Software-defined Networking I

Advanced Computer Networks

Summer Semester 2017

The status of networks today

- Today, routers implement a lot of functionality
  - They forward packets (*data plane*)
  - And run the *control plane* software (routing algorithms etc.)

Data Plane? Control Plane?

- Data plane
  - The actual forwarding actions
  - Receiving a packet on an input port, looking up output port, forwarding packet via output port

Data Plane? Control Plane?

- Data plane
  - The actual forwarding actions
  - Receiving a packet on an input port, looking up output port, forwarding packet via output port

- Control plane
  - Defines what the data plane does
  - Installs instructions into data plane devices (e.g., installs forwarding rules)
  - Example: routing protocols, traffic engineering

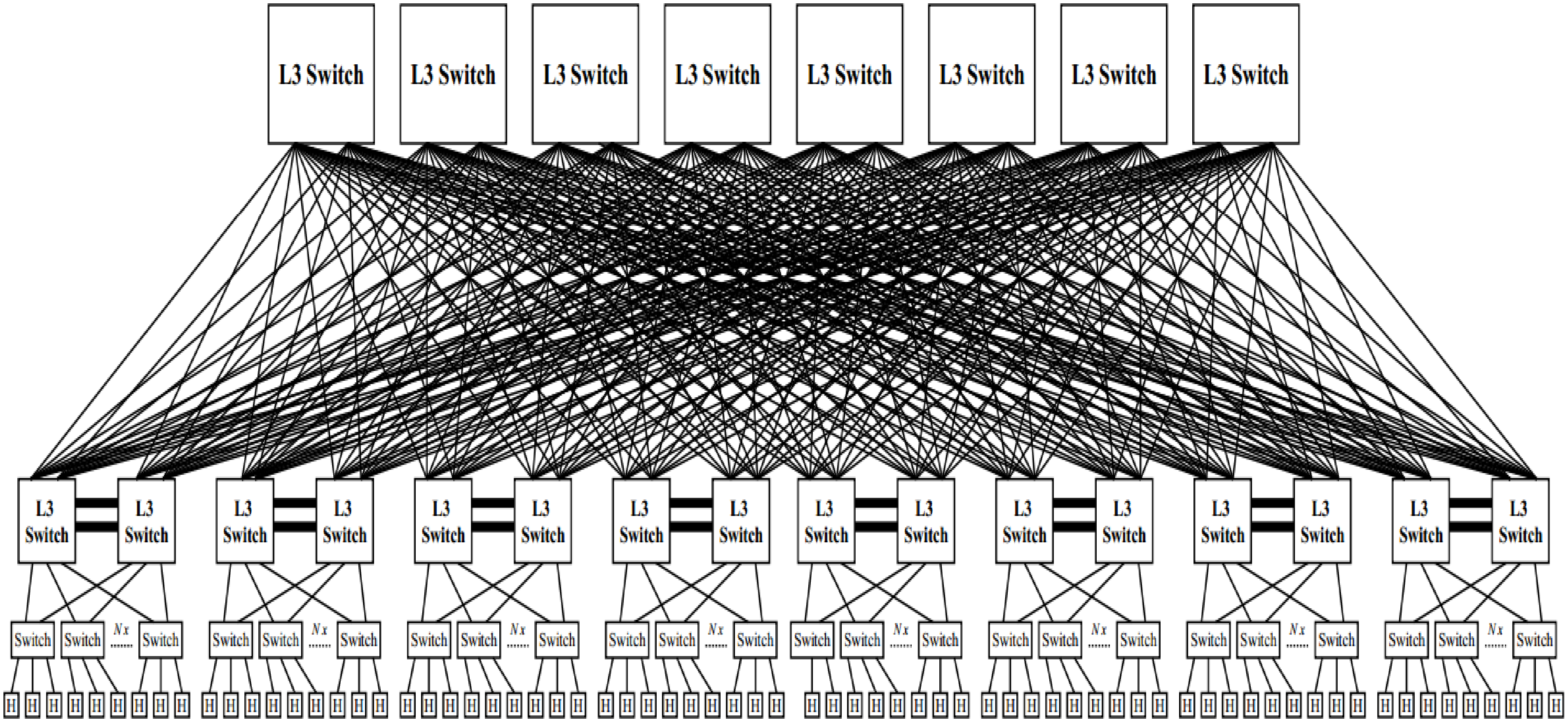
Problems with Networks today

- 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



computer NETWORKS

Problems with Networks today

- 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

Software-defined Networking in one Slide

- SDN networks break up with this concept
    - 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**

computer
N E T
W O R K S

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.

computer NETWORKS

Software-defined Networking (SDN)?

„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*

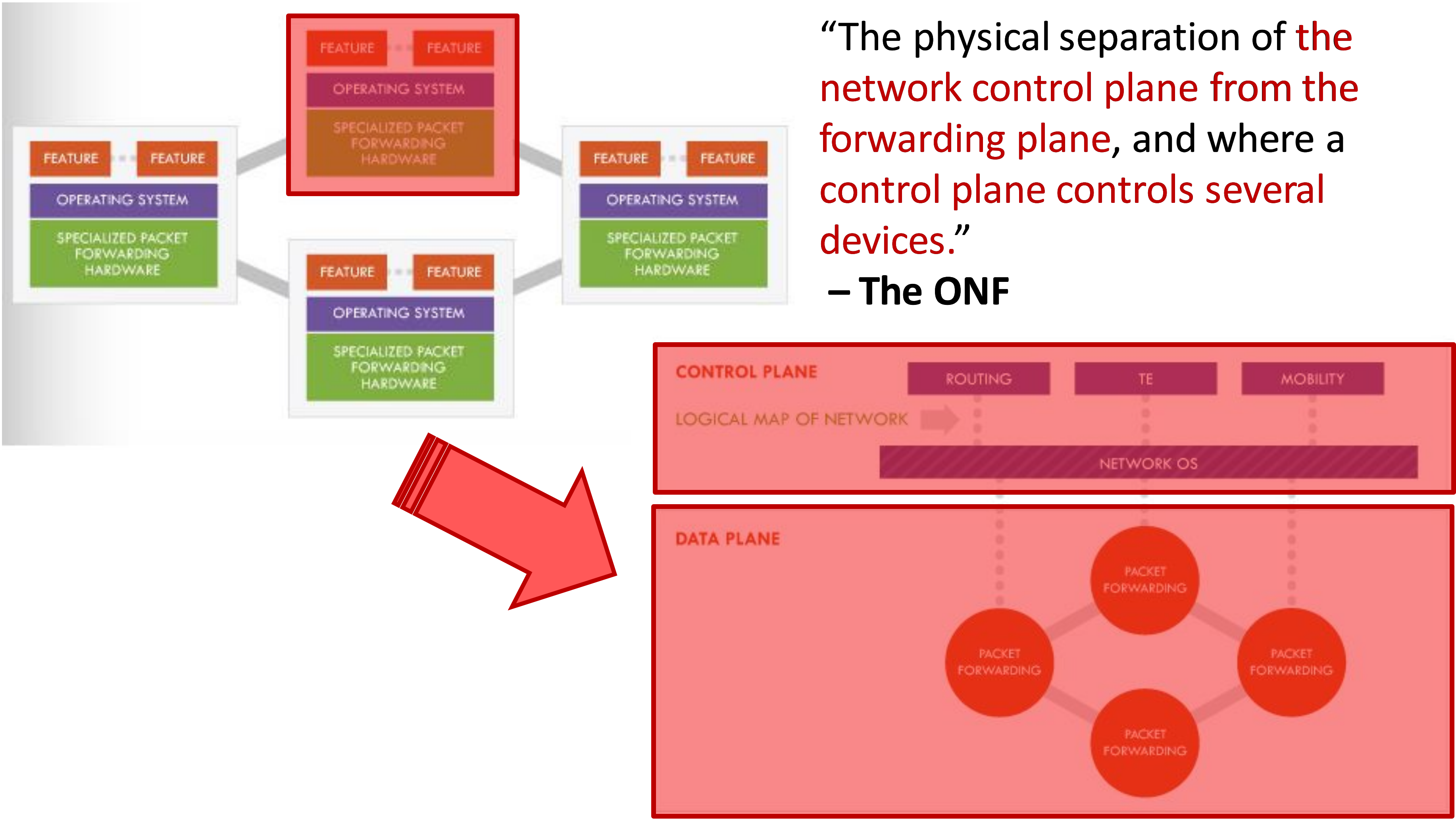
computer NETWORKS

What is SDN?

"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 one Slide

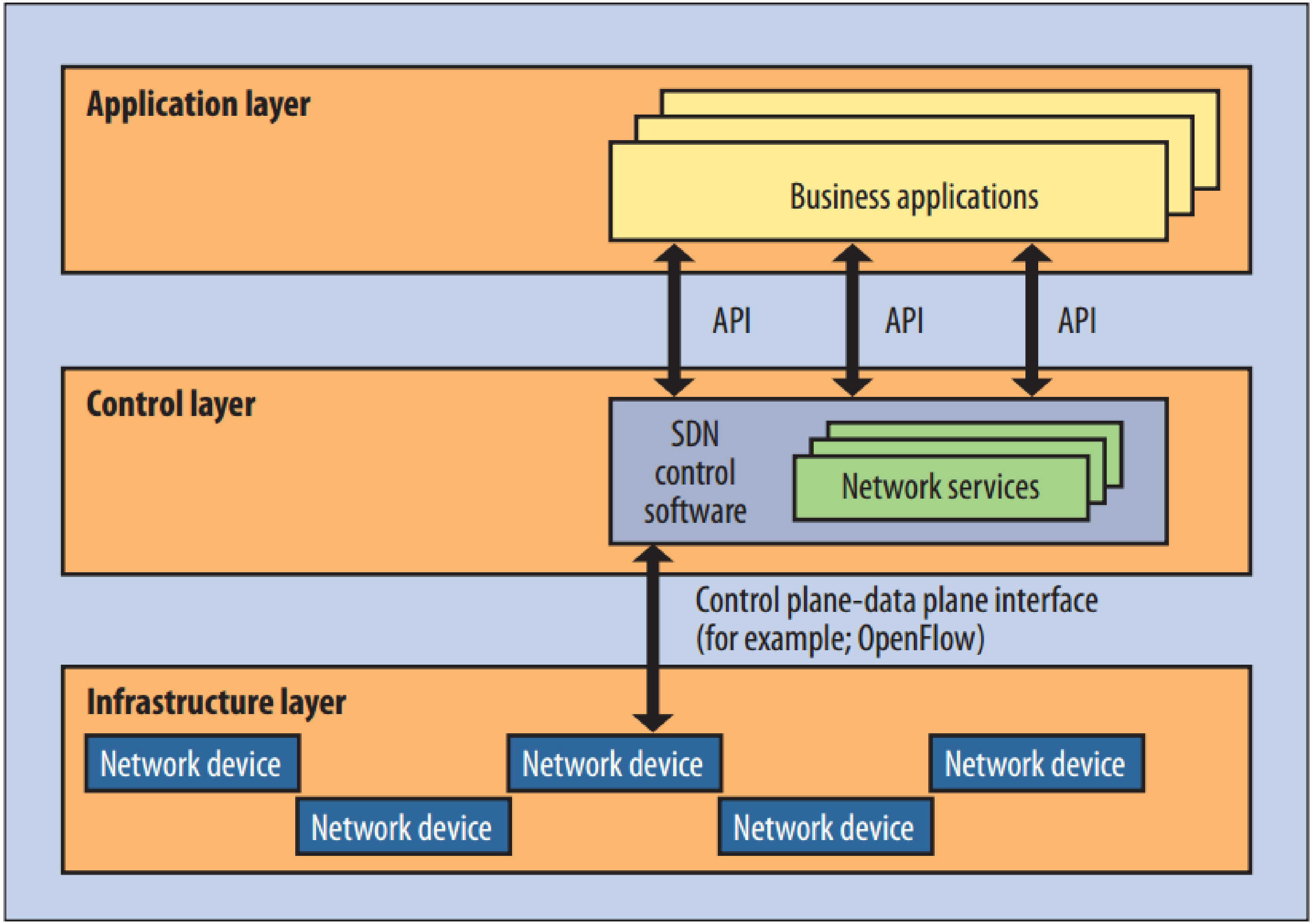


"The physical separation of the network control plane from the forwarding plane, and where a control plane controls several devices."

– **The ONF**

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

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

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

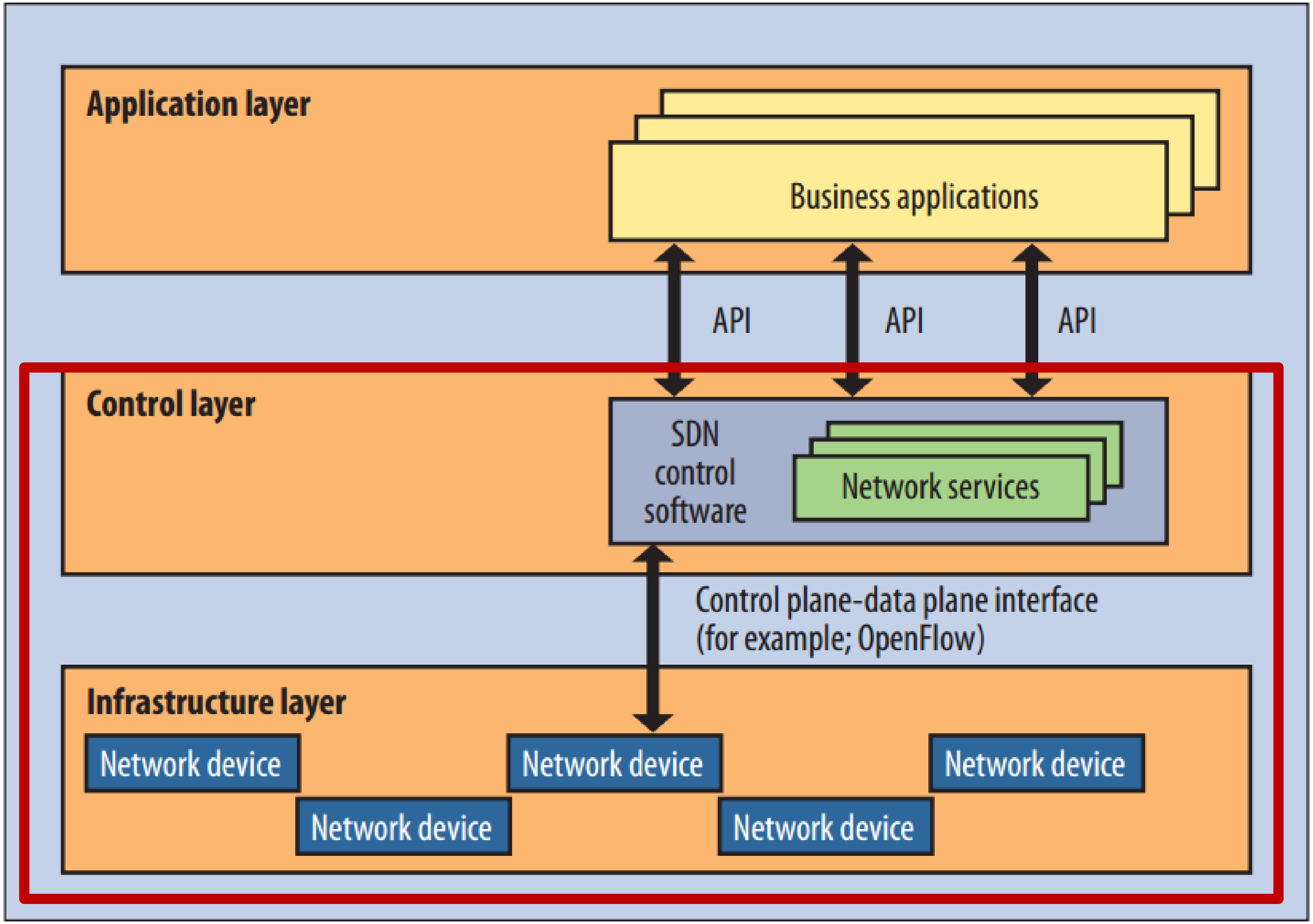
- SDN widely accepted as "**future of networking**"
  - ~1000 engineers at latest Open Networking Summit
  - *Acceptance in both industry and academia*
- Insane level of SDN hype, and still:
  - SDN doesn't work miracles, merely makes things easier

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

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

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

OpenFlow

OpenFlow is one implementation of the Southbound interface in SDN

Standardized by the ONF

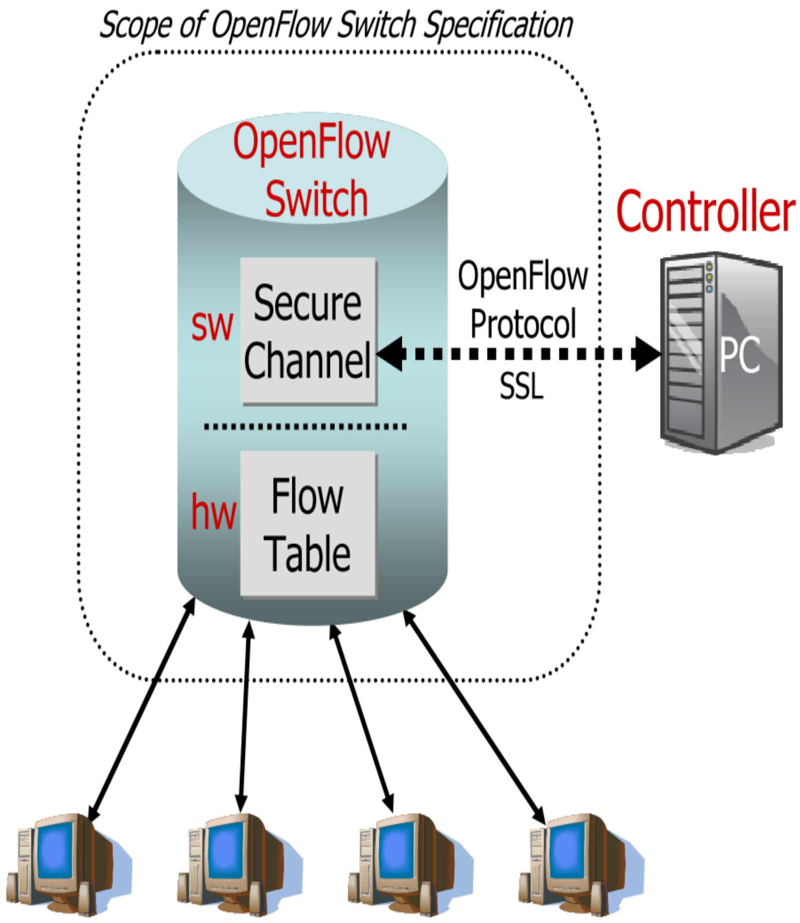
OpenFlow is NOT SDN

OpenFlow is NOT THE ONLY Southbound interface

(see, e.g., Cisco OpFlex)

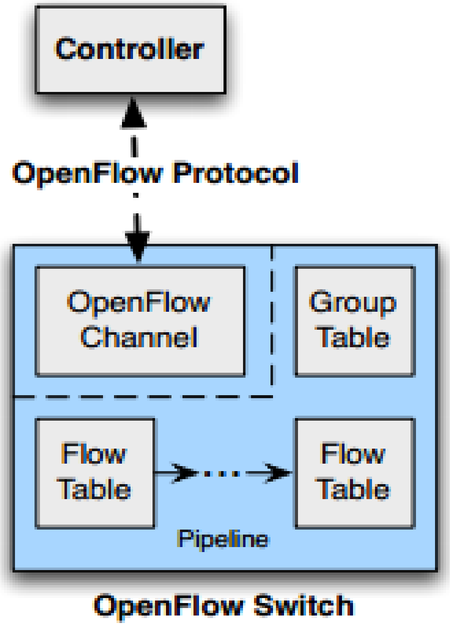
Components of an OpenFlow Network

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



OpenFlow

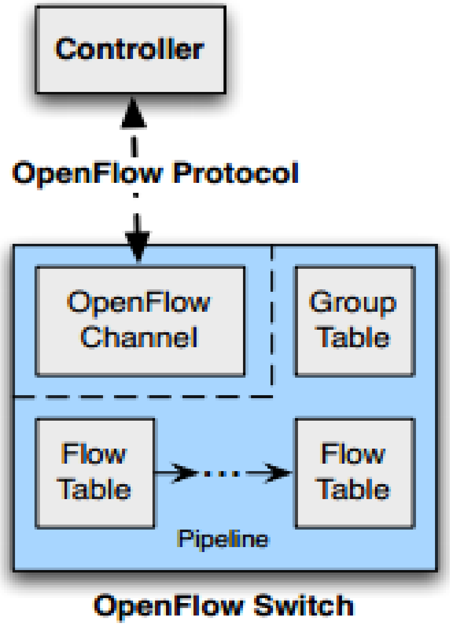
- Communication between the controller and the network devices (i.e., switches)



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)

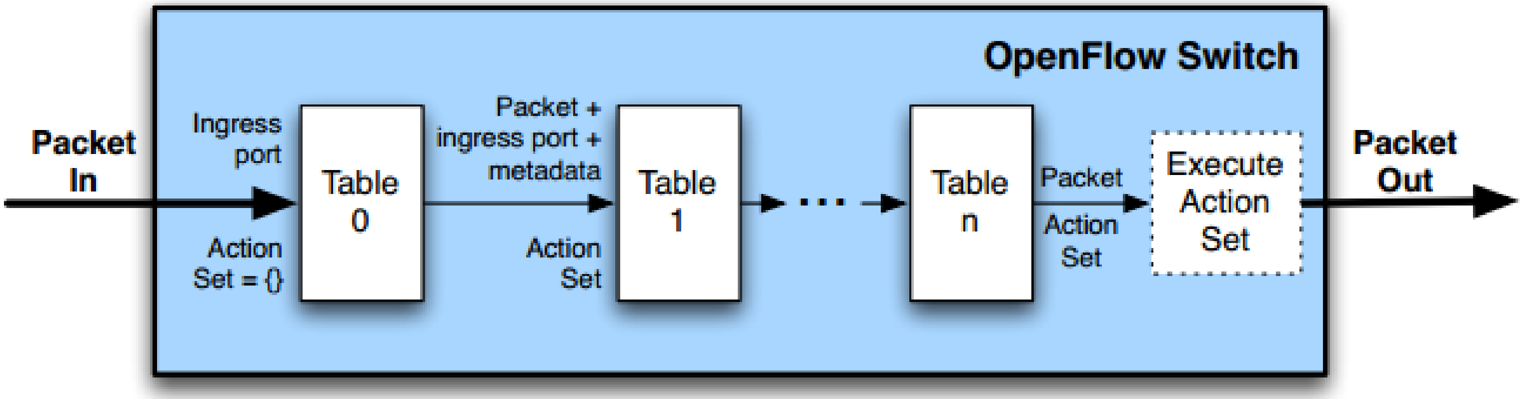
OpenFlow – Components

- 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



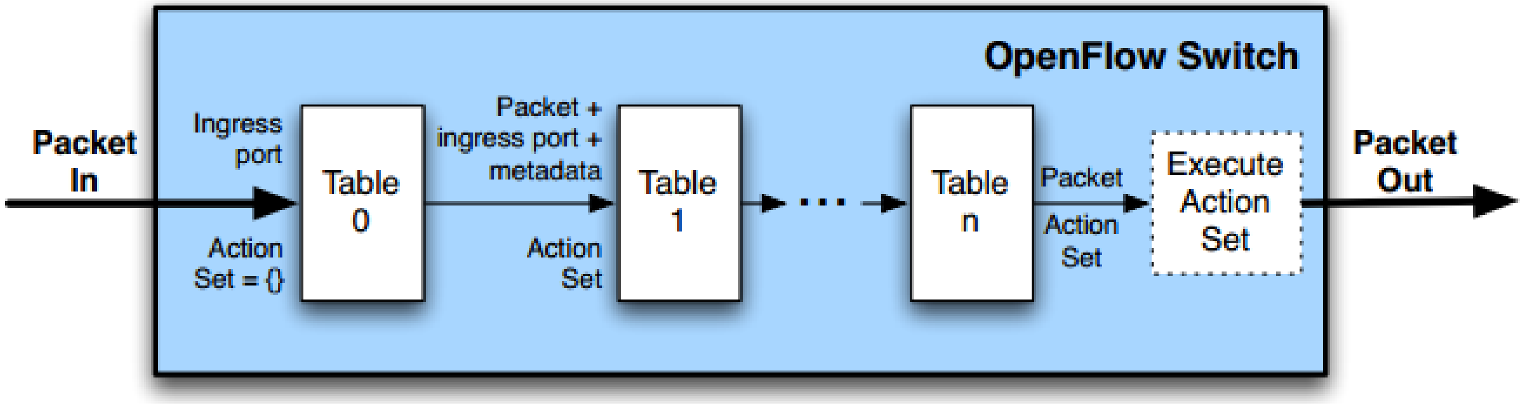
OpenFlow – Switches

- 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

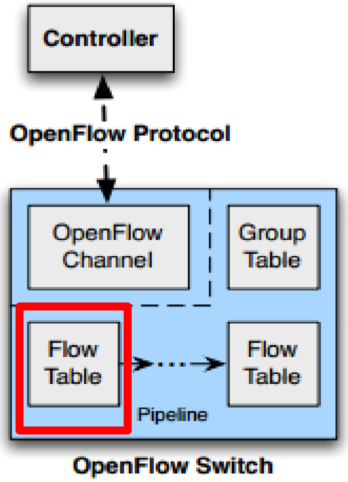


OpenFlow – Switches

- 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

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

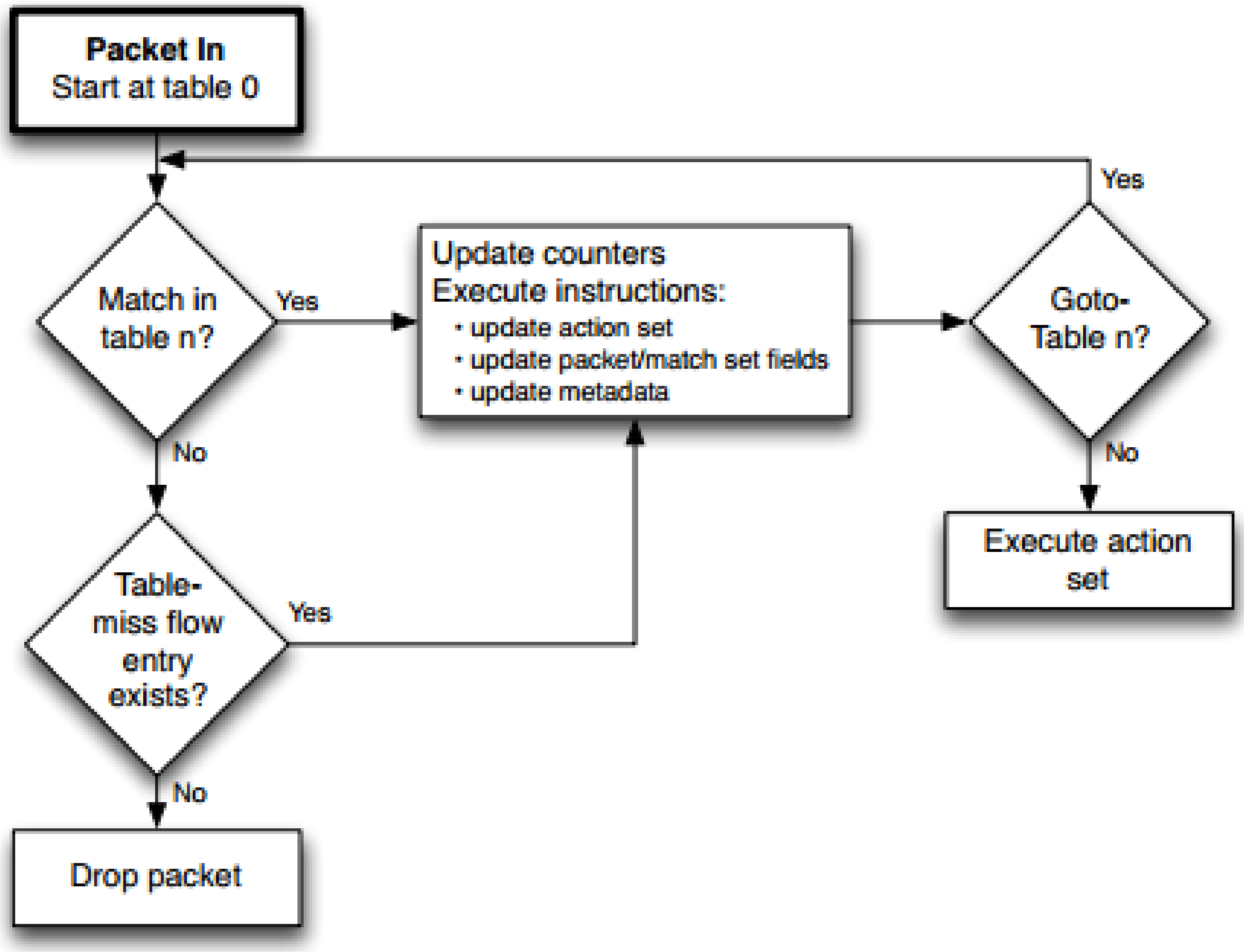
Routing

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

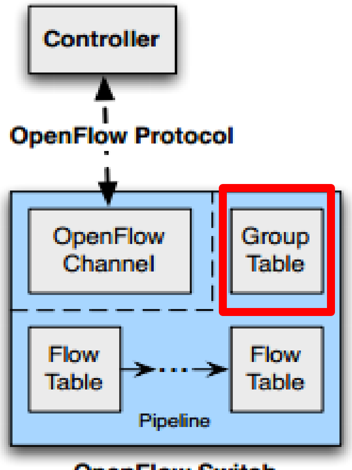


OpenFlow – Switches

- 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

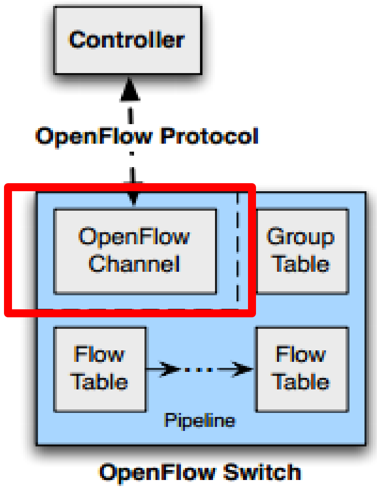
- Group Table entry structure:

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 – More features

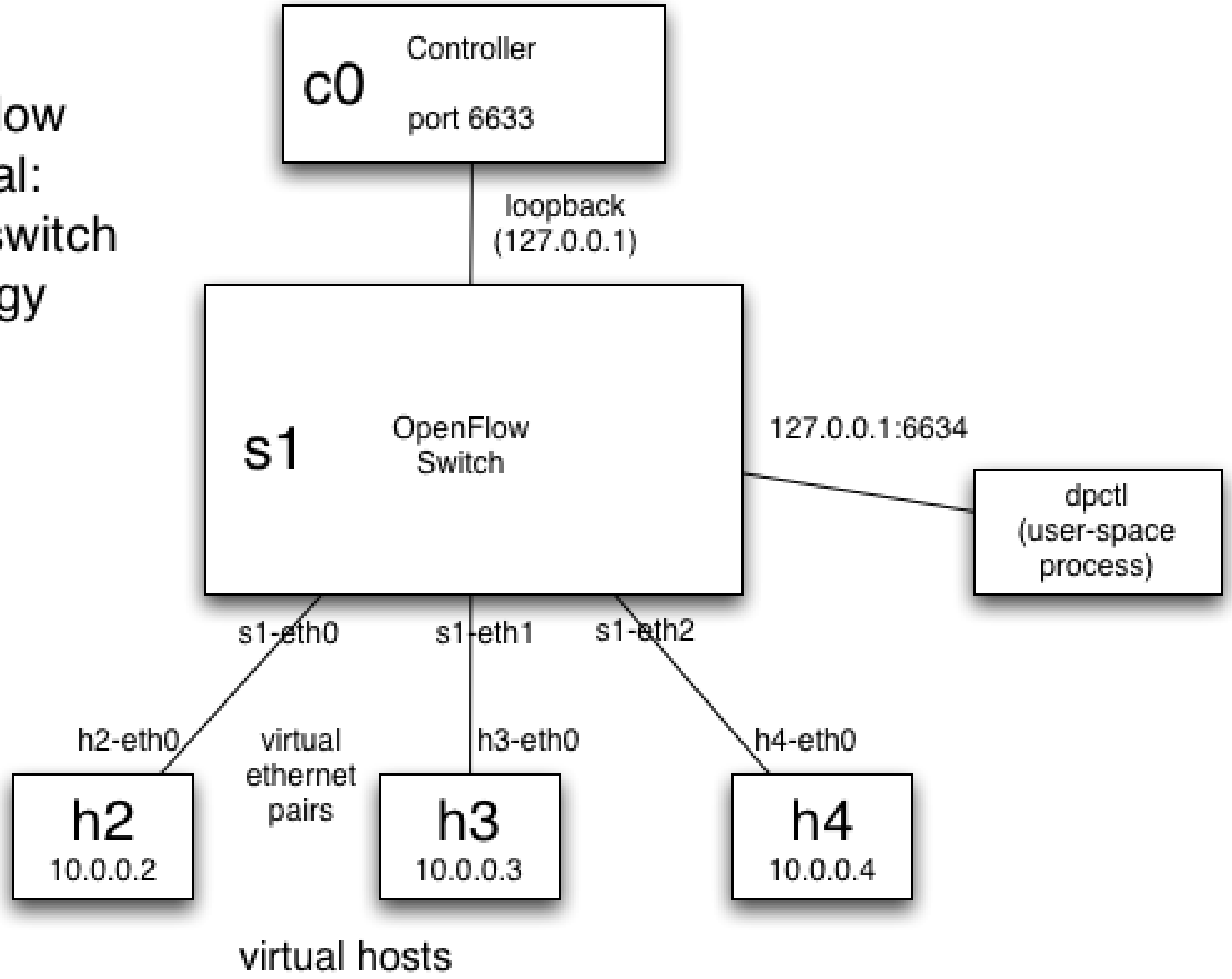
- Tools for traffic management
  - Meter tables for flows
  - Allow for traffic shaping

- Tools for traffic monitoring
  - Statistics can be gathered from switches

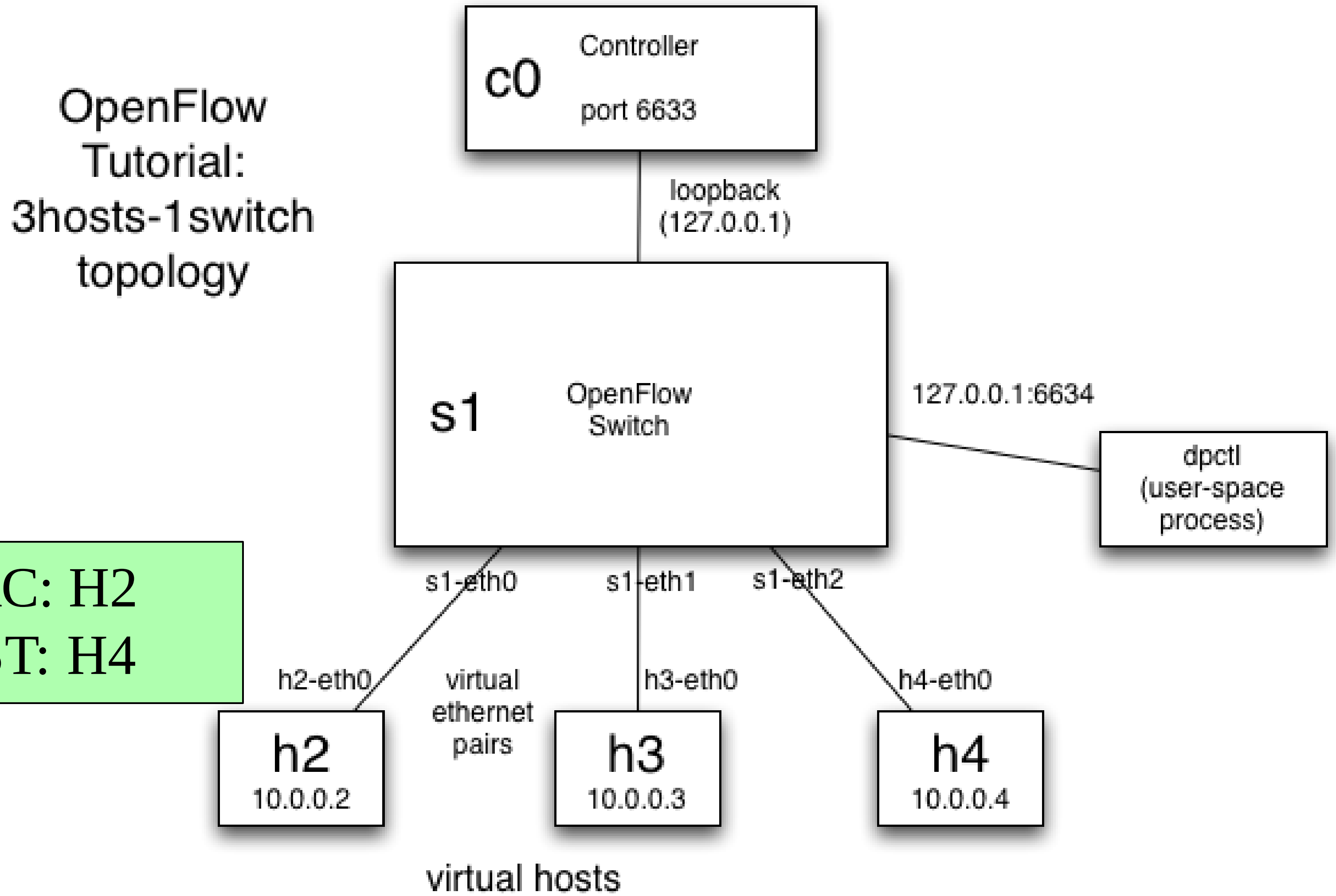
- Details out of scope of this lecture

OpenFlow - Example

OpenFlow Tutorial: 3hosts-1switch topology

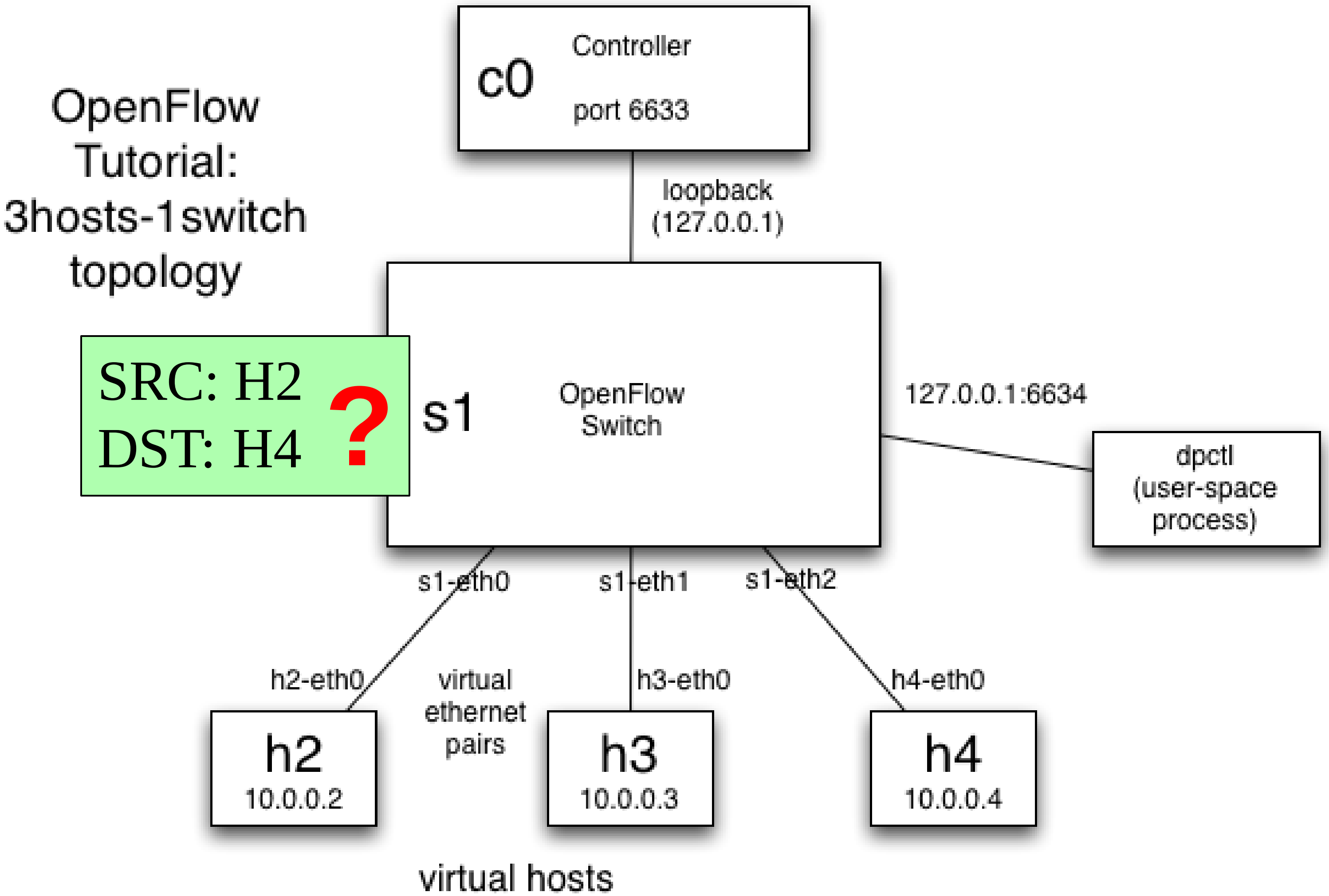


OpenFlow - Example

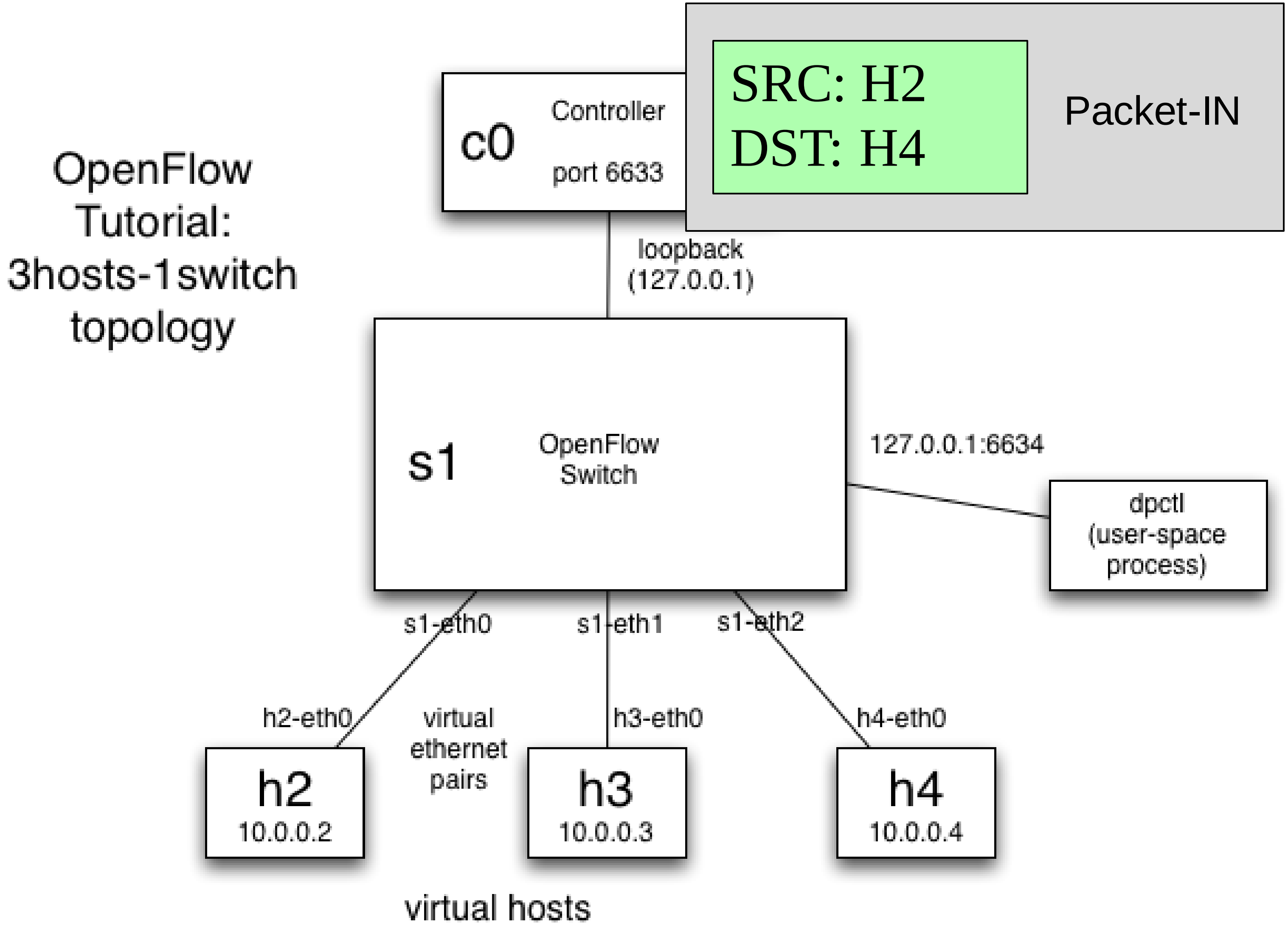


SRC: H2
DST: H4

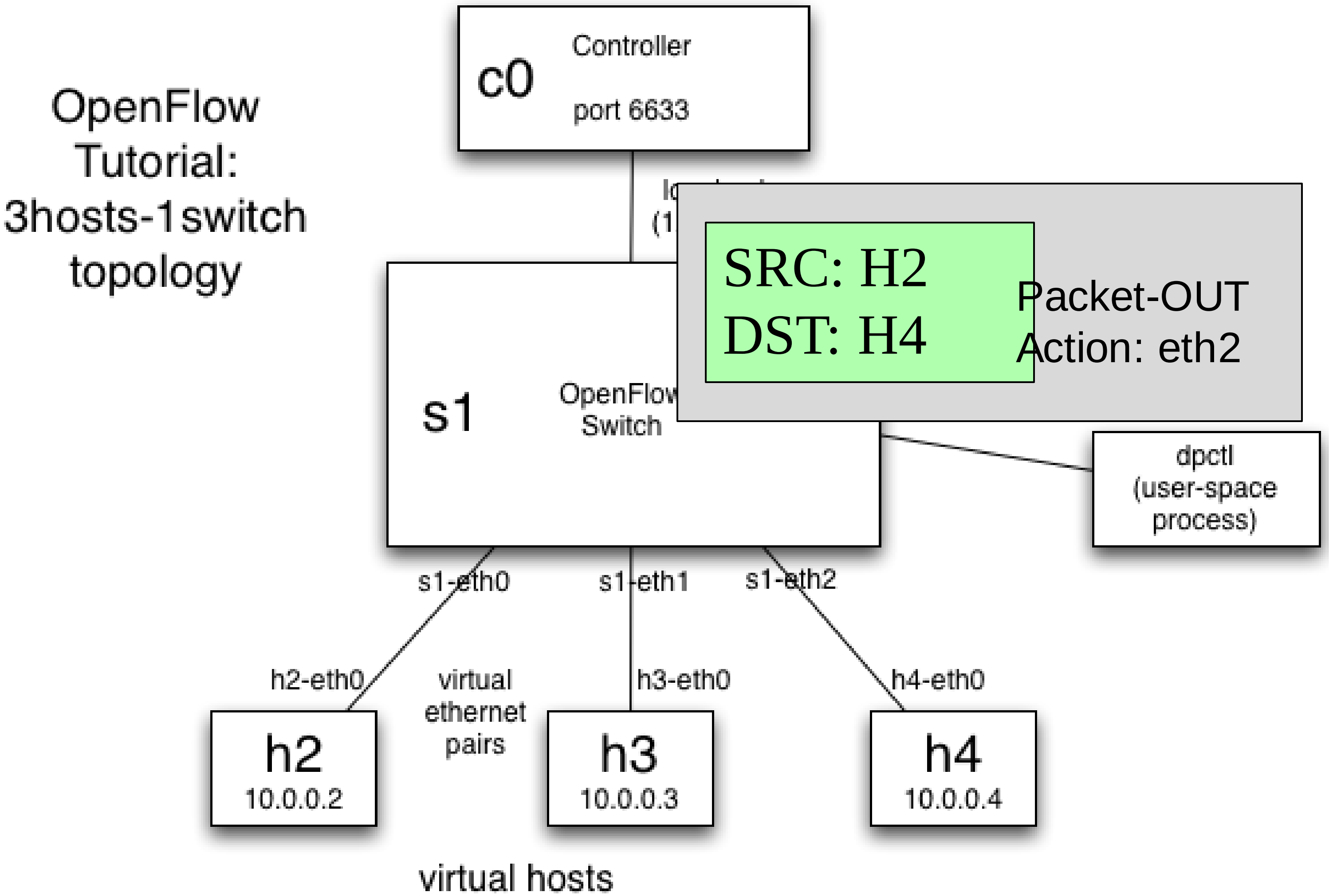
OpenFlow - Example



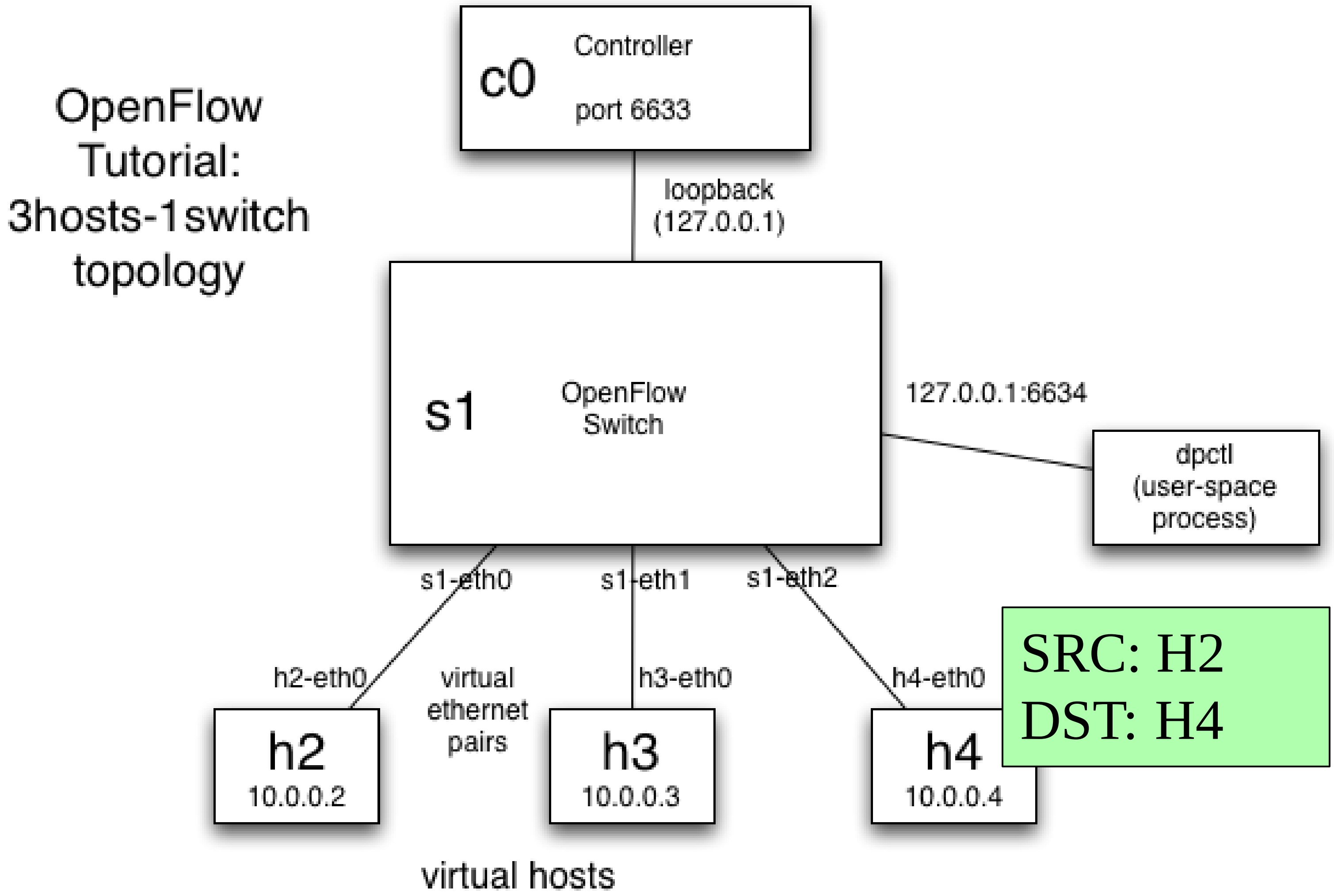
OpenFlow - Example



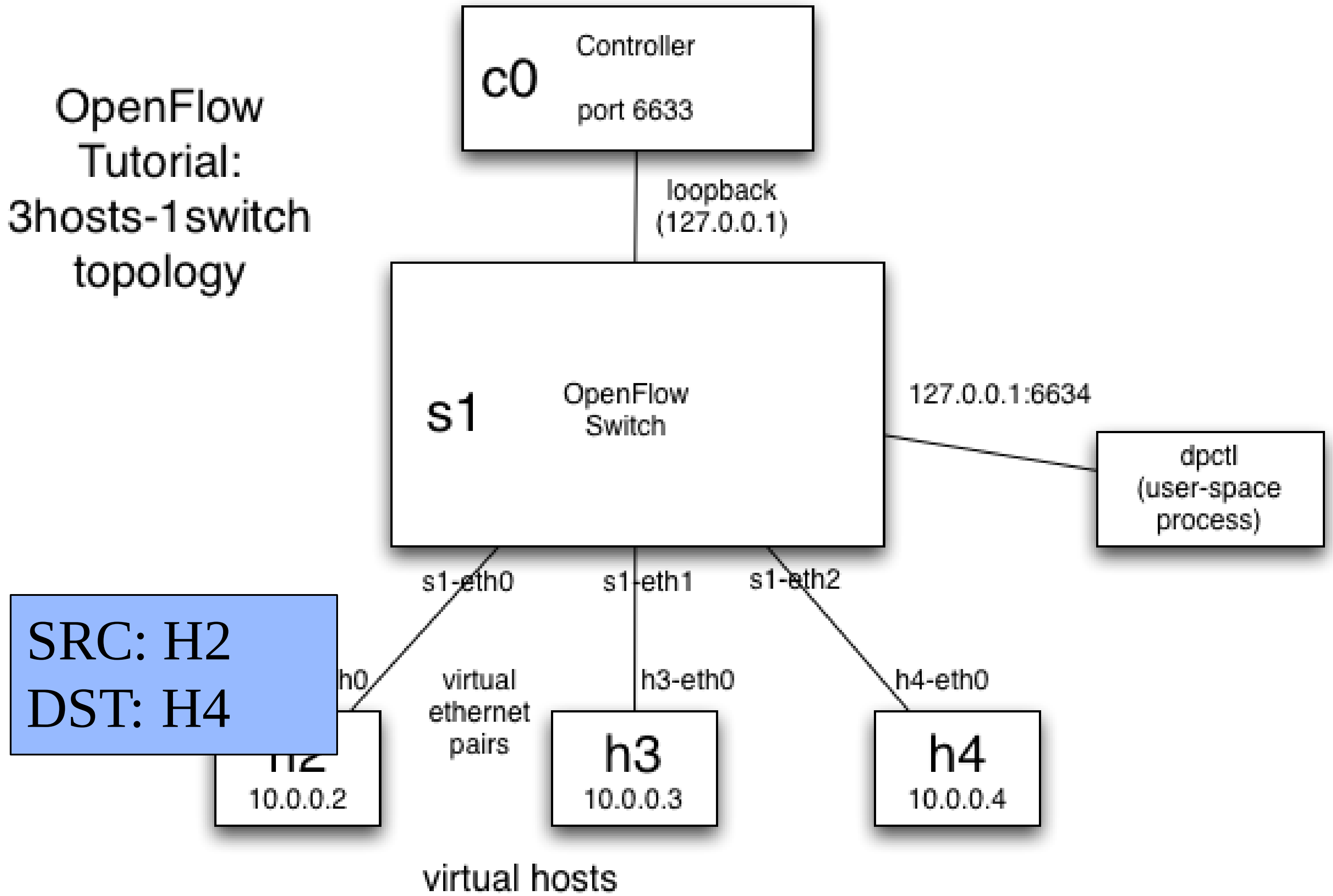
OpenFlow - Example



OpenFlow - Example

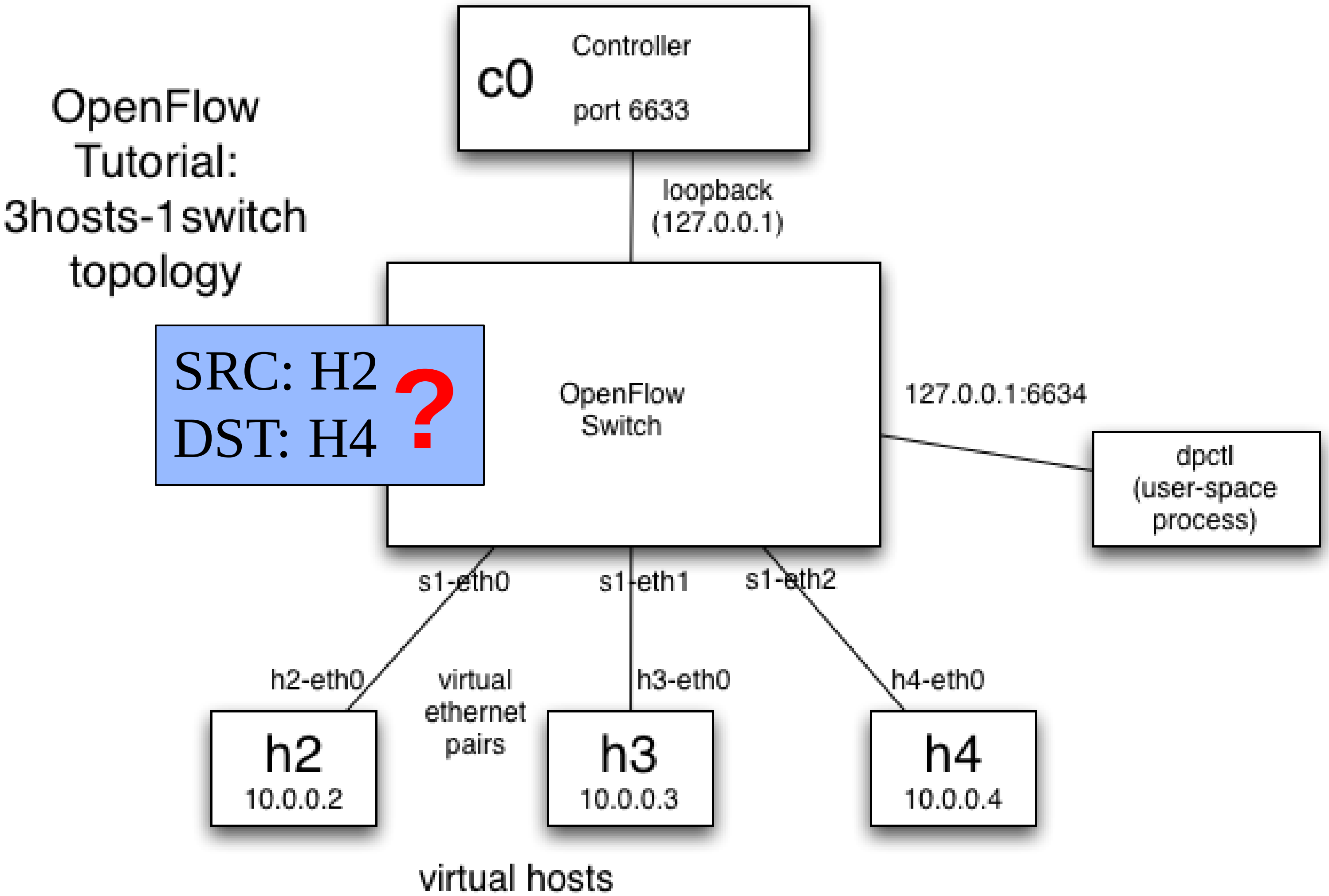


OpenFlow - Example

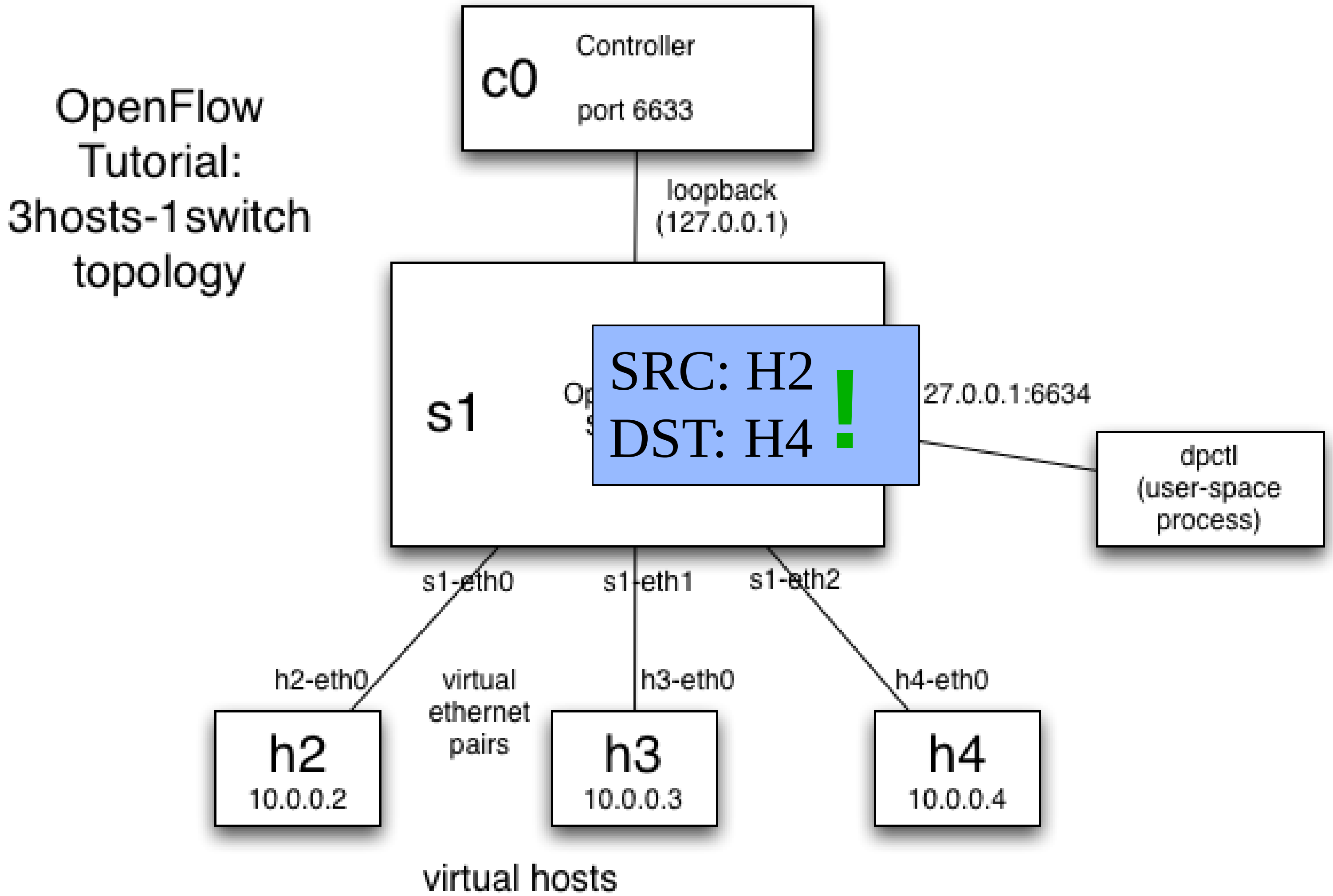


SRC: H2
DST: H4

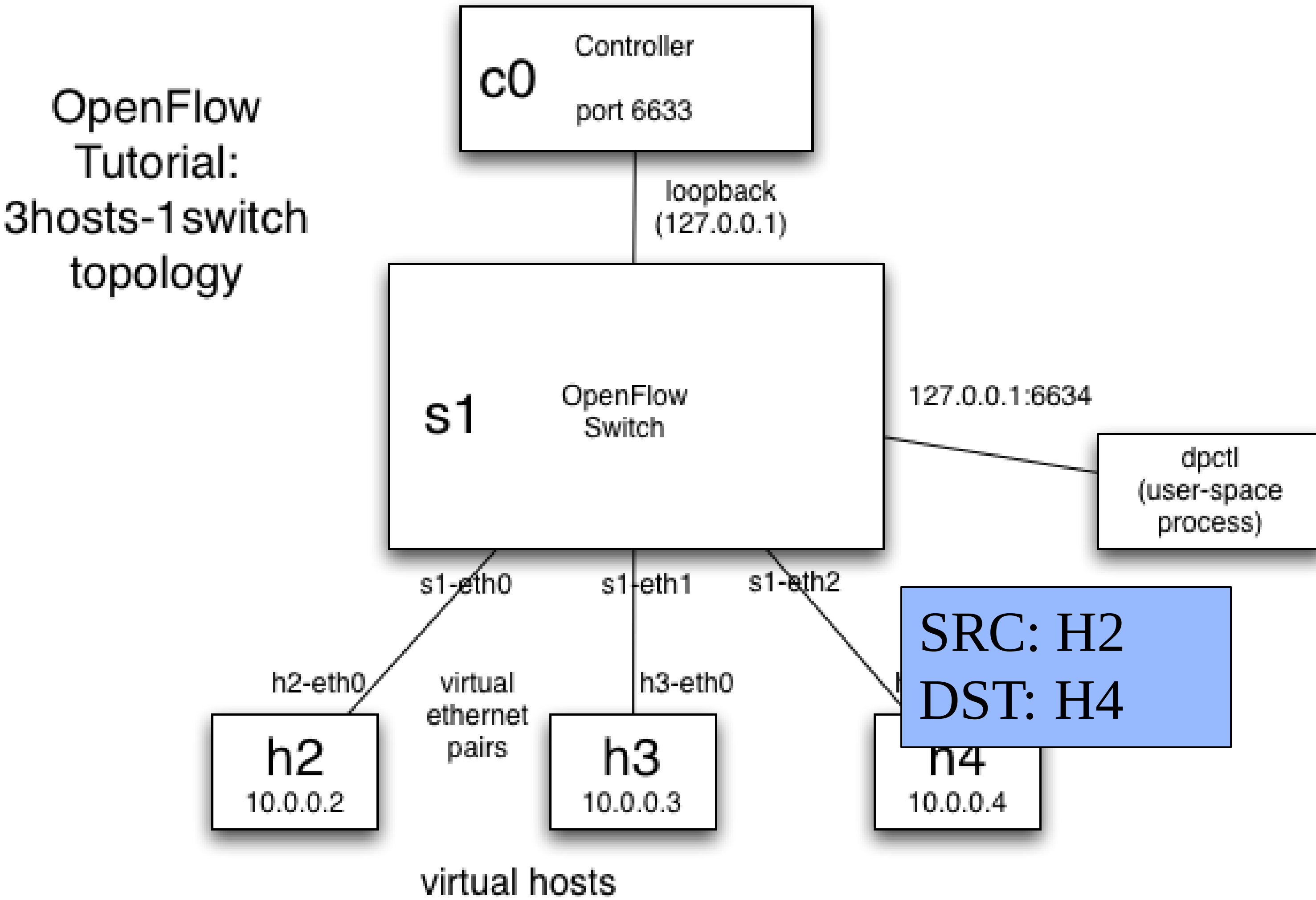
OpenFlow - Example



OpenFlow - Example



OpenFlow - Example



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.

computer NETWORKS

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

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

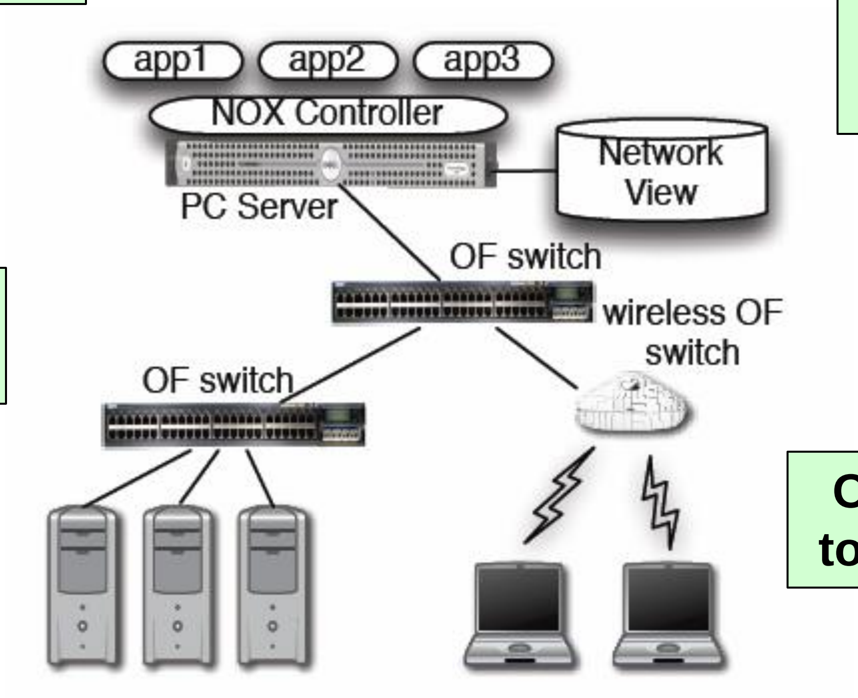
[1] Gude et al. "NOX: towards an operating system for networks." *ACM SIGCOMM CCR* 38.3 (2008): 105-110.

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.

computer NETWORKS

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

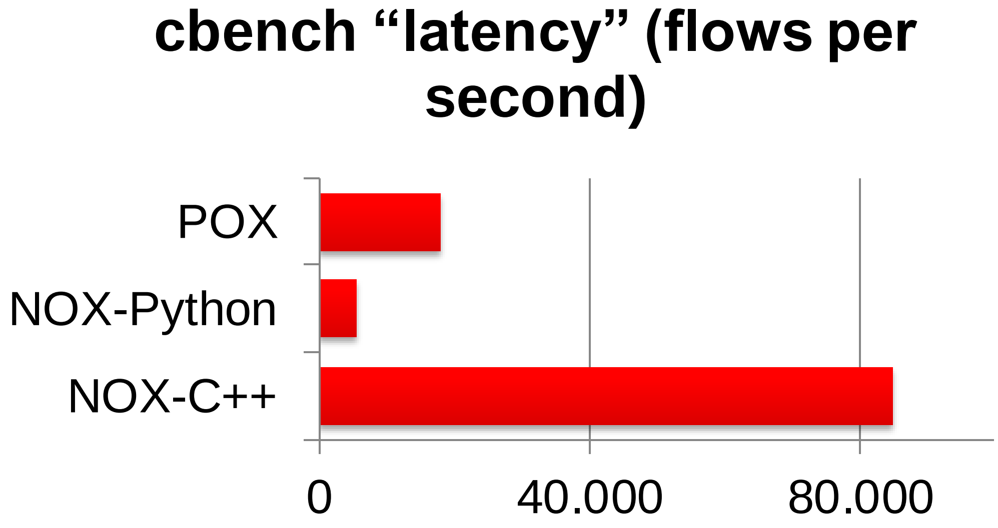
POX [1]

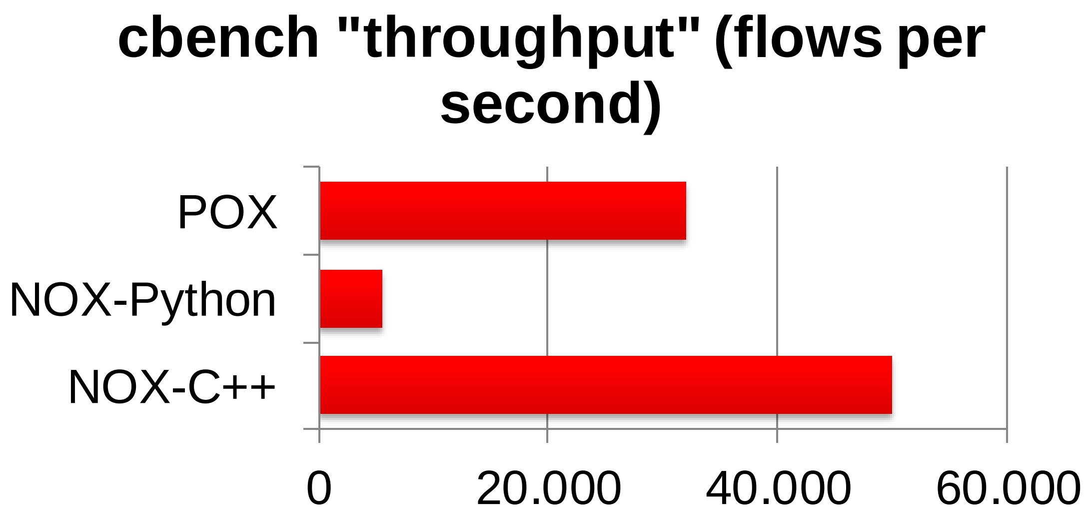
- **POX = NOX in Python**

- 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
- 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

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

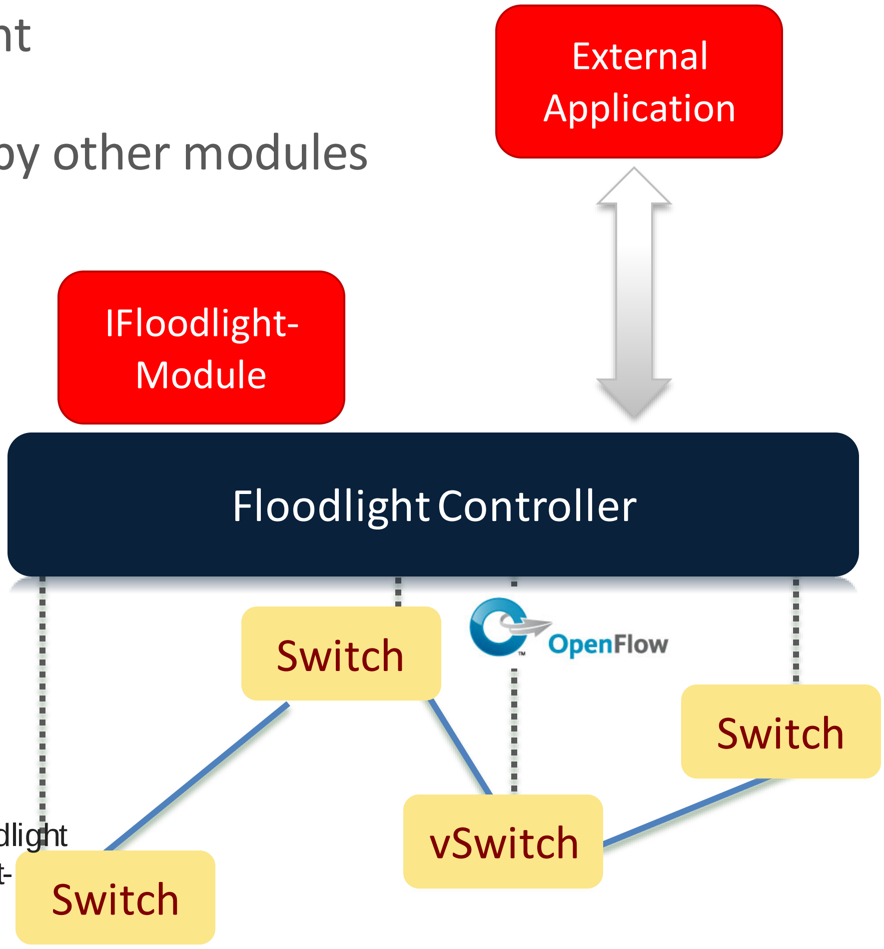
FloodlightProvider (IFloodlightProviderService)	• Translates OF messages to Floodlight events • Managing connections to switches via Netty
TopologyManager (ITopologyManagerService)	• Computes shortest path using Dijsktra • Keeps switch to cluster mappings
LinkDiscovery (ILinkDiscoveryService)	• Maintains 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

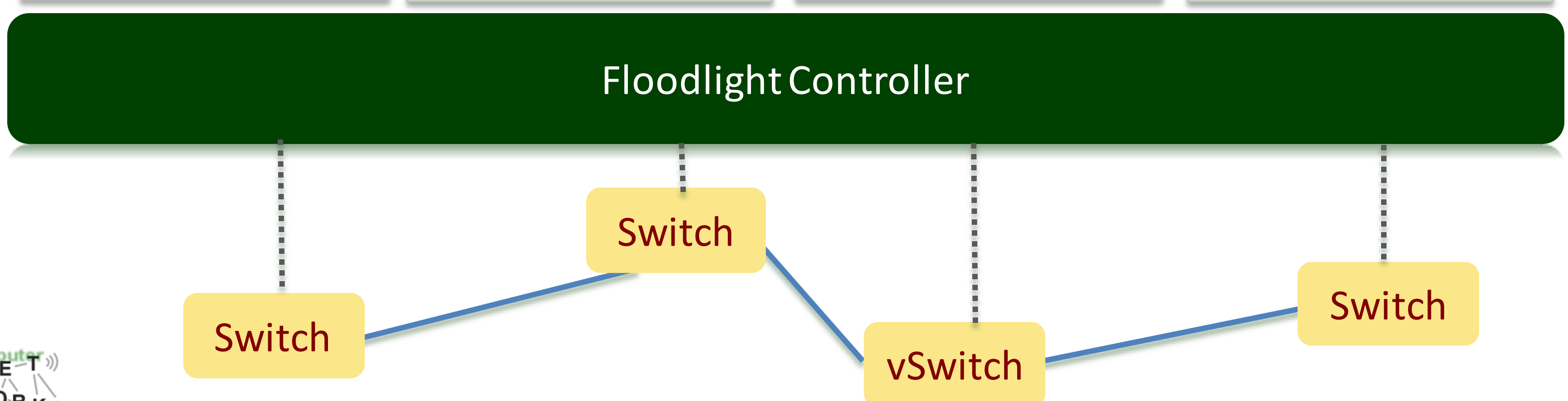
Delete Network

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

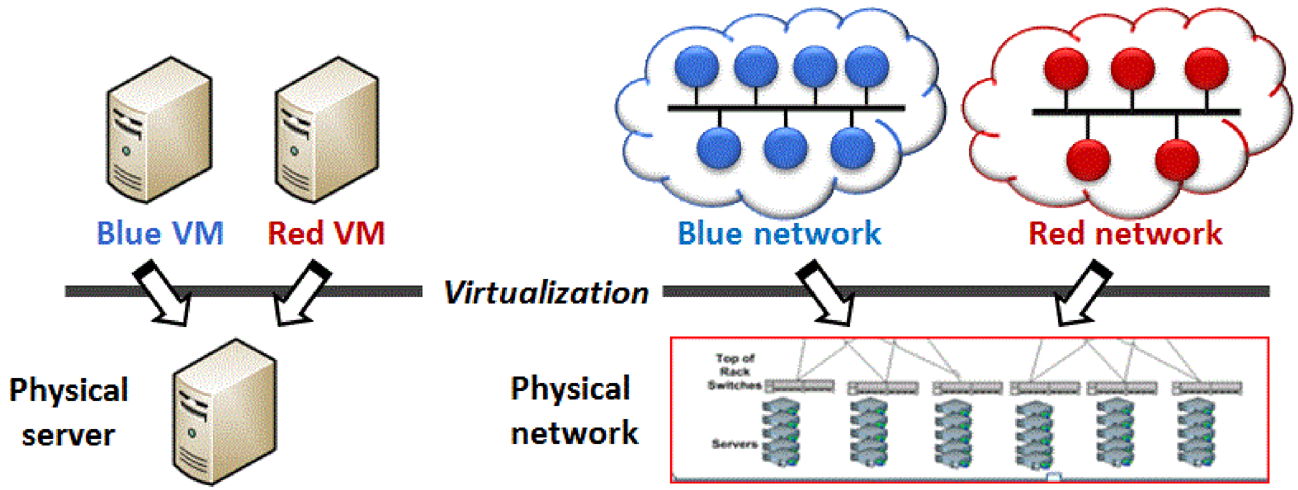
- If you know JAVA
- If you need production-level performance
- Have/want to use REST API

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

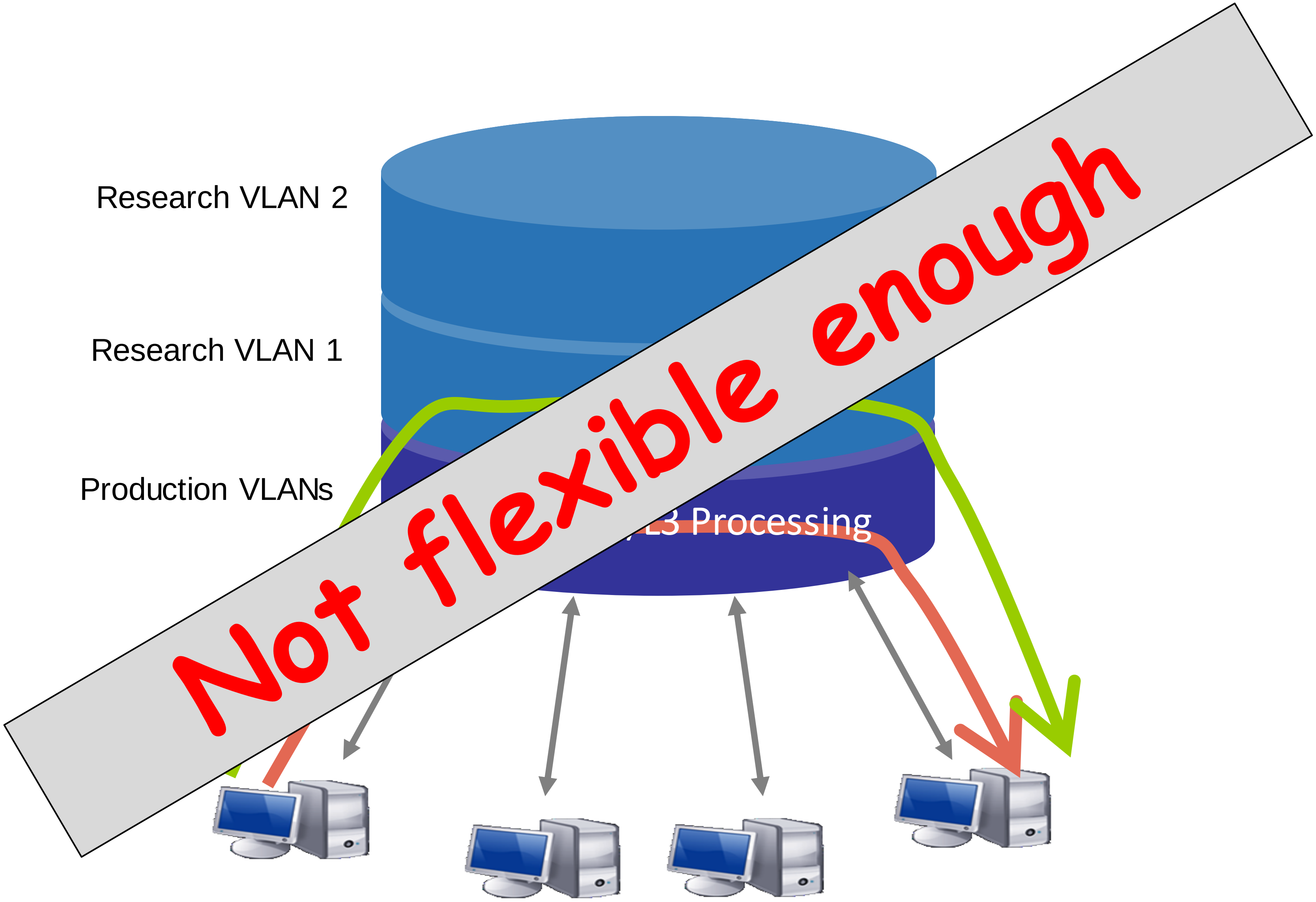
- Run multiple virtual servers on a physical server
- Each VM has illusion it is running as a physical server

Network virtualization

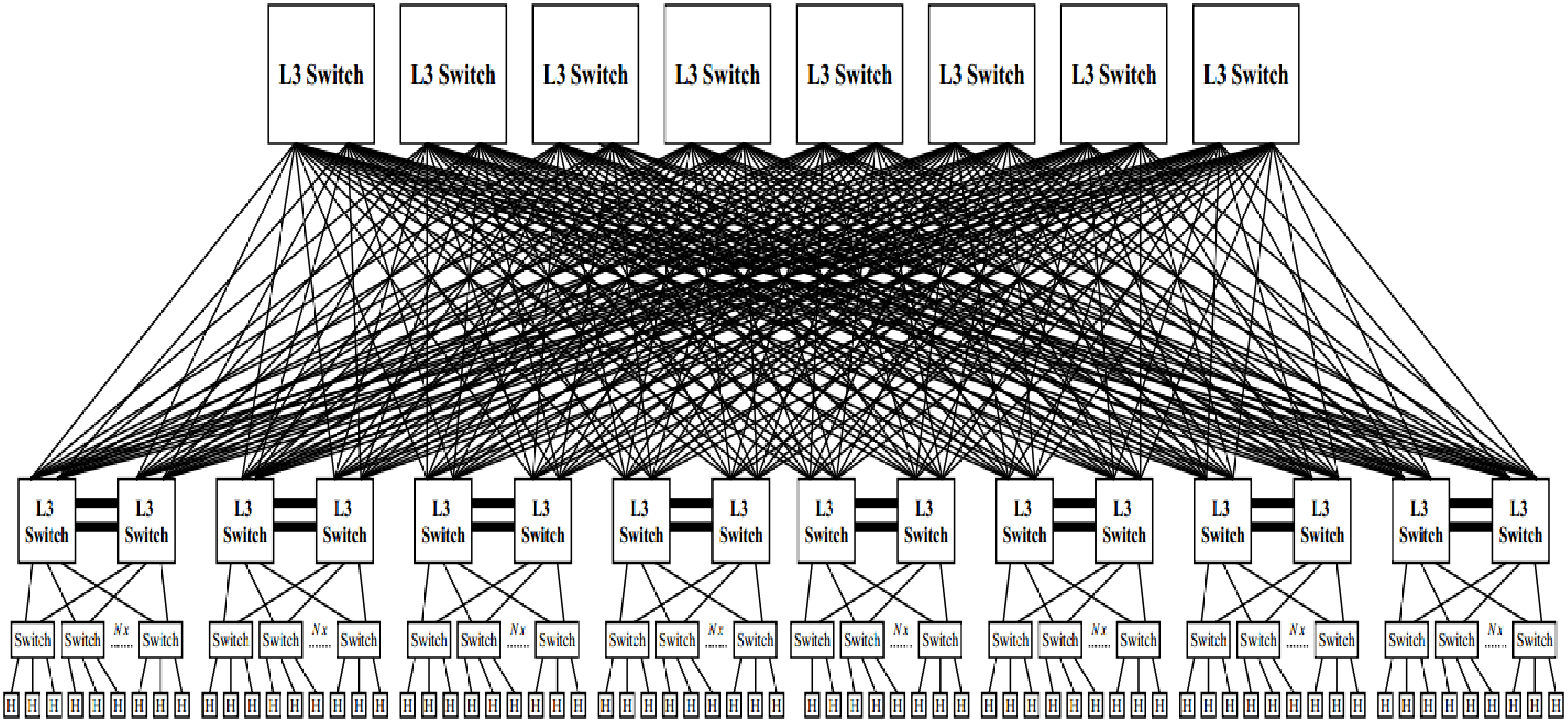
- Run multiple virtual networks on a physical network
- Each virtual network has illusion it is running as a physical network

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

Virtualization: VLANs

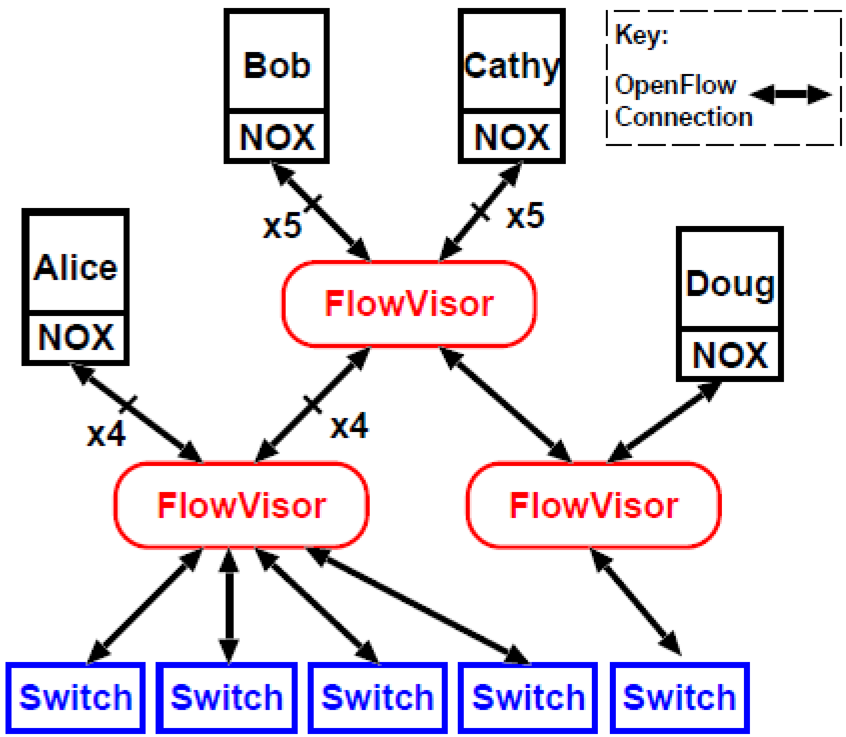


Example: Datacenter Networks



FlowVisor [1]

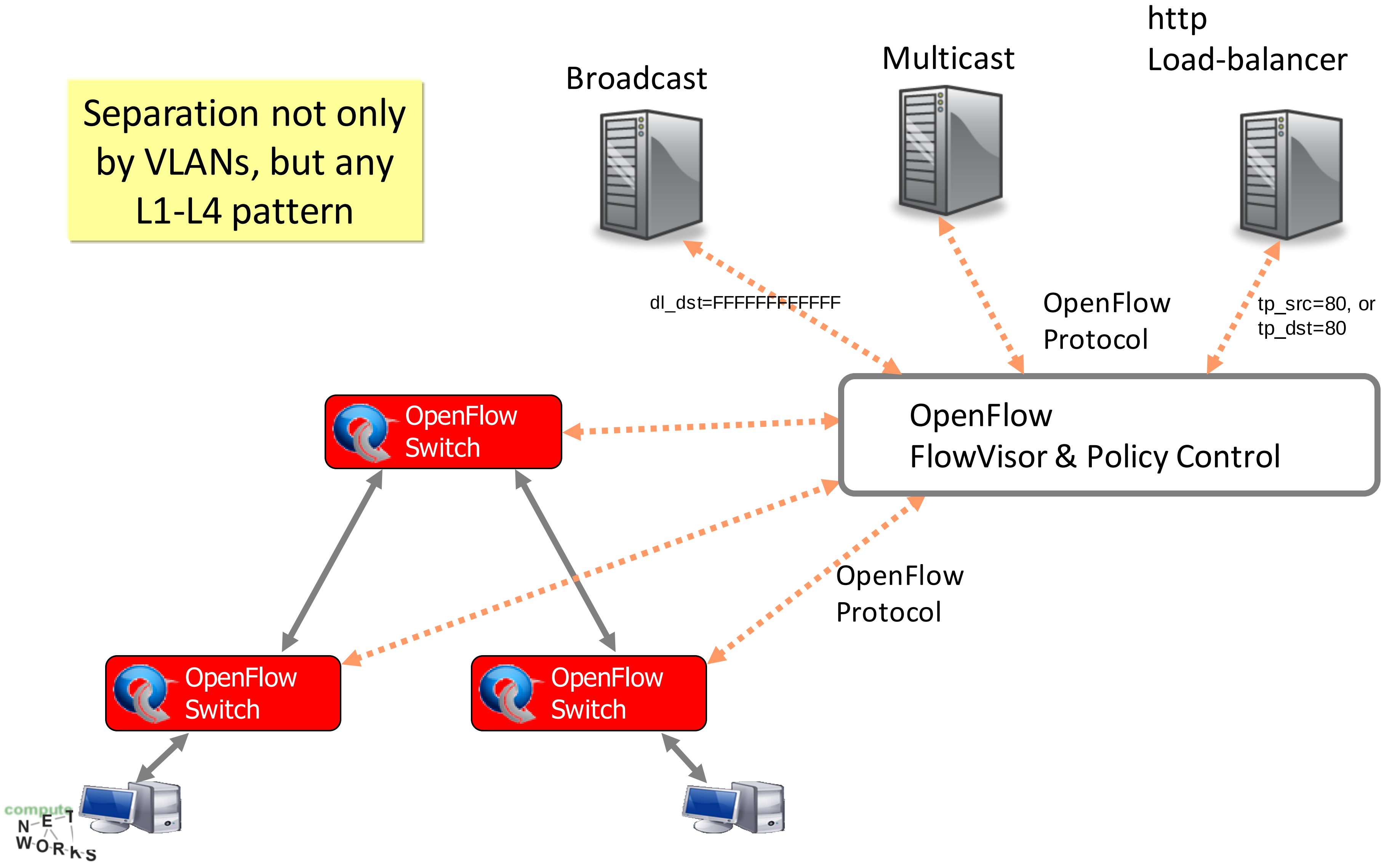
- A network hypervisor developed by Stanford
- A software proxy between the forwarding and control



[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

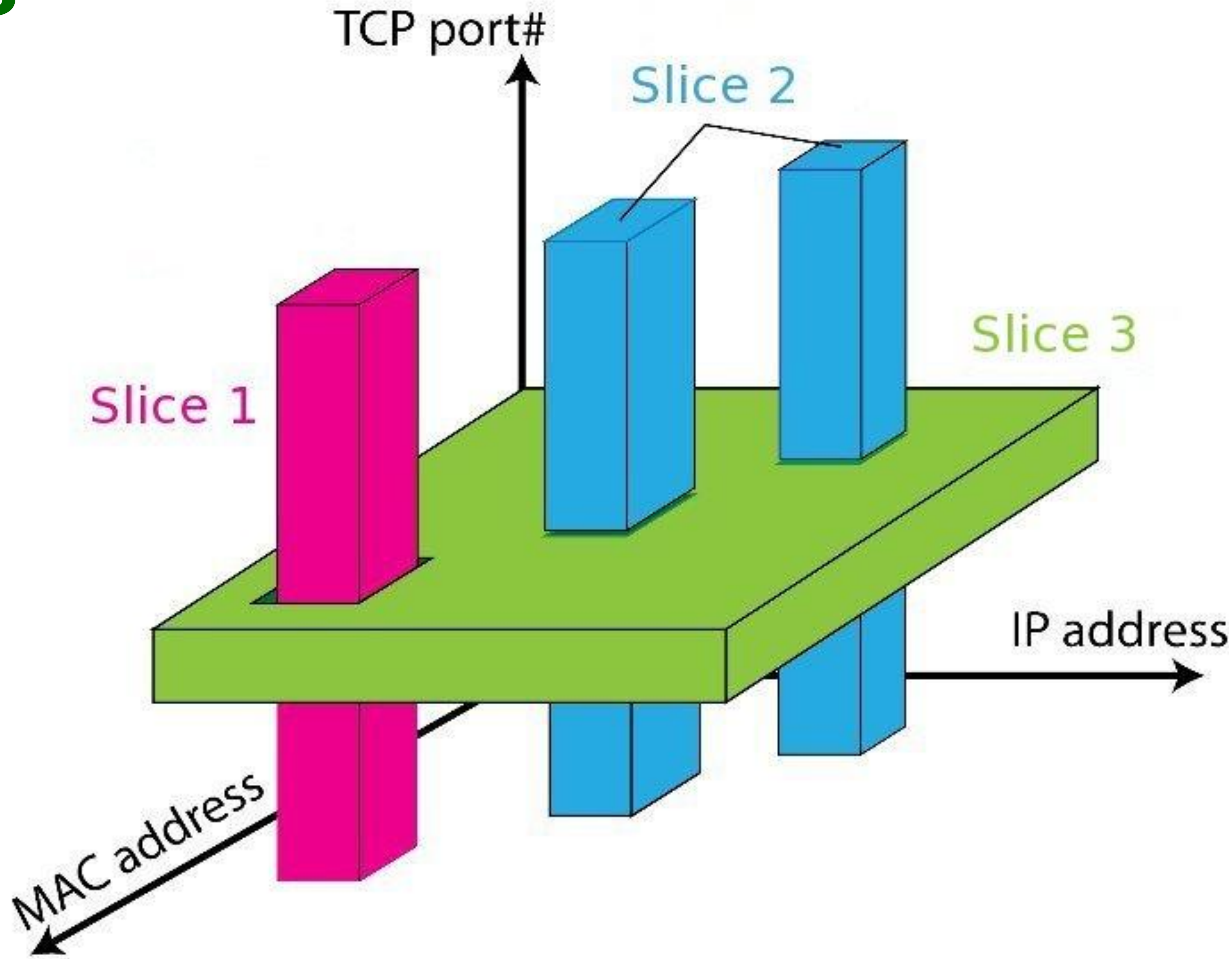
	Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport
Dave	*	*	*	*	1,2,3	*	*	*	*	*
Larry	*	*	*	*	4,5,6	*	*	*	*	*
Steve	*	*	*	*	7,8,9	*	*	*	*	*

Use Case: CDN

- 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

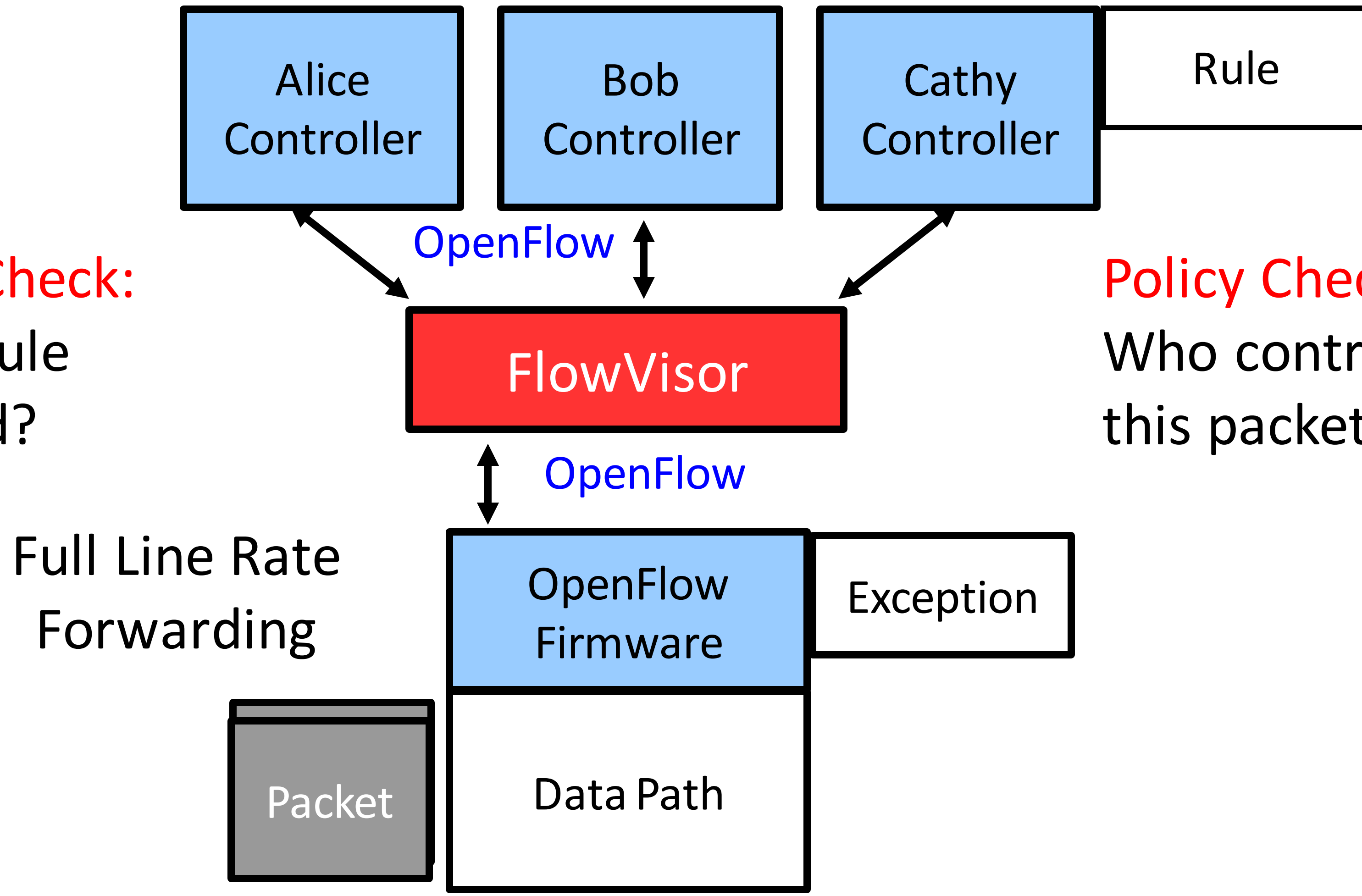
- FlowVisor intercepts OpenFlow messages from controllers
  - Rewrites flow insertion, deletion & modification rules so they don't violate the slice definition
    - Flow definition – ex. Limit Control to HTTP traffic only
    - Actions – ex. Limit forwarding to only ports in the slice

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?

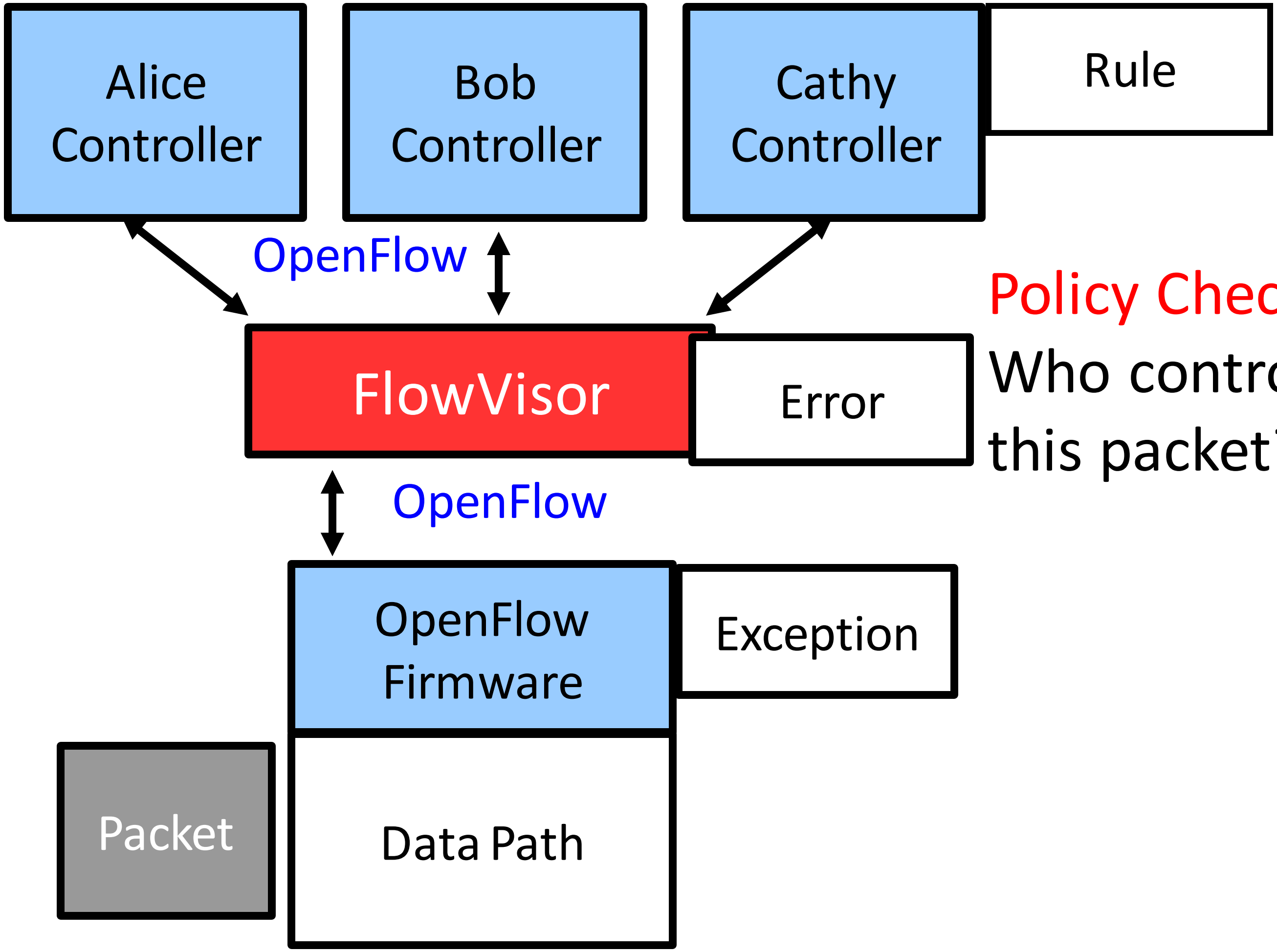


Policy Check:
Who controls this packet?

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

FlowVisor Limitations & Outlook

- Controllers can only act on disjoint sets of traffic
- Solution to this and more advanced concepts handled in dedicated SDN course
- Next week: Programmability of OpenFlow; Northbound interface