Advanced Network Programming

Software Defined Networking

Lin Wang

Fall 2020, Period 1

VU VRIJE UNIVERSITEIT AMSTERDAM

Part of the content is adapted from Scott Shenker

Part 2: network infrastructure

Lecture 7: Network forwarding and routing

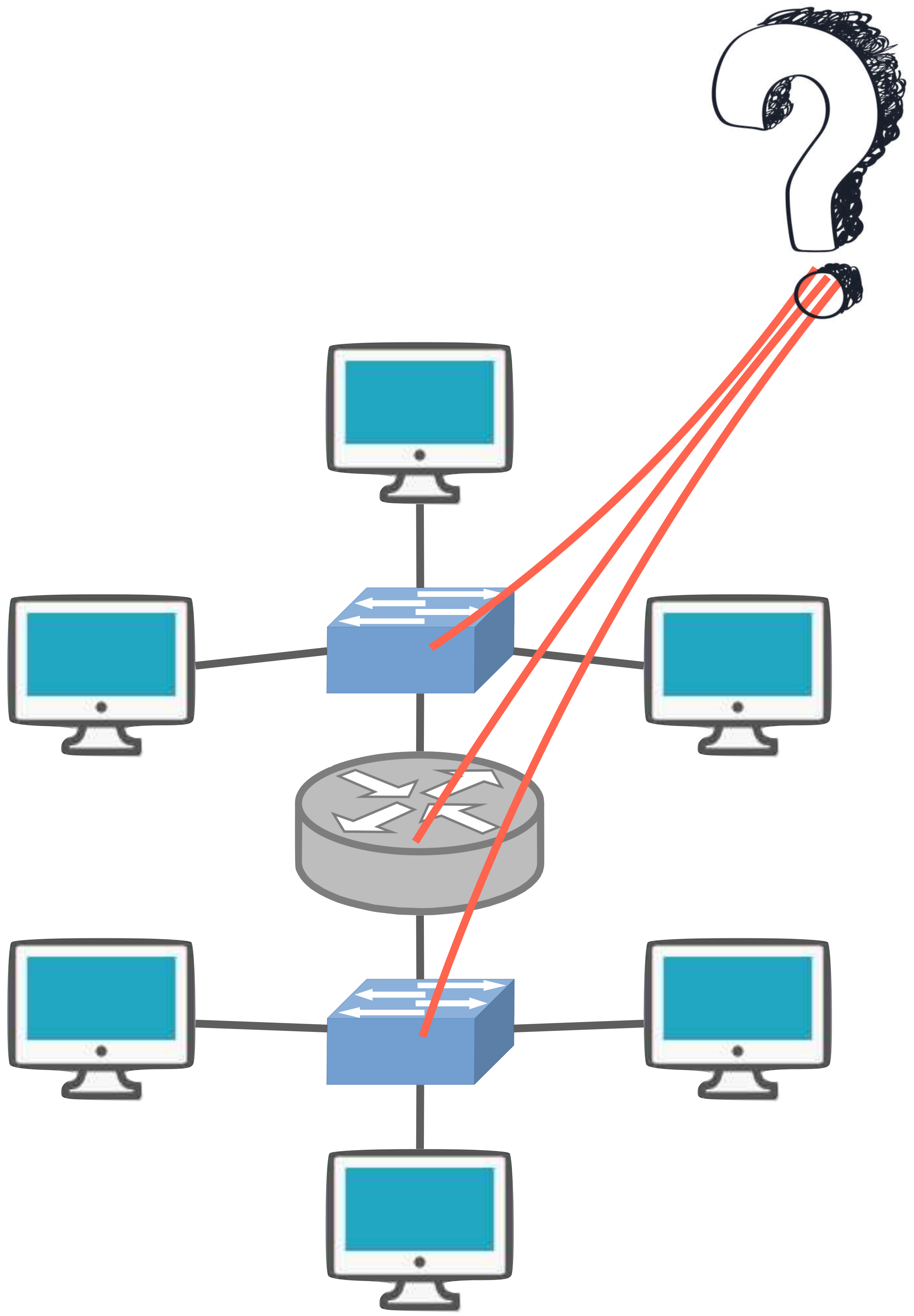
Lecture 8: Software defined networking

- Complexity in network control and management
- Centralized control, OpenFlow
- Network virtualization, FlowVisor

Lecture 9: Programmable data plane

Lecture 10: Cloud networking

Lecture 11: Beyond networking

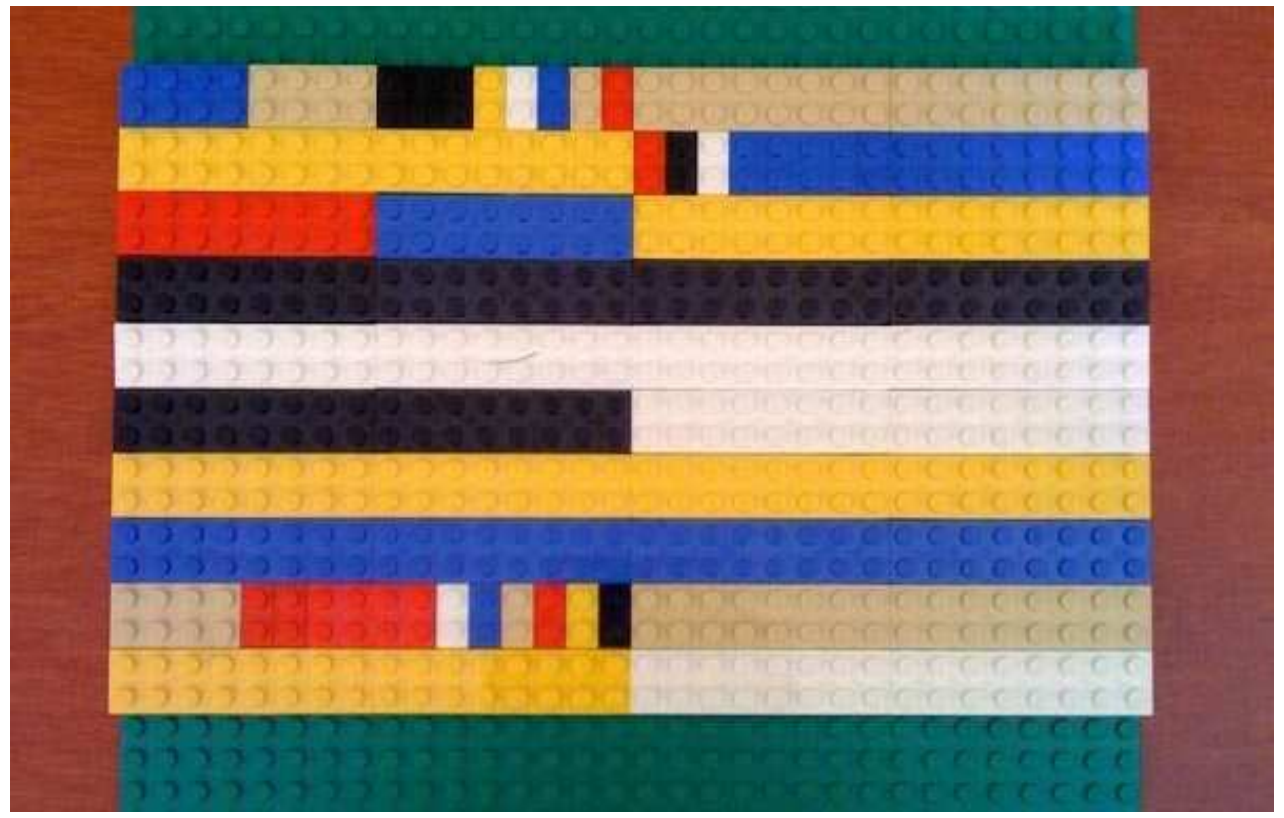


What is your impression about computer networking so far?

A plethora of protocol acronyms

DCCP
FTP
DNS ARP
Infrared
NAT ICMP
L2TP
HTTP
UDP
IPX
POP3 SCTP IP
IRC
ATM
WiFi
100BASE-T VLAN
TCP
PPTP
DASH DSL
SMTP

A heap of header formats



```
 0      7 8     15 16    23 24    31
+--------+--------+--------+--------+
|          source address           |
+--------+--------+--------+--------+
|        destination address        |
+--------+--------+--------+--------+
|  zero  |protocol|   UDP length    |
+--------+--------+--------+--------+
```

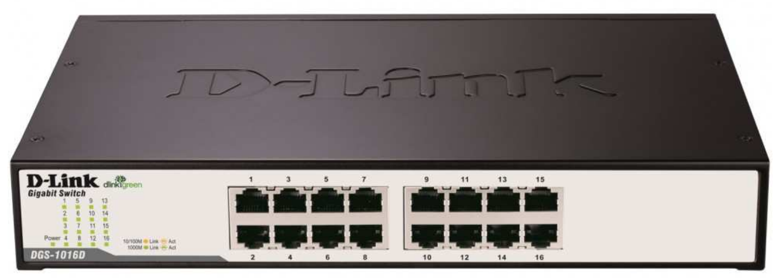
```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Acknowledgment Number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data |           |U|A|P|R|S|F|                               |
| Offset| Reserved  |R|C|S|S|Y|I|            Window             |
|       |           |G|K|H|T|N|N|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |         Urgent Pointer        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

A bunch of boxes

Access Point

Router



Switch

Middlebox (Firewall, Load Balancer, IDS)

A ton of tools

ping

tcpdump

traceroute

iperf

wget

rancid

nmap

dig

nslookup

ntop

wireshark

syslog

whois

Why are there so many artifacts?

Complexity in networking

We need **different functionalities**, also new ones

- Different physical layers and applications, traffic engineering, congestion control, security

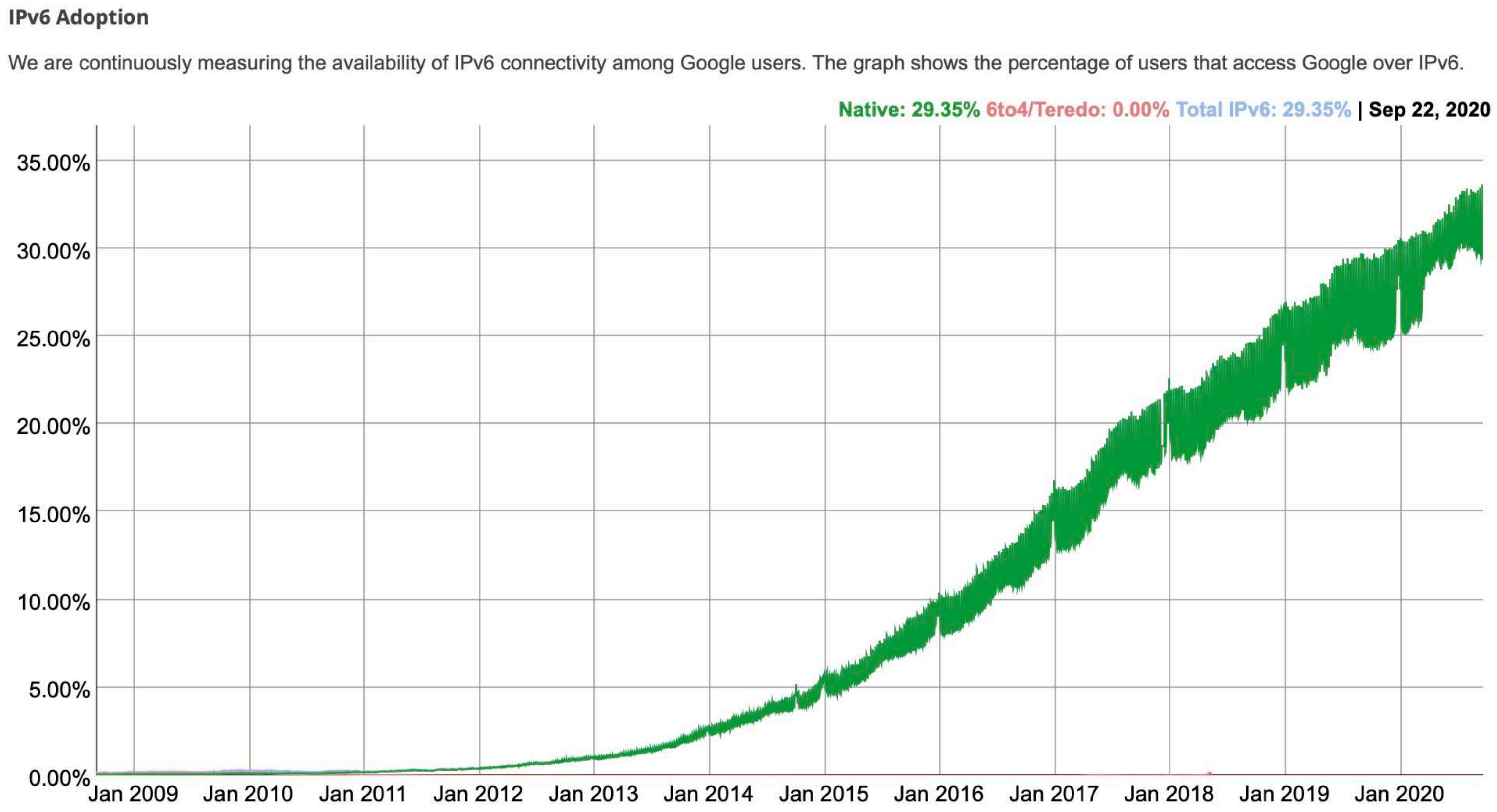
Networks run in a **distributed**, **autonomous** way

- Scalability is important

All these add to **complexity**, innovations are active in academia, but suffer from poor adoption of deployment

- Example: IPv6
- Deadlock between innovation and adoption

RFC 2460 (1998)



https://www.google.com/intl/en/ipv6/statistics.html#tab=ipv6-adoption

Once upon a time

AT&T updates their internal network infrastructure (routers and switches) **every 18 months** to keep up with the current demands for network

One upgrade requires **billion of USD**

- A Cisco top of line switch costs $27K USD
- Significant **manpower** is needed to upgrade the network

THE WALL STREET JOURNAL.

English Edition ▾ | Print Edition | Video | Podcasts | Latest Headlines

Home World U.S. Politics Economy Business Tech Markets Opinion Life & Arts Real Estate WSJ. Magazine

AT&T Targets Flexibility, Cost Savings With New Network Design

Move Could Cut the Company's Capital Costs by Billions of Dollars

By Thomas Gryta
Updated Feb. 24, 2014 12:25 pm ET

PRINT TEXT

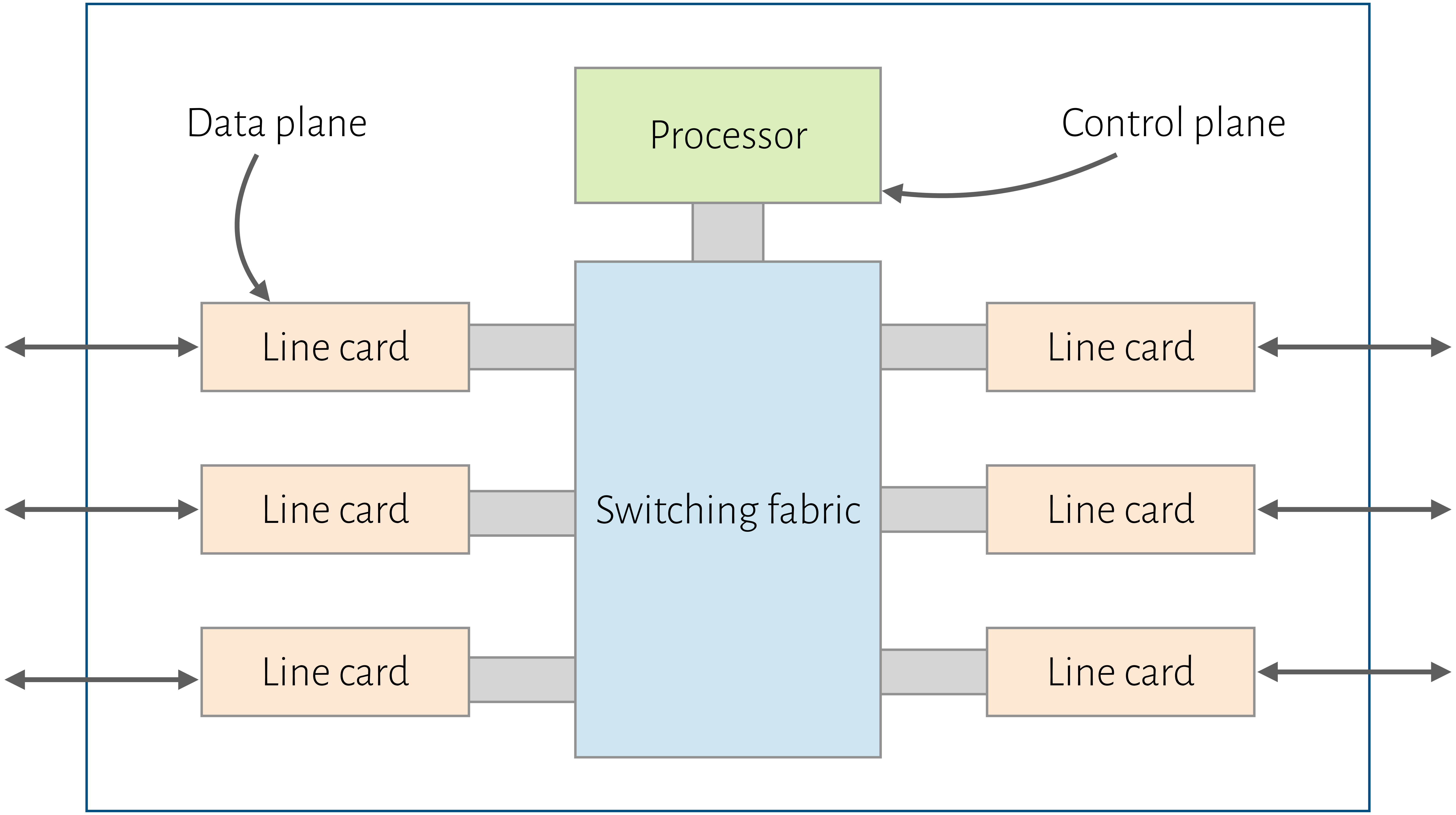
AT&T Inc. is planning to rebuild its sprawling network with less expensive, off-the-shelf equipment controlled by software, a move that could cut its capital costs by billions of dollars and put further pressure on telecom gear makers.

The shift will mean the second-largest U.S. carrier will buy less specialized equipment from vendors such as Ericsson , Alcatel-Lucent SA and Cisco Systems Inc., and instead purchase more generic hardware from a wider variety of producers. That equipment will be tied together with software, making it easier and cheaper to upgrade to new technologies, roll out new services or respond to changes in demand for connectivity.

RECOMMENDED VIDEOS

1. Biden Takes Aim at Trump on Social Security
2. Gyms Brace for Fewer Members as At-Home Fitness Rises During Pandemic
3. Some Covid-19 Survivors Grapple With Large Medical Bills
4. The Rise of TikTok:

Complexity in network planes



Complexity in the control plane

Control plane needs to achieve goals such as connectivity, inter-domain policy, isolation, access control...

Currently, these goals are achieved by many mechanisms/protocols:

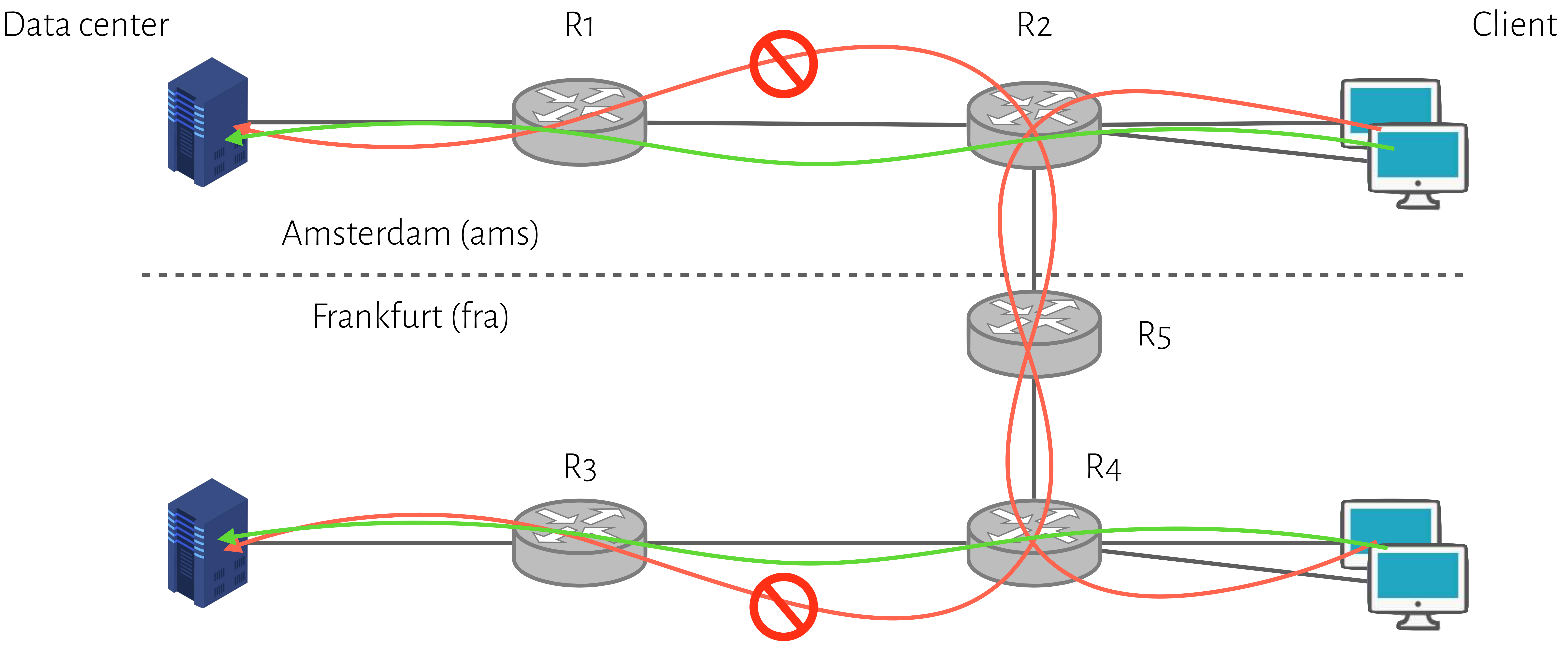
- Globally **distributed**: routing algorithms
- **Manual**/scripted configuration: Access Control Lists, VLANs
- **Centralized** computation: traffic engineering (indirect control)

Even worse, these mechanisms/protocols **interact with each other**

- Routing, addressing, access control, QoS

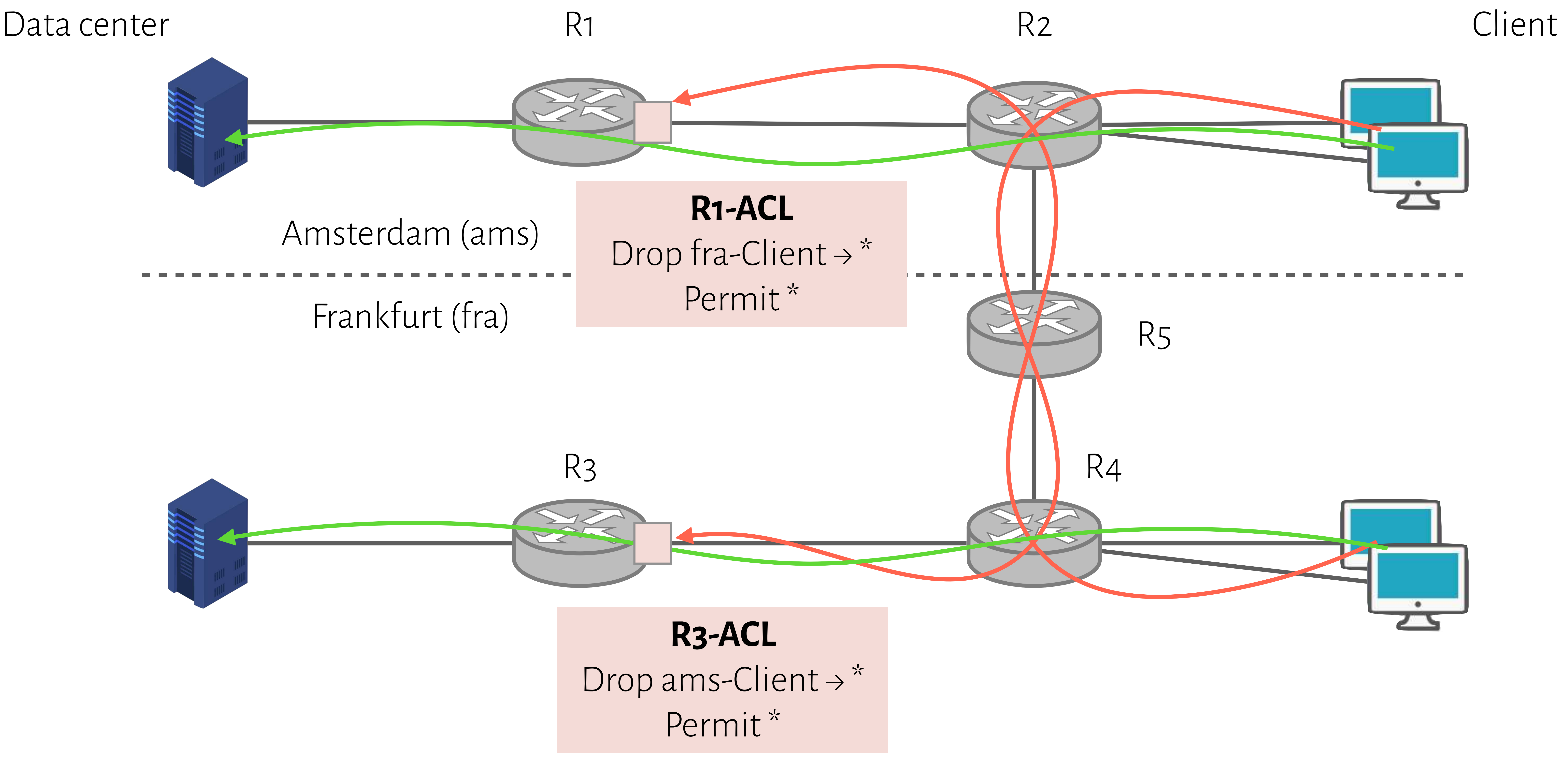
Network control plane is a complicated mess!

Example: access control configuration

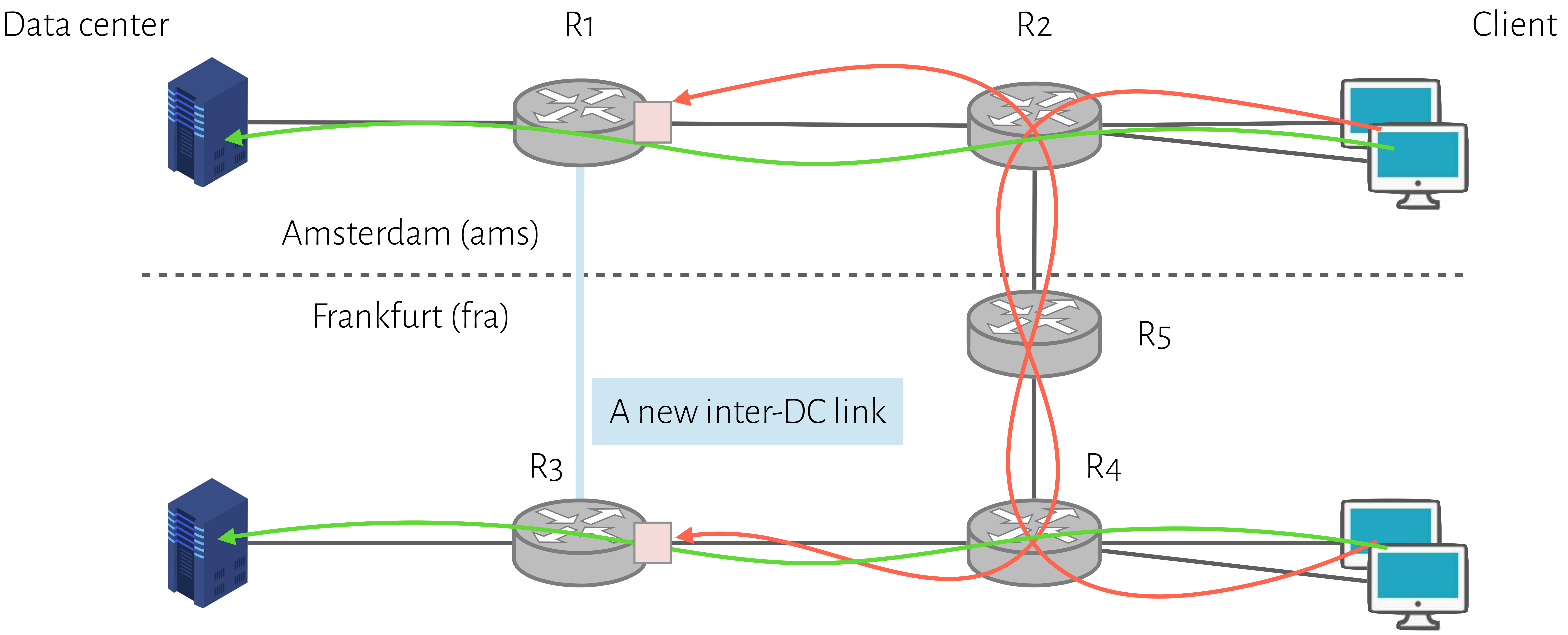


How can we block cross-region traffic?

Example: access control configuration

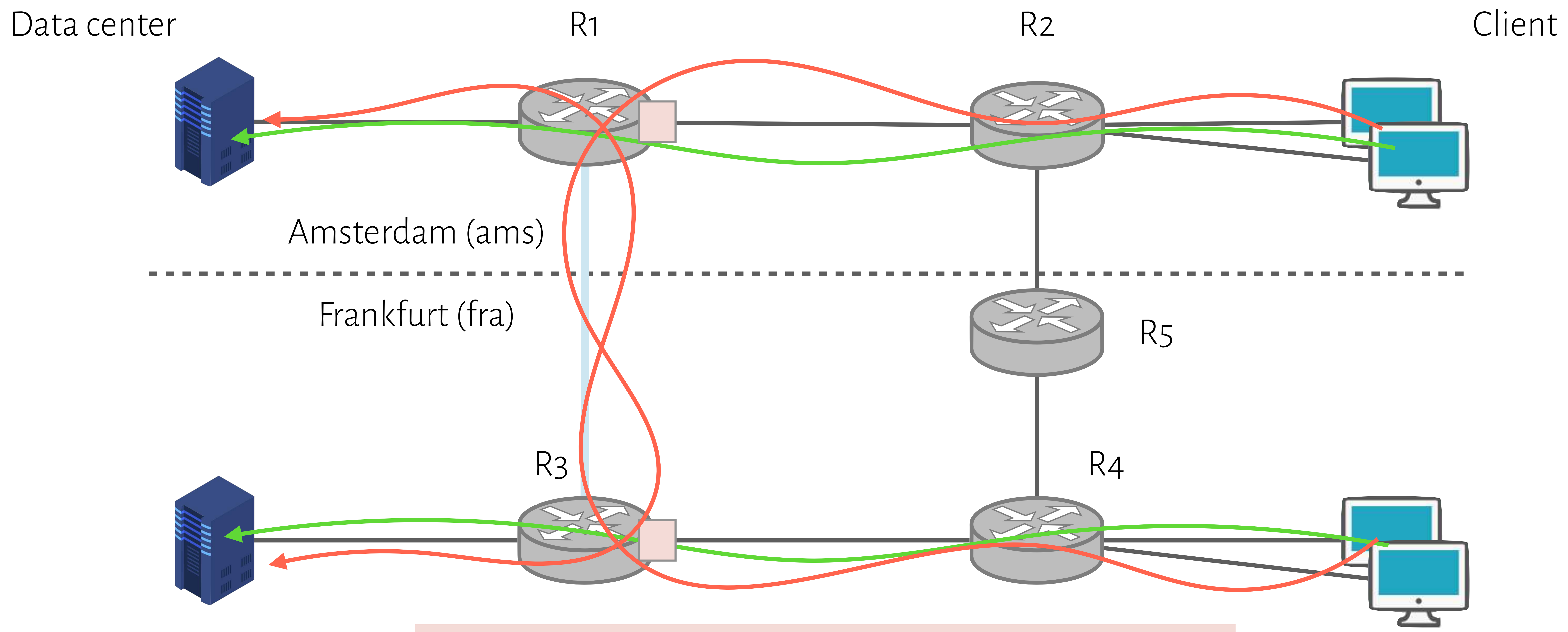


Example: access control configuration



What would be the problem?

Example: access control configuration



Network traffic can bypass the ACLs now.
→ Routing and access control are dependent!

How have we managed to survive?

Network administrators miraculously master this complexity

- Understand all aspects of networks
- Must keep myriad details in mind

The ability to master complexity is both a blessing and a curse!

The **ability to master complexity** is valuable but not the same as **the ability to extract simplicity**

The Gordian Knot

Actually, networks work better during the weekends

News › Science

Fewer heart attack patients die when top cardiologists are away at conferences, study finds

share | Save 14

FOLLOW THE TELEGRAPH

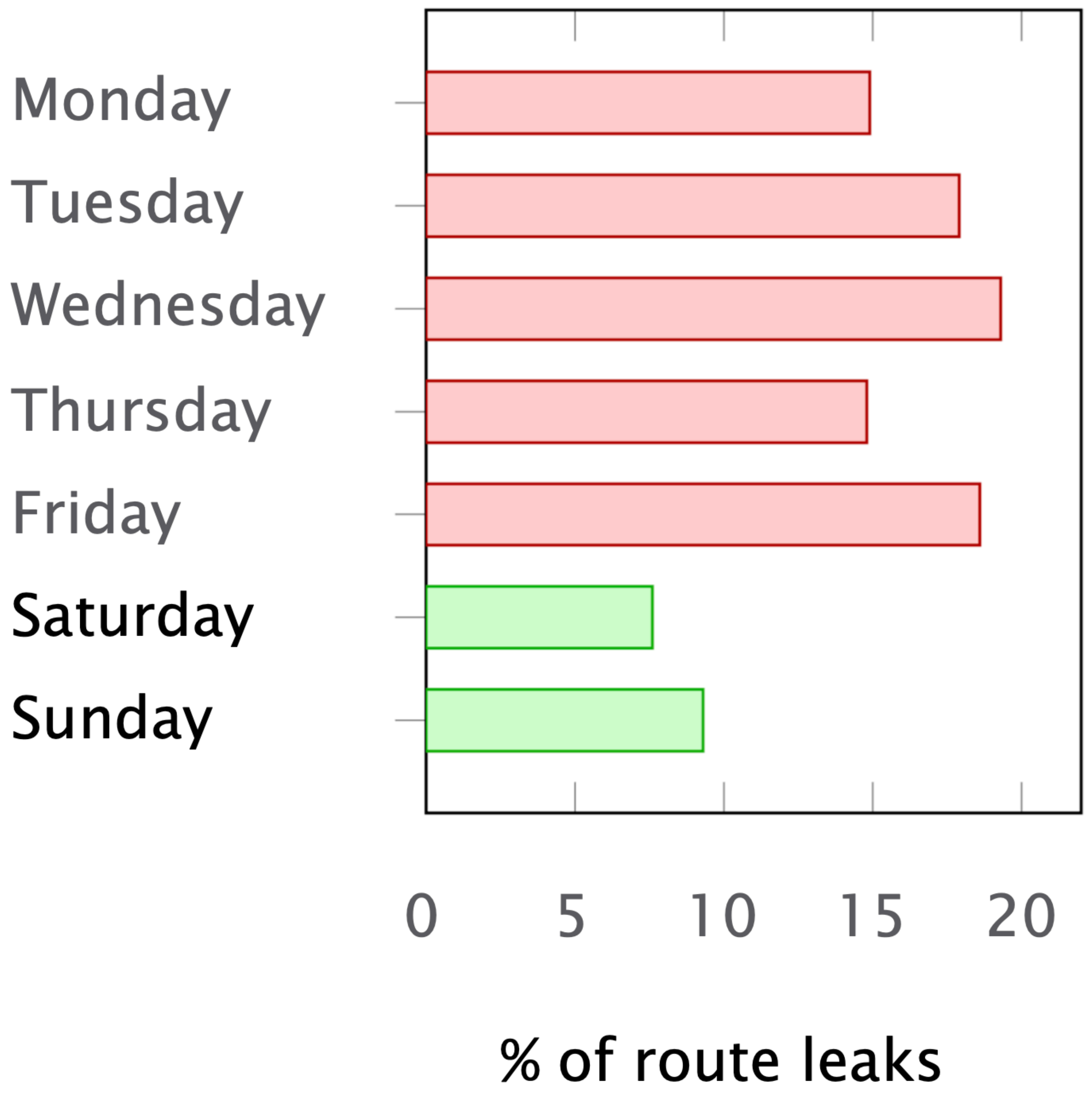
Follow on Facebook | Follow on Twitter | Follow on Instagram | Follow on LinkedIn

Specialists may do more harm than good, a study suggests CREDIT: LYNNE CAMERON PA

Follow

By Sarah Knapton, SCIENCE EDITOR

9 MARCH 2018 • 2:56PM



source: Job Snijders (NTT)

How to extract simplicity? Any examples?

Example: programming

Machine languages: no abstractions

- Hard to deal with low-level details
- Mastering complexity is crucial

High-level languages: operating systems and other abstractions

- File systems, virtual memory, abstract data types...

Modern languages: even more abstractions

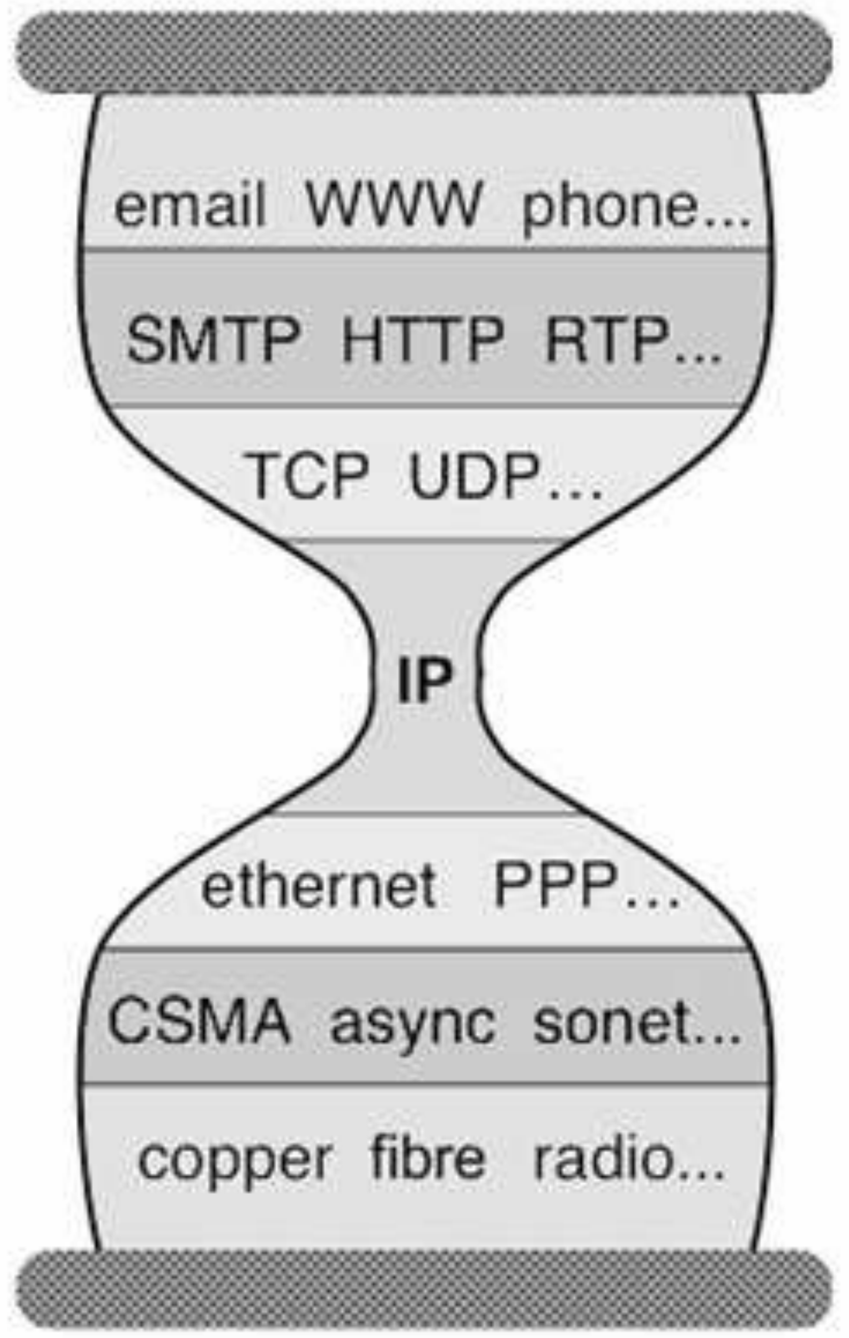
- Object oriented, garbage collection...

"Modularity based on abstractions is the way things get done!"

Barbara Liskov
(MIT, ACM Turing Award 2008, pioneer in programming languages, operating systems, distributed computing)

What abstractions do we have in networking?

Abstractions for data plane



Applications, built on...

Reliable (or unreliable) transport, built on...

Best-effort global packet delivery, built on...

Best-effort local packet delivery, built on...

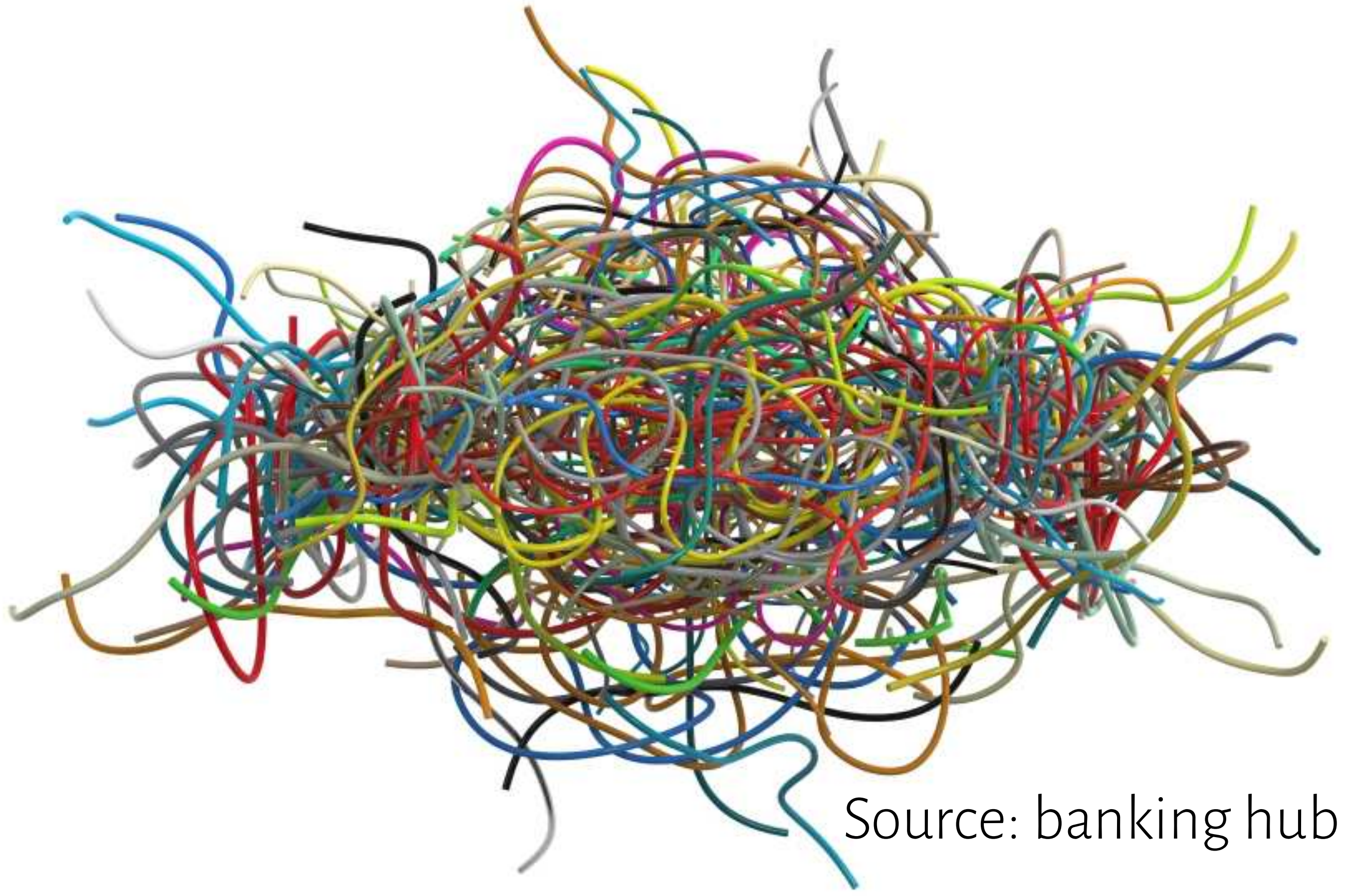
Physical transfer of bits

What about the control plane? Any abstractions?

Control plane is full of mechanisms

Variety of goals, **no modularity**

- Routing: distributed routing algorithms
- Isolation: ACLs, VLANs, Firewalls,
- Traffic engineering: adjusting weights, MPLS,...



Source: banking hub

Control plane: **mechanism without abstraction**

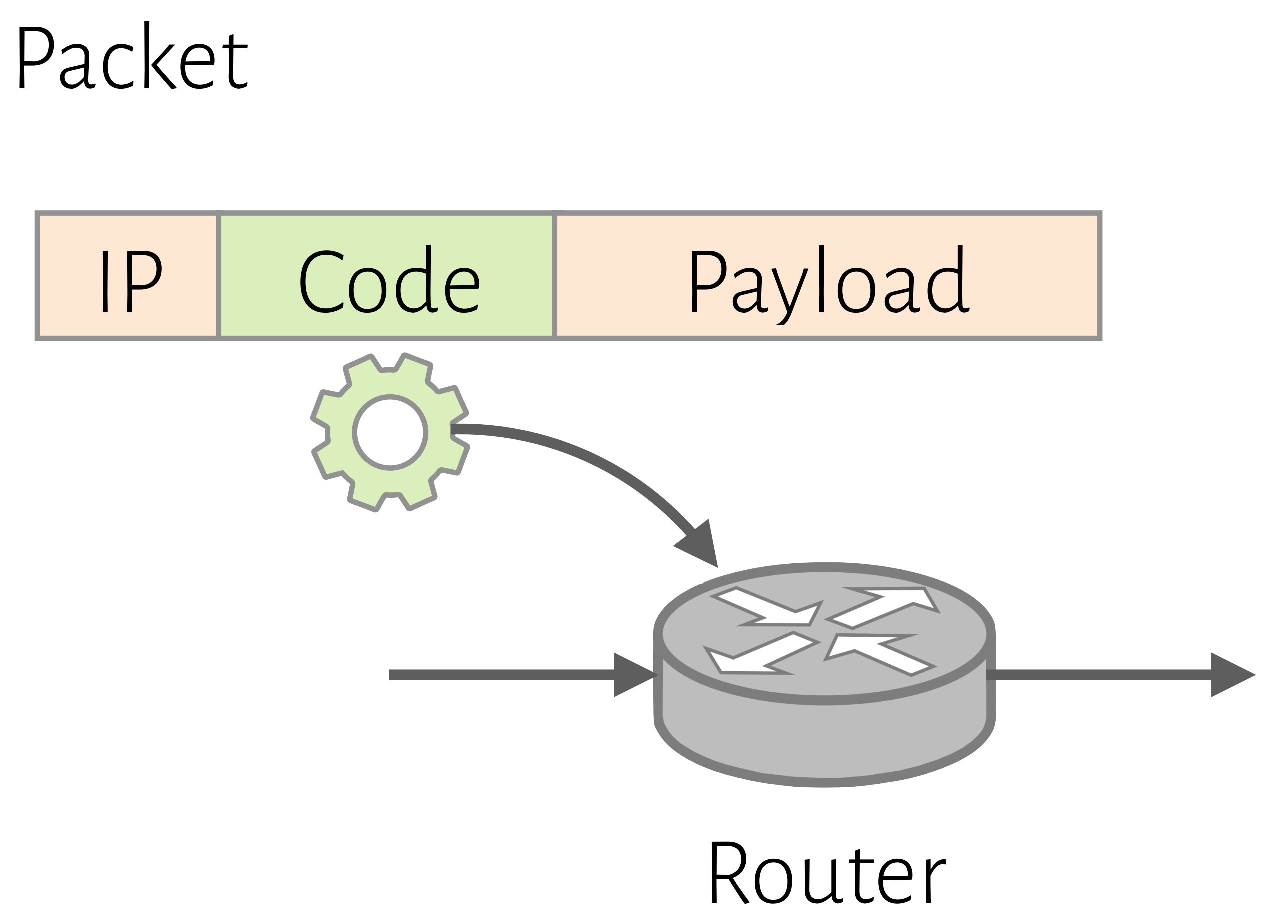
You probably feel that computer networking is all about artifacts (e.g., artifacts, tools), and seems not a real discipline... (at least not like computer architecture, operating systems)

We need abstractions and ultimately, we should be able to **program the network** as we do for computers.

Questions?

The evolution: active networking (1990s)

First attempt making networks **programmable**: demultiplexing packets to software programs



In-band approach: The packet encapsulates a small piece of code that can be executed on the router, based on which the router decides what to do with the packet

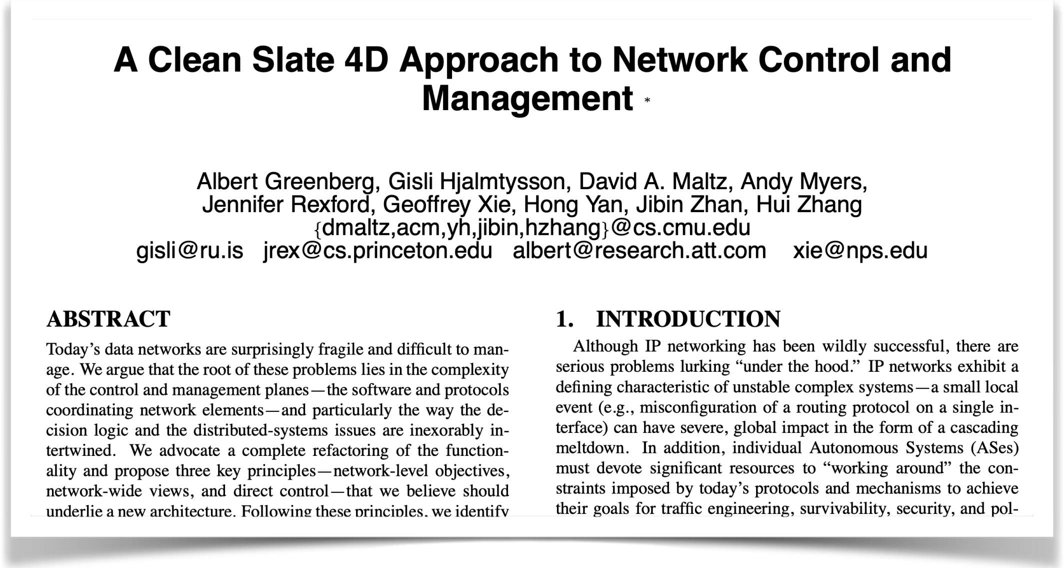
Out-band approach: User injects the code to be executed beforehand → the programmable network approach which received a lot of attention recently.

We will discuss it in our next lecture!

The evolution: control/data plane separation (2003-2007)

4D (ACM SIGCOMM CCR 2004)

- Data, discovery, dissemination, decision
- Clean-slate: network-wide view, direct control, network-global objectives



A Clean Slate 4D Approach to Network Control and Management *

Albert Greenberg, Gisli Hjalmtysson, David A. Maltz, Andy Myers, Jennifer Rexford, Geoffrey Xie, Hong Yan, Jibin Zhan, Hui Zhang
{dmaltz,acm,yh,jibin,hzhang}@cs.cmu.edu
gisli@ru.is jrex@cs.princeton.edu albert@research.att.com xie@nps.edu

ABSTRACT

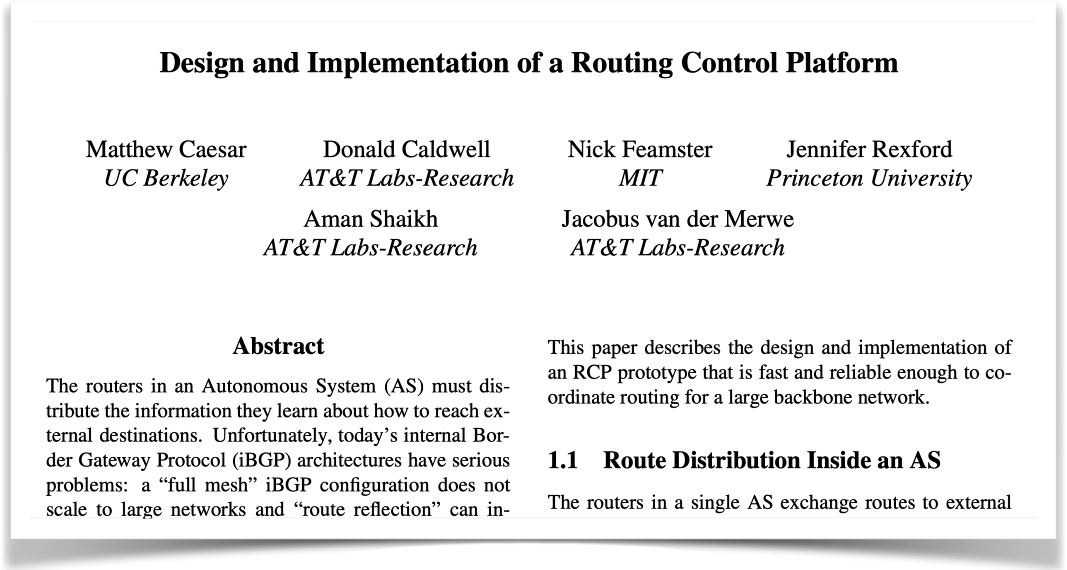
Today's data networks are surprisingly fragile and difficult to manage. We argue that the root of these problems lies in the complexity of the control and management planes—the software and protocols coordinating network elements—and particularly the way the decision logic and the distributed-systems issues are inexorably intertwined. We advocate a complete refactoring of the functionality and propose three key principles—network-level objectives, network-wide views, and direct control—that we believe should underlie a new architecture. Following these principles, we identify

1. INTRODUCTION

Although IP networking has been wildly successful, there are serious problems lurking "under the hood." IP networks exhibit a defining characteristic of unstable complex systems—a small local event (e.g., misconfiguration of a routing protocol on a single interface) can have severe, global impact in the form of a cascading meltdown. In addition, individual Autonomous Systems (ASes) must devote significant resources to "working around" the constraints imposed by today's protocols and mechanisms to achieve their goals for traffic engineering, survivability, security, and pol-

RCP (USENIX NSDI 2005)

- Routing Control Platform for centralized inter-AS routing, replacing iBGP



Design and Implementation of a Routing Control Platform

Matthew Caesar, *UC Berkeley*; Donald Caldwell, *AT&T Labs-Research*; Nick Feamster, *MIT*; Jennifer Rexford, *Princeton University*
Aman Shaikh, *AT&T Labs-Research*; Jacobus van der Merwe, *AT&T Labs-Research*

Abstract

The routers in an Autonomous System (AS) must distribute the information they learn about how to reach external destinations. Unfortunately, today's internal Border Gateway Protocol (iBGP) architectures have serious problems: a "full mesh" iBGP configuration does not scale to large networks and "route reflection" can in-

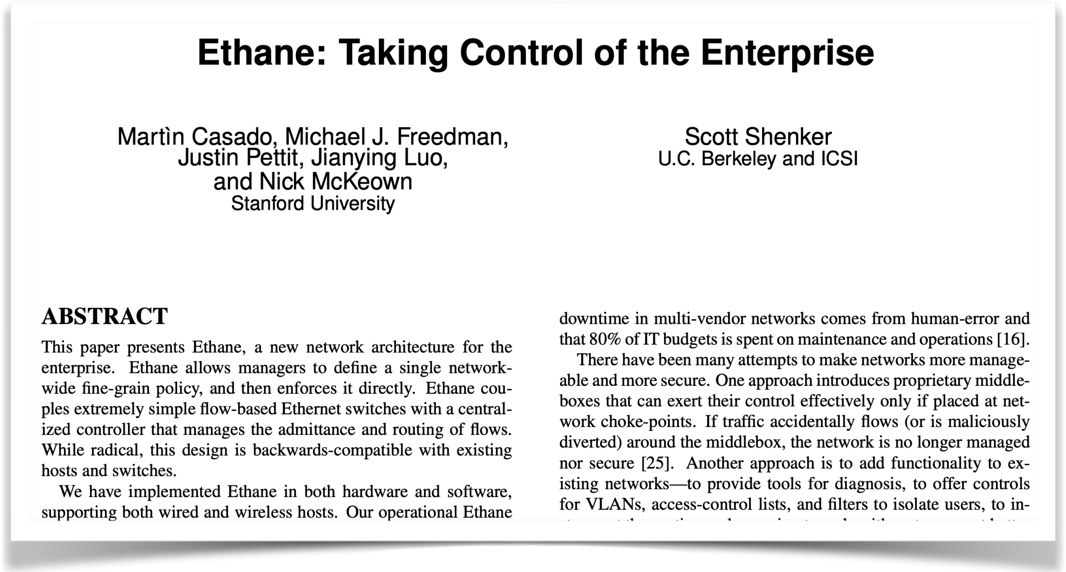
This paper describes the design and implementation of an RCP prototype that is fast and reliable enough to coordinate routing for a large backbone network.

1.1 Route Distribution Inside an AS

The routers in a single AS exchange routes to external

Ethane (ACM SIGCOMM 2007)

- Flow-based switching with centralized control for enterprise
- Precursor of SDN



Ethane: Taking Control of the Enterprise

Martìn Casado, Michael J. Freedman, Justin Pettit, Jianying Luo, and Nick McKeown
Stanford University
Scott Shenker
U.C. Berkeley and ICSI

ABSTRACT

This paper presents Ethane, a new network architecture for the enterprise. Ethane allows managers to define a single network-wide fine-grain policy, and then enforces it directly. Ethane couples extremely simple flow-based Ethernet switches with a centralized controller that manages the admittance and routing of flows. While radical, this design is backwards-compatible with existing hosts and switches.

We have implemented Ethane in both hardware and software, supporting both wired and wireless hosts. Our operational Ethane

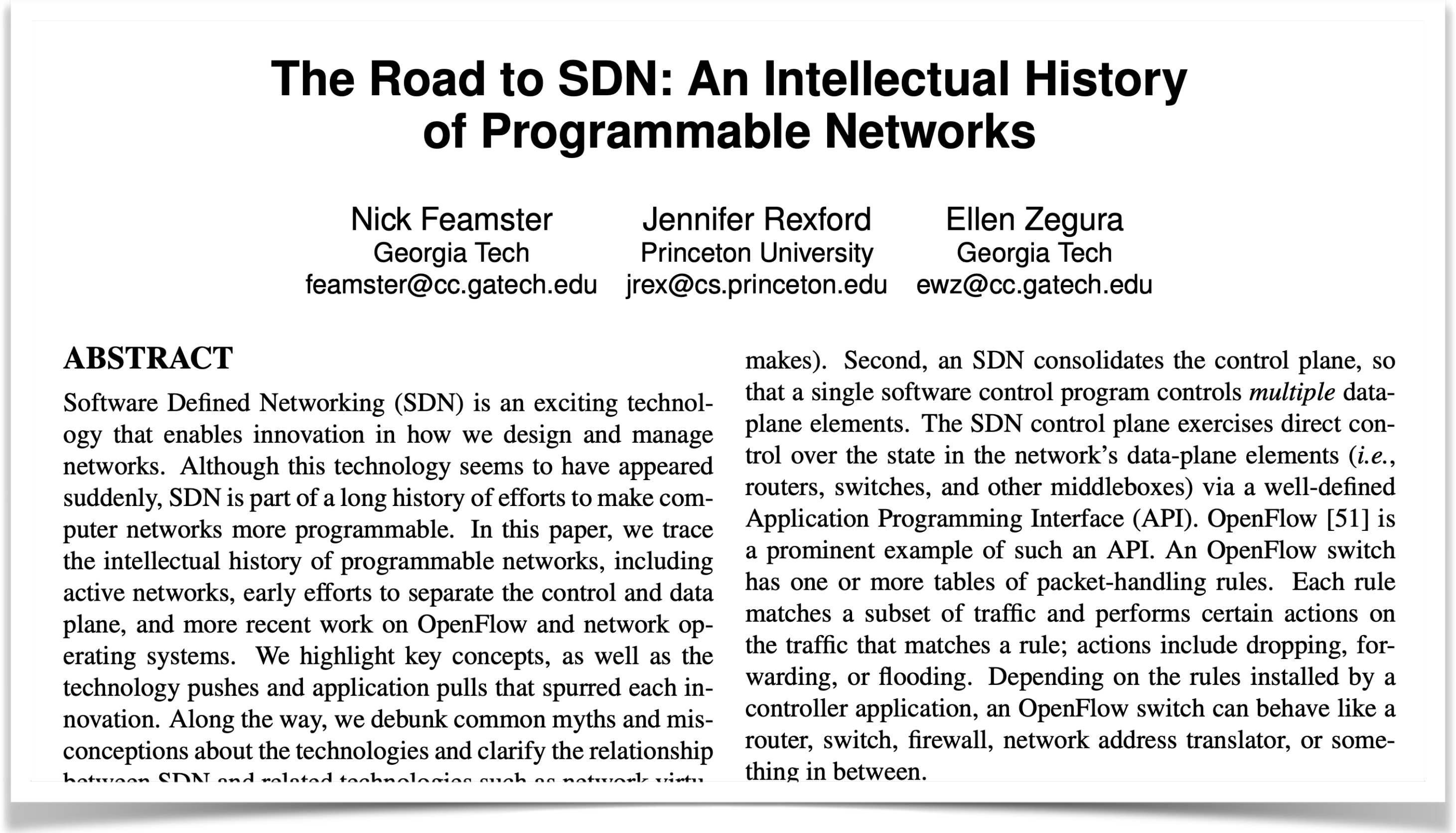
downtime in multi-vendor networks comes from human-error and that 80% of IT budgets is spent on maintenance and operations [16].

There have been many attempts to make networks more manageable and more secure. One approach introduces proprietary middleboxes that can exert their control effectively only if placed at network choke-points. If traffic accidentally flows (or is maliciously diverted) around the middlebox, the network is no longer managed nor secure [25]. Another approach is to add functionality to existing networks—to provide tools for diagnosis, to offer controls for VLANs, access-control lists, and filters to isolate users, to in-

Software defined network

A network in which

- The **control** plane is physically **separate from** the **data** plane
- A **single** (logically centralized) **control** plane controls several forwarding devices



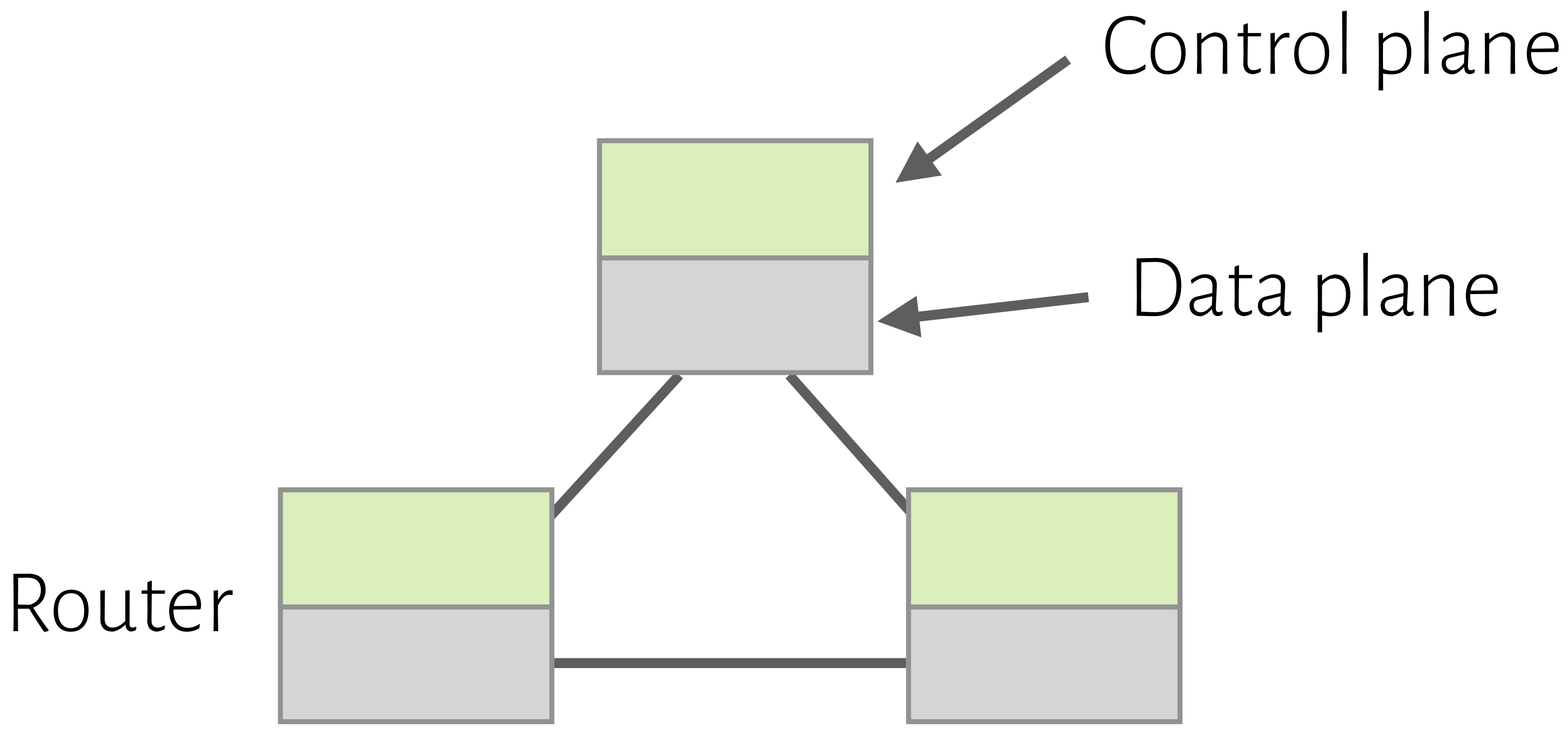
The Road to SDN: An Intellectual History of Programmable Networks

Nick Feamster, Georgia Tech, feamster@cc.gatech.edu
Jennifer Rexford, Princeton University, jrex@cs.princeton.edu
Ellen Zegura, Georgia Tech, ewz@cc.gatech.edu

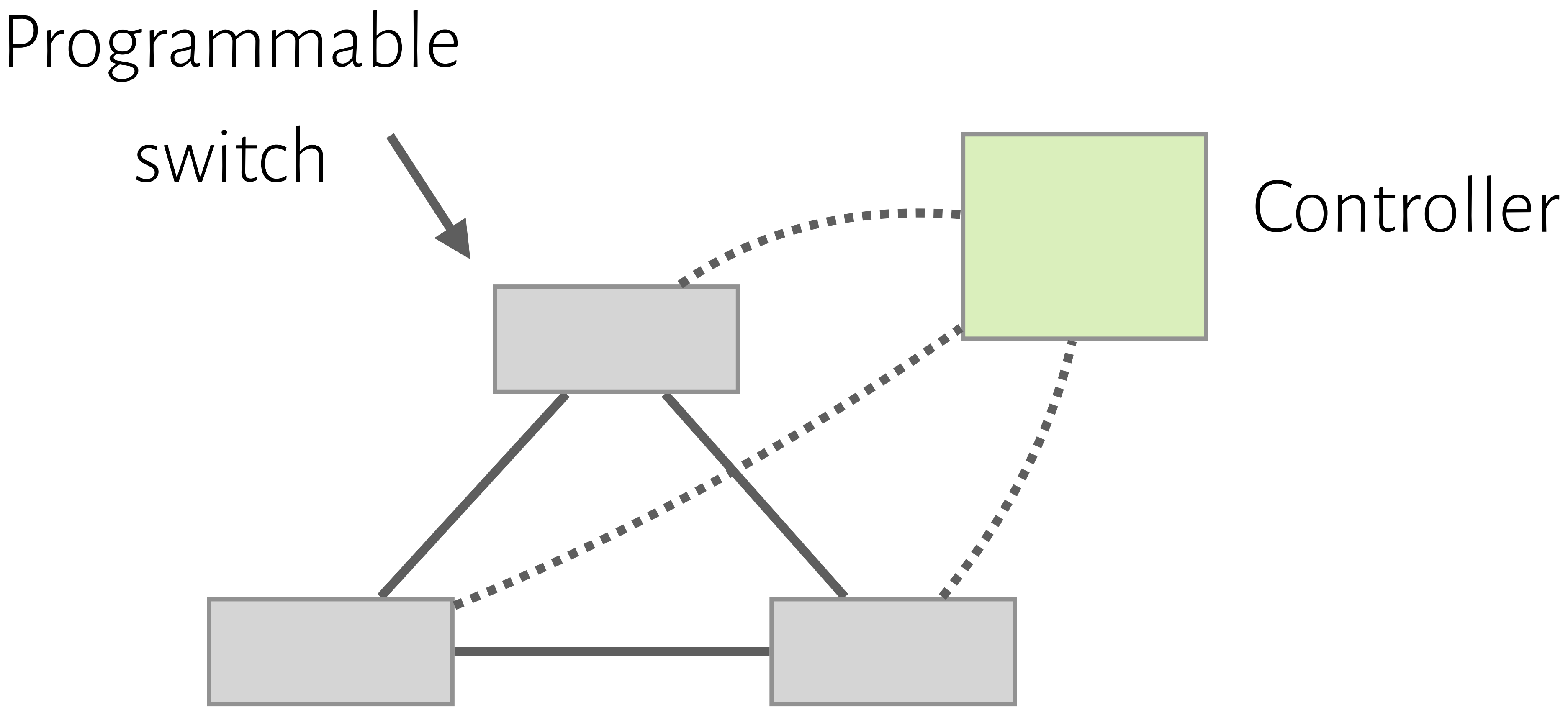
ABSTRACT

Software Defined Networking (SDN) is an exciting technology that enables innovation in how we design and manage networks. Although this technology seems to have appeared suddenly, SDN is part of a long history of efforts to make computer networks more programmable. In this paper, we trace the intellectual history of programmable networks, including active networks, early efforts to separate the control and data plane, and more recent work on OpenFlow and network operating systems. We highlight key concepts, as well as the technology pushes and application pulls that spurred each innovation. Along the way, we debunk common myths and misconceptions about the technologies and clarify the relationship

makes). Second, an SDN consolidates the control plane, so that a single software control program controls *multiple* data-plane elements. The SDN control plane exercises direct control over the state in the network's data-plane elements (*i.e.*, routers, switches, and other middleboxes) via a well-defined Application Programming Interface (API). OpenFlow [51] is a prominent example of such an API. An OpenFlow switch has one or more tables of packet-handling rules. Each rule matches a subset of traffic and performs certain actions on the traffic that matches a rule; actions include dropping, forwarding, or flooding. Depending on the rules installed by a controller application, an OpenFlow switch can behave like a router, switch, firewall, network address translator, or something in between.

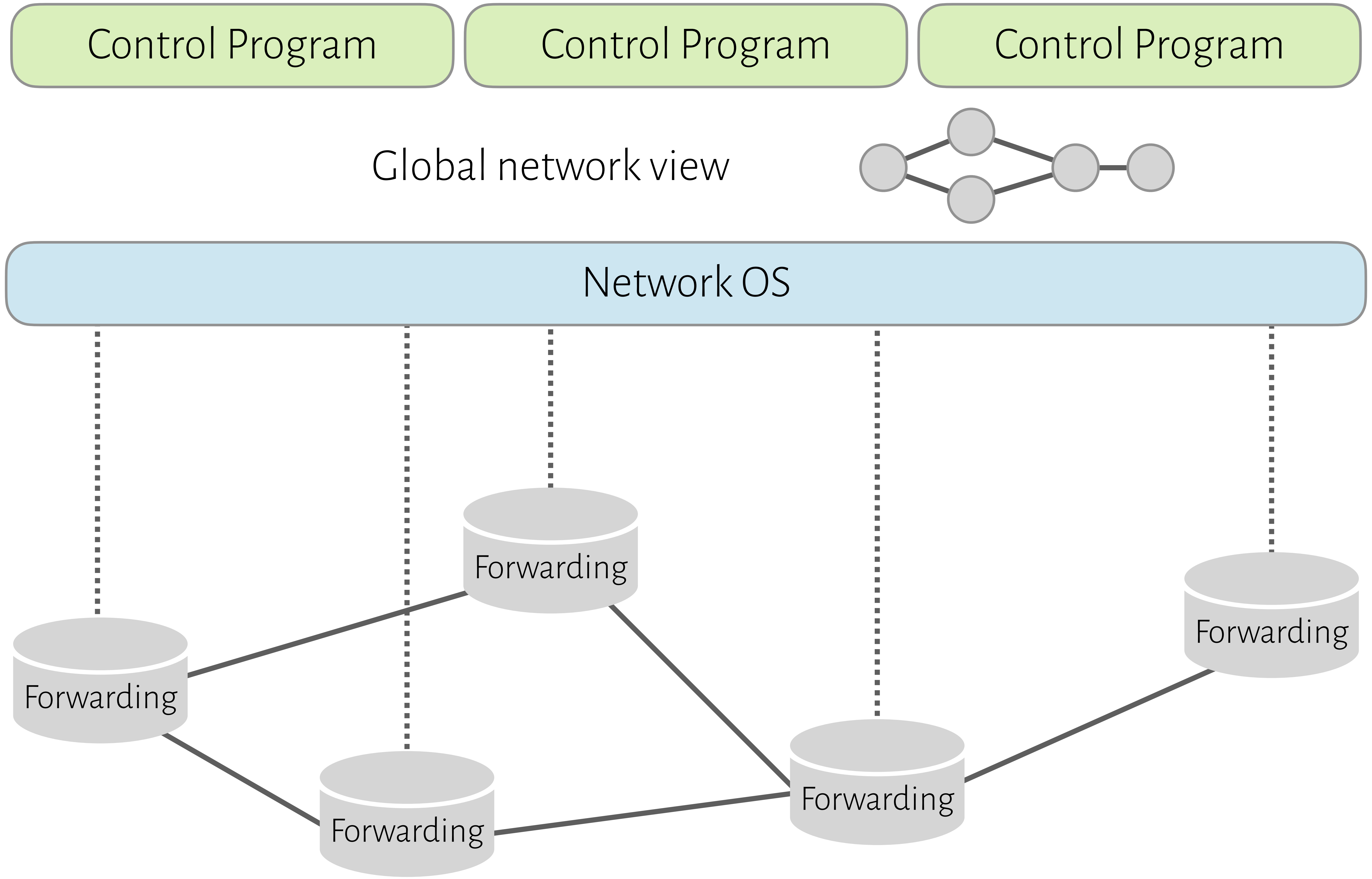


Traditional network

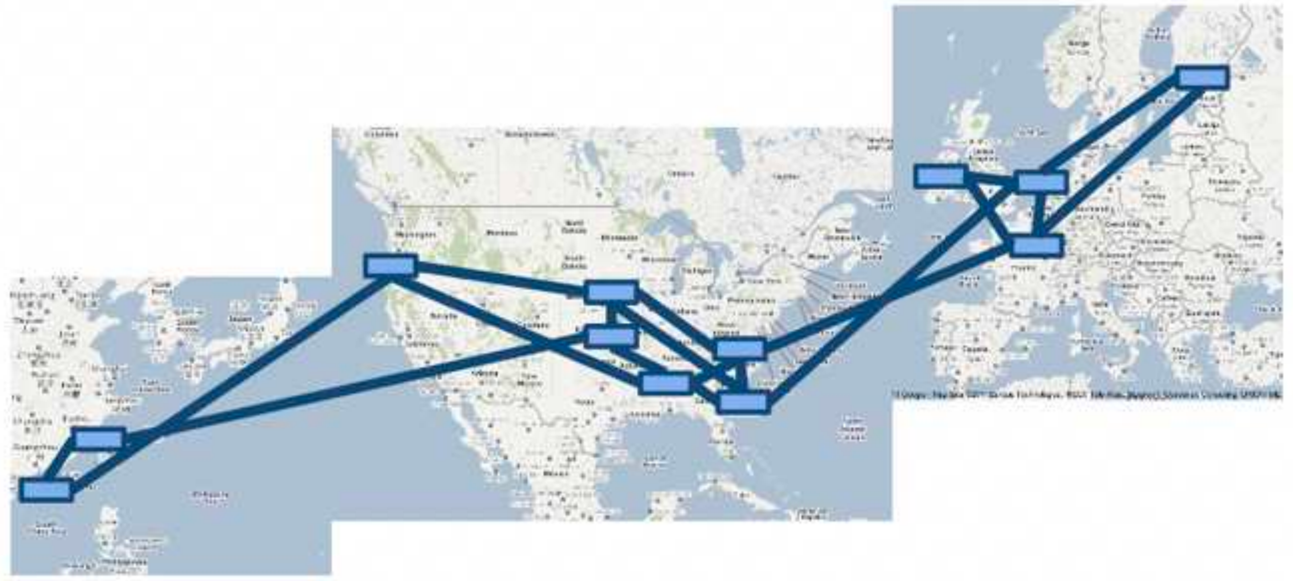


Software define network

SDN architecture overview

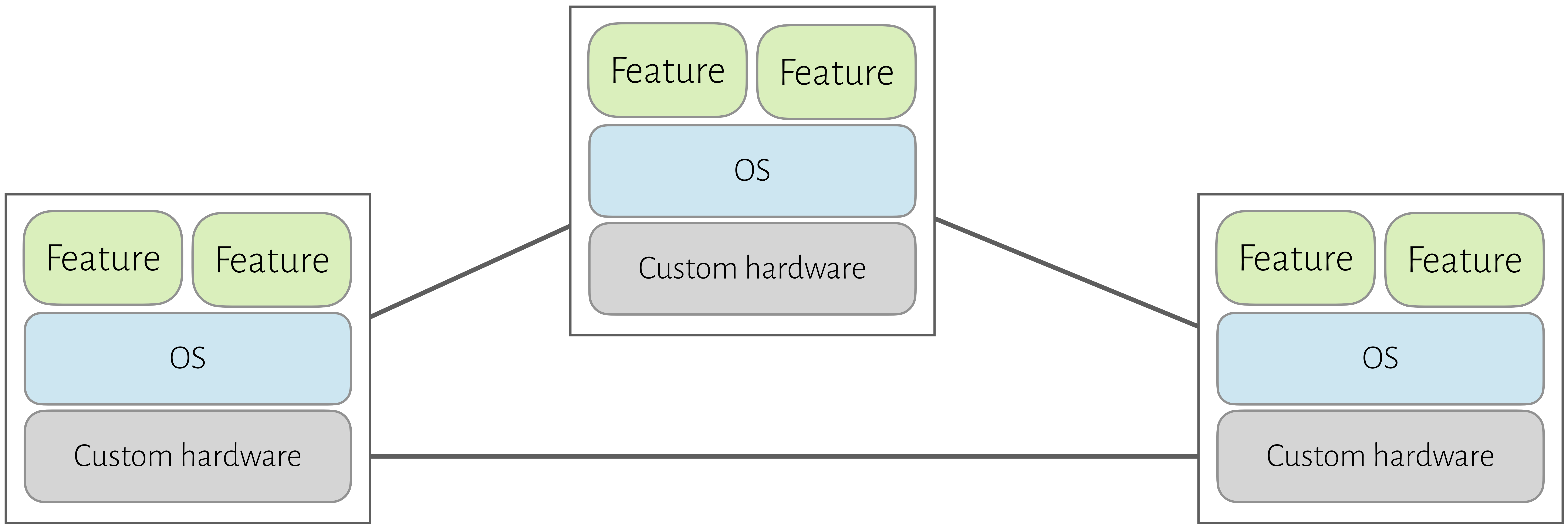


A major trend in networking

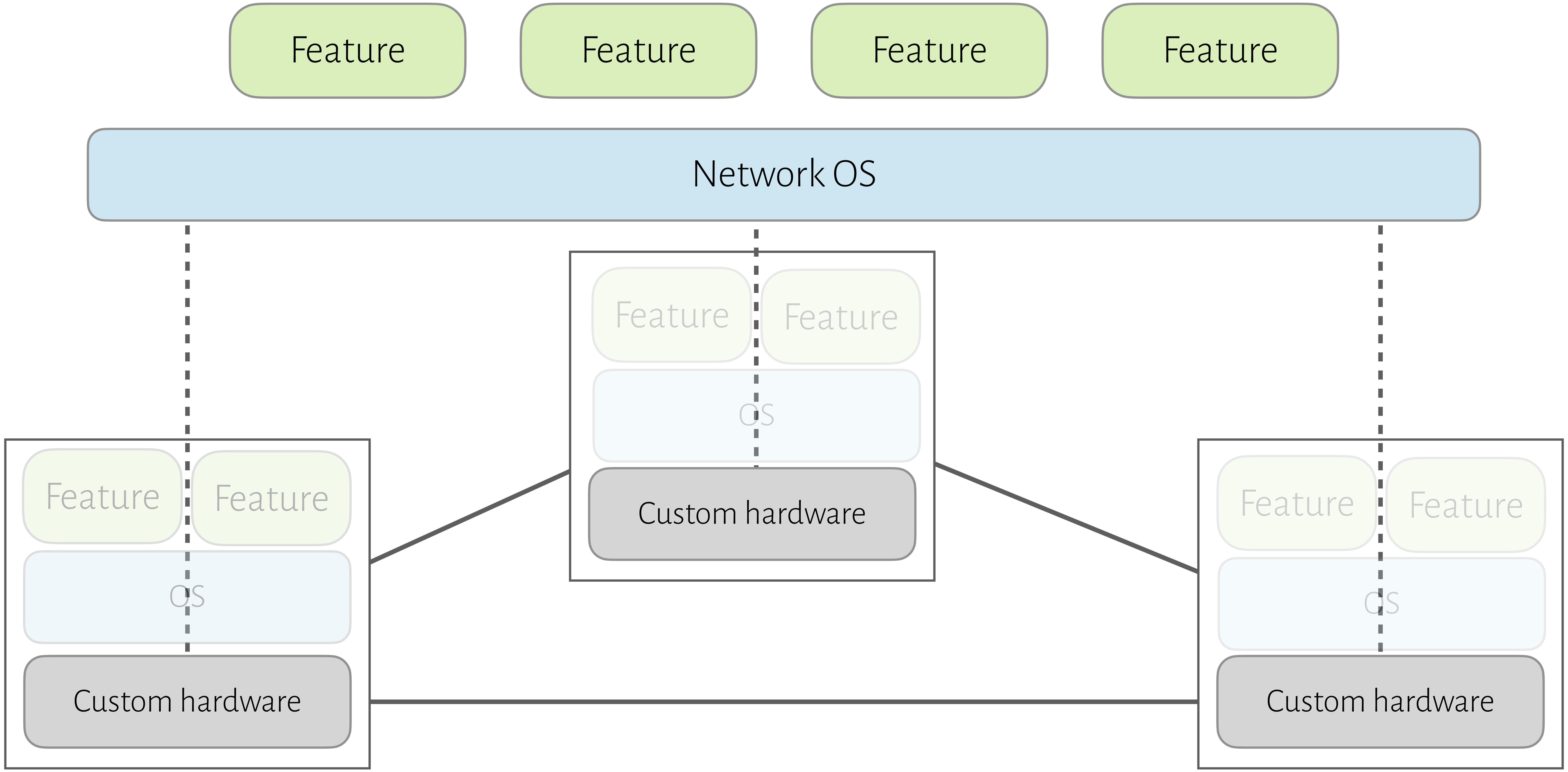


Startup from Stanford and Berkeley in 2007, acquired by VMware in 2012 for $1.26 billion.

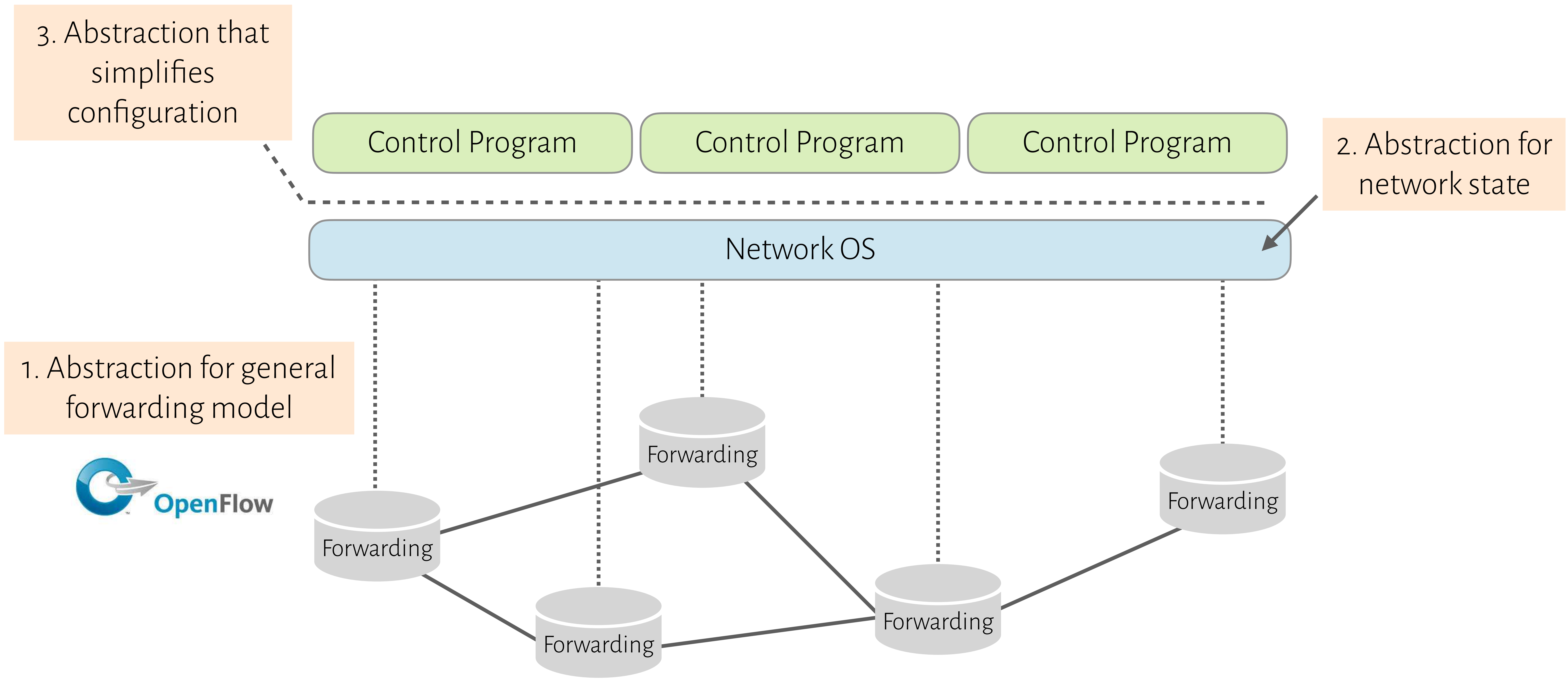
How SDN changes the network



How SDN changes the network



Abstractions in SDN



Abstraction #1: forwarding abstraction

Express intent independent of implementation

OpenFlow is the current proposal for forwarding

- Standardized interface to switch: non-proprietary COTS hardware and software
- Configuration in terms of flow entries: <header, action>
- No hardware modifications needed, simply a firmware update

Design details concern exact nature of **match+action**

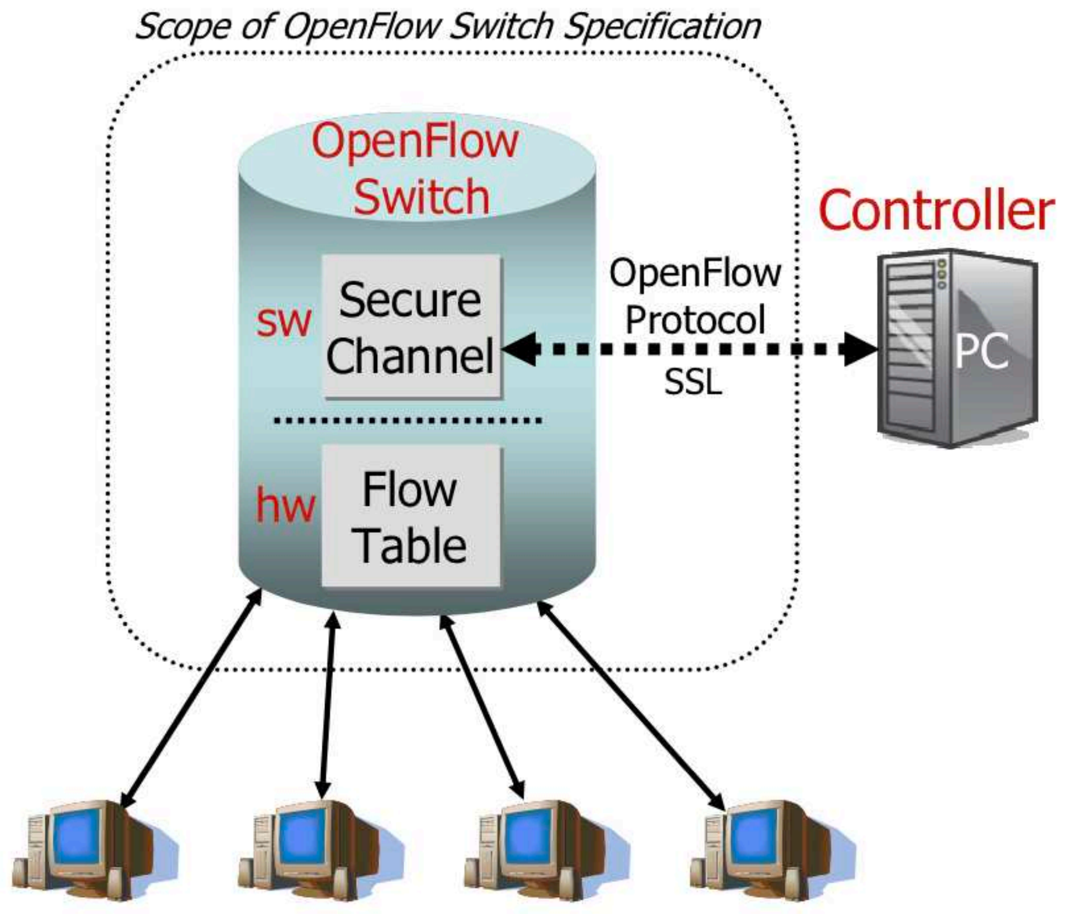
Benefits

- Much cheaper, no more $27K for a single switch
- No vendor lock-in

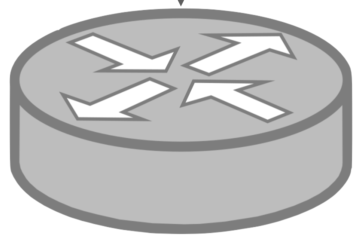
OpenFlow

Control Program | Control Program | Control Program

Network OS



OpenFlow switch



OpenFlow protocol

Flow tables:
match+action

OPEN NETWORKING FOUNDATION

OpenFlow Switch Specification

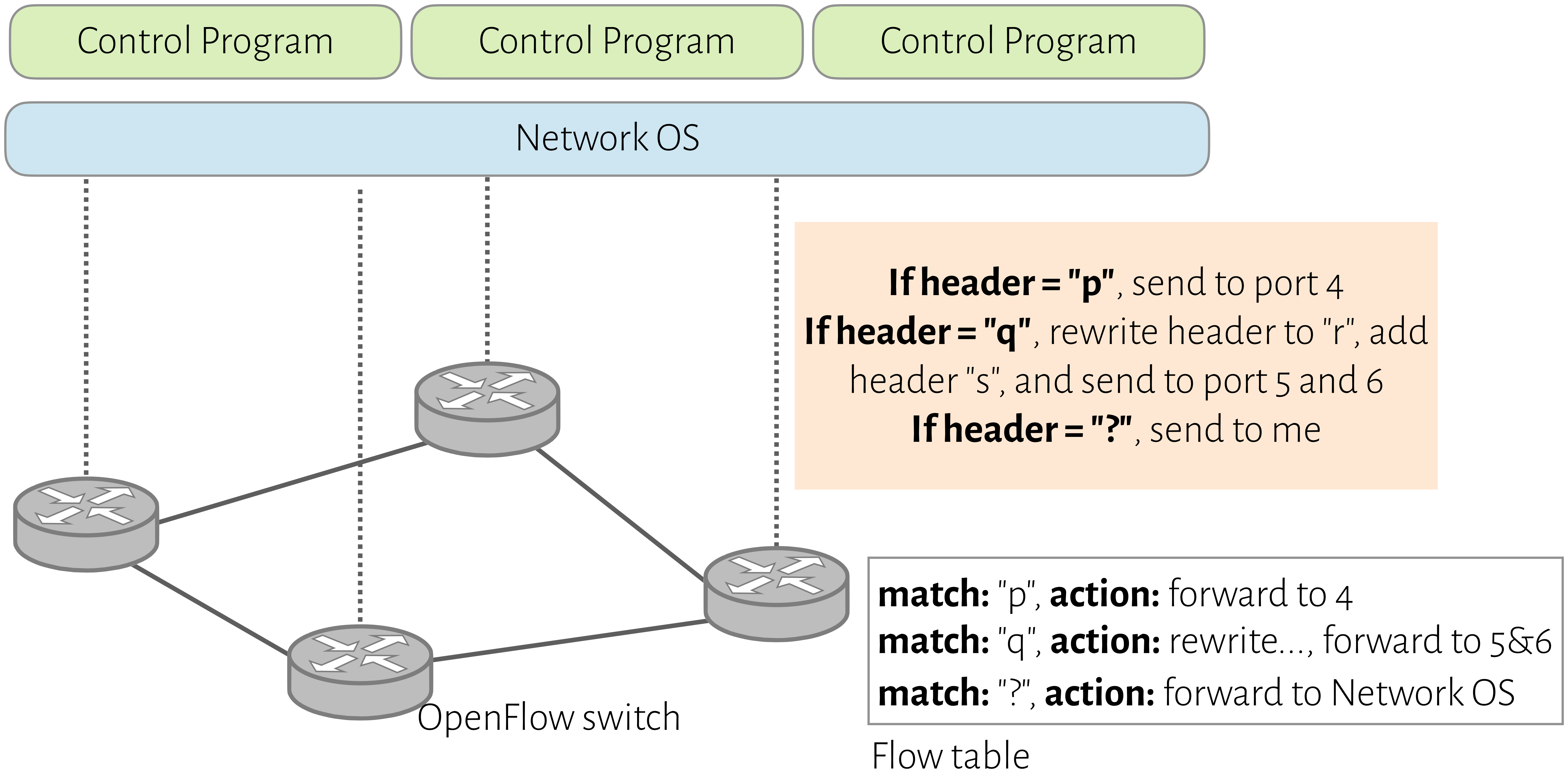
Version 1.5.1 (Protocol version 0x06)

March 26, 2015

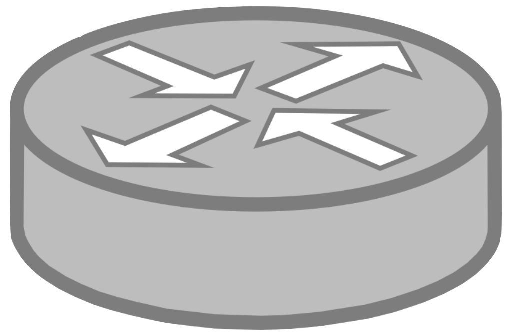
ONF TS-025

https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf

OpenFlow example



Flow table(s) on OpenFlow switches



Flow 1	Rule (exact & wildcard)	Action	Statistics	Priority
Flow 2	Rule (exact & wildcard)	Action	Statistics	Priority
Flow 3	Rule (exact & wildcard)	Action	Statistics	Priority
		……		
Flow *N*	Rule (exact & wildcard)	Action	Statistics	Priority

Exploit the forwarding tables that are already in routers, switches, and chipsets

Match+action

Match arbitrary bits in headers

- Match on any header, or new header
- Allows any flow granularity

In	VLAN	Ethernet			IP			TCP	
Port	ID	SA	DA	Type	SA	DA	Proto	Src	Dst

Match: 1000X01XX0101001X

Action

- Forward to port(s), drop, send to the controller
- Overwrite header with mask, push or pop
- Forward at specific bit-rate
- Do not support payload-related network functions like deep packet inspection

Abstraction #2: network state abstraction

Global network view

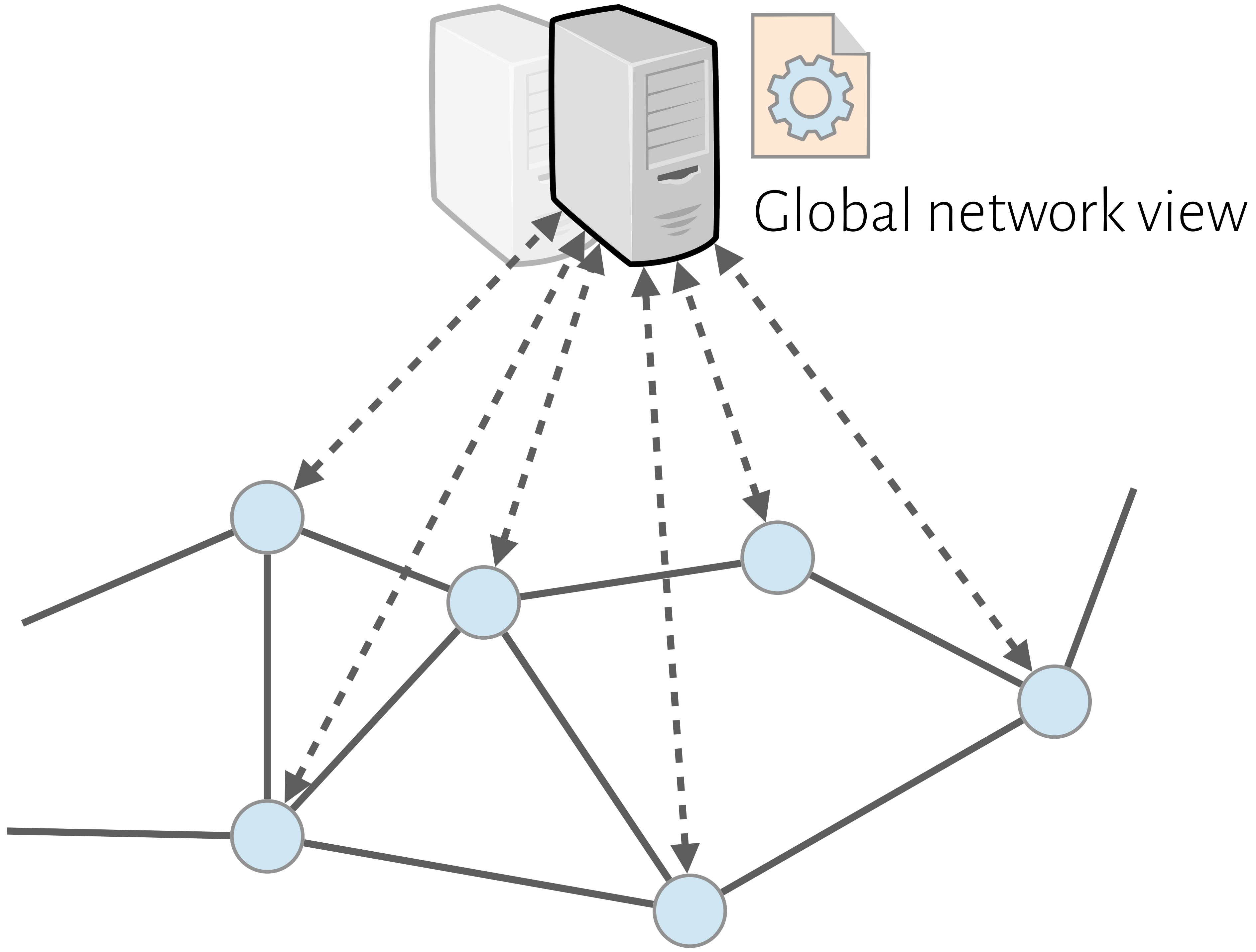
- Annotated network graph provided through an API

Implementation: "Network Operating Systems"

- Runs on servers in network (as "controllers")
- Replicated for reliability

Information flows both ways

- Information from routers/switches to form view
- Configurations to routers/switches to control forwarding

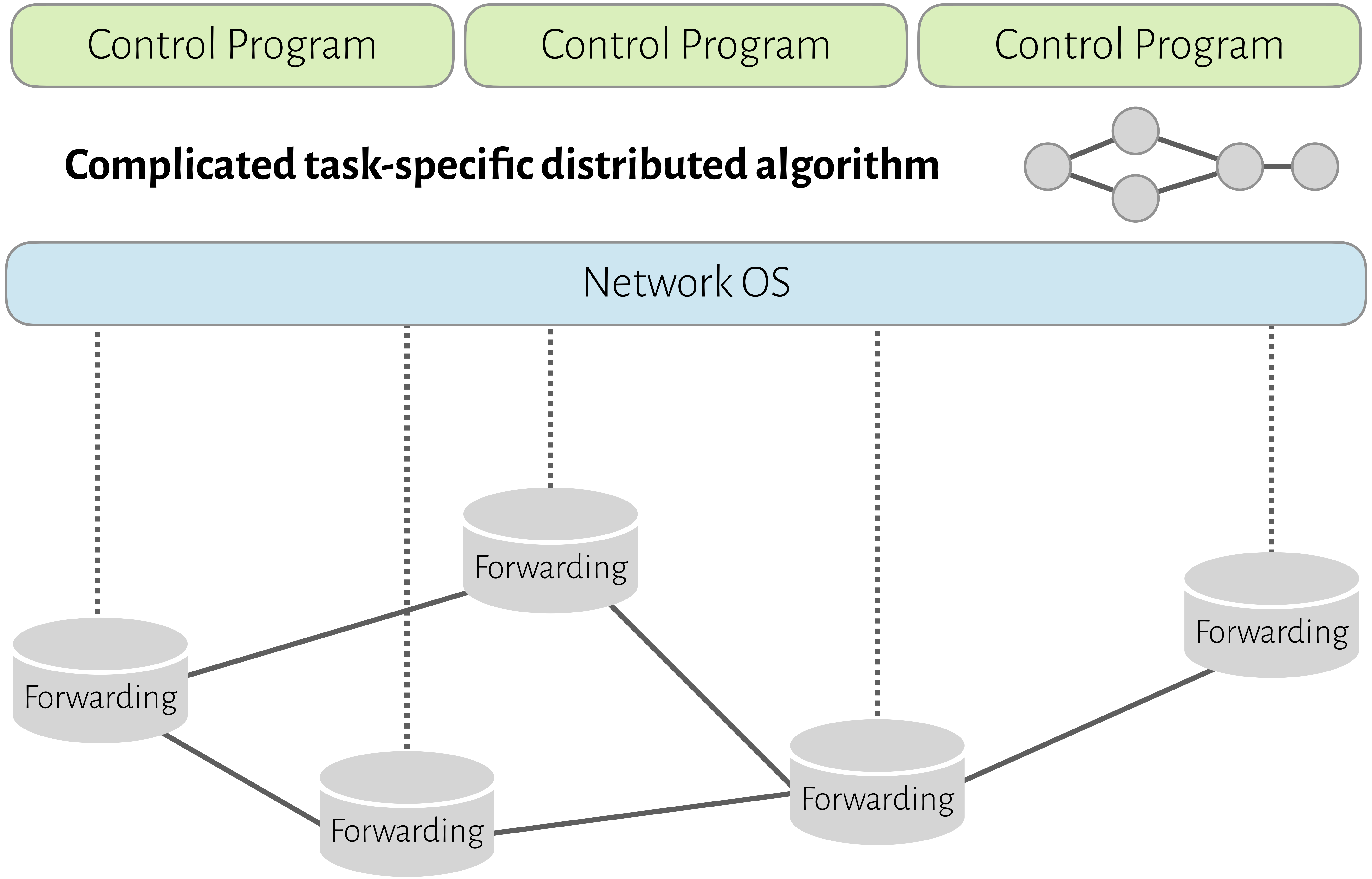


Major change in paradigm

Control program: Configuration = Function(View)

Control mechanism now is a **program** using Network OS APIs

Not a distributed protocol, just a graph algorithm



Abstraction #3: specification abstraction

Control mechanism expresses **desired behavior**

- Whether it be isolation, access control, or QoS

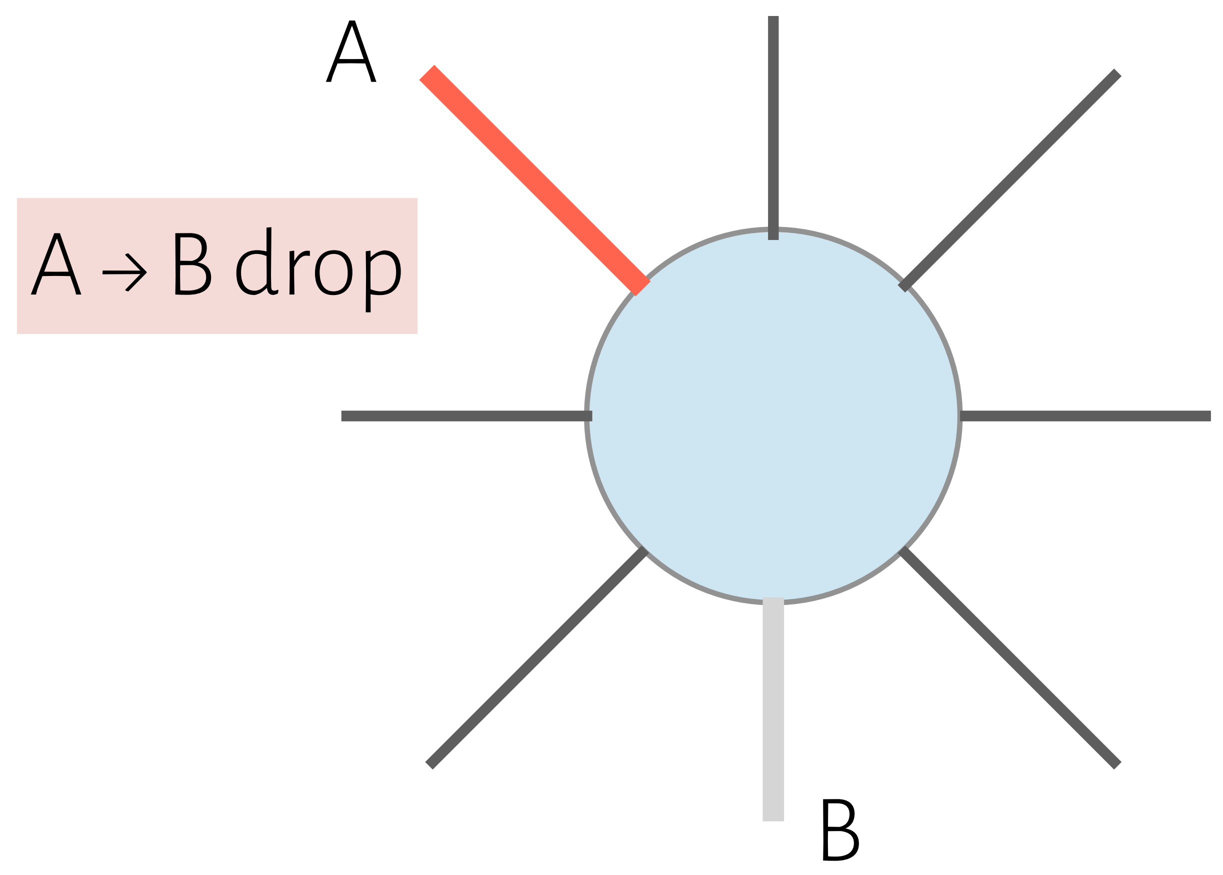
It should not be responsible for implementing that behavior on physical network infrastructure

- Requires configuring the forwarding tables in each switch

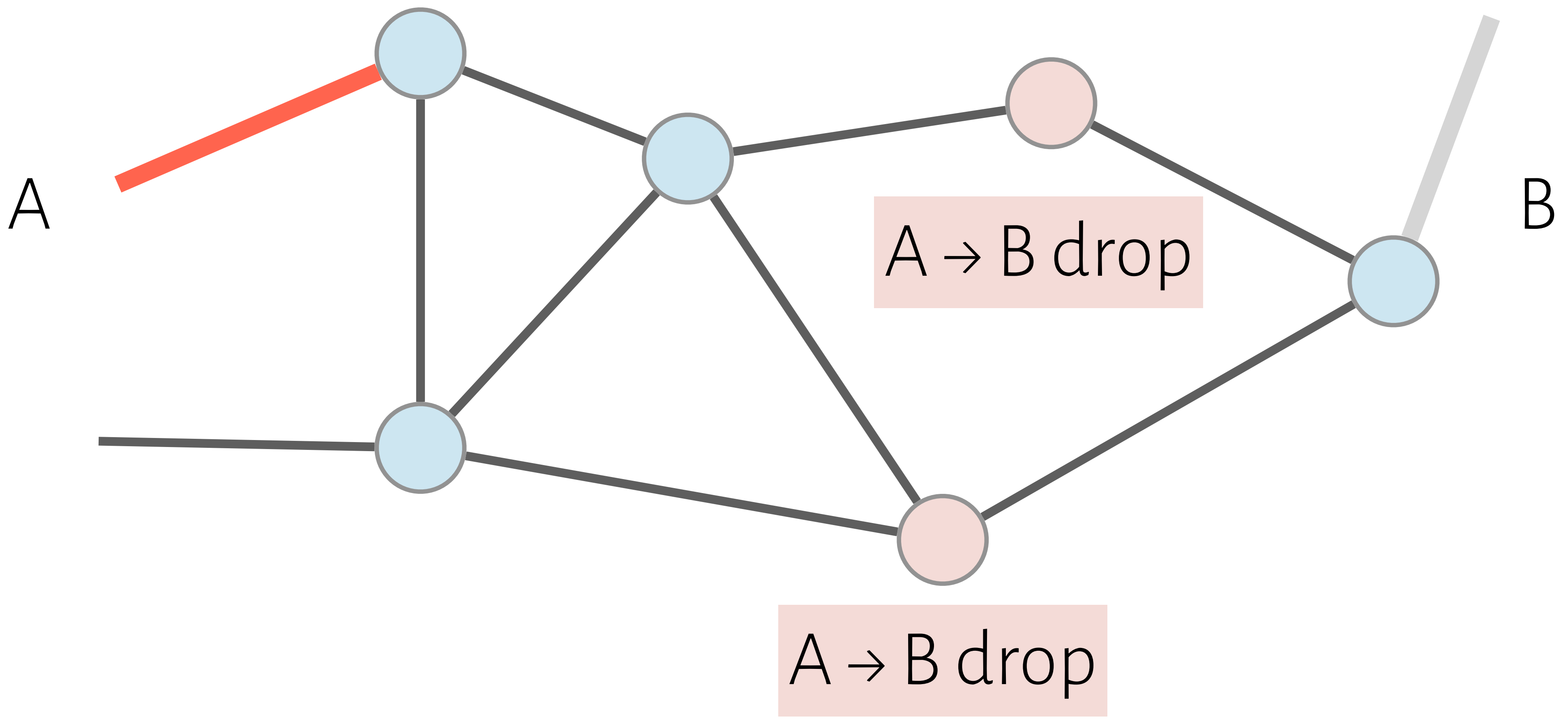
Proposed abstraction: **abstract view of the network**

- Abstract view models only enough detail to specify goals
- Will depend on task semantics

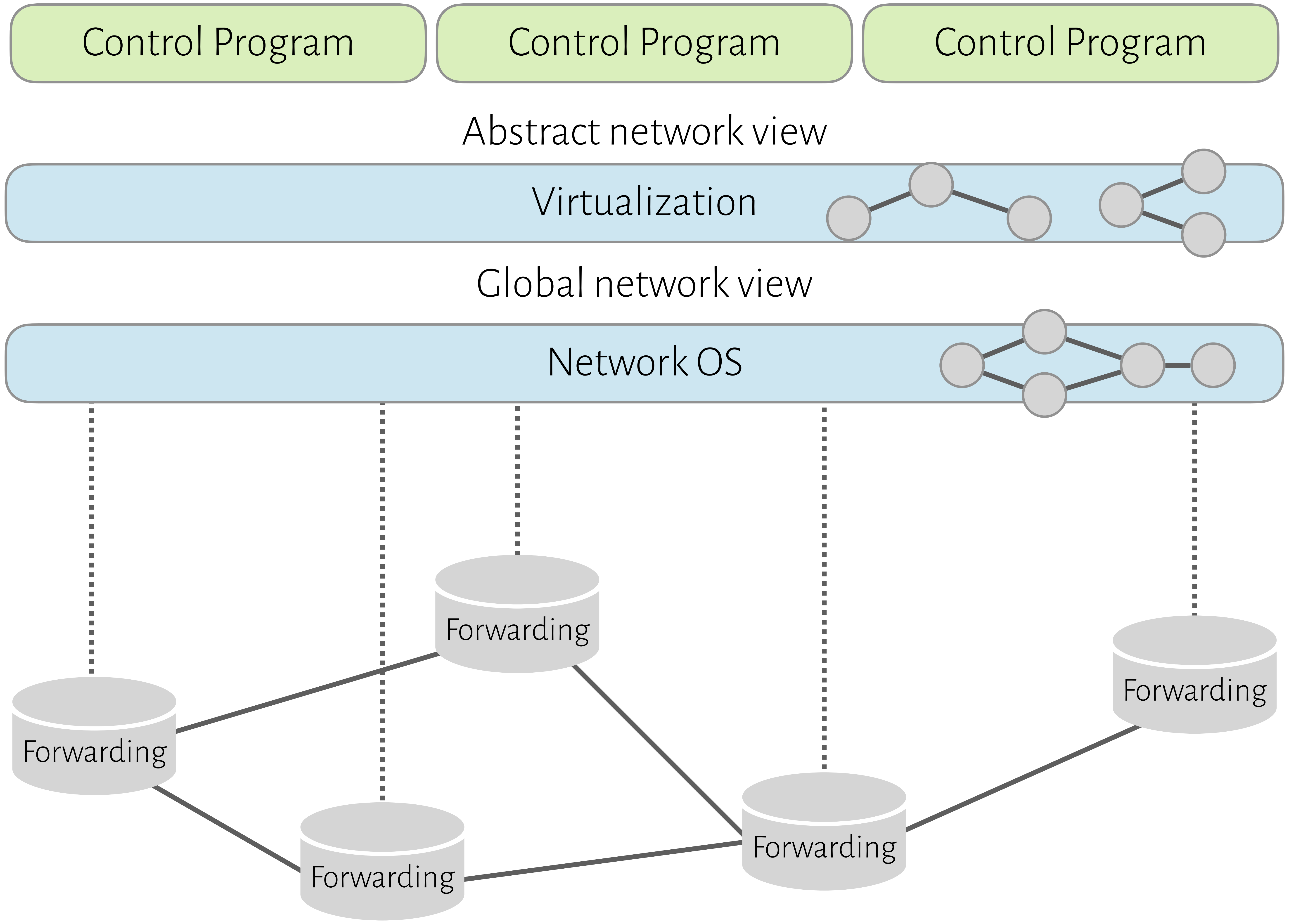
Abstract network view



Global network view



SDN control plane layers

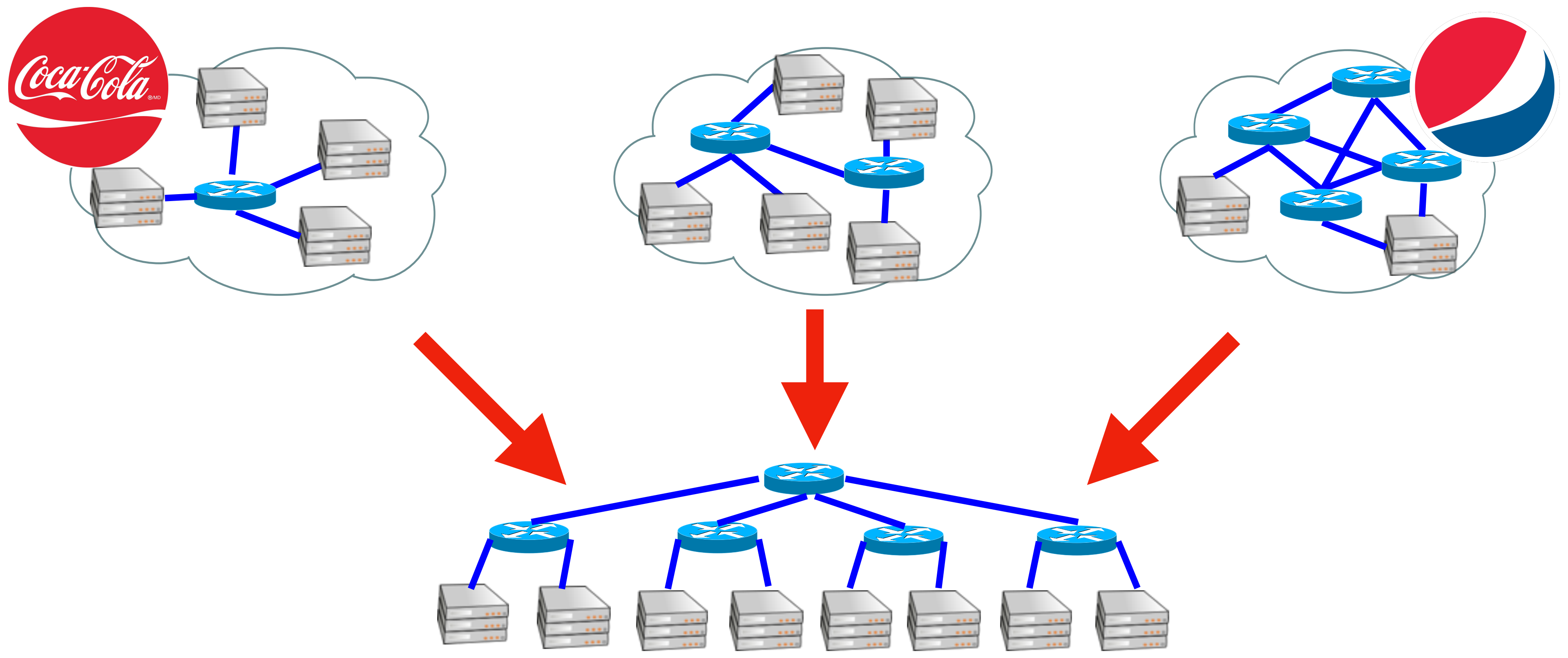


Questions?

Network virtualization

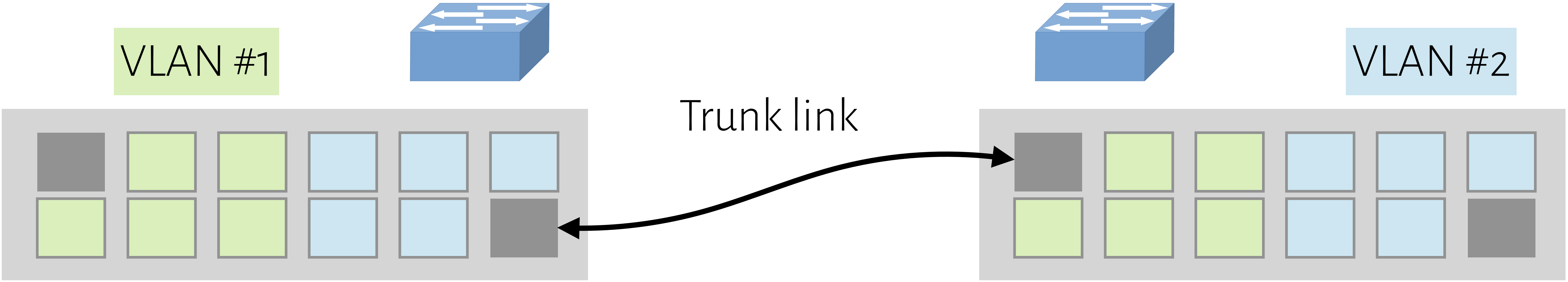
Ability to run multiple virtual networks that

- Each has a separate control and data plane
- Co-exist together on top of one physical network
- Can be managed by individual parties that potentially do not trust each other



What techniques we have learned that can be used for network virtualization?

VLAN



VLAN provides basic isolation for Ethernet LANs, but it has many problems for network virtualization

- **Cannot program packet forwarding:** stuck with learning switch and spanning tree
- **No obvious opt-in mechanisms:** who maps a packet to a VLAN?
- **Resource isolation** is problematic

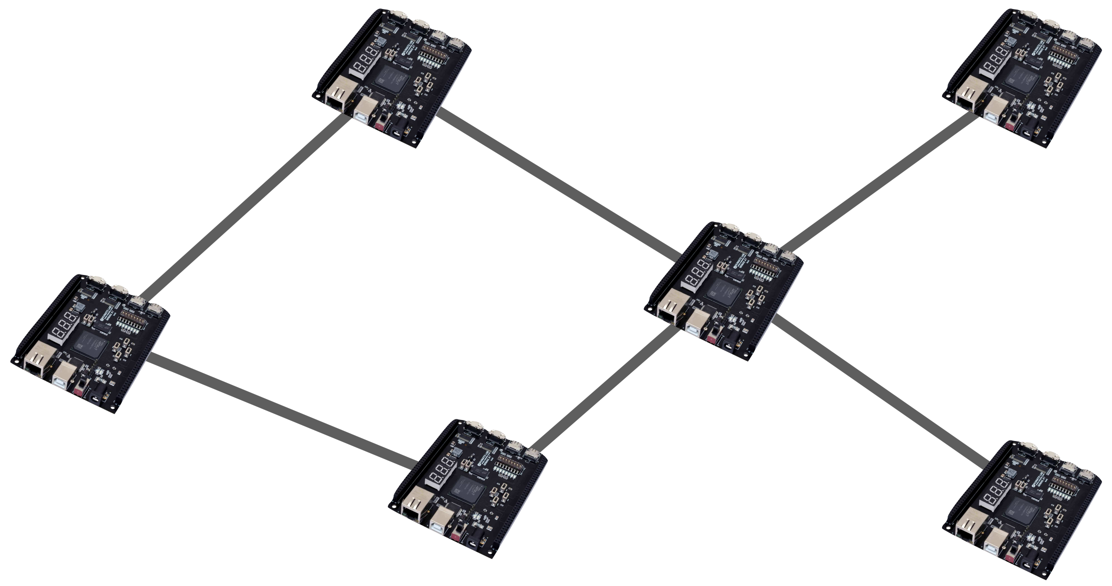
Network virtualization: use case

Imagine you come up with a novel network service, e.g., a new routing protocol, network load-balancer, how would you convince people that this is useful?

Network virtualization: use case

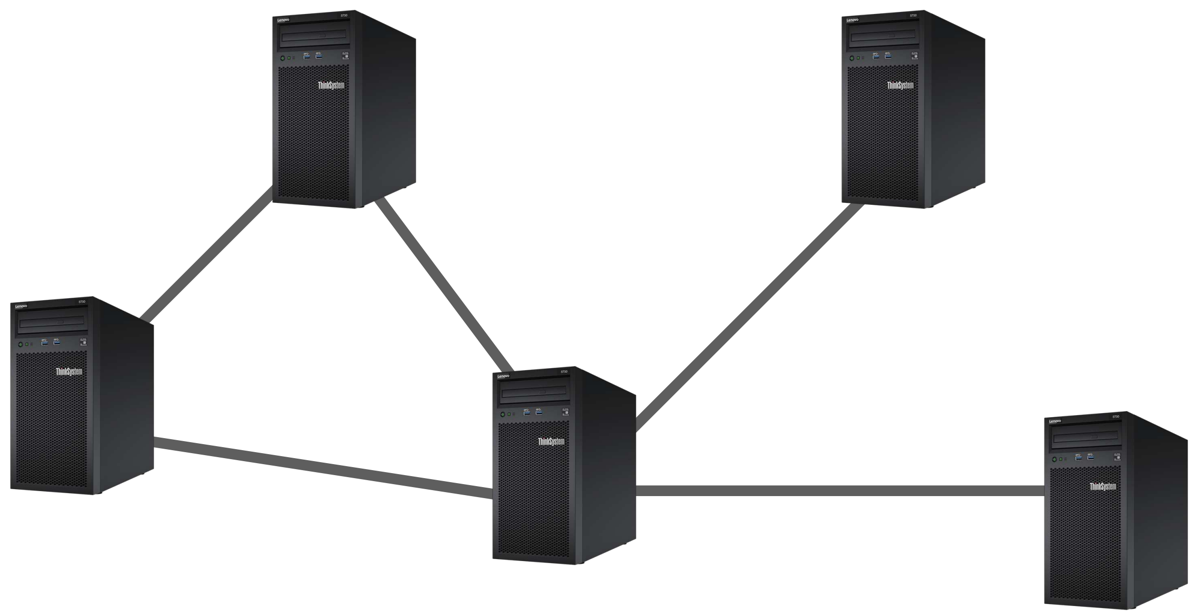
Imagine you come up with a novel network service, e.g., a new routing protocol, network load-balancer, how would you convince people that this is useful?

Hardware testbed



Expensive! Small-scale (fanout is small due to limited port number on NetFPGA)!

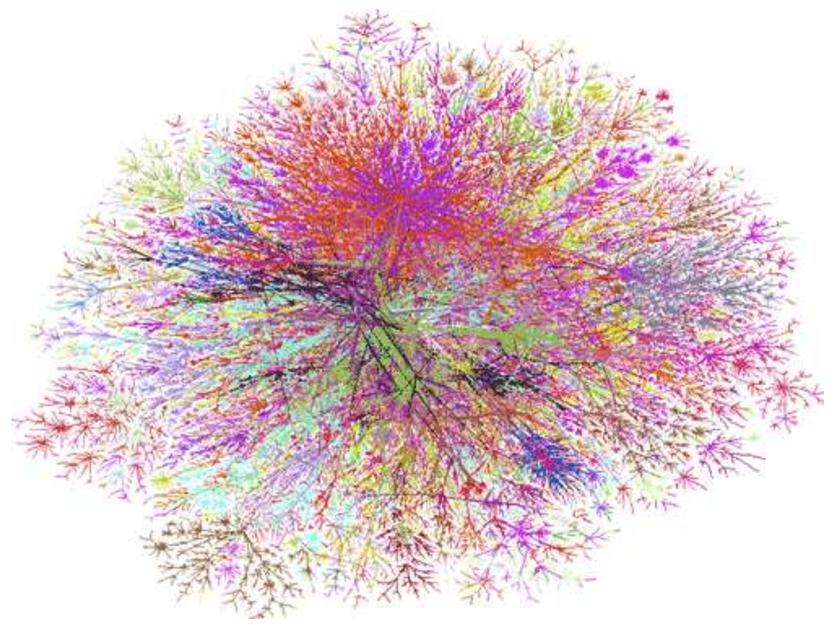
Software testbed



Large-scale (VINI/PlanetLab, Emulab)

Performance is slow (CPU-based), no realistic topology, hard to maintain!

Wild test on the Internet



Convincing network operators to try something new is very difficult! (Outages are the worst)

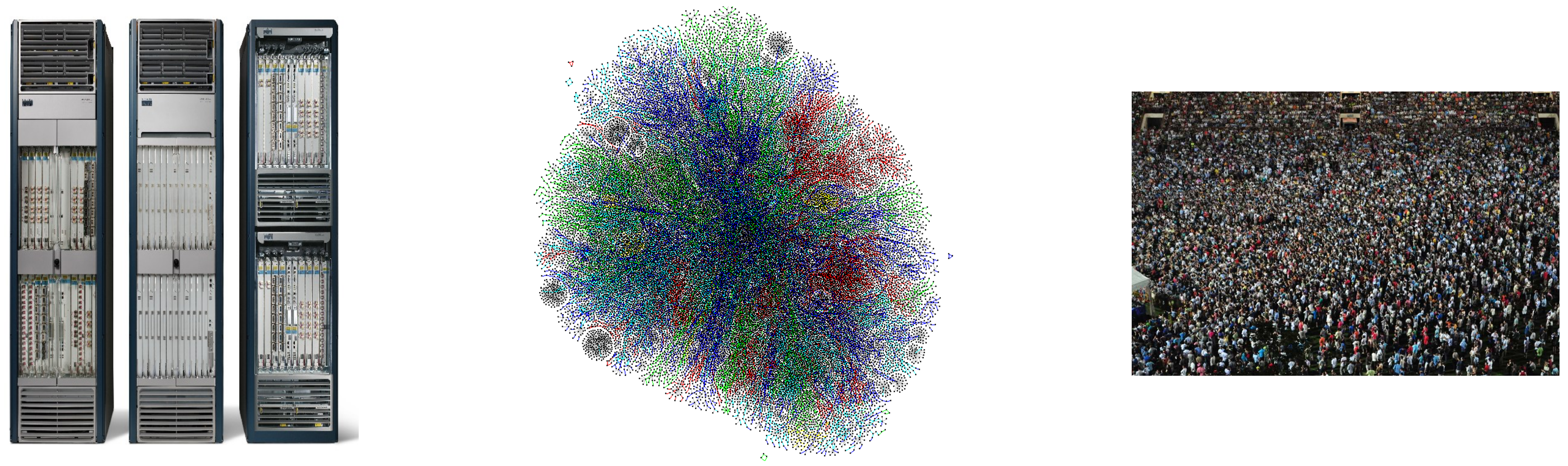
Problem

Realistically evaluating new network services is **hard**

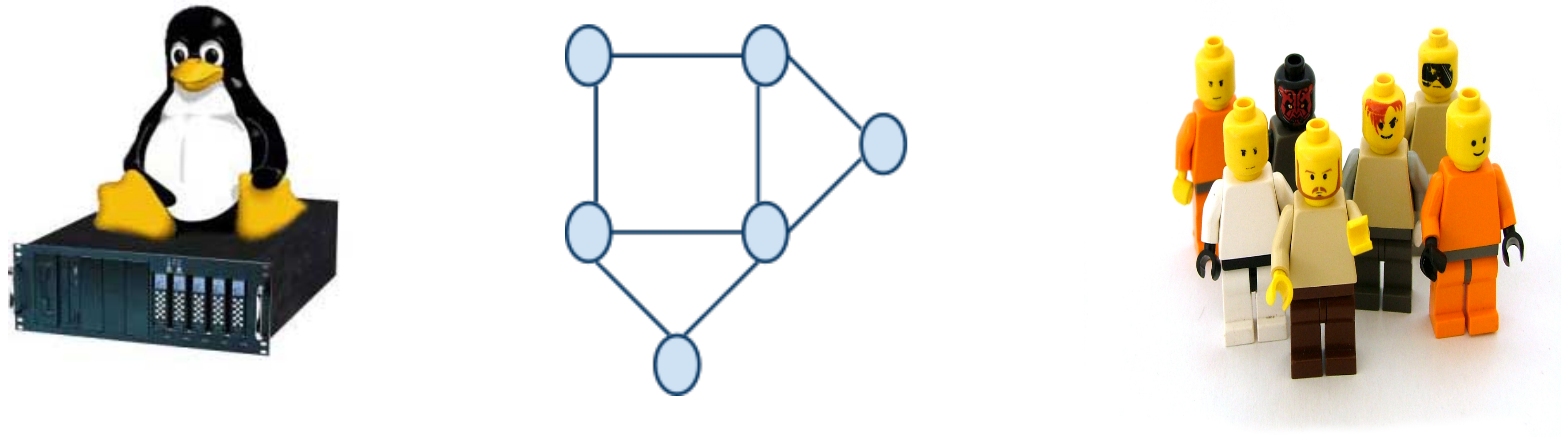
- Services that require changes to switches and routers
- For example: routing protocols, traffic monitoring services, IP mobility

Results

- Many good ideas do not get deployed
- Many deployed services still have bugs



Real networks



Test environments

Solution: network slicing

Divide the production network into logical **slices**

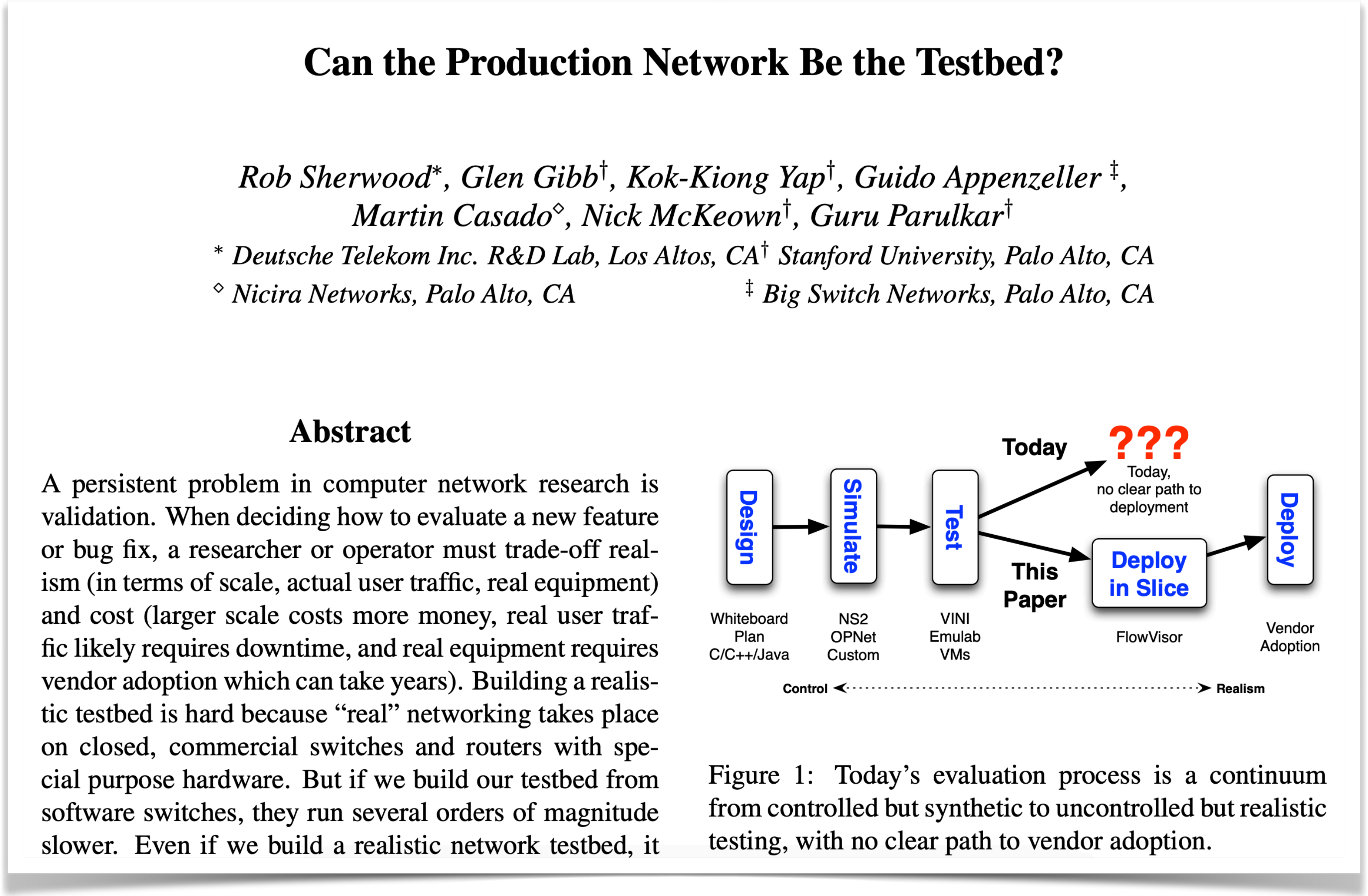
- Each slice/service controls its own packet forwarding
- Users pick which slice controls their traffic: opt-in
- Existing production services run in their own slice: spanning tree, OSPF/BGP

Enforce **strong isolation** between slices

- Actions in one slice do not affect others

Allow the (logical) **testbed** to mirror the **production** network

- Real hardware, performance, topologies, scale, users



Can the Production Network Be the Testbed?

Rob Sherwood, Glen Gibb†, Kok-Kiong Yap†, Guido Appenzeller ‡, Martin Casado◇, Nick McKeown†, Guru Parulkar†*
** Deutsche Telekom Inc. R&D Lab, Los Altos, CA† Stanford University, Palo Alto, CA*
◇ Nicira Networks, Palo Alto, CA ‡ Big Switch Networks, Palo Alto, CA

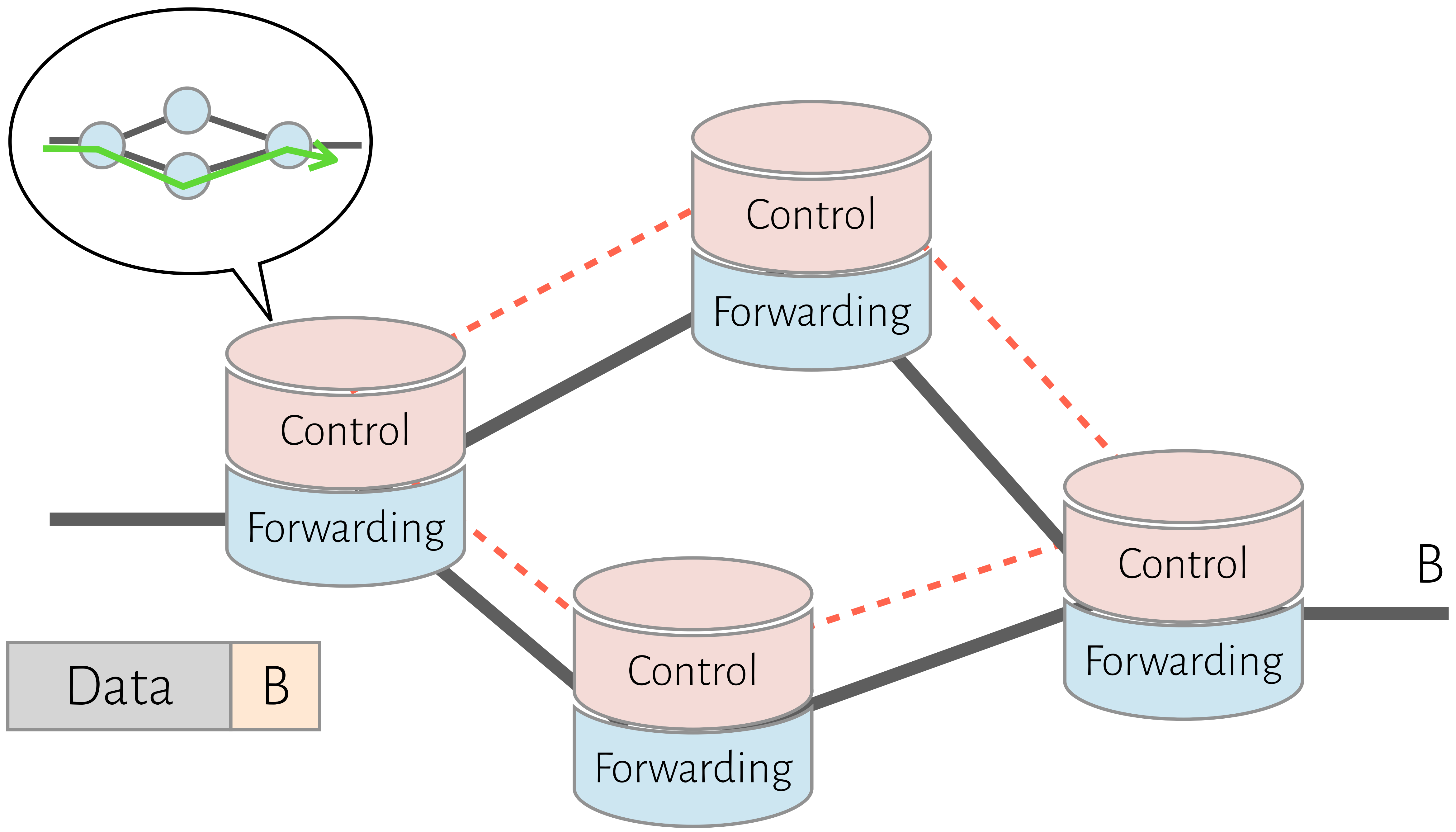
Abstract

A persistent problem in computer network research is validation. When deciding how to evaluate a new feature or bug fix, a researcher or operator must trade-off realism (in terms of scale, actual user traffic, real equipment) and cost (larger scale costs more money, real user traffic likely requires downtime, and real equipment requires vendor adoption which can take years). Building a realistic testbed is hard because "real" networking takes place on closed, commercial switches and routers with special purpose hardware. But if we build our testbed from software switches, they run several orders of magnitude slower. Even if we build a realistic network testbed, it

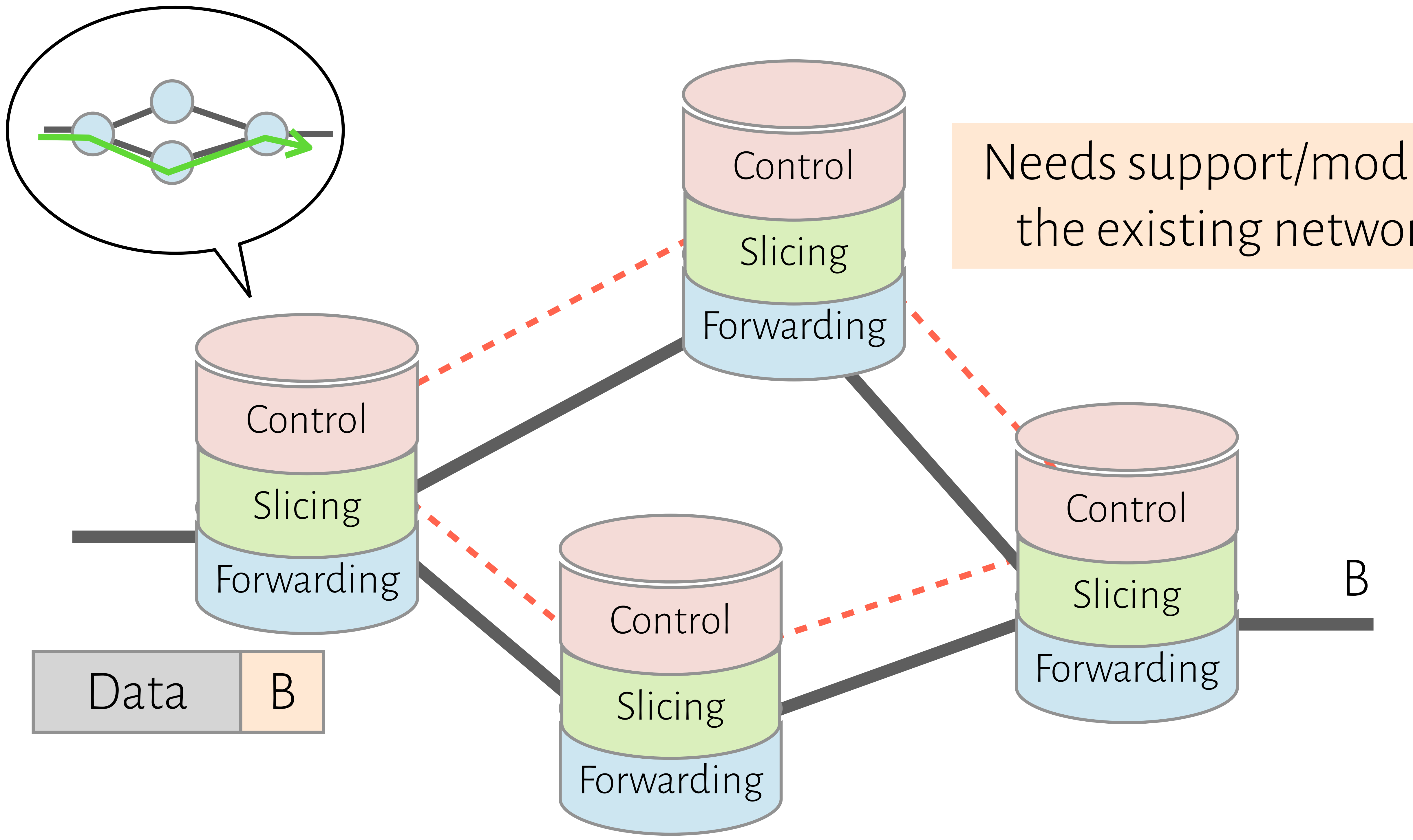
Figure 1: Today's evaluation process is a continuum from controlled but synthetic to uncontrolled but realistic testing, with no clear path to vendor adoption.

USENIX OSDI 2010

Traditional network

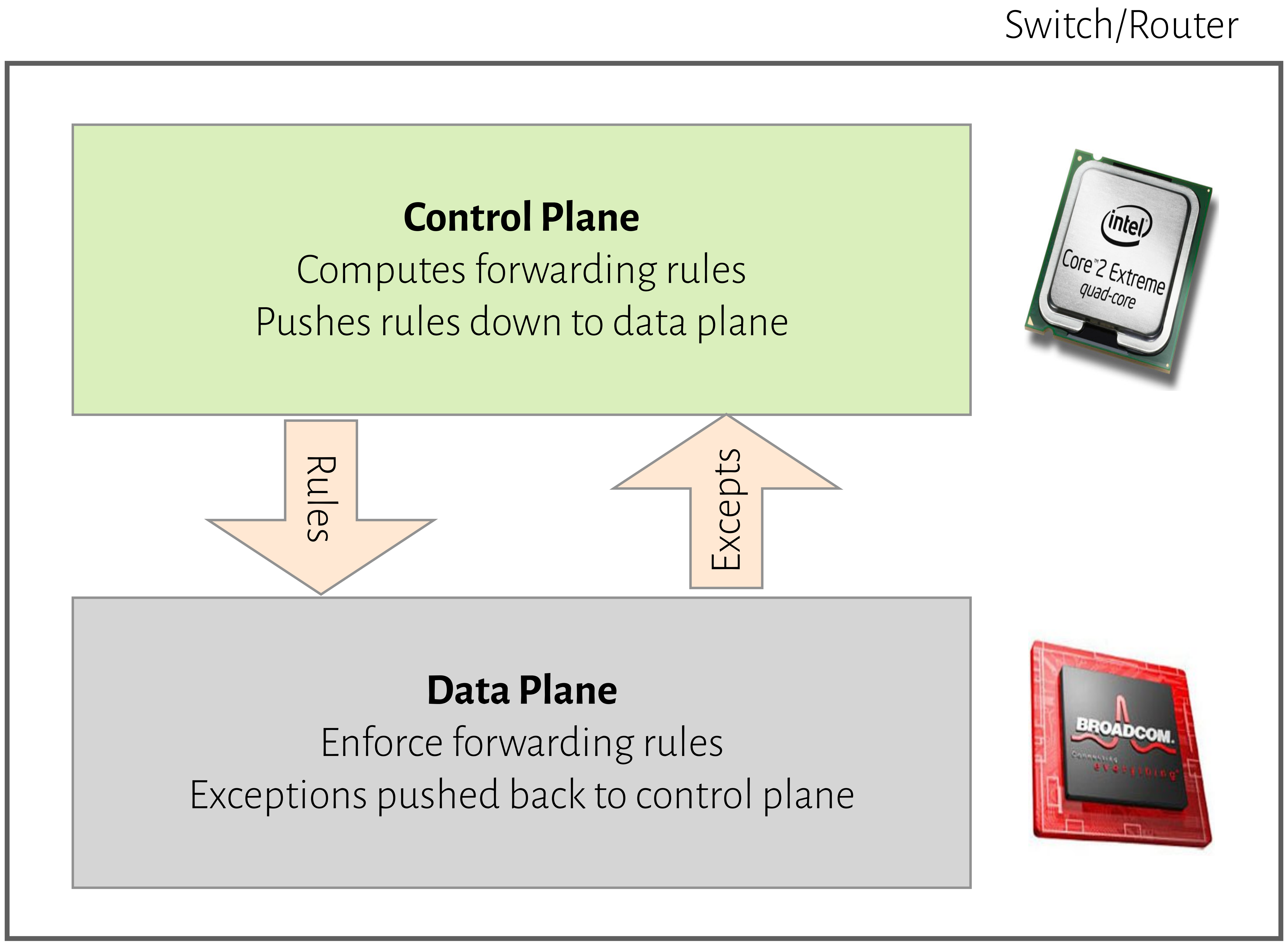


Slicing a traditional network

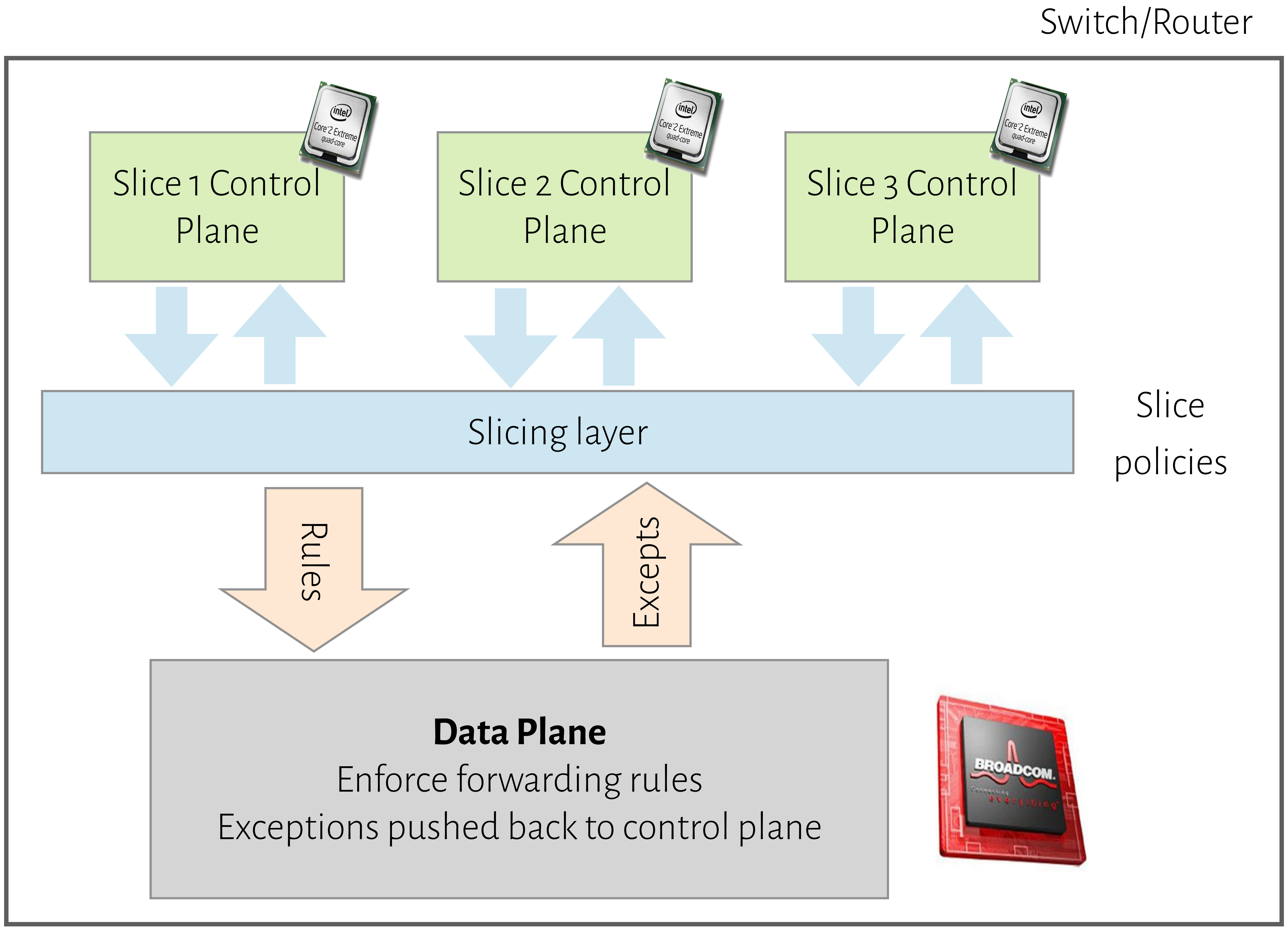


Needs support/modification on the existing network devices

Current network devices



Slicing layer



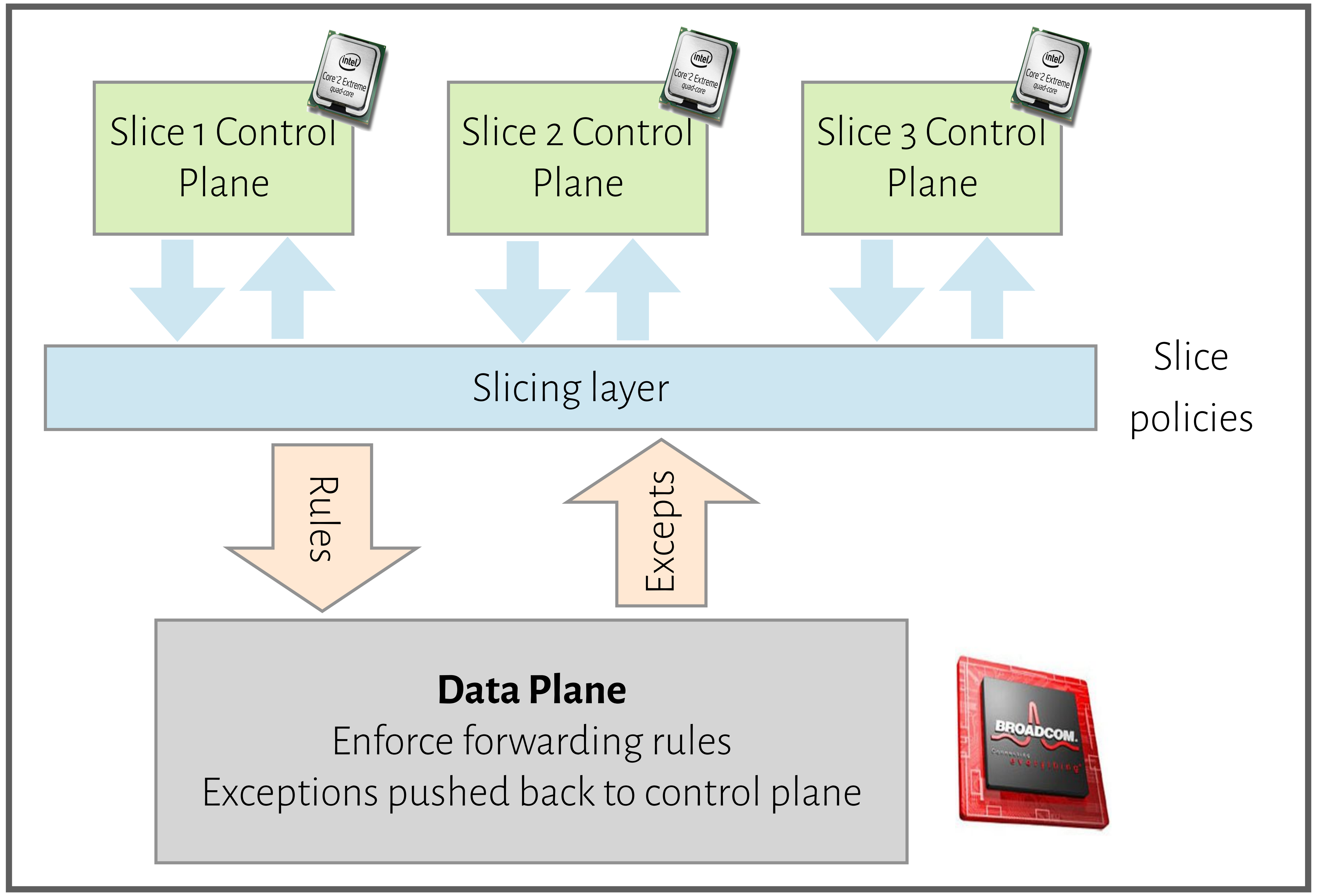
Network slicing architecture

A network slice is a collection of sliced switches/routers

- Data plane is unmodified
- Packets forward with **no performance penalty**
- Slicing with existing ASIC

Transparent slicing layer

- Each slice believes it owns the data path
- Enforces isolation between slides: rewrites, drops rules to adhere to slice policy
- Forwards exceptions to correct slice(s)



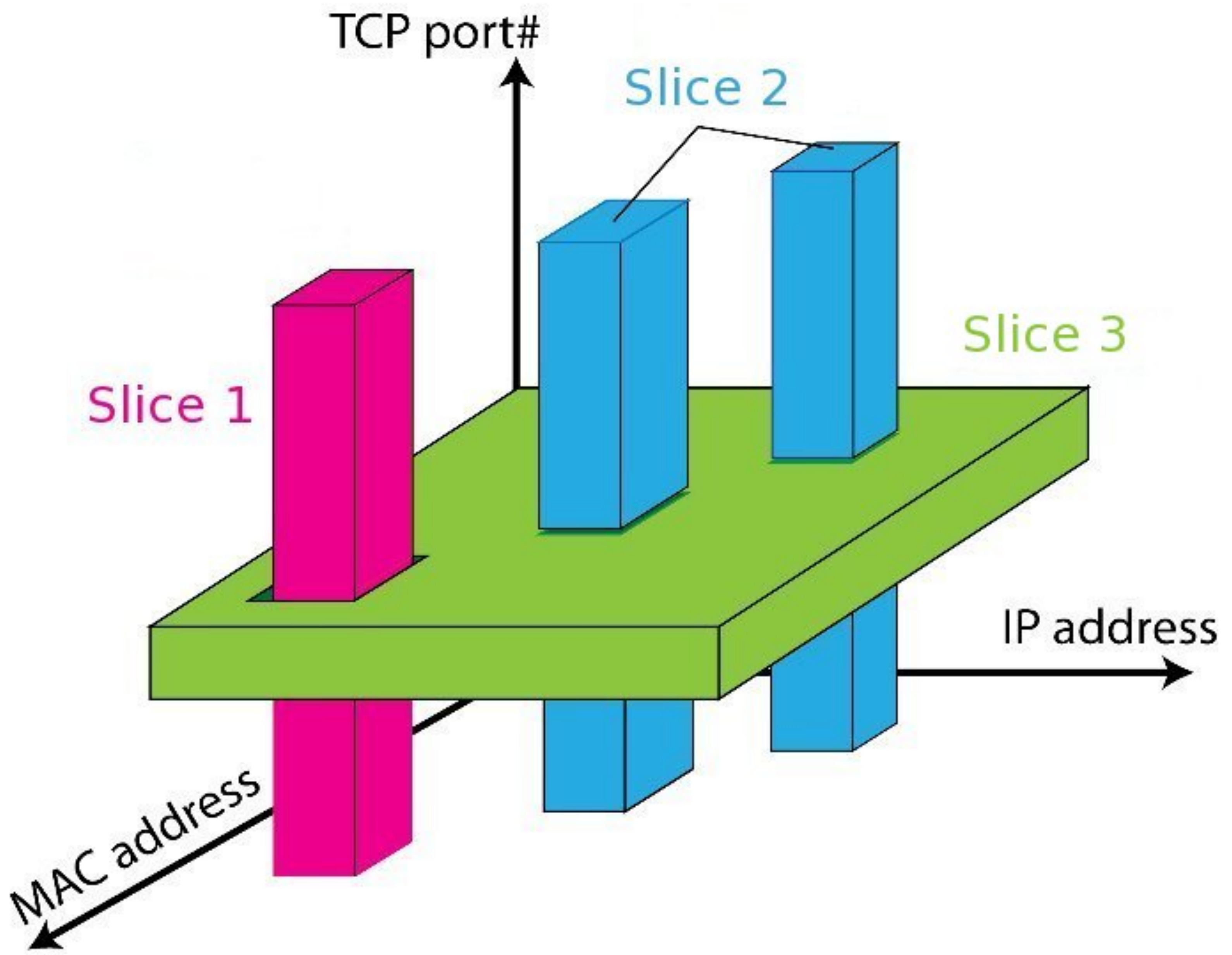
Slicing policies

The slicing policy specifies the **resource limit** for each slice:

- Link bandwidth
- Maximum number of forwarding rules (on switches)
- Topology
- Fraction of switch/router CPU

FlowSpace: which packet does the slice control?

- Maps packets to slices according to their "classes" defined by the packet header fields



Real user traffic: opt-in

Allow users to opt-in to services in real time

- Users can delegate control of individual flows to slices
- Add new FlowSpace to each slice's policy

Examples

- "Slice 1 will handle my HTTP traffic"
- "Slice 2 will handle my VoIP traffic"
- "Slice 3 will handle everything else"

Creates incentives for building high-quality services!

Source: gacovinolack.com

Slice definition

Bob's experimental slice: all HTTP traffic to/from users who opted in

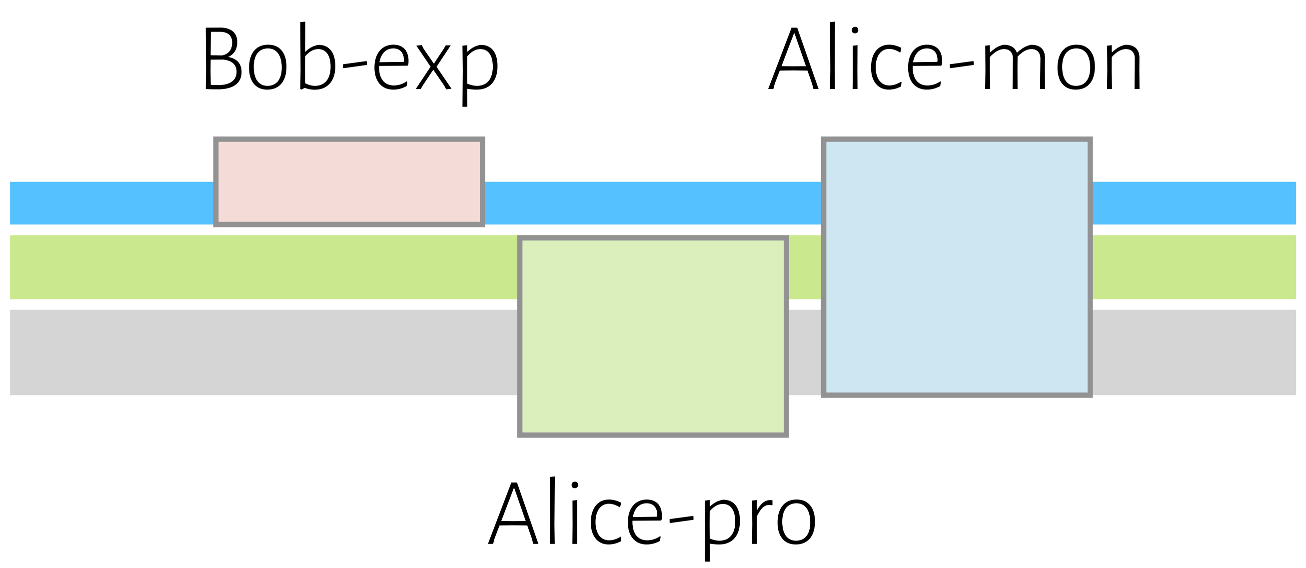
- Allow: tcp_port=80 and ip=user_ip

Alice's production slice: complementary to Bob's slice

- Deny: tcp_port=80 and ip=user_ip
- Allow: all

Alice's monitoring slice: all traffic in all slices

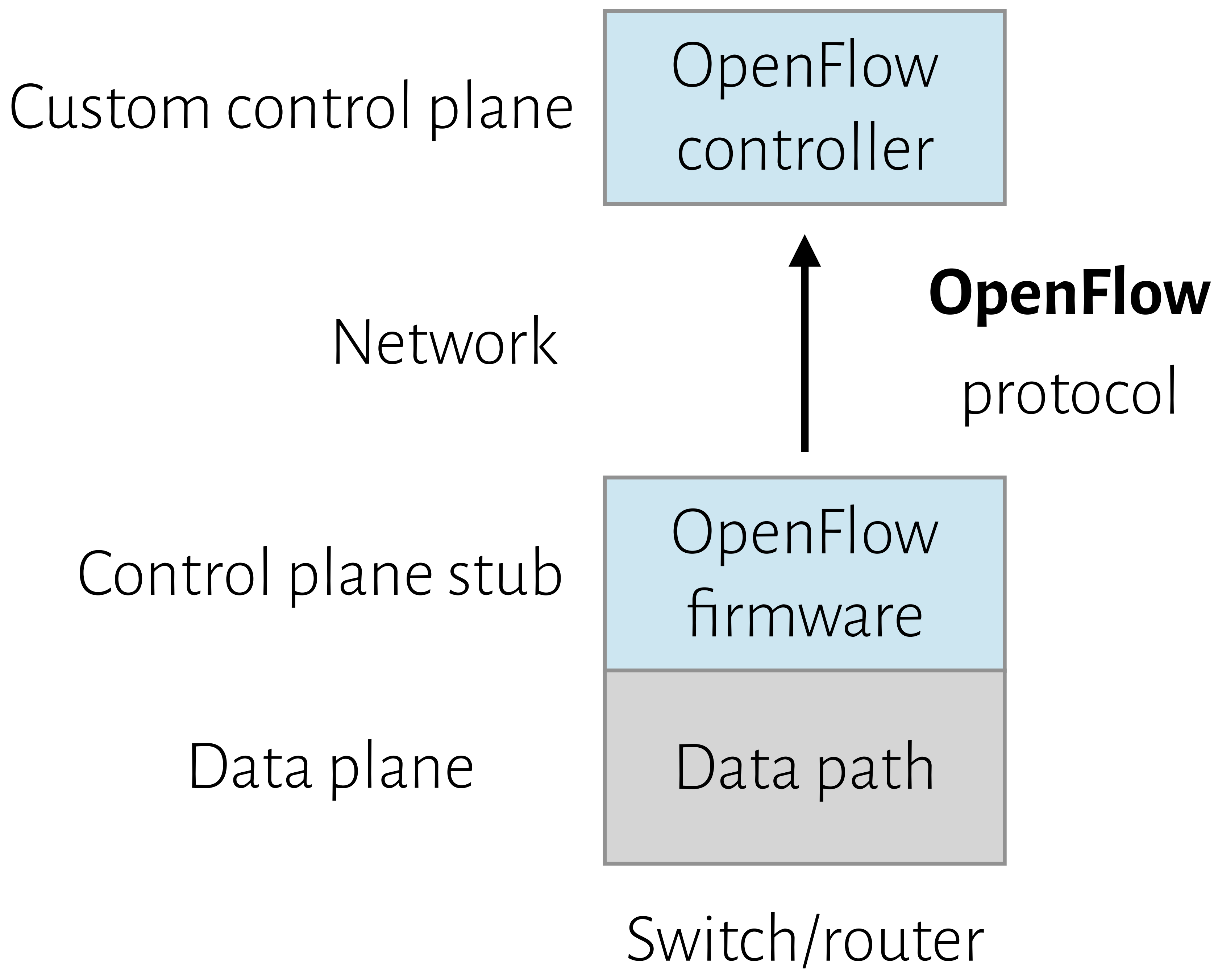
- Read-only: all



Slicing with OpenFlow

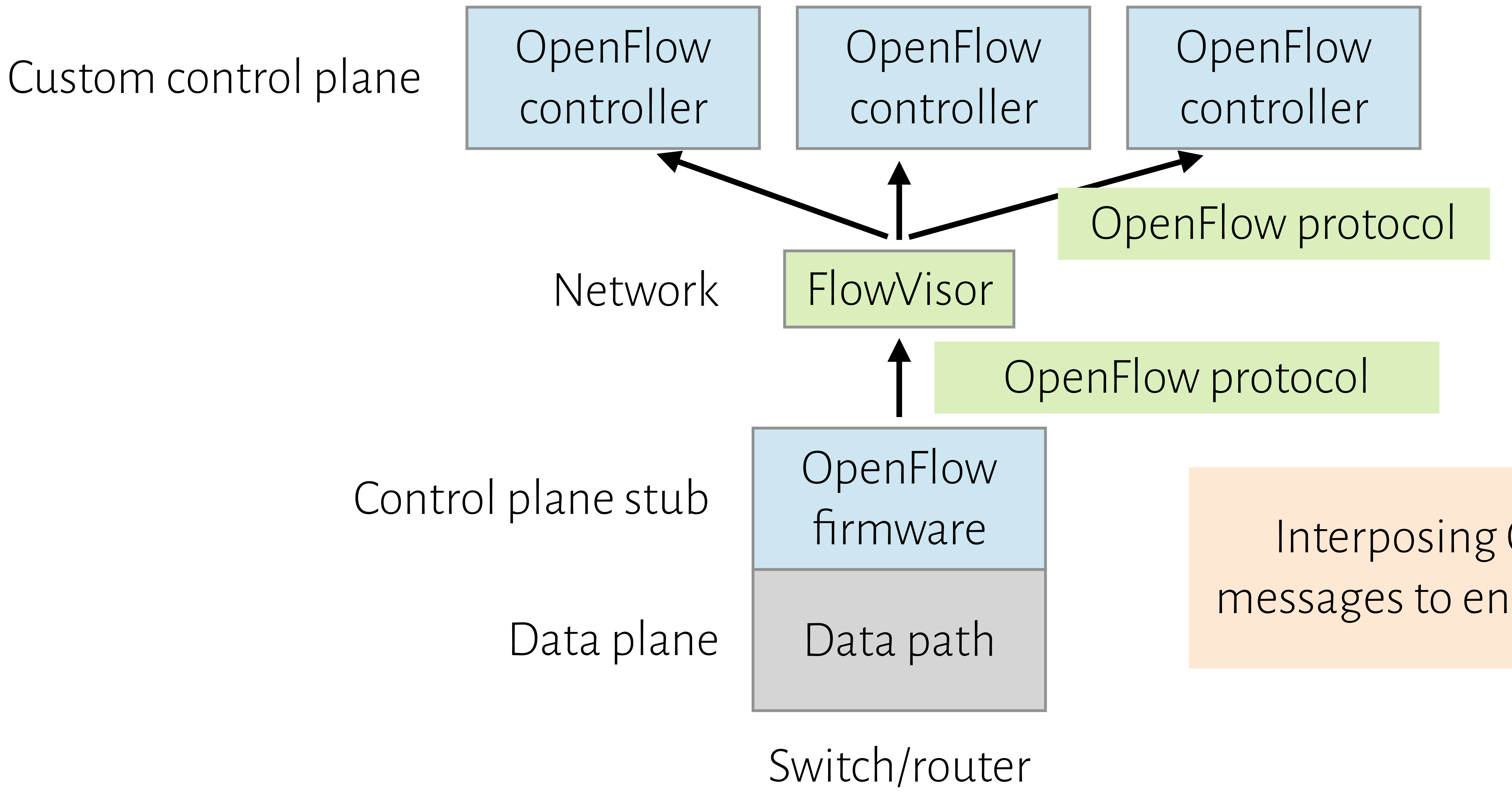
Recall OpenFlow:

- API for controlling packet forwarding
- Abstraction of control/data plane protocols
- Works on commodity hardware (via firmware upgrade)



How should we slice an OpenFlow-based software defined network?

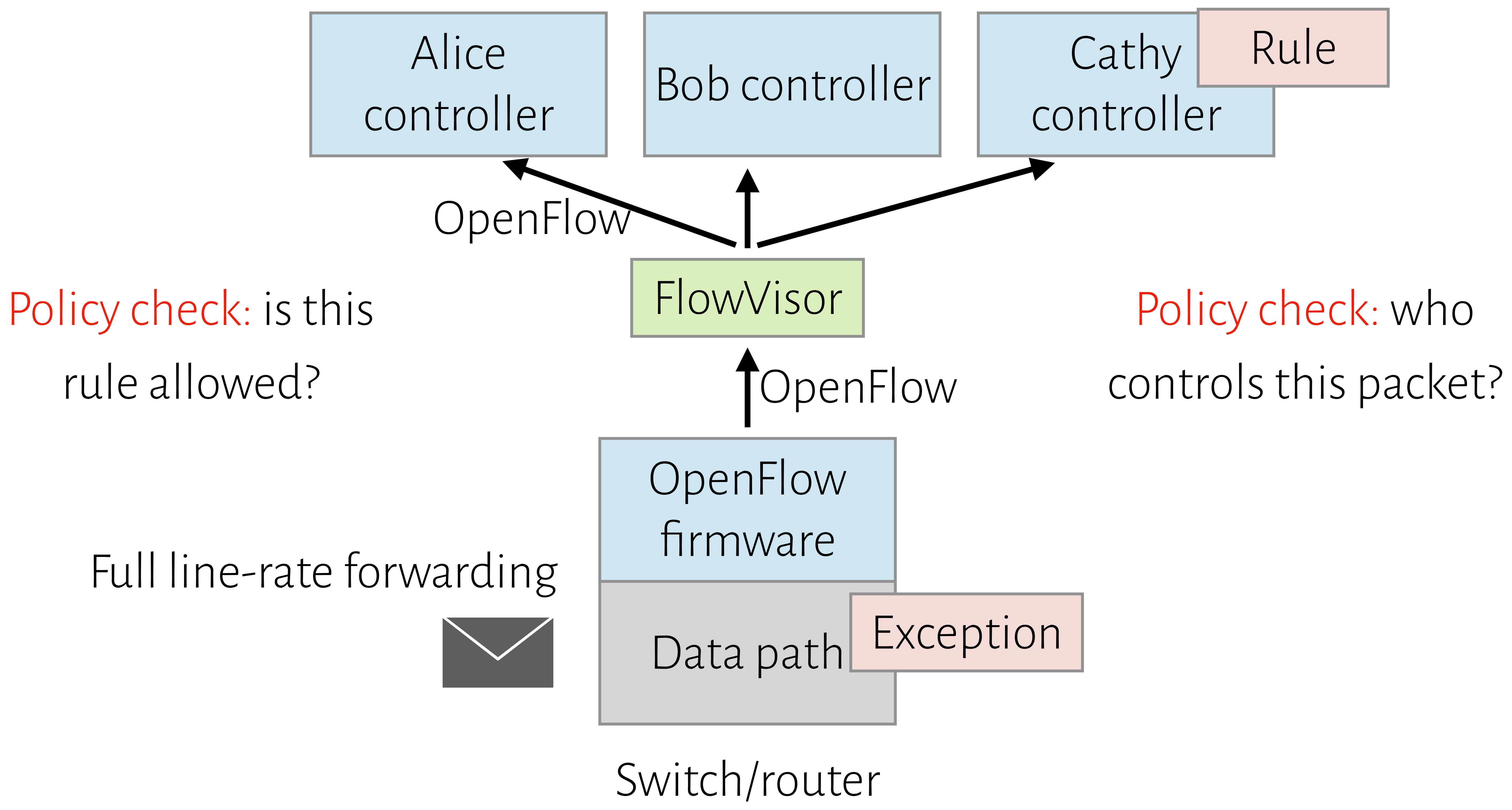
FlowVisor



Interposing OpenFlow control messages to enforce network slicing

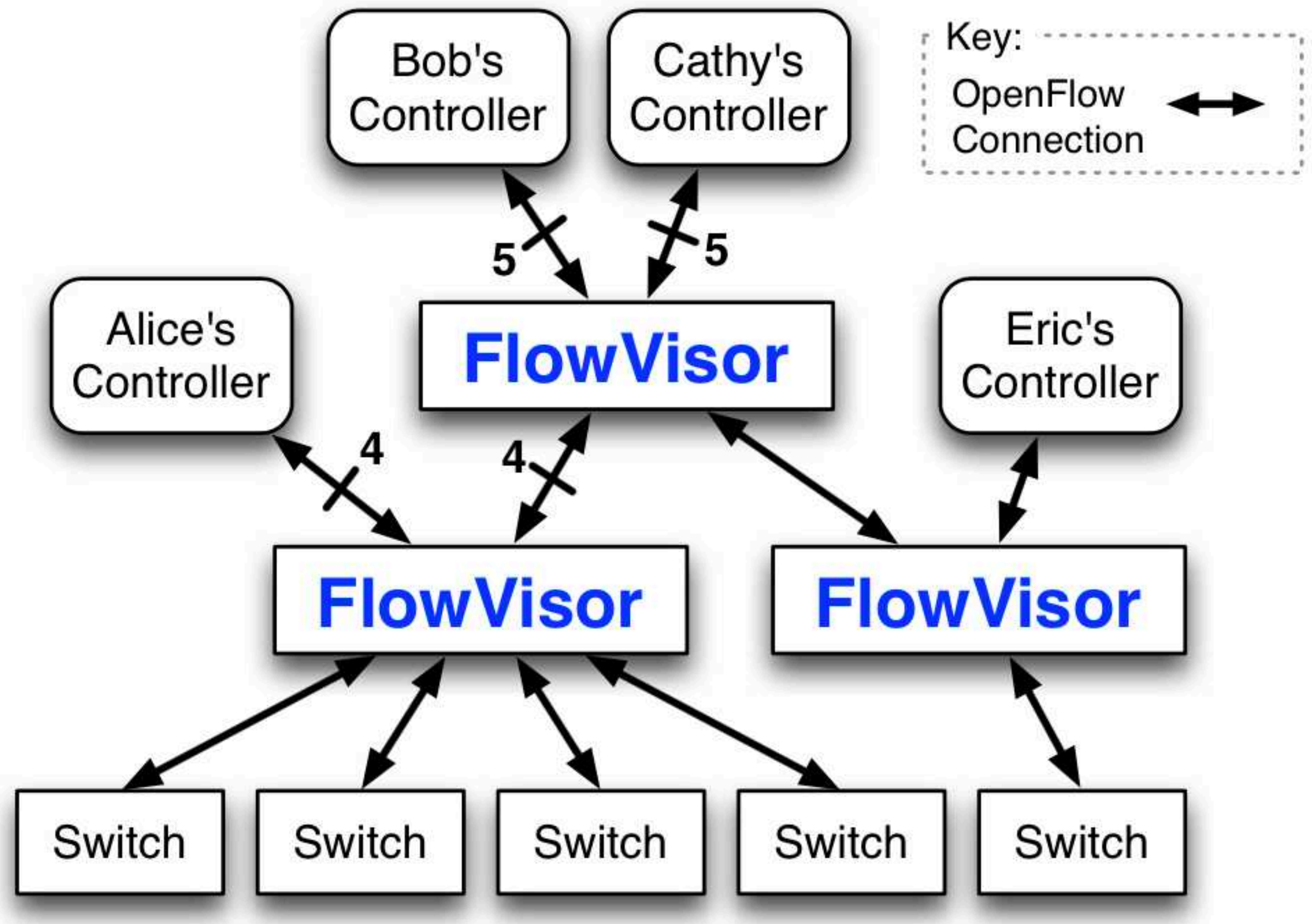
https://github.com/opennetworkinglab/flowvisor

FlowVisor packet handling



Recursive slicing

FlowVisor can trivially recursively slice an already sliced network, creating hierarchies of FlowVisors

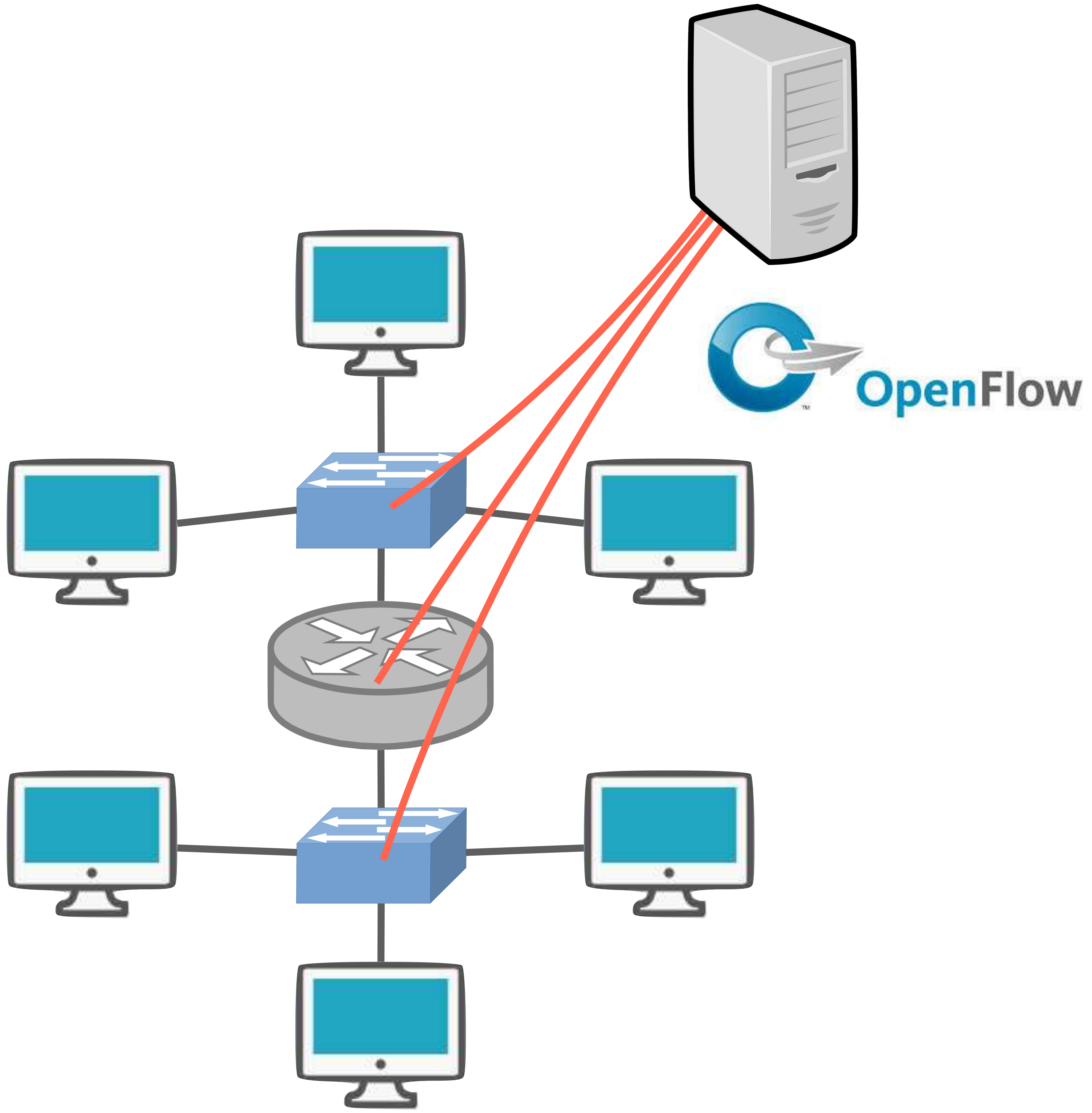


Questions?

Summary

Lecture 8: Software defined networking

- Complexities in network control
- Software defined networking idea
- Abstractions in SDN
- Network virtualization, FlowVisor
- Mininet for experimentation



Next week: programmable data plane

Limitations of OpenFlow

- Bounded to existing protocols
- New protocols/packet formats not supported

How to support arbitrary packet format and enable a fully programmable network?