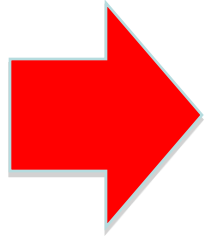




Software-defined Networking

Infocom, April 2009

Nick McKeown
nickm@stanford.edu

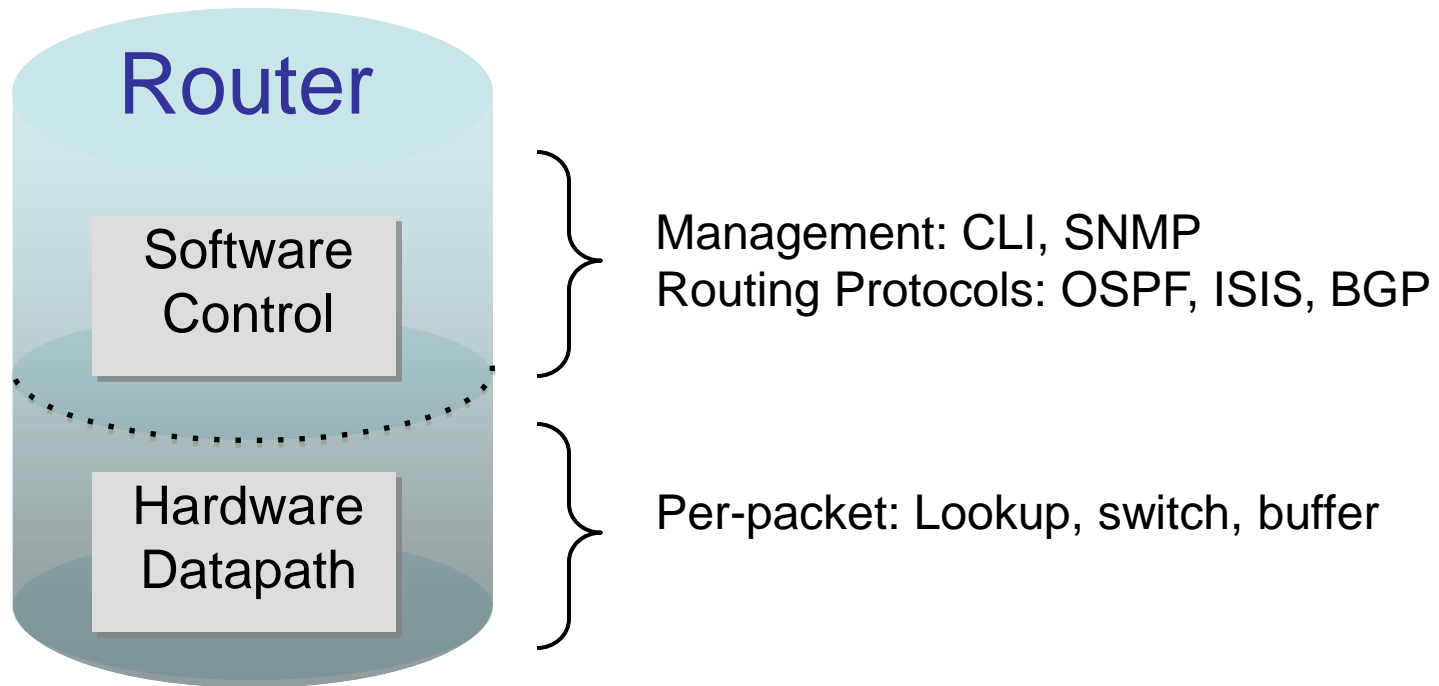


Part 1: Inside the box

Switch and Router Design

Part 2: Outside the box

Software-defined networking

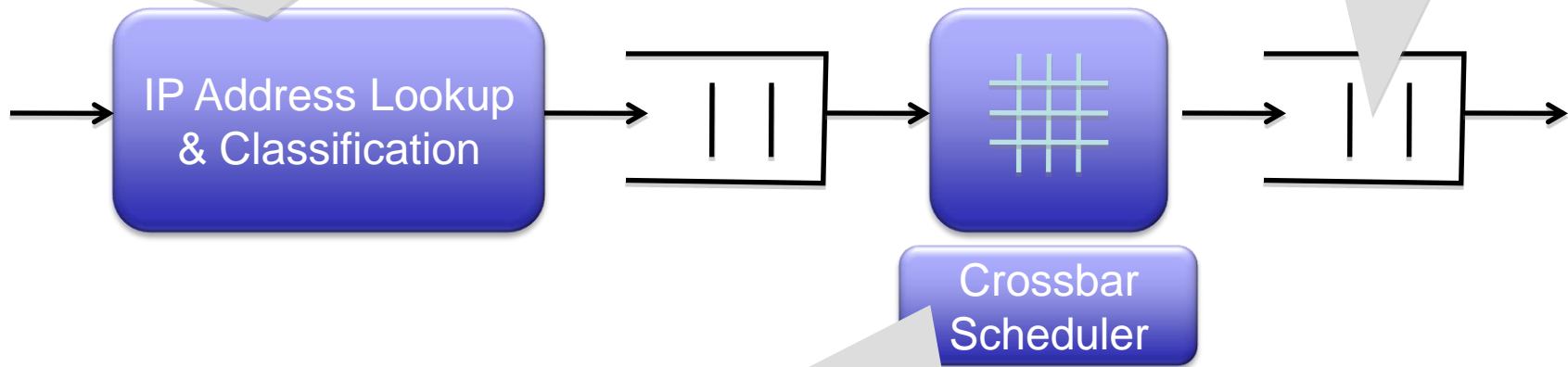


How big should buffers be? [1/□N]

How to build really fast buffers? [Nemo]

How to lookup quickly in hardware? [24-8]

Heuristic classification algorithms [HiCuts]



Which schedulers give 100% throughput? [MWM]

Which schedulers are practical in hardware? [iSLIP]

How to emulate an output queued switch? [MUCFA]

How to schedule multicast? [ESLIP]

How to run the scheduler slower? [PPS]

How to avoid scheduling altogether? [VLB]

Three Open Topics

1. There's something special about "2x speedup"
2. Deterministic (instead of probabilistic) switch design
3. Making routers simpler

Three Open Topics

1. There's something special about "2x speedup"

- ▶ A *maximal* match crossbar scheduler gives 100% throughput [Dai&Prabhakar]
- ▶ Makes a Clos network strictly non-blocking [Clos]
- ▶ Allows a CIOQ switch to precisely emulate an output-queued switch [Chuang]

Three Open Topics

1. There's something special about "2x speedup" (contd.)
 - ▶ Allows a parallel stack of small switches to precisely emulate one big switch [Iyer]
 - ▶ Valiant Load-Balanced switch (or network) can give 100% throughput [Valiant]

Related observations

- ▶ “2x speedup” is key for both deterministic & probabilistic systems
- ▶ A maximum size bipartite match is at most twice the size of a maximal match
- ▶ A switch has two simultaneous constraints: input and output
- ▶ Local “selfish” routing decisions cost twice as much as “global” ones [Roughgarden]

Three Open Topics

1. There's something special about "2x speedup"

2. **Deterministic (instead of probabilistic) switch design**

We need more analytical tools for "mimicking"

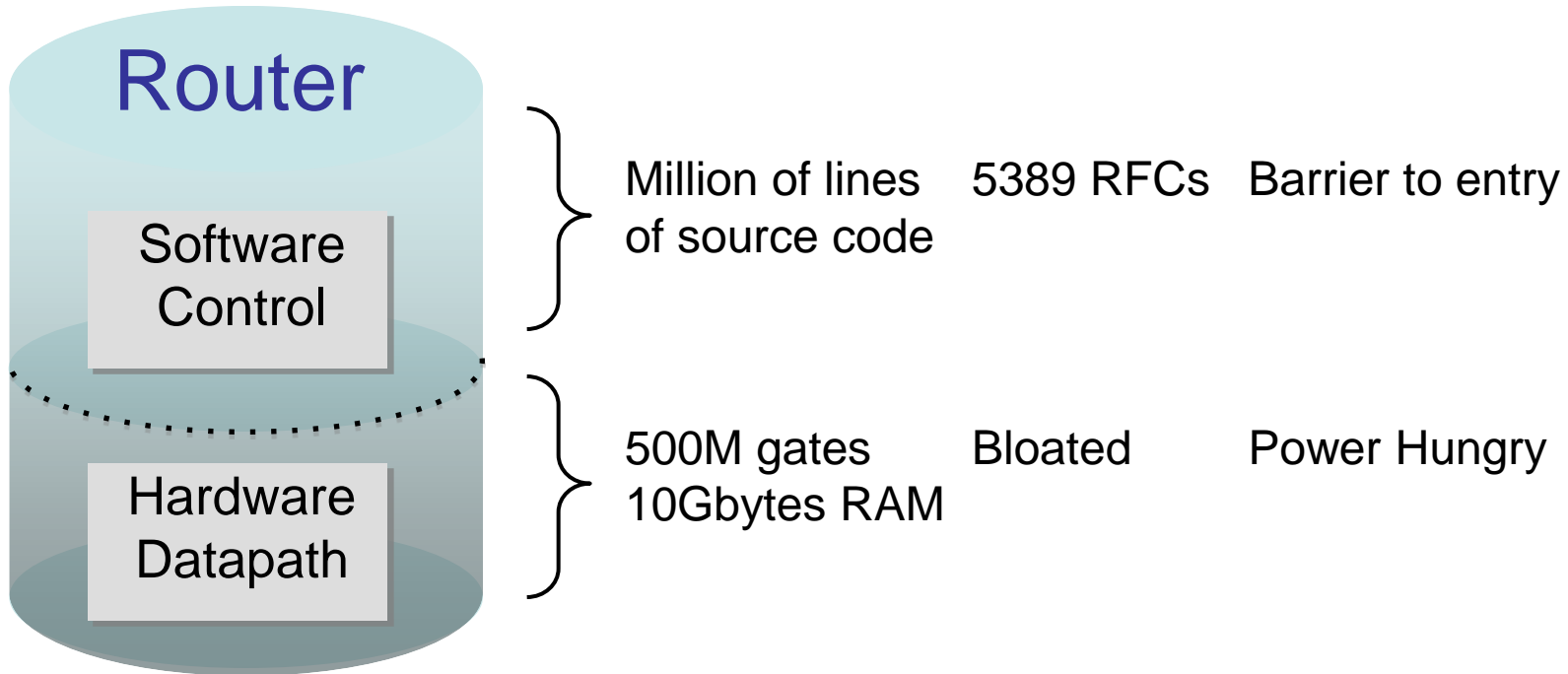
Generalized pigeon-hole principles

3. Making routers simpler

Three Open Topics

1. There's something special about "2x speedup"
2. Deterministic (instead of probabilistic) switch design
1. Making routers simpler

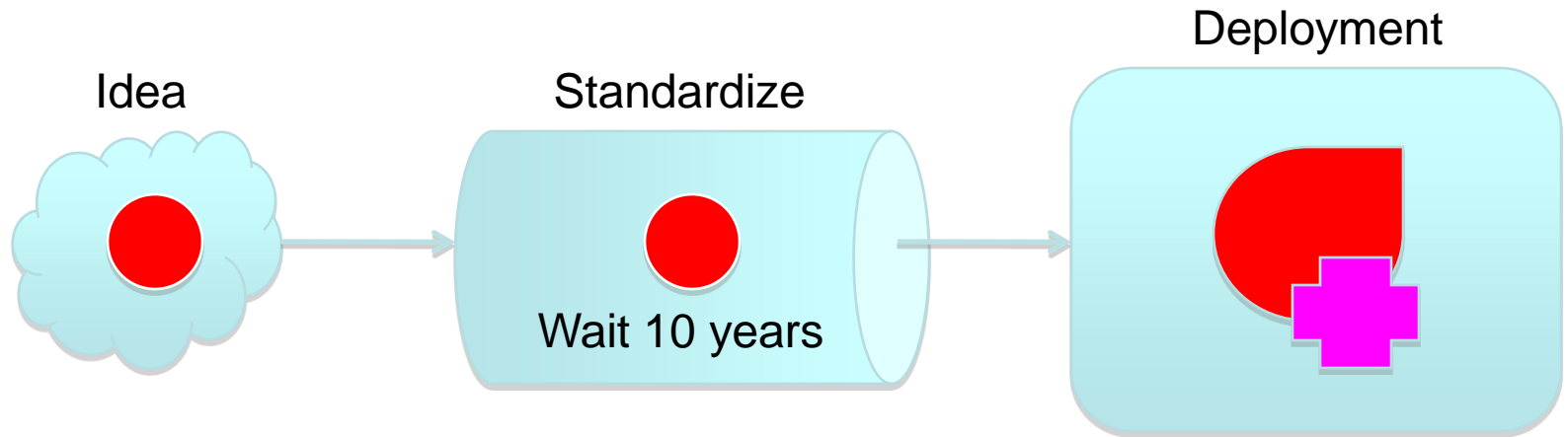
We have lost our way



Many complex functions baked into the infrastructure

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

Process of innovation



Almost no technology transfer
from academia

Personal regret

I wish I had said it **sooner** and **louder**

Our “dumb, minimal”
datapath turned into a
bloated 1960s mainframe!

The essence of my talk (1 of 2)

Hardware Substrate

- ▶ The PC industry found a simple, common, hardware substrate (x86 instruction set)

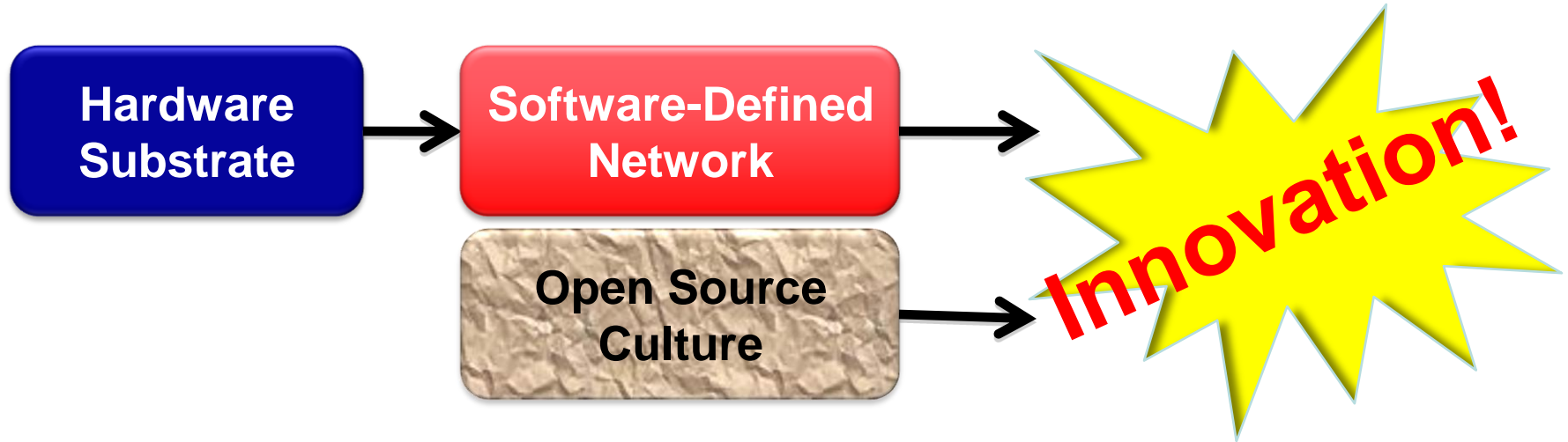
Software-definition

- ▶ Innovation exploded on top (applications) and in the infrastructure itself (operating systems, virtualization)

Open-source

- ▶ 100,000s of developers blew apart the standards process, accelerated innovation

The essence of my talk (2 of 2)



It is up to us to make it happen.

Until we (someone) does, it remains ossified.

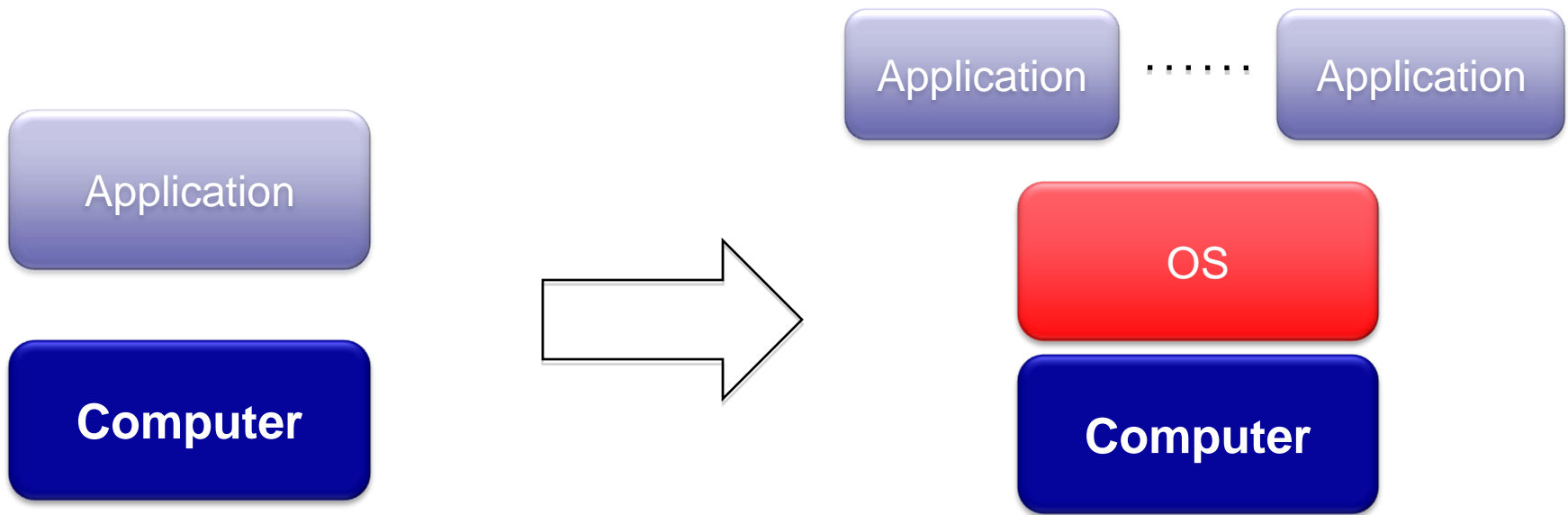
Let's define the substrate.

Part 1: Inside the box

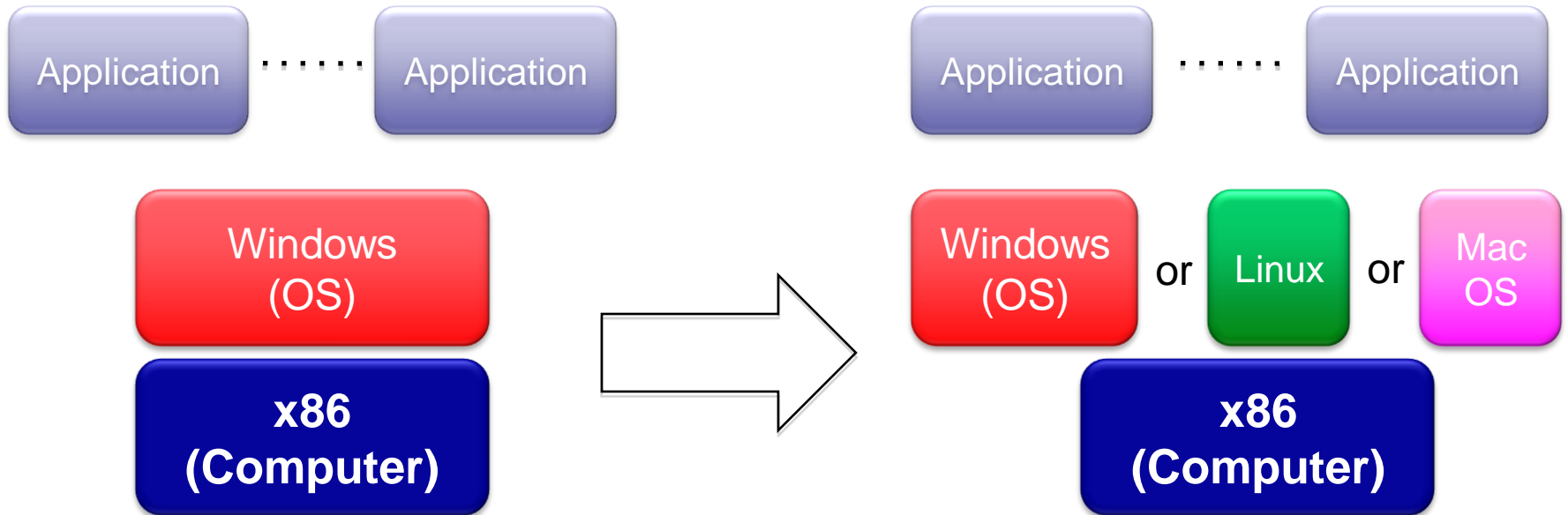


Part 2: Outside the box

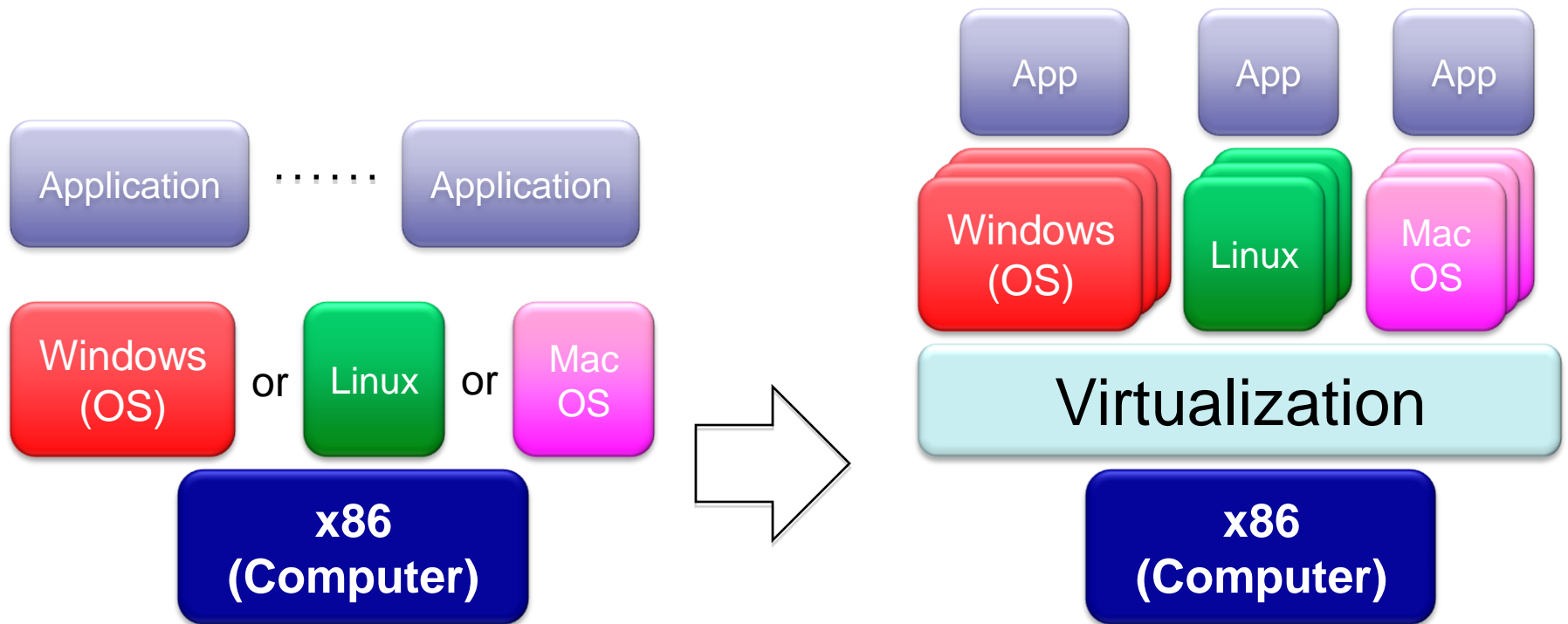
- ▶ The need for a substrate
- ▶ The inevitability of software-defined networking



OS abstracts hardware substrate
→ Innovation in applications



Simple, common, stable, hardware substrate below
+ Programmability
+ Competition
→ Innovation in OS and applications



Simple, common, stable, hardware substrate below
+ Programmability
+ Strong isolation model
+ Competition above
→ Innovation in infrastructure

A simple stable common substrate

1. **Allows applications to flourish**

Internet: Stable IPv4 lead to the web

2. **Allows the infrastructure on top to be defined in software**

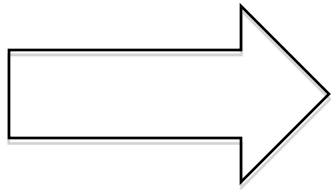
Internet: Routing protocols, management, ...

3. **Rapid innovation of the infrastructure itself**

Internet: er...? What's missing? What is the substrate...?

Mid-1990s:

“To enable innovation in the network, we need to program on top of a simple hardware datapath”

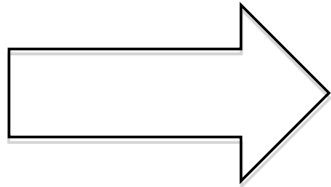


Active networking

Problems: isolation, performance, complexity

Late-1990s:

“To enable innovation in the network, we need the datapath substrate to be programmable”



Network processors

Problem: Accelerated complexity
of the datapath substrate

(Statement of the obvious)

In networking, despite several attempts...

We've never agreed upon a clean separation between:

1. A simple common hardware substrate
2. And an open programming environment on top

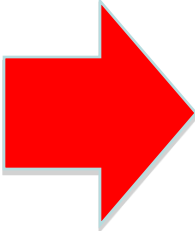
But things are changing fast in data centers and service provider networks.

Observations

Prior attempts have generally

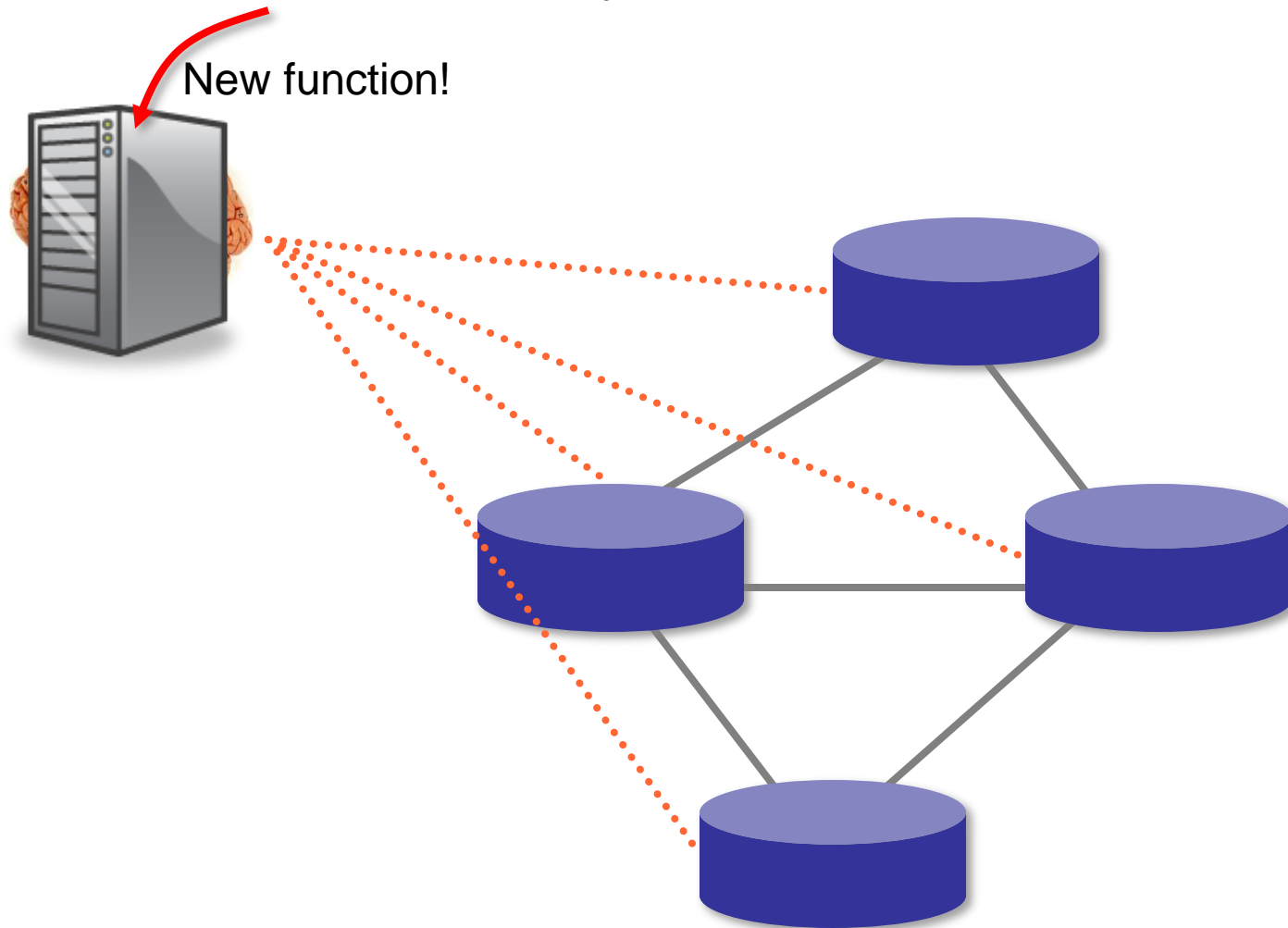
1. Assumed the current IP routing substrate is fixed, and tried to program it externally
 - ▶ Including the routing protocols
2. Defined the programming and control model up-front
 - ▶ But to pick the right x86 instruction set, Intel didn't define Windows XP, Linux or VMware

We need...

- 
1. A clean separation between the substrate and an open programming environment
 2. A simple hardware substrate that generalizes, subsumes and simplifies the current substrate
 3. Very few preconceived ideas about how the substrate will be programmed
 4. Strong isolation

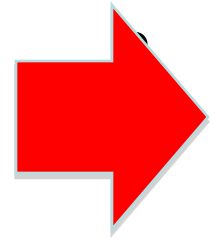
Step 1: Separate intelligence from datapath

Operators, users, 3rd party developers, researchers, ...



We need...

1. A clean separation between the substrate and an open programming environment
2. A simple hardware substrate that generalizes, subsumes and simplifies the current substrate
3. Very few preconceived ideas about how the substrate will be programmed
4. Strong isolation

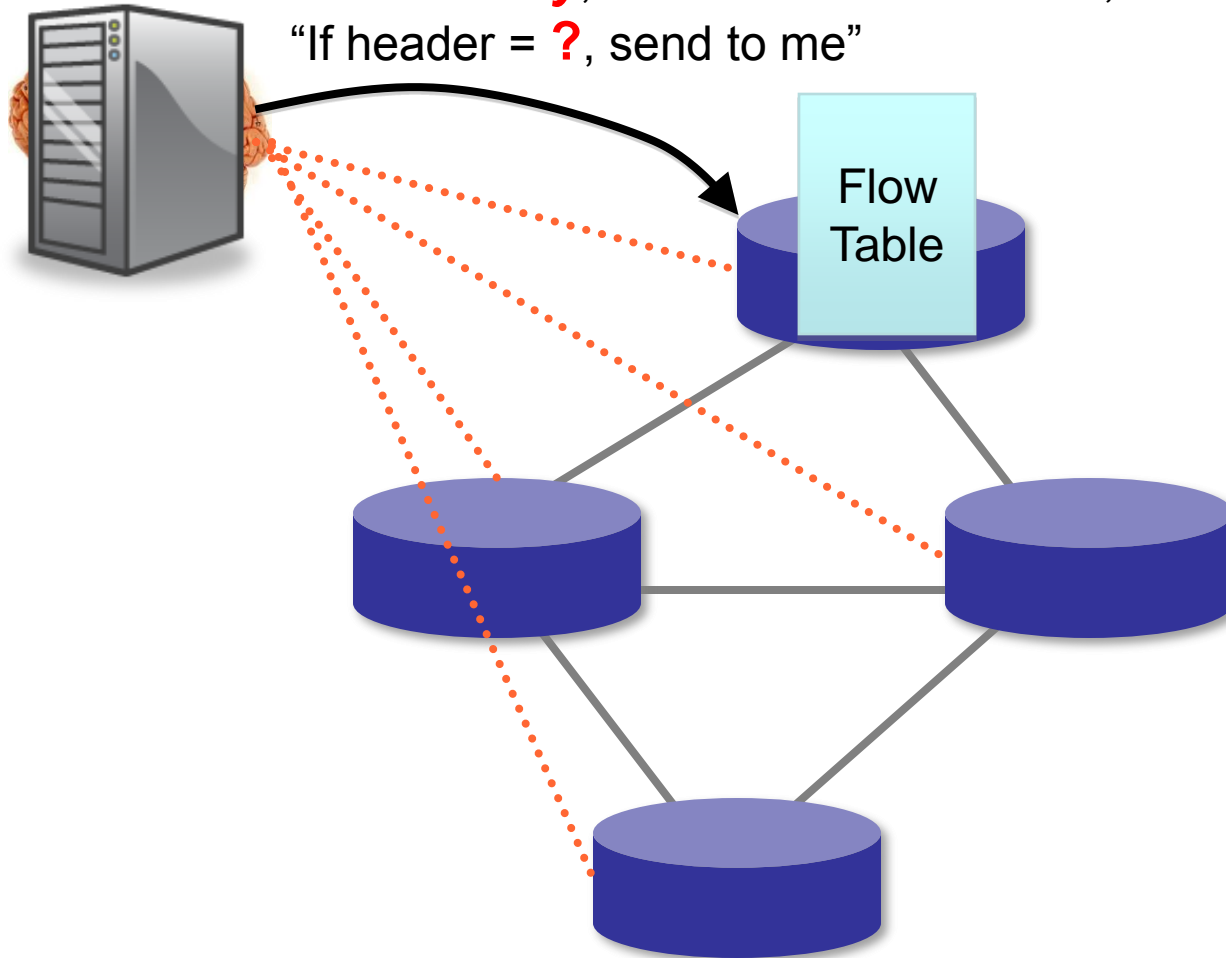


Step 2: Cache decisions in minimal flow-based datapath

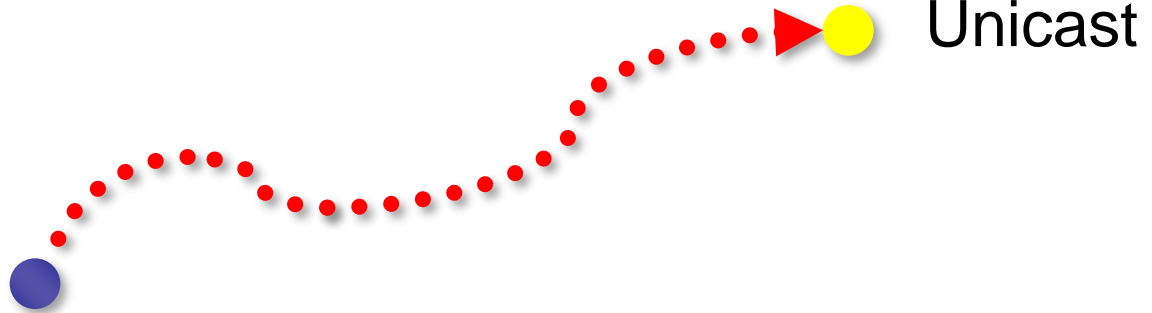
“If header = **x**, send to port 4”

“If header = **y**, overwrite header with **z**, send to ports 5,6”

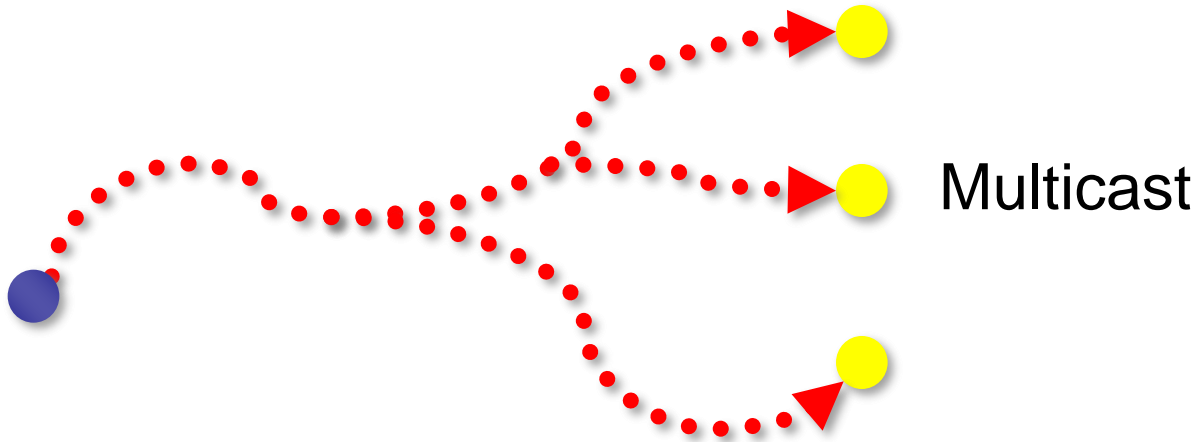
“If header = **?**, send to me”



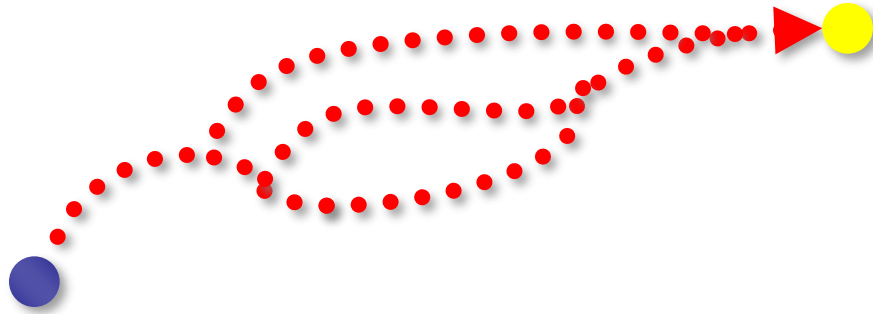
1.



2.



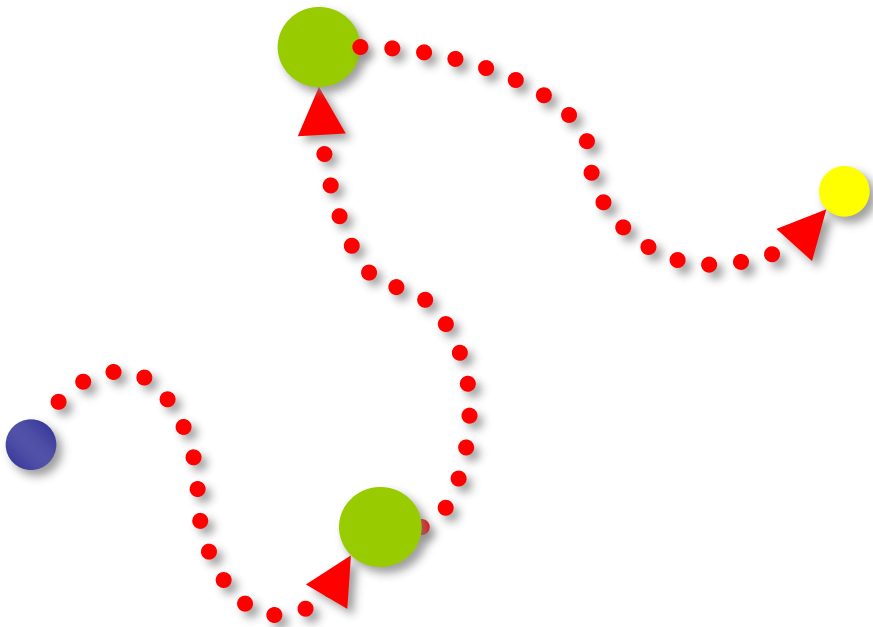
3.



Multipath

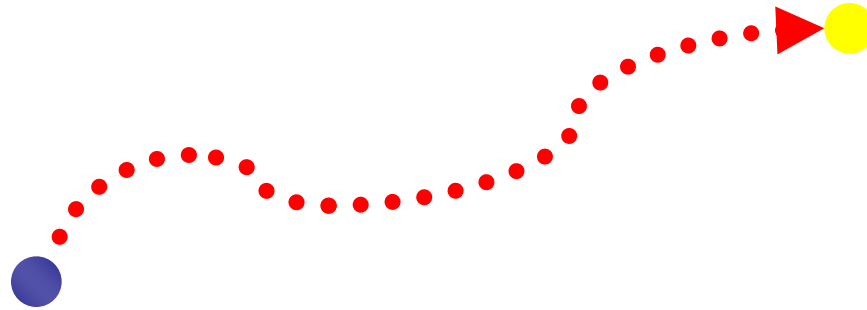
- Load-balancing
- Redundancy

4.



Waypoints

- Middleware
- Intrusion detection
- ...



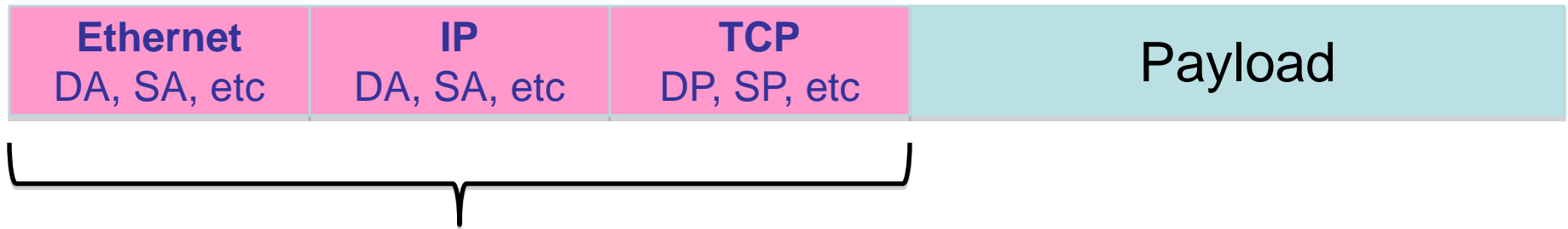
What is a flow?

- Application flow
- All http
- Jim's traffic
- All packets to Canada
- ...

Types of action

- Allow/deny flow
- Route & re-route flow
- Isolate flow
- Make flow private
- Remove flow

Packet-switching substrate



Collection of bits to plumb flows
(of different granularities)
between end points

Properties of a flow-based substrate

We need flexible definitions of a flow

- ▶ Unicast, multicast, waypoints, load-balancing
- ▶ Different aggregations

We need direct control over flows

- ▶ Flow as an entity we program: To route, to make private, to move, ...

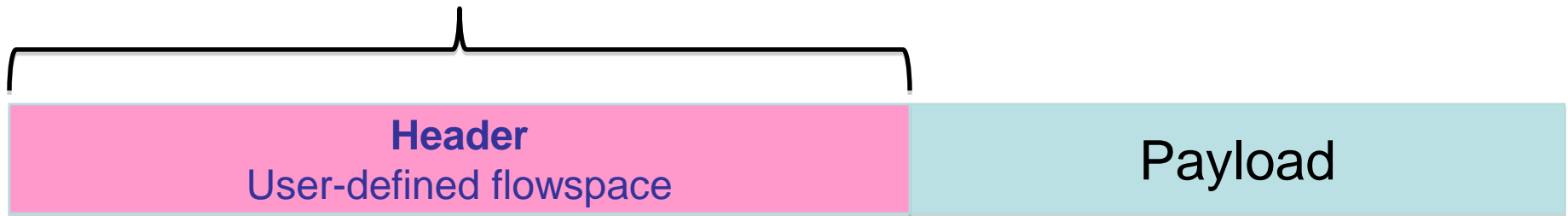
Exploit the benefits of packet switching

- ▶ It works and is universally deployed
- ▶ It's efficient (when kept simple)

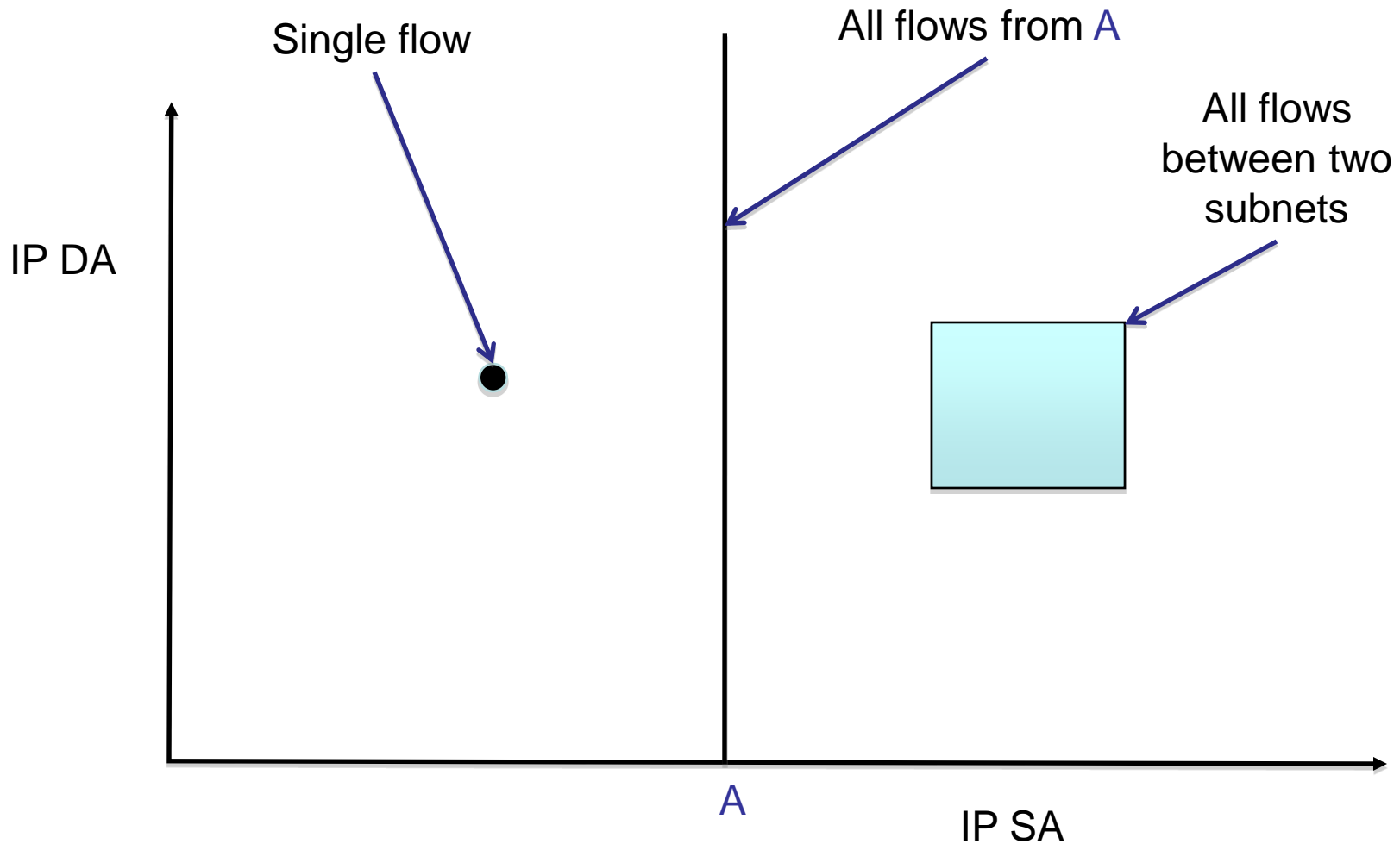
Substrate: “FlowSpace”



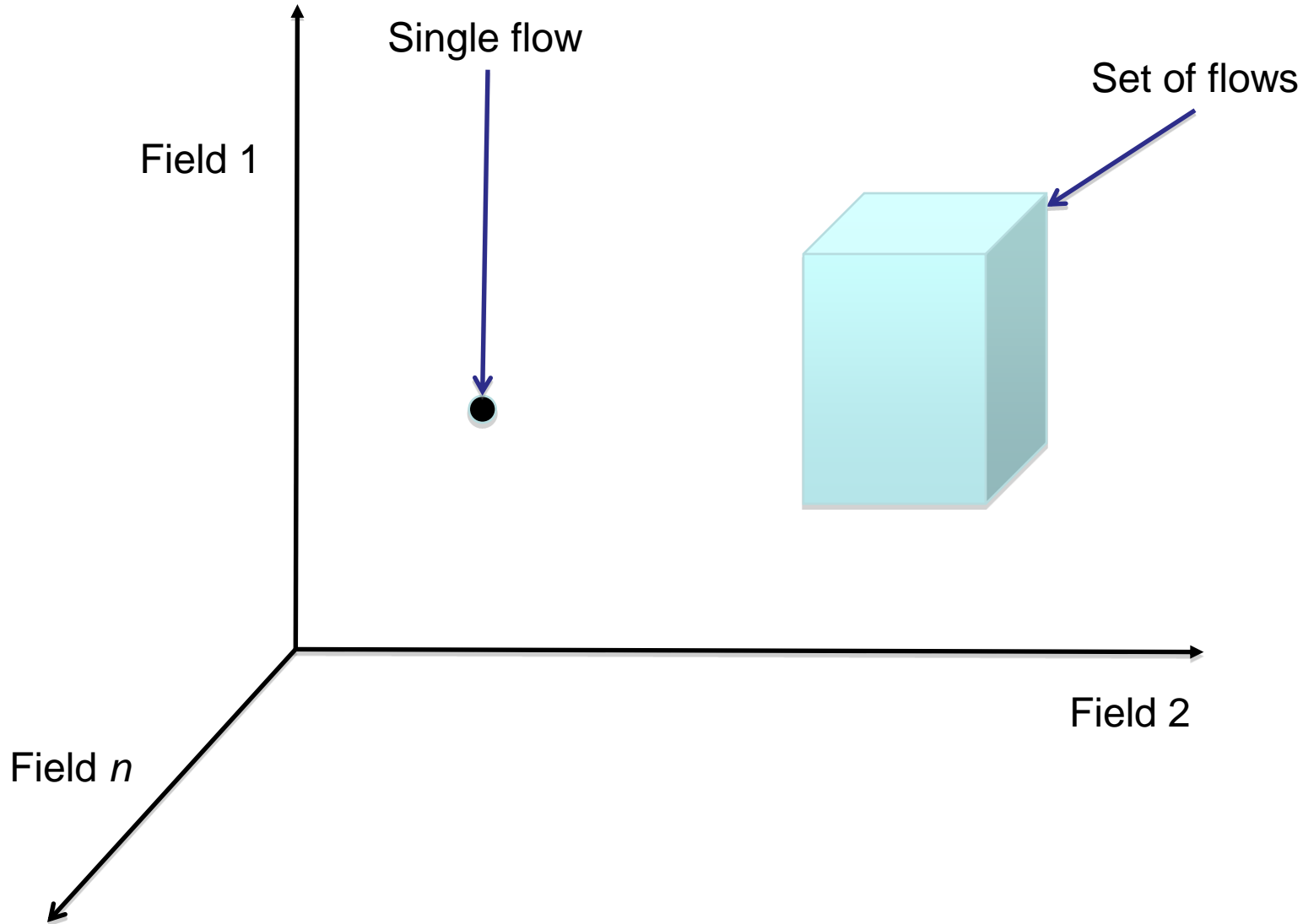
Collection of bits to plumb flows
(of different granularities)
between end points



Flowspace: Simple example



Flowspace: Generalization



Properties of FlowSpace

Backwards compatible

- ▶ Current layers are a special case
- ▶ No end points need to change

Easily implemented in hardware

- ▶ e.g. TCAM flow-table in each switch

Strong isolation of flows

- ▶ Simple geometric construction
- ▶ Can prove which flows can/cannot communicate

A substrate

Flow-based

Small number of actions for each flow

- ▶ Plumbing: Forward to port(s)
- ▶ Control: Forward to controller
- ▶ Routing between flow-spaces: Rewrite header
- ▶ Bandwidth isolation: Min/max rate

External open API to flow-table

OpenFlow as a strawman
flow-based substrate

Our Approach

1. Define the substrate

OpenFlow is an open external API to a flow-table

Version 1.0

- ▶ Defined to be easy to add to existing hardware switches, routers, APs, ...
- ▶ Timeframe: Now

Version 2.0

- ▶ OpenFlow-optimized hardware
- ▶ General “flowspace”
- ▶ Timeframe: 2011

Our Approach

2. Deploy

Deploy on college campuses

Deploy in national research backbone networks

Enable researchers to freely innovate on top

OpenFlow Hardware



Juniper MX-series



NEC IP8800



WiMax (NEC)



HP Procurve
5400



Cisco Catalyst
6k



PC Engines



Quanta LB4G

More coming soon...

An OpenFlow Controller

Controller



- ▶ “Nicira” created NOX controller
- ▶ Available at <http://NOXrepo.org>



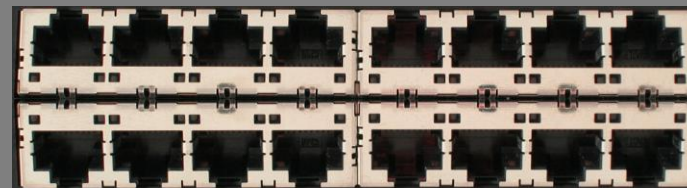
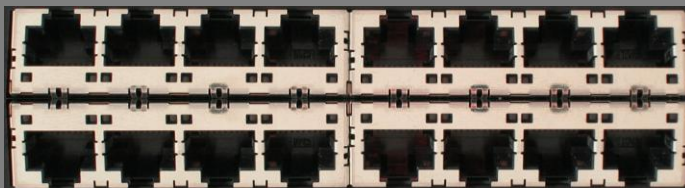
Martin
Casado



Scott
Shenker

OpenFlow Basics

Ethernet Switch



Control Path (Software)

Data Path (Hardware)

OpenFlow Controller

OpenFlow Protocol (SSL)



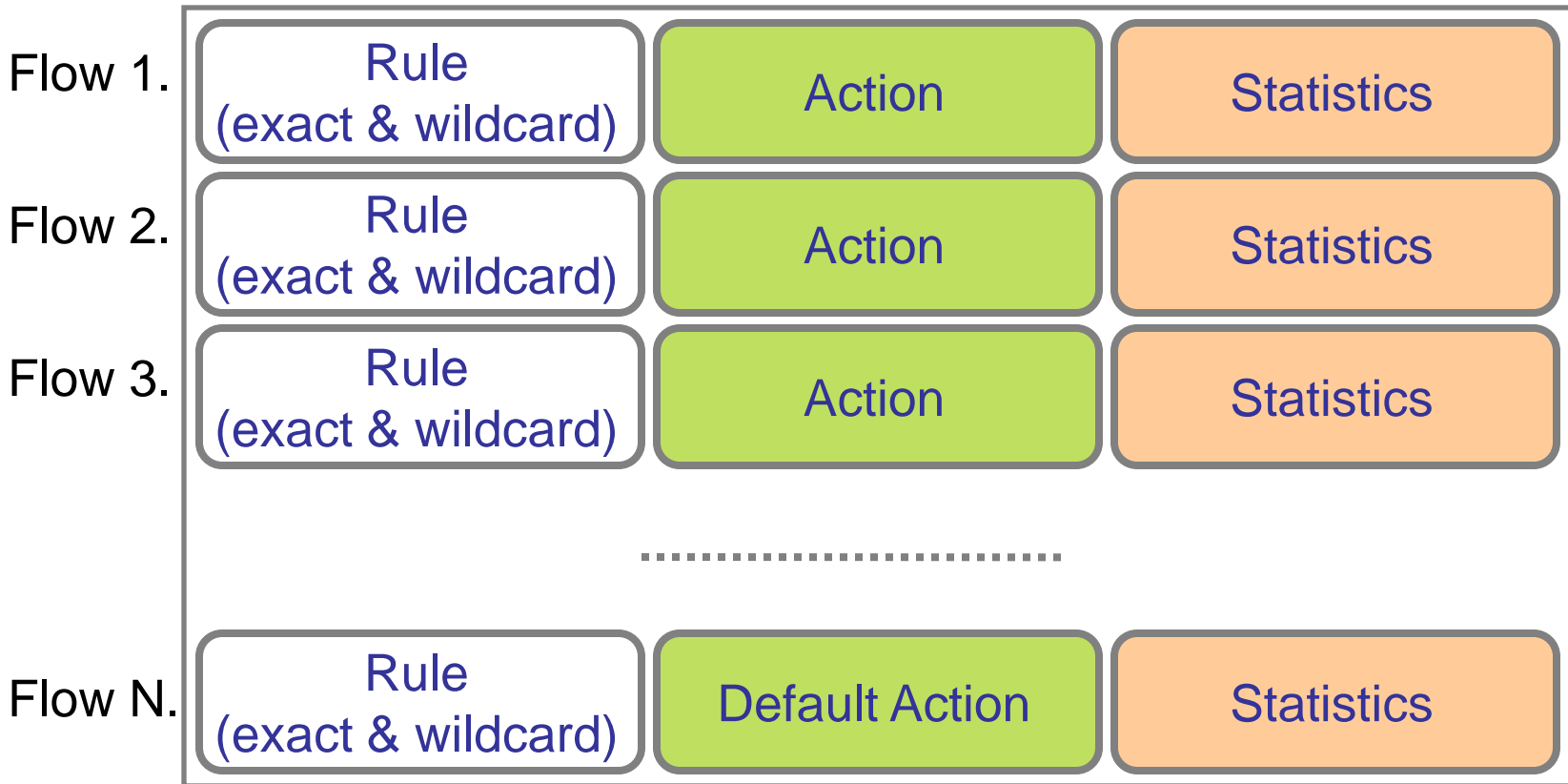
Control Path

OpenFlow

Data Path (Hardware)

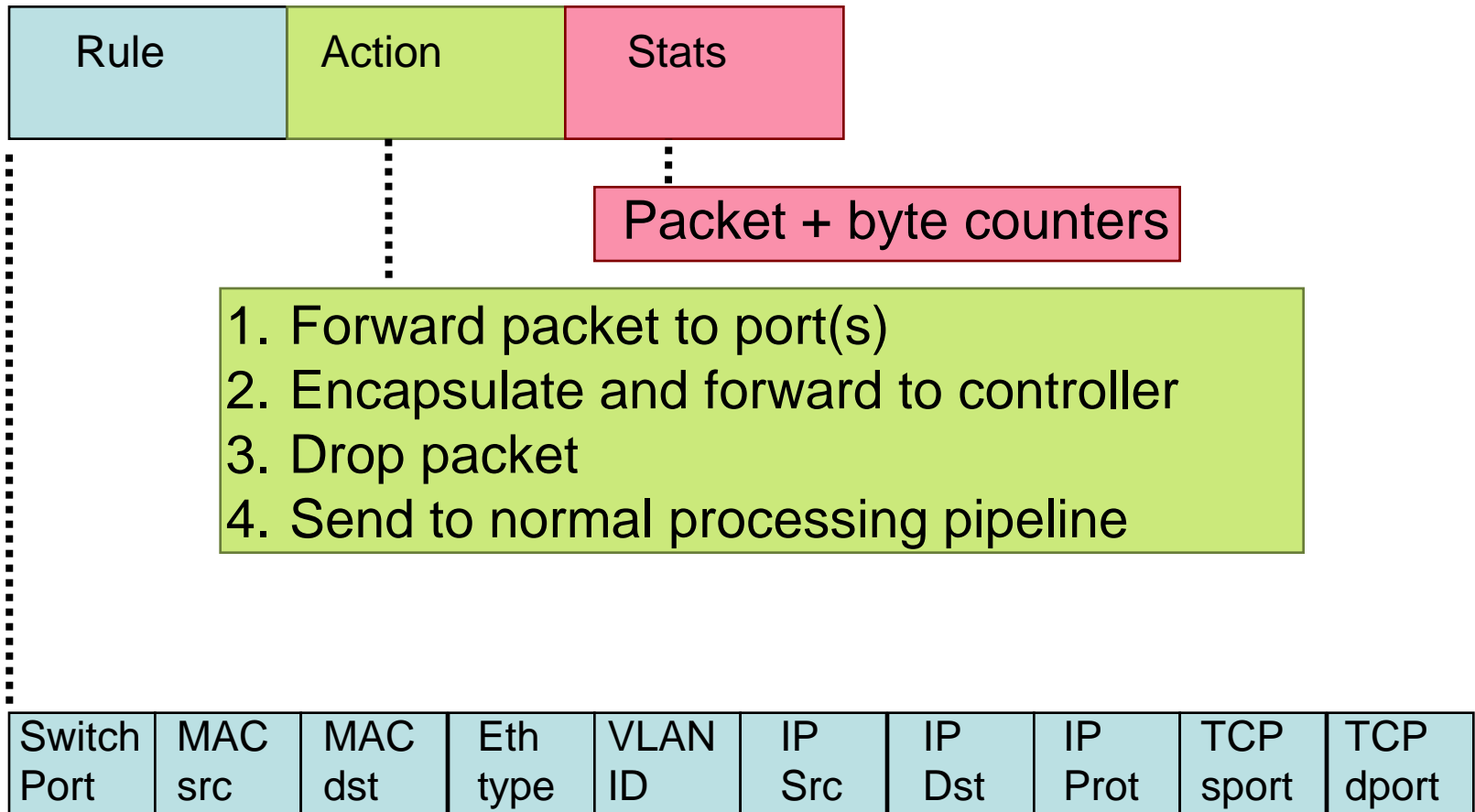
OpenFlow Basics (1)

Exploit the flow table in switches, routers, and chipsets



Flow Table Entry

OpenFlow Protocol Version 1.0



+ mask what fields to match

Examples

Routing

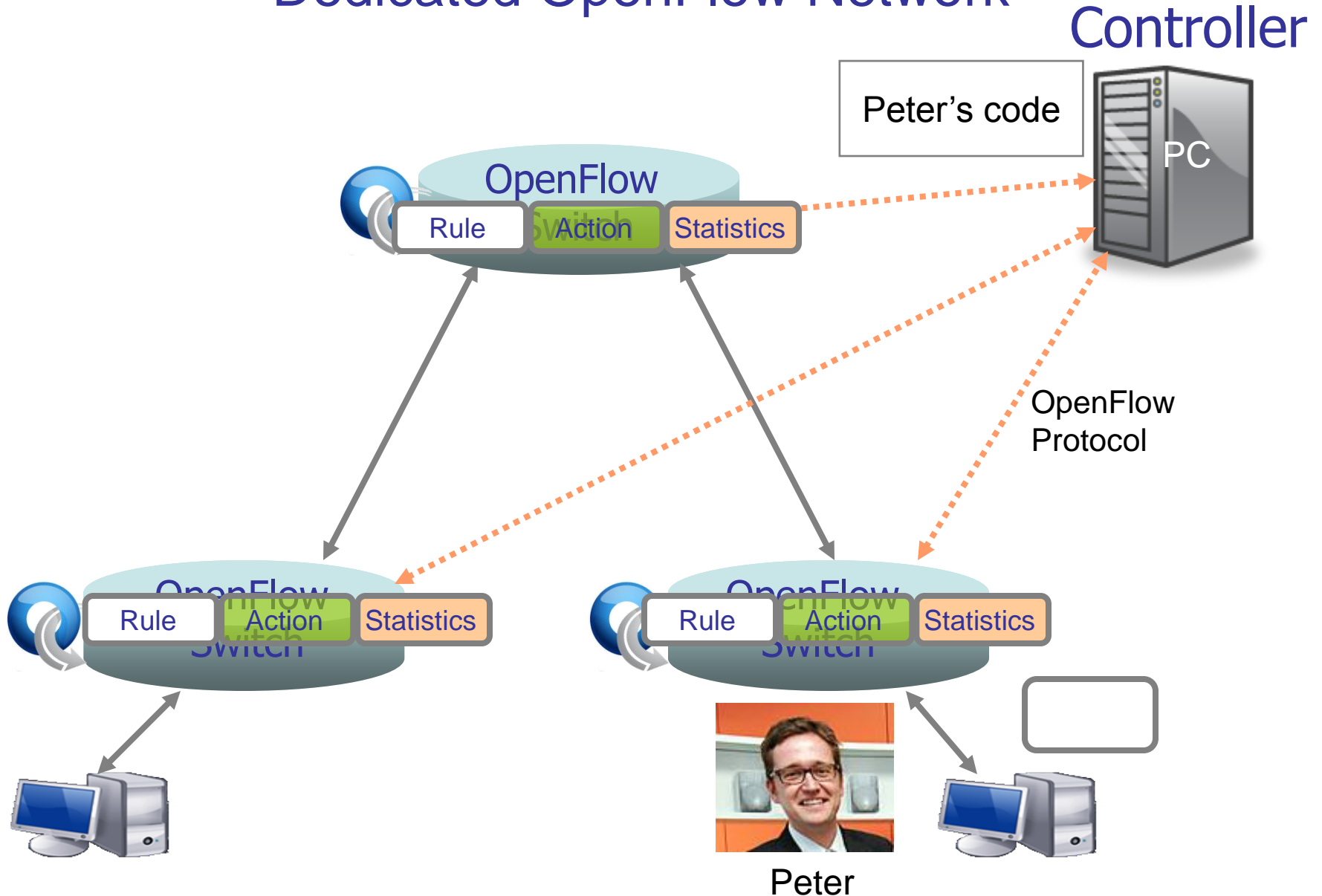
Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	5.6.7. 8	*	*	*	port6

VLAN

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

OpenFlow Usage

Dedicated OpenFlow Network

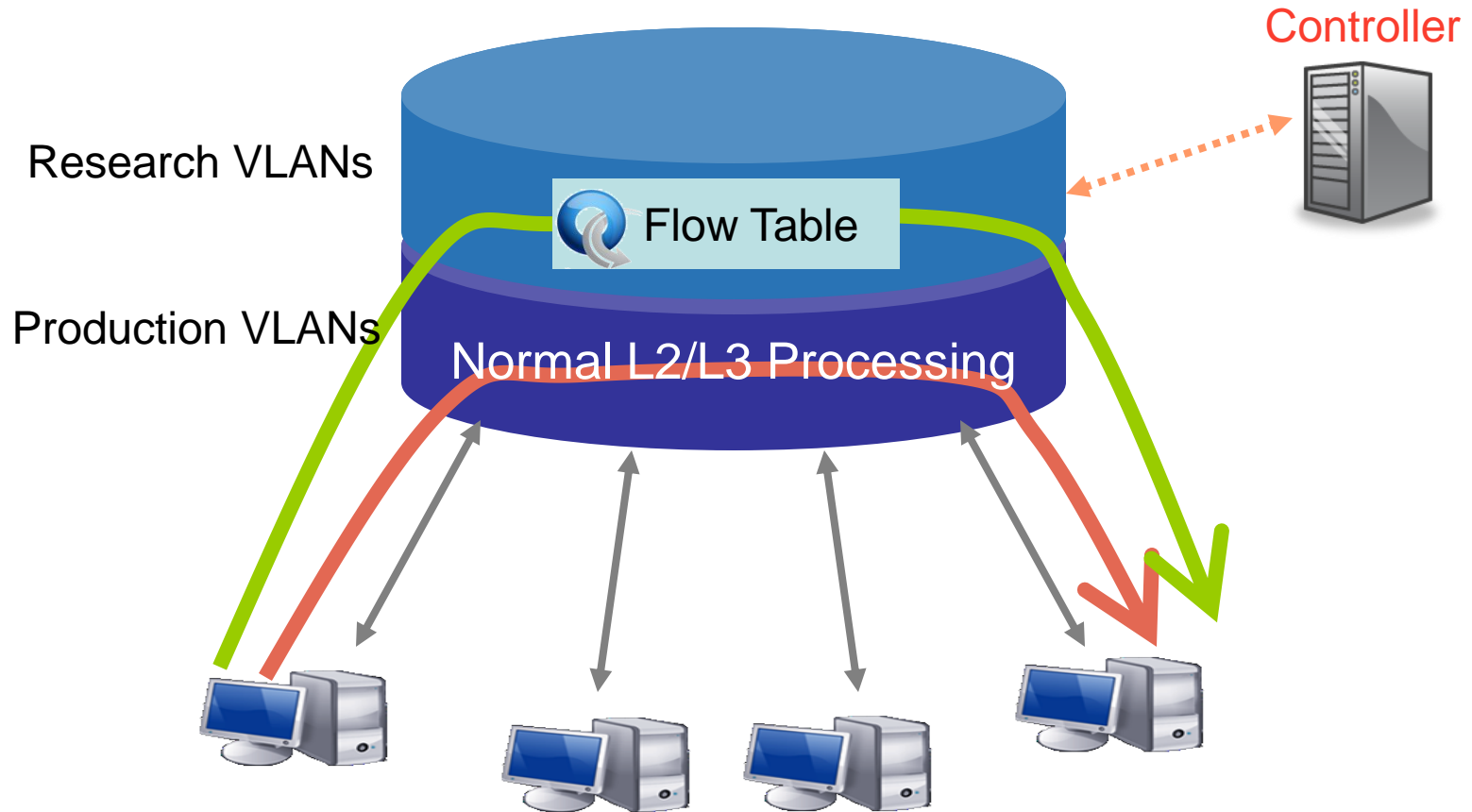


Usage examples

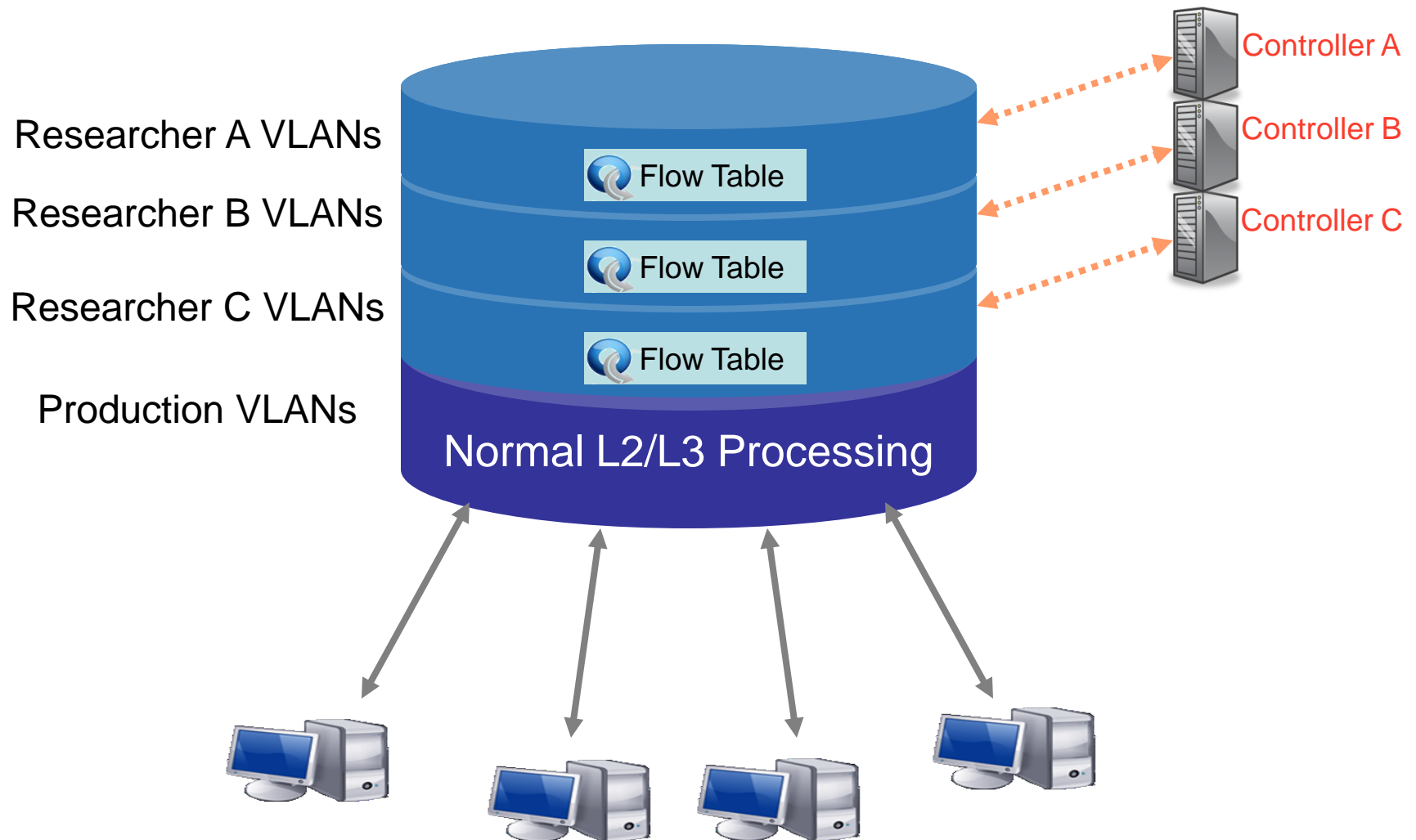
Peter's code:

- ▶ Static "VLANs"
- ▶ His own new routing protocol: unicast, multicast, multipath, load-balancing
- ▶ Network access control
- ▶ Home network manager
- ▶ Mobility manager
- ▶ Energy manager
- ▶ Packet processor (in controller)
- ▶ IPvPeter
- ▶ Network measurement and visualization
- ▶ ...

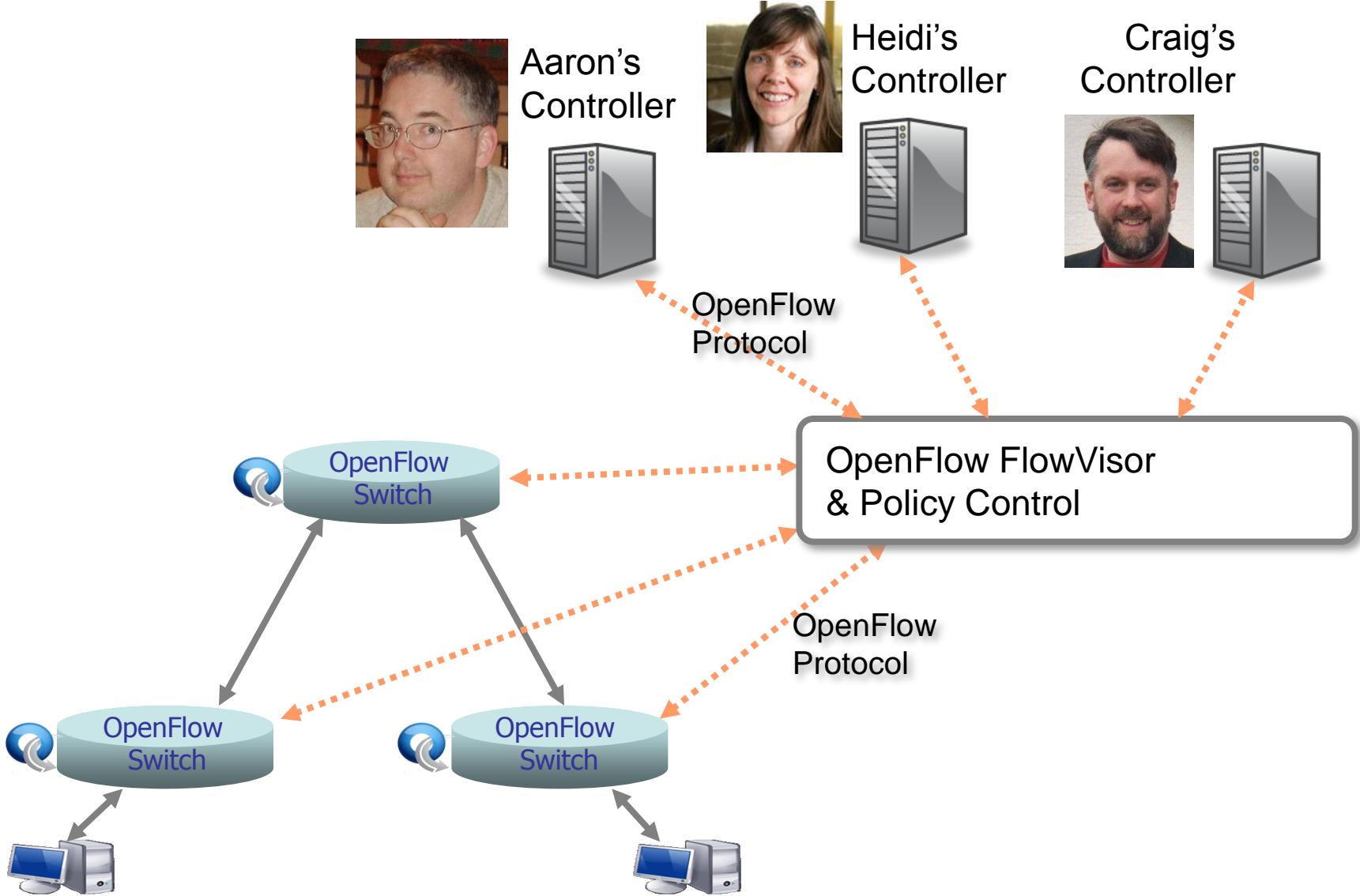
Separate VLANs for Production and Research Traffic



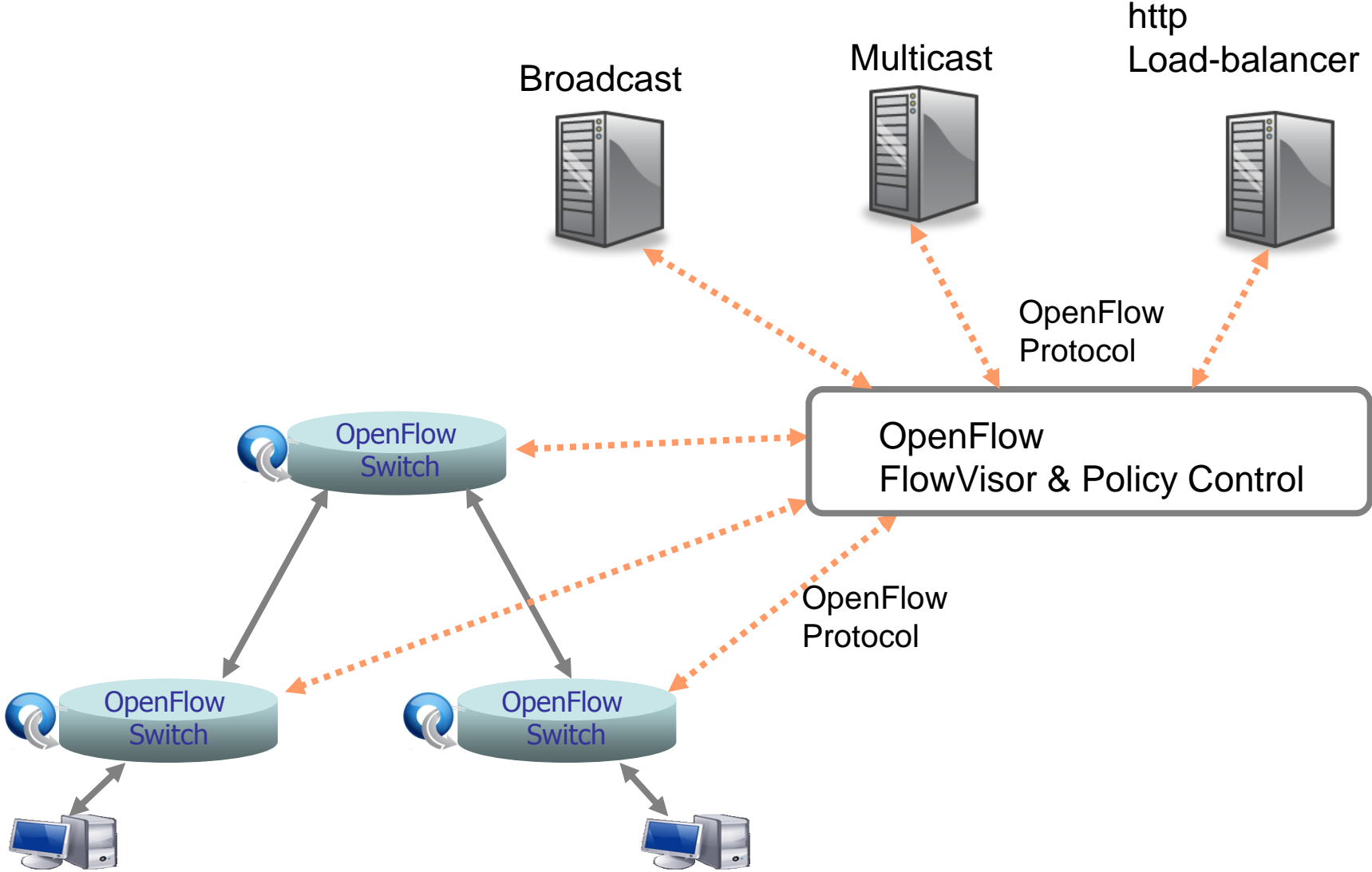
Virtualize OpenFlow Switch

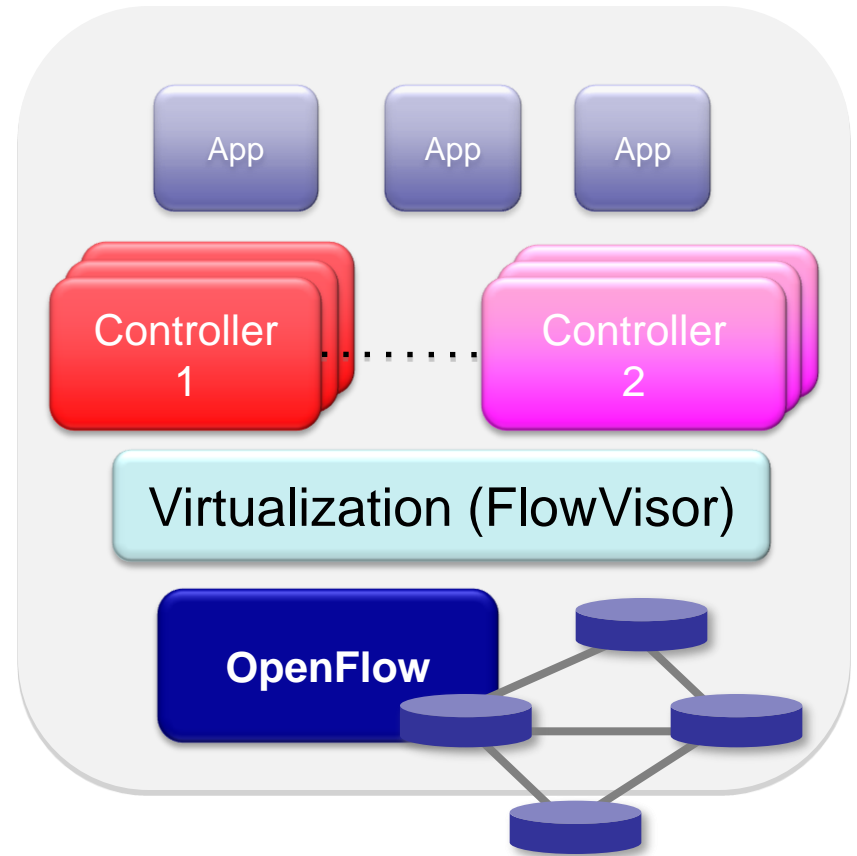
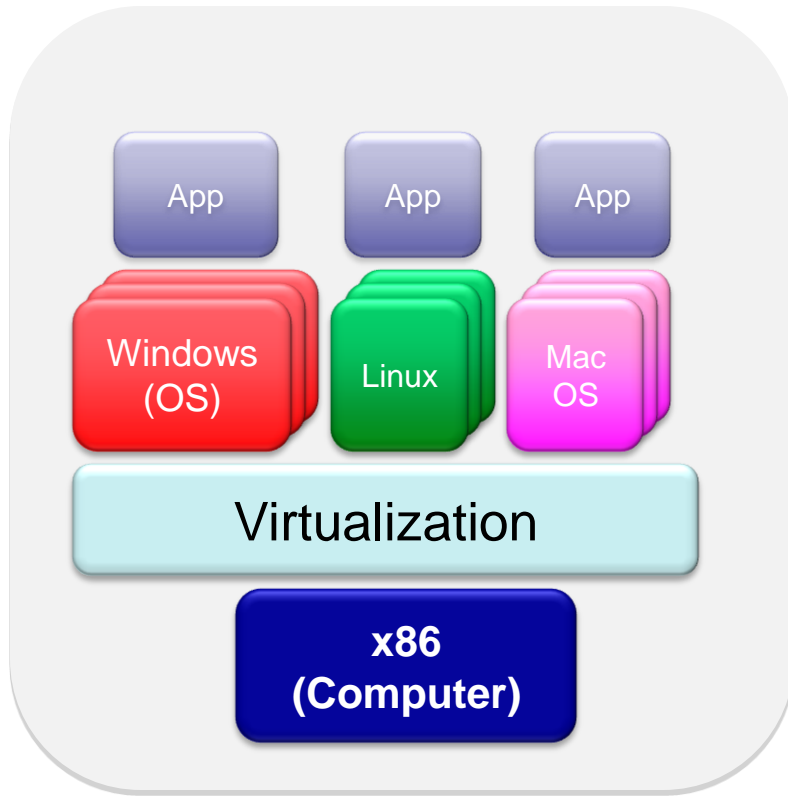


Virtualizing OpenFlow



Virtualizing OpenFlow





Simple, common, stable, hardware substrate below
+ Programmability
+ Strong isolation model
+ Competition above
→ Faster innovation

OpenFlow Deployment

OpenFlow Deployments

Stanford Deployments

- ▶ **Wired:** CS Gates building, EE CIS building, EE Packard building
- ▶ **WiFi:** 100 OpenFlow APs across SoE
- ▶ **WiMAX:** OpenFlow service in SoE

Other deployments

- ▶ Internet2 (NetFPGA switches)
- ▶ JGN2plus, Japan (NEC switches)
- ▶ 10-15 research groups have switches

OpenFlow Deployments

Plans in 2009-10

Campus deployments

- ▶ Lab + production use
- ▶ “Enterprise GENI” (NSF/GPO)

Backbone deployments

- ▶ National research backbones
- ▶ Research + Production use

How to get involved (1)

Visit <http://OpenFlowSwitch.org>

Experiment with reference switches

- ▶ Linux soft switch
- ▶ NetFPGA hardware switch

Explore with your network administrator/CIO
about trial production deployment

- ▶ Look at prototype commercial hardware

How to get involved (2)

Experiment with controllers

- ▶ Simple test controllers
- ▶ NOX: <http://NOXrepo.org>

Add a new experiment/feature

Run a class

Thank You!

