Getting Started

- (1) Copy to your hard disk from a USB Key or DVD:
  - Copy needed files (VirtualBox, terminal, possibly an X server) for your platform (Win/Mac/Linux)
  - Copy Java 6 and Eclipse for your platform, if you want to use Java
  - **Copy VM image: OpenFlowTutorial-101311.zip**
  - Pass on the DVD or USB key to someone else!

- (2) Unzip OpenFlowTutorial-101311.zip

- (3) Point browser to instructions:
  - http://www.openflow.org/wk/index.php/OpenFlow_Tutorial (note the underscore)

- You should NOT need to download any large files – spare the WiFi!

Tutorial 1: SDN for Engineers

part of the the Open Networking Summit
Santa Clara Marriott
April 16, 2012

Brandon Heller, Rob Sherwood, David Erickson, Hideyuki Shimonishi, Srini Seetharaman, Murphy McCauley

only possible help from all the people

listed on the next few pages

Getting Started

- (1) Copy to your hard disk from a USB Key or DVD:
  - Copy needed files (VirtualBox, terminal, possibly an X server) for your platform (Win/Mac/Linux)
  - Copy Java 6 and Eclipse for your platform, if you want to use Java
  - **Copy VM image: OpenFlowTutorial-101311.zip**
  - Pass on the DVD or USB key to someone else!

- (2) Unzip OpenFlowTutorial-101311.zip

- (3) Point browser to instructions:
  - http://www.openflow.org/wk/index.php/OpenFlow_Tutorial (note the underscore)

- You should NOT need to download any large files – spare the WiFi!

Welcome

Who’s in this room?

Who's in this room?

>= 3:

- HP >= 19
- Cisco >= 15
- Juniper >= 7
- Huawei >= 6
- Brocade >= 4
- ZAO >= 3
- Google >= 3

Plus >= 60 companies w/2 or less..

Who's in this room?

- Engineer
- Senior Engineer
- Principal Systems Engineer
- Distinguished Engineer
- Senior Distinguished Engineer
- Senior Technical Marketing Engineer

Who's in this room?

- Student
- Graduate Student
- PhD / Doctor
- Senior Research Scientist
- Professor

Who's in this room?

- CTO
- VP
- Founder
- Solutions Architect
- Cloud Architect
- Chief Tech Expert
- Chief

Goals

Overall Goal:

think critically about SDN

Goals of this Tutorial

- By the end, everyone should know:
  - Knowledge about SDN/OpenFlow
    - what these are
    - how they relate
    - what's available now
    - where it's going
    - how it's used
  - SDN and You
    - how *you* can use it
    - how *you* can build on top of what's available
    - How *you* can build something completely new
- **Have fun**

Agenda

Time	Description
9:00-10:00	OpenFlow/SDN Introduction
10:00-10:30	The SDN Stack, Part 1: Switches & Controllers
10:45-12:00	The SDN Stack, Part 2: Virtualization & SDN Applications
12:00-1:00	Lunch
1:00-3:00	Controllers + Q&A
3:15-4:15	Hands-On
4:20-5:00	SDN Deployment Experiences and Wrap-up

SDN + OpenFlow History

Why OpenFlow?

[Historical motivations, for researchers]

The Networking Industry (2007)

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

Feature ..... Feature

Operating System

Specialized Packet Forwarding Hardware

Million of lines of source code — 5400 RFCs — Barrier to entry

Billions of gates — Complex — Power Hungry

Many complex functions baked into the infrastructure

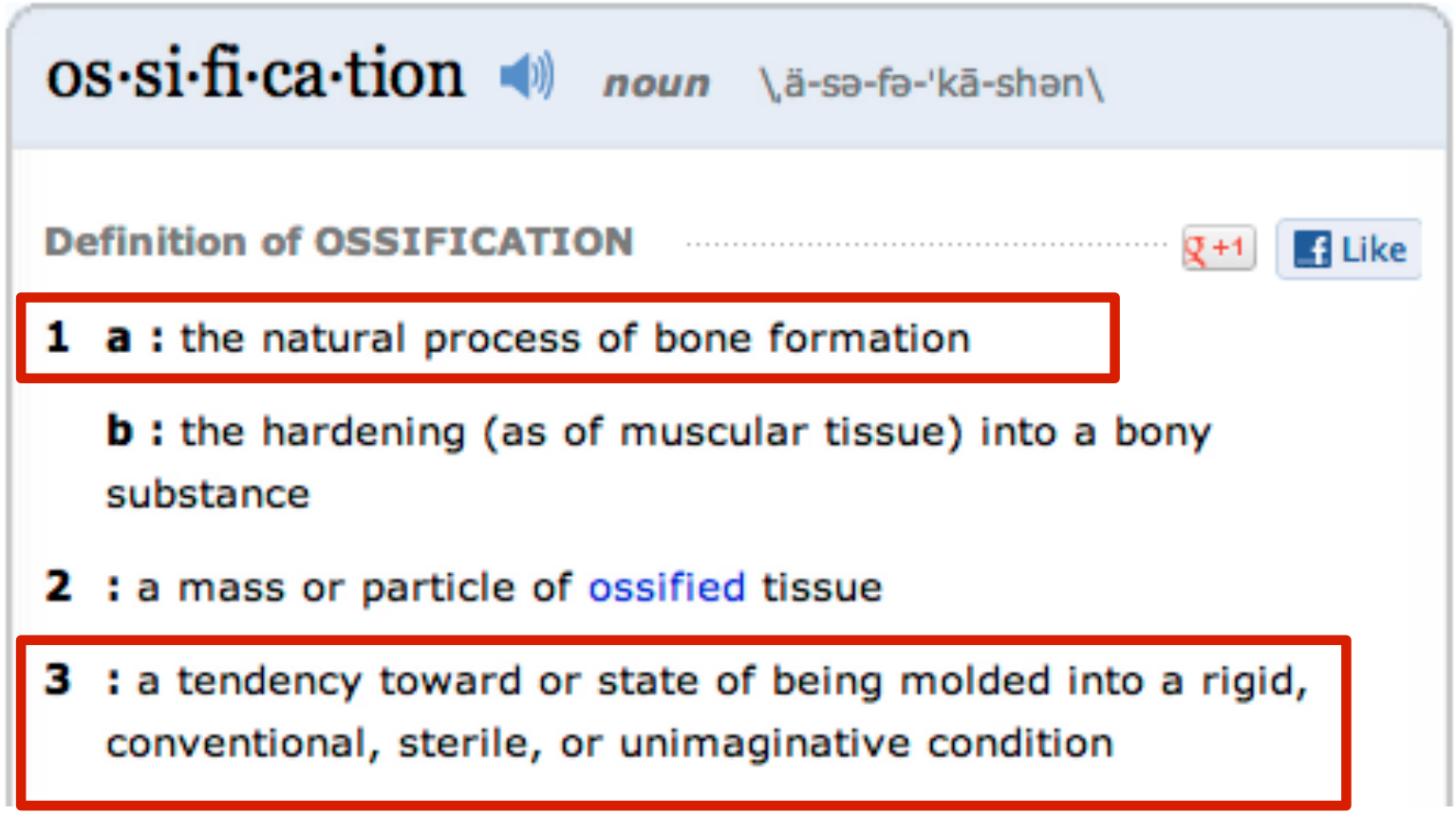
OSPF, BGP, multicast, differentiated services,
Traffic Engineering, NAT, firewalls, MPLS, redundant layers, …

An industry with a "mainframe-mentality"

Little ability for non-telco network operators to get what they want

Functionality defined by standards, put in hardware, deployed on nodes

Ossification



os·si·fi·ca·tion *noun* \ä-sə-fə-'kā-shən\

Definition of OSSIFICATION

1 **a** : the natural process of bone formation

b : the hardening (as of muscular tissue) into a bony substance

2 : a mass or particle of ossified tissue

3 : a tendency toward or state of being molded into a rigid, conventional, sterile, or unimaginative condition

Source: http://www.merriam-webster.com/

Research Stagnation (circa 2007): Faster networks but not *better* networks

- Lots of *deployed* innovation in other areas
  - OS: filesystems, schedulers, virtualization
  - DS: DHTs, CDNs, MapReduce
  - Compilers: JITs, new language paradigms
- Networks are largely the same as years ago
  - Ethernet, IP, WiFi
- Rate of change of the network seems slower in comparison
  - Need better tools and abstractions to demonstrate and deploy

Another problem:
Closed Systems (Vendor Hardware)

- Can't extend
- Stuck with interfaces (CLI, SNMP, etc)

- Hard to meaningfully extend
- Hard to meaningfully collaborate
- Vendors starting to open up, but not usefully

Open Systems

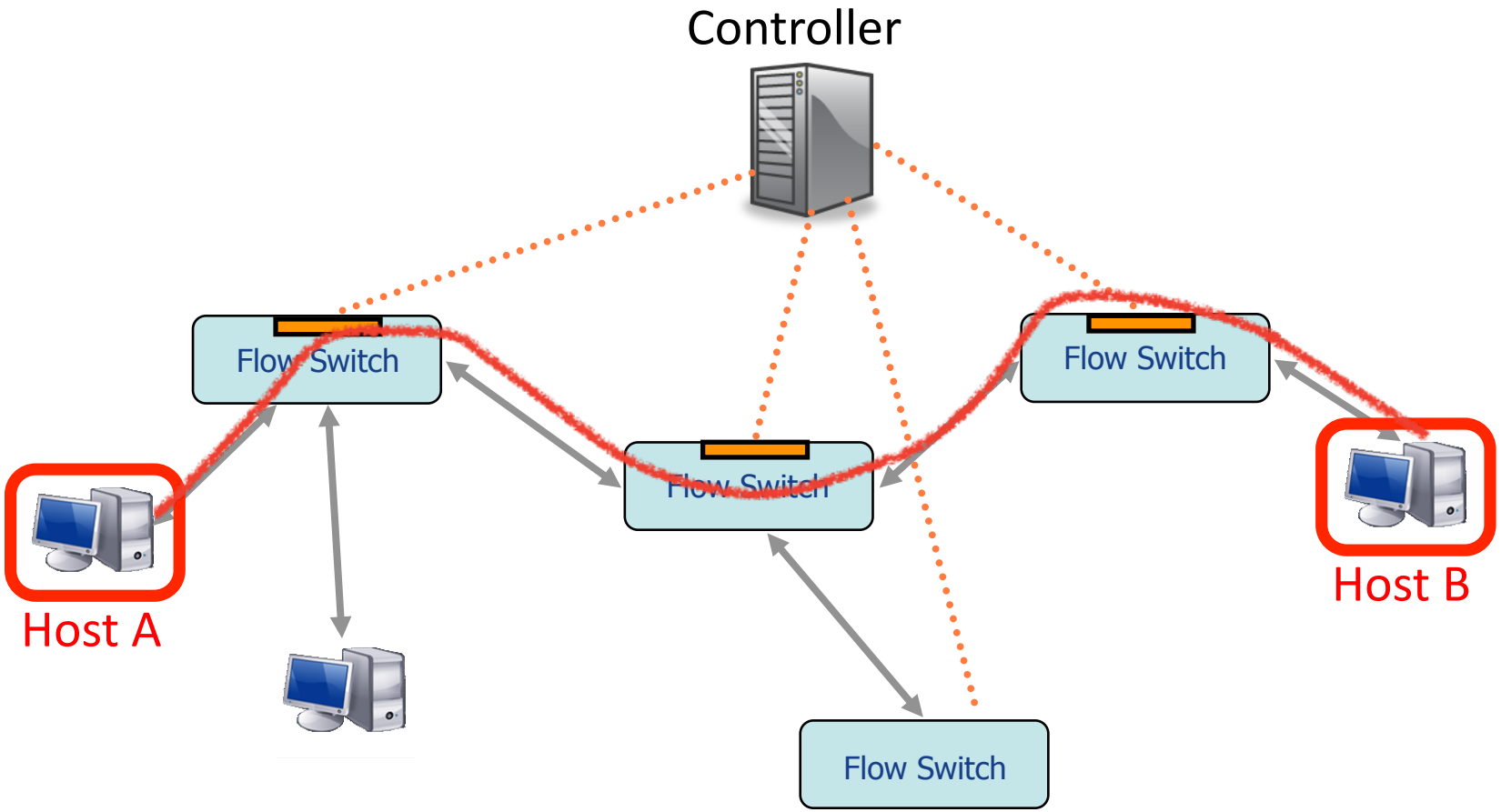
	Performance Fidelity	Scale	Real User Traffic?	Complexity	Open
Simulation	medium	medium	**no**	medium	yes
Emulation	medium	**low**	**no**	medium	yes
Software Switches	**poor**	**low**	yes	medium	yes
NetFPGA	high	**low**	yes	**high**	yes
Network Processors	high	medium	yes	**high**	yes
Vendor Switches	**high**	**high**	**yes**	**low**	**no**

gap in the tool space

none have **all** the desired attributes!

Ethane, a precursor to OpenFlow

Centralized, reactive, per-flow control



See Ethane SIGCOMM 2007 paper for details

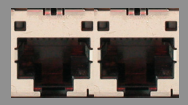
Ethane's dumb, simple switches might be more broadly useful…

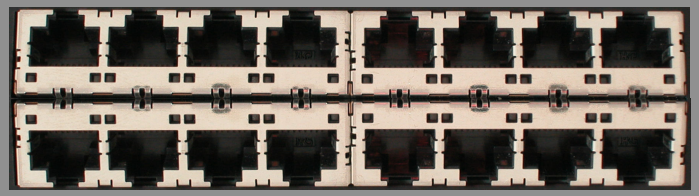
OpenFlow: a pragmatic compromise

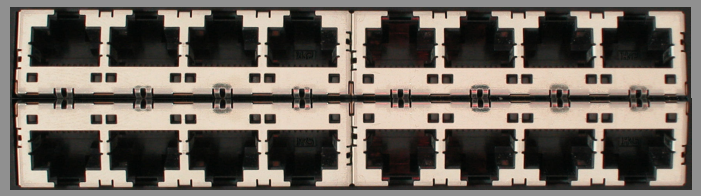
+ Speed, scale, fidelity of vendor hardware
+ Flexibility and control of software and simulation
+ Vendors don't need to expose implementation
+ Leverages hardware inside most switches today (ACL tables)
- Least-common-denominator interface may prevent using all hardware features
- Limited table sizes
- Switches not designed for this
- New failure modes to understand

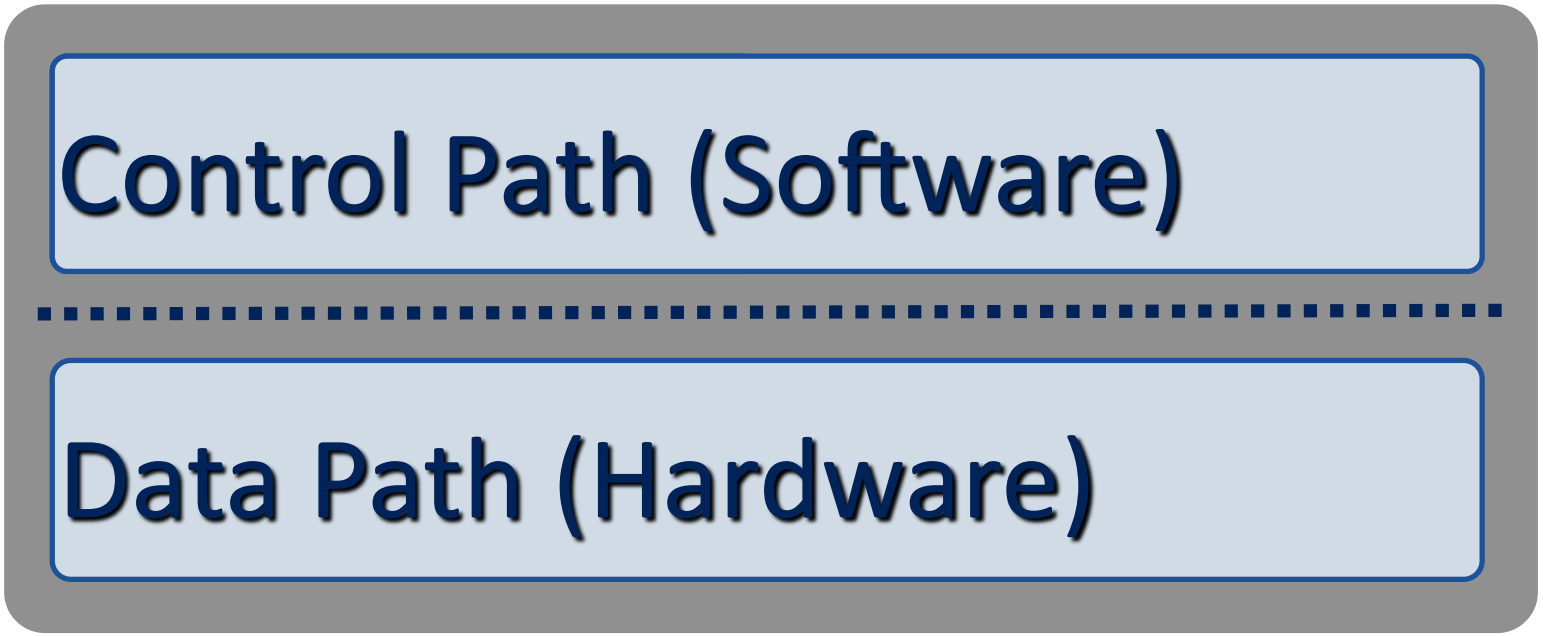
How does OpenFlow work?

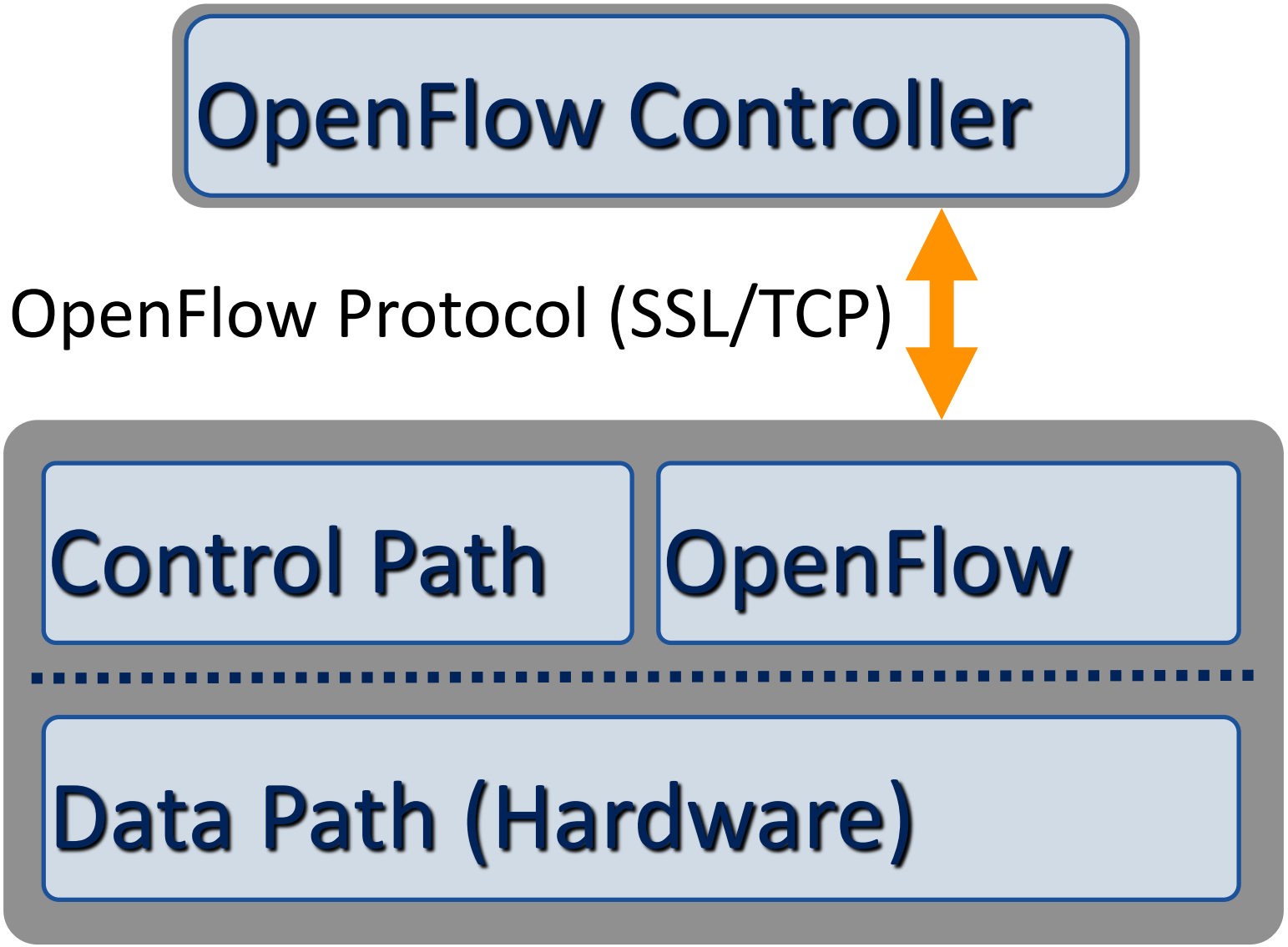
Ethernet Switch



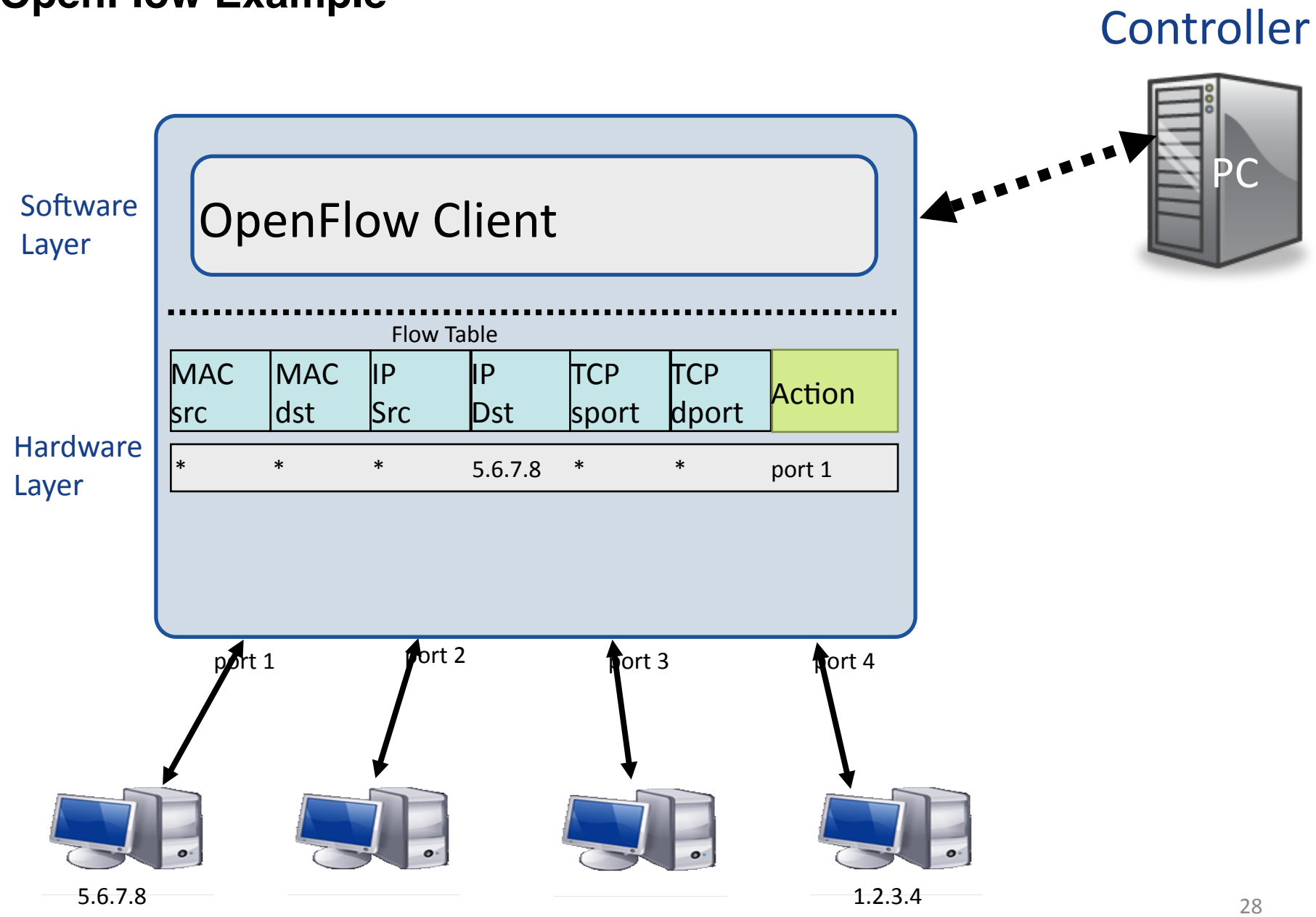





Control Path (Software)
Data Path (Hardware)

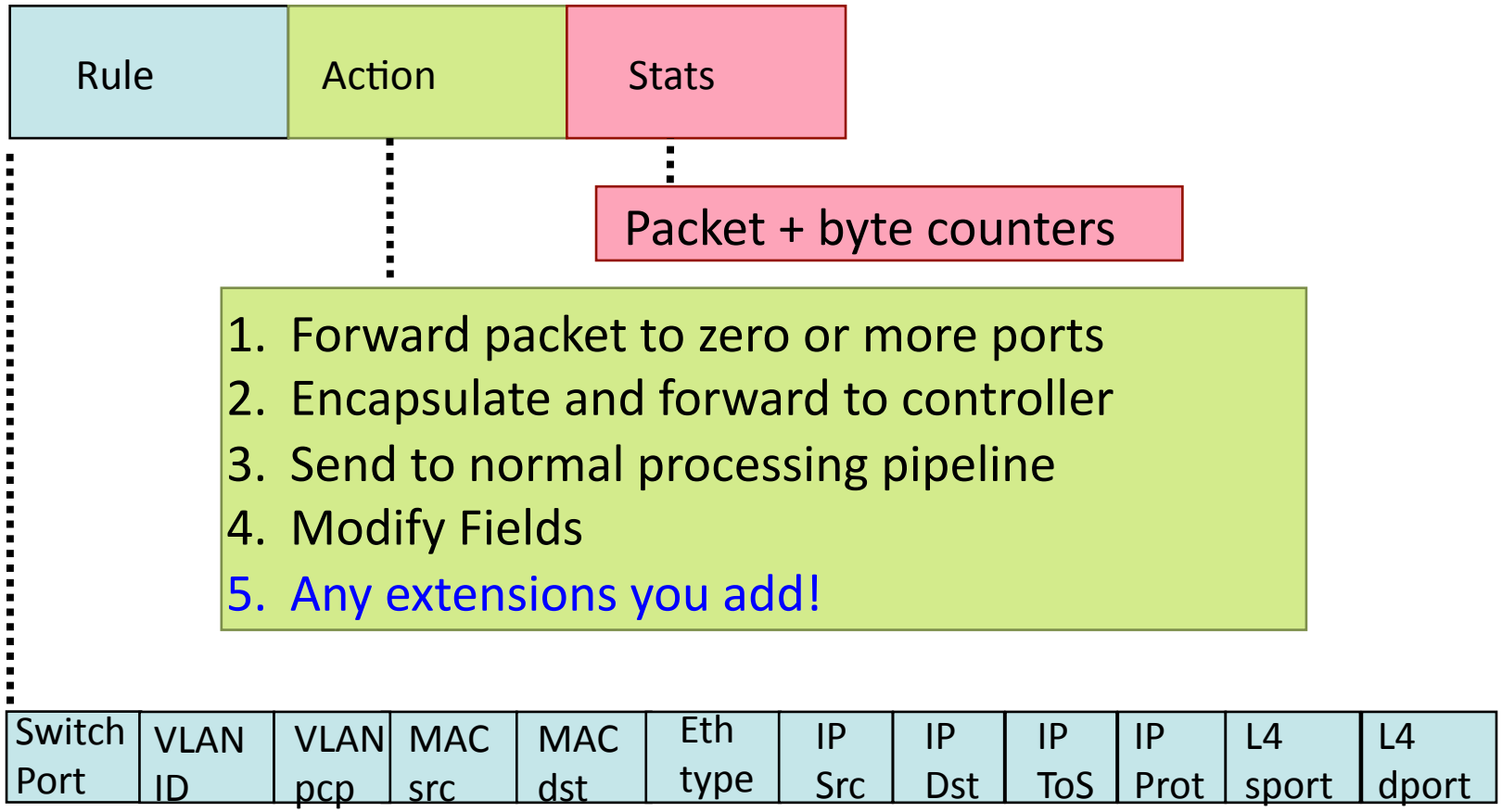

OpenFlow Controller
OpenFlow Protocol (SSL/TCP)
Control Path
OpenFlow
Data Path (Hardware)

OpenFlow Example



OpenFlow Basics

Flow Table Entries



+ 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

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

OpenFlow is not enough.

OpenFlow is not enough…

- Adds the ability to modify, experiment…
- But still harder than it should be to add features to a network
- Effectively assembly programming or an ISA

[OpenFlow is just a forwarding table management protocol]

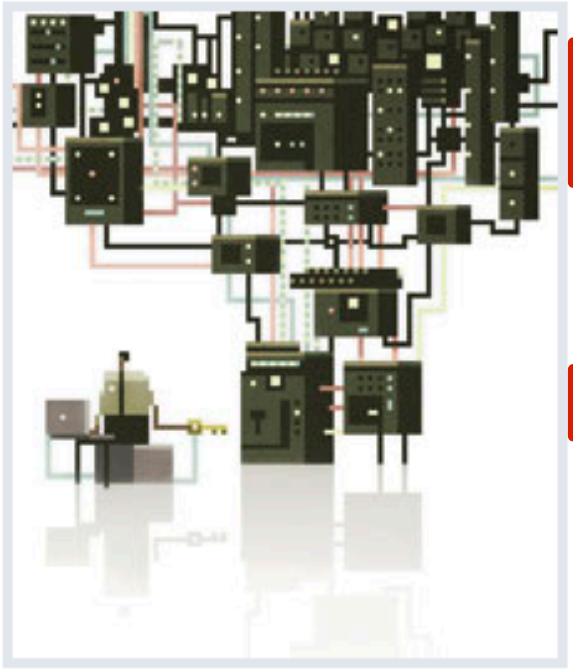
It's hard to add a feature to a network

- It's not just that we lack access to line-rate forwarding that we can control
- Fully distributed algorithms are hard, especially when defined at the protocol level
- Your protocol must implement its own mechanisms
- Must work on constrained and heterogeneous resources

This is where Software-Defined Networking comes in: need a control plane abstraction too

From OpenFlow to SDN

technology review
Published by MIT



Superbrothers

10 EMERGING TECHNOLOGIES

TR10: Software-Defined Networking

Nick McKeown believes that remotely controlling network hardware with software can bring the Internet up to speed.

MARCH/APRIL 2009 BY KATE GREENE

Audio »

For years, computer scientists have dreamed up ways to improve networks' speed, reliability, energy efficiency, and security. But their schemes have generally remained lab projects, because it's been impossible to test them on a large enough scale to see if they'd work: the routers and switches at the core of the Internet are locked down, their software the intellectual property of companies such as Cisco and Hewlett-Packard.

Frustrated by this inability to fiddle with Internet routing in the real world, Stanford

SDN is actually much older than ‘09

- Key ideas present in some form in 4D, RCP, Sane, Ethane, …

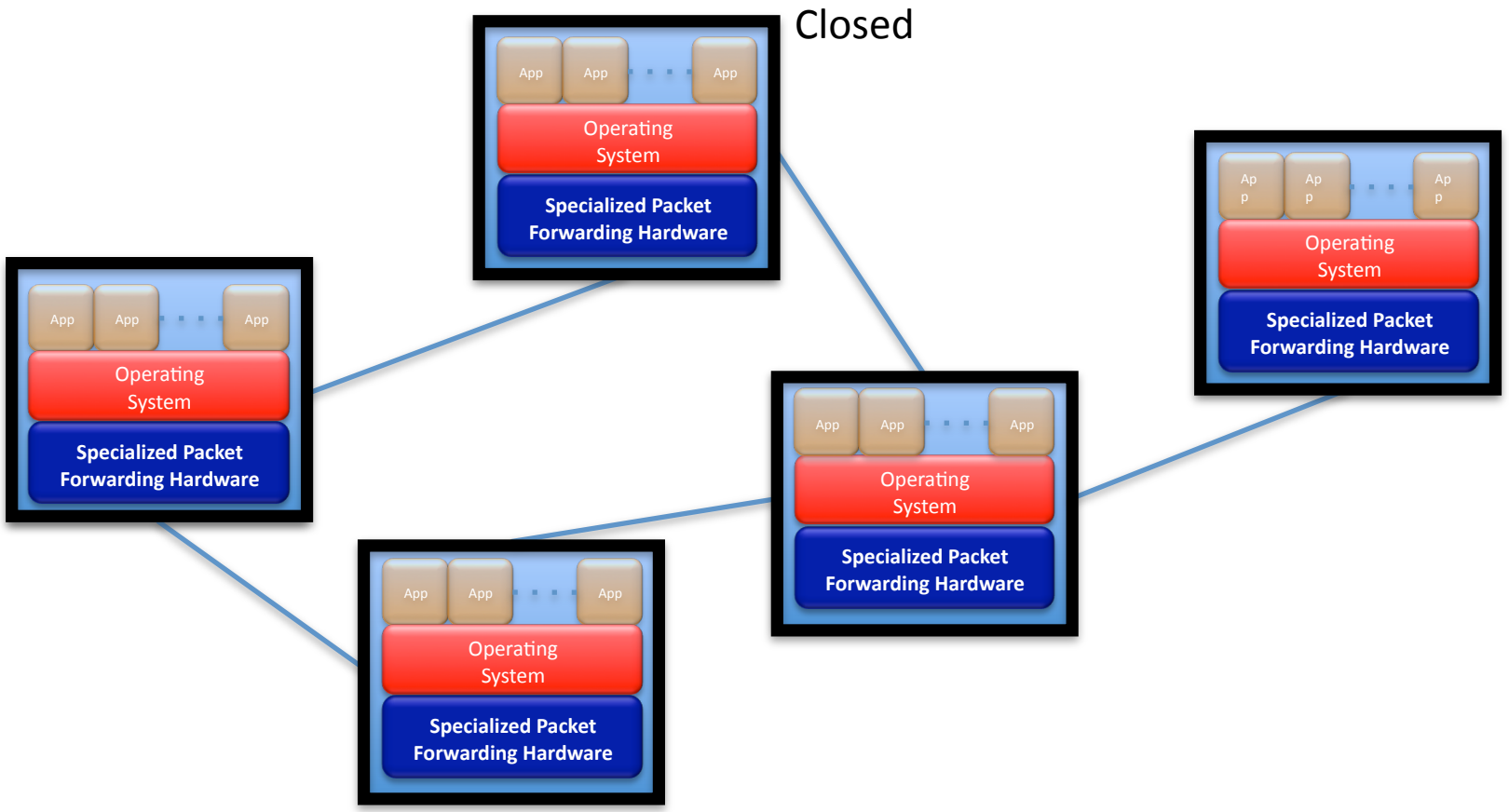
What is SDN, opt. 1

"The McKeown View":

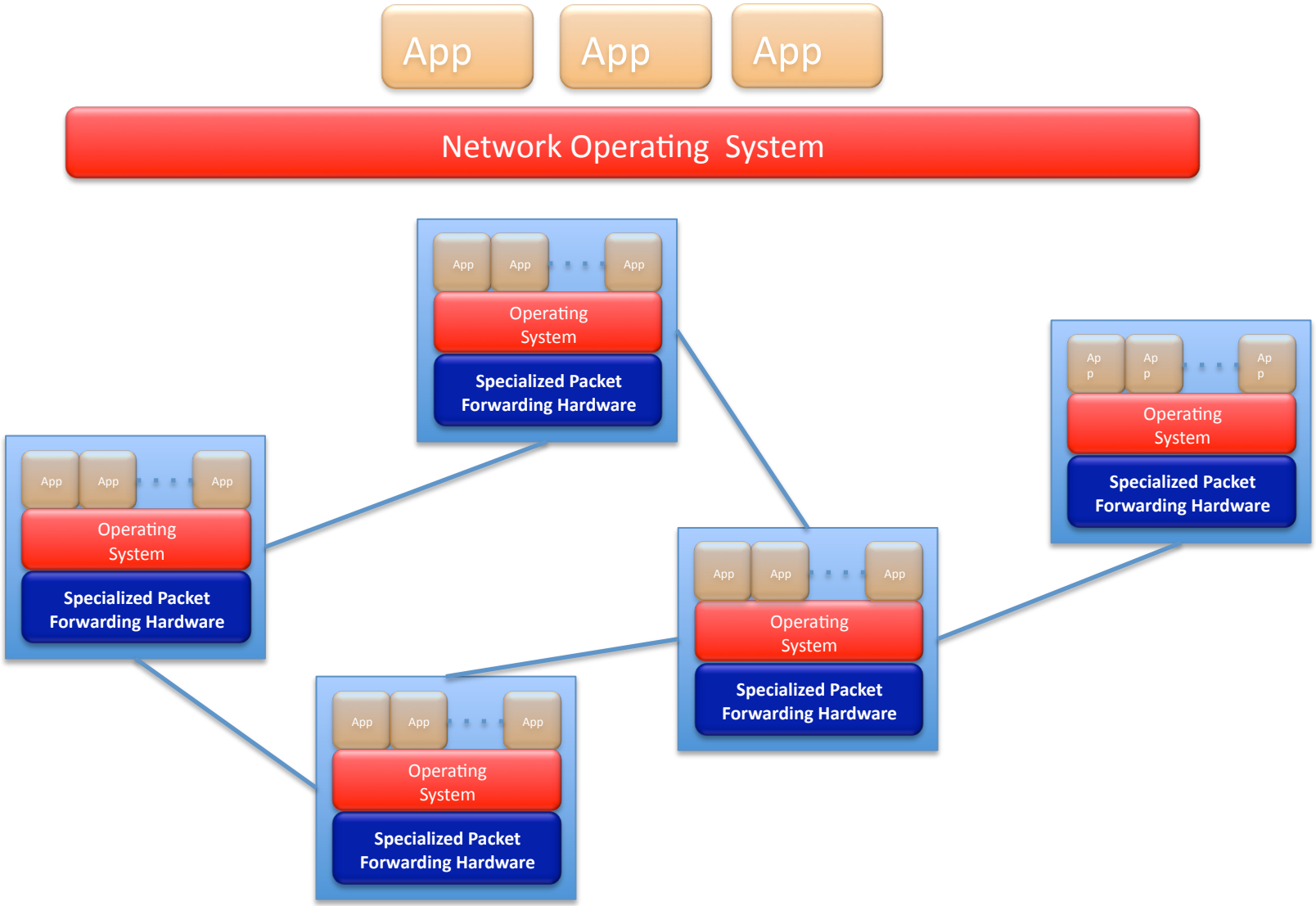
Refactoring Functionality

Define SDN by its placement of functionality.

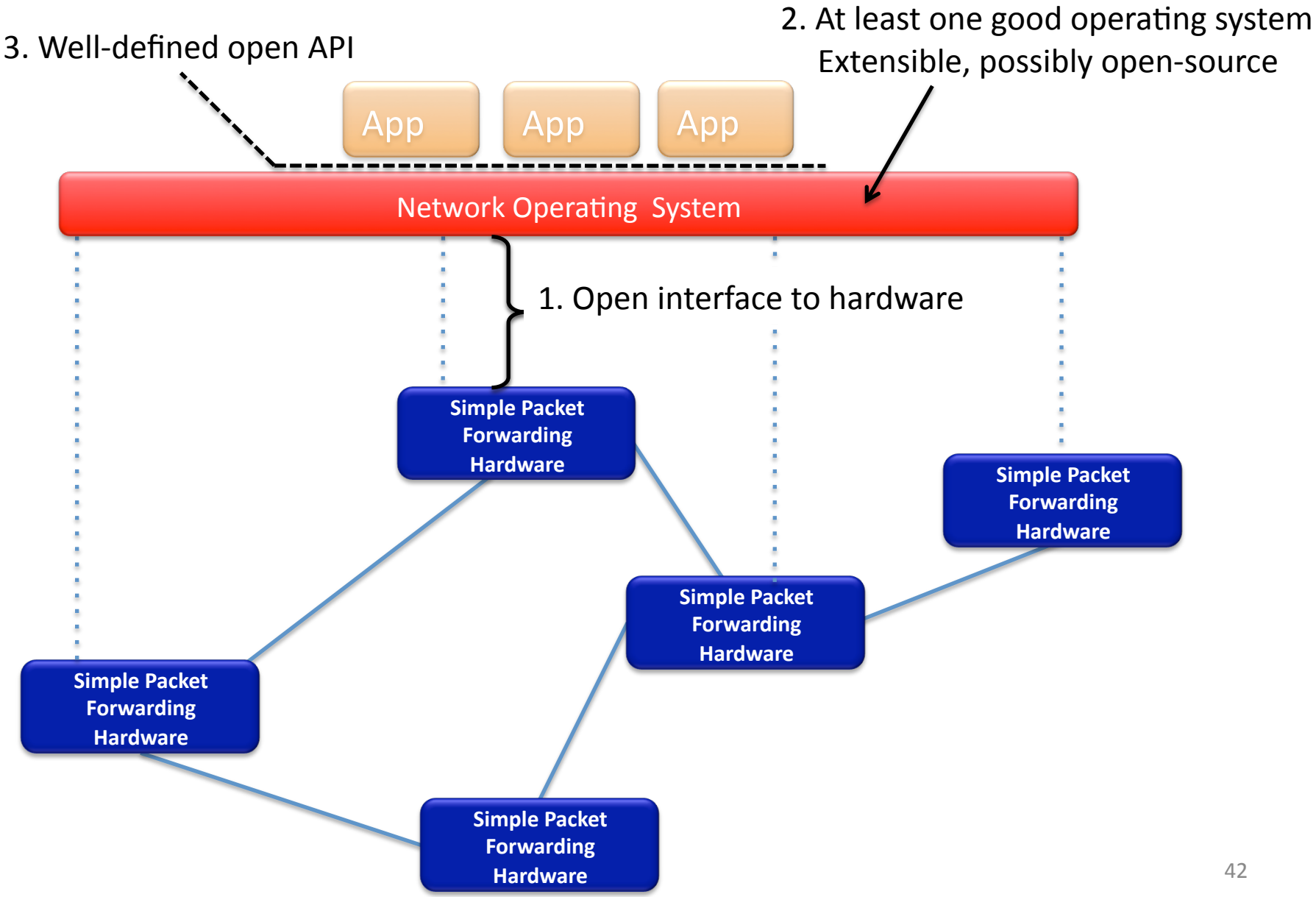
Today
Closed Boxes, Fully Distributed Protocols

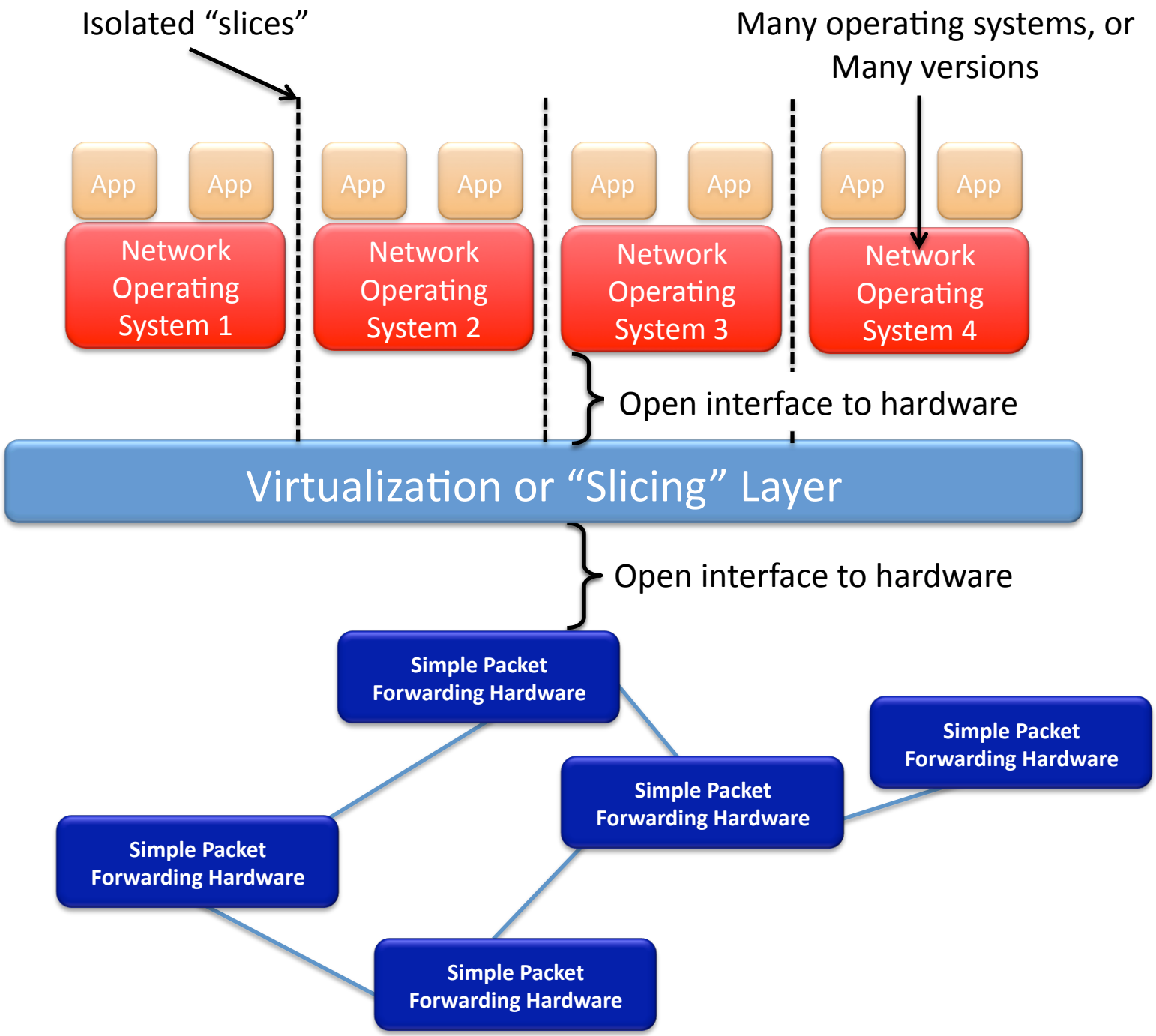


“Software Defined Networking” approach to open it



The "Software-defined Network"




Isolated “slices”
Many operating systems, or
Many versions
App
App
App
App
App
App
App
App
Network Operating System 1
Network Operating System 2
Network Operating System 3
Network Operating System 4
Open interface to hardware
Virtualization or “Slicing” Layer
Open interface to hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware

What is SDN, opt. 2

"The Shenker View":

Redefining Abstractions

Define SDN by the abstractions it provides to software (and people writing it).

“The Shenker View”

- Scott Shenker has a killer presentation
  - (Keynote at last ONS)
  - You should watch this
  - http://www.slideshare.net/martin_casado/sdn-abstractions
  - The Future of Networking, and the Past of Protocols

- Many bullet points on next few slides are from this talk

“The Shenker View”: the gist

- Network control planes need abstractions
  - Abstractions solve architectural problems and enable evolvability
  - Today’s layers (L2, L3, ..) are good abstractions but terrible for control interfaces
- Networks work because we can master complexity
  - but what we should be doing is extracting simplicity, with the right abstractions

Programming Made the Transition

- Machine Languages: no abstractions
- Higher-level languages, OS + other abstraction
  - files, virtual memory, data structures, …
- Modern languages: even more abstractions
  - objects, garbage collection, threads, locks, …

Abstractions simplify programming: they make it easier to write, maintain, and reason about programs.

Could networking follow this same path?

Forwarding Abstraction

- Forwarding behavior specified by a control program.
- Possibilities: x86, MPLS, OpenFlow

State Distribution Abstraction

- Control program should not have to handle 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

Short version: programs operate on graphs

Specification Abstraction

- Give control program abstract view of network
- Provide enough detail to specify goals, but not to implement them

What is SDN, opt 3

My view:

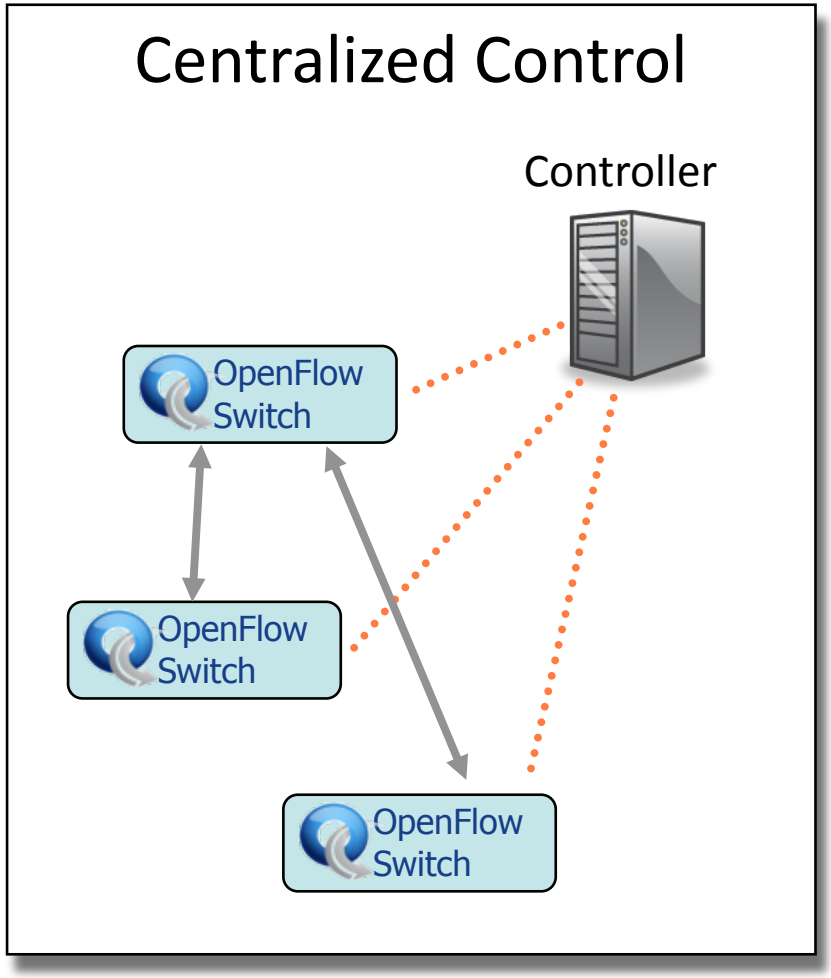
Opening Up
Design Axes

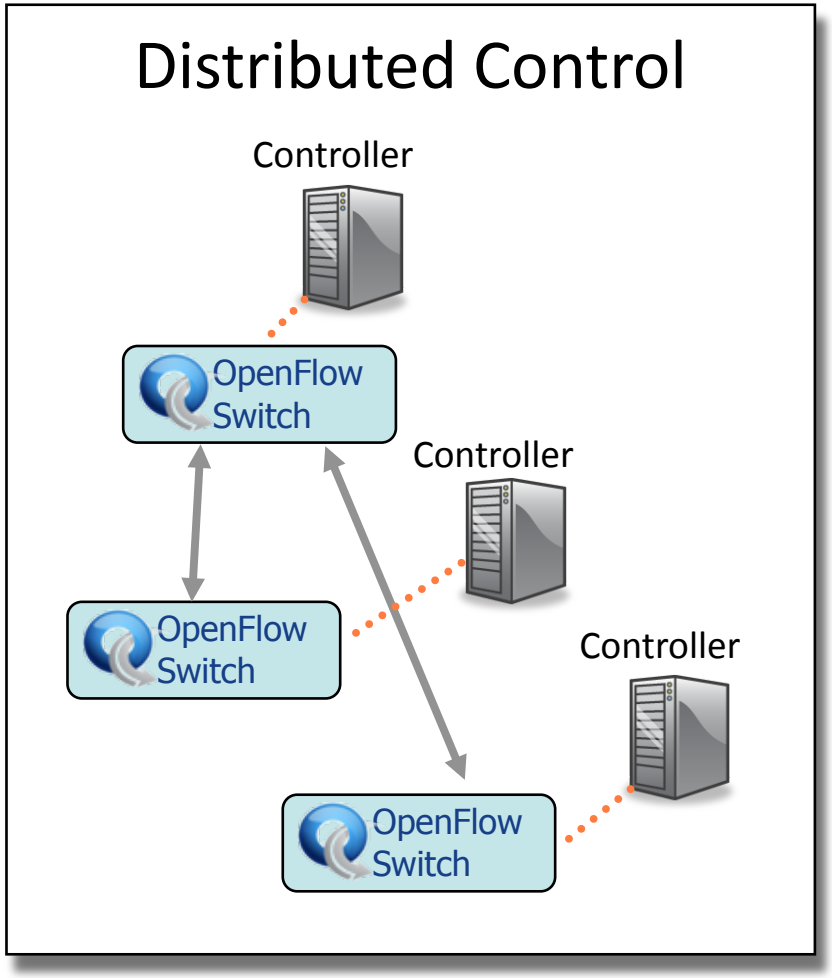
Define SDN not by what it looks like or how we think about it, but the flexibility it provides.

SDN opens up implementation design axes.

An SDN is *any* network that gives us the flexibility to choose between points on the following design axes.

Centralized vs Distributed Control





Microflow vs. Aggregated

Microflow

- Every flow is individually set up by controller
- Exact-match flow entries
- Flow table contains one entry per flow

- Good for fine grain control, policy, and monitoring, e.g. campus

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)

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
- Extremely simple fault recovery

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

Virtual vs Physical

Virtual

- Assumes configurable switching within a host: in the OS or hypervisor
- Software! Memory, processing, arbitrary modifications
- Massive flow rates
- Limited to the hardware below

Physical

- No assumption of software changes; unmodified end hosts
- Greater control over expensive forwarding resources

Fully Consistent vs Eventually Consistent

Fully Consistent

- Certainty about state
- Consistent state is harder to scale
- Easier to reason about state and its transitions
- May eliminate route flaps

Eventually Consistent

- Uncertainty about state now, but eventually converges
- Probabilistic state is easier to scale
- Introduces the possibility of long-lived route flaps and unstable control systems

Here’s a picture

These axes, today (BGP):

Many choices arise from assuming decentralized administration…

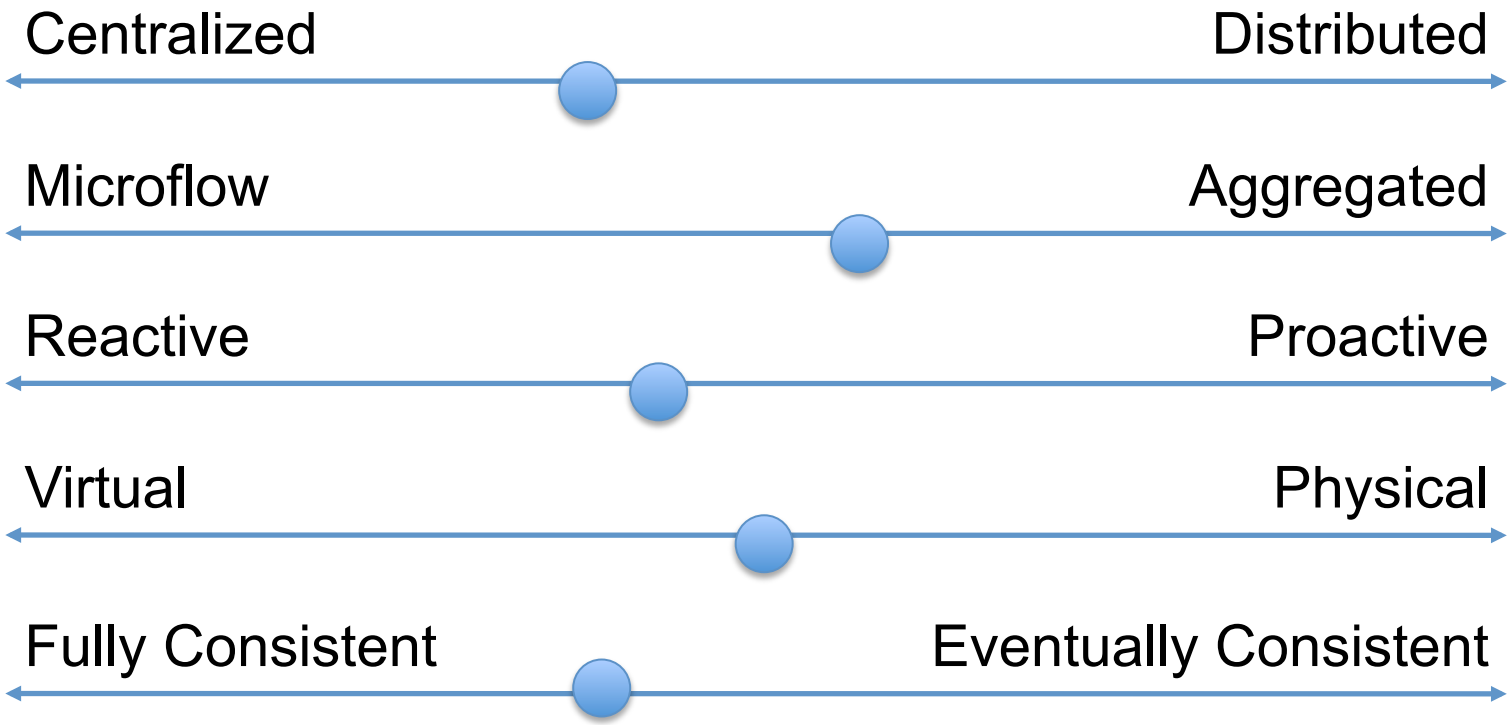
These axes, for Ethane

Many choices arise from assuming centralized administration…

These axes, for Hedera:

Hedera controls a data center network.
See NSDI ‘10 paper for more details.

Your SDN network or application?



Hybrid approaches often have desirable properties; can use these now.

Example applications and their places on the scales

High-order bit of SDN:

adds flexibility to control-plane implementation choices

SDN examples using OpenFlow

Many applications benefit from flexibility

Stanford Demos

- Wireless mobility
- VM mobility/migration
- Network virtualization
- Power management
- Hardware
- Load balancing
- Traffic Engineering

Others

- Removing spanning tree
- Network visualization
- Network debugging
- Packet-circuit convergence
- Home networks
- Flexible access control
- Scale-out routers
- Scale-out data centers

What SDN really means is up in the air.

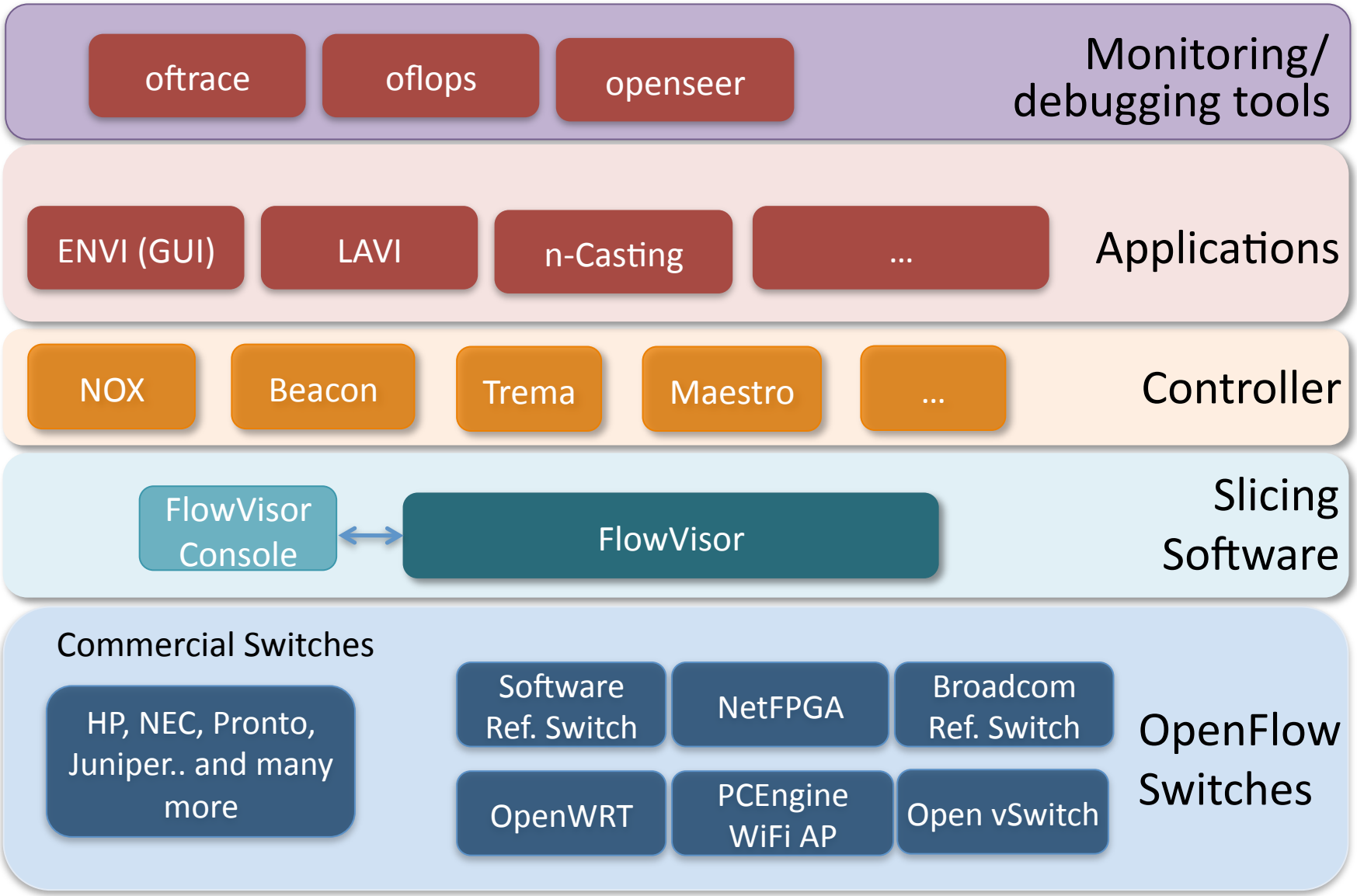
Here's a good definition, though:

Software Defined Networking (SDN) is a refactoring of the relationship between network devices and the software that controls them.

[Paraphrased from the HotSDN '12 Solicitation]

The SDN Stack

The SDN Stack



Switches (all OpenFlow 1.0 for now)

Current SDN hardware (as of ~2010)

Juniper MX-series

NEC IP8800

WiMax (NEC)

HP Procurve 5400

Netgear 7324

PC Engines

Pronto 3240/3290

Ciena Coredirector

Ask your vendors

Commercial Switches

Vendor	Models	Virtualize?	Notes	Pic
HP ProCurve	5400zl, 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/ 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	

Open Switches

Name	Lang	Platform(s)	License	Original Author	Notes
OpenFlow Reference	C	Linux	OpenFlow License	Stanford/ Nicira	not designed for extensibility
Open vSwitch	C/ Python	Linux/BSD?	Apache 2.0	Ben Pfaff/ Nicira	In Linux kernel 3.3+
Indigo	C/Lua	Linux-based Hardware Switches	GPL v2	Dan Talayco/ BigSwitch	Bare OpenFlow switch
Xorplus	?	Linux-based Hardware Switches	?	Pica8	Support

Ignore these slides. Ask your vendor for the latest.

What you cannot do with OpenFlow v1.0

- Non-flow-based (per-packet) networking
  - ex. Per-packet next-hop selection (in wireless mesh)
  - yes, this is a fundamental limitation
  - BUT OpenFlow can provide the plumbing to connect these systems
- Use all tables on switch chips
  - yes, a major limitation (cross-product issue)
  - BUT OF version 1.1 exposes these, providing a way around the cross-product state explosion

What can cannot do with OpenFlow v1.0 (2)

- New forwarding primitives
  - BUT provides a nice way to integrate them through extensions
- New packet formats/field definitions
  - BUT extensible matches are coming in v1.2/1.3+
- Optical Circuits
  - BUT efforts underway to apply OpenFlow model to circuits
- Low-setup-time individual flows
  - BUT can push down flows proactively to avoid delays

Where it's going

- 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

Controllers

Open Controllers

Name	Lang	Platform(s)	License	Original Author	Notes
OpenFlow Reference	C	Linux	OpenFlow License	Stanford/ Nicira	not designed for extensibility
NOX	Python, C++	Linux	GPL	Nicira	actively developed
Beacon	Java	Win, Mac, Linux, Android	GPL (core), FOSS Licenses for your code	David Erickson (Stanford)	runtime modular, web UI framework, regression test framework
Maestro	Java	Win, Mac, Linux	LGPL	Zheng Cai (Rice)	
Trema	Ruby, C	Linux	GPL	NEC	includes emulator, regression test framework
RouteFlow	?	Linux	Apache	CPqD (Brazil)	virtual IP routing as a service

Open Controllers (2)

Name	Lang	Platform(s)	License	Original Author	Notes
OpenFaucet	Python				Library
Mirage	OCaml				
POX	Python	Any			
Floodlight	Java	Any		BigSwitch, based on Beacon	

Too many to easily list of keep track of…

Growing list….

- Martin Casado recently added a list:
- http://yuba.stanford.edu/~casado/of-sw.html

List of OpenFlow Software Projects (that I know of)

(I am trying to keep a running list of all OpenFlow-related software projects where either the bits or the source are available online. If you know of one that I'm missing, please e-mail me and I'll include it)

Switch Software and Stand-Alone OpenFlow Stacks

Open vSwitch: (C/Python) Open vSwitch is a an OpenFlow stack that is used both as a vswitch in virtualized environments and has been ported to multiple hardware platforms. It is now part of the Linux kernel (as of 3.3).

OpenFlow Reference: (C) The OpenFlow reference implementation is a minimal OpenFlow stack that tracks the spec.

Pica8: (C) An open switch software platform for hardware switching chips that includes an L2/L3 stack and support for OpenFlow.

Indigo: (C) Indigo is a for-hardware-switching OpenFlow implementation based on the Stanford reference implementation.

Pantou: (C) Pontou is an OpenFlow port to the OpenWRT wireless environment.

OpenFaucet: (Python) OpenFaucet is a pure Python implementation of the OpenFlow 1.0.0 protocol, based on Twisted. OpenFaucet can be used to implement both switches and controllers in Python.

OpenFlowJ: (Java) OpenFlow stack written in Java.

Oflib-node: (Javascript) Oflib-node is an OpenFlow protocol library for Node. It converts between OpenFlow wire protocol messages and Javascript objects.

Nettle: (Haskell) OpenFlow library written in Haskell.

Controller Platforms

NOX: (C++/Python) NOX was the first OpenFlow controller.

POX: (Python) Pox as a general SDN controller that supports OpenFlow. It has a high-level SDN API including a queriable topology graph and support for virtualization.

Jaxon: (Java) Jaxon is a NOX-dependent Java-based OpenFlow Controller.

Trema: (C/Ruby) Trema is a full-stack framework for developing OpenFlow controllers in Ruby and C.

Related Research

- DIFANE
  - Rule partitioning for controller-less flow insertion
- UCSD Fat Tree Series: Scalable Commodity Data Center, PortLand, Hedera
  - Scale-out data centers that use OpenFlow
- Tesseract
  - Centralized WAN in the 4D Architecture
- ONIX
  - Fault-tolerant controller platform from Nicira, Google, NEC
- DevoFlow
  - Practical scalability limits to OpenFlow and modifications to get around them

Related Research

- Frenetic/Nettle
  - Functional Reactive Programming for more composable, reusable controller code
- Resonance
  - State-machine-based network control
- Consistency Primitives
  - Per-packet or per-flow routing guarantees to simplify network versioning

Up Next:

- Come back at **10:45** for:

"The SDN Stack Part 2: Virtualization and SDN Applications"

Getting Started

- (1) Copy to your hard disk from a USB Key or DVD:
  - Copy needed files (VirtualBox, terminal, possibly an X server) for your platform (Win/Mac/Linux)
  - Copy Java 6 and Eclipse for your platform, if you want to use Java
  - **Copy VM image: OpenFlowTutorial-101311.zip**
  - Pass on the DVD or USB key to someone else!

- (2) Unzip OpenFlowTutorial-101311.zip

- (3) Point browser to instructions:
  - http://www.openflow.org/wk/index.php/OpenFlow_Tutorial (note the underscore)

- You should NOT need to download any large files – spare the WiFi!

Virtualization and SDN Applications

Virtualization

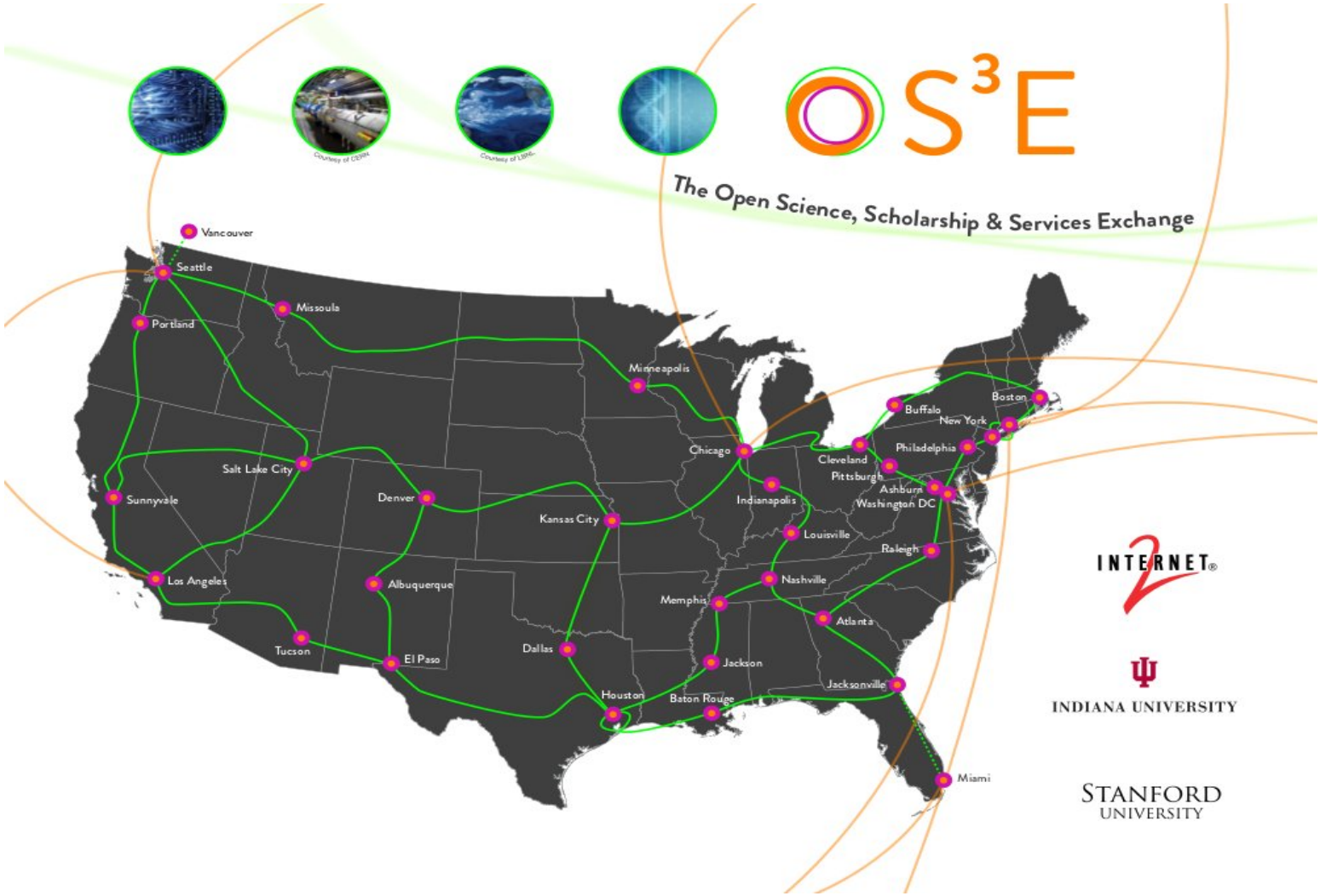
Rob Sherwood, (BigSwitch)

Current Trials

- 68 trials/deployments spanning 13 countries

Internet2 OpenFlow deployment initiative.

35+ 100G POPs, nationwide.

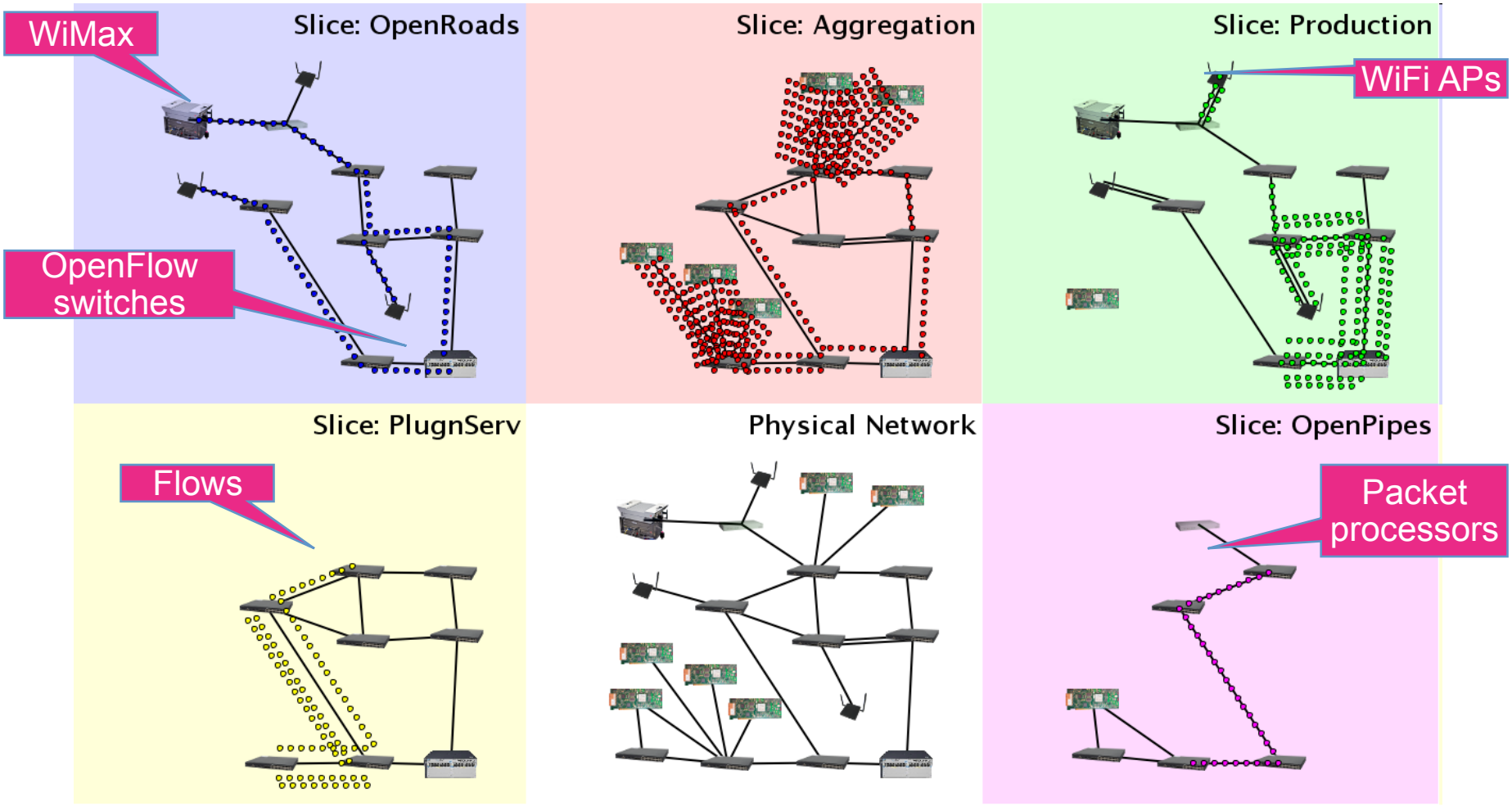


Applications of SDN

OpenFlow Demonstration Overview

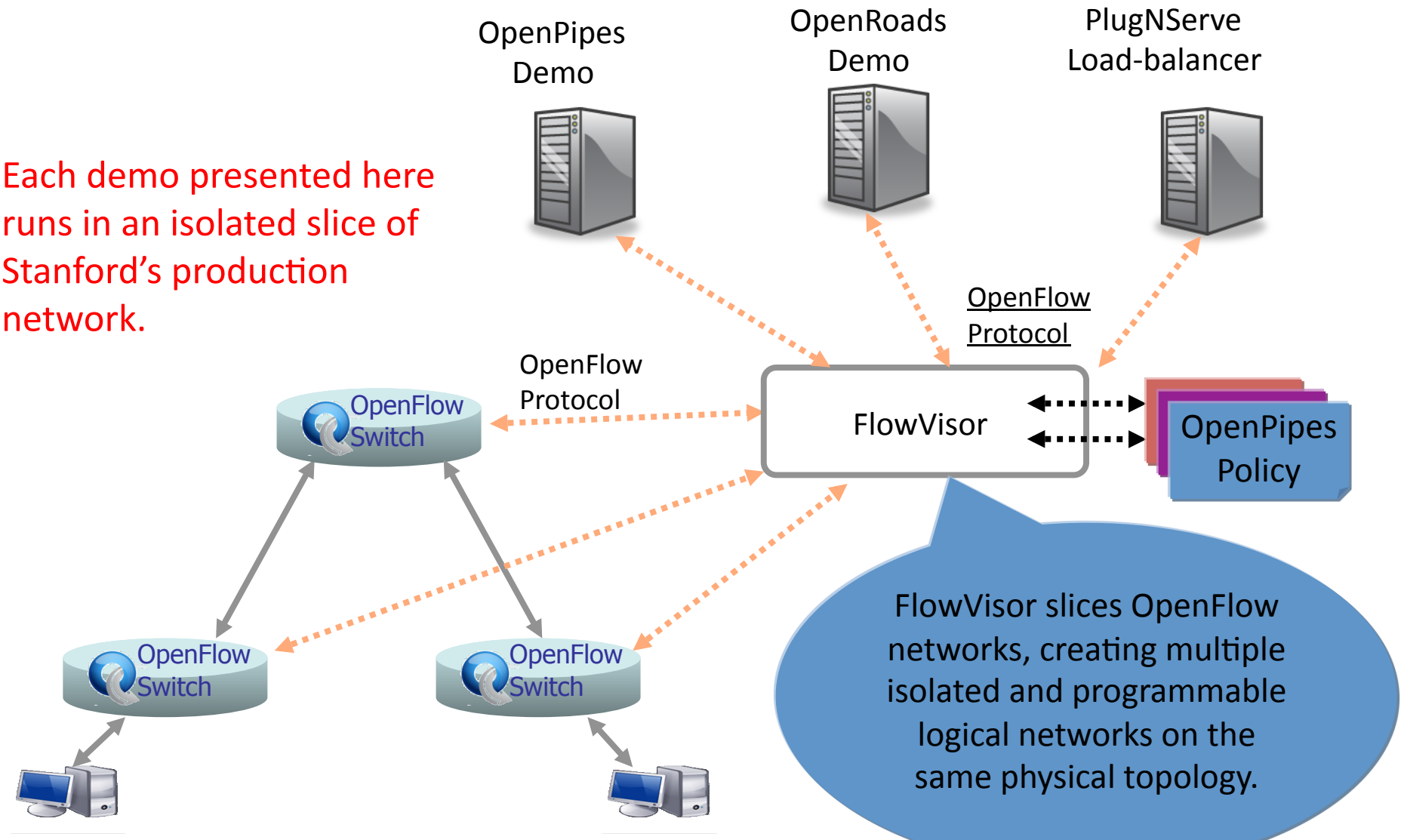
Topic	Demo
Network Virtualization	FlowVisor
Hardware Prototyping	OpenPipes
Load Balancing	PlugNServe
Energy Savings	ElasticTree
Mobility	MobileVMs
Traffic Engineering	Aggregation
Wireless Video	OpenRoads

Demo Infrastructure with Slicing



FlowVisor Creates Virtual Networks

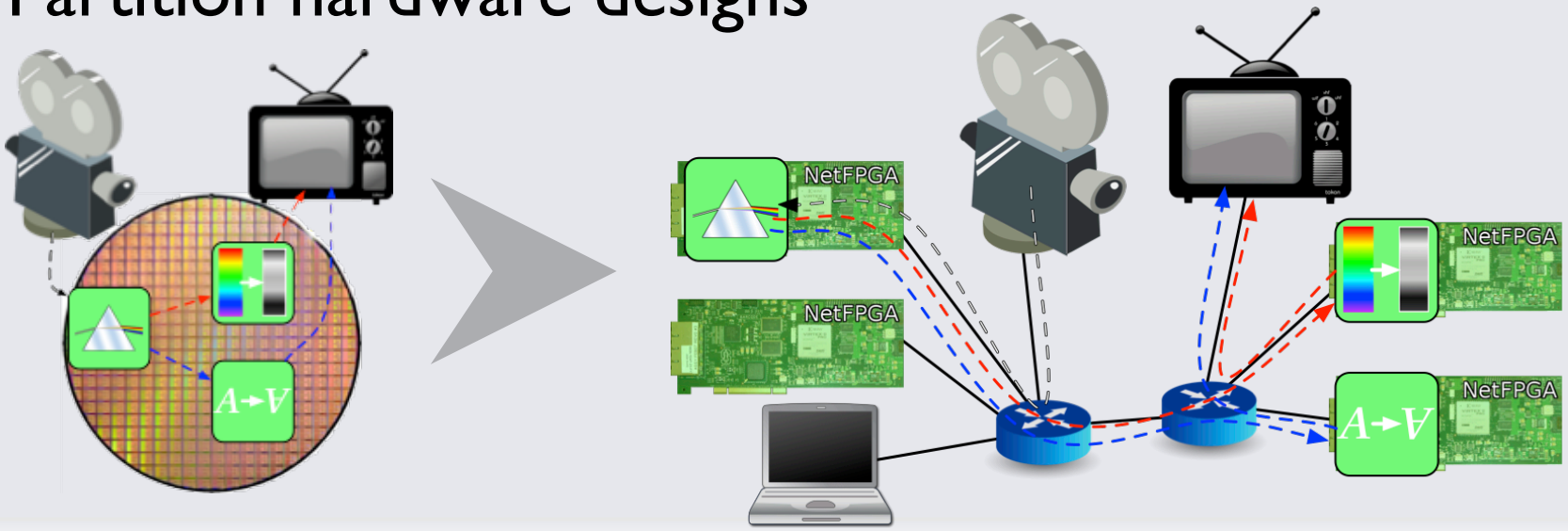
Each demo presented here runs in an isolated slice of Stanford's production network.



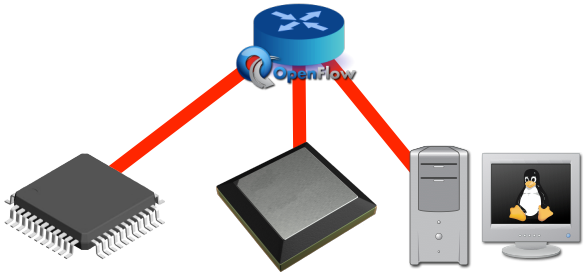
OpenPipes

•Plumbing with OpenFlow to build hardware systems

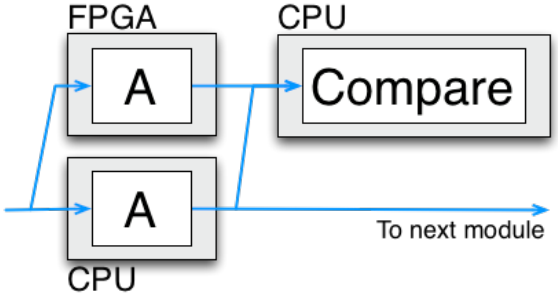
Partition hardware designs



Mix resources

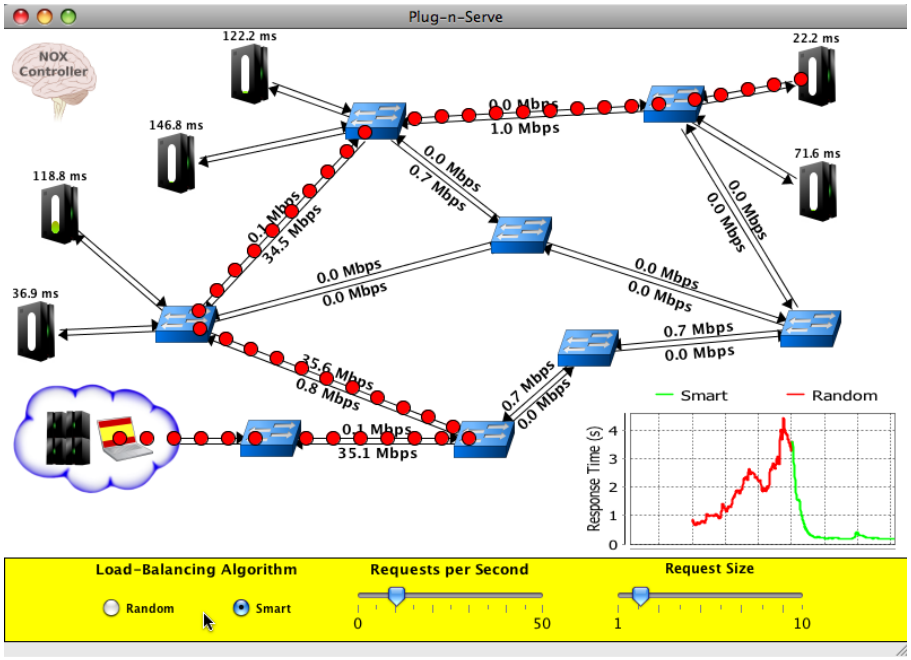


Test



Plug-n-Serve:
Load-Balancing Web Traffic using OpenFlow

Goal: Load-balancing requests in unstructured networks



What we are showing

- OpenFlow-based distributed load-balancer
  - Smart load-balancing based on network and server load
  - Allows incremental deployment of additional resources

OpenFlow means…

- Complete control over traffic within the network
  - Visibility into network conditions
  - Ability to use existing commodity hardware

This demo runs on top of the FlowVisor, sharing the same physical network with other experiments and production traffic.

Dynamic Flow Aggregation on an OpenFlow Network

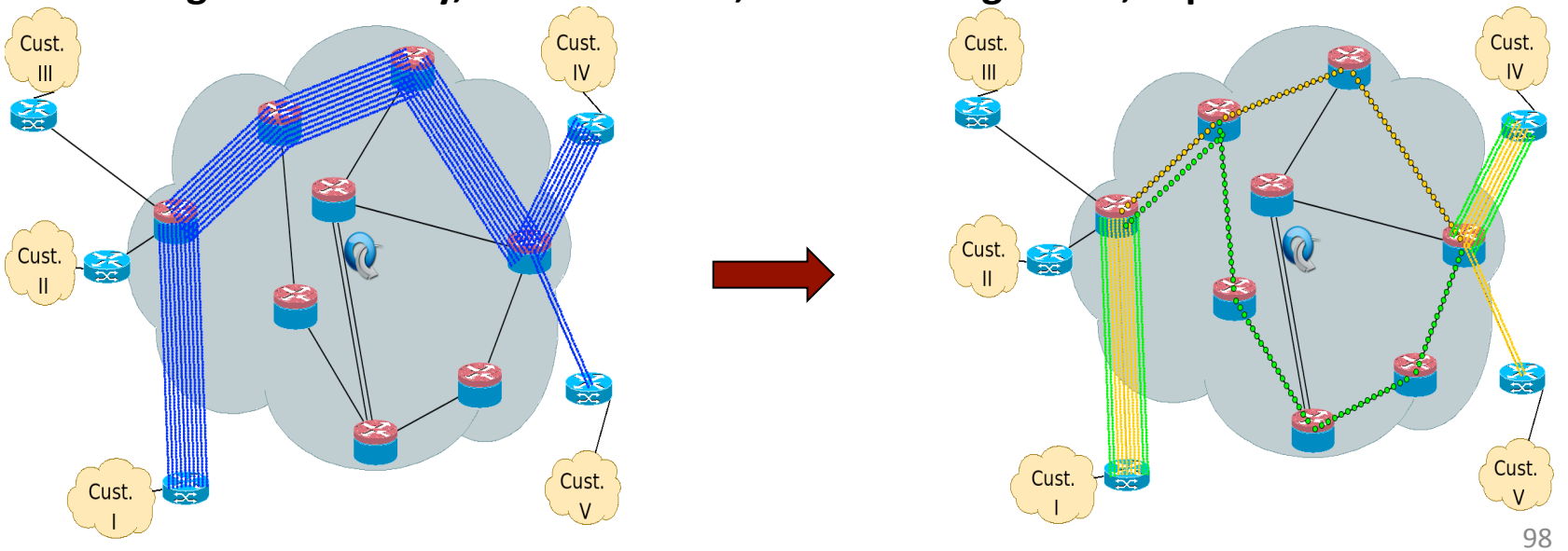
Scope

- Different Networks want different flow granularity (ISP, Backbone,…)
- Switch resources are limited (flow entries, memory)
- Network management is hard
- Current Solutions : MPLS, IP aggregation

How OpenFlow Helps?

- Dynamically define flow granularity by wildcarding arbitrary header fields
- Granularity is on the switch flow entries, no packet rewrite or encapsulation
- Create meaningful bundles and manage them using your own software (reroute, monitor)

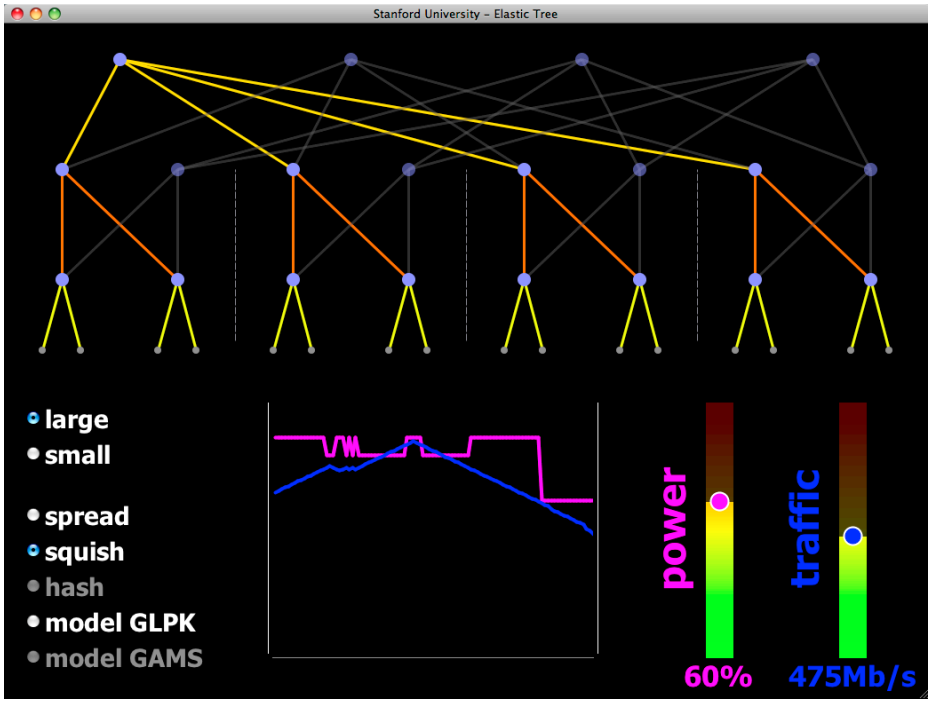
Higher Flexibility, Better Control, Easier Management, Experimentation



ElasticTree:

Reducing Energy in Data Center Networks

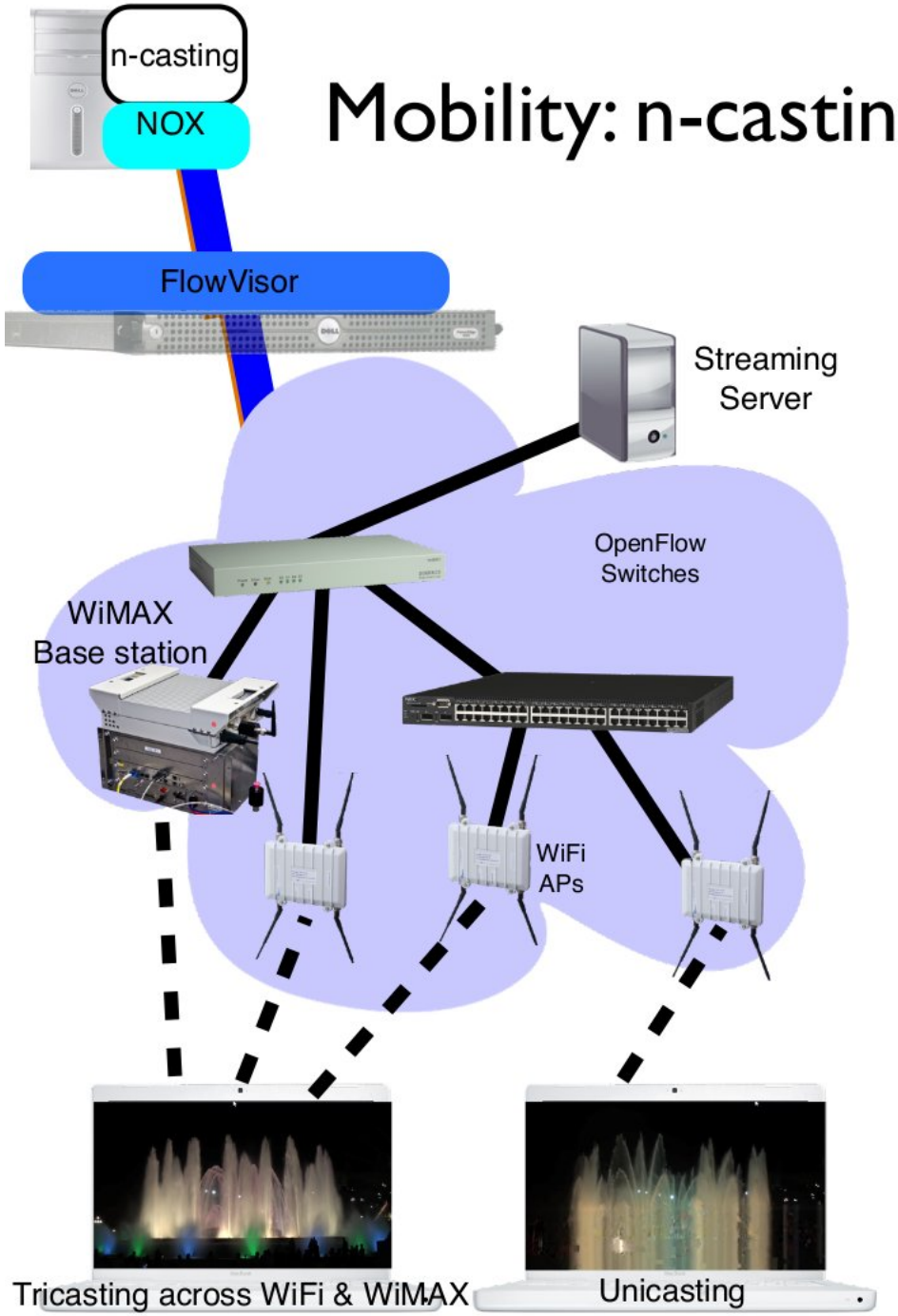
- Shuts off links and switches to reduce data center power
- Choice of optimizers to balance power, fault tolerance, and BW
- OpenFlow provides network routes and port statistics



- The demo:
- Hardware-based 16-node Fat Tree
- Your choice of traffic pattern, bandwidth, optimization strategy
- Graph shows live power and latency variation

demo credits: Brandon Heller, Srini Seetharaman, Yiannis Yiakoumis, David Underhill

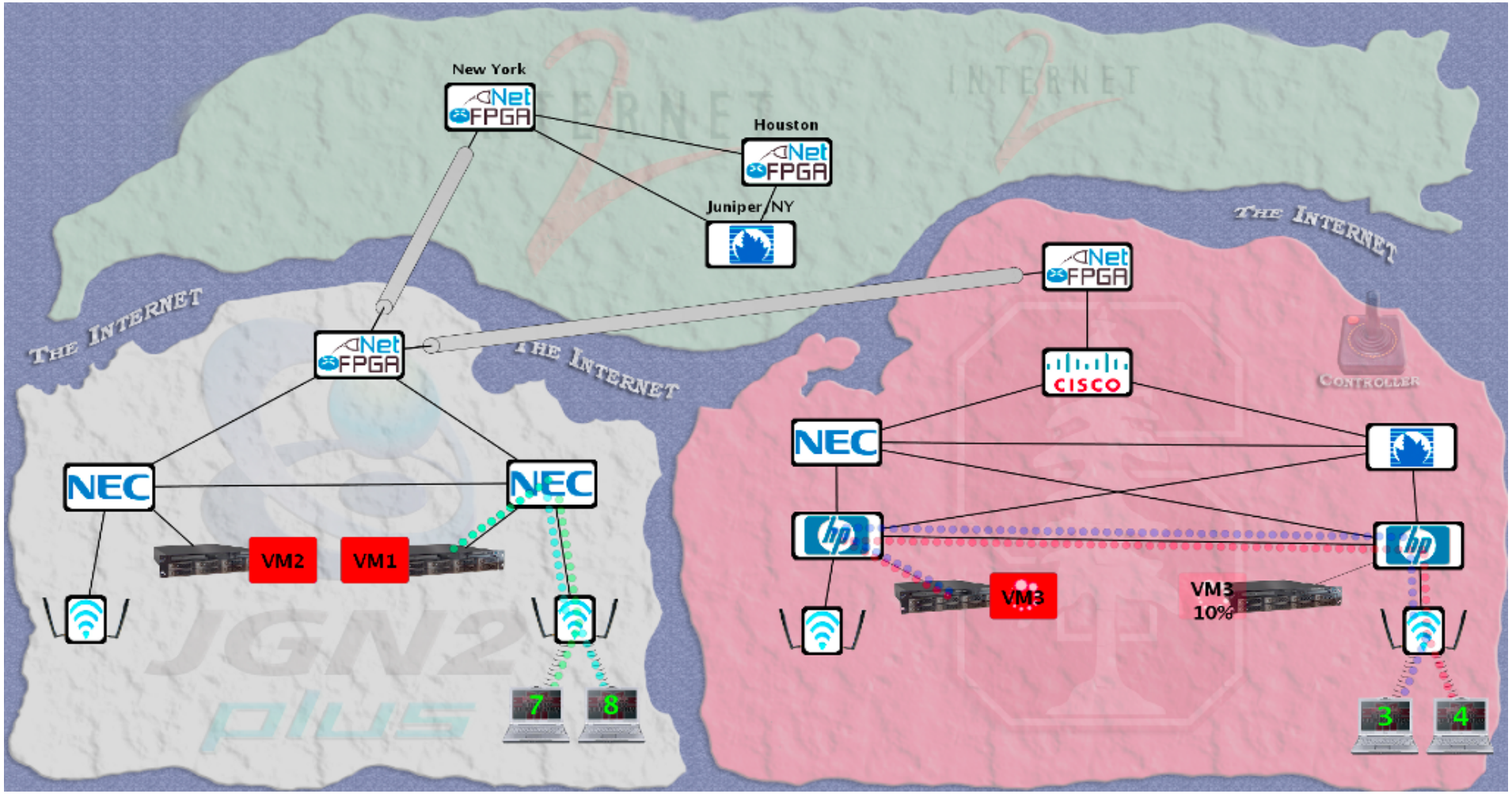
Mobility: n-casting with OpenFlow



- Demonstrate what flexibility of routing enables in mobile networks
- Show how technology agnostic handover can be easily achieved
- Customized network services for applications, devices and technologies
- Simplify control and services
- Unified control for wireline and wireless networking equipments

- Demonstration: n-casting
  - Reroute flows between WiFi and WiMAX without additional logic
  - n-casting provided over for video streaming where application handles duplication well
  - coded in 227 lines of C/C++

Intercontinental VM Migration



Moved a VM from Stanford to Japan without changing its IP.

VM hosted a video game server with active network connections.

OpenFlow

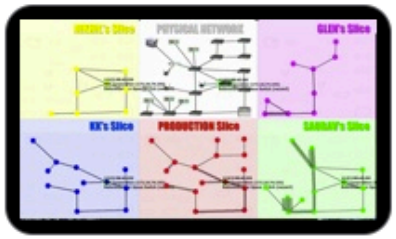
Google™ Custom Se Search

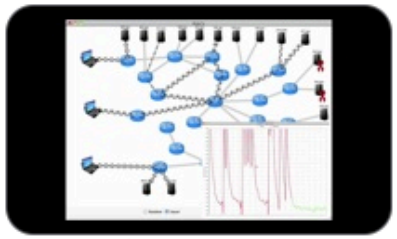
Home | Videos | Documents | News | Research | About

Videos of Research Demos

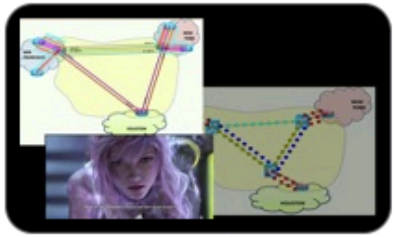
These videos demonstrate different research experiments that build on top of OpenFlow. If you have similar videos that demonstrate your research and are interested in hosting them here, please contact Nikhil Handigol.

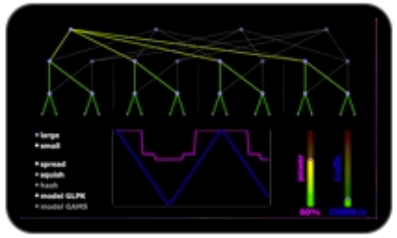
Introduction

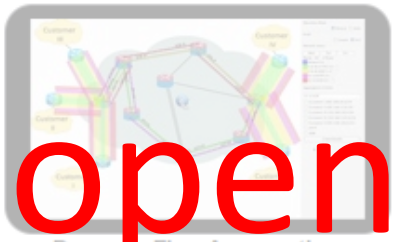

FlowVisor Demo


Aster*x: Load-Balancing as a Network Primitive


Using All Wireless Networks Around Me


Packet and Circuit Network Convergence


ElasticTree: Reducing Energy in Data Center Networks


Dynamic Flow Aggregation in an OpenFlow Network


Open Pipes: Hardware System Design with OpenFlow


Providing QoS Services with OpenFlow

openflow.org/videos

Quick Navigation

- OpenFlow Specs
- Bug Tracking
- Wiki
- Legal
- Log in

OpenFlow White Paper

Download the OpenFlow Whitepaper (PDF)

OpenFlow Specification

Download v1.1.0 Implemented (PDF)

GEC-9 Demos

Play Videos:
Wireless Handover
Load balancing

Times are changing

- No longer “what can I now do that I couldn’t do before”

- Clouds
- Cost efficiency
- Manageability
- Programmability

Where SDN make sense (and doesn't)

[from Martin Casado's blog: http://networkheresy.wordpress.com/2011/11/17/is-openflowsdn-good-at-forwarding/]

So the question really boils down to whether the algorithm needed to compute the datapath state is easily distributed ... if the state management algorithm has any of the following properties it probably isn't, and is therefore a potential candidate for SDN.

- It is not amenable to being split up into many smaller pieces (as opposed to fewer, larger instances). The limiting property in these cases is often excessive communication overhead between the control nodes.
- It is not amenable to running on heterogeneous compute environments. For example those with varying processor speeds and available memory.

Where SDN make sense (and doesn't)

[from Martin Casado's blog: http://networkheresy.wordpress.com/2011/11/17/is-openflowsdn-good-at-forwarding/]

- It is not amenable to relatively long RTT's for communication between distributed instances. In the purely distributed case, the upper bound for communicating between any two nodes scales linearly with the longest loop free path.
- The algorithm requires sophisticated distributed coordination between instances (for example distributed locking or leader election)

SDN stack

- Monitoring/Debugging Tools
- Applications
- Controllers
- Virtualization
- Switches
- Topology

Up Next:

1:00 - 3:00pm Controllers:

- NOX/POX (Murphy)
- Trema (Hideyuki)
- Beacon (David)
- Floodlight (Rob)
- RouteFlow (video)
- + more Q&A

Q & A

Getting Started

- (1) Copy to your hard disk from a USB Key or DVD:
  - Copy needed files (VirtualBox, terminal, possibly an X server) for your platform (Win/Mac/Linux)
  - Copy Java 6 and Eclipse for your platform, if you want to use Java
  - **Copy VM image: OpenFlowTutorial-101311.zip**
  - Pass on the DVD or USB key to someone else!

- (2) Unzip OpenFlowTutorial-101311.zip

- (3) Point browser to instructions:
  - http://www.openflow.org/wk/index.php/OpenFlow_Tutorial (note the underscore)

- You should NOT need to download any large files – spare the WiFi!

Break

- Come back at **1:00** for Controllers

Controllers

POX:

Murphy McCauley

Beacon:

David Erickson

Trema:

Hideyuki Shimonishi

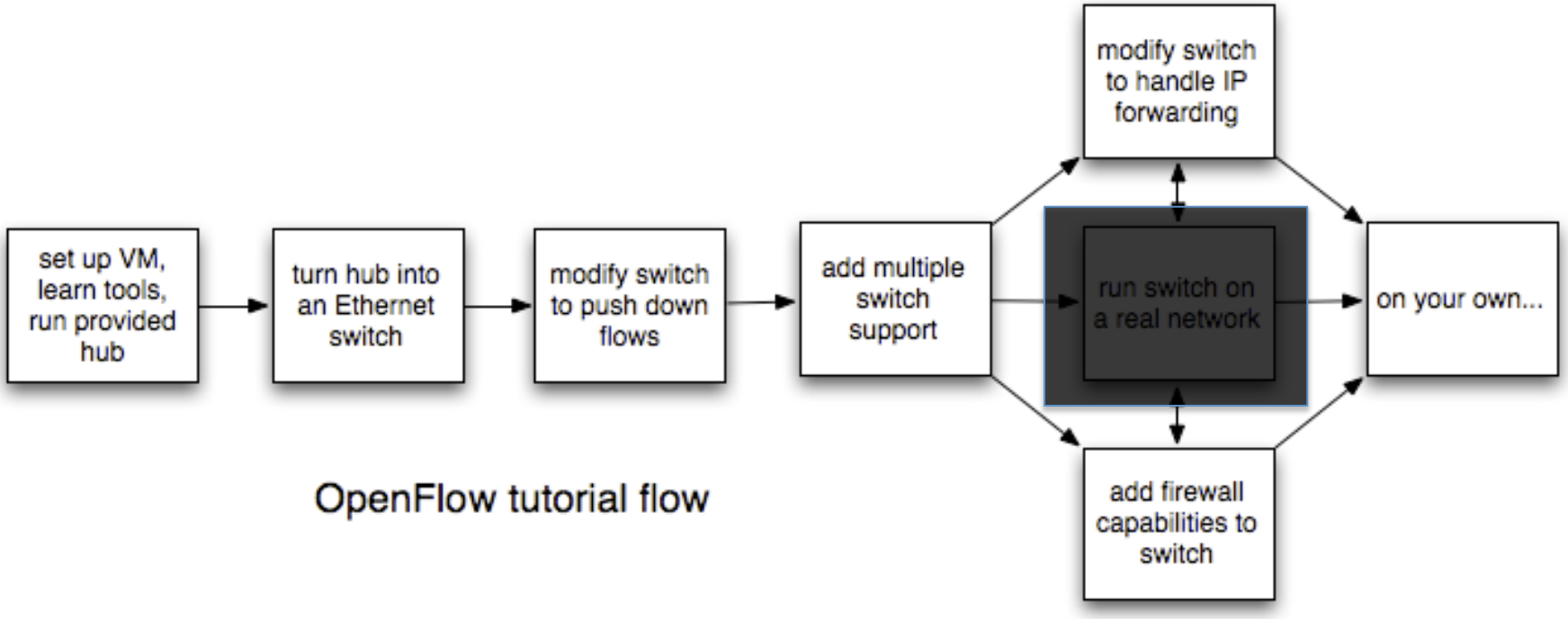
Floodlight:

Rob Sherwood

VM Setup/ Overview

[Hands-on Tutorial] Overview

TutorialFlow



OpenFlow tutorial flow

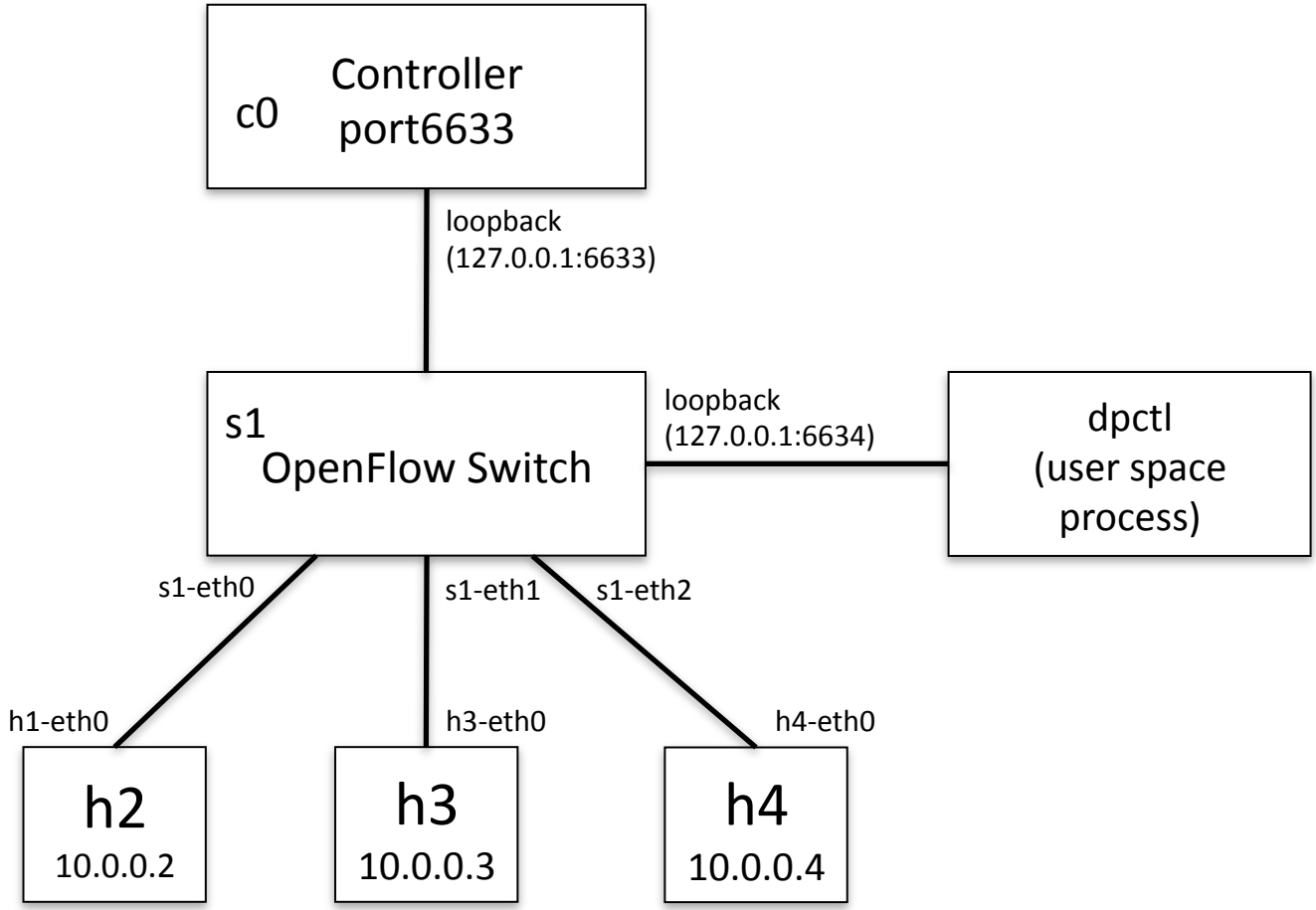
Stuff you'll use

- Ref Controller / NOX / Beacon / Trema / Floodlight
- Open vSwitch
- Mininet

- iperf
- tcpdump
- Wireshark

Tutorial Setup

OpenFlow Tutorial
3hosts-1switch
topology



virtual hosts

Pick a controller!

POX:

Murphy McCauley

Beacon:

David Erickson

Trema:

Hideyuki Shimonishi

Floodlight:

Rob Sherwood

Hands-on Tutorial

Instructions at:

www.openflow.org/wk/index.php/OpenFlow_Tutorial

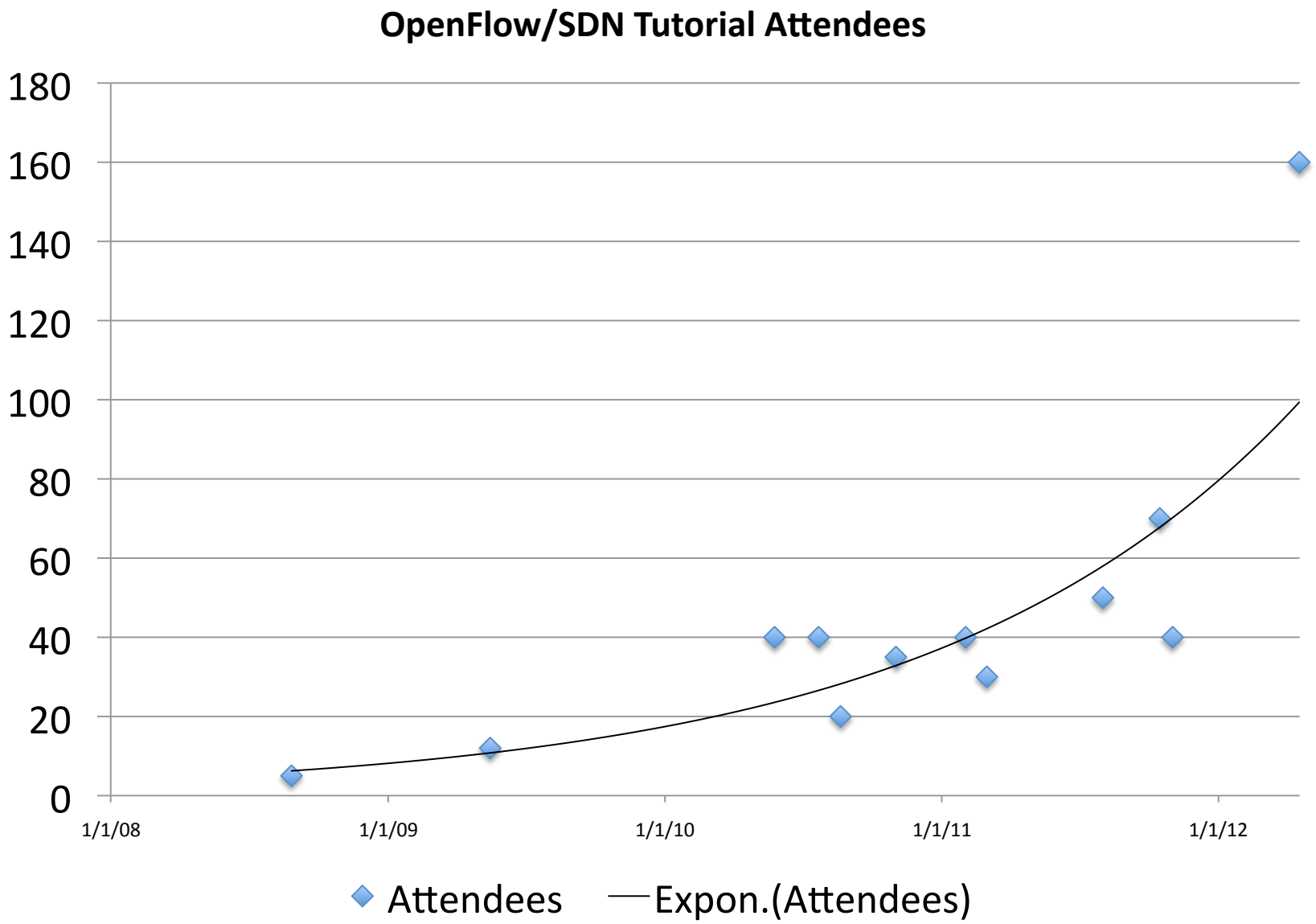
Q&A

Deployment Experiences

Deployment:

Srini Seetharaman

Wrap-Up


OpenFlow/SDN Tutorial Attendees
180
160
140
120
100
80
60
40
20
0
1/1/08
1/1/09
1/1/10
1/1/11
1/1/12
Attendees
Expon.(Attendees)

Growing Community

Note: Level of interest varies

Closing Thoughts

- SDN & OpenFlow
- Open Source Ecosystem
- SDN Switches and Controllers are available, used, and improving: open & closed

- These are the early stages for SDN

Get involved!

- Ask and answer questions on mailing lists:
  - openflow-discuss
  - openvswitch-{discuss/dev}
  - ONF forums
- Share via wiki, github, etc.
- Submit bug-reports and/or patches to OF reference implementation and Open vSwitch
- Release open-source applications
- Write a new controller!

This tutorial wouldn’t be possible without:

- Speakers
  - Rob Sherwood
  - David Erickson
  - HIDEyuki Shimonishi
  - Murphy McCauley
  - Srini Seetharaman

ACKs

(acknowledgements)

This tutorial wouldn't be possible without:

- OpenFlow Experts
  - Tatsuya Yabe
  - Nikhil Handigol
  - TY Huang
  - James Hongyi Zeng
  - Bob Lantz
  - Masahiko Takahashi
  - Ali Al-Shabibi
  - Ali Yahya
  - Glen Gibb

This tutorial wouldn't be possible without:

- Past slides from:
  - Nick McKeown
  - Rob Sherwood
  - Guru Parulkar
  - Srini Seetharaman
  - Yiannis Yiakoumis
  - Guido Appenzeller
  - Masa Kobayashi
  - Scott Shenker
  - others

This tutorial wouldn’t be possible without:

- Behind-the-scenes organizers:
  - Flora Freitas
  - Asena Gencel
  - Guru Parulkar

Thanks!

SDN Team at Stanford