Software-defined Networking Session I

Advaned Computer Networks

David Koll

Partly based on slides of Nick McKeown, Scott Shenker, Nick Feamster, and Jennifer Rexford

Why this course?

„Software-Defined Networks **– the counter model of the internet“**
– *heise.de*

“**November 2014: Cisco** declares “game over” for SDN competitors […], prompting reaction from two industry groups that the game has just begun; **Alcatel-Lucent** and **Juniper** also virtualize their routers […]; **AT&T** and others unveil […] an alternative […].”
– *networkworld.com*

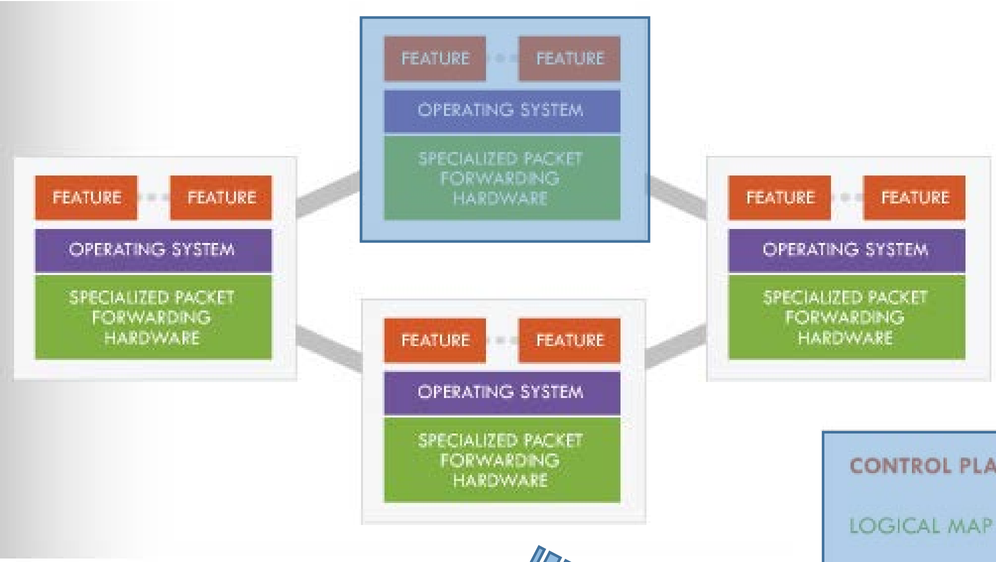
“Many solution providers believe 2015 is the year that **SDN will truly begin to reshape the networking landscape**”
– *crn.com*

What is Software-defined Networking?

“The physical separation of the network control plane from the forwarding plane, and where a control plane controls several devices.”
– The Open Networking Foundation*

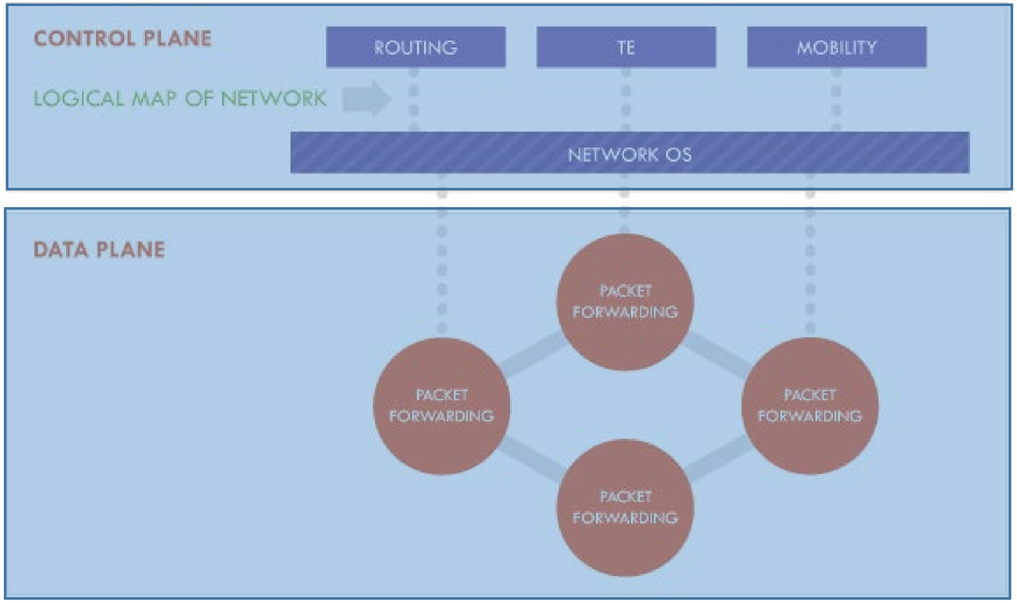
*** Google, Facebook, Microsoft, Deutsche Telekom, Verizon, Yahoo, Cisco, Citrix, Dell, Ericsson, HP, IBM, Juniper Networks, NEC, Netgear, VMWare, …
…and various institutions from academia (e.g., Stanford, Berkeley)**

SDN in a Nutshell


FEATURE
FEATURE
OPERATING SYSTEM
SPECIALIZED PACKET FORWARDING HARDWARE
FEATURE
FEATURE
OPERATING SYSTEM
SPECIALIZED PACKET FORWARDING HARDWARE
FEATURE
FEATURE
OPERATING SYSTEM
SPECIALIZED PACKET FORWARDING HARDWARE
FEATURE
FEATURE
OPERATING SYSTEM
SPECIALIZED PACKET FORWARDING HARDWARE

"The physical separation of the network control plane from the forwarding plane, and where a control plane controls several devices."

– The ONF



Taken from: http://www.opennetsummit.org/archives/apr12/site/why.html

The History behind the Hype

Going to talk about…

What are the origins of SDN?

Why do we need SDN?

Where are we now?

… before we dive into the technical details of SDN

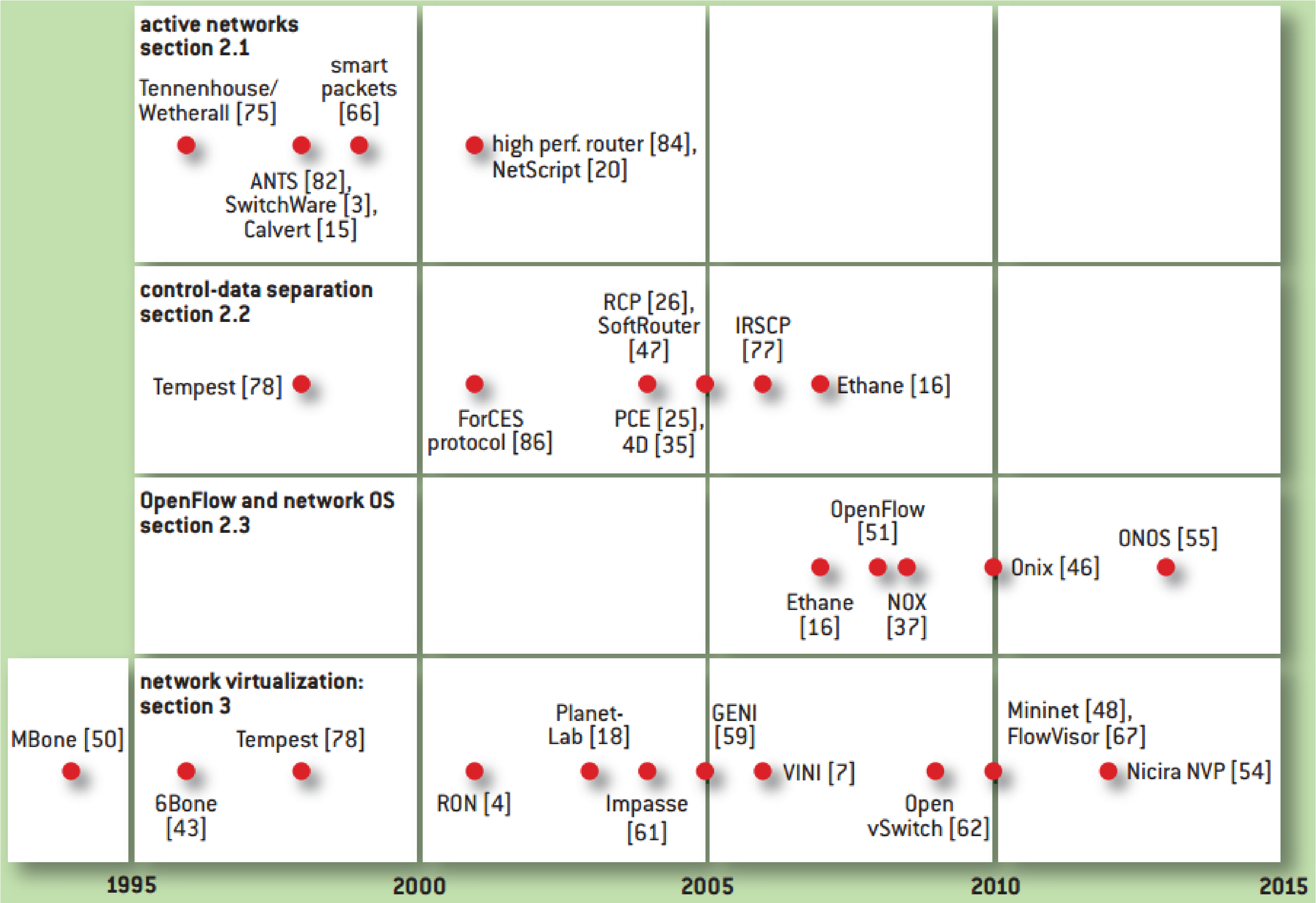
The History behind the Hype

The concepts behind SDN are not really new!

Scott Shenker: *„[SDN is] not a revolutionary technology, [it is] just a way of organizing network functionality.“[1]*

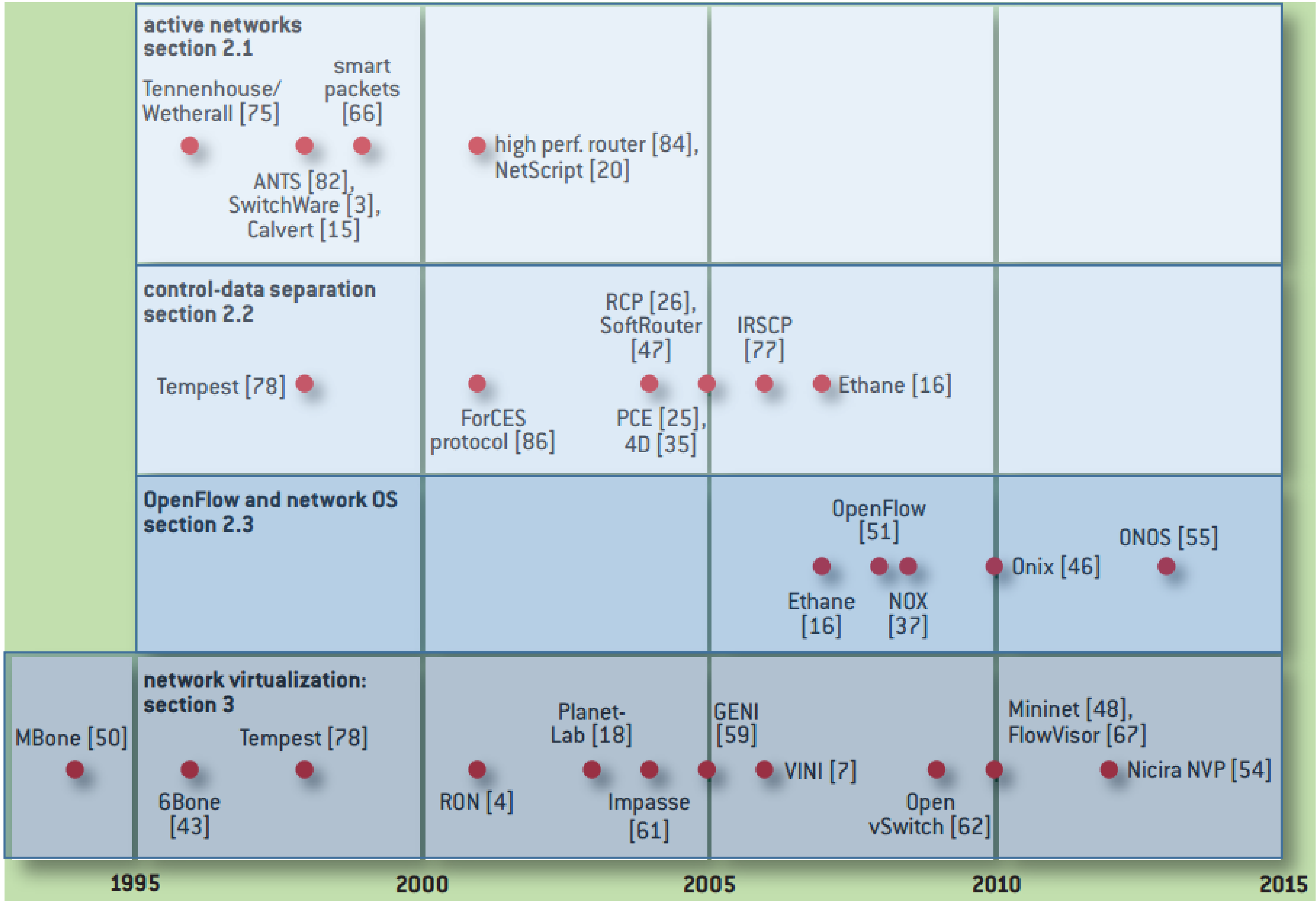
[1] S. Shenker in his talk „A Gentle Introduction to Software-defined Networks“

The History behind the Hype



N. Feamster et al.: „The Road to SDN – An intellectural history of programmable networks" *ACM SIGCOMM Computer Communication Review* 44.2 (2014): 87-98.

The History behind the Hype



N. Feamster et al.: „The Road to SDN – An intellectual history of programmable networks" *ACM SIGCOMM Computer Communication Review* 44.2 (2014): 87-98.

A brief history of programmable networks:

Active Networks

Active Networks?

- End of 1990s: network ossification (idea->deployment: 10 years!)

- Goal: opening up network control
- Envisioned method: make network devices programmable via an API

- API could be accessed via two models:
  - **Capsule model:** code included in data packets transmitted in-band [1]
  - **Programmable router/switch model:** code transmitted out-of-band [2]

[1] **Wetherall, et al**.: *"ANTS: a toolkit for building and dynamically deploying network protocols."* In Proceedings of IEEE OpenArch 1998.
[2] **Bhattacharjee, S., Calvert, K.L. et al.:** *"An architecture for active networks".* In Proceedings of High-Performance Networking 1997.

Active Networks

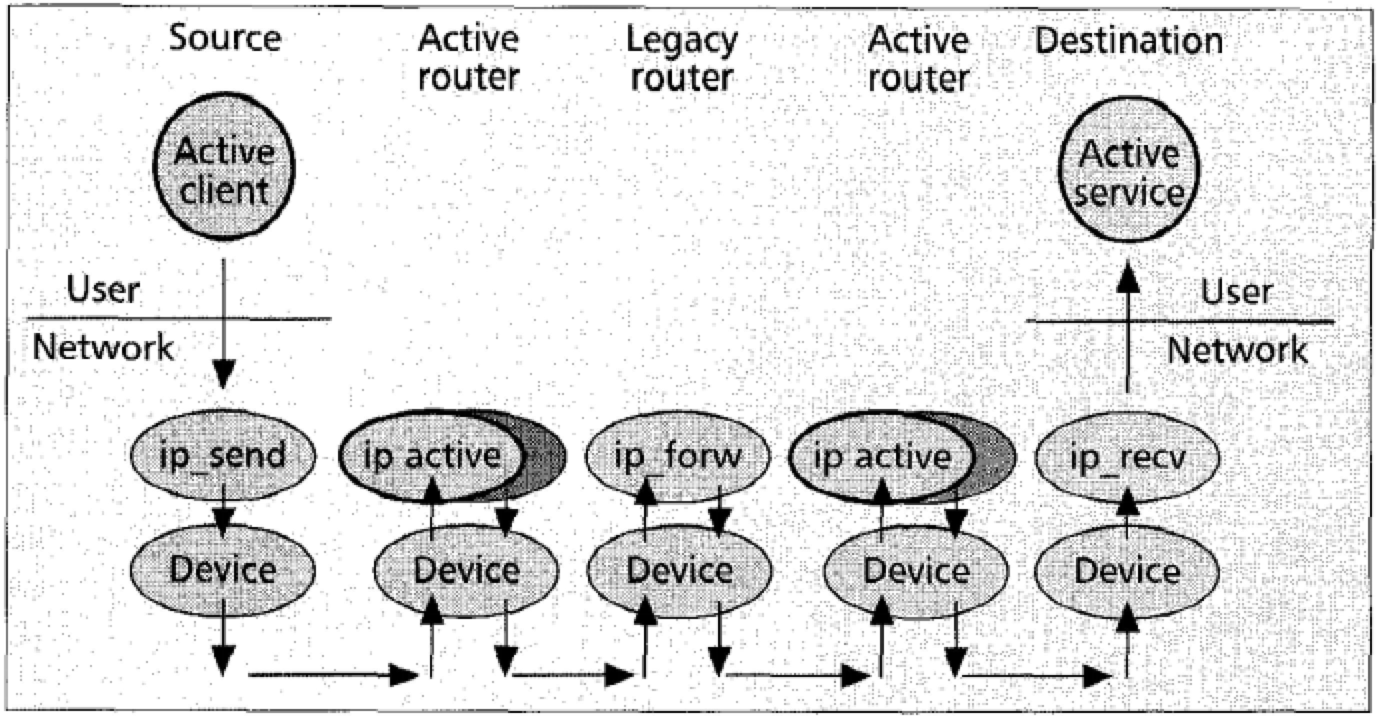


Figure 1. *Application-specific processing within the nodes of an active network.*

Co-existence of legacy routers with active routers

[1] Tennenhouse, et al.: *"A survey of active network research." IEEE Communications Magazine,* 35.1 (1997): 80-86.

Why did active networks fail?

- Timing was off
  - End of 1990s: no data-centers/clouds yet
  - Hardware was expensive (compared to 2015)

- Conceptual mistakes:
  - Programmable by end-users (security?)
  - Limited interoperability

The Legacy of Active Networks

- Intellectual contributions of Active Networks:
  - Programmable network functions
  - Network virtualizations (de-multiplexing of packets according to their header)

The concepts behind SDN are not really new!

(we see both contributions in today‘s SDN)

[1] **Wetherall, et al**.: *”ANTS: a toolkit for building and dynamically deploying network protocols.”* In Proceedings of IEEE OpenArch 1998.
[2] **Bhattacharjee, Calvert, et al.:** “*An architecture for active networks”.* In Proceedings of High-Performance Networking 1997.

A brief history of programmable networks:

Control and data plane separation

Control and Data Plane Separation

- Early 2000s: increasing traffic volumes, network sizes
  - need for traffic engineering
- But: conventional routers/switches: tight integration of data and control planes
  - Problem: Hard to debug and control router behaviour
- Goal: Traffic control and configuration should be easier
- Envisioned method: decouple control and data plane

Control and Data Plane Separation

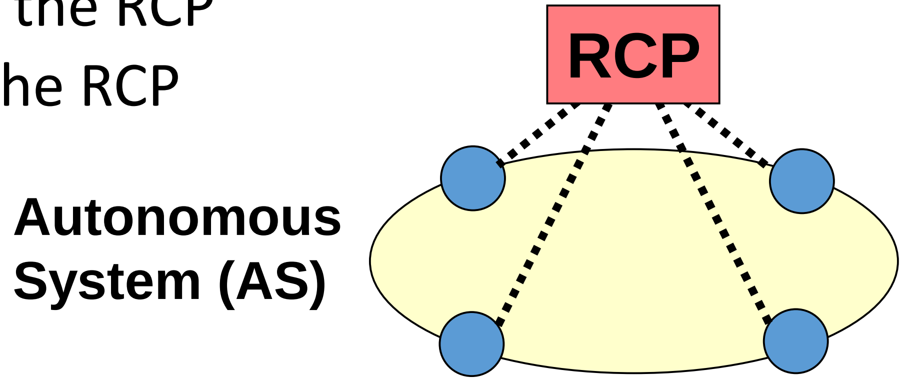
- Mainly two innovations:
  - Open interface between the control and data plane (e.g., ForCES [1])
  - *Logically* centralized control of the network (e.g., RCP [2])

- Compared to active networks:
  - **Targeted at network administrators** rather than end-users
  - **Programmability in control plane** rather than in data plane
  - **Network wide control** rather than device-level configuration

[1] **Yang, et al**. *"Forwarding and control element separation (ForCES) framework."* RFC 3746, April, 2004.
[2] **Caesar, et al.** *"Design and implementation of a routing control platform."* Proceedings of Usenix NSDI, 2005.

RCP - Separating *Inter*domain Routing [1]

- Compute interdomain routes for the routers
  - Input: BGP-learned routes from neighboring ASes
  - Output: forwarding-table entries for each router
- Backwards compatibility with legacy routers
  - RCP speaks to routers using BGP protocol
- Routers still run intradomain routing protocol
  - So the routers can reach the RCP
  - To reduce overhead on the RCP



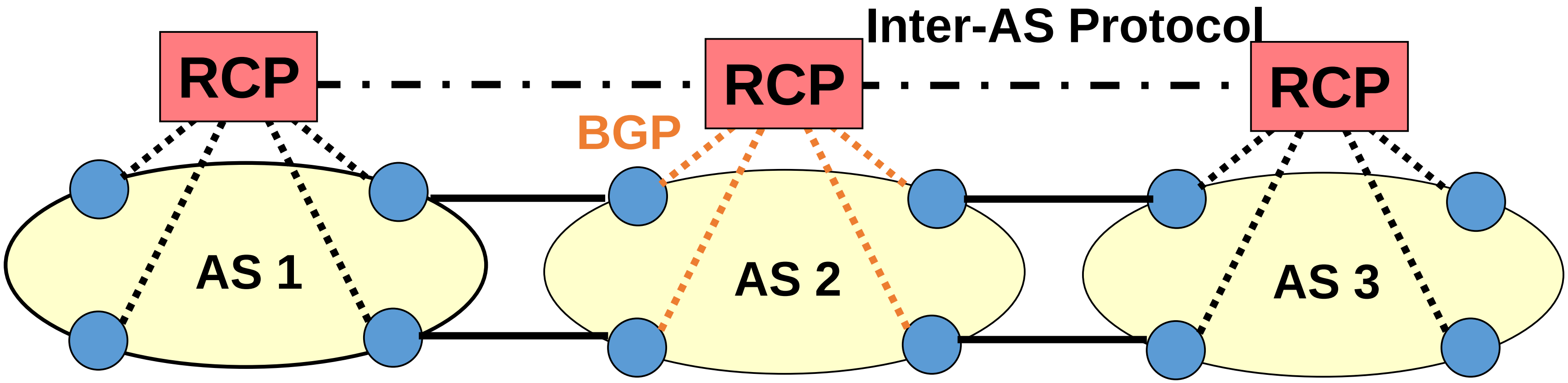
[1] **Caesar, et al.** *"Design and implementation of a routing control platform."* Proceedings of Usenix NSDI, 2005.

Incremental Deployability

- Backwards compatibility
  - Work with existing routers and protocols
- Incentive compatibility
  - Offer significant benefits, even to the first adopters
    - E.g., reducing overhead at routers

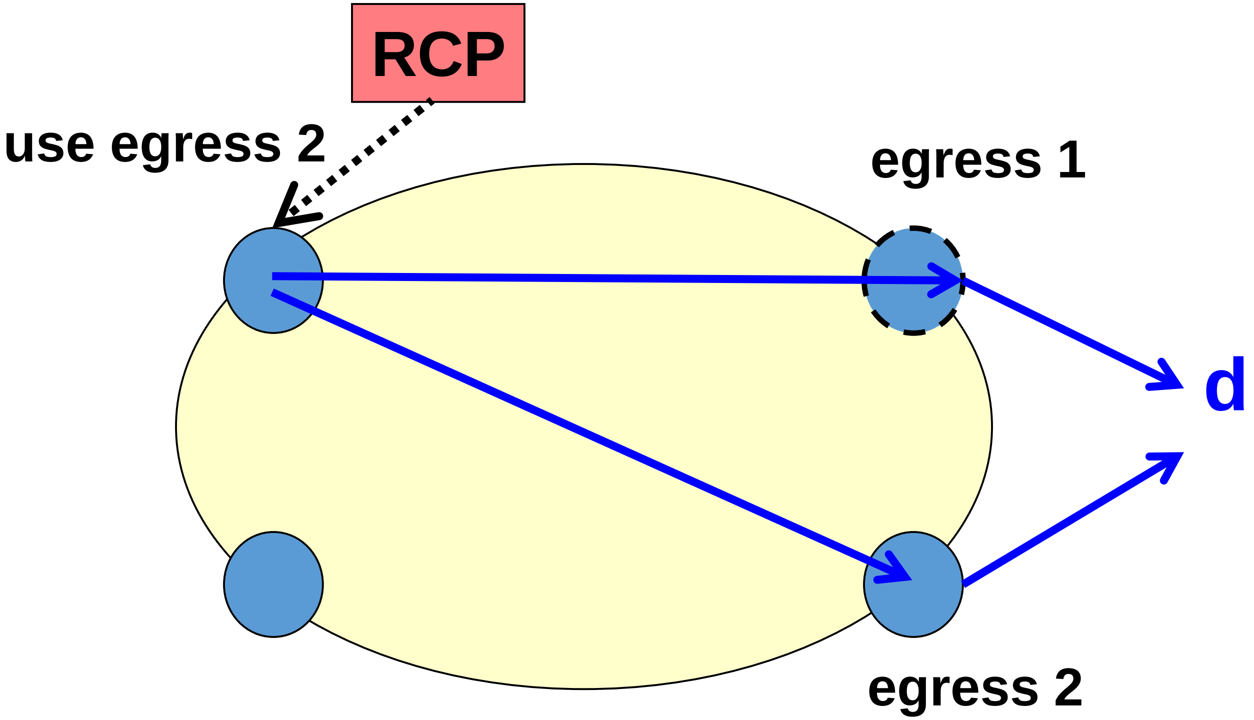
[illegible]

AS while using BGP to control the legacy routers



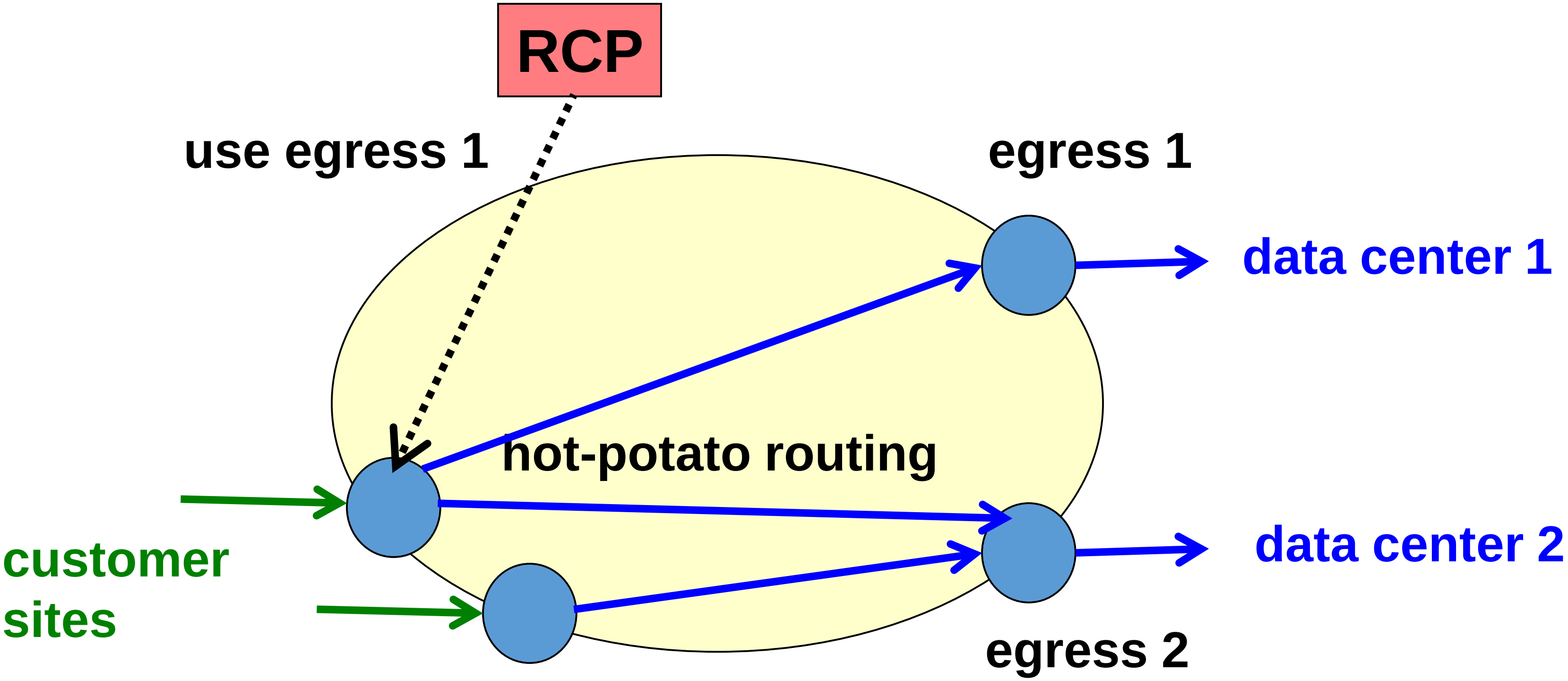
Example: Maintenance Dry-Out

- Planned maintenance on an edge router
  - Drain traffic off of an edge router
  - *Before* bringing it down for maintenance



Example: Egress Selection

- Customer-controlled egress selection
  - Multiple ways to reach the same destination
  - Giving customers control over the decision



RCP – The big "BUT"

- RCP still uses BGP, a single routing protocol
  - This is not what we need
  - However, we can learn from it!

The Legacy of the Separation

- Recall the two innovations:
  - Open interface between the control and data plane (e.g., ForCES [1])
  - *Logically* centralized control of the network (e.g., RCP [2])

The concepts behind SDN are not really new!

(we see both contributions in today‘s SDN)

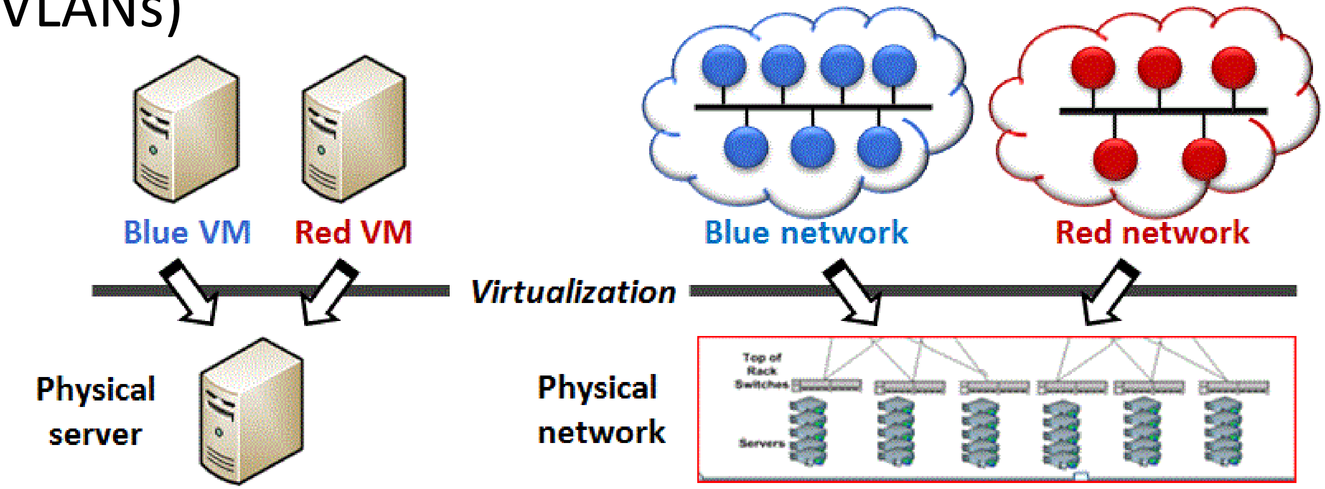
[1] **Yang, et al**. *”Forwarding and control element separation (ForCES) framework.”* RFC 3746, April, 2004.
[2] **Caesar, et al.** *"Design and implementation of a routing control platform."* Proceedings of Usenix NSDI, 2005.

A brief history of programmable networks:

Network virtualization

Network Virtualization?

- Abstraction of a network that is decoupled from theunderlying physical network (e.g., VLANs)



Server virtualization

- Run multiple virtual servers on a physical server
- Each VM has illusion it is running as a physical server

Network virtualization

- Run multiple virtual networks on a physical network
- Each virtual network has illusion it is running as a physical network

Microsoft Technet.: https://gallery.technet.microsoft.com/scriptcenter/Simple-Hyper-V-Network-d3efb3b8

Network Virtualization

First steps:

- Overlay networks as virtual networks on top of legacy technology
  - Own control protocol, encapsulation over legacy network (tunneling)
  - MBone [1] (for multicast), 6Bone [2] (for IPv6)

In contrast to active networks, overlay networks do not require any support from network equipment

Later:

Virtual networks inside the underlying network (e.g., VINI [3])

[1] **Almeroth et al**, "Multicast group behavior in the Internet's multicast backbone (MBone)." IEEE*Communications Magazine, IEEE*35.6
[2] **Fink et al,** *6bone (IPv6 testing address allocation) phaseout*. RFC 3701, March, 2004.
[3] **Bavier, et al.** *"In VINI veritas: realistic and controlled network experimentation."* ACM CCR. Vol. 36. No. 4. ACM, 2006.

How to Validate an Idea?

Emulation

VINI

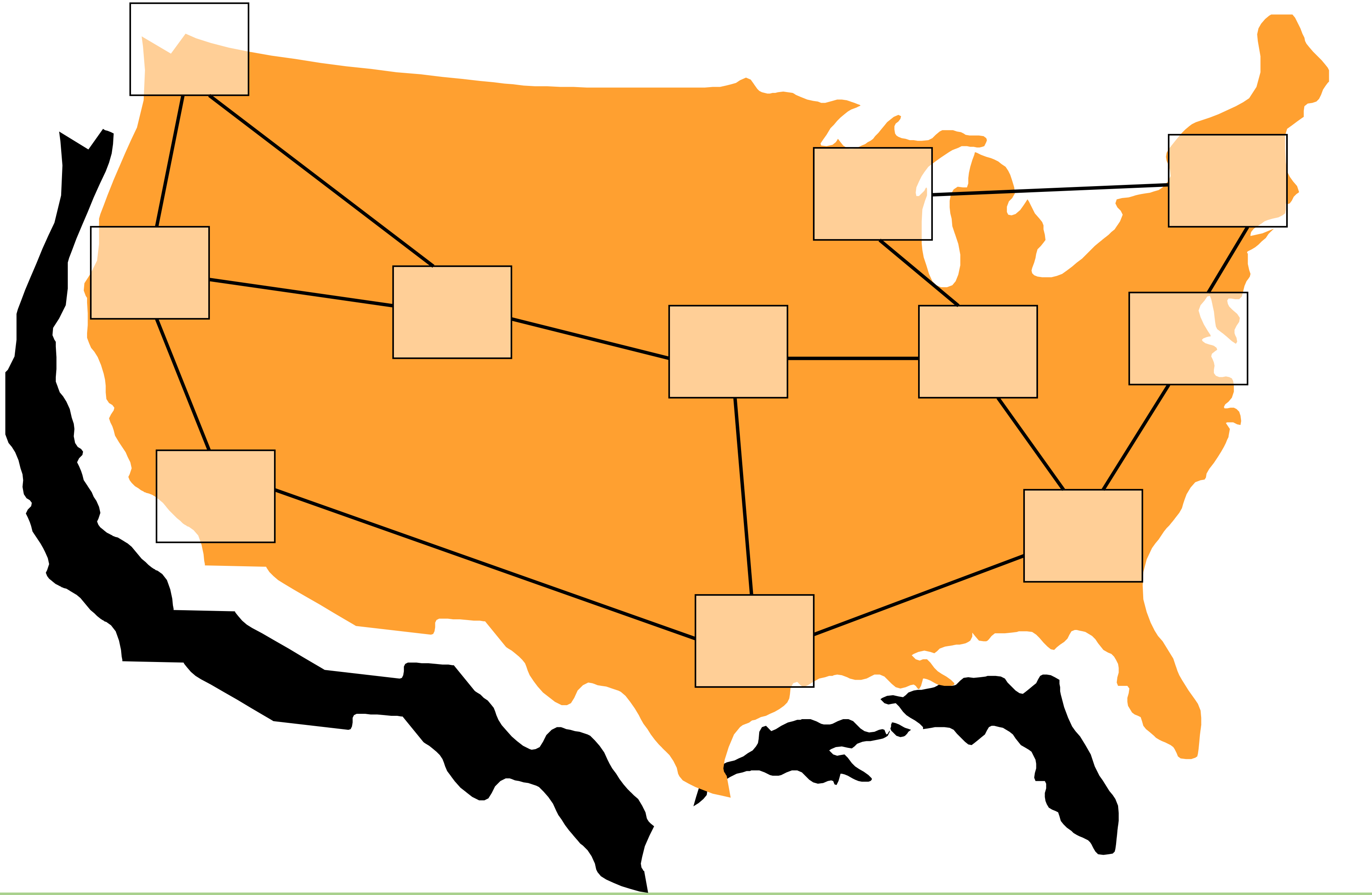
Simulation

Small-scale experiment

Live deployment

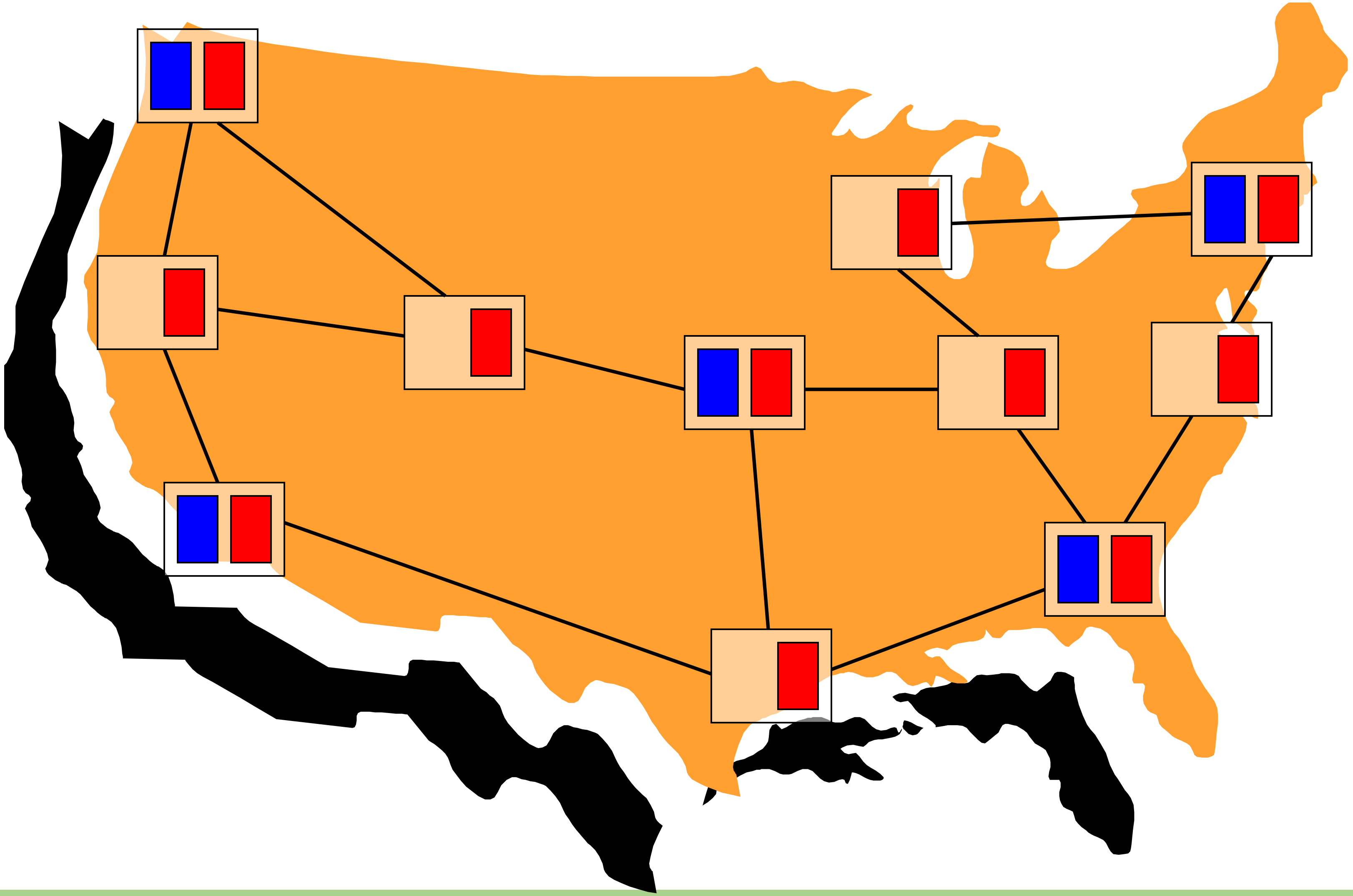
- Fixed infrastructure, shared among many experiments
- Runs real routing software
- Exposes realistic network conditions
- Gives control over network events
- Carries traffic on behalf of real users

Fixed Infrastructure

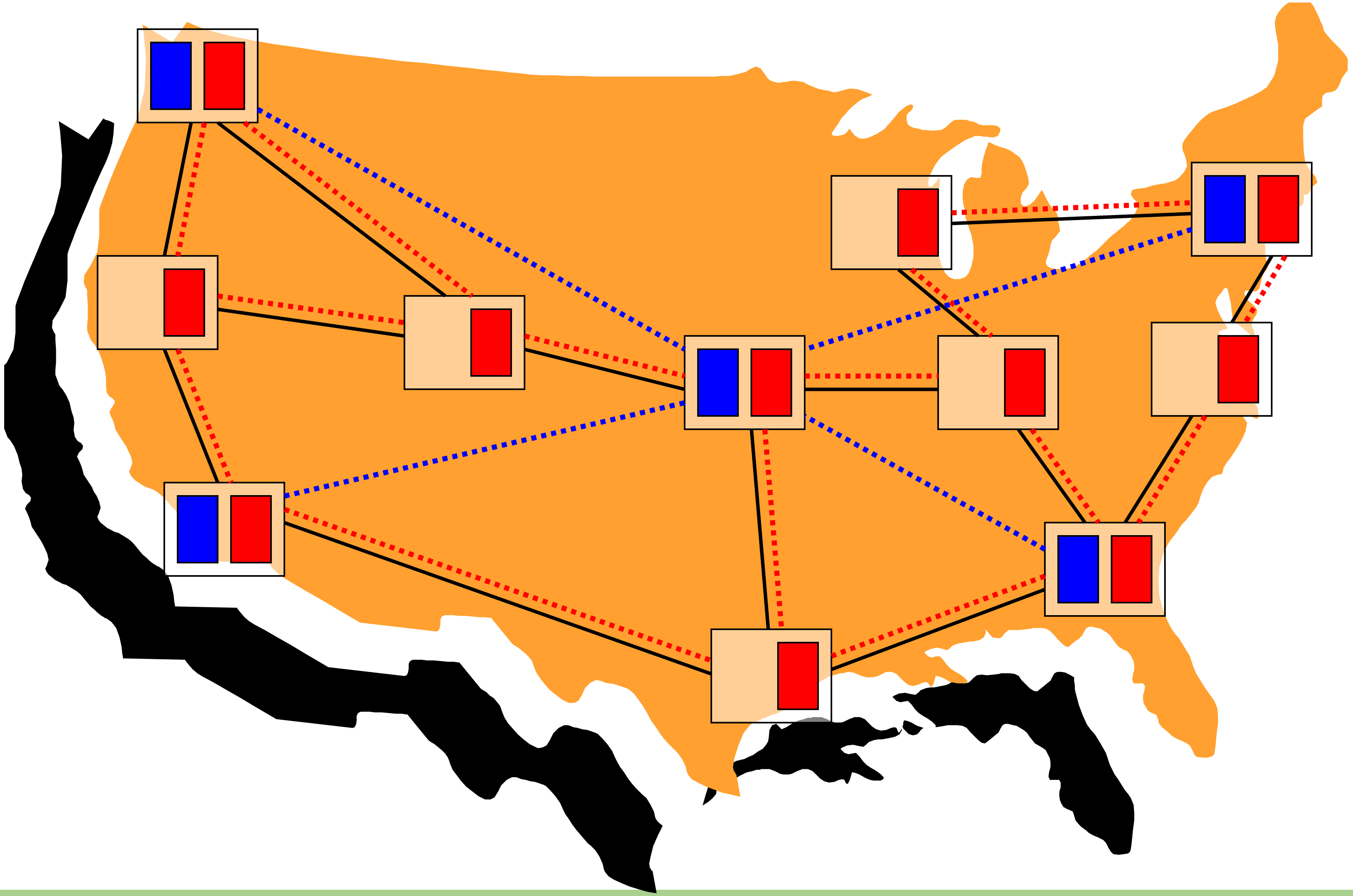


[3] Bavier, *"VINI and its future directions"*
http://www.fp7-federica.eu/pres_eventi/VINI_TNC2008.ppt

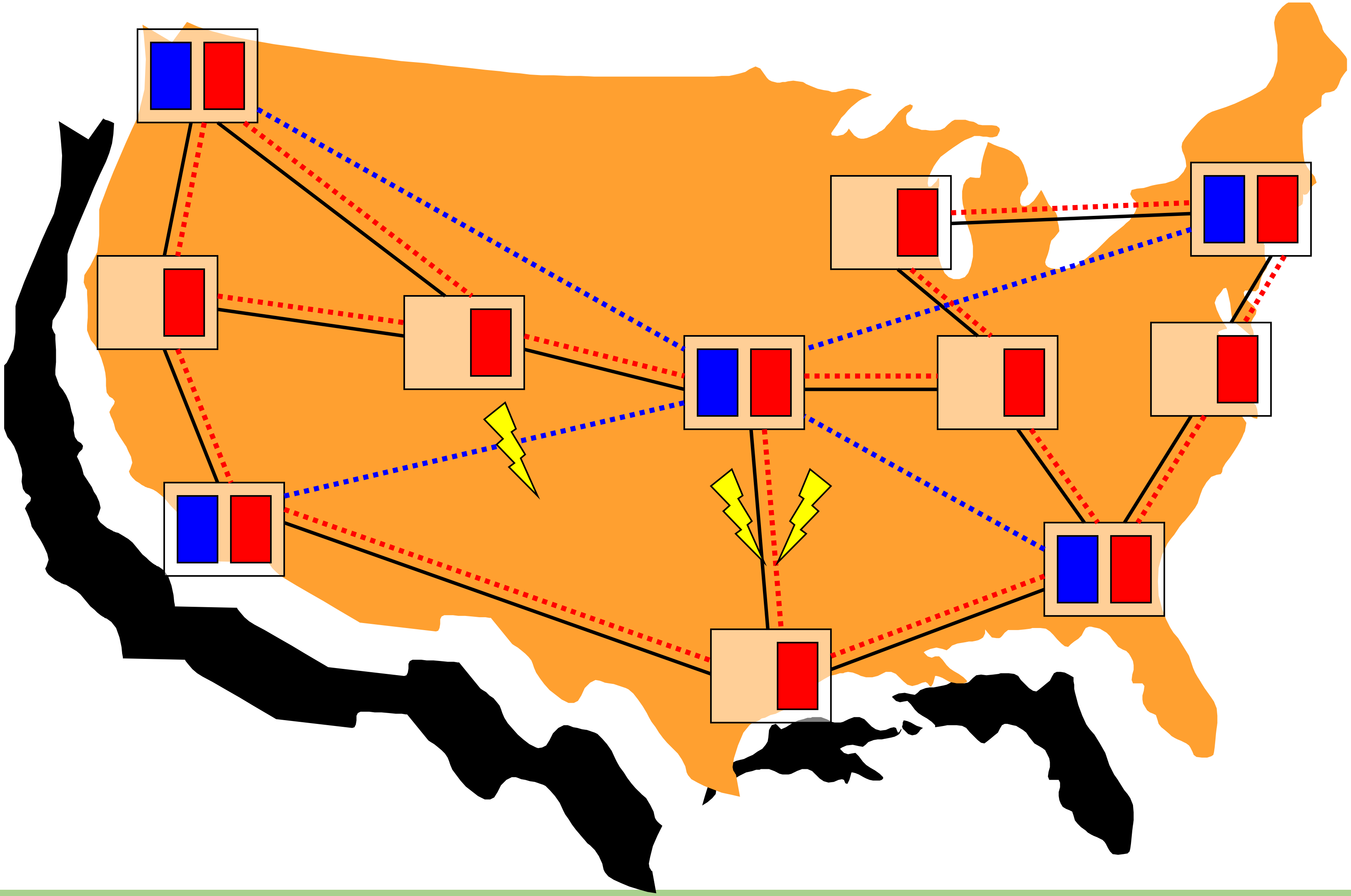
Shared Infrastructure



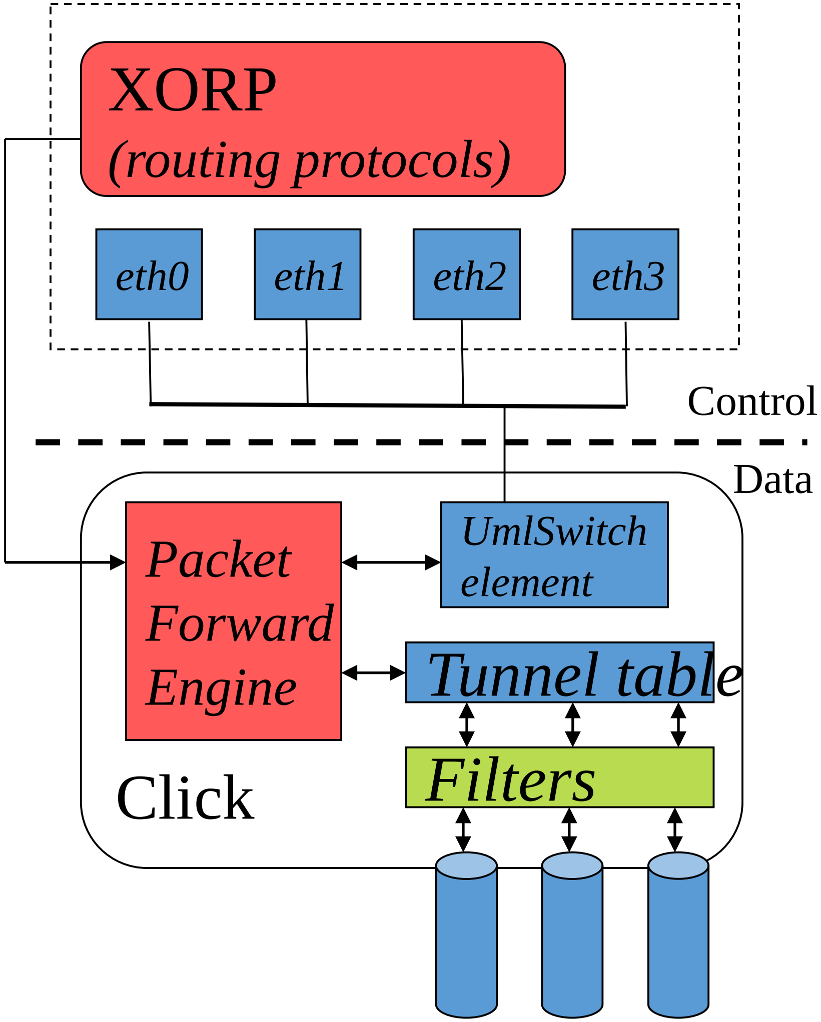
Flexible Topology



Network Events



VINI: Control/Data Plane Separation



- Interfaces ⇨ tunnels
  - Click UDP tunnels correspond to UML network interfaces
- Filters
  - “Fail a link” by blocking packets at tunnel

The Legacy of Network Virtualization

- Three main ideas
  - Separate service from infrastructure
  - Have multiple controllers (virtual networks) for the same switch
  - Logical network topologies

The concepts behind SDN are not really new!

(we see these contributions in today‘s SDN)

Control and Data Plane Separation – What *is* SDN actually?

SDN: Control and Data Plane Separation

Control Plane

logic for controlling the forwarding elements

routing protocols (e.g., BGP, OSPF), middlebox configuration, etc.

Data Plane

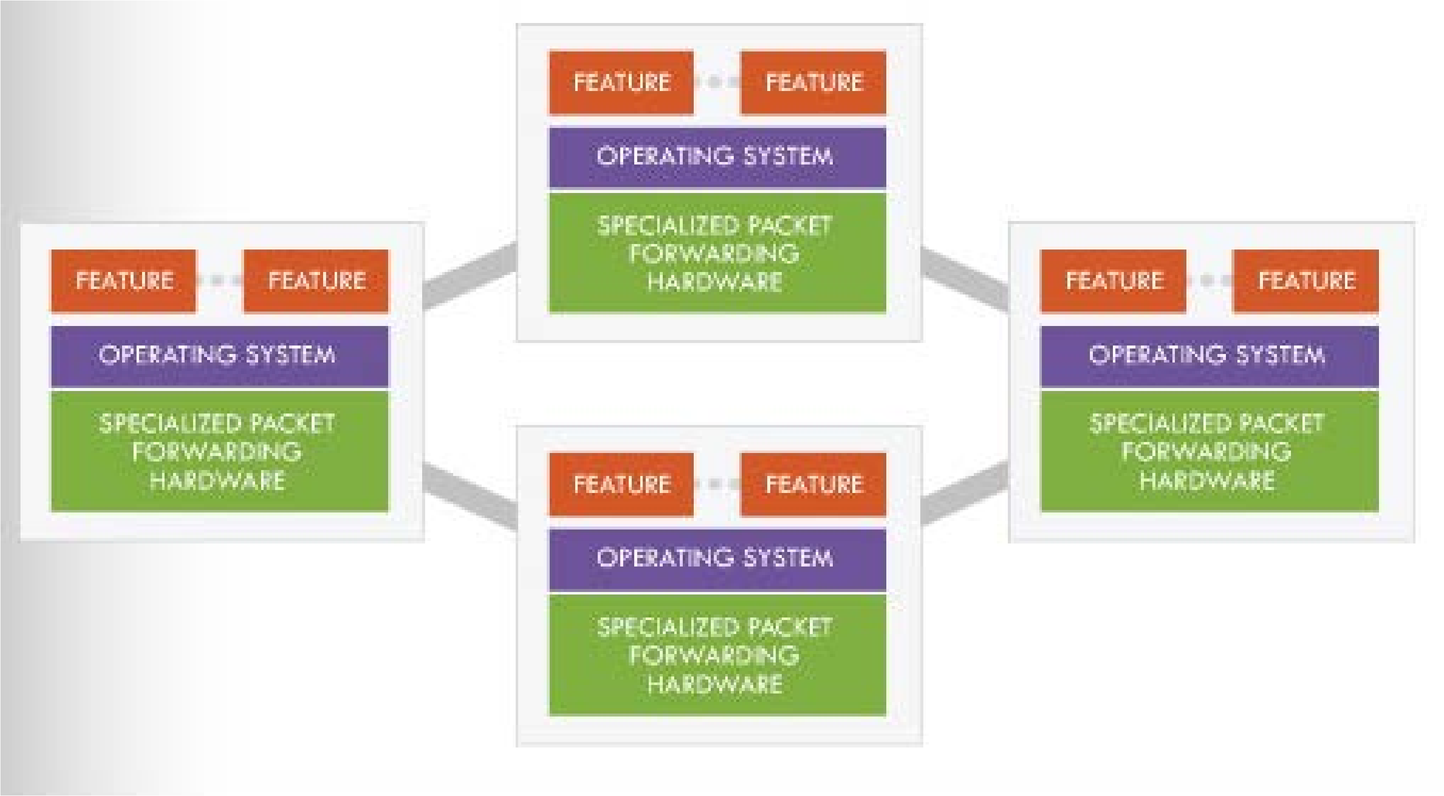
forward data based on rules set by the control logic

IP forwarding, layer 2 switching, etc.

<u>Today, routers implement both</u>

Why separate?

Currently, routers implement *both:*


FEATURE
FEATURE
OPERATING SYSTEM
SPECIALIZED PACKET FORWARDING HARDWARE
FEATURE
FEATURE
OPERATING SYSTEM
SPECIALIZED PACKET FORWARDING HARDWARE
FEATURE
FEATURE
OPERATING SYSTEM
SPECIALIZED PACKET FORWARDING HARDWARE
FEATURE
FEATURE
OPERATING SYSTEM
SPECIALIZED PACKET FORWARDING HARDWARE

Taken from: http://www.opennetsummit.org/archives/apr12/site/why.html

What do we gain from separating control and data plane?

Key to Internet Success: Layers

Applications

…built on…

Reliable (or unreliable) transport

…built on…

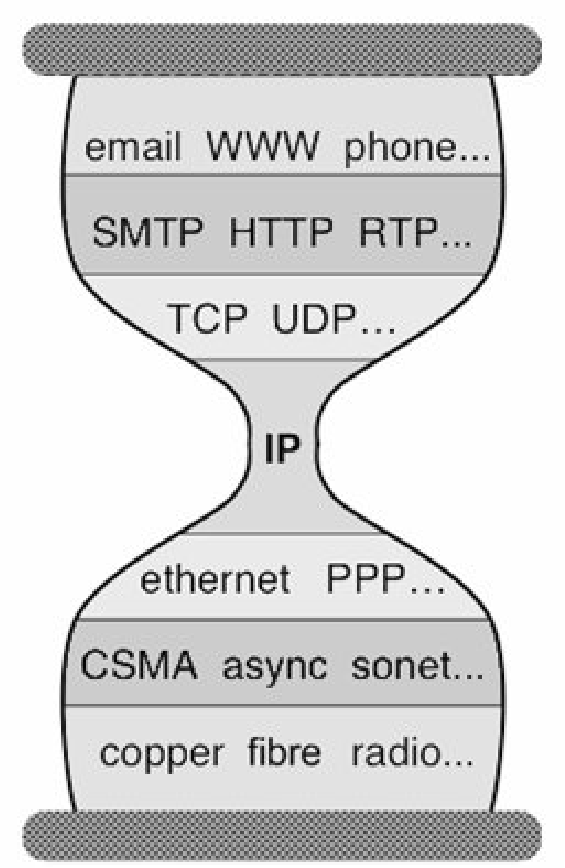
Best-effort global packet delivery

…built on…

Best-effort local packet delivery

…built on…

Physical transfer of bits



Why Is Layering So Important?

- It provides ***abstraction:*** decomposed delivery into fundamental components
- Independent but compatible innovation at each layer
- A practical success (it still works!)

The Problem in Computer Networks

- Layers only deal with the **data plane**
- We have no powerful ***control plane*** ***abstractions!***

Control Plane Abstractions

?

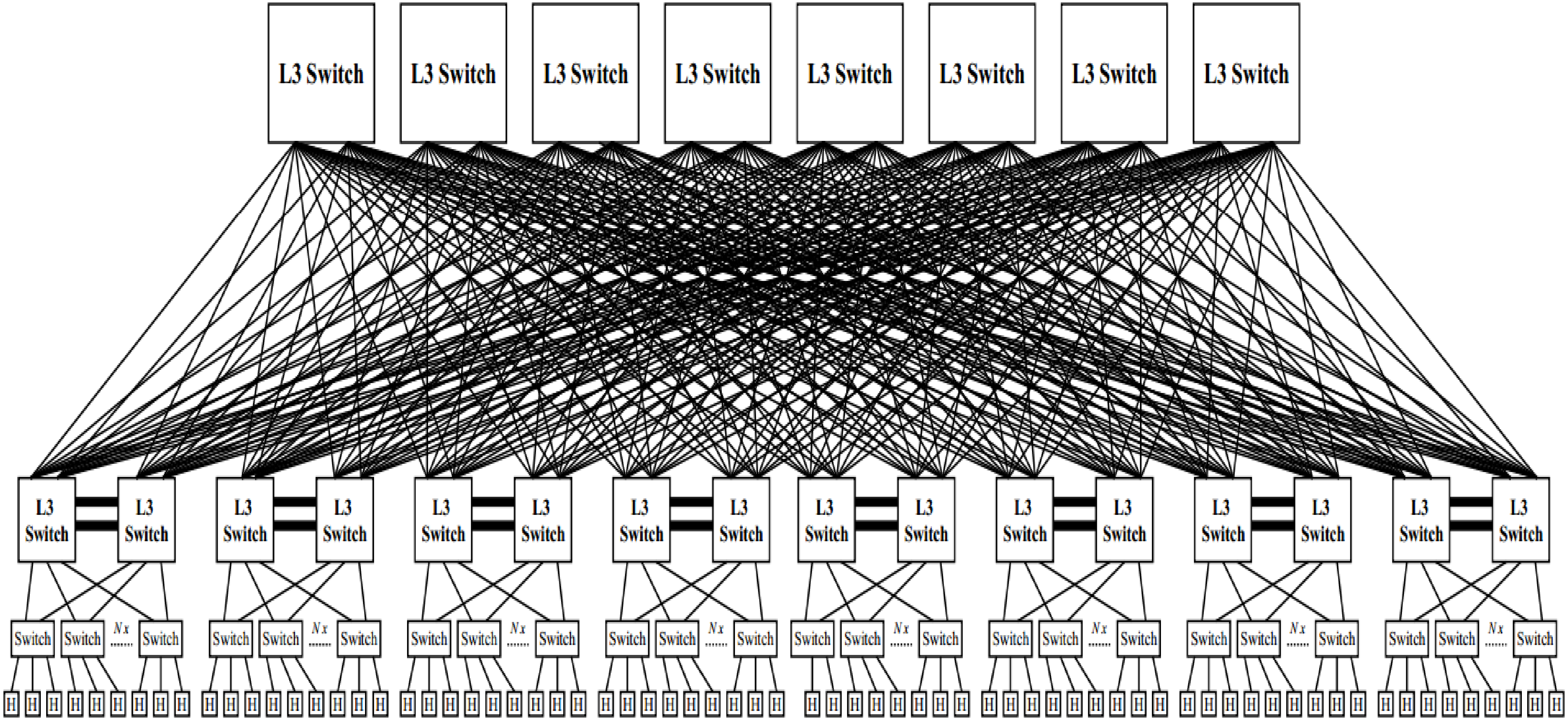
The Problem in Computer Networks

- Many different control plane mechanisms
- **Designed from scratch for specific goal**
- Variety of implementations
  - **Globally distributed:** routing algorithms
  - **Manual/scripted configuration:** ACLs, VLANs
  - **Centralized computation:** Traffic engineering
- Network control plane is a complicated mess!

The Problem in Computer Networks

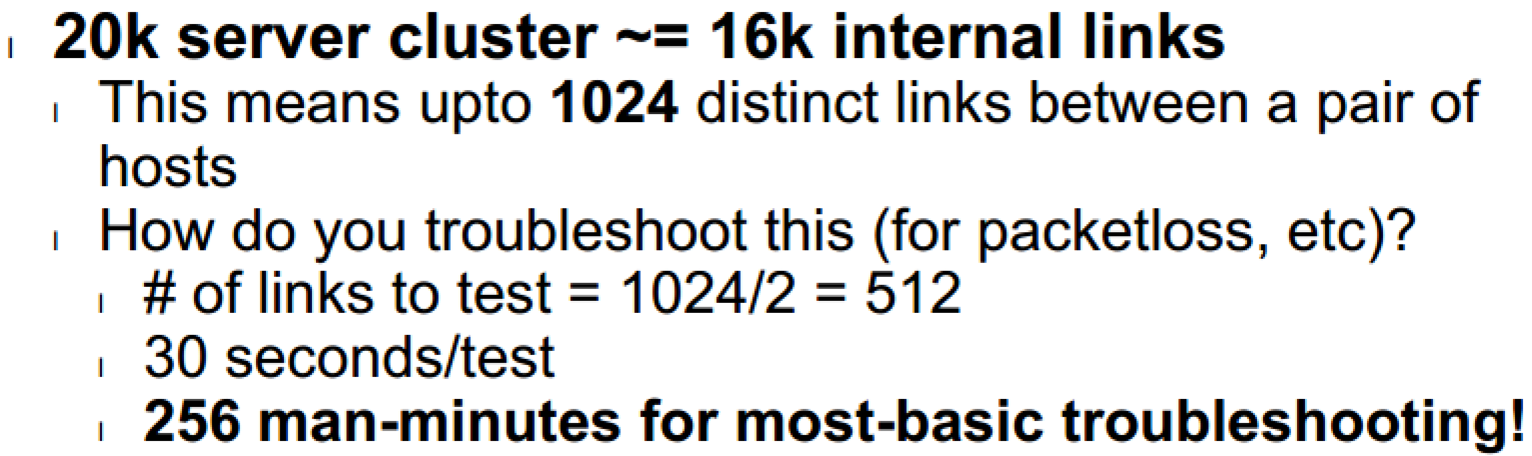
- Complexity has increased to “unmanageable” levels
- Consider datacenters:
  - 100,000s machines, 10,000s switches
  - 1000s of customers
    - Each with their own logical networks: ACLs, VLANs, etc
- Way beyond what we can handle
  - Leads to brittle, ossified configurations
  - Inefficient as well

Example: Datacenter Networks



Example: Datacenter Networks

- Complexity has increased to “unmanageable” levels



- **20k server cluster ~= 16k internal links**
  - This means upto **1024** distinct links between a pair of hosts
  - How do you troubleshoot this (for packetloss, etc)?
    - # of links to test = 1024/2 = 512
    - 30 seconds/test
    - **256 man-minutes for most-basic troubleshooting!**

The Problem is not only Complexity

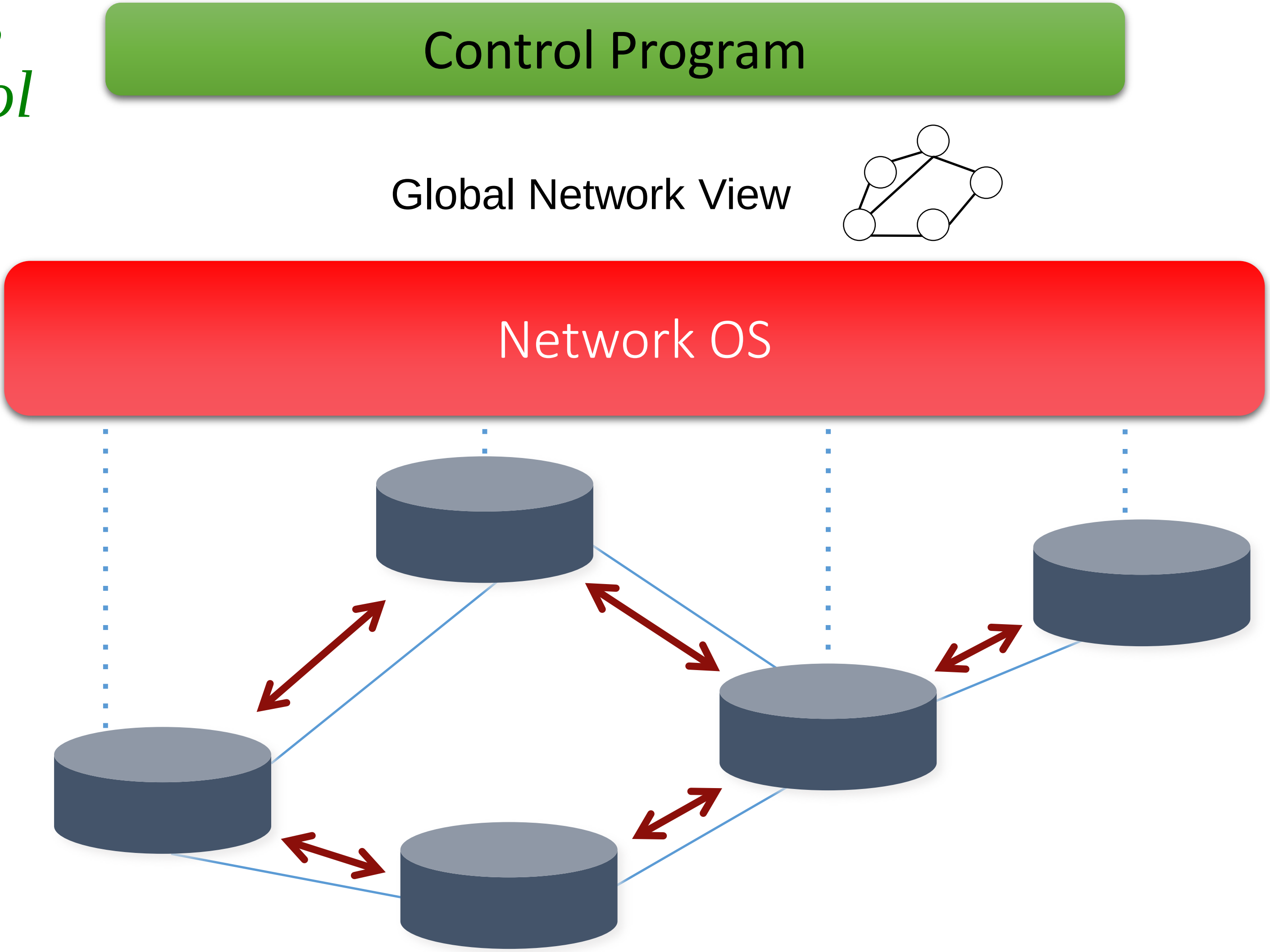
- Closed equipment
  - Software bundled with hardware
  - Vendor-specific interfaces
- Over specified
  - Slow protocol standardization
- Few people can innovate
  - Equipment vendors write the code
  - Long delays to introduce new features

Enter SDN

- Today, routers implement both planes
  - They forward packets
  - And run the control plane software
- SDN networks
  - Data plane implemented by switches
    - Switches act on local forwarding state
  - Control plane implemented by controllers
    - All forwarding state computed by SDN platform
  - Open protocols!
- **A technical change with broad implications**

Enter SDN

e.g. routing, access control

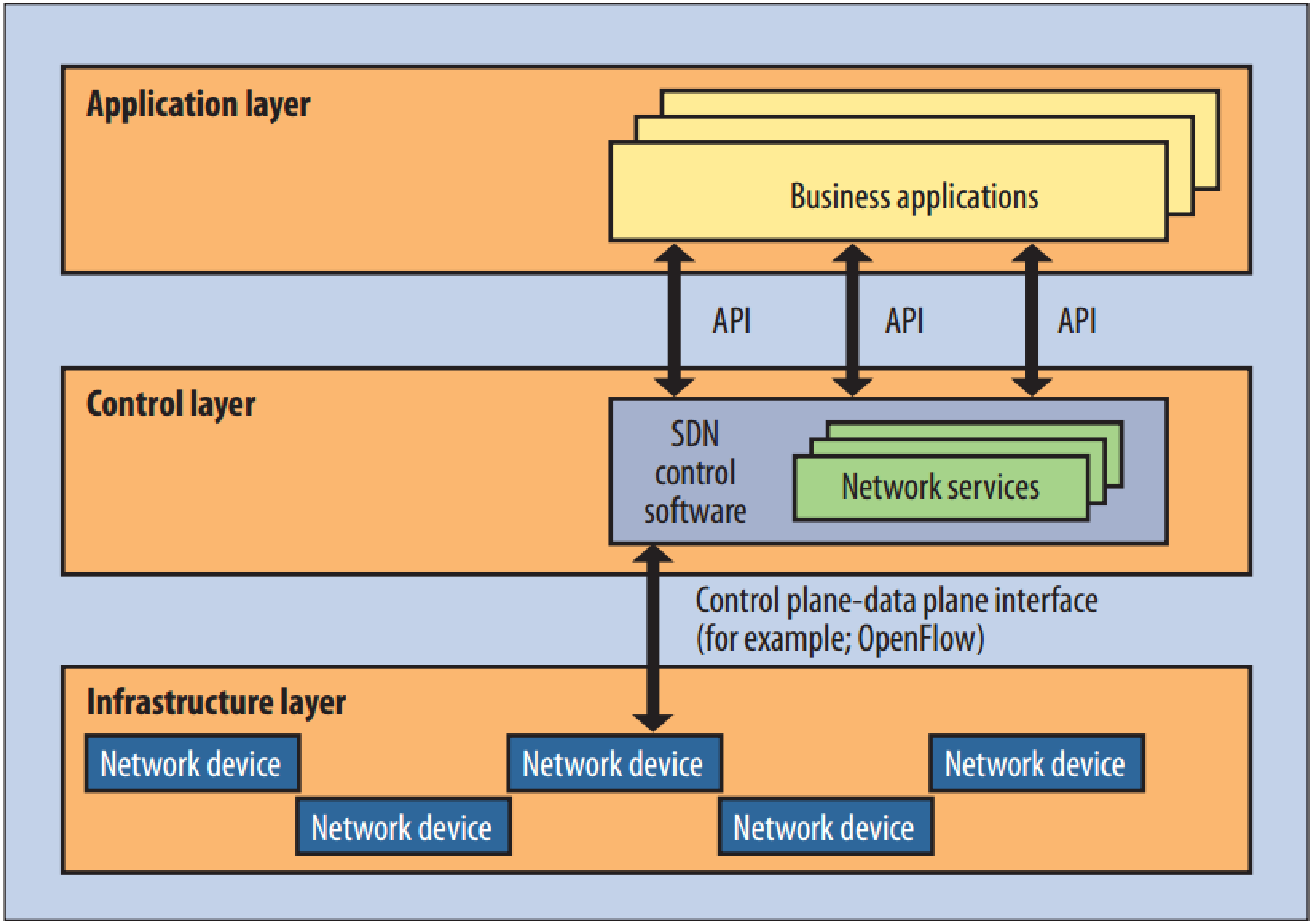


Control Program

Global Network View

Network OS

Another View



http://www.networkcomputing.com/networking/searching-for-an-sdn-definition-what-is-software-defined-networking/

Anology

- You are lost in a city and are trying to reach a destination
- Todays networks: ask other people you meet to obtain information (routing protocols)
- SDN: pull out your cellphone and start Google maps – it will calculate the route for you

Changes

- Less vendor lock-in
  - Can buy HW/SW from different vendors

- Changes are easier
  - Can test components separately
    - *HW has to forward*
    - *Can simulate controller*
    - *Can do verification on logical policy*
  - Can change topology and policy independently
  - Greater rate of innovation

Challenges of the Separation

- Talked a lot about the opportunities
- What about the challenges?

Practical Challenges

- Scalability
  - Control elements responsible for many routers
- Response time
  - Delays between control elements and routers
- Reliability
  - Surviving failures of control elements and routers
- Consistency
  - Ensuring multiple control elements behave consistently
- Security
  - Network vulnerable to attacks on control elements
- Interoperability
  - Legacy routers and neighboring domains

Example - Scalability

- Take routing: the controller has to make routing decisions for a lot of routers
  - Potentially 1000s

- Also has to store these routes
  - a lot of routing tables

- Single controller node for this task?
  - Compare with current standard OSPF: distributed

Current Status of SDN

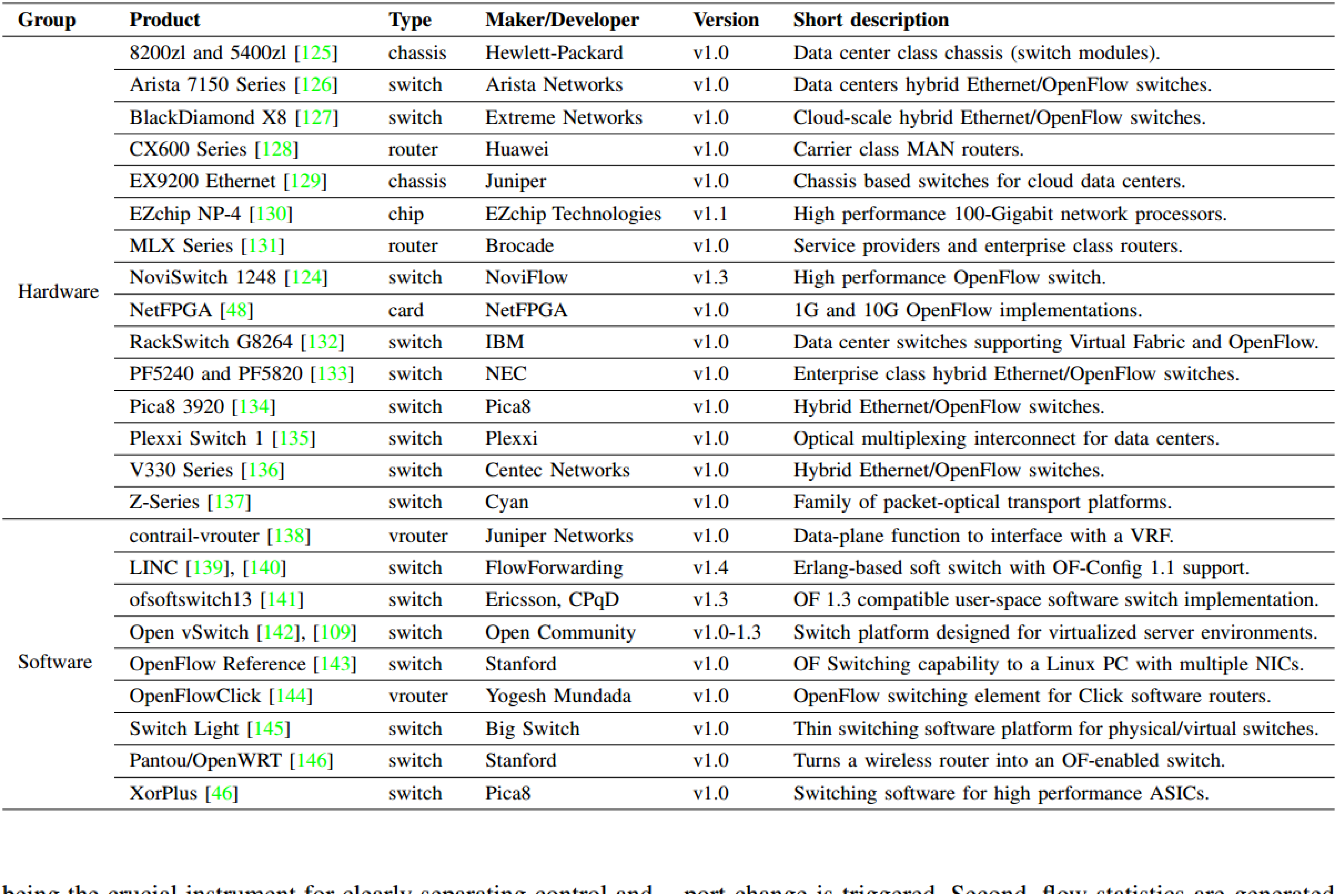
- SDN widely accepted as “**future of networking**”
  - ~1000 engineers at latest Open Networking Summit
  - *Much more acceptance in industry than in academia*

- Insane level of SDN hype, and still:
  - SDN doesn’t work miracles, merely makes things easier

Current Status of SDN

- Most innovations in southbound interface, controllers, northbound interface, and applications
  - OpenFlow (as ONE example of the sb interface)
  - NOX, POX, ONOS, etc.
  - Pyretic, Frenetic, etc.

- But: also changes in network devices
  - Most global players offer SDN switches now



Group	Product	Type	Maker/Developer	Version	Short description
Hardware	8200zl and 5400zl [125]	chassis	Hewlett-Packard	v1.0	Data center class chassis (switch modules).
	Arista 7150 Series [126]	switch	Arista Networks	v1.0	Data centers hybrid Ethernet/OpenFlow switches.
	BlackDiamond X8 [127]	switch	Extreme Networks	v1.0	Cloud-scale hybrid Ethernet/OpenFlow switches.
	CX600 Series [128]	router	Huawei	v1.0	Carrier class MAN routers.
	EX9200 Ethernet [129]	chassis	Juniper	v1.0	Chassis based switches for cloud data centers.
	EZchip NP-4 [130]	chip	EZchip Technologies	v1.1	High performance 100-Gigabit network processors.
	MLX Series [131]	router	Brocade	v1.0	Service providers and enterprise class routers.
	NoviSwitch 1248 [124]	switch	NoviFlow	v1.3	High performance OpenFlow switch.
	NetFPGA [48]	card	NetFPGA	v1.0	1G and 10G OpenFlow implementations.
	RackSwitch G8264 [132]	switch	IBM	v1.0	Data center switches supporting Virtual Fabric and OpenFlow.
	PF5240 and PF5820 [133]	switch	NEC	v1.0	Enterprise class hybrid Ethernet/OpenFlow switches.
	Pica8 3920 [134]	switch	Pica8	v1.0	Hybrid Ethernet/OpenFlow switches.
	Plexxi Switch 1 [135]	switch	Plexxi	v1.0	Optical multiplexing interconnect for data centers.
	V330 Series [136]	switch	Centec Networks	v1.0	Hybrid Ethernet/OpenFlow switches.
	Z-Series [137]	switch	Cyan	v1.0	Family of packet-optical transport platforms.
Software	contrail-vrouter [138]	vrouter	Juniper Networks	v1.0	Data-plane function to interface with a VRF.
	LINC [139], [140]	switch	FlowForwarding	v1.4	Erlang-based soft switch with OF-Config 1.1 support.
	ofsoftswitch13 [141]	switch	Ericsson, CPqD	v1.3	OF 1.3 compatible user-space software switch implementation.
	Open vSwitch [142], [109]	switch	Open Community	v1.0-1.3	Switch platform designed for virtualized server environments.
	OpenFlow Reference [143]	switch	Stanford	v1.0	OF Switching capability to a Linux PC with multiple NICs.
	OpenFlowClick [144]	vrouter	Yogesh Mundada	v1.0	OpenFlow switching element for Click software routers.
	Switch Light [145]	switch	Big Switch	v1.0	Thin switching software platform for physical/virtual switches.
	Pantou/OpenWRT [146]	switch	Stanford	v1.0	Turns a wireless router into an OF-enabled switch.
	XorPlus [46]	switch	Pica8	v1.0	Switching software for high performance ASICs.

being the crucial instrument for clearly separating control and

port change is triggered. Second, flow statistics are generated

Examples of Current SDN Hardware

Juniper MX-series

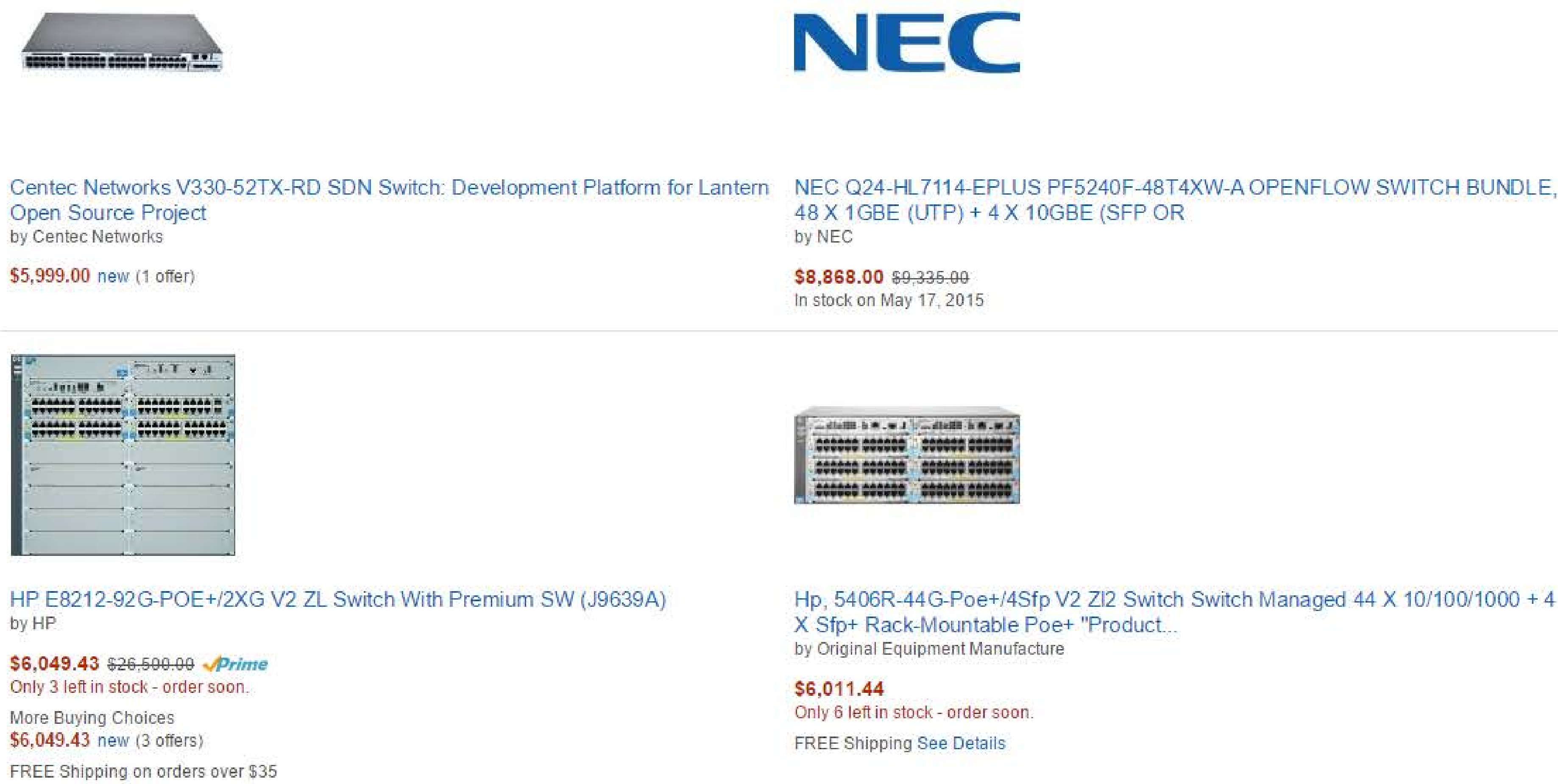
NEC IP8800

WiMax (NEC)

HP Procurve 5400

Netgear 7324

Examples of Current SDN Hardware



Centec Networks V330-52TX-RD SDN Switch: Development Platform for Lantern Open Source Project
by Centec Networks

$5,999.00 new (1 offer)

NEC Q24-HL7114-EPLUS PF5240F-48T4XW-A OPENFLOW SWITCH BUNDLE, 48 X 1GBE (UTP) + 4 X 10GBE (SFP OR
by NEC

$8,868.00 ~~$9,335.00~~
In stock on May 17, 2015

HP E8212-92G-POE+/2XG V2 ZL Switch With Premium SW (J9639A)
by HP

$6,049.43 ~~$26,500.00~~ ✓Prime
Only 3 left in stock - order soon.

More Buying Choices
$6,049.43 new (3 offers)

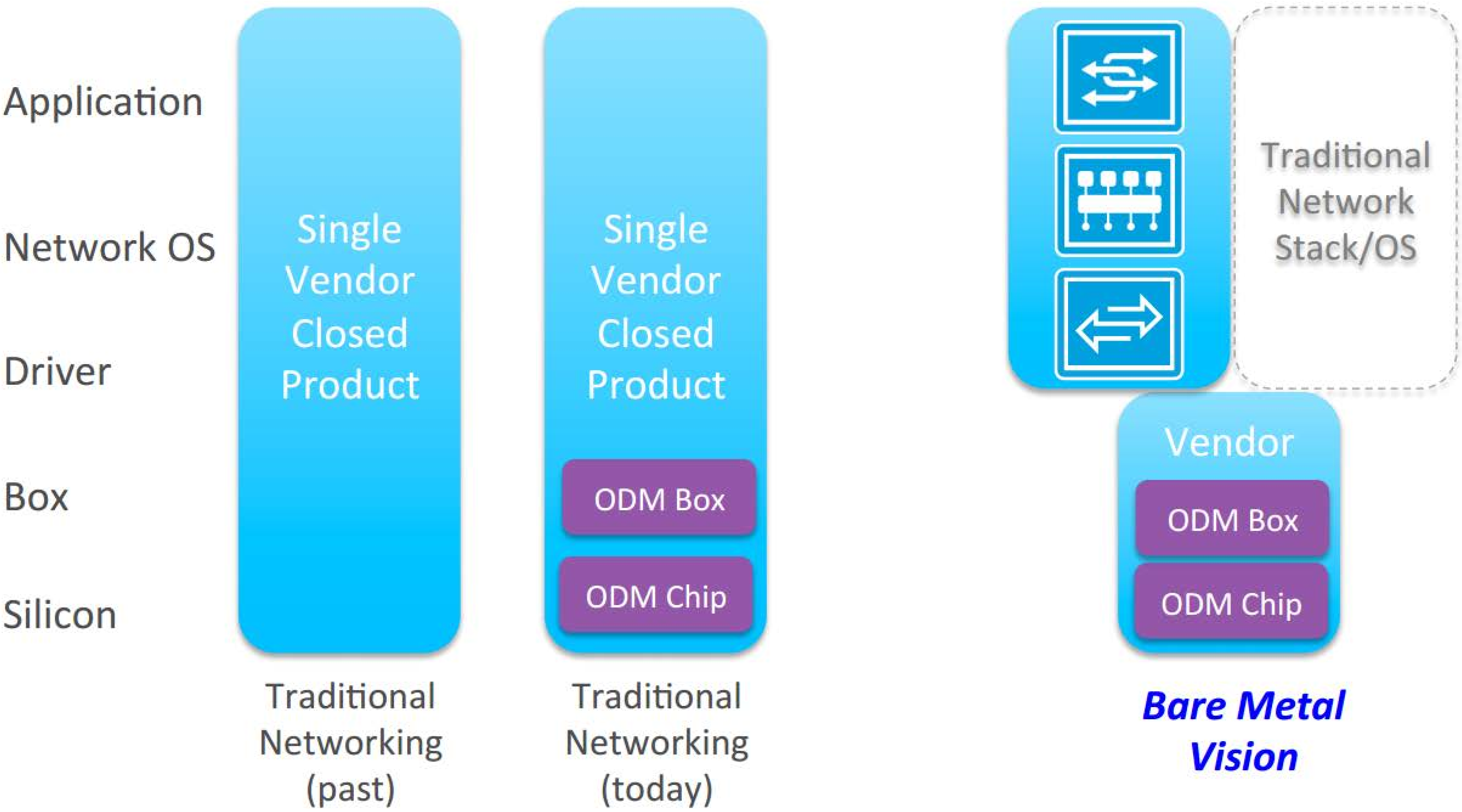
FREE Shipping on orders over $35

Hp, 5406R-44G-Poe+/4Sfp V2 Zl2 Switch Switch Managed 44 X 10/100/1000 + 4 X Sfp+ Rack-Mountable Poe+ "Product...
by Original Equipment Manufacture

$6,011.44
Only 6 left in stock - order soon.

FREE Shipping See Details

SDN = Open, also in HW



Application

Network OS

Driver

Box

Silicon

Single Vendor Closed Product

Traditional Networking (past)

Single Vendor Closed Product

ODM Box

ODM Chip

Traditional Networking (today)

Traditional Network Stack/OS

Vendor

ODM Box

ODM Chip

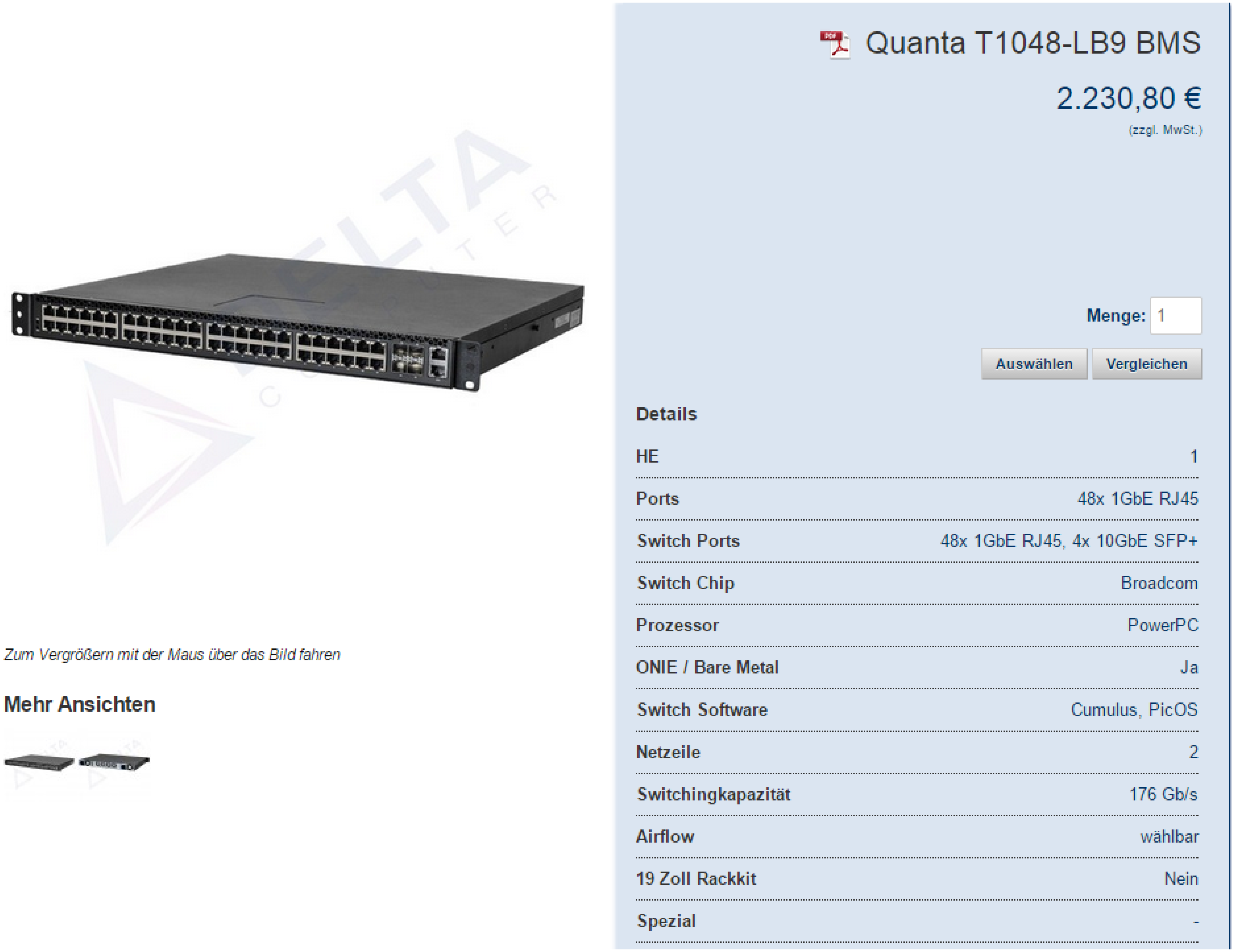
Bare Metal Vision

Onie

The Open Network Install Environment (ONIE) is **an open source initiative** that defines an open **"install environment" for bare metal network switches**. ONIE enables a bare metal network switch ecosystem where end users have a choice among different network operating systems.

In other words: Network OS Bios

Examples of HW



Zum Vergrößern mit der Maus über das Bild fahren

Mehr Ansichten

Quanta T1048-LB9 BMS

2.230,80 €

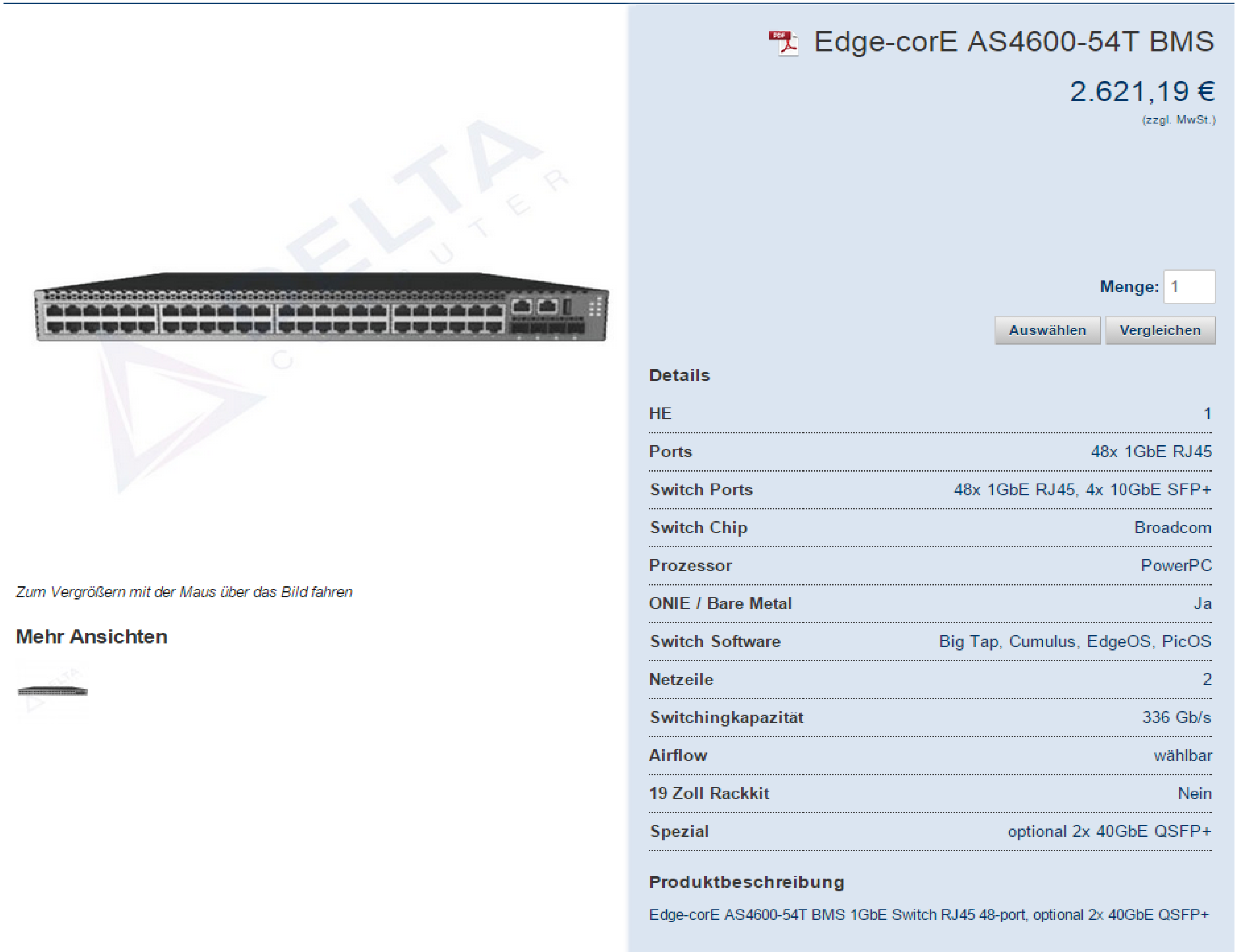
(zzgl. MwSt.)

Menge: 1

Details

HE	1
Ports	48x 1GbE RJ45
Switch Ports	48x 1GbE RJ45, 4x 10GbE SFP+
Switch Chip	Broadcom
Prozessor	PowerPC
ONIE / Bare Metal	Ja
Switch Software	Cumulus, PicOS
Netzeile	2
Switchingkapazität	176 Gb/s
Airflow	wählbar
19 Zoll Rackkit	Nein
Spezial	-

Examples of HW



Edge-corE AS4600-54T BMS

2.621,19 €
(zzgl. MwSt.)

Menge: 1

Zum Vergrößern mit der Maus über das Bild fahren

Mehr Ansichten

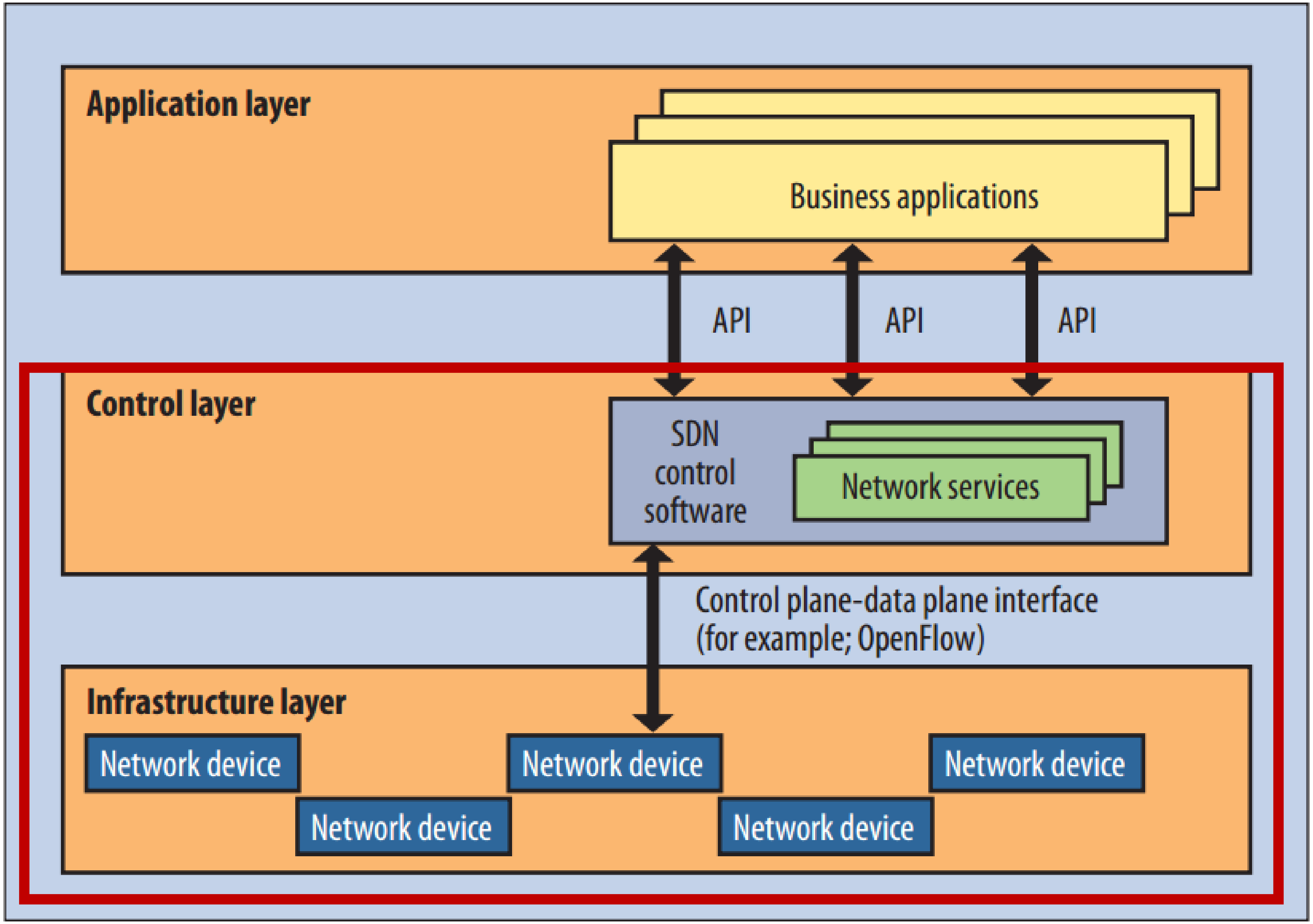
Details

HE	1
Ports	48x 1GbE RJ45
Switch Ports	48x 1GbE RJ45, 4x 10GbE SFP+
Switch Chip	Broadcom
Prozessor	PowerPC
ONIE / Bare Metal	Ja
Switch Software	Big Tap, Cumulus, EdgeOS, PicOS
Netzeile	2
Switchingkapazität	336 Gb/s
Airflow	wählbar
19 Zoll Rackkit	Nein
Spezial	optional 2x 40GbE QSFP+

Produktbeschreibung

Edge-corE AS4600-54T BMS 1GbE Switch RJ45 48-port, optional 2x 40GbE QSFP+

Up Next


Application layer
Business applications
API
API
API
Control layer
SDN control software
Network services
Control plane-data plane interface (for example; OpenFlow)
Infrastructure layer
Network device
Network device
Network device
Network device
Network device

OpenFlow – The de-facto standard Southbound interface

What is OpenFlow

OpenFlow is one implementation of the Southbound interface in SDN

OpenFlow is NOT SDN

OpenFlow is NOT THE ONLY Southbound interface

(see, e.g., Cisco OpFlex)

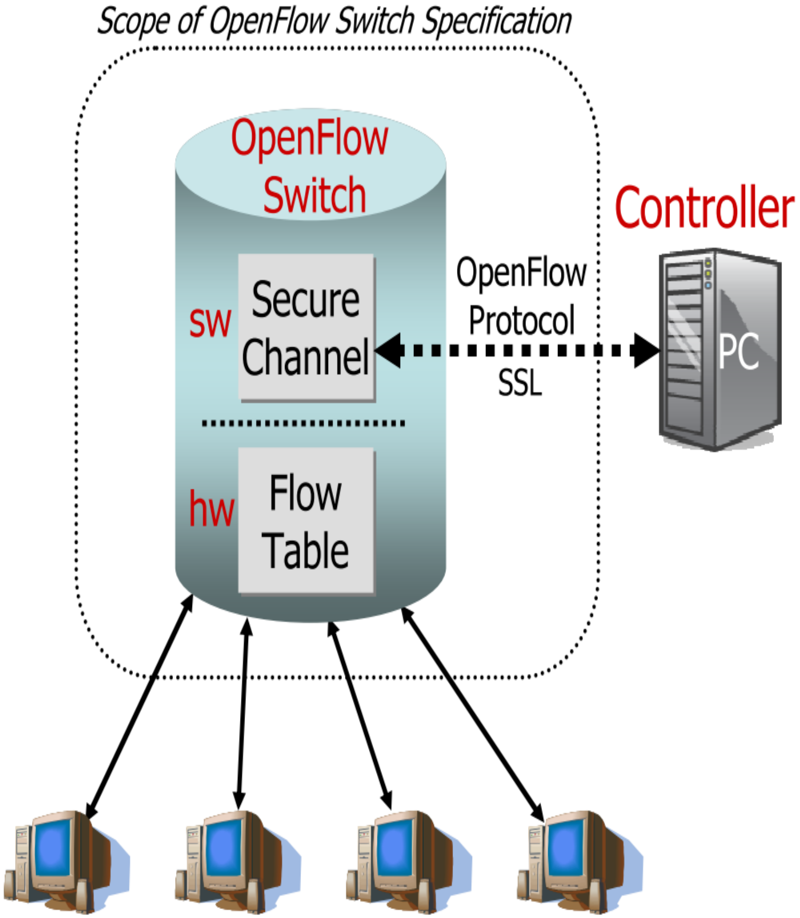
OpenFlow Consortium

http://OpenFlowSwitch.org

- Free membership for all researchers
- Whitepaper, OpenFlow Switch Specification, Reference Designs
- Licensing: Free for research and commercial use

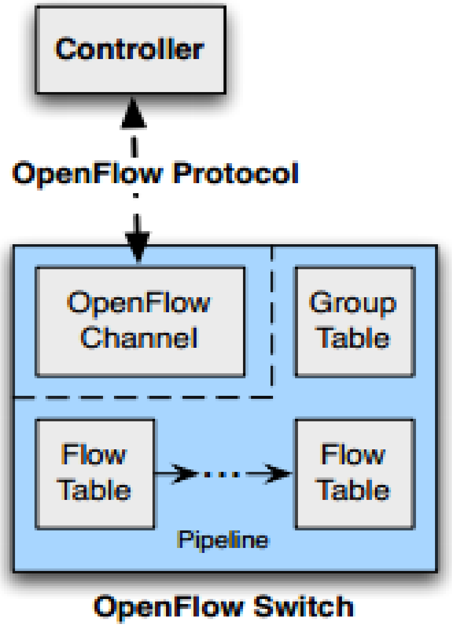
Components of OpenFlow Network

- Controller
  - OpenFlow protocol messages
  - Controlled channel
  - Processing
    - Pipeline Processing
    - Packet Matching
    - Instructions & Action Set
- OpenFlow switch
  - Secure Channel (SC)
  - Flow Table
    - Flow entry



OpenFlow

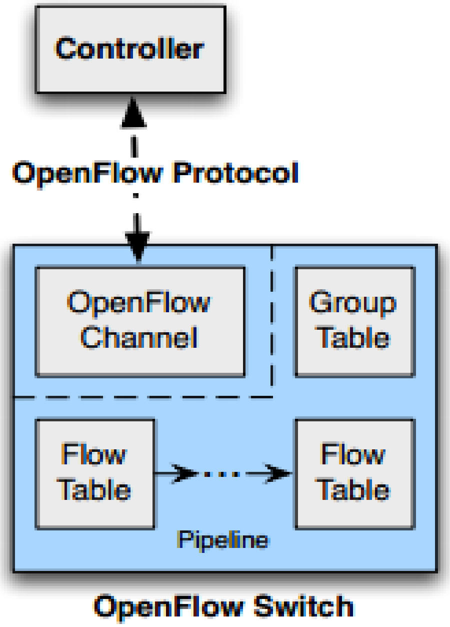
- Communication between the controller and the network devices (i.e., switches)



From the specification by the Open Networking Foundation: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf (Oct 2013)

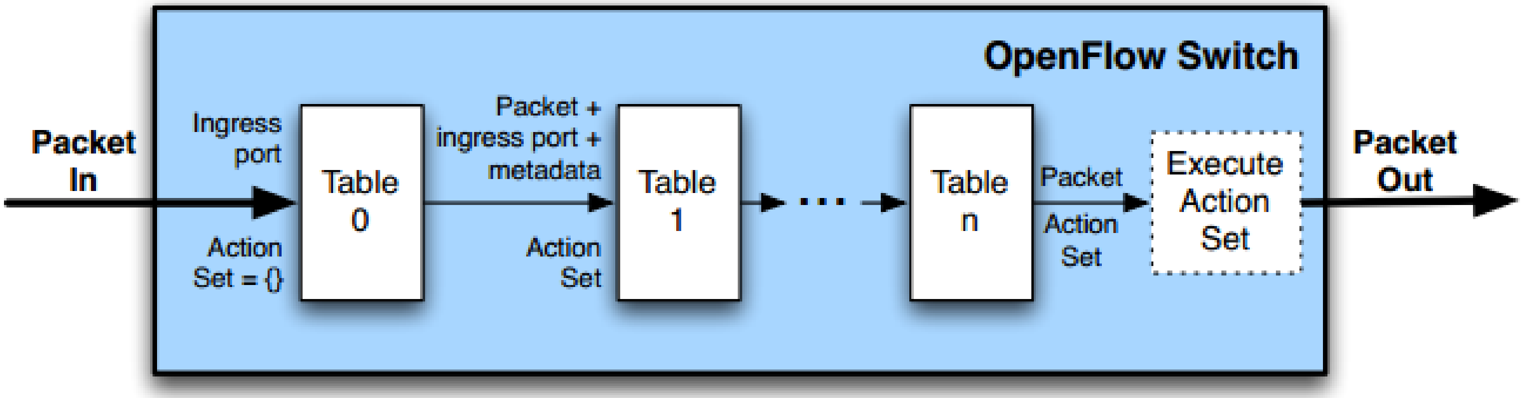
OpenFlow – A SDN Protocol

- Main components: *Flow* and *Group Tables*
  - Controller can manipulate these tables via the OpenFlow protocol (*add, update, delete*)
  - Flow Table: reactively or proactively defines how incoming packets are forwarded
  - Group Table: additional processing



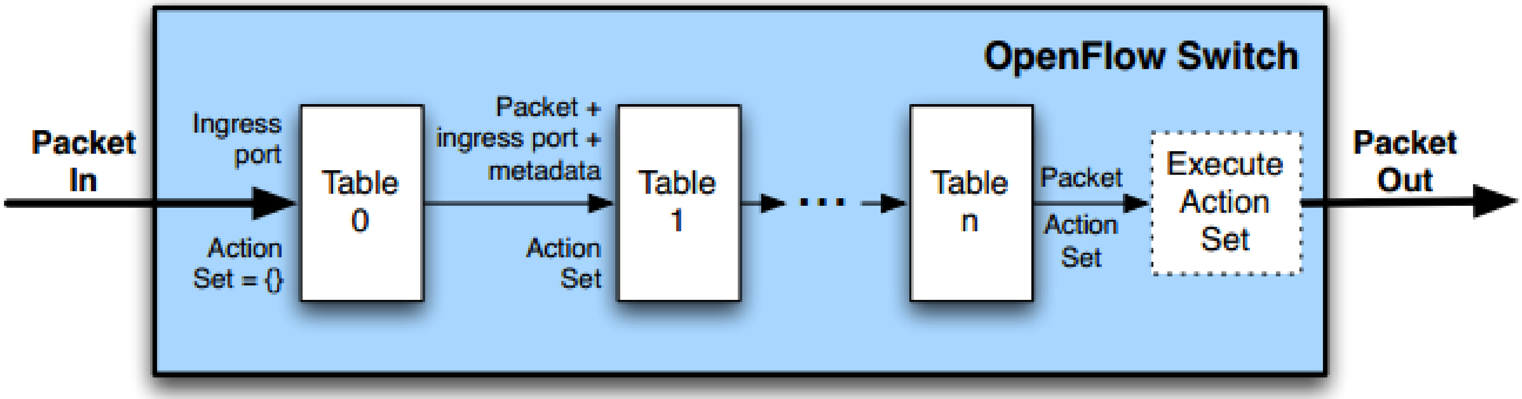
OpenFlow – Switches

- Two different versions of an OpenFlow Switch
  - *OF-only* (packets can only be processed by OF tables) and *OF-hybrid* (allow optional normal Ethernet handling (see CN lecture))
- OF-only: all packets go through a *pipeline*
  - Each pipeline contains one or multiple flow tables with each containing one or multiple *flow entries*



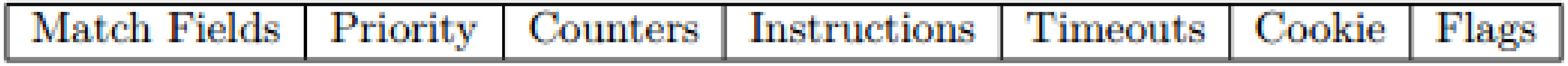
OpenFlow – Switches

- Incoming packets are matched against Table 0 first
- Find highest priority match and execute instructions (might be a Goto-Table instruction)
- Goto: Only possible forward



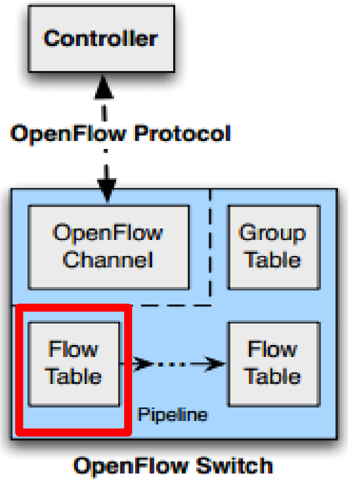
OpenFlow – Switches

- Flow Table entry structure:



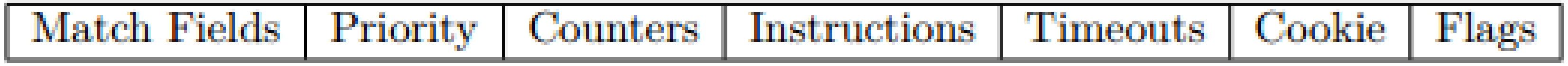
Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags

- Match fields: where matching applies
- Priority: matching precedence of flow entry
- Counters: update on packet match with entry
- Instructions: what to do with the packet
- Timeout: max idle time of flow before ending



OpenFlow – Switches

- Flow Table entry structure:



Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags

- Match fields: where matching applies (i.e., ingress port, packet (IP, eth) headers, etc.)

- A flow entry with all match fields as wildcard and priority 0: *table miss* entry

OpenFlow – Switches

- If no match in table: *table miss*
- Handling: depends on table configuration – might be *drop packet, forward to other table, forward to controller*
- Forward to controller allows to set up a flow entry (i.e., at the beginning of a flow)

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

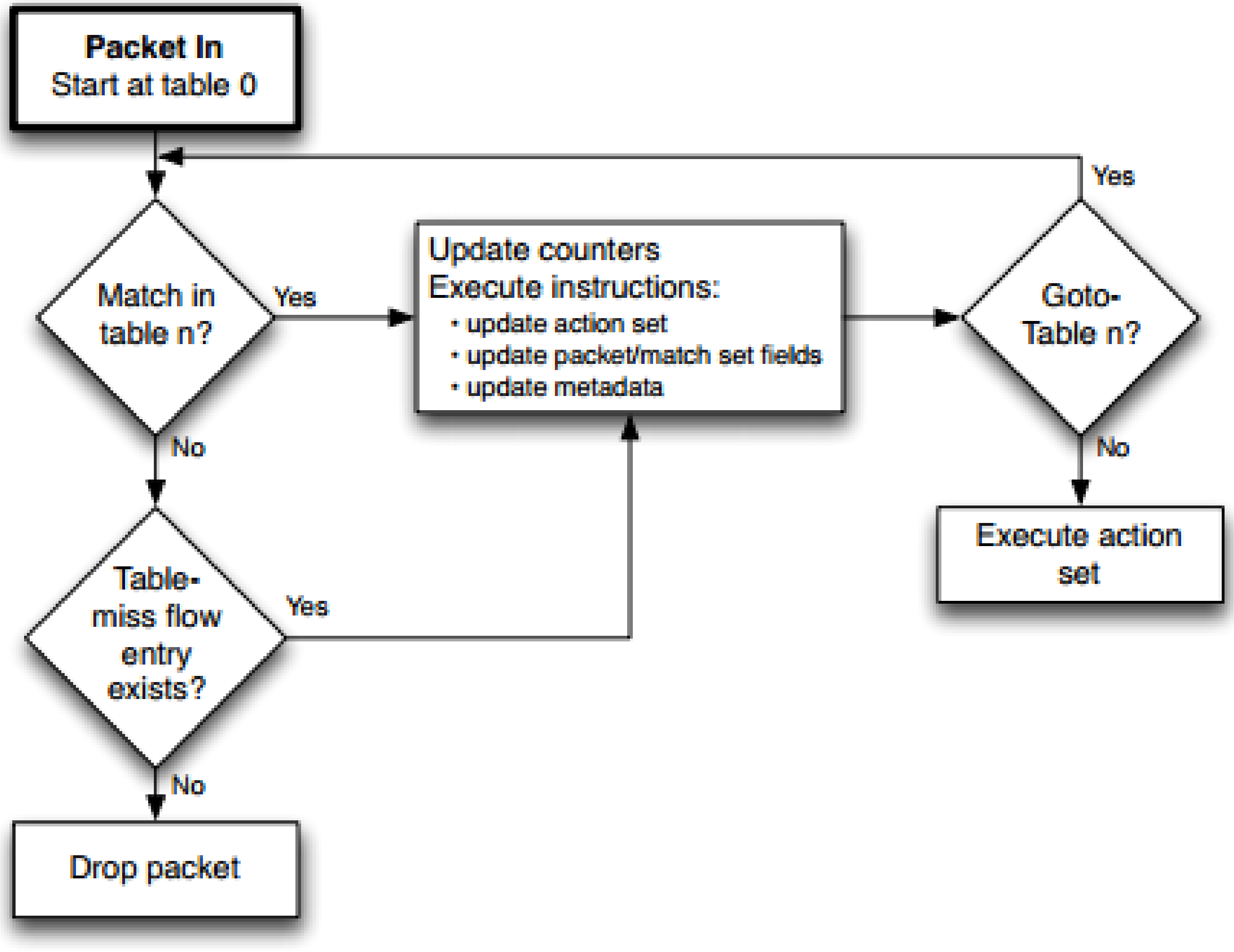
Routing

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

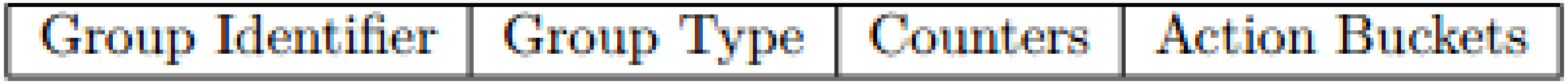
Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f..	*	vlan1	*	*	*	*	*	port6, port7, port9

OpenFlow - Matching



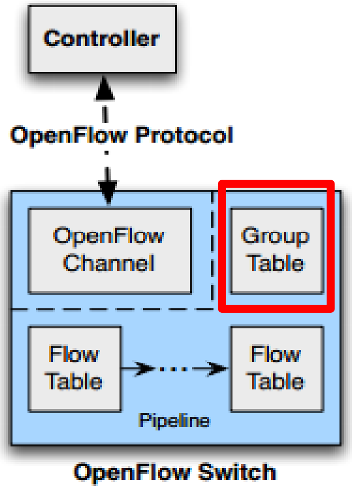
OpenFlow – Switches

- Group Table entry structure:



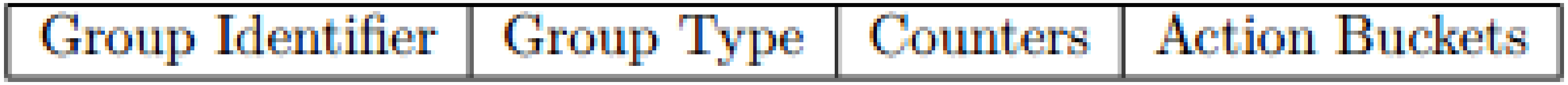
Group Identifier	Group Type	Counters	Action Buckets

- Group Identifier: 32-bit ID to uniquely define group on the switch (locally)
- Group Type: *indirect/all/fast failover/select*
    - Specifies which *action bucket* is executed
- Counters: update on packet processed
- Action Buckets: ordered list of buckets, each containing a *set* of instructions



OpenFlow – Switches

- Group Table entry structure:

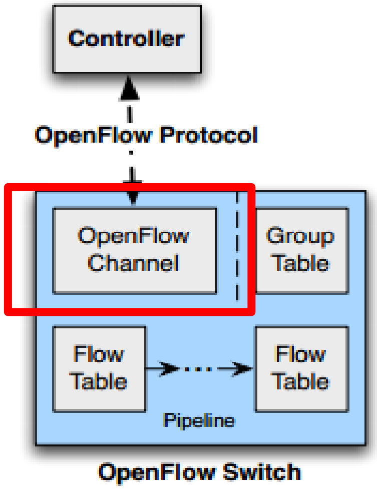


Group Identifier	Group Type	Counters	Action Buckets

- Group Tables allow for more complex forwarding
  - E.g., multicast: use *all* group type to execute all action buckets (packet will be cloned for each bucket, and then forwarded through the instruction set)

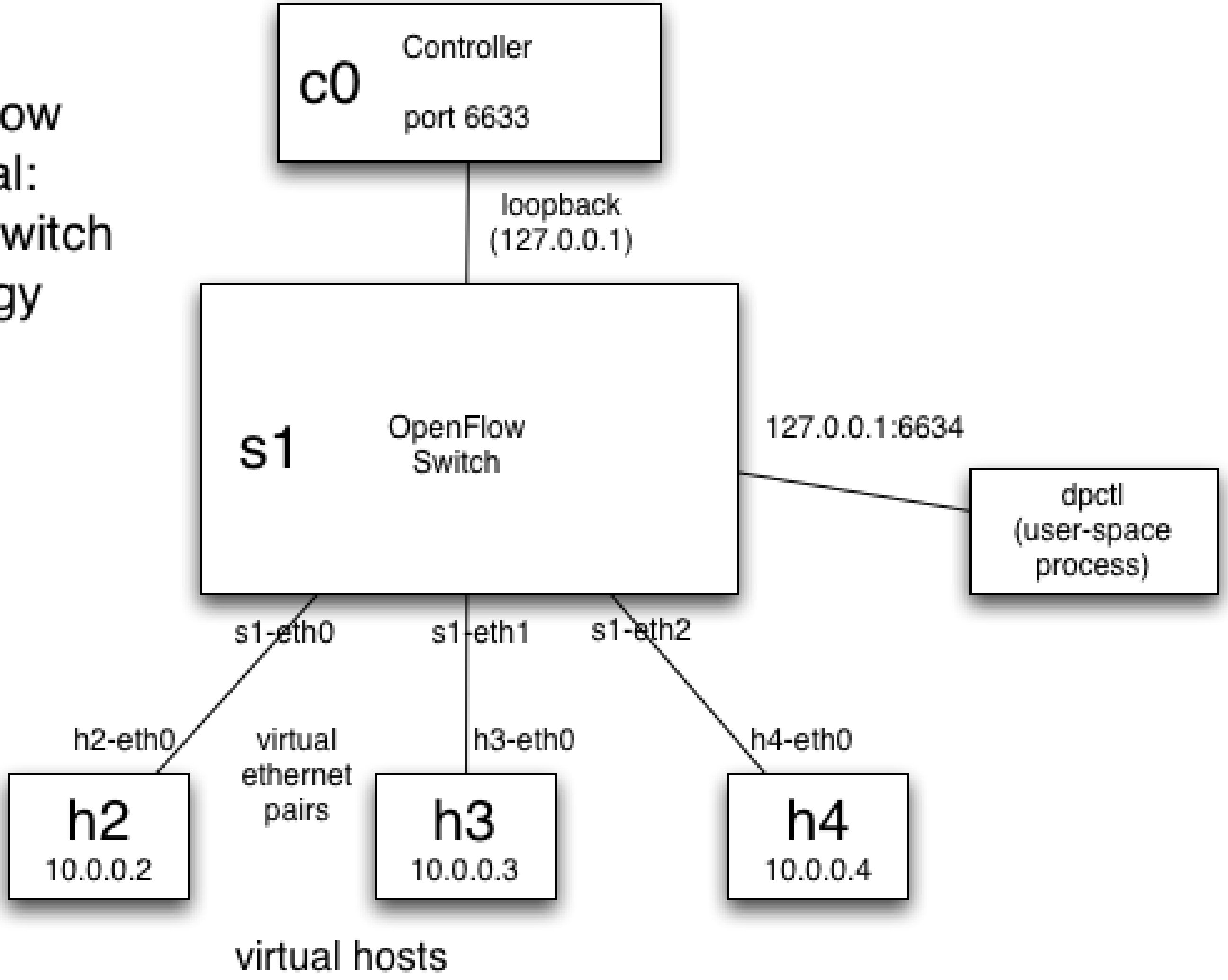
OpenFlow – OpenFlow Channel

- Different message types available:
  - *Controller-to-Switch*, *Asynchronous* or *Symmetric*
- Controller-to-Switch:
  - Lets the controller control the switch
  - E.g., *Modify-State* command to manipulate flow tables
- Asynchronous:
  - Switch-to-controller requests (e.g., at table miss)
- Symmetric:
  - May be sent from both ends (e.g., echo command)



OpenFlow - Example

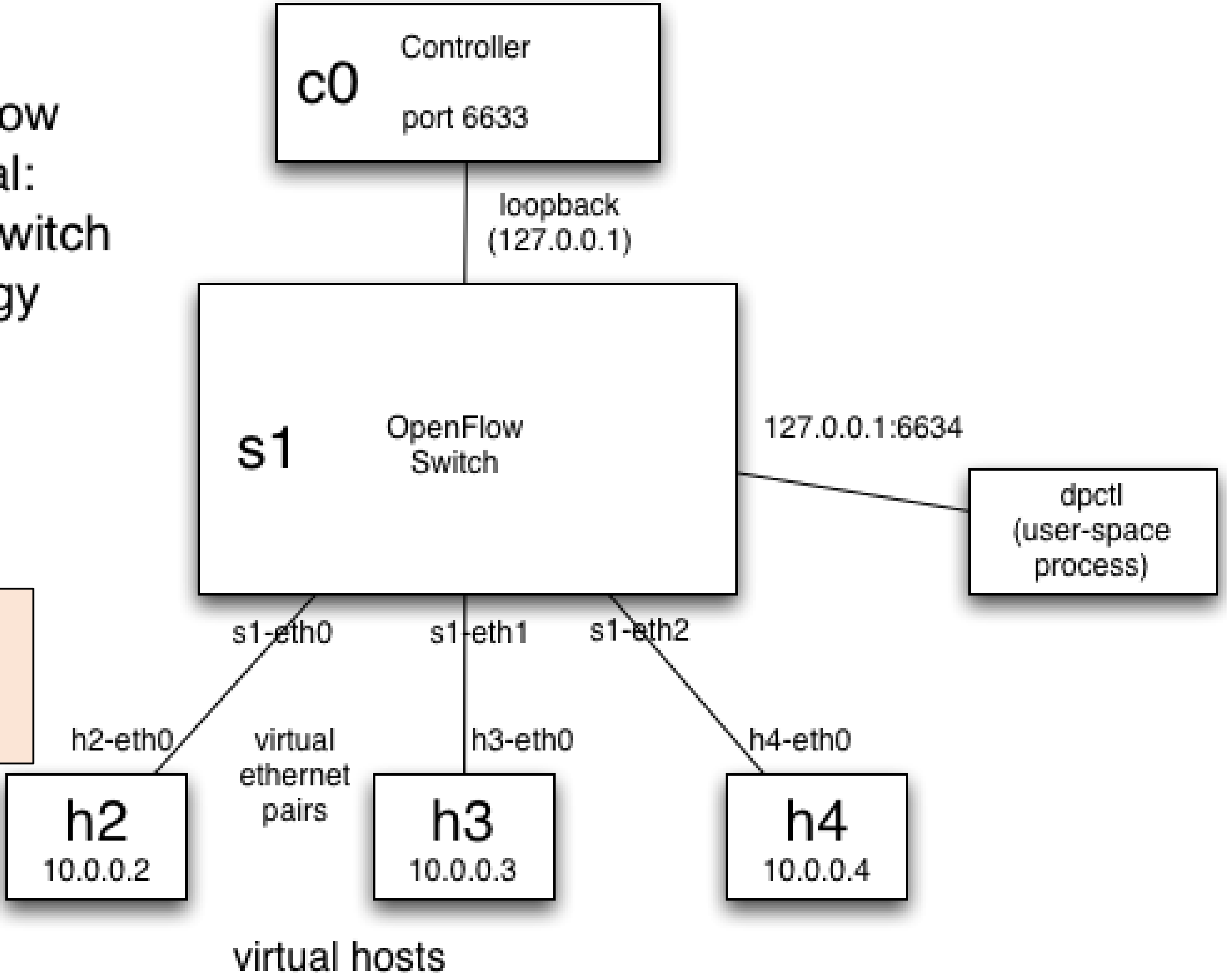
OpenFlow Tutorial: 3hosts-1switch topology



OpenFlow - Example

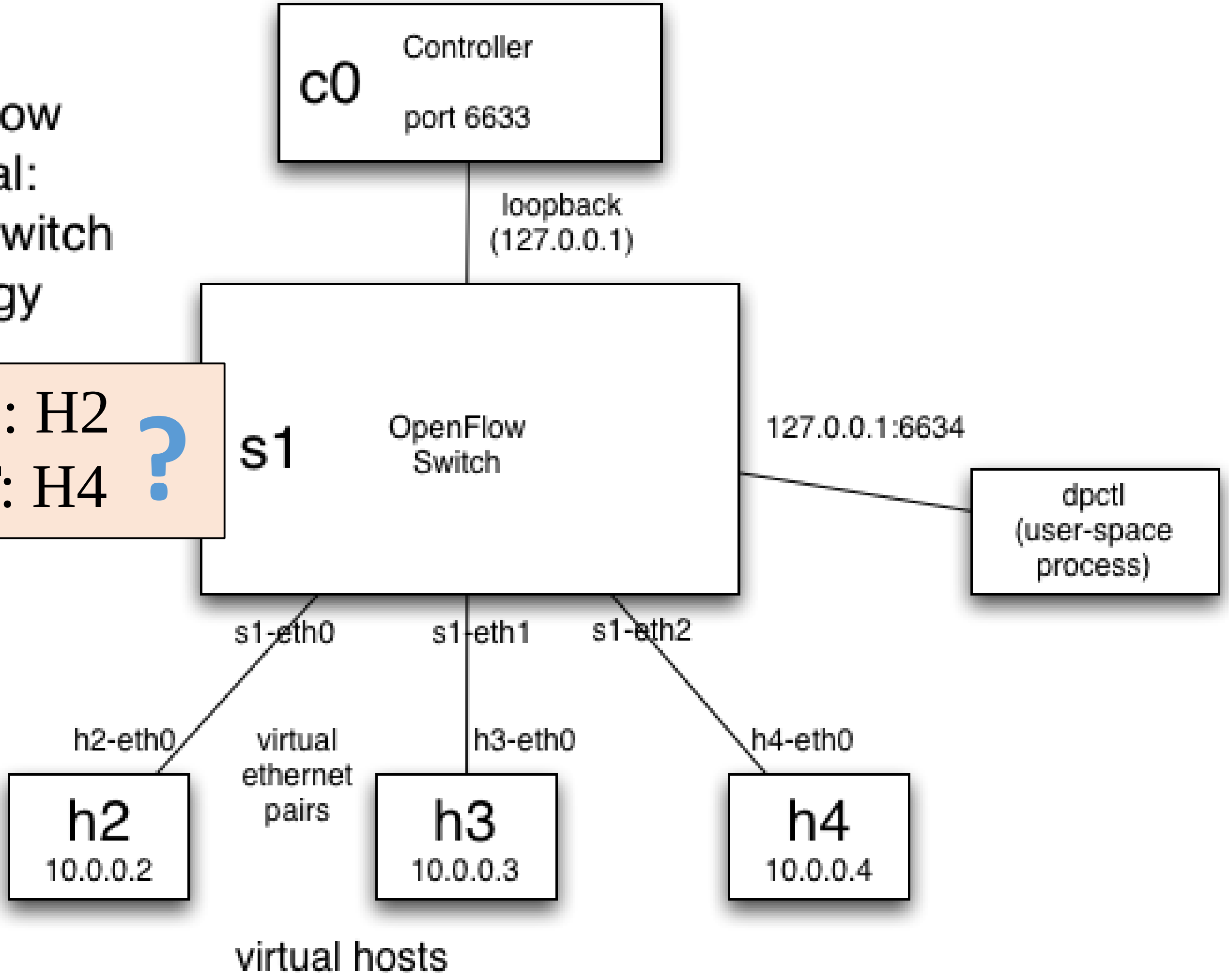
OpenFlow Tutorial: 3hosts-1switch topology

SRC: H2
DST: H4



OpenFlow - Example

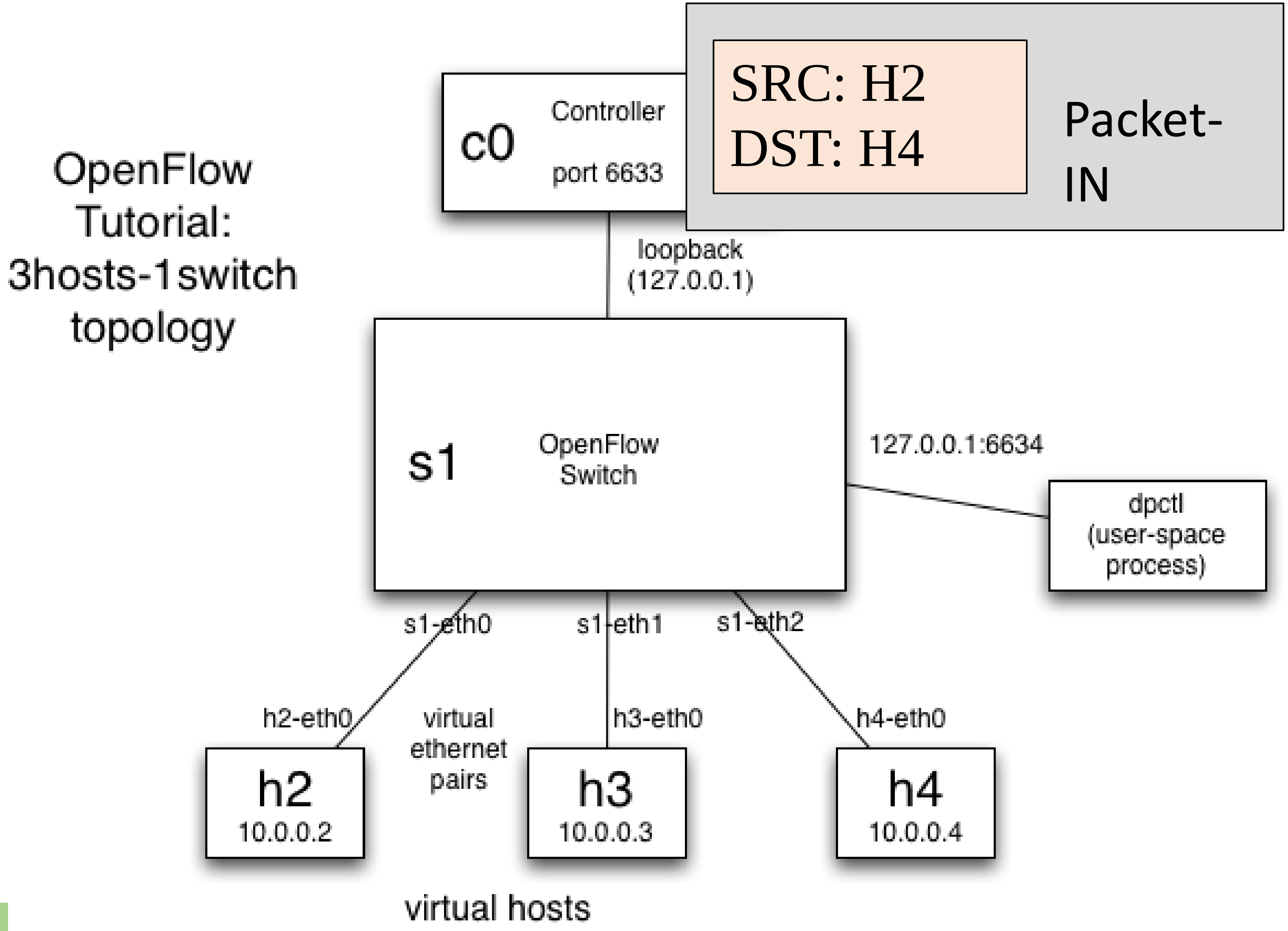
OpenFlow Tutorial: 3hosts-1switch topology



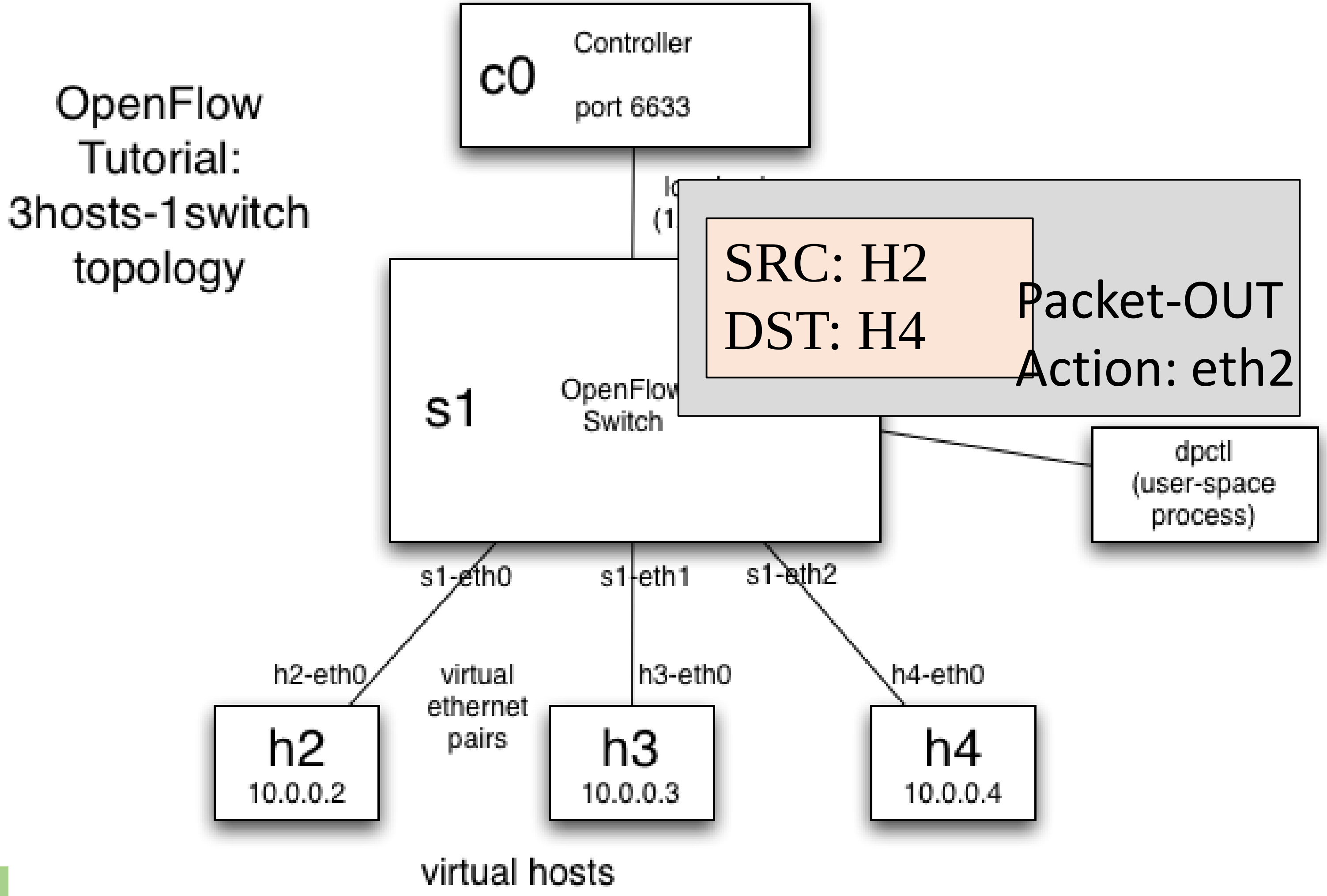
SRC: H2

DST: H4 ?

OpenFlow - Example



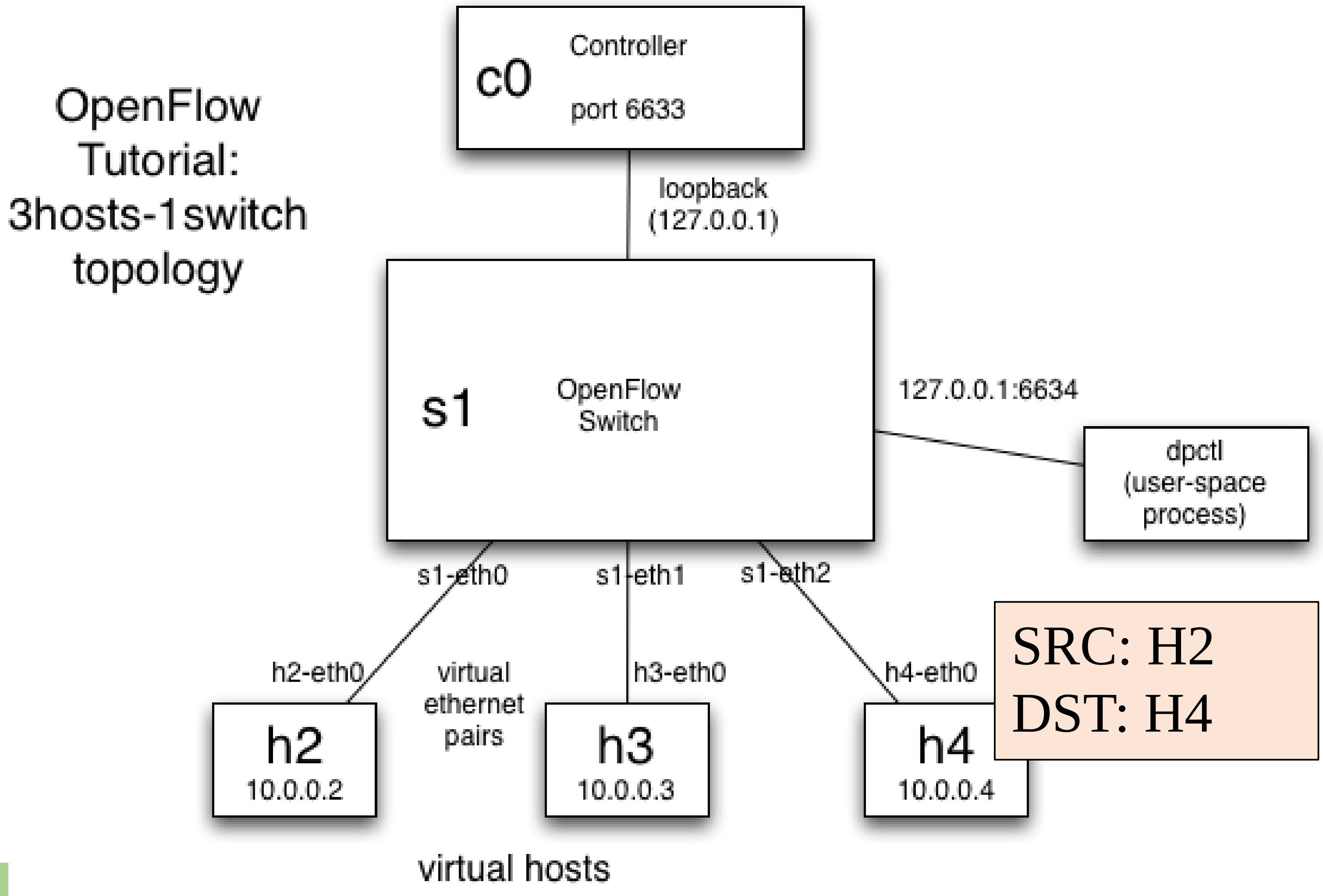
OpenFlow - Example



SRC: H2
DST: H4

Packet-OUT
Action: eth2

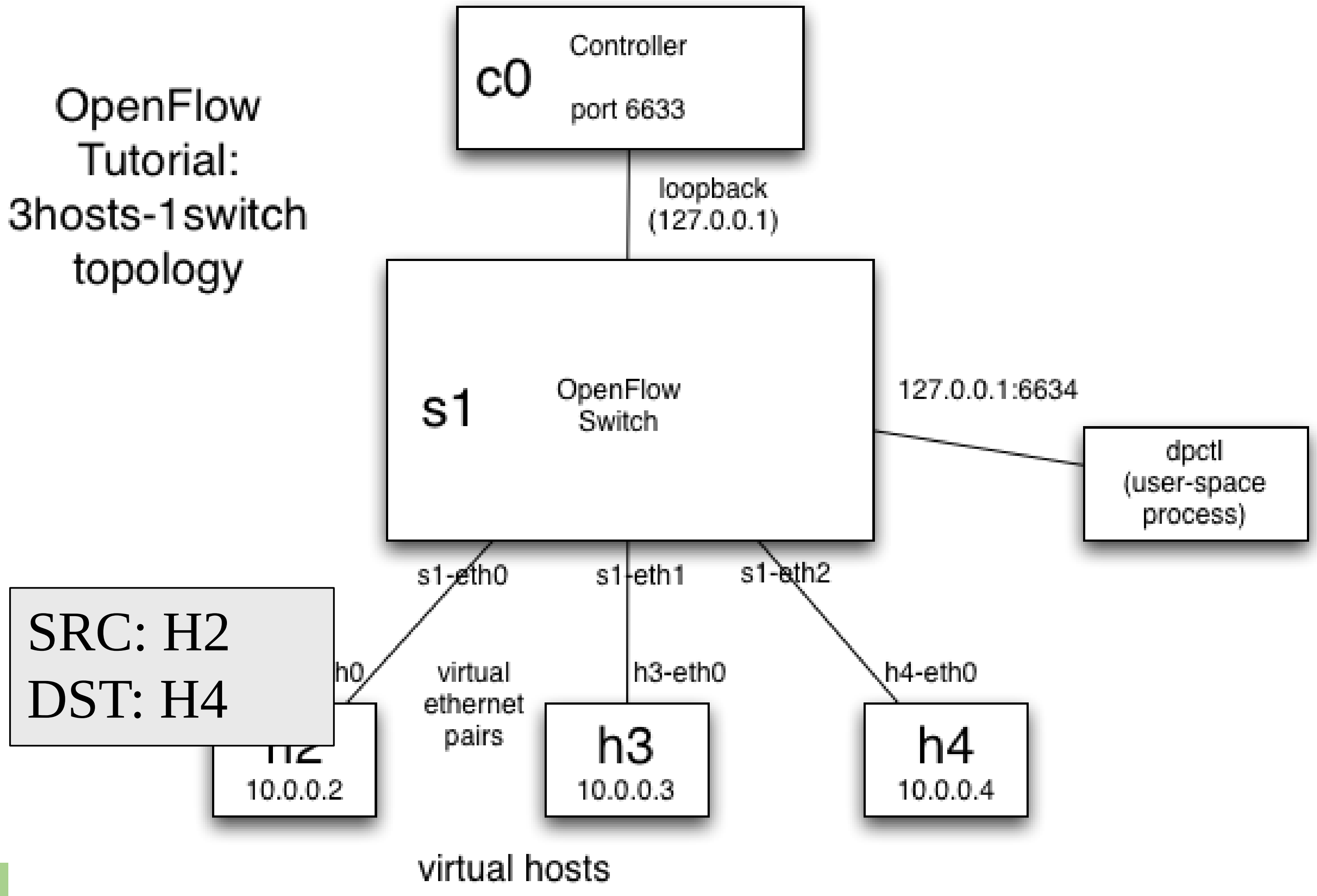
OpenFlow - Example



SRC: H2

DST: H4

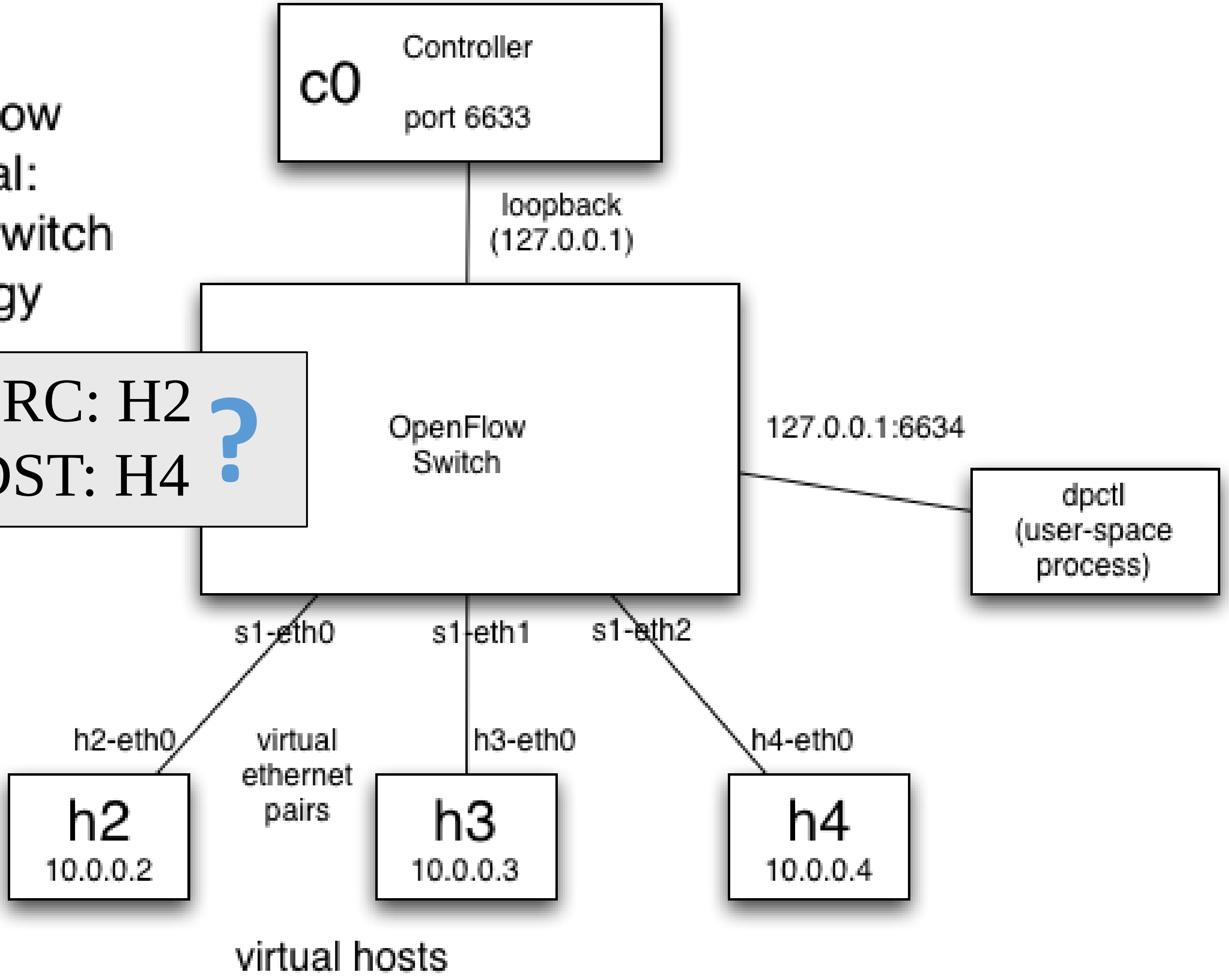
OpenFlow - Example



SRC: H2
DST: H4

OpenFlow - Example

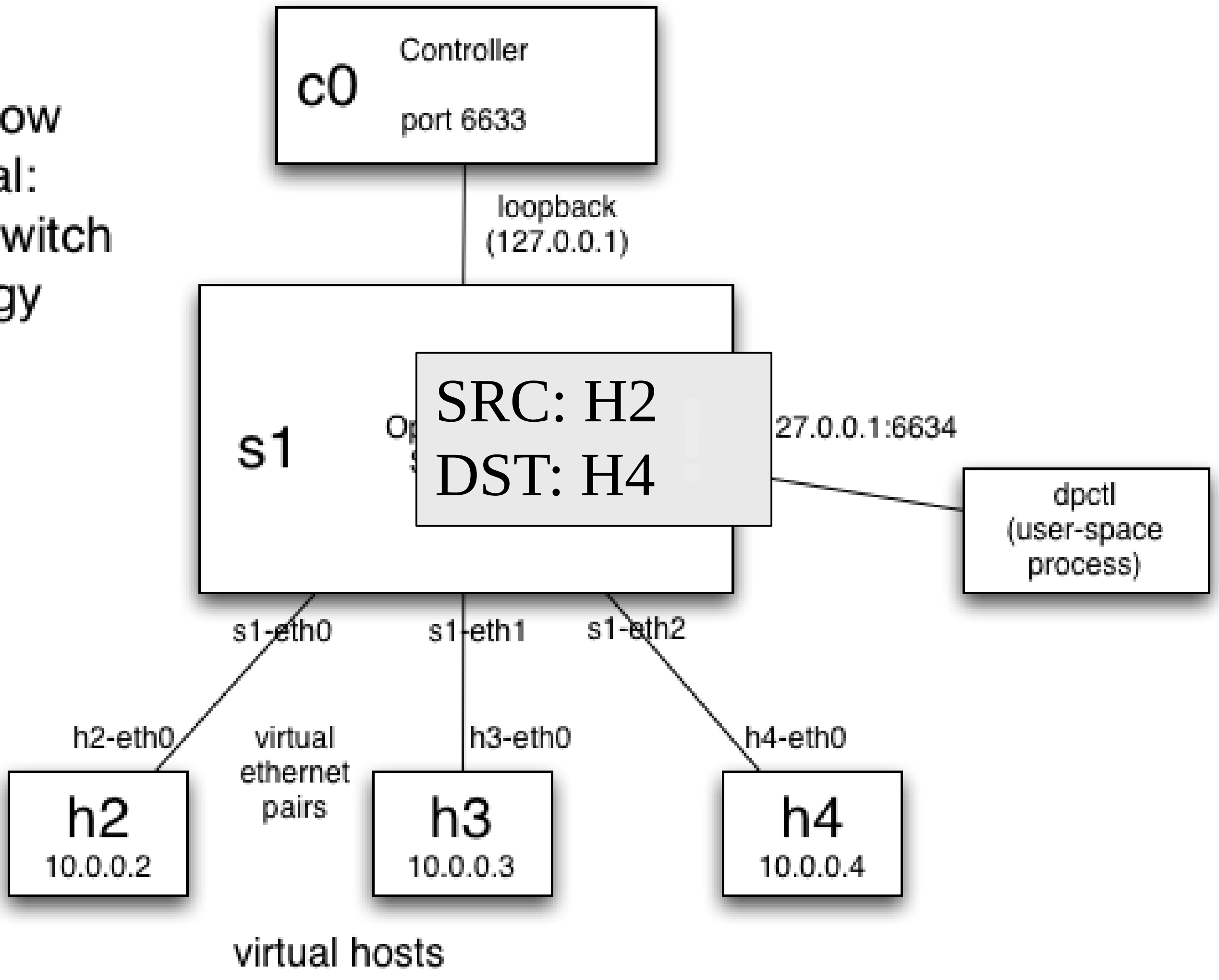
OpenFlow Tutorial: 3hosts-1switch topology



SRC: H2
DST: H4 ?

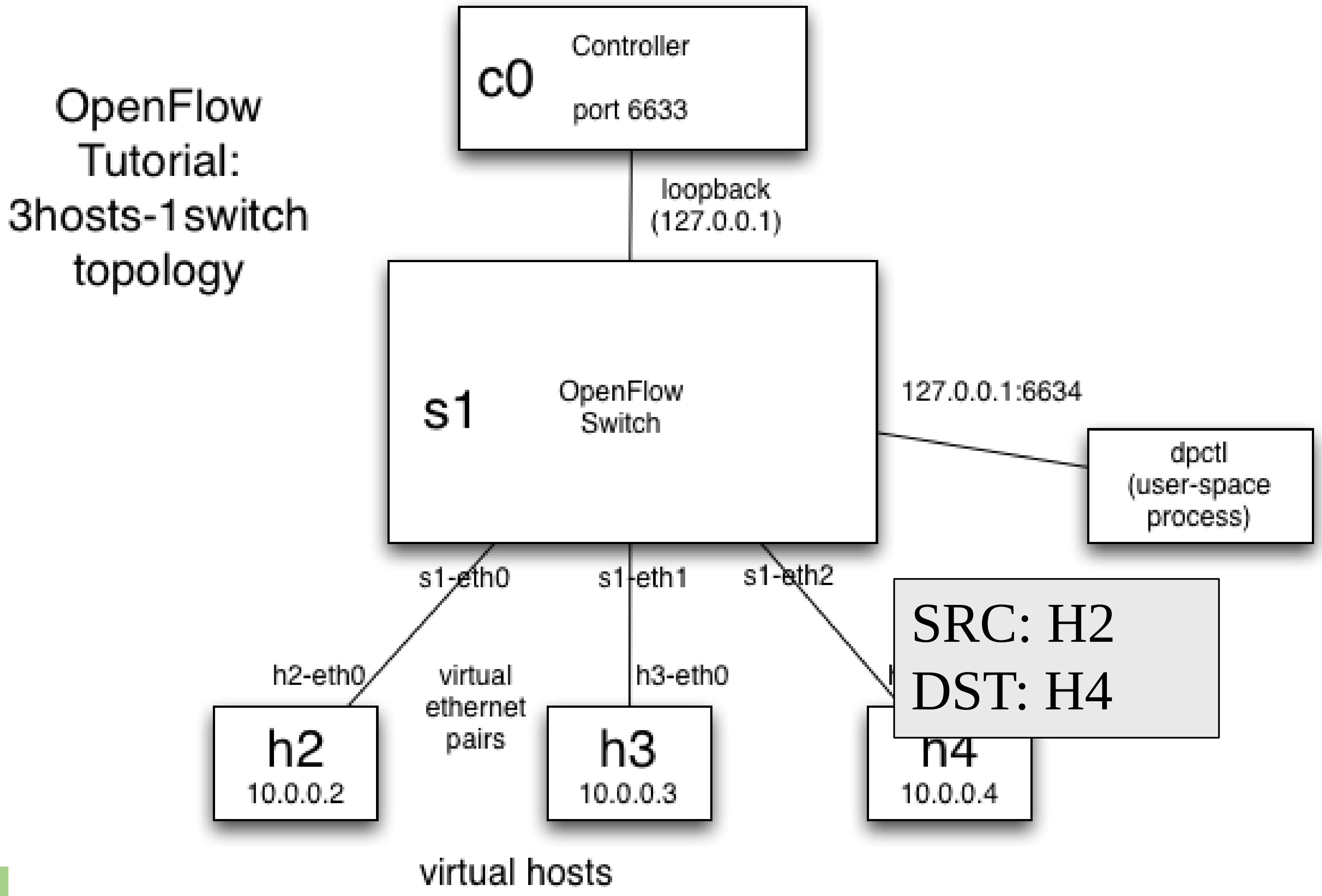
OpenFlow - Example

OpenFlow Tutorial: 3hosts-1switch topology



SRC: H2
DST: H4

OpenFlow - Example

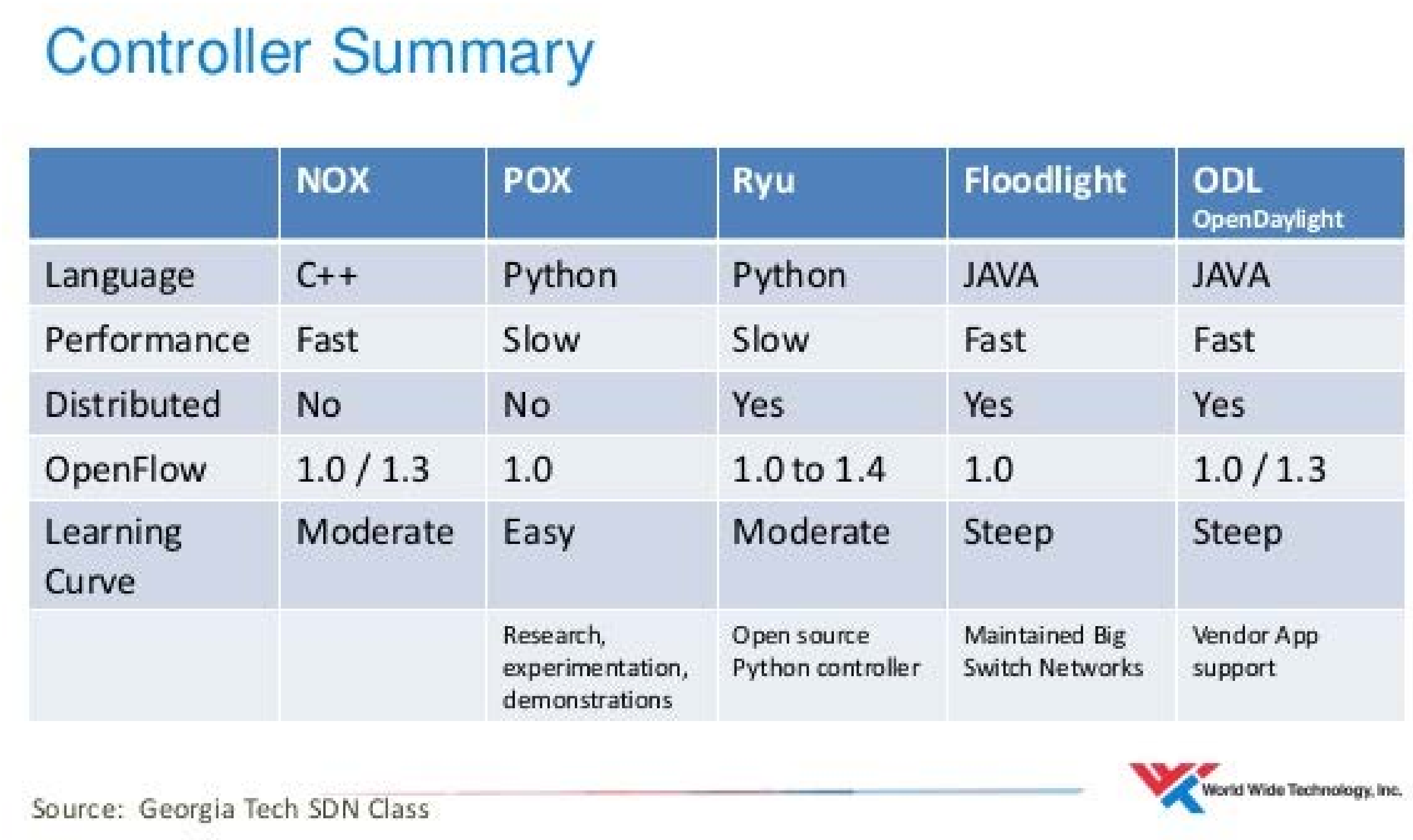


SRC: H2

DST: H4

OpenFlow Controllers

OpenFlow Controllers



Controller Summary

	NOX	POX	Ryu	Floodlight	ODL OpenDaylight
Language	C++	Python	Python	JAVA	JAVA
Performance	Fast	Slow	Slow	Fast	Fast
Distributed	No	No	Yes	Yes	Yes
OpenFlow	1.0 / 1.3	1.0	1.0 to 1.4	1.0	1.0 / 1.3
Learning Curve	Moderate	Easy	Moderate	Steep	Steep
		Research, experimentation, demonstrations	Open source Python controller	Maintained Big Switch Networks	Vendor App support

Source: Georgia Tech SDN Class

World Wide Technology, Inc.

…and many more: Beacon, Trema, OpenContrail, POF, etc.

That‘s a Lot of Controllers!?

„There are almost as many controllers for SDNs as there are SDNs“ – Nick Feamster

Which controller should I use for what problem?

Which controller?

Concept?

Architecture?

Programming language and model?

Advantages / Disadvantages?

Learning Curve?

Developing Community?

Type of target network?

NOX [1]

- **The first controller**
  - Open source
  - Stable

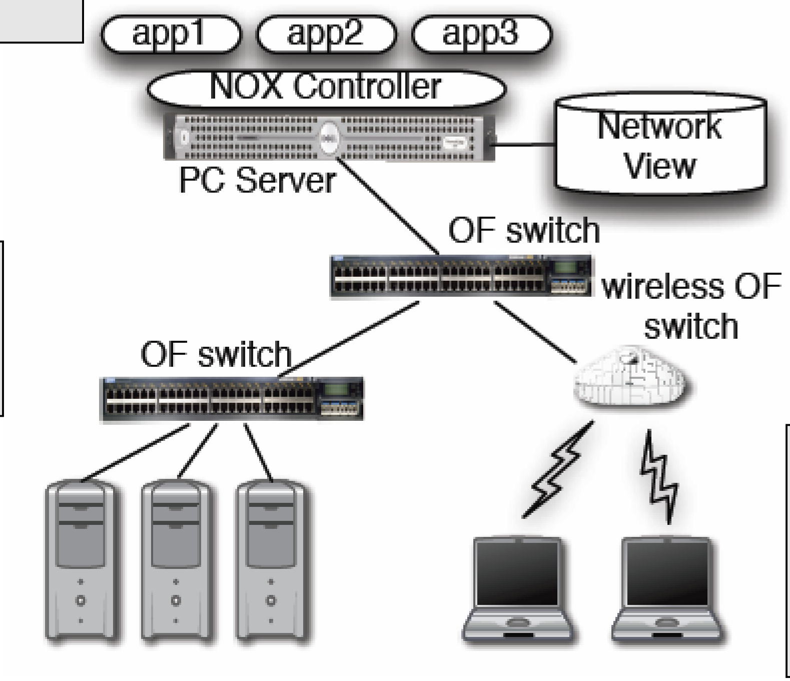
    - No longer supported

    - „New“ NOX: C++ only
      - OF version supported: 1.0

[1] Gude et al. "NOX: towards an operating system for networks." *ACM SIGCOMM CCR* 38.3 (2008): 105-110.

NOX Architecture

Granularity of Control: Per Flow



Controller maintains a network view

switches and attached servers

OpenFlow is used to control switches

[1] Gude et al. "NOX: towards an operating system for networks." *ACM SIGCOMM CCR* 38.3 (2008): 105-110.

NOX Architecture

Programming model: Controller listens for OF events

Programmer writes action handlers for events

When to use NOX

- Need to use low-level semantics of OpenFlow
  - NOX does not come with many abstractions

- Need of good performance (C++)
  - E.g.: production networks

POX [1]

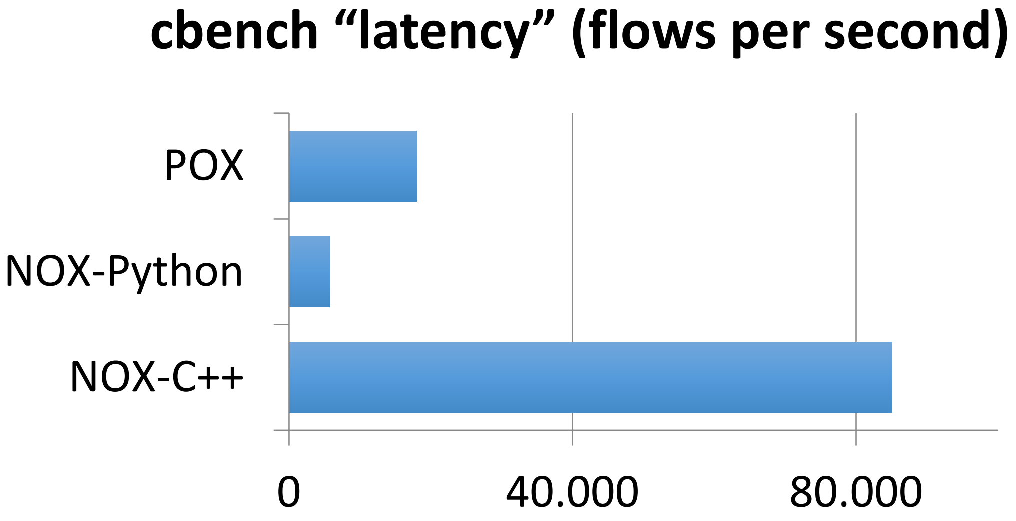
- **POX = NOX in Python**

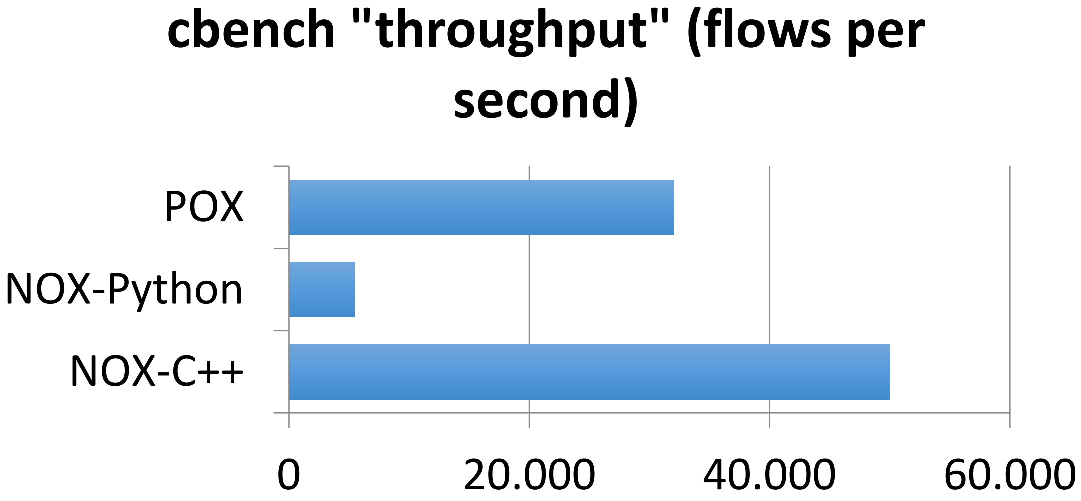
- Advantages:
  - Widely used, maintained and supported
  - Relatively easy to write code for

- Disadvantage:
  - Performance (Python is slower than C++)
  - But: can feed POX ideas back to NOX for production use

[1] Mccauley, J. "Pox: A python-based openflow controller.“ http://www.noxrepo.org/pox/about-pox/

POX





http://www.noxrepo.org/pox/about-pox/

When to use POX

- Learning, testing, debugging, evaluation

In this class :)

- Probably not in large production networks

Just one more: Floodlight [1]

- Java
- Advantages:
  - Documentation,
  - REST API conformity
  - Production-level performance
- Disadvantage:
  - Steep learning curve

[1] http://www.projectfloodlight.org/floodlight/

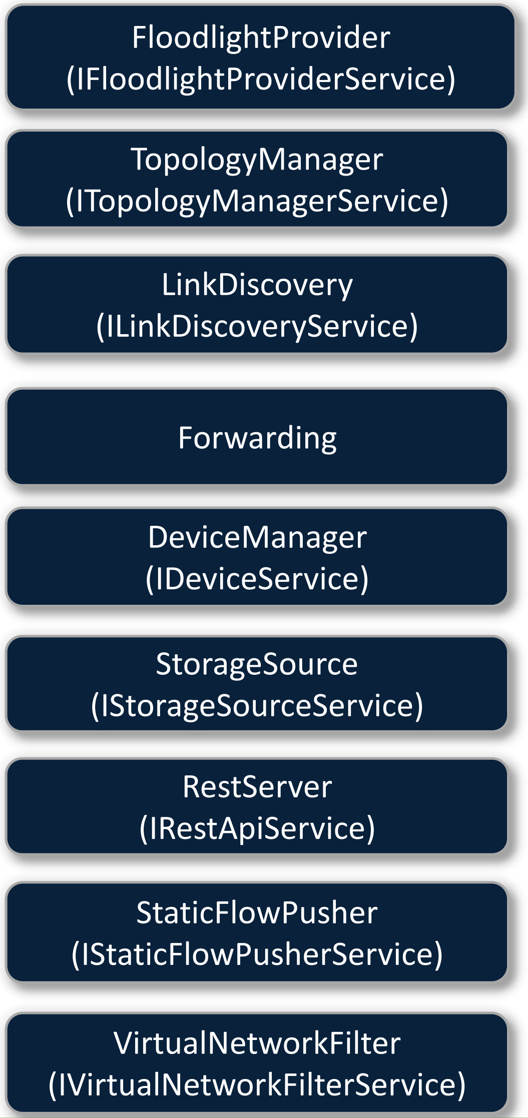
Floodlight: Users

intel
Leland Stanford Junior University 1891
ORACLE
Goldman Sachs
Mellanox Technologies
boundary
Georgia Tech
IBM
FUJITSU
ARMOUR
DELL
BROCADE
Elbrys Networks
hp
ARISTA
PLURIBUS NETWORKS
NC State
Microsoft
CISCO
extreme networks
NEC
LUXOFT Engineering Business Performance
Princeton University
CITRIX
JUNIPER NETWORKS
CLEMSON UNIVERSITY
INDIANA UNIVERSITY
UNIVERSITÄT PASSAU
THALES
UNIVERSITY OF KENTUCKY
MARIST

Floodlight Adopters:

- University research
- Networking vendors
- Users
- Developers / startups

Floodlight Overview



FloodlightProvider (IFloodlightProviderService)

TopologyManager (ITopologyManagerService)

LinkDiscovery (ILinkDiscoveryService)

Forwarding

DeviceManager (IDeviceService)

StorageSource (IStorageSourceService)

RestServer (IRestApiService)

StaticFlowPusher (IStaticFlowPusherService)

VirtualNetworkFilter (IVirtualNetworkFilterService)

- Floodlight is a collection of modules
- Some modules (not all) export services
- All modules in Java
- Rich, extensible REST API

Taken from: Cohen et al, “Software-Defined Networking and the Floodlight Controller”, available at http://de.slideshare.net/openflowhub/floodlight-overview-13938216

Floodlight Overview

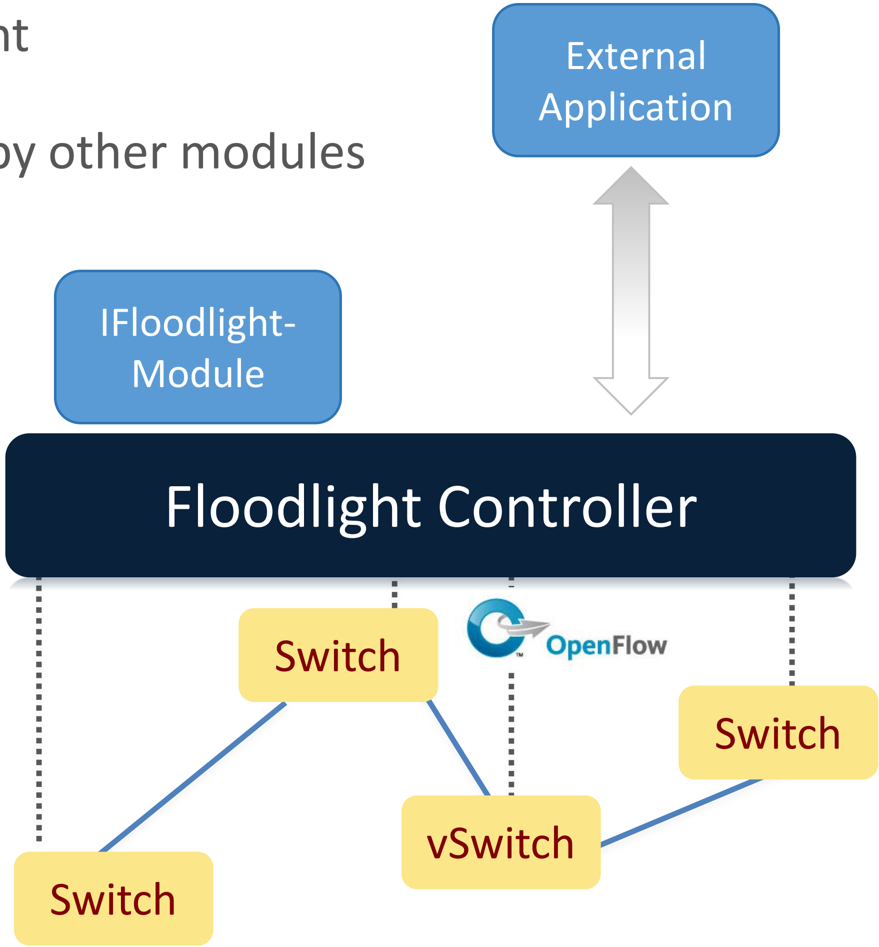
FloodlightProvider (IFloodlightProviderService)	• Translates OF messages to Floodlight events • Managing connections to switches via Netty
TopologyManager (ITopologyManagerService)	• Computes shortest path using Dijsktra • Keeps switch to cluster mappings
LinkDiscovery (ILinkDiscoveryService)	• Maintains state of links in network • Sends out LLDPs
Forwarding	• Installs flow mods for end-to-end routing • Handles island routing
DeviceManager (IDeviceService)	• Tracks hosts on the network • MAC -> switch,port, MAC->IP, IP->MAC
StorageSource (IStorageSourceService)	
RestServer (IRestApiService)	• Implements via Restlets (restlet.org) • Modules export RestletRoutable
StaticFlowPusher (IStaticFlowPusherService)	• Supports the insertion and removal of static flows • REST-based API
VirtualNetworkFilter (IVirtualNetworkFilterService)	• Create layer 2 domain defined by MAC address

Taken from: Cohen et al, “Software-Defined Networking and the Floodlight Controller”, available at http://de.slideshare.net/openflowhub/floodlight-overview-13938216

Floodlight Programming Model

IFloodlightModule

- Java module that runs as part of Floodlight
- Consumes services and events exported by other modules
  - OpenFlow (ie. Packet-in)
  - Switch add / remove
  - Device add /remove / move
  - Link discovery



External Application

- Communicates with Floodlight via REST

Taken from: Cohen et al, “Software-Defined Networking and the Floodlight Controller”, available at http://de.slideshare.net/openflowhub/floodlight-overview-13938216

Floodlight Modules

Network State

List Hosts

List Links

List Switches

GetStats (DPID)

GetCounters (OFType…)

Static Flows

Add Flow

Delete Flow

List Flows

RemoveAll Flows

Virtual Network

Create Network

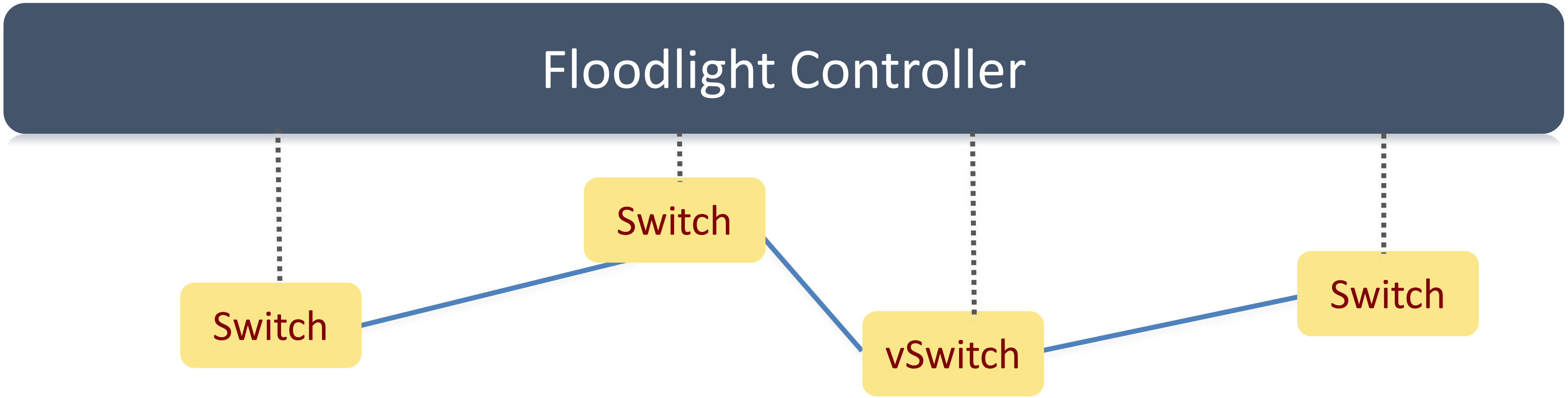
Delete Network

Add Host

Remove Host

User Extensions

…



Taken from: Cohen et al, “Software-Defined Networking and the Floodlight Controller”, available at http://de.slideshare.net/openflowhub/floodlight-overview-13938216

When to use Floodlight

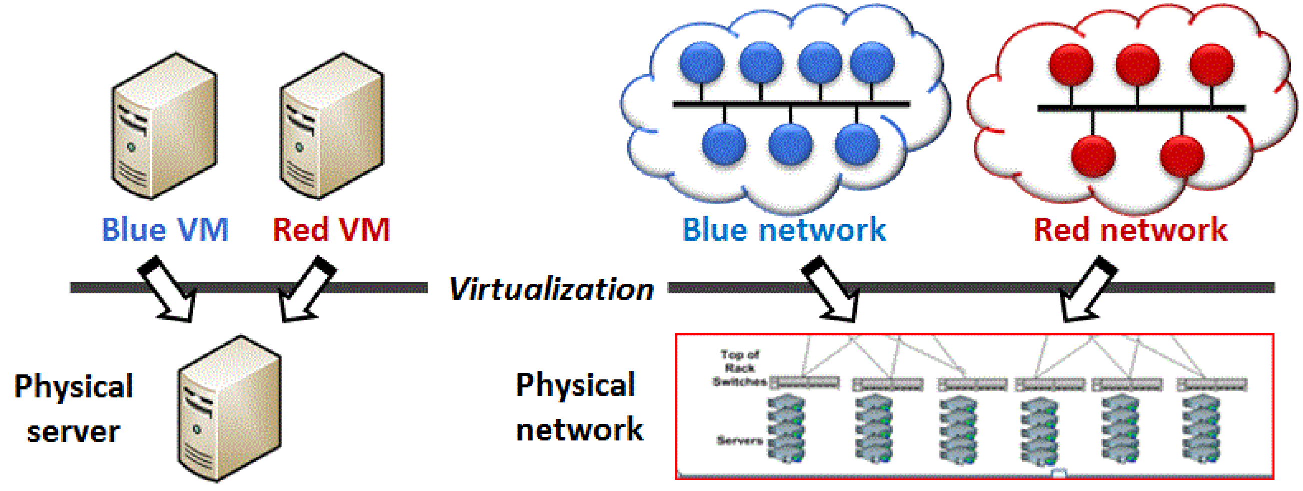
- If you know JAVA
- If you need production-level performance
- Have/want to use REST API

Network Virtualization with OpenFlow

Virtualizing OpenFlow

- Network operators “Delegate” control of subsets of network hardware and/or traffic to other network operators or users
- Multiple controllers can talk to the same set of switches
- Imagine a hypervisor for network equipments
- Allow experiments to be run on the network in isolation of each other and production traffic

Virtualizing OpenFlow



Server virtualization

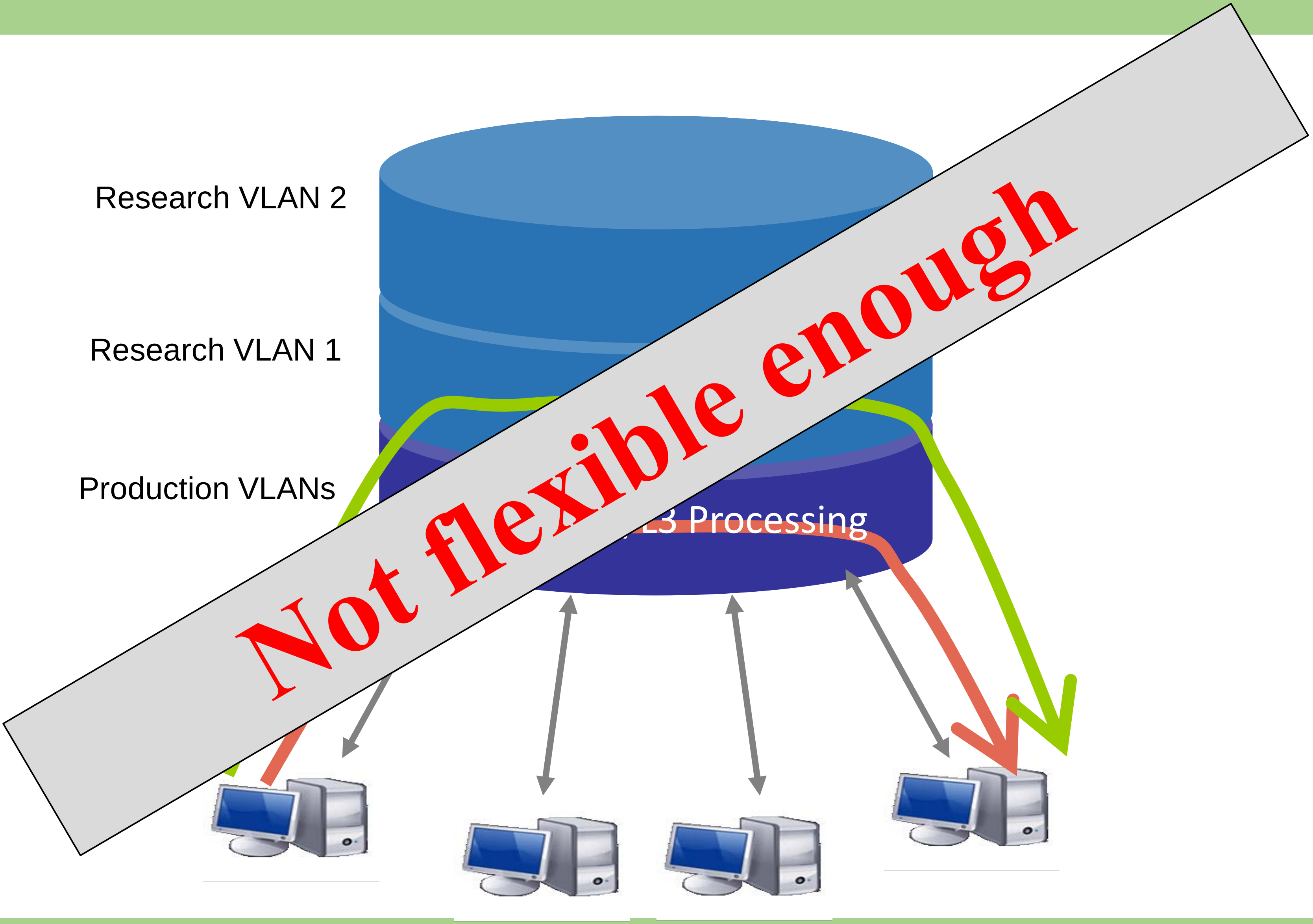
- Run multiple virtual servers on a physical server
- Each VM has illusion it is running as a physical server

Network virtualization

- Run multiple virtual networks on a physical network
- Each virtual network has illusion it is running as a physical network

https://gallery.technet.microsoft.com/scriptcenter/Simple-Hyper-V-Network-d3efb3b8

Virtualization: VLANs



Research VLAN 2

Research VLAN 1

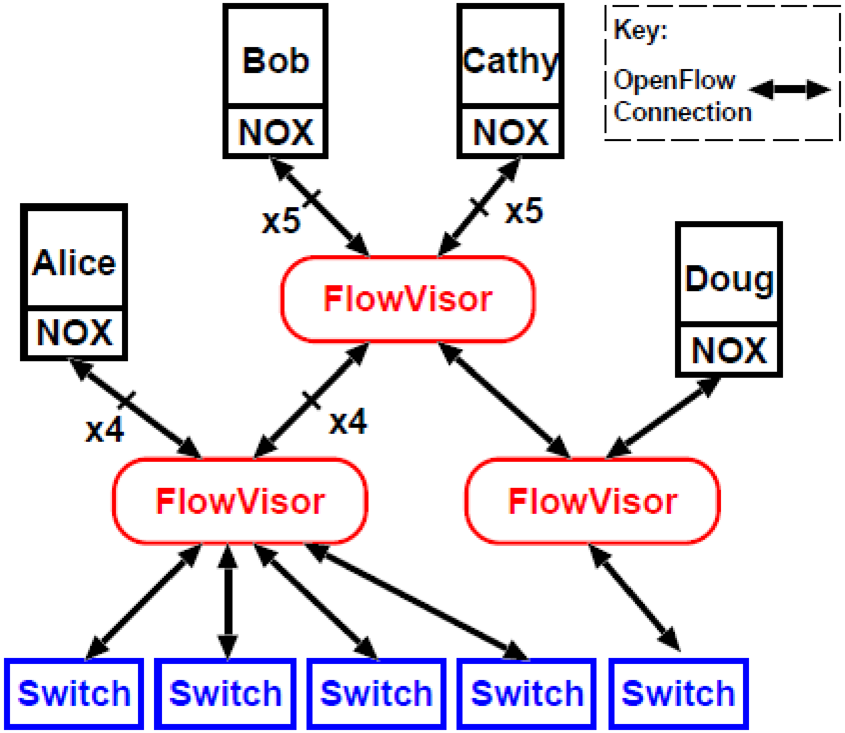
Production VLANs

L3 Processing

Not flexible enough

FlowVisor [1]

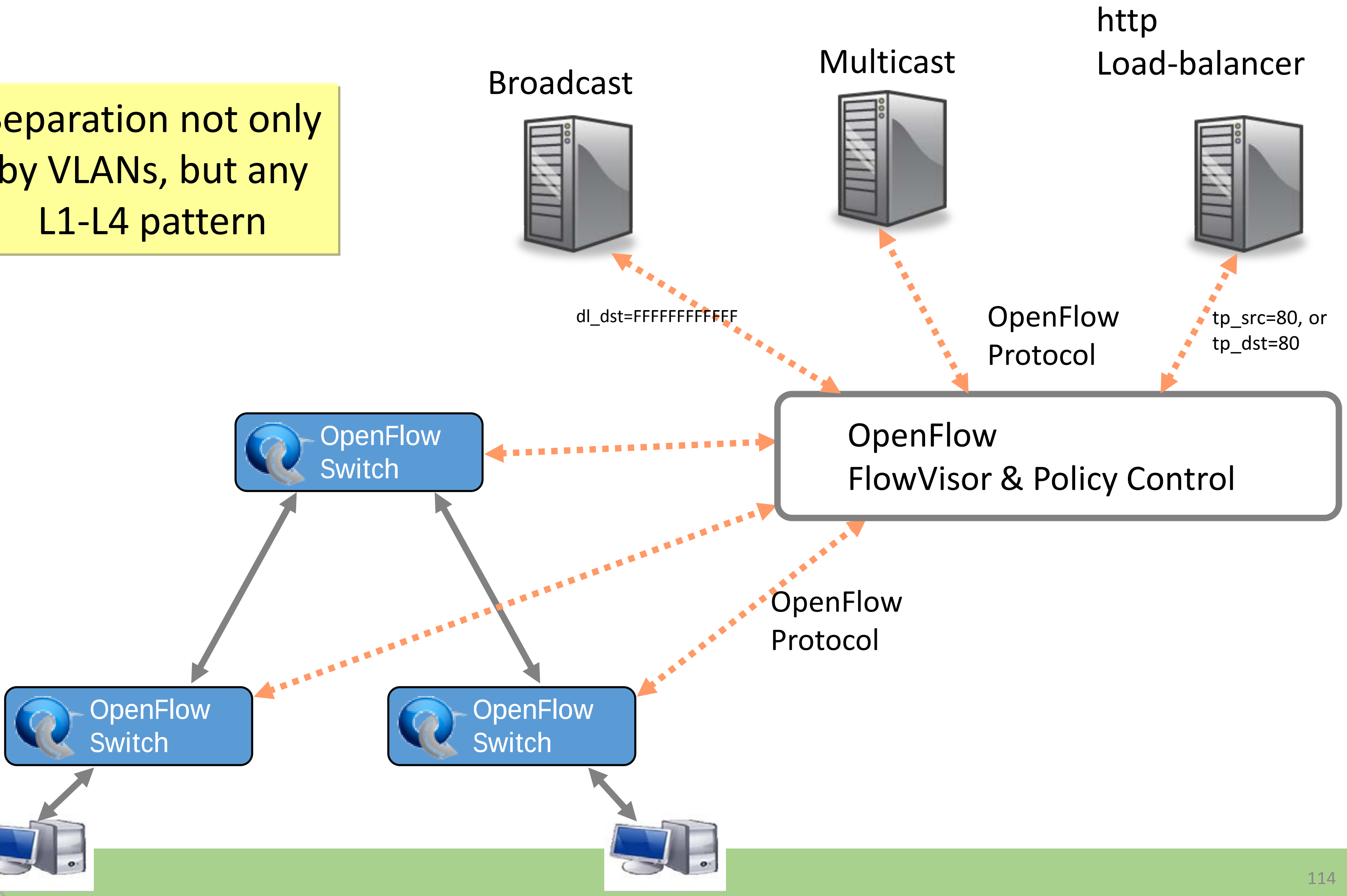
- A network hypervisor developed by Stanford
- A software proxy between the forwarding and control planes of network devices



[1] Sherwood, et al. *"Flowvisor: A network virtualization layer." OpenFlow Switch Consortium, Tech. Rep* (2009).

FlowVisor-based Virtualization

Separation not only by VLANs, but any L1-L4 pattern



Slicing Policies

- The policy specifies resource limits for each slice:
  - Link bandwidth
  - Maximum number of forwarding rules
  - Topology
  - Fraction of switch/router CPU
  - *FlowSpace: which packets does the slice control?*

FlowVisor Resource Limits

- FV assigns hardware resources to “Slices”
  - Topology
    - Network Device or Openflow Instance (DPID)
    - Physical Ports
  - Bandwidth
    - Each slice can be assigned a per port queue with a fraction of the total bandwidth

FlowVisor Resource Limits (cont.)

- FV assigns hardware resources to “Slices”
  - CPU
    - Employs Course Rate Limiting techniques to keep new flow events from one slice from overrunning the CPU
  - Forwarding Tables
    - Each slice has a finite quota of forwarding rules per device

FlowVisor FlowSpace

- FlowSpace is defined by a collection of packet headers and assigned to “Slices”
  - Source/Destination MAC address
  - VLAN ID
  - Ethertype
  - IP protocol
  - Source/Destination IP address
  - ToS/DSCP
  - Source/Destination port number

Use Case: VLAN Partitioning

- Basic Idea: Partition Flows based on Ports and VLAN Tags
  - Traffic entering system (e.g. from end hosts) is tagged
  - VLAN tags consistent throughout substrate

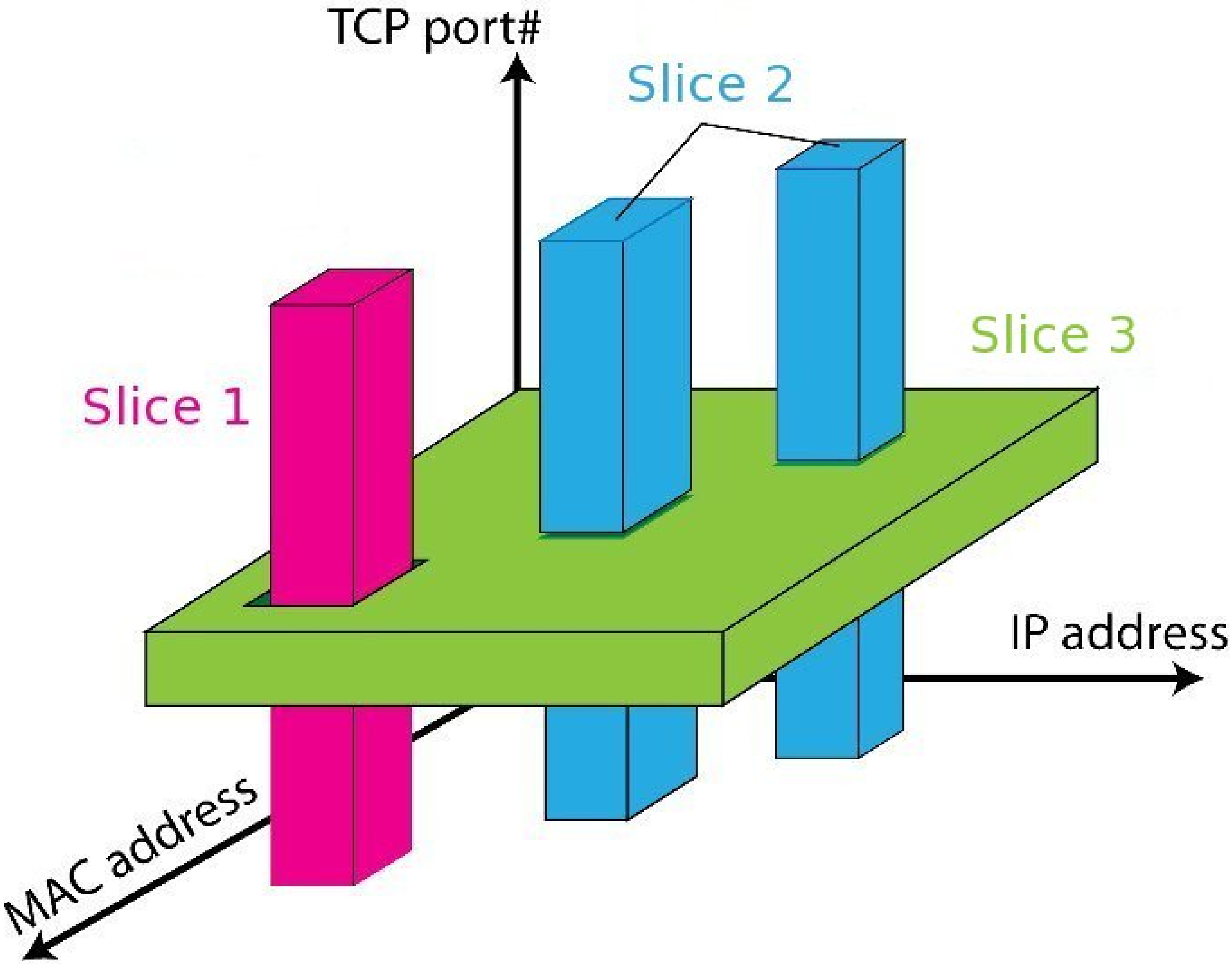
	Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport
Dave	*	*	*	*	1,2,3	*	*	*	*	*
Larry	*	*	*	*	4,5,6	*	*	*	*	*
Steve	*	*	*	*	7,8,9	*	*	*	*	*

Use Case: Content Distribution Network

- Basic Idea: Build a CDN where you control the entire network
  - All traffic to or from CDN IP space controlled by Experimenter
  - All other traffic controlled by default routing
  - Topology is the entire network

	Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport
From CDN	*	*	*	*	*	84.65.*	*	*	*	*
To CDN	*	*	*	*	*	*	84.65.*	*	*	*
Default	*	*	*	*	*	*	*	*	*	*

FlowSpace: Maps Packets to Slices



Taken from: Rob Sherwood's presentation at ONS:
http://www.opennetsummit.org/archives/apr12/sherwood-mon-flowvisor.pdf

FlowVisor Slicing Policy

- FlowVisor intercepts OpenFlow messages from devices
  - Send control plane messages to the slice controller only if source is in slice topology.
  - Rewrite OpenFlow feature negotiation messages so the slice controller only sees the ports in it’s slice
  - Port up/down messages are pruned and only forwarded to affected slices

FlowVisor Slicing Policy

- FlowVisor intercepts OpenFlow messages from controllers
  - Rewrites flow insertion, deletion & modification rules so they don’t violate the slice definition
    - Flow definition – ex. Limit Control to HTTP traffic only
    - Actions – ex. Limit forwarding to only ports in the slice

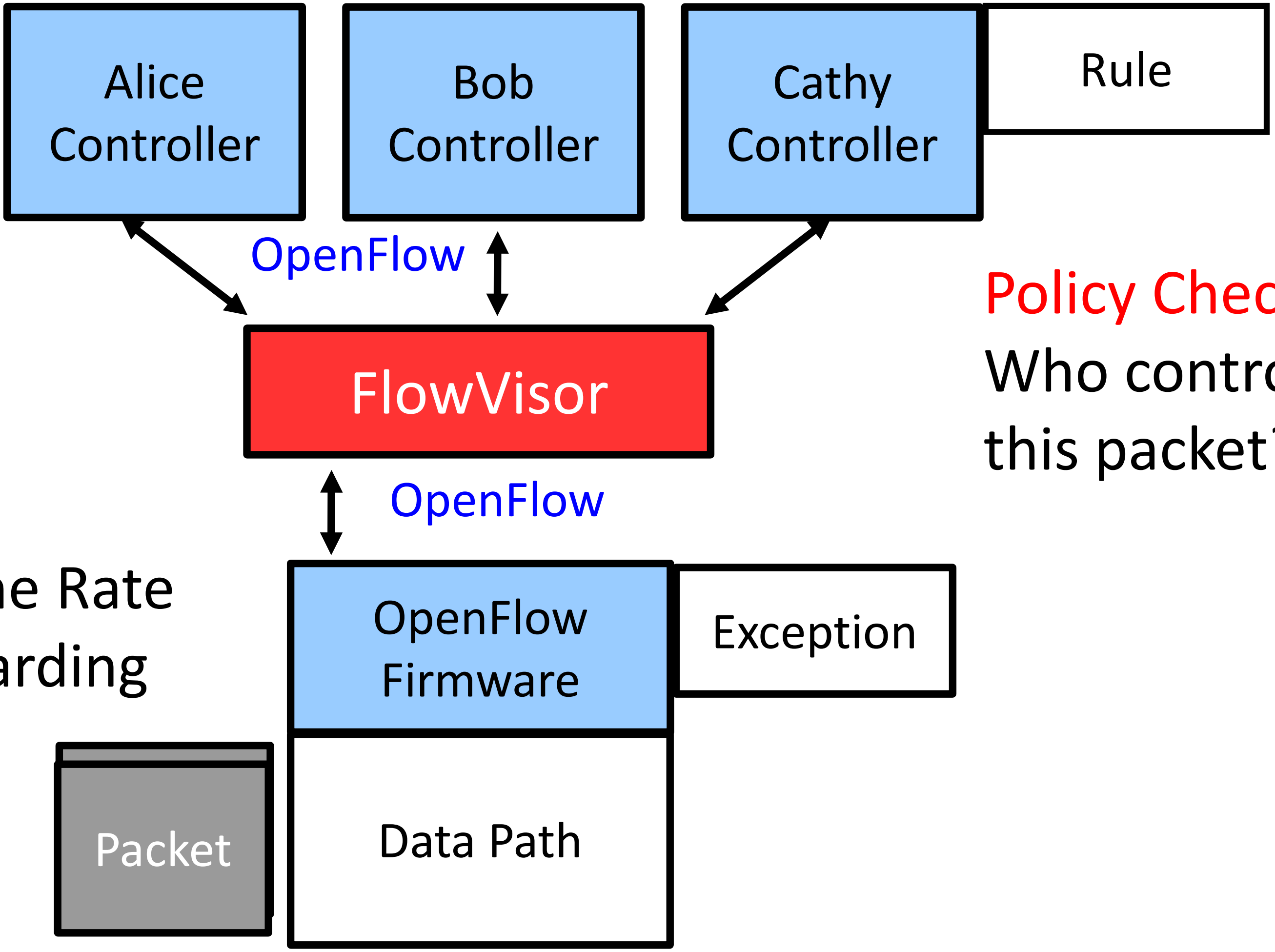
FlowVisor Slicing Policy

- FlowVisor intercepts OpenFlow messages from controllers
  - Expand Flow rules into multiple rules to fit policy
    - Flow definition – ex. If there is a policy for John’s HTTP traffic and another for Uwe’s HTTP traffic, FV would expand a single rule intended to control all HTTP traffic into 2 rules.
    - Actions – ex. Rule action is send out all ports. FV will create one rule for each port in the slice.

    - Returns “action is invalid” error if trying to control a port outside of the

FlowVisor Message Handling

Policy Check:
Is this rule allowed?



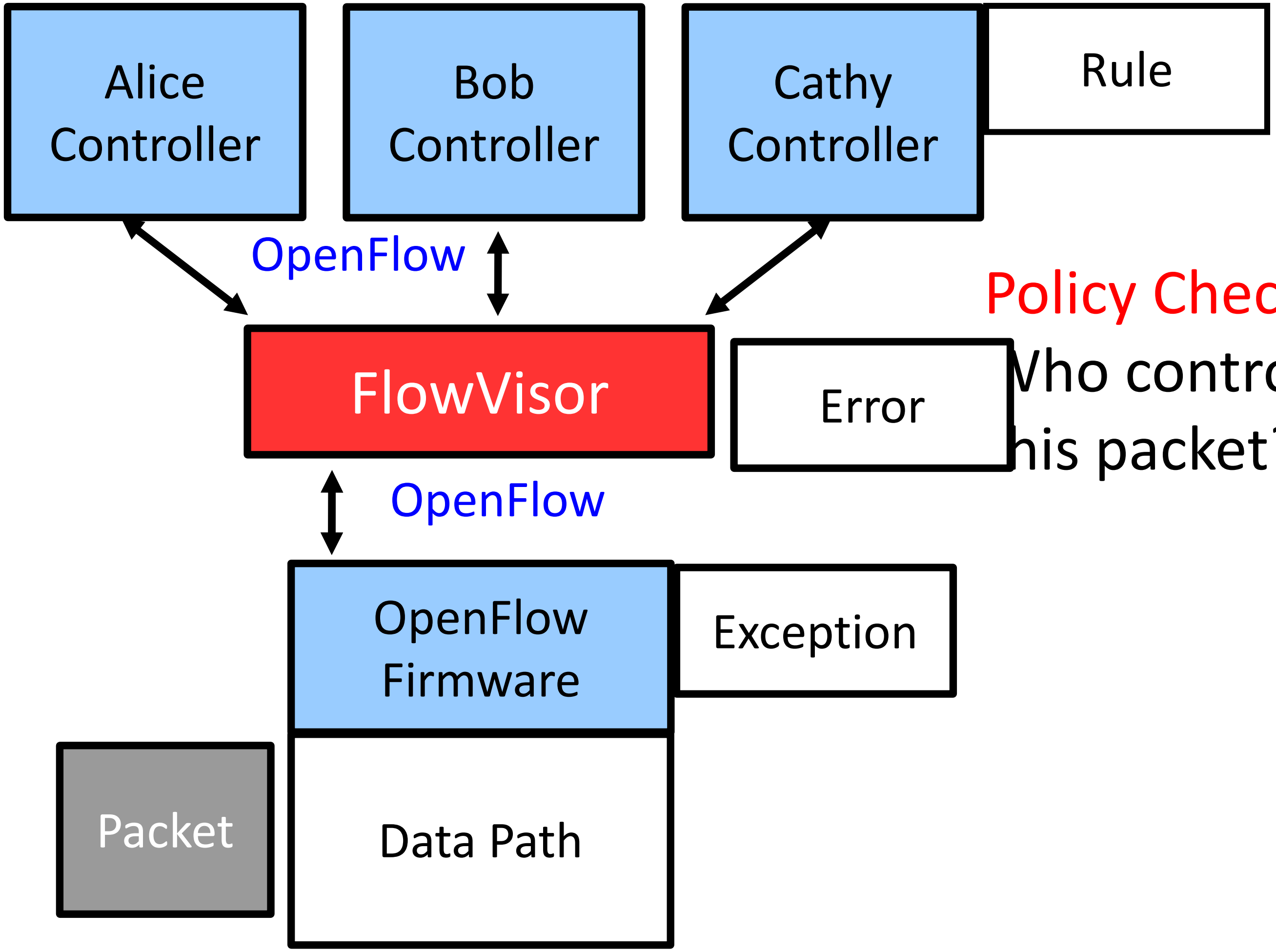
Policy Check:
Who controls this packet?

Full Line Rate Forwarding

Taken from: Rob Sherwood's presentation at ONS:
http://www.opennetsummit.org/archives/apr12/sherwood-mon-flowvisor.pdf

FlowVisor Message Handling

Policy Check:
Is this rule allowed?



Policy Check:
Who controls this packet?

Taken from: Rob Sherwood's presentation at ONS:
http://www.opennetsummit.org/archives/apr12/sherwood-mon-flowvisor.pdf