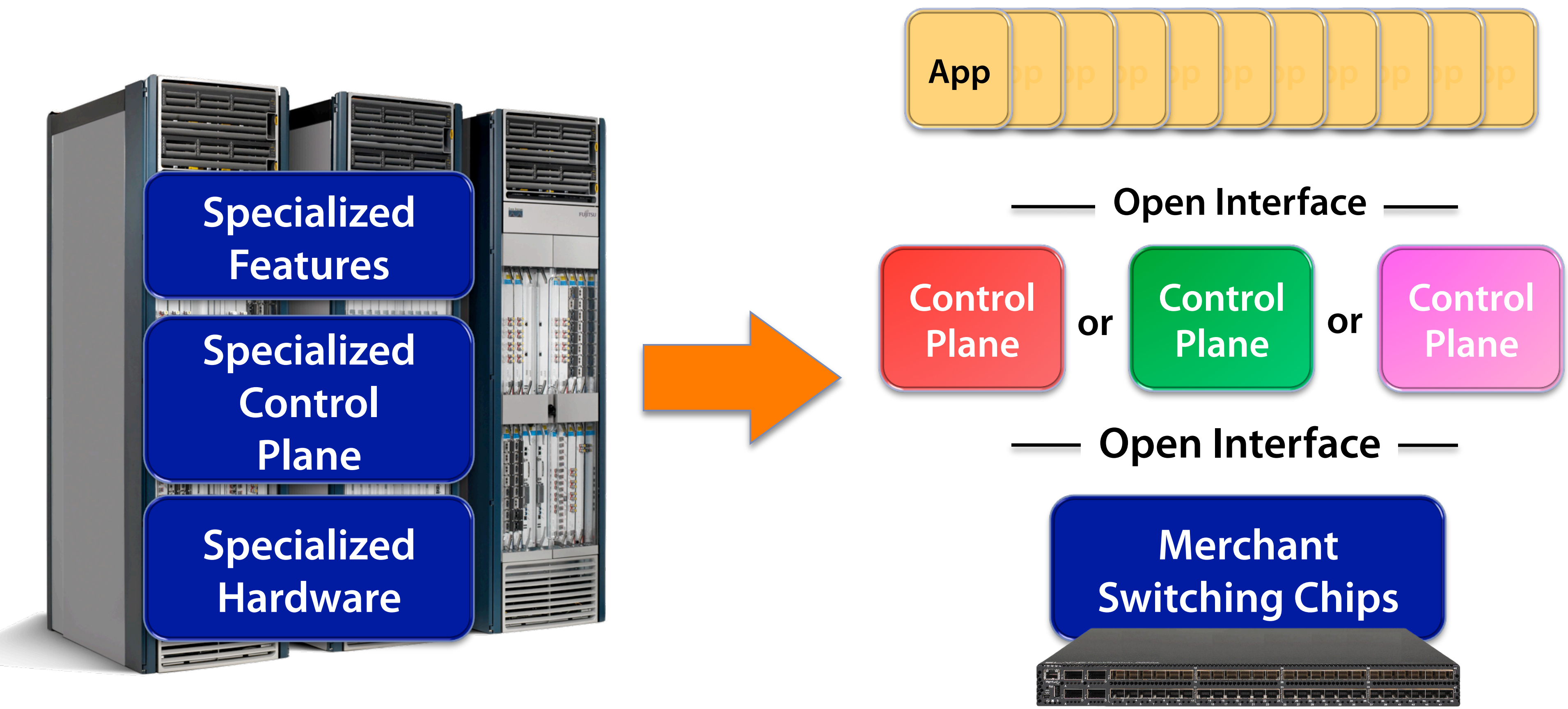
Homework #1

- Goes out today (on CMS)
- Due in two weeks
- Topic: Basic OpenFlow programming



Vertically integrated
Closed, proprietary
Slow innovation
Small industry

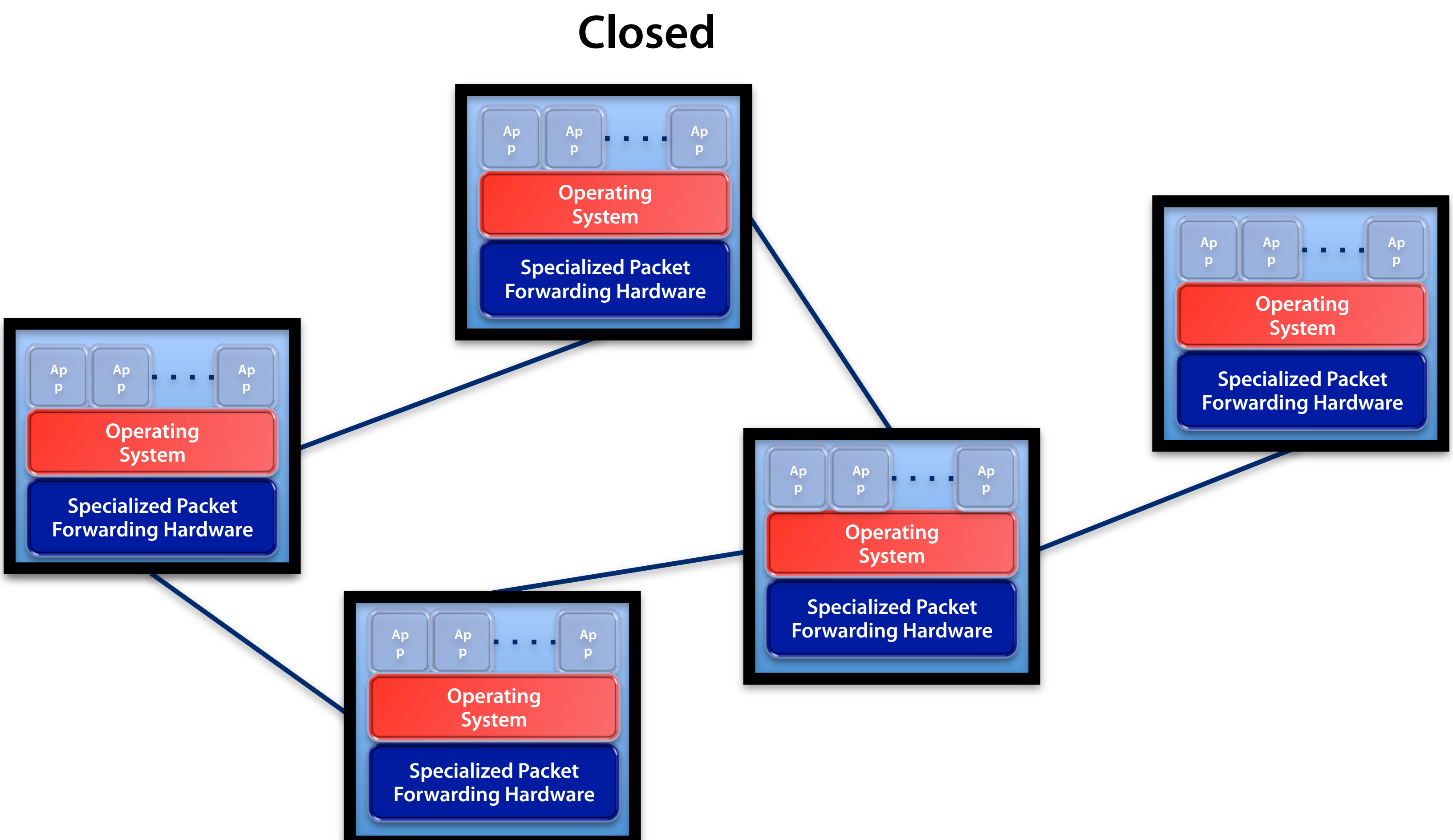
Horizontal
Open interfaces
Rapid innovation
Huge industry



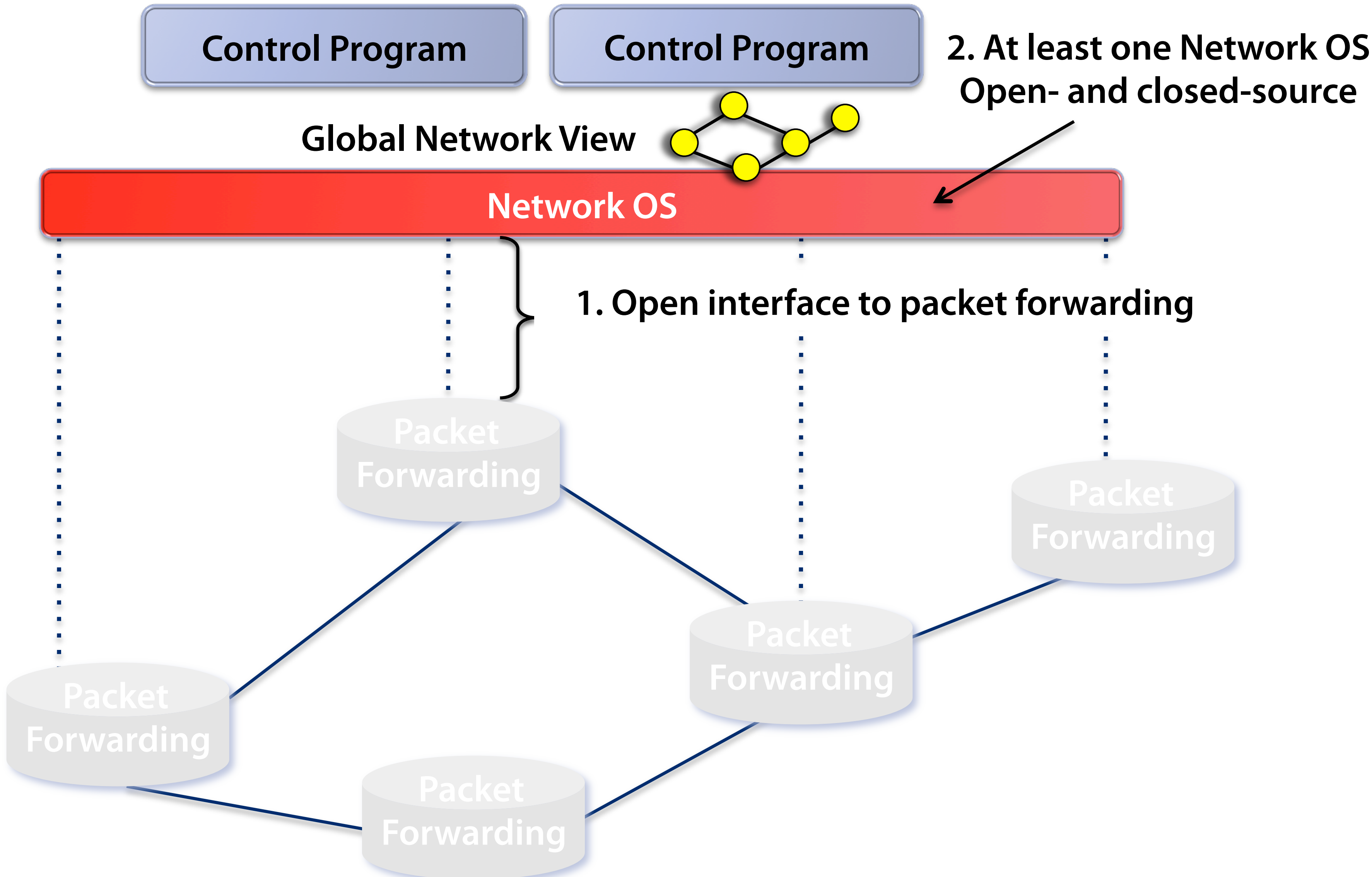
Vertically integrated
Closed, proprietary
Slow innovation

Horizontal
Open interfaces
Rapid innovation

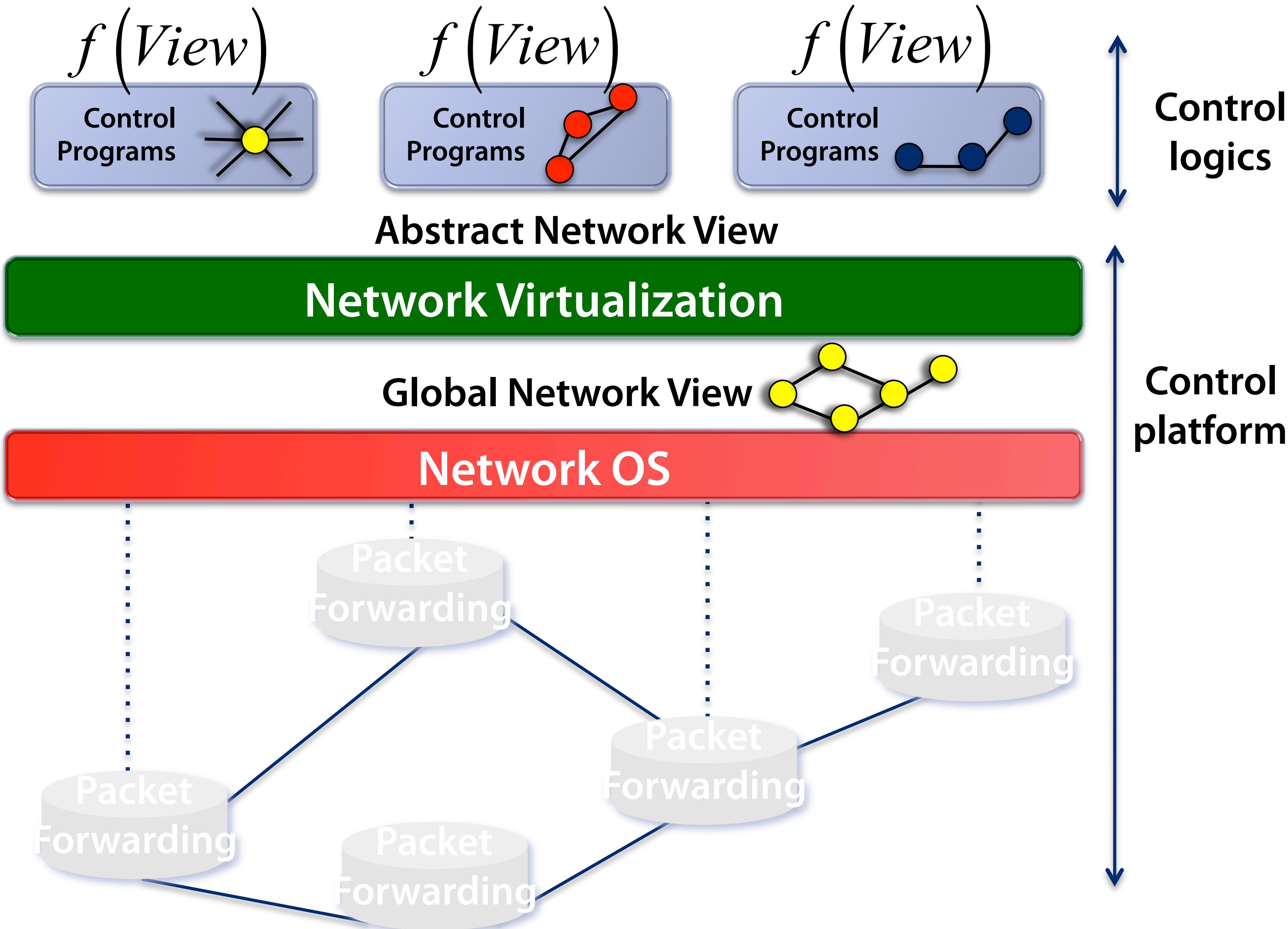
Today
Closed Boxes, Fully Distributed Protocols



Software Defined Network (SDN)



Software Defined Network (SDN)



Control Logic

- Runs on one or more controllers
- Manages computation of forwarding state and perhaps coordination among instances
- Control platform provides basic services to ease the latter (e.g., state distribution mechanisms)
- Logic must decide how to partition computation, deal with failover, and implement the consistency model

Control Platform

Schedule computations over the network graph

Store network state and support for different consistency models

Most control platforms today run a single application

- Not clear yet how to resolve interference (e.g., policy routing vs. traffic engineering)
- We're trying to answer some of these questions in the Frenetic project

State Distribution Abstraction

Control program should not have to handle all distributed-state details

Proposed abstraction: global network view

Control program operates on network view

- Input: global network view (graph)
- Output: configuration of each network device

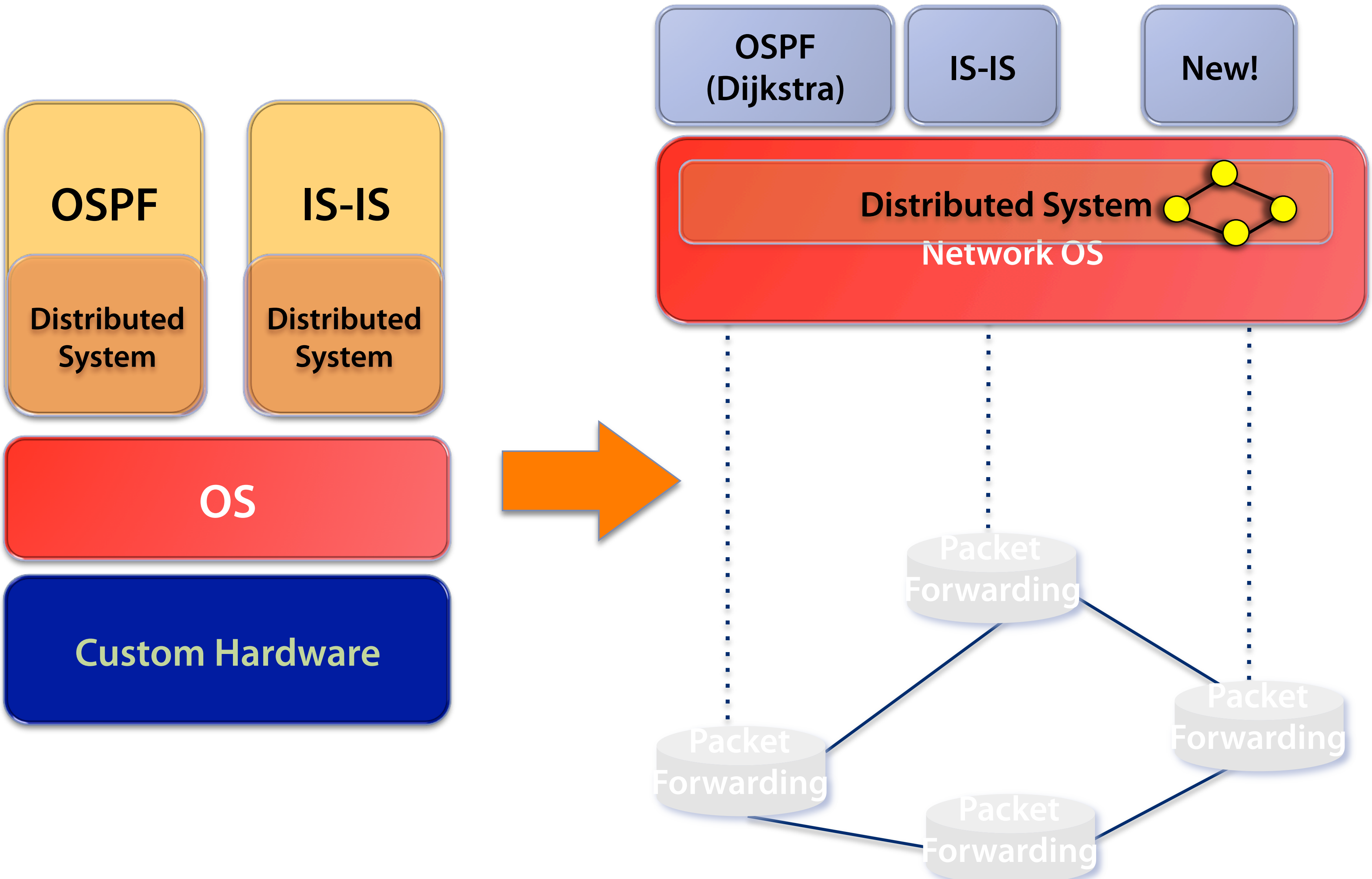
Network OS provides network view

Forwarding Abstraction

Forwarding behavior specified by a control program.

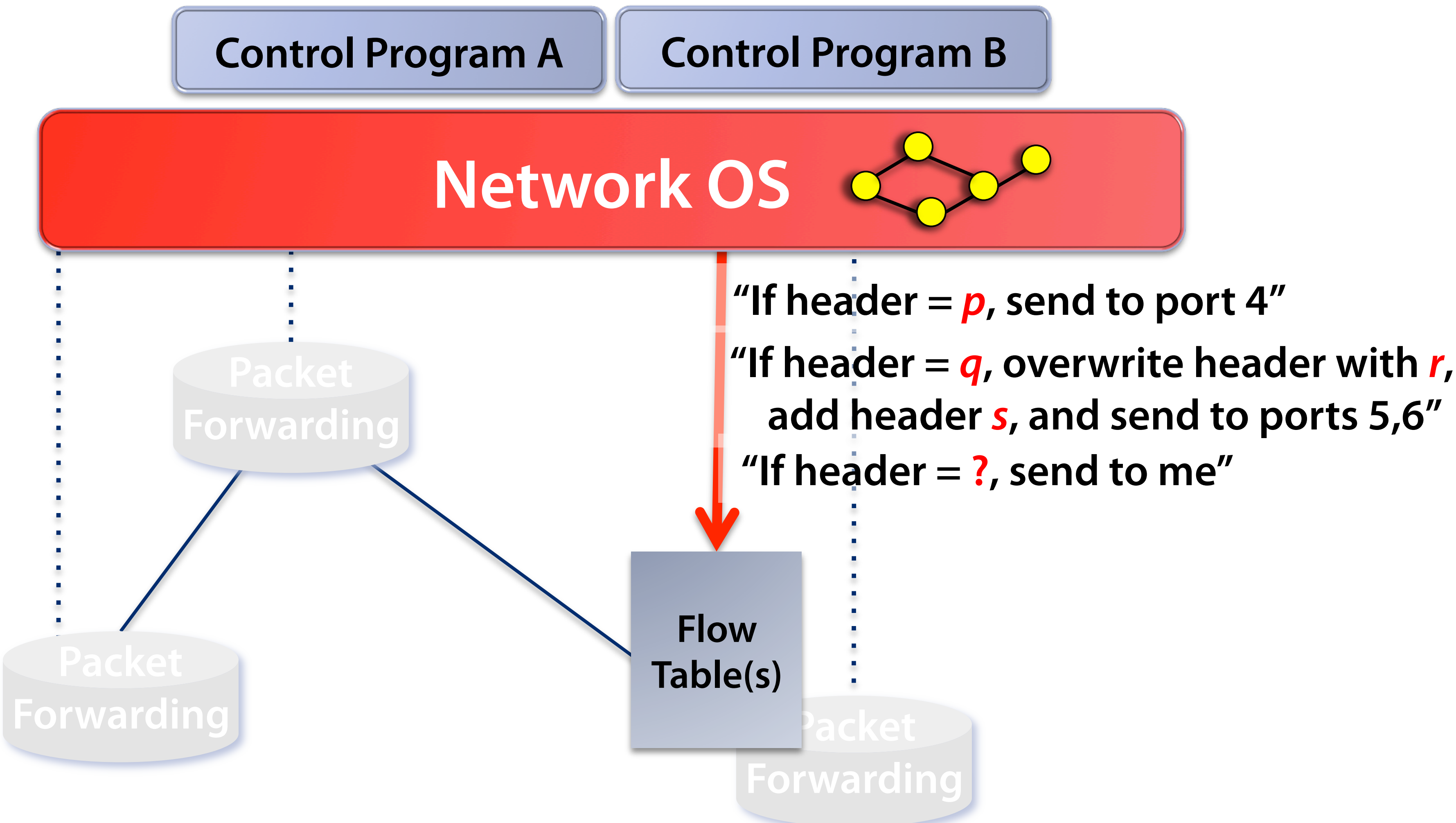
Possibilities: x86, MPLS, OpenFlow

Example

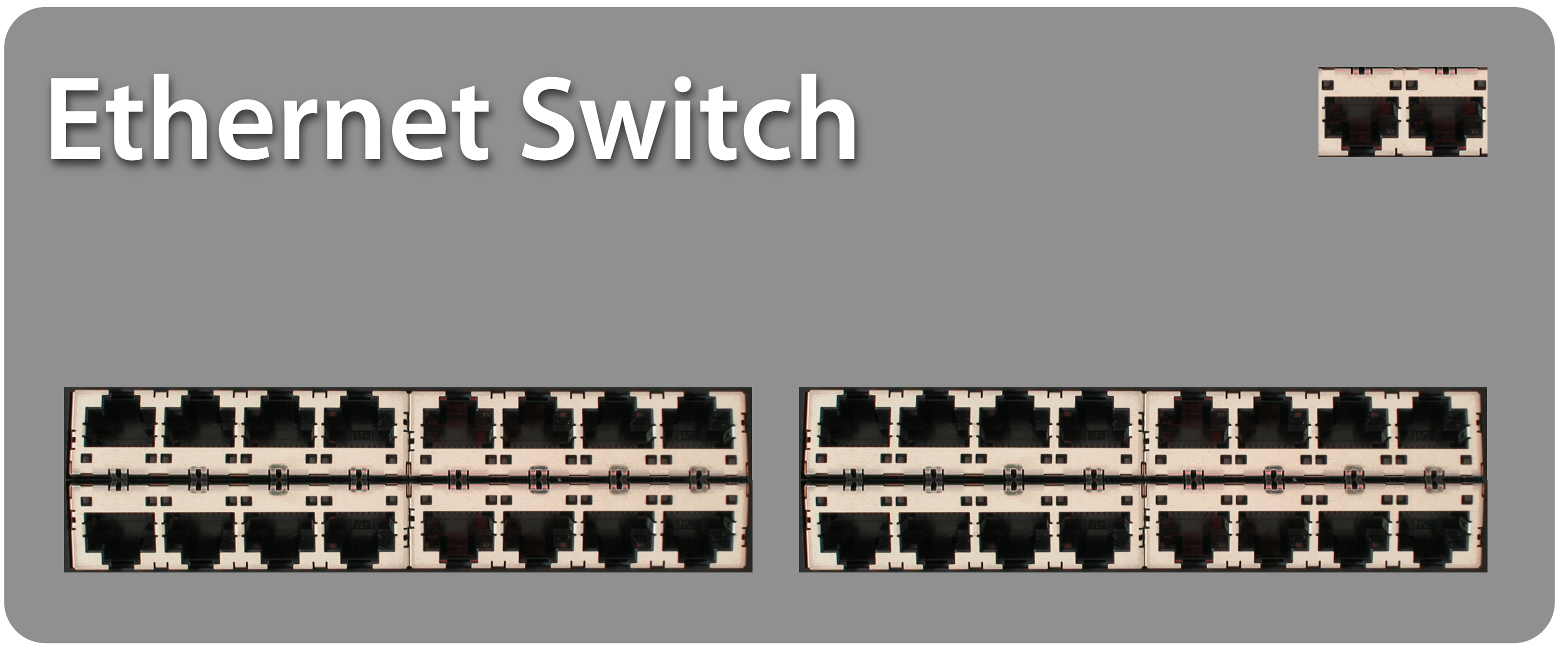


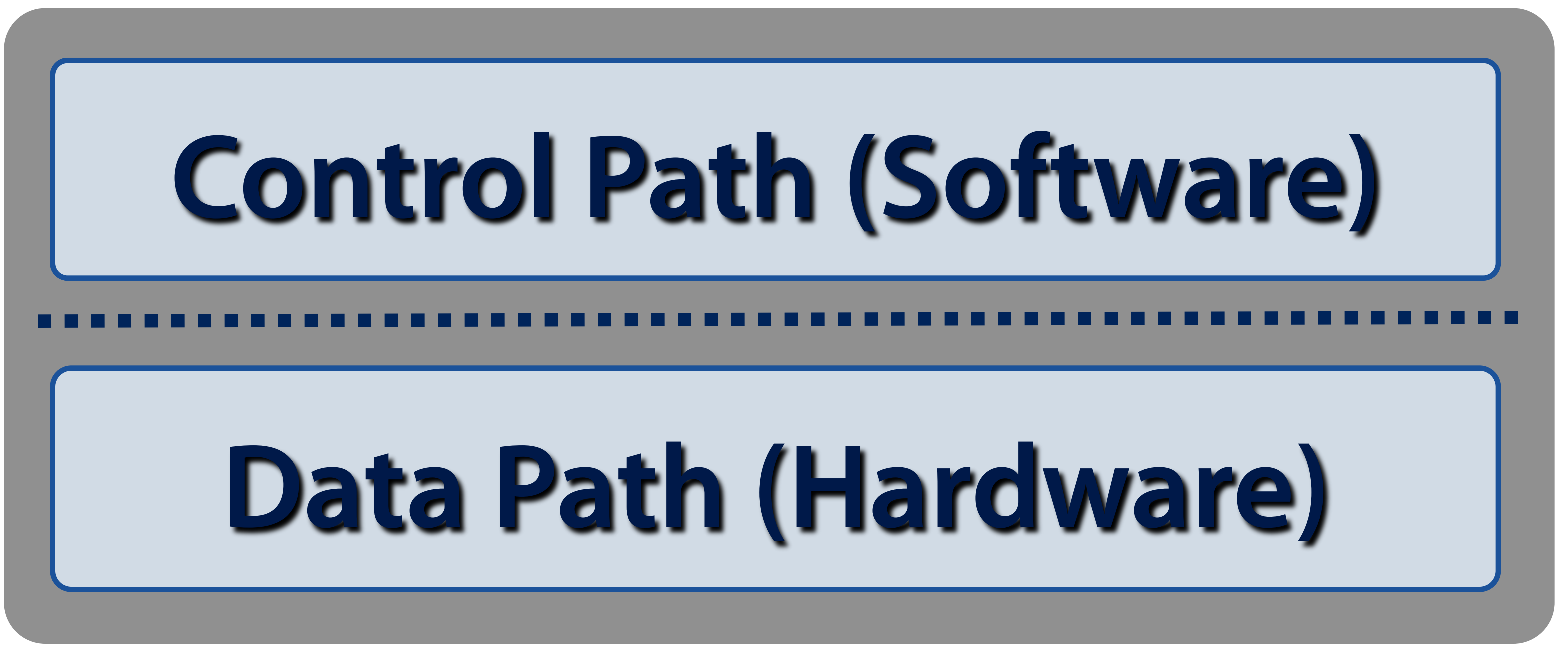
OSPF
IS-IS
Distributed System
Distributed System
OS
Custom Hardware
OSPF (Dijkstra)
IS-IS
New!
Distributed System
Network OS
Packet Forwarding
Packet Forwarding
Packet Forwarding
Packet Forwarding

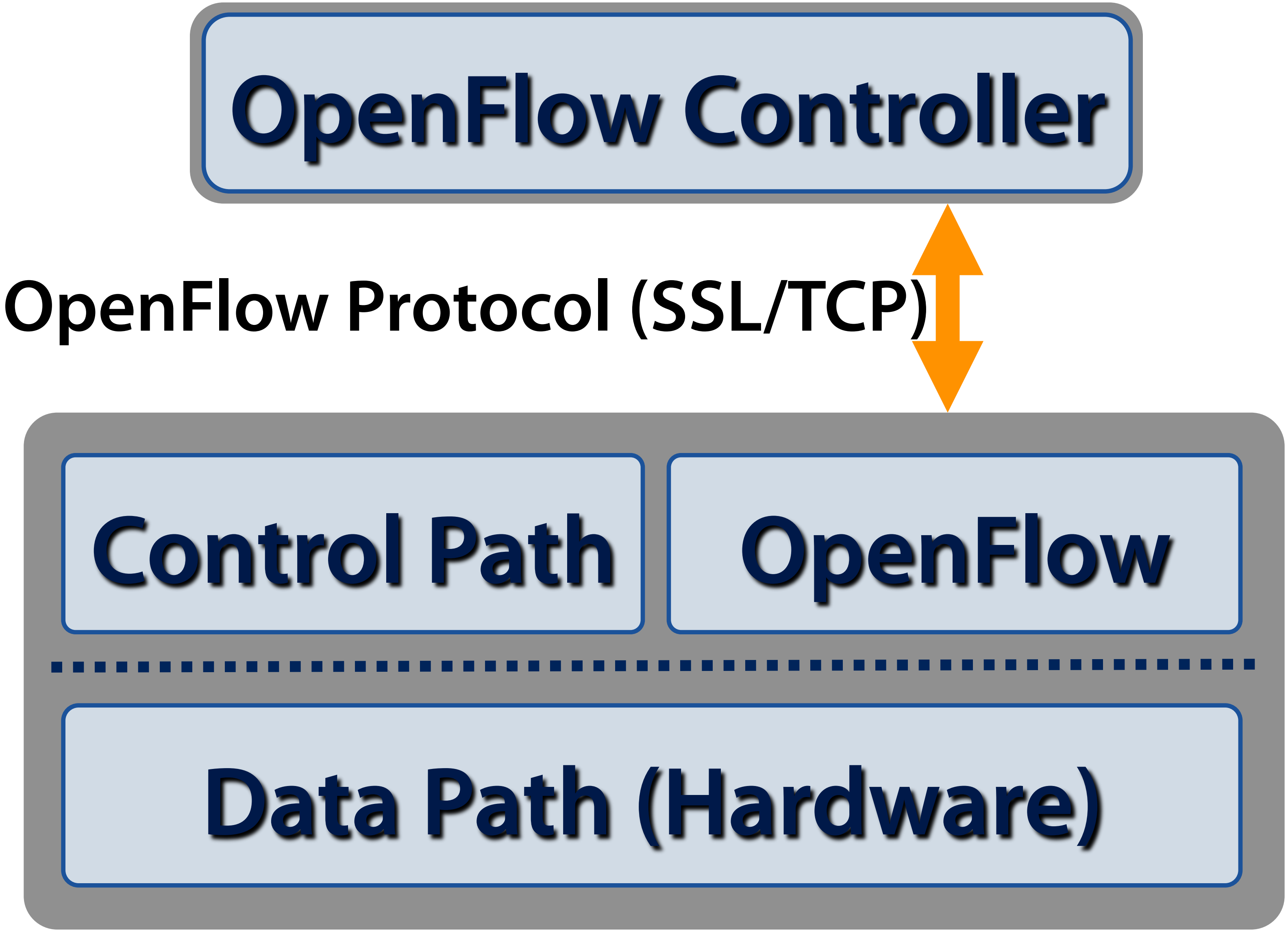
OpenFlow Forwarding Abstraction



How does OpenFlow work?




Control Path (Software)
Data Path (Hardware)


OpenFlow Controller
OpenFlow Protocol (SSL/TCP)
Control Path
OpenFlow
Data Path (Hardware)

OpenFlow Forwarding Abstraction

Patterns

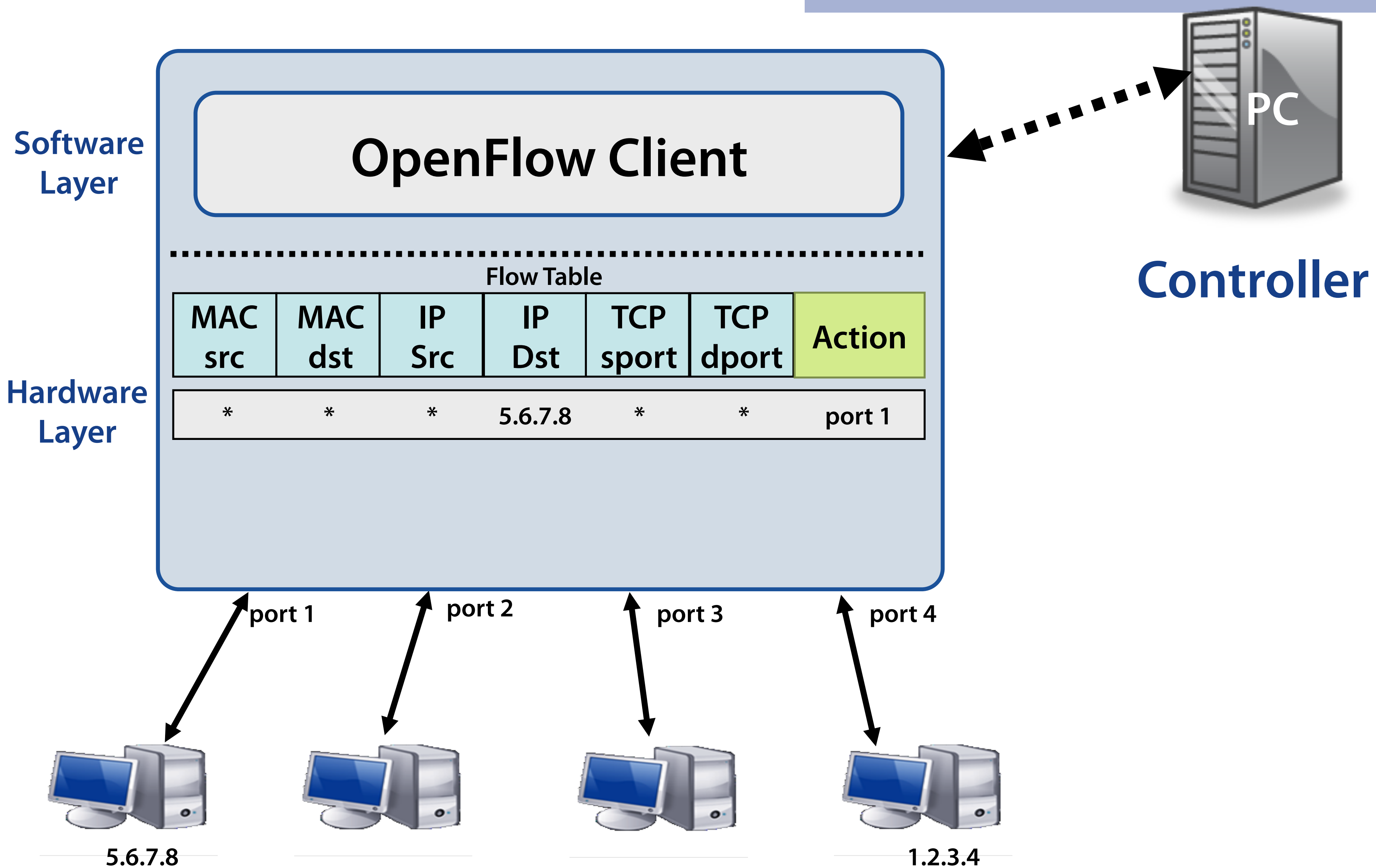
Example: 1000x01xx0101001x

- Allows any flow granularity

Actions

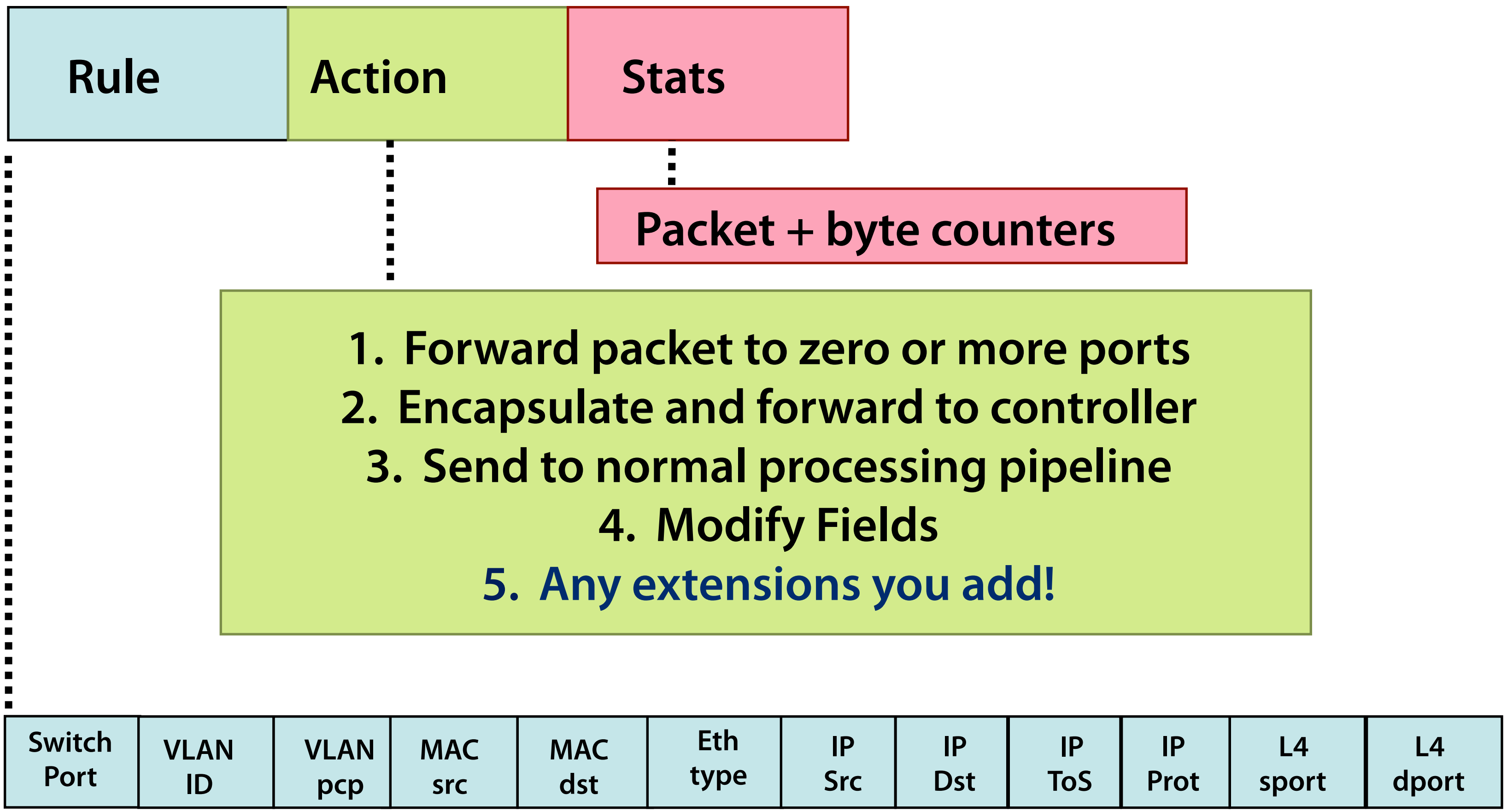
- Forward to port(s), drop, send to controller
- Overwrite header with mask, push or pop labels
- Forward at specific bit-rate

Example



Software Layer
OpenFlow Client
Flow Table
MAC src
MAC dst
IP Src
IP Dst
TCP sport
TCP dport
Action
Hardware Layer
*
*
*
5.6.7.8
*
*
port 1
PC
Controller
port 1
port 2
port 3
port 4
5.6.7.8
1.2.3.4

OpenFlow Flow Tables



+ mask what fields to match

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

OpenFlow Forwarding Abstraction

Protocol Independent

- Construct Ethernet, IPv4, VLAN, MPLS, …
- Construct new forwarding methods

Backward Compatible

- Run in existing networks

Technology Independent

- Switches, routers, WiFi APs
- Cellular basestations
- WDM/TDM circuits

Things to Note about Forwarding

Common OpenFlow model is to use first packets of flows to compute and push state

Flows vs state: SDN allows more general models of forwarding state management independent of traffic

- Events trigger changes, e.g., failures, control traffic
- Managing inconsistencies is critical

Fabrics vs switches: Control logics don't have to deal with switches

- They essentially program a fabric that looks like one large switch and supports end-to-end connectivity by default
- Complex logic pushed to the edge

Virtual Data Paths

Thinking in terms of fabrics essentially means control logics have to deal with simpler topologies

- Topology captured by "virtual data paths"
  - Depending on control logic, can be very simple: for access controls it is just data path through a single switch
- Every virtual element uses familiar forwarding abstractions, e.g., L2, L3 and ACLs

Control platform responsible for mapping virtual data path to the physical network

SDN in development

Domains

Data centers

Public clouds

Enterprise/campus

Cellular backhaul

Enterprise WiFi

WANs

Home networks

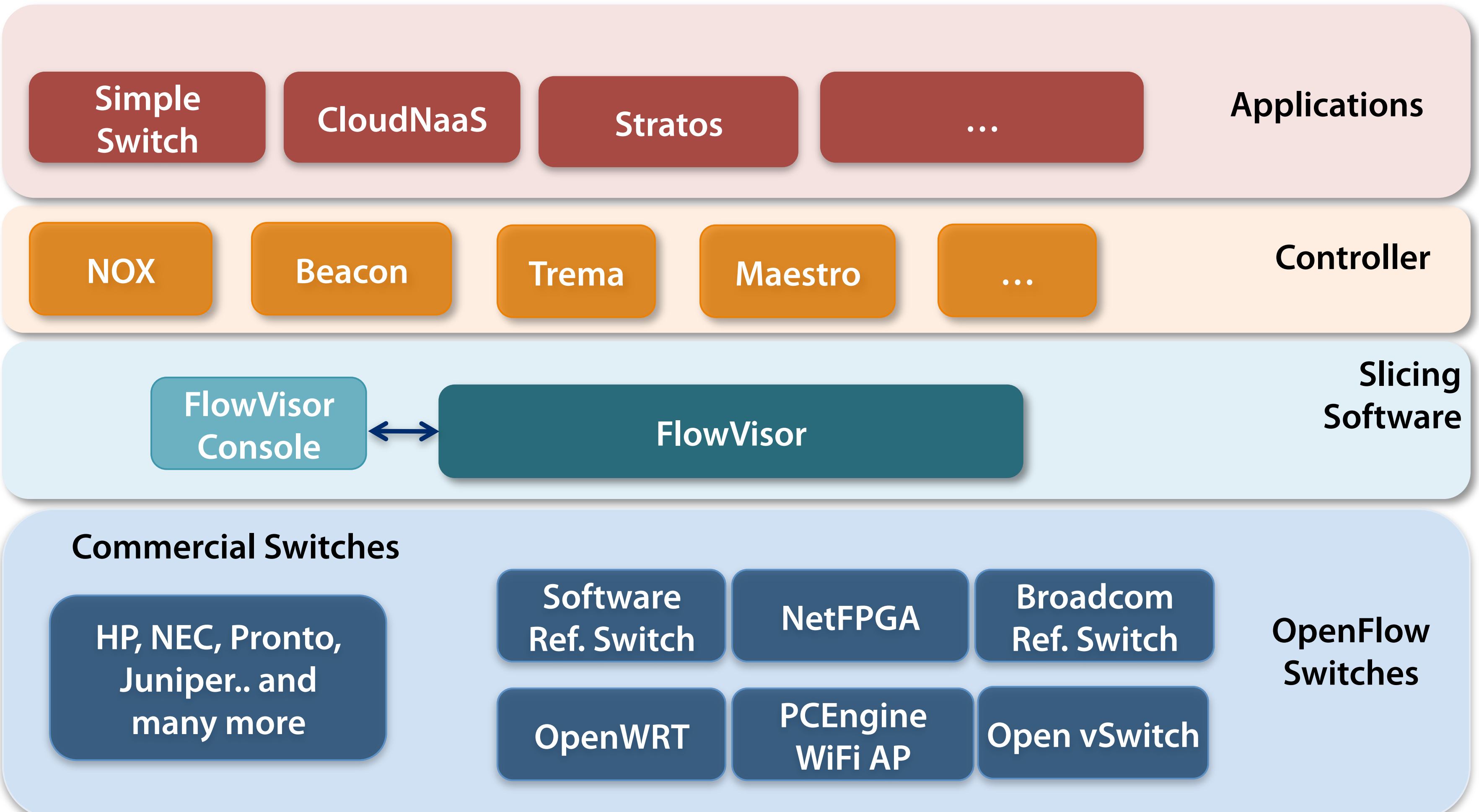
Products

Switches, routers:
  About 15 vendors

Software: 8-10 vendors
  and startups

New startups. Lots of hiring in networking.

The SDN Stack



The SDN Stack

Controller

OpenFlow
Switches

OpenFlow Progression

OF v1.0: released end of 2009: “Into the Campus”

OF v1.1: released March 1 2011: “Into the WAN”

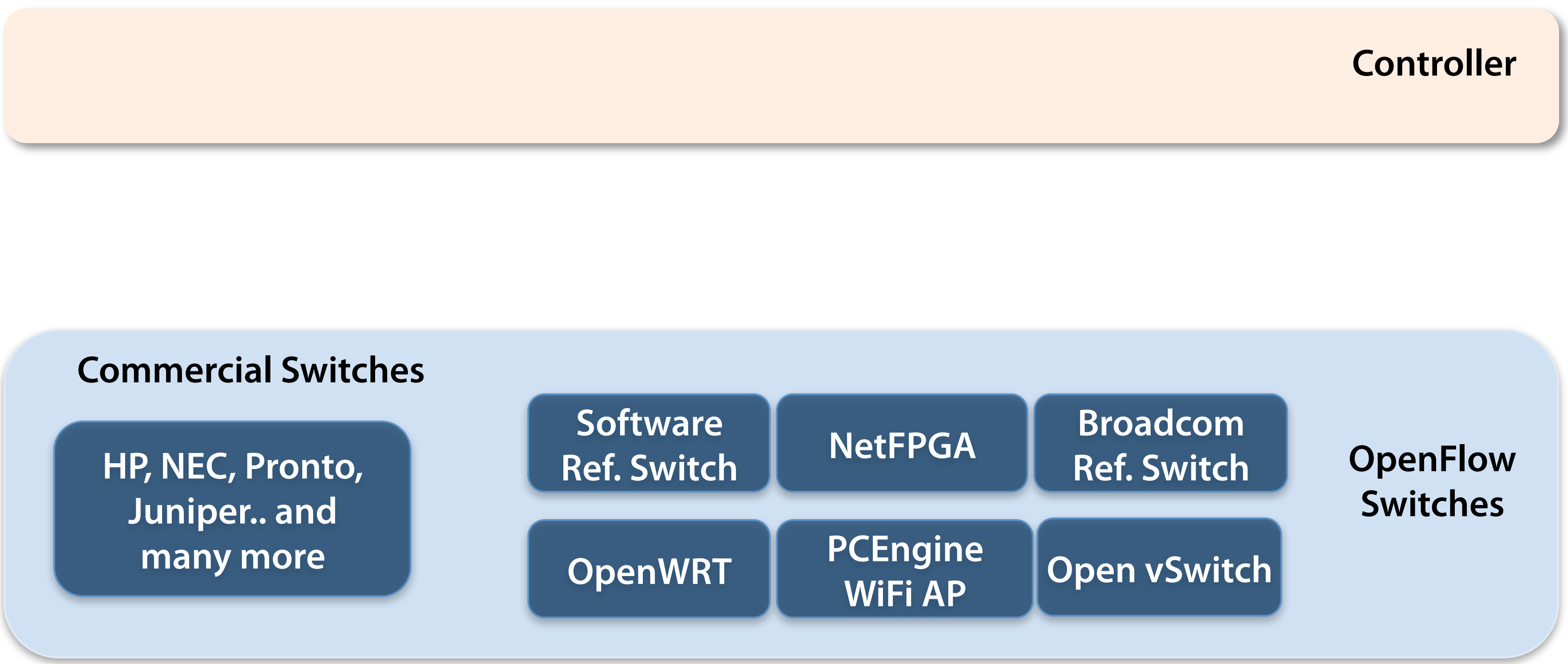
- multiple tables: leverage additional tables
- tags and tunnels: MPLS, VLAN, virtual ports
- multipath forwarding: ECMP, groups

OF v1.2: approved Dec 8 2011: “Extensible Protocol”

- extensible match
- extensible actions
- IPv6
- multiple controllers

OF v1.3: approved May 17 2012

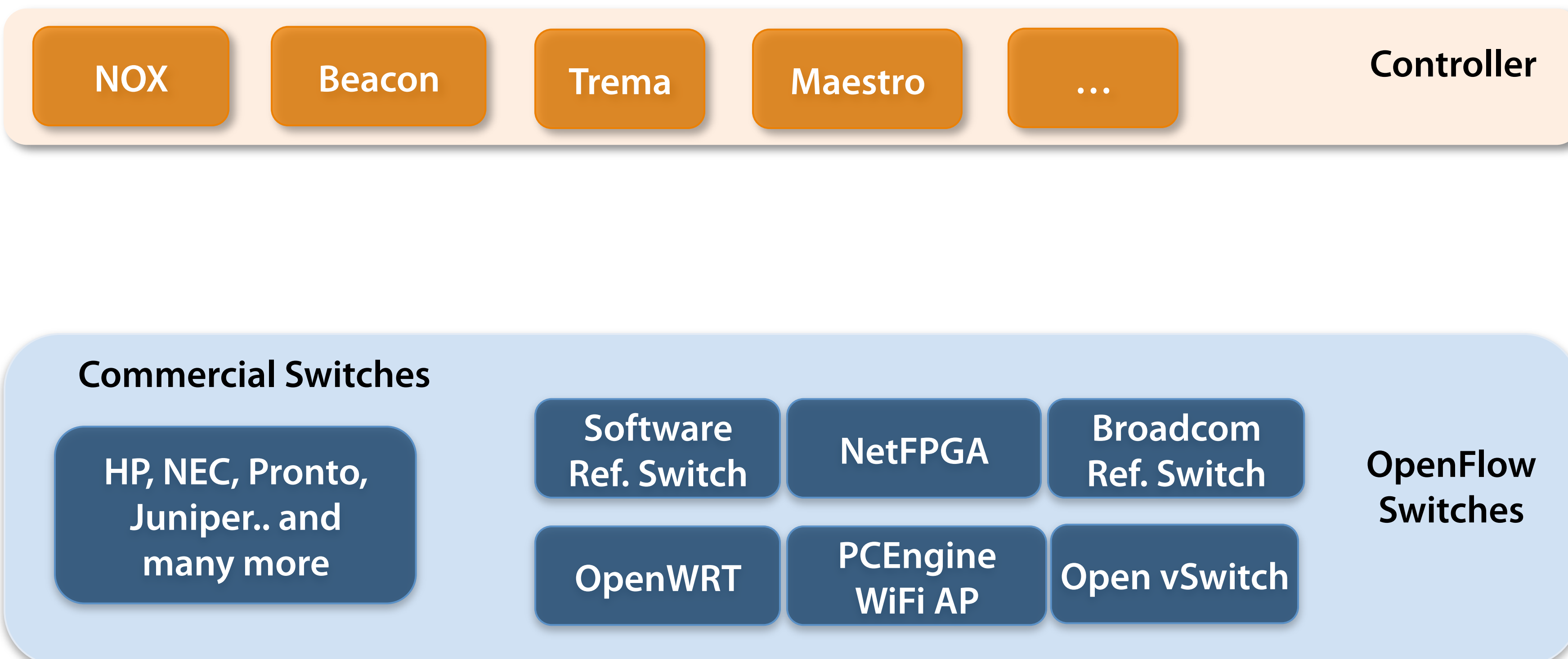
The SDN Stack



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

The SDN Stack



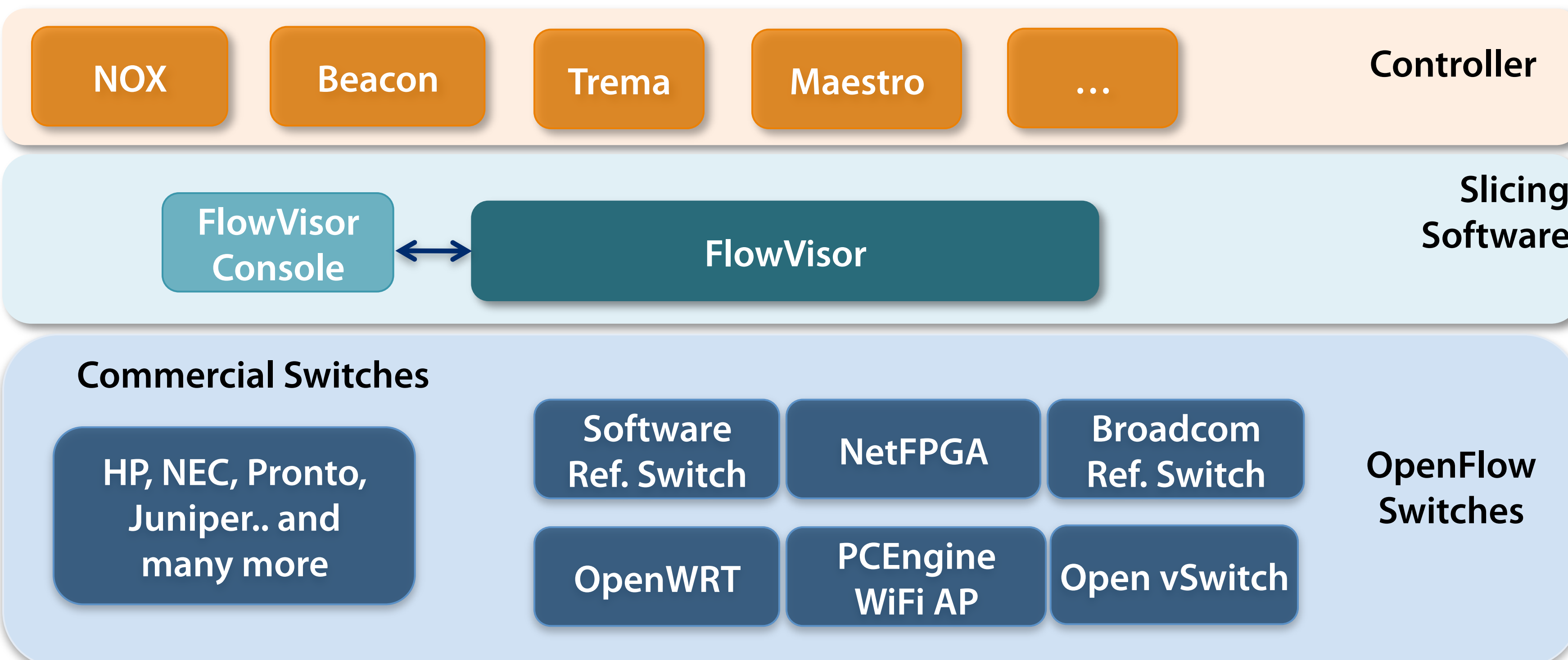
Controllers

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	

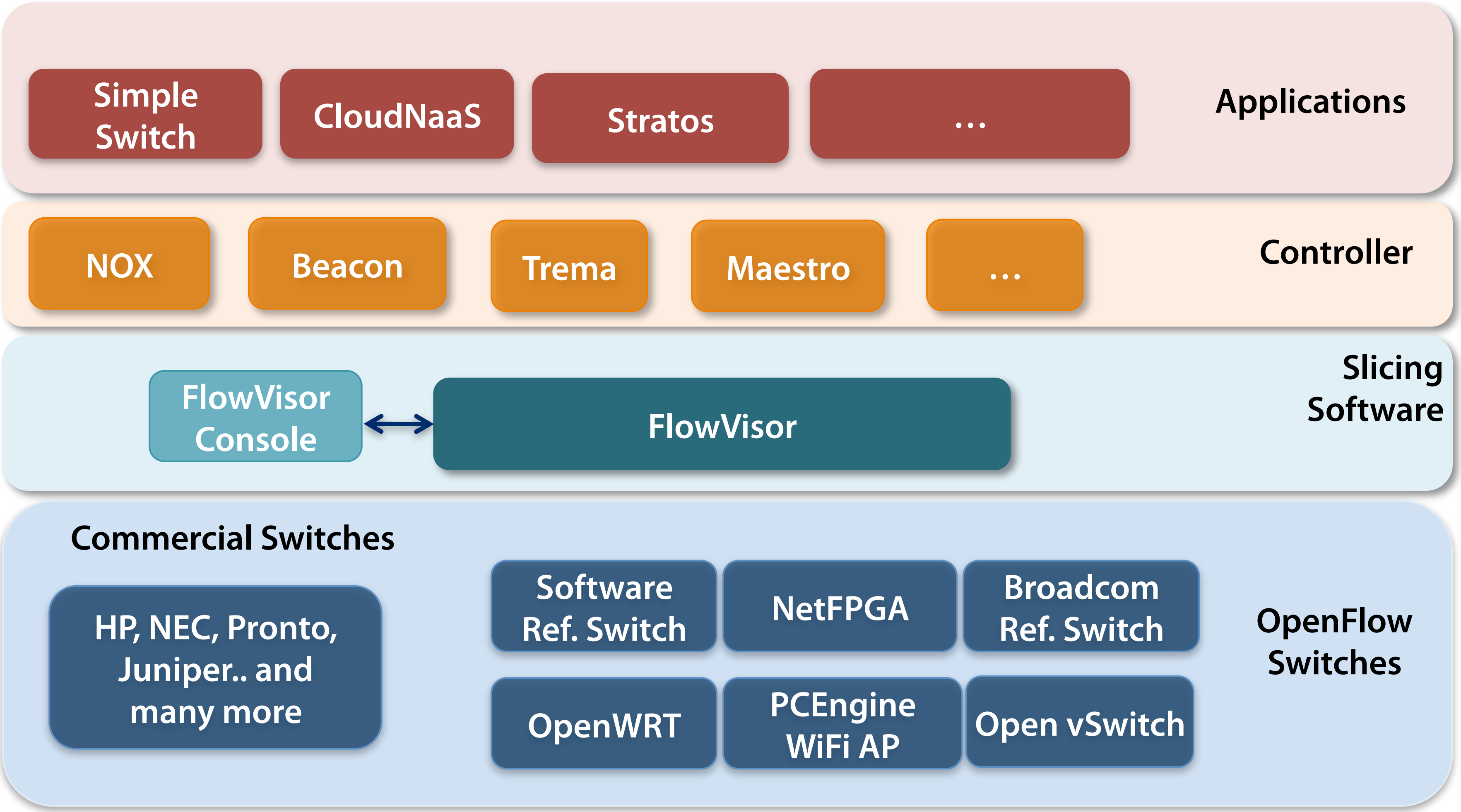
Too many to easily keep track of…

http://yuba.stanford.edu/~casado/of-sw.html

The SDN Stack



The SDN Stack



How SDN will shape networking

1. Empower network owners and operators
   - Customize networks to local needs
   - Eliminate unneeded features
   - Creation of virtual, isolated networks

2. Increase the pace of innovation
   - Innovation at software speed
   - Standards (if any) will follow software deployment
   - Technology exchange with partners
   - Technology transfer from universities

Summary

Networks becoming

- More programmatic
- Defined by owners and operators, not vendors
- Faster changing, to meet operator needs
- Lower opex, capex and power

Abstractions

- Will shield programmers from complexity
- Make behavior formally verifiable
- “Will take us places we can’t yet imagine”