Lecture #10: Software-defined Networking

WPI CS4516 Spring 2019 D term

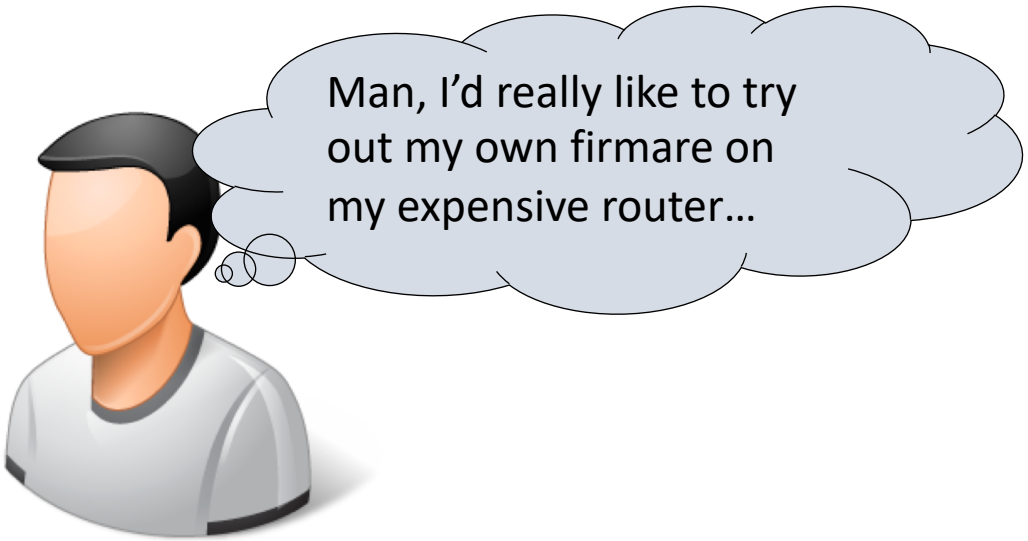
Instructor: Lorenzo De Carli (ldecarli@wpi.edu)

(slides include material from Christos Papadopoulos, CSU and Craig Shue, WPI)

What is this lecture about?

- OpenFlow: a paradigm to **simplify network management** by **centralizing control**
- We will review:
  - History & motivation
  - How OpenFlow works
  - Possible applications
  - Problems & possible solutions

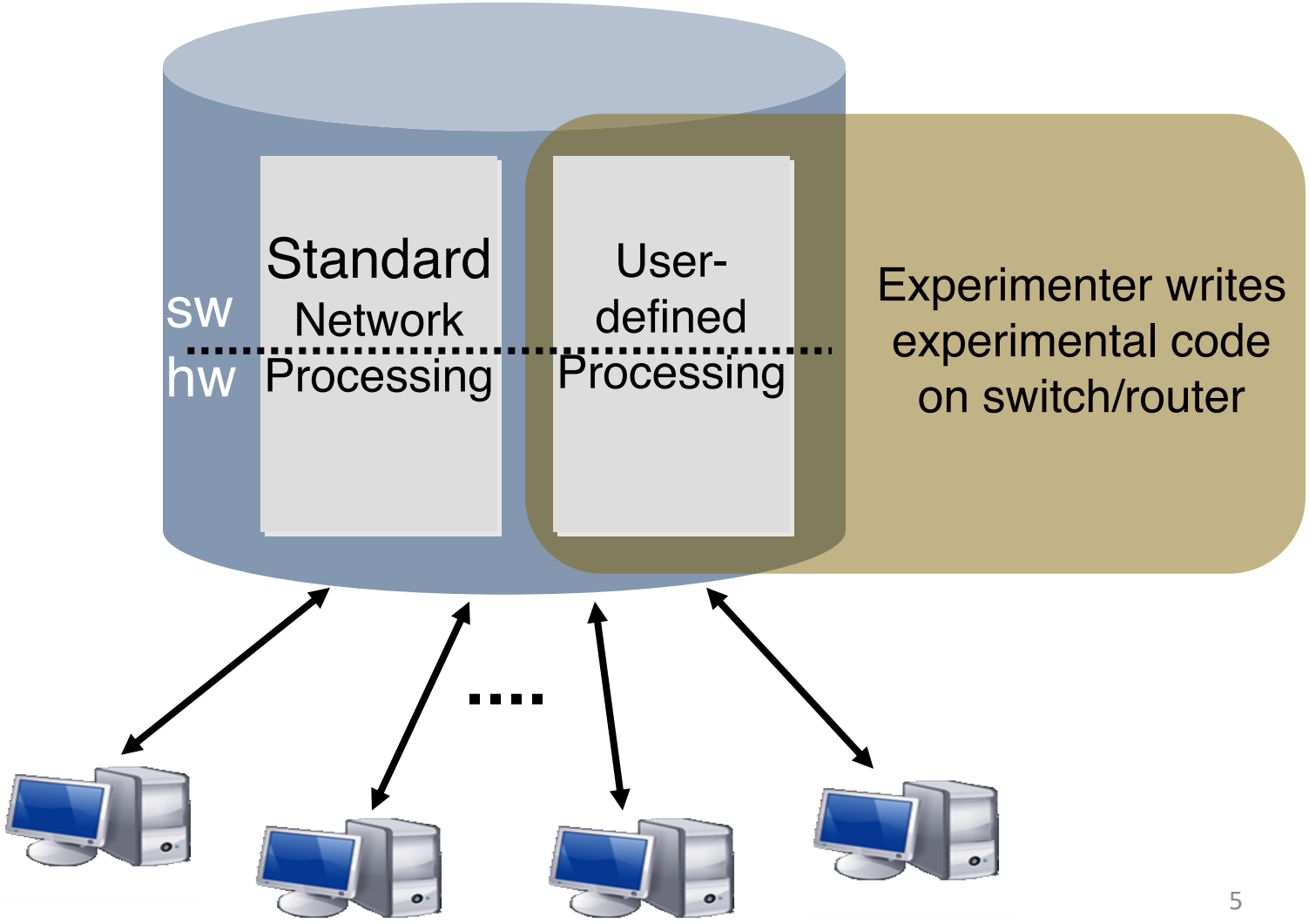
Motivation



Networking researcher

Motivation

Experimenter's Dream



Options?

- Commercial vendor **won't open software and hardware development environment**
  - Complexity of support
  - Market protection and barrier to entry

- Hard to build my own
  - **Prototypes are flakey**
  - Software only: **Too slow**
  - Hardware/software: **Fanout too small** (need >100 ports for wiring closet)

Some research equipment pre-dating OpenFlow

- **Click (Kohler et al. ‘99): modular software router**
  - Lego approach to software router
  - Composition of modules offering various functions (packet routing, scheduling, classification, …)
  - **Issue:** software routing on PC is slow
- **NetFPGA (Naous et al. ‘08): reconfigurable FGPA for networking**
  - Standard FPGA w/ Ethernet interfaces and a set of reusable packet processing modules
  - **Issue:** only four network interfaces (not enough for realistic switching/routing tasks!)

More on the Wish List

- **Isolation:** Regular production traffic untouched
- **Virtualized and programmable:** Different flows processed in different ways
- Equipment we can **trust** in our wiring closet
- **Open development environment** for all researchers (e.g. Linux, Verilog, etc).
- **Flexible definitions of a flow**
  - Individual application traffic
  - Aggregated flows
  - Alternatives to IP running side-by-side
  - …

In summary…

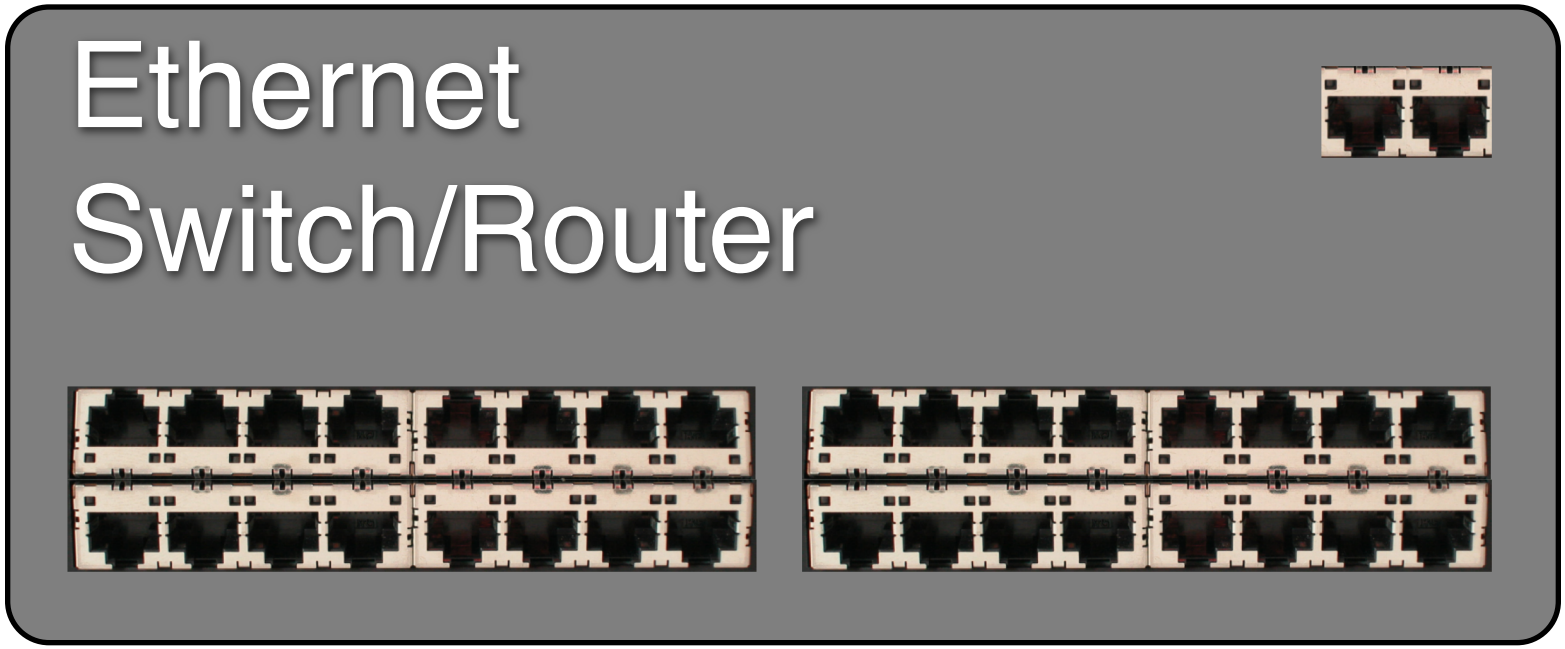
- **Research** requires **flexible networking equipment**
- **Custom-built solutions** (software, hardware) are **too limited** for many applications
- **Adding flexibility to commercial switching/routing equipment** is a much more powerful solution
  - OpenFlow was created to achieve this!

Is this all there is to know?

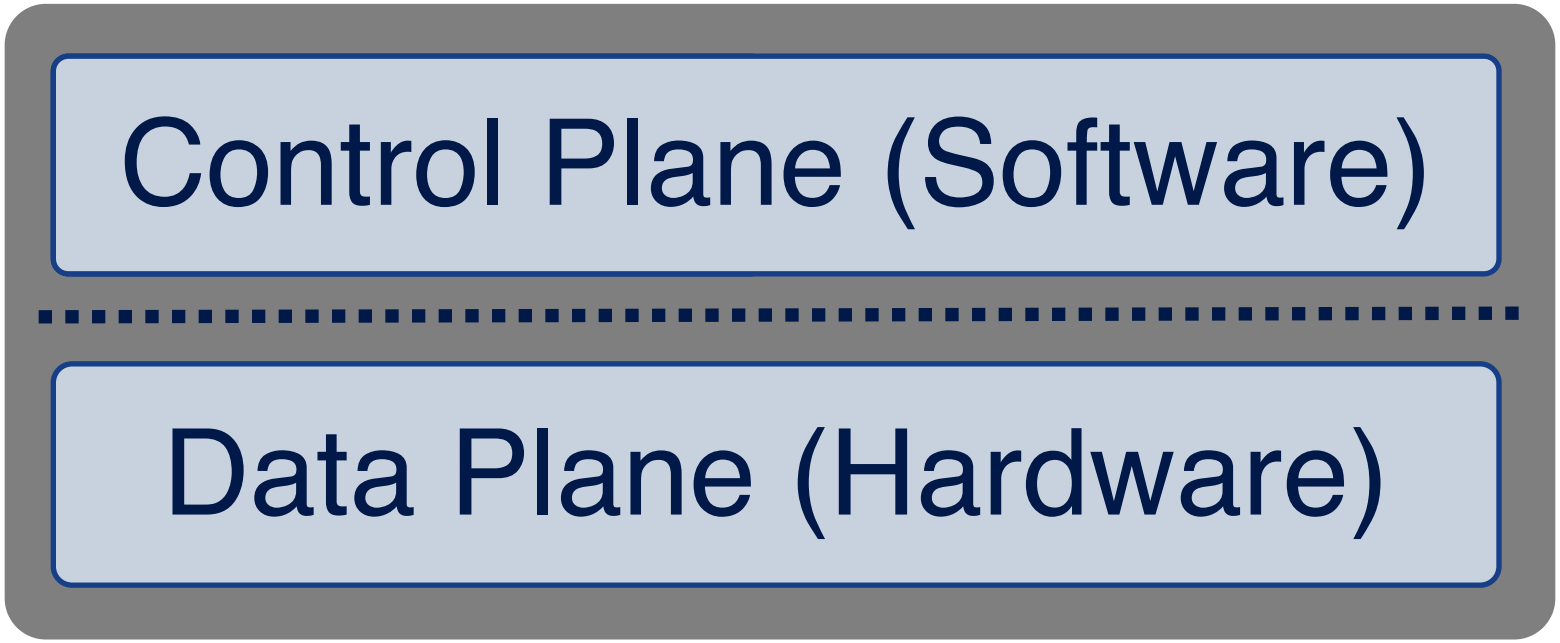
- **No!**
- OpenFlow started as an academic project but found plenty of industrial applications!
- But let’s begin from the beginning…

How does OpenFlow work?

Today's Switching Hardware

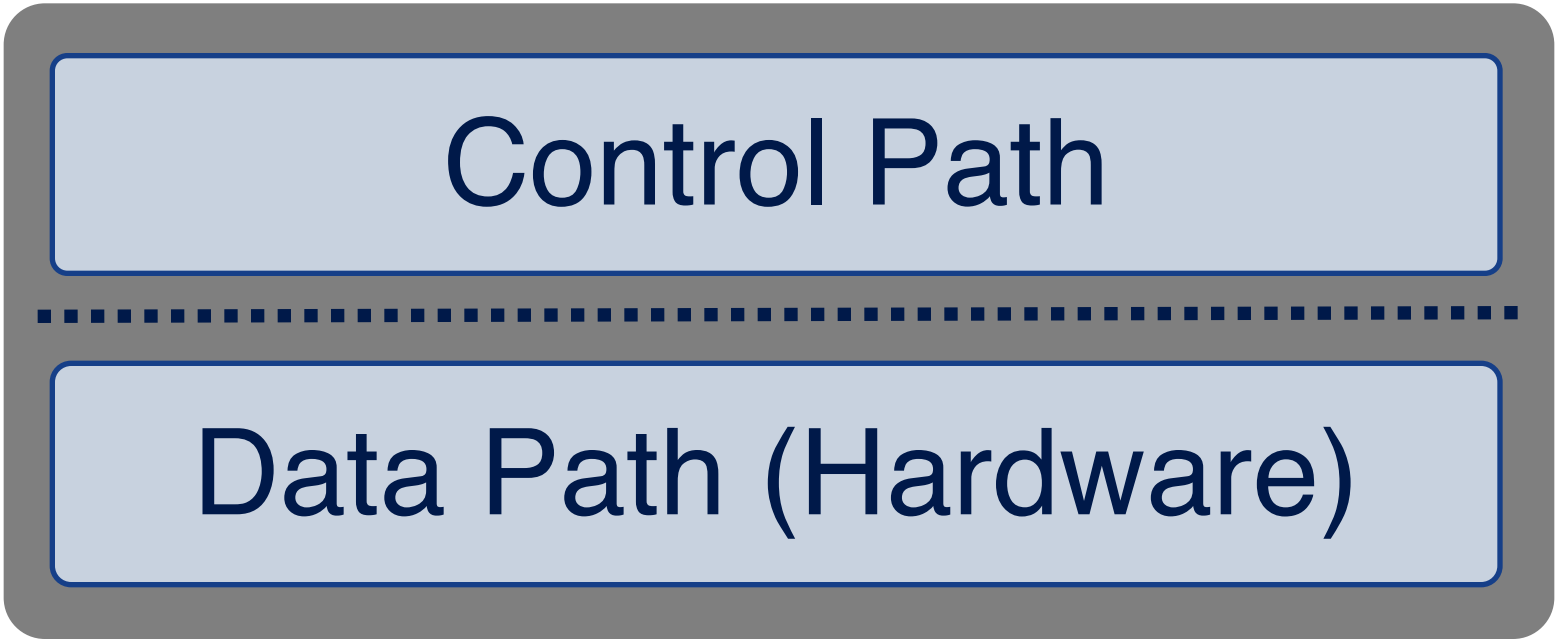


Logical Division in Networking

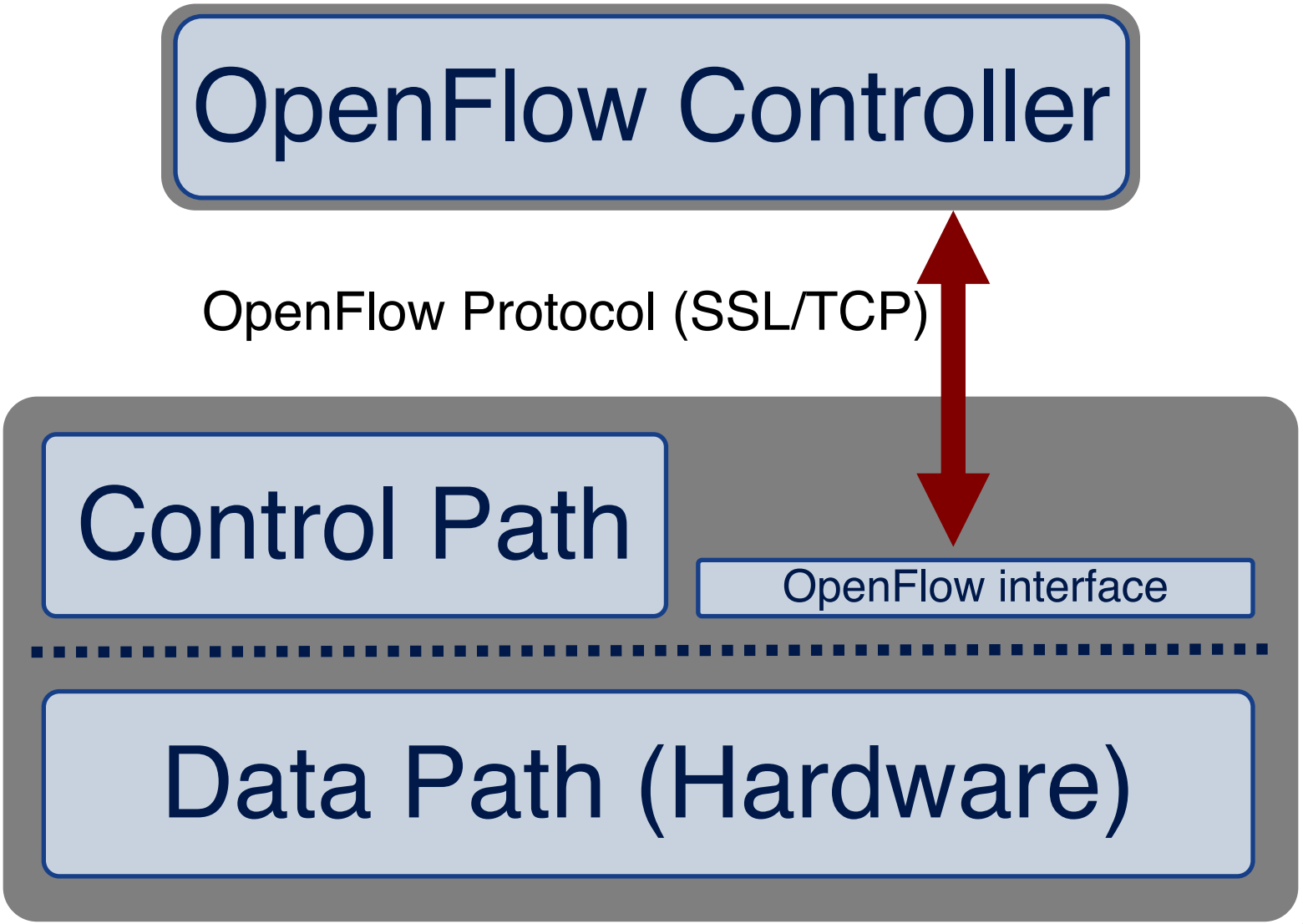


Can someone explain what this means?

Traditional switch



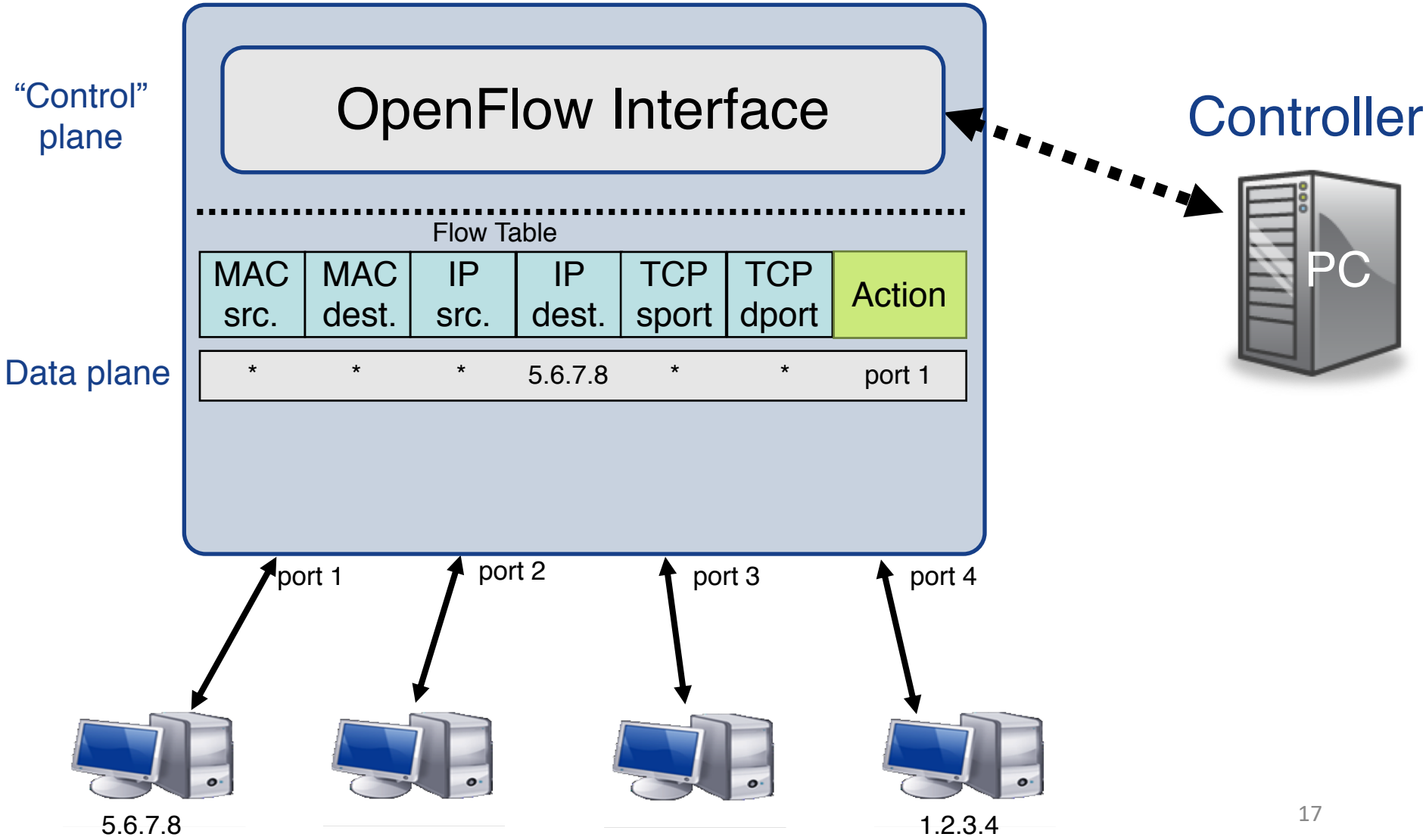
Openflow switch



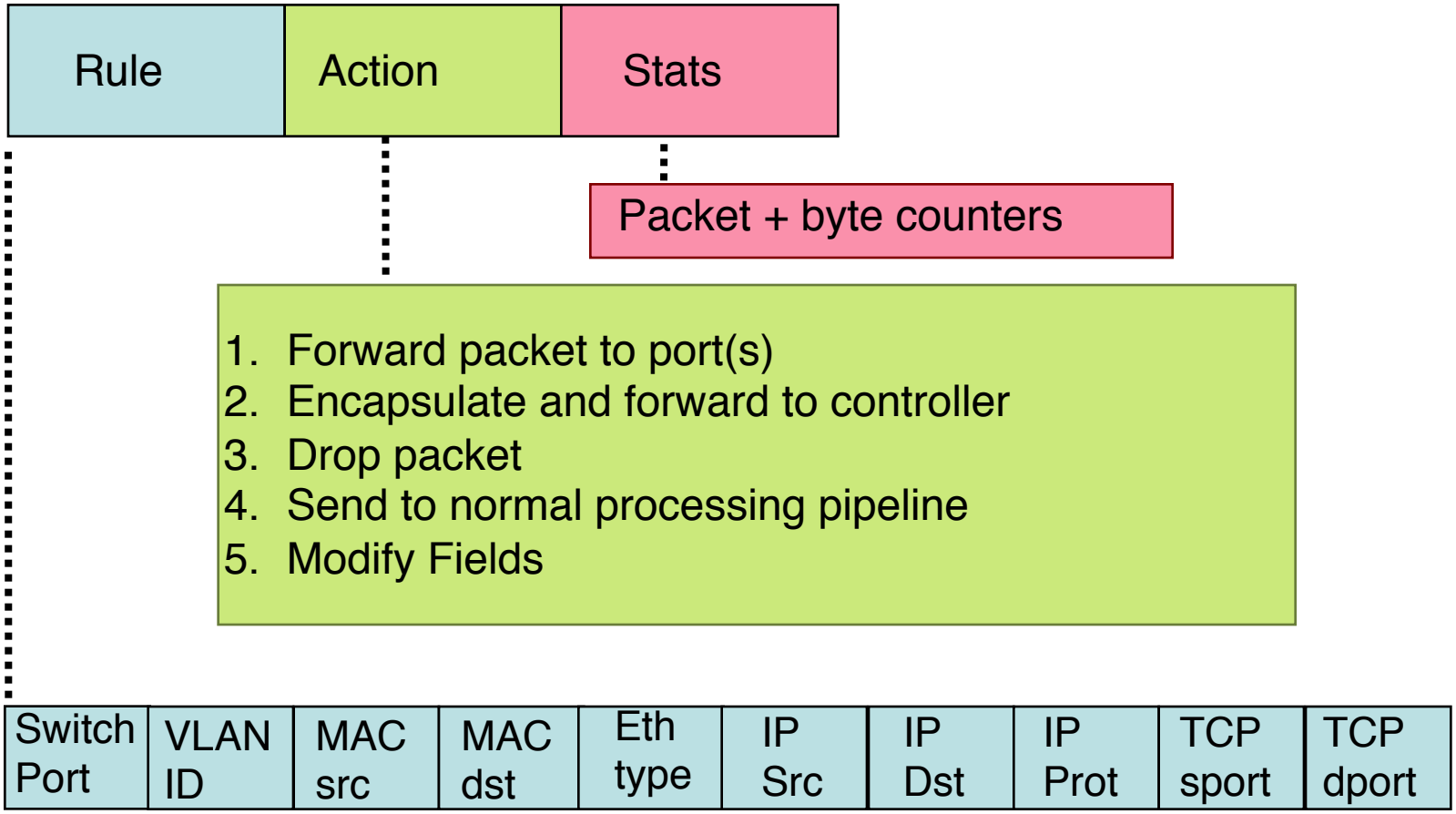
OpenFlow principles

- A traditional switch implements:
  - The **data plane** logic to forward frames
  - The **control plane** logic to decide how to forward frames (forwarding tables, spanning tree, etc.)
- An OpenFlow switch implements:
  - A somewhat **flexible data plane logic** supporting various types of rules
  - **No control plane** – all decisions are delegated to an OpenFlow controller
  - Typically OpenFlow controller installs **rules** in switches, so that it does not have to analyze every packet

OpenFlow model



Flow Table Entries



+ mask what fields to match

Examples

Switching

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

Flow Switching

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
port3	00:20..	00:1f..	0800	vlan1	1.2.3.4	5.6.7.8	4	17264	80	port6

Firewall

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	*	*	*	*	22	drop

Examples

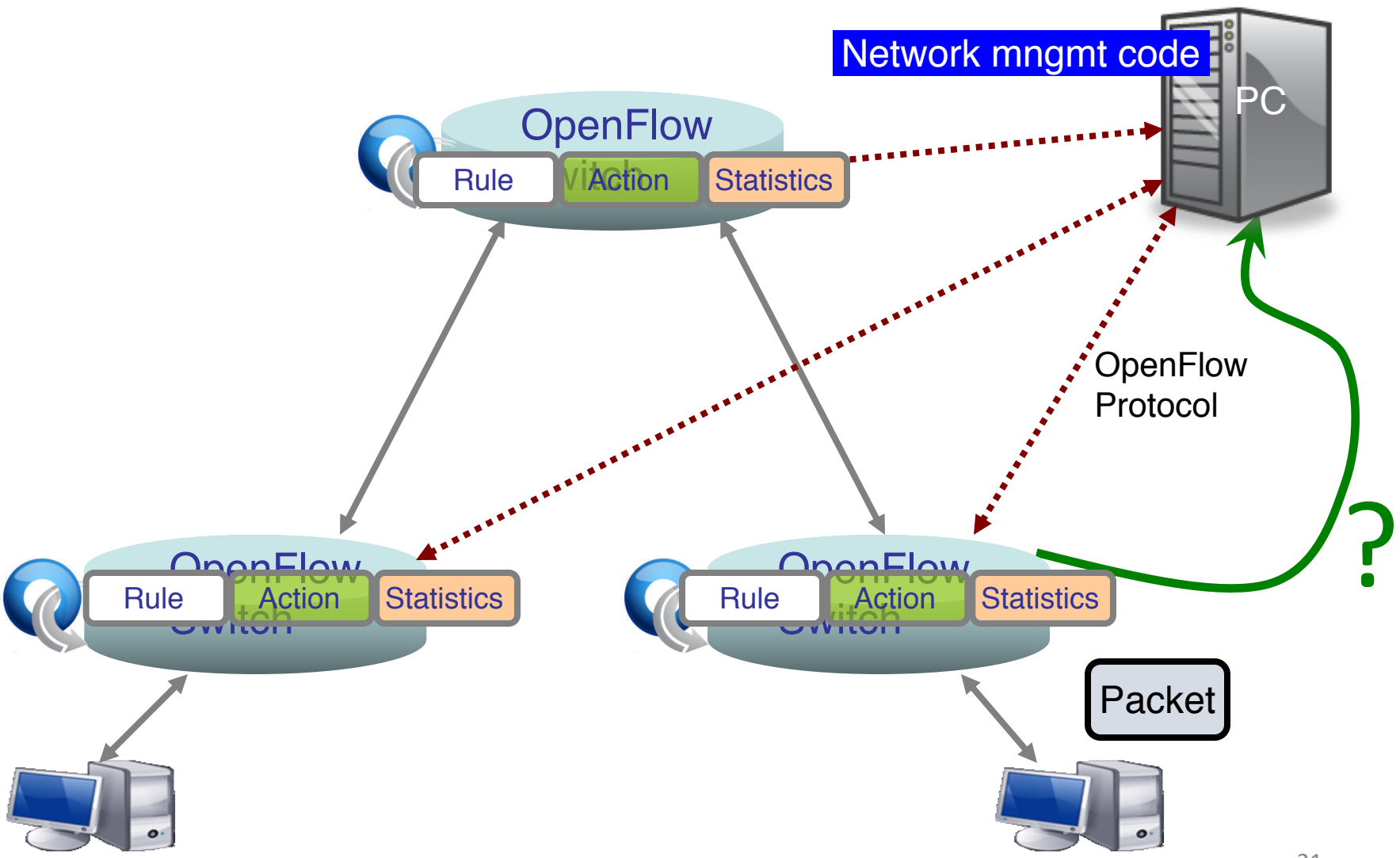
Routing

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

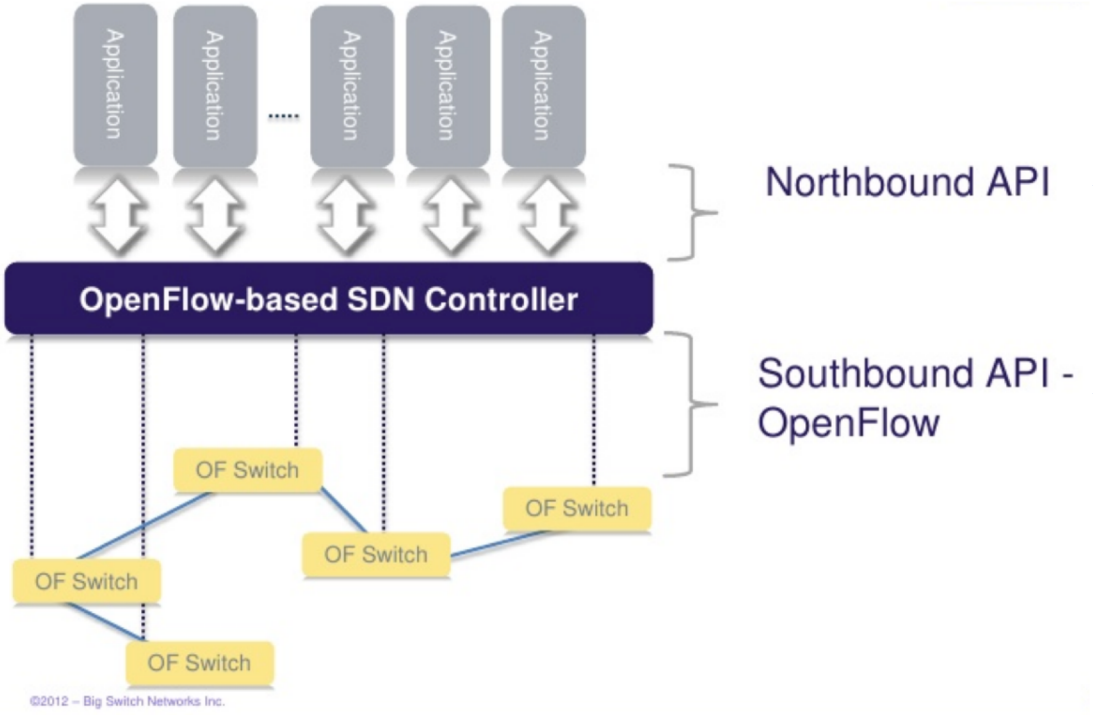
VLAN Switching

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 Example



Some terminology



Northbound API ← Configure network policies

Southbound API - OpenFlow ← Configure match/action table entries

(Source: Floodlight/Big Switch Networks)

Summarizing

- General principles
  - **Controller** can install **rules in switches**. **Rules** can match various **header fields**. A **rule** may entail **forwarding** packets to another switch/host, **dropping** packets, or **forwarding** packets to the controller
  - Typically, **any packet that does not match a rule is forwarded to the controller**

Lots of flexibility!

- Fine vs. coarse grained rules
- Reactive vs. Proactive rule creation
- **There is no “default” or “best-practice” way to use OpenFlow!**

Flow Routing vs. Aggregation

Both models are possible with OpenFlow

Flow-Based

- Every flow is individually set up by controller
- Exact-match flow entries
- Flow table contains one entry per flow
- Good for fine grain control, e.g. campus networks

Aggregated

- One flow entry covers large groups of flows
- Wildcard flow entries
- Flow table contains one entry per category of flows
- Good for large number of flows, e.g. backbone

Reactive vs. Proactive

Both models are possible with OpenFlow

Reactive

- First packet of flow triggers controller to insert flow entries
- Efficient use of flow table
- Every flow incurs small additional flow setup time
- If control connection lost, switch has limited utility

Proactive

- Controller pre-populates flow table in switch
- Zero additional flow setup time
- Loss of control connection does not disrupt traffic
- Essentially requires aggregated (wildcard) rules

OpenFlow Application: Network Slicing

- Divide the production network into **logical slices**
  - each slice/service controls **its own packet forwarding**
  - users pick **which slice controls their traffic**: opt-in
  - existing production services run in their own slice
    - e.g., Spanning tree, OSPF/BGP
- Enforce **strong isolation between slices**
  - actions in one slice do not affect another
- Allows the (logical) testbed to **mirror the production network**
  - real hardware, performance, topologies, scale, users

Activity time!

- Form groups of 4/5 people
- Prepare a list of possible applications of OpenFlow (i.e., what can I do in an OpenFlow network that I could not do in a traditional network?)
  - You are encouraged to think beyond what discussed in the paper ☺

OpenFlow beyond academia

Academia & beyond

- OpenFlow was conceived as a research oriented solution
- …but it has found application outside academia

Commercial implementations

- OpenFlow is the first ”mainstream” implementation of an idea called **software-defined networking (SDN)**
- Various SDN startups since 2007:
  - Nicira
  - Big Switch Networks
  - Embrane
  - …
- **Takeway:** the ideas behind OpenFlow are relevant well beyond the academic world

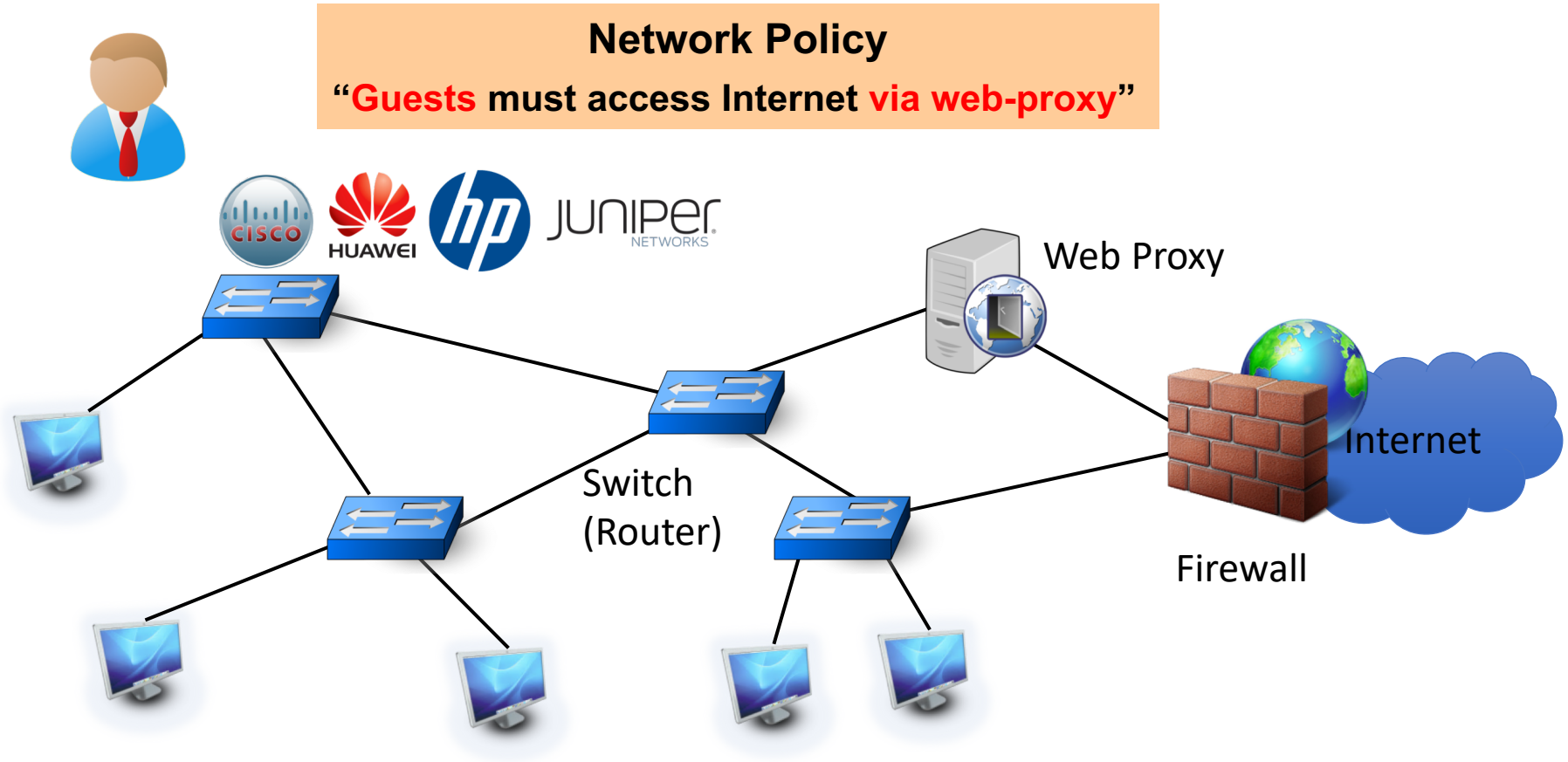
So, what's the **real** point of SDN?

SDN as a solution for network management

- SDN is relevant because it simplifies various tasks in **network management**

Introduction

- Network management is driven by policy requirements



Option1: Command-line Interface (CLI)

- Vendor-specific, low-level
- Script needs frequent maintenance


```
(Huawei)

Switch# display interface ethernet 1/0/0.1
Ethernet1/0/0.1 current state : UP
Line protocol current state : UP
Description : HUAWEI, AR Series, Ethernet1/0/0.1 Interface
.....
```

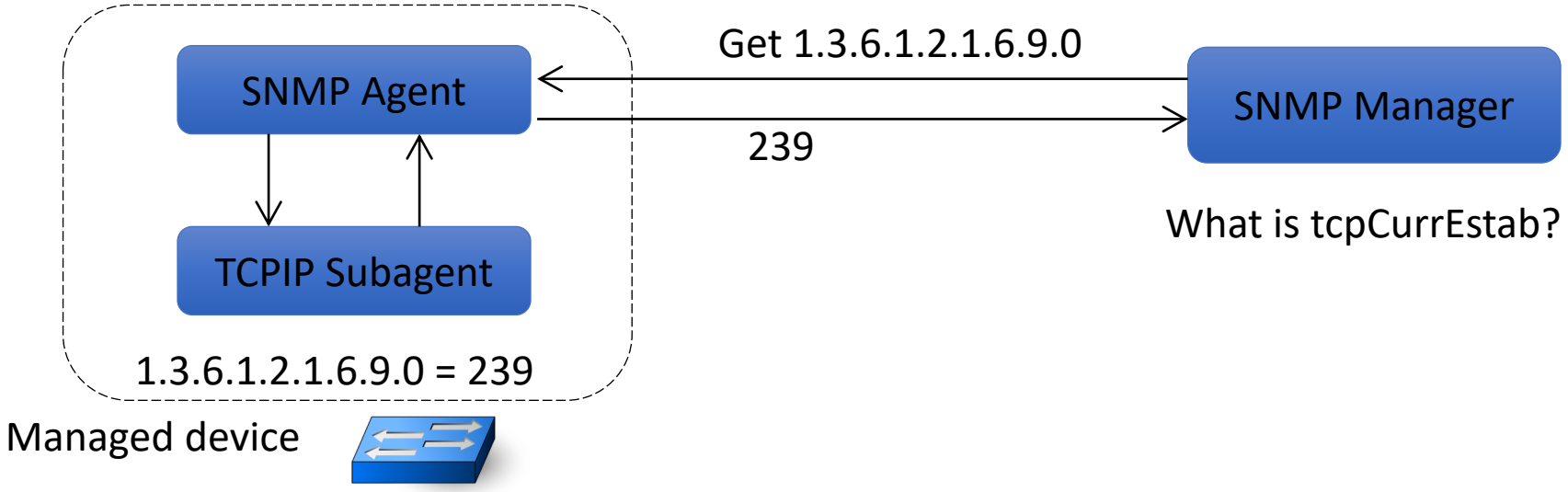
```
(Cisco)

Switch# show interfaces ethernet 0
 Ethernet0 is up, line protocol is up
   Hardware is Lance, address is 0060.3ef1.702b (bia 0060.3ef1.702b)
   Internet address is 172.21.102.33/24
.....
```

What could go wrong?

Option 2: Simple Network Management Protocol (SNMP)

- A framework that provides facilities for managing or monitoring network devices in IP networks
  - Query/Set provided variables in devices



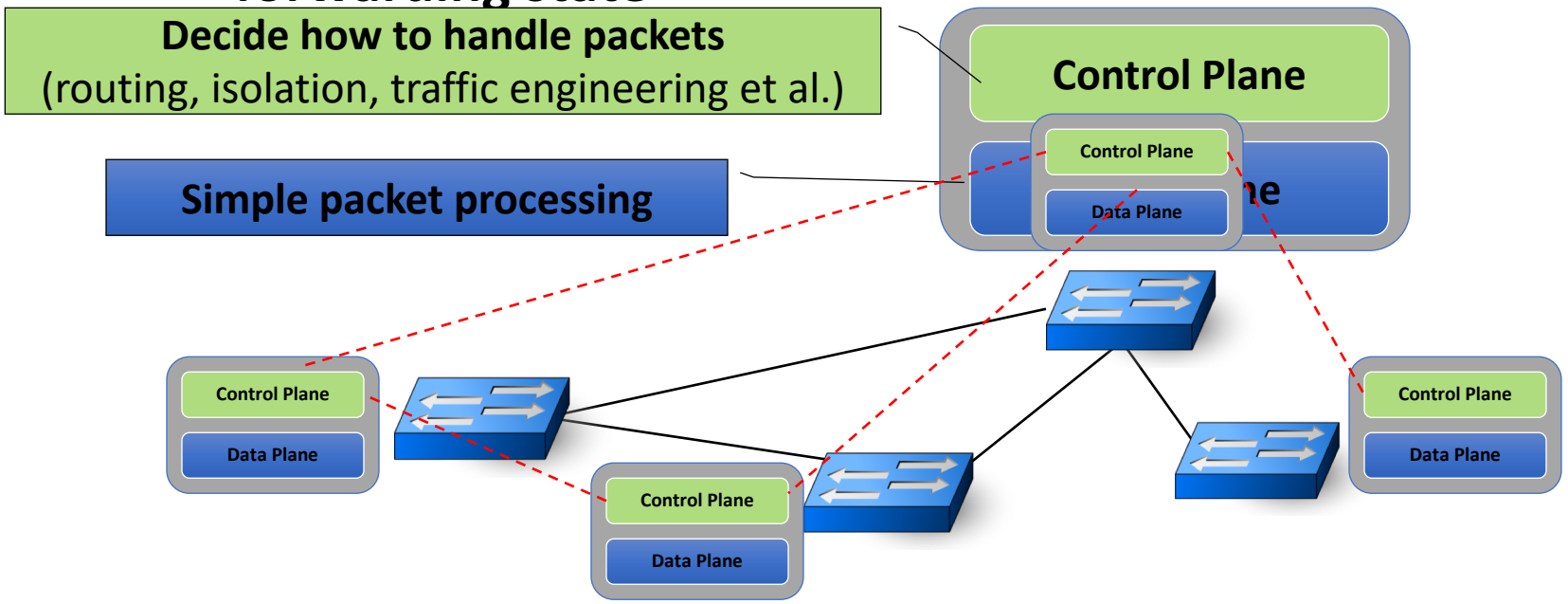
- Potential issues?

Network Management is Challenging

- Current networks don't have good management approaches
  - CLI: vendor-specific, low-level
  - SNMP: limited capabilities
  - Third party management tools: bind to software vendor
- Today's management requires
  - Easy configuration, fine-grained control, and react to continually changing network conditions timely
- Traditional approaches cannot satisfy today's management requirements

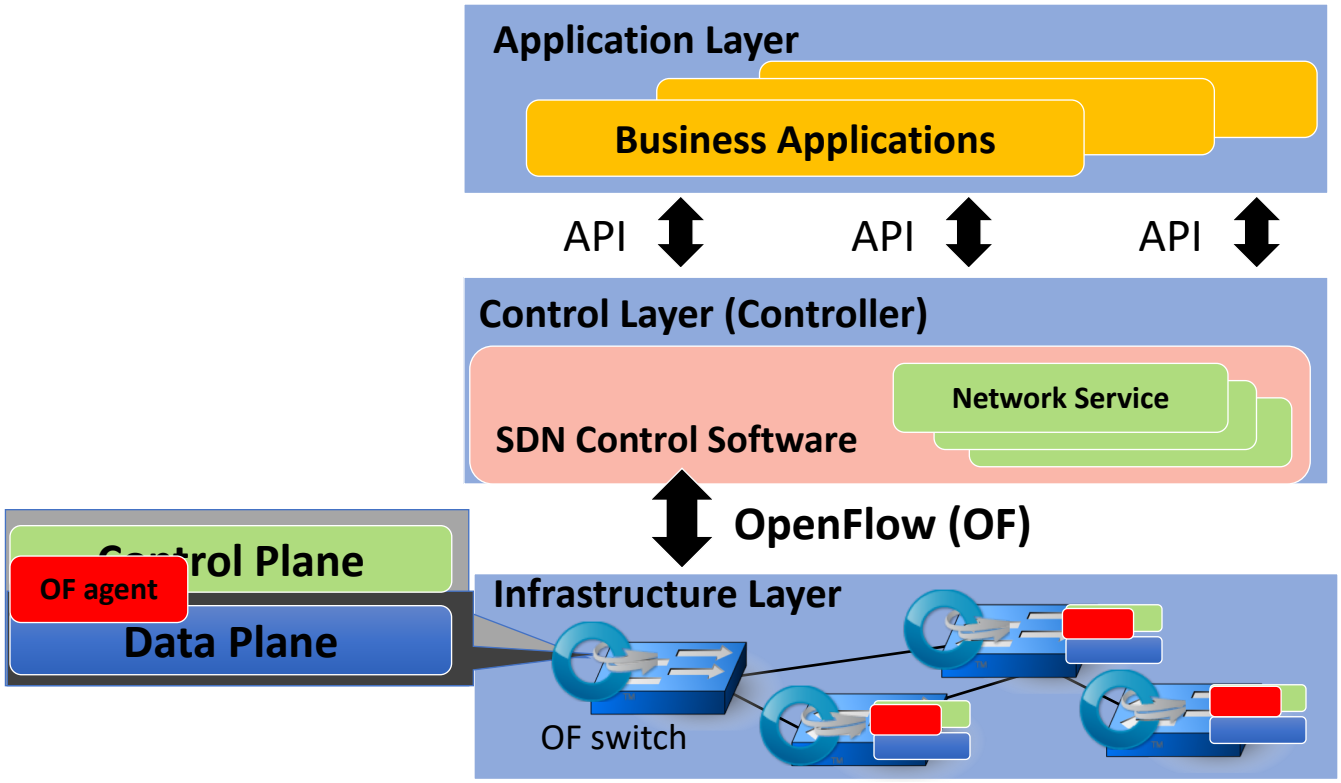
What Is the Root Cause for the Complex Network Management?

- Control plane has **no good abstractions**
  - Okay for a single device, **not good for the whole network**
  - **No modularity** and **no standard way** of defining **forwarding state**

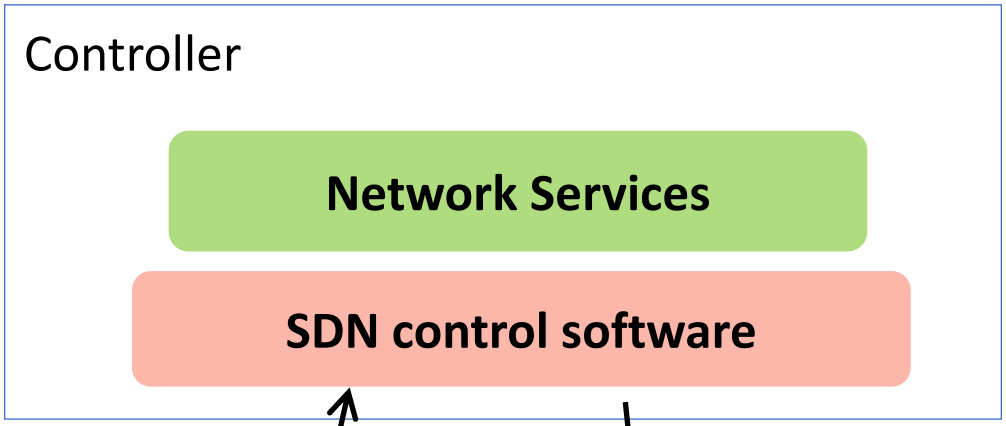


Software Defined Networking (SDN) Architecture

- A network architecture where network control is decoupled from forwarding and is directly programmable



Programmable Control Plane



Events from switches
- Topology changes
- Traffic statistics
- Arriving packets

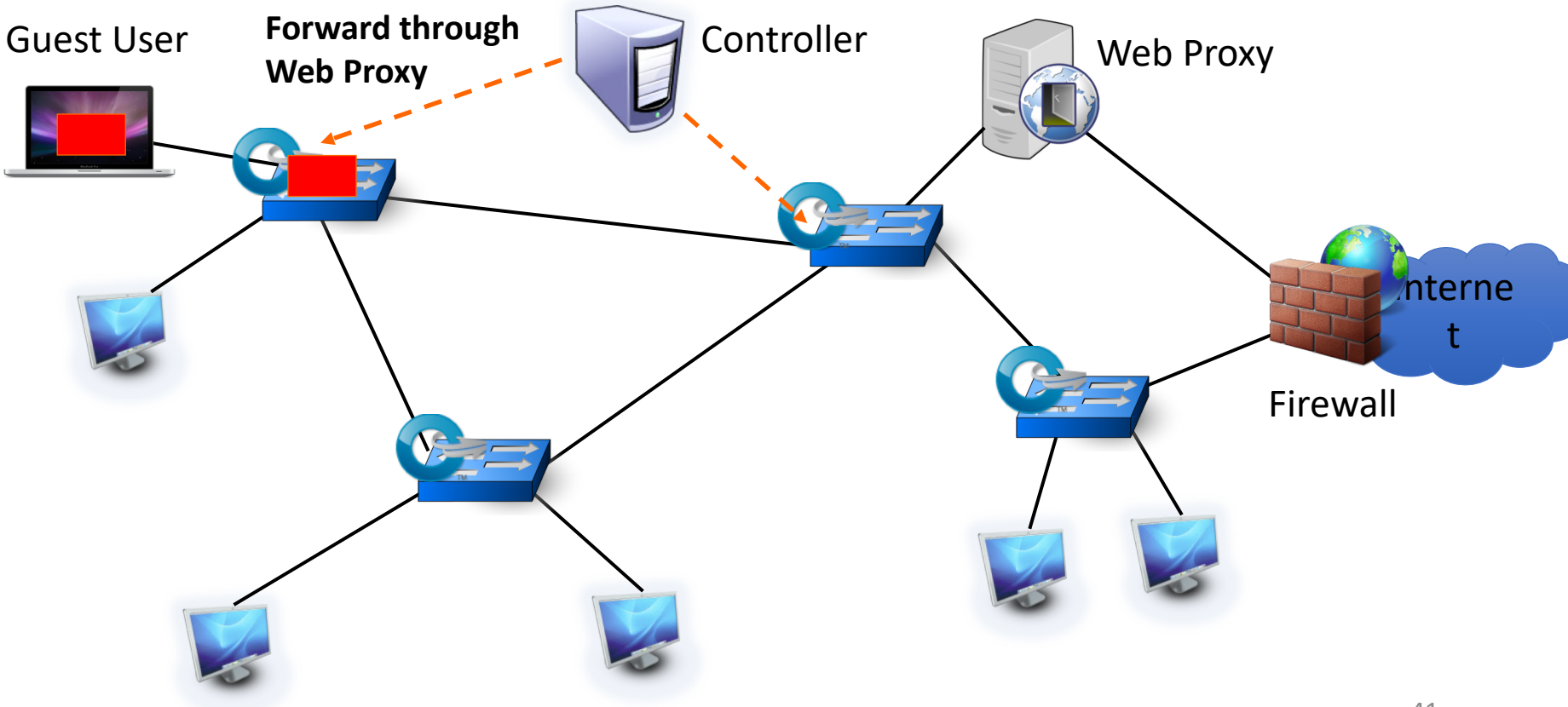
Commands to switches
- (un)install rules
- Query statistics
- Send packets

[1] OpenFlow: enabling innovation in campus networks. (SigComm CCR'08)

Standard SDN Workflow

Network Policy

"Guests must access Interest via web-proxy"



[2] Rethinking Enterprise Network Control, TON'09

Advantages of SDN

- Decouple control plane and data plane
  - The software control of the network can evolve independently of the hardware
- Logically centralized network control
  - Network-wide view, more deterministic, more efficient

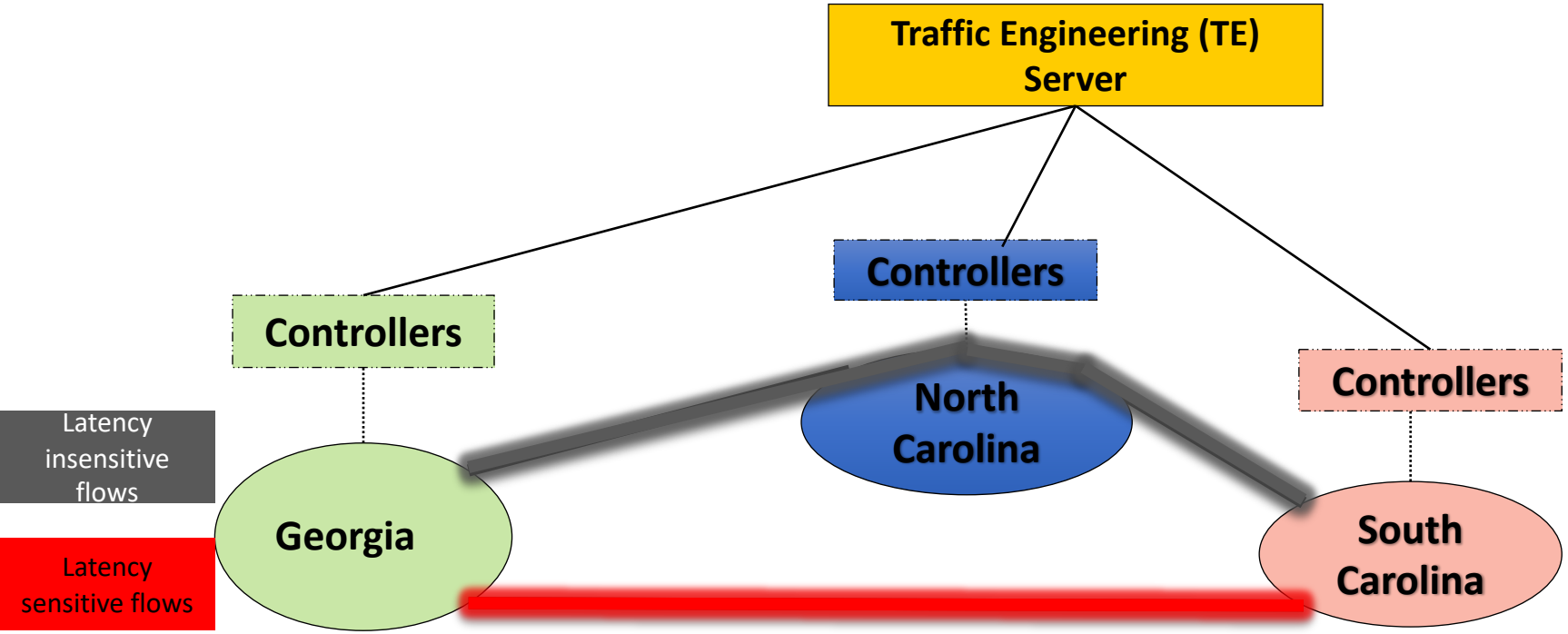
A SDN Use Case

- Google inter-datacenters WAN (B4) is the first and largest SDN/OpenFlow deployment

- Overprovisioning introduces low link utilization (30-40%)

[3] B4: Experience with a globally-deployed software defined WAN. SigComm'13

SDN Improves Link Utilization



- SDN drives link utilization to almost 100%
  - TE server assign flows to idle network links according to the application priority
  - Controllers enforce the assignment and control bursts

Network Function Virtualization

- Take a physical device's actions and convert it to a VM performing the same tasks
  - Physical intrusion detection system becomes VM inspecting traffic
  - Physical firewall becomes VM with iptables
  - …
- VM-based services can be scaled, deployed as needed etc.
- **SDN simplifies deployment of NFV (why?)**

SDN Tools

- OpenFlow controllers:
  - Floodlight, OpenDayLight (Java)
  - Nox (C++)
  - Pox (Python)
  - …
- Switches:
  - Open vSwitch (virtual software switch w/ OpenFlow support)
  - Hardware switches from various vendors

Some limitations of software-defined networking

Scalability Issues of SDN

- **Flow-level granularity** places **significant stress** on switch state size and controller involvement
- A switch can support a **limited number of OpenFlow entries**
  - OpenFlow rules are stored in TCAM (only support ~1500 entries in HP 5406zl switch)
  - Ethernet forwarding uses hash lookup in standard memory (support ~64k entries)
- Limited computation resources on controller
  - NOX controller can handle 30k flows/s
  - Datacenters with 100 edge-switches may have more than 10M flows/s

[4] DevoFlow: scaling flow management for high-performance networks, SigComm'11

Scalability Issues of SDN (cont.)

- Flow setup delay
  - Forward the initial packet: 5us on ASIC vs. 5ms if controller involved
  - **Unacceptable flow setup delay** for high performance networks

- **Frequent statistics-pulling** reduces the **flow setup rate**
  - Tasks (e.g. traffic engineering) need statistics gathering
  - 275 connections/s (no pull) vs. 50 connections/s (5 requests/s) using HP 5406zl switch

[4] DevoFlow: scaling flow management for high-performance networks, SigComm'11

Reliability issues of SDN

- **Let's hear some opinions first**
- Most significant issue: single-point-of-failure
  - The controller is responsible for managing the entire network
  - What if:
    - It cannot sustain the rate of requests from switches?
    - It is attacked?
    - …or just crashes?

Some possible solutions

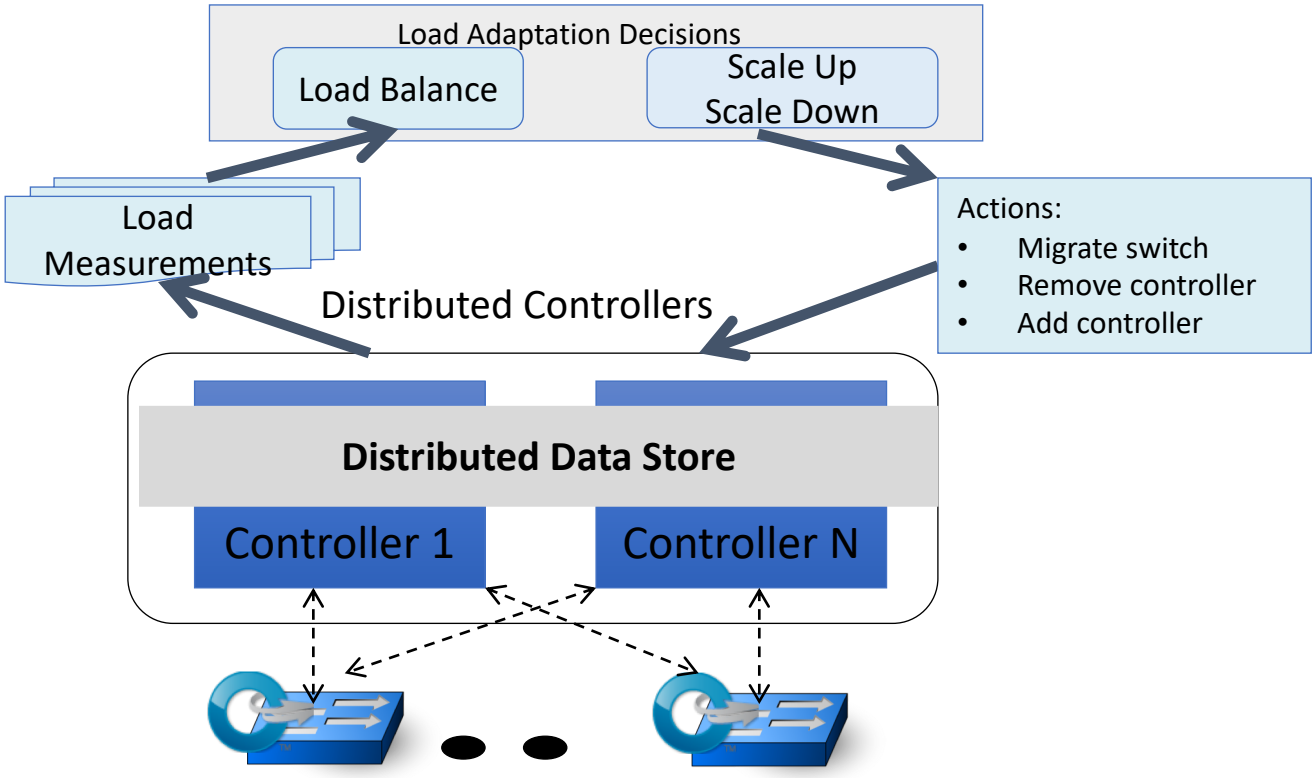
Scaling Control-plane

- Approaches in this category aim to **improve the limited computation resources on control layer**
  - Use better hardware, make it distributed, etc.

- **Distributed system** is better than centralized system
  - Better fault tolerance
  - Better scalability
  - Better load balancing

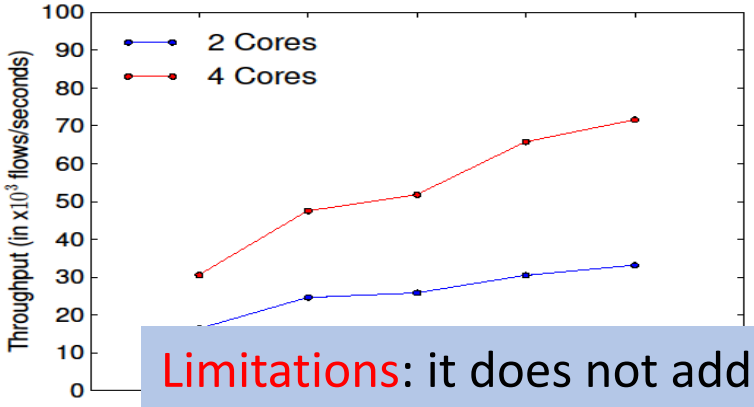
Controller replication: ElastiCon

- **Make central controller a controller pool**
- Dynamically adds/removes controllers according to traffic conditions



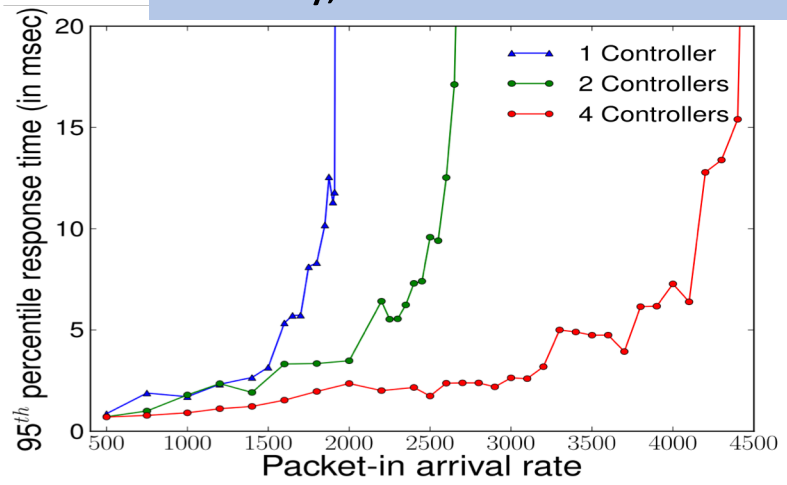
[5] ElastiCon: An Elastic Distributed SDN Controller, ANCS'14

Evaluation



Adding controller nodes increases the throughput

Limitations: it does not address the issue of flow setup delay directly, and introduces another layer of complexity



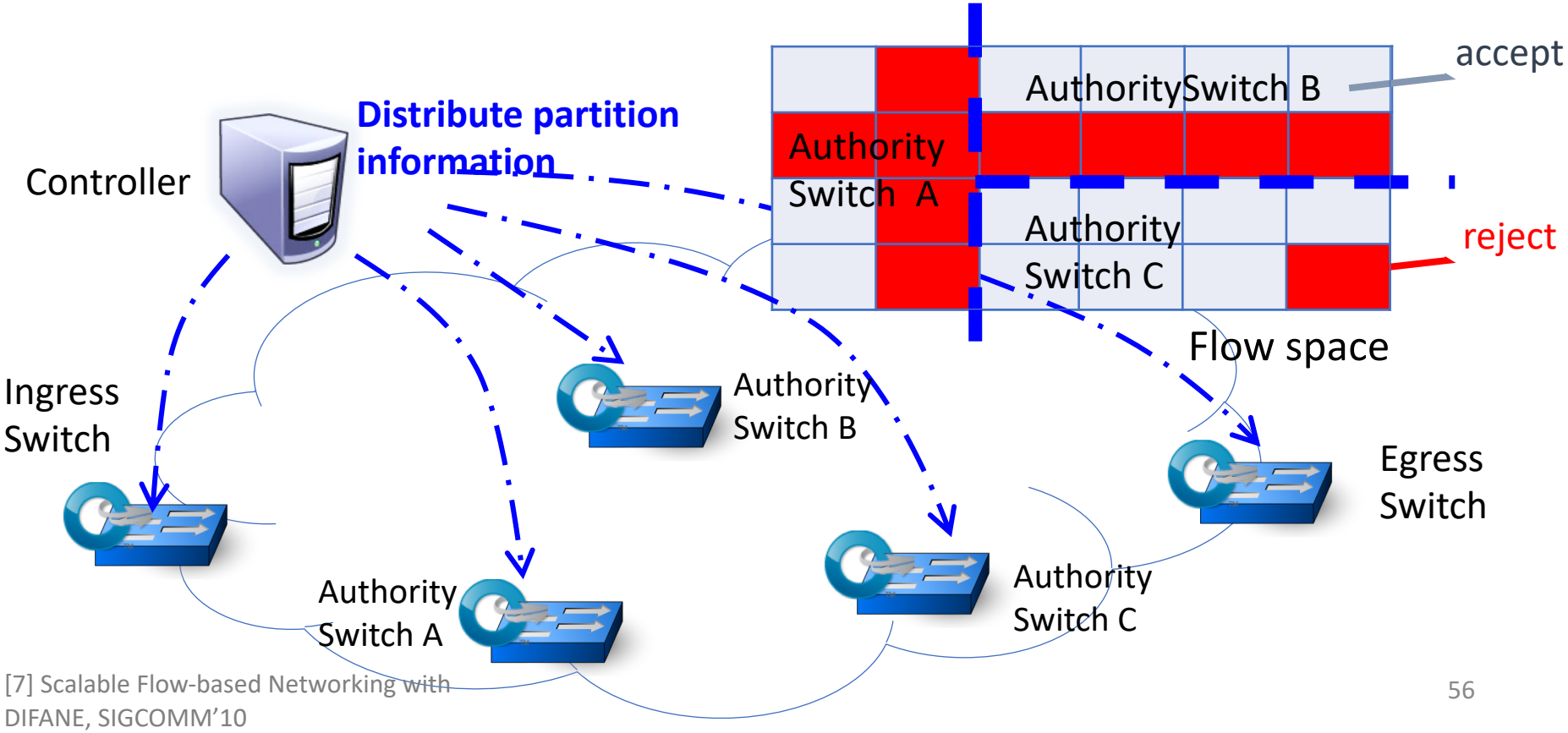
More controllers can make the response time stay in a certain level

Making the controller less critical

- **Delegation:** transfer some responsibility from controller to switches by making the switches "smarter"
- **Hybrid** between traditional network management and SDN

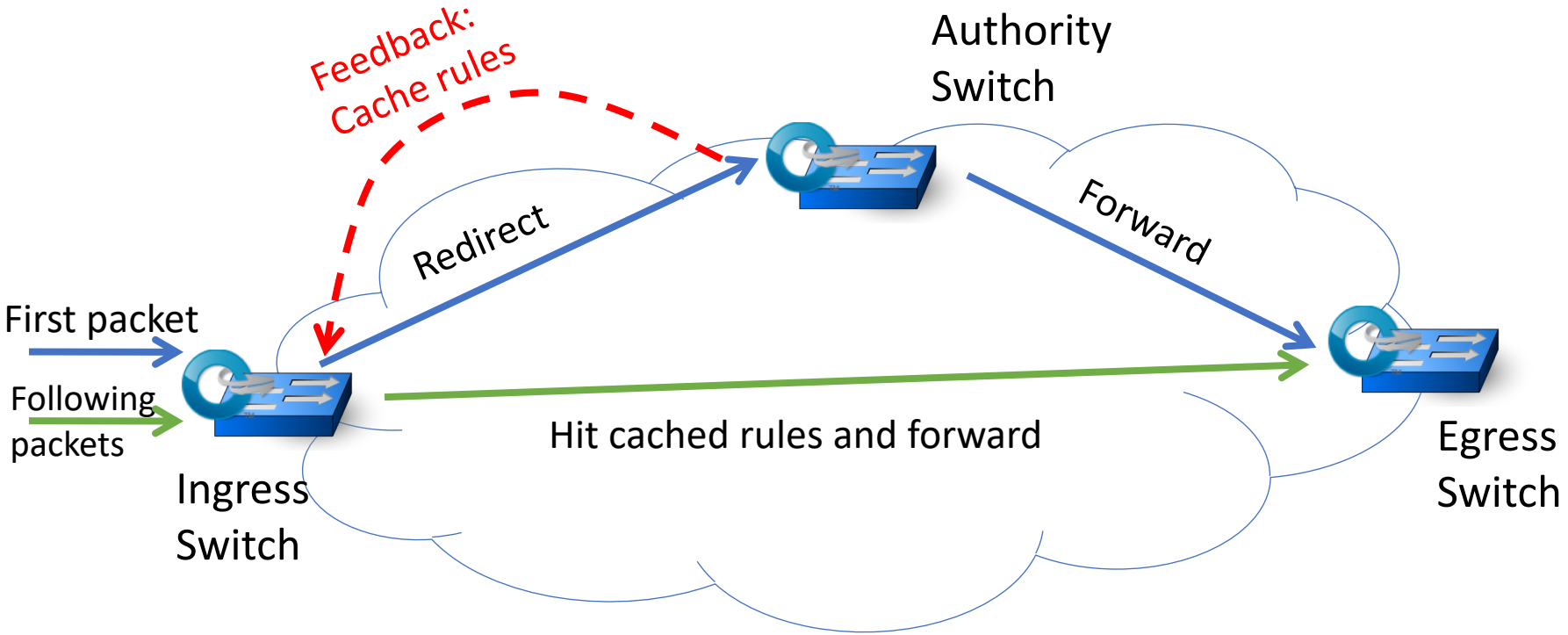
DIFANE

- Controller delegates some switches to handle real-time jobs
  - Authority switches with larger TCAM space and processing capability
- Controller generates the rules, and installs them on switches



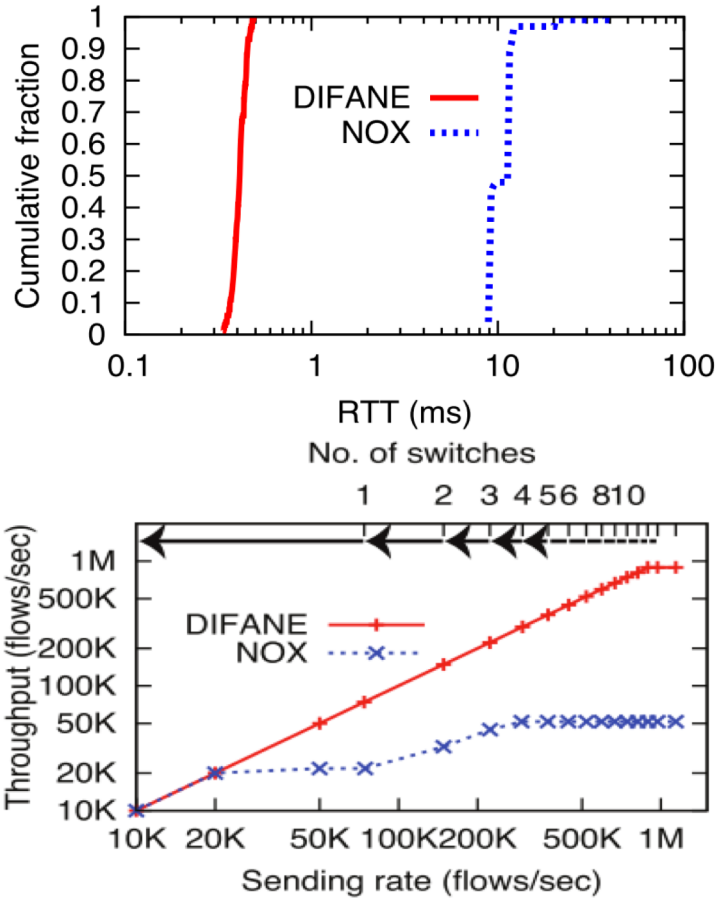
[7] Scalable Flow-based Networking with DIFANE, SIGCOMM'10

DIFANE (cont.)


Feedback:
Cache rules
Authority
Switch
Redirect
Forward
First packet
Following
packets
Hit cached rules and forward
Ingress
Switch
Egress
Switch

DIFANE (cont.)

- DIFANE outperforms NOX …



DIFANE achieves small delay for first packet (0.4ms vs 10ms)

Throughput in DIFANE increases linearly with the number of switches.

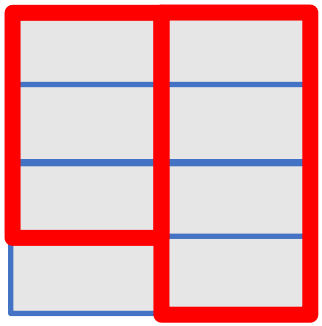
What about switch memory limitations?

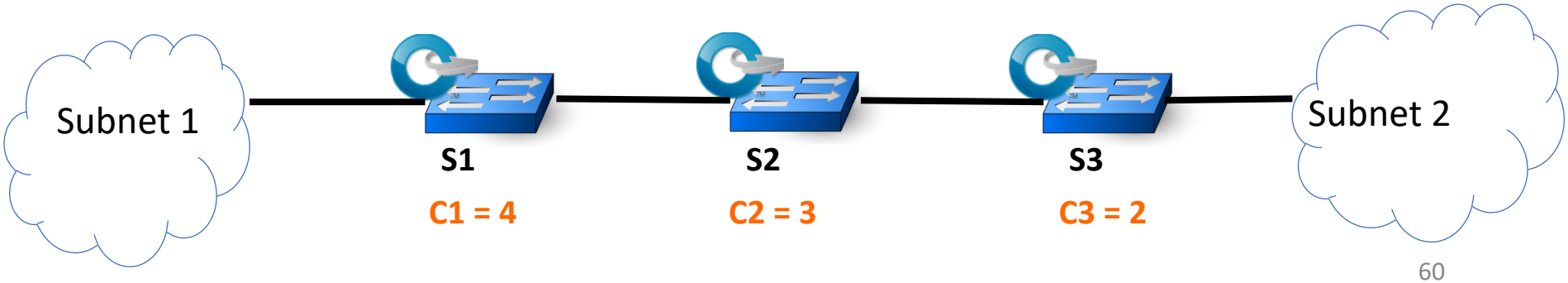
- **Problem:** switch can only store a limited number of rules
  - What if the network policy requires more?
- **Possible solution:** only install rules on a per-needed basis
  - Generates a lot of work for controller
- **Alternate solution:** optimize rule placement
  - SDN controller naturally has the network wide view
  - Controller computes optimal placement of rules in switches' TCAMs
  - Advantages: less controller involvement, avoid TCAM space exhaustion
  - One big switch [8] abstraction

[8] Optimizing the "One Big Switch" Abstraction in Software-Defined Networks. CoNEXT'13

One Big Switch

- Controller installs rules along the path while obeys the table size constraints
  - Assign subset of the endpoint rules (e.g. ACL) to a routing path
  - Place part of the rules on each switch along the path according to the table size





One Big Switch (cont.)

- Synthetic 100-node topologies with real firewall configurations
- More than 60% unwanted traffic are dropped in the first 20% of the path
- **Limitations**
  - Introduce unwanted traffic
  - Statistic gathering is slow

Conclusions

- Lots of hope when OpenFlow came out (~2008)
- Some of those hopes have materialized
  - See ”success stories” like Google’s
- **Lots of tricky issues**
- Vendors initially not very enthusiastic, then some support
  - Vendor-specific solutions integrating some SDN ideas
  - New buzzword: **policy-based networking**