6.888
Lecture 14:
Software Defined Networking

Mohammad Alizadeh

✧ Many thanks to Nick McKeown (Stanford), Jennifer Rexford (Princeton), Scott Shenker (Berkeley), Nick Feamster (Princeton), Li Erran Li (Columbia), Yashar Ganjali (Toronto)

Spring 2016

Outline

What is SDN?

OpenFlow basics

Why is SDN happening now? (a brief history)

4D discussion

What is SDN?

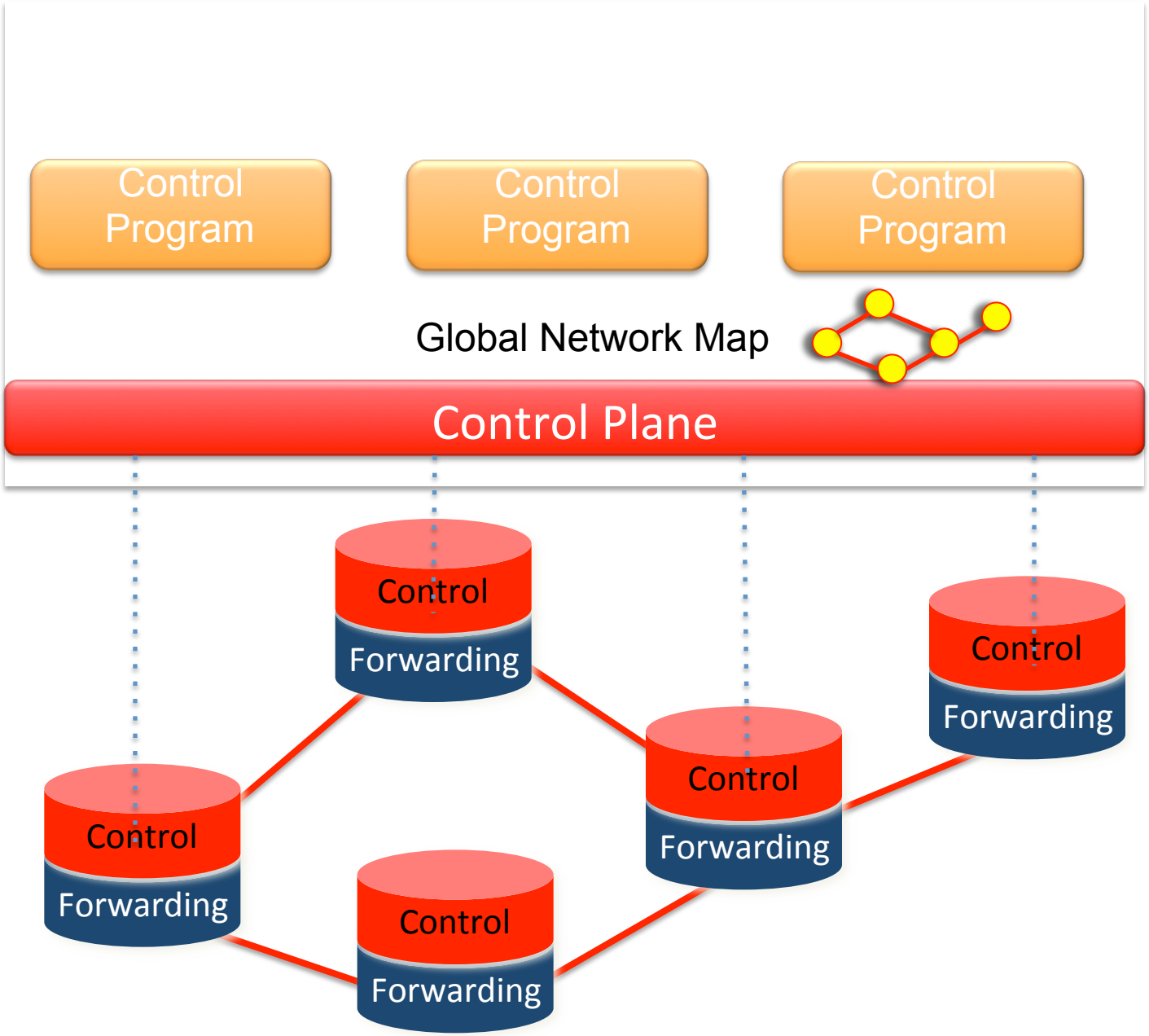
Software Defined Network

A network in which the control plane is physically separate from the data plane.

and

A single (logically centralized) control plane controls several forwarding devices.

Software Defined Network (SDN)



What You Said

“Overall, the idea of SDN feels a little bit unsettling to me because it is proposing to change one of the main reasons for the success of computer networks: fully decentralized control. Once we introduce a centralized entity to control the network we have to make sure that it doesn’t fail, which I think is very difficult.”

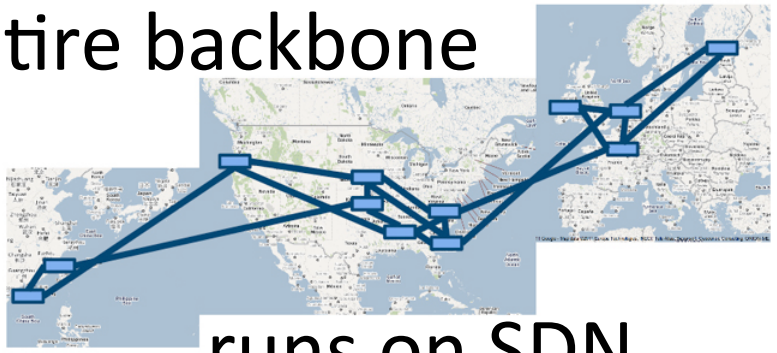
A Major Trend in Networking

OPEN NETWORKING FOUNDATION

Deutsche Telekom · facebook · Goldman Sachs · Google · Microsoft · NTT Communications · verizon · YAHOO!

Google

Entire backbone



runs on SDN

nicira

Bought for **$1.2 billion**
(mostly cash)

The Networking "Planes"

Data plane: processing and delivery of packets with local forwarding state

- Forwarding state + packet header → forwarding decision
- Filtering, buffering, scheduling

Control plane: computing the forwarding state in routers

- Determines how and where packets are forwarded
- Routing, traffic engineering, failure detection/recovery, …

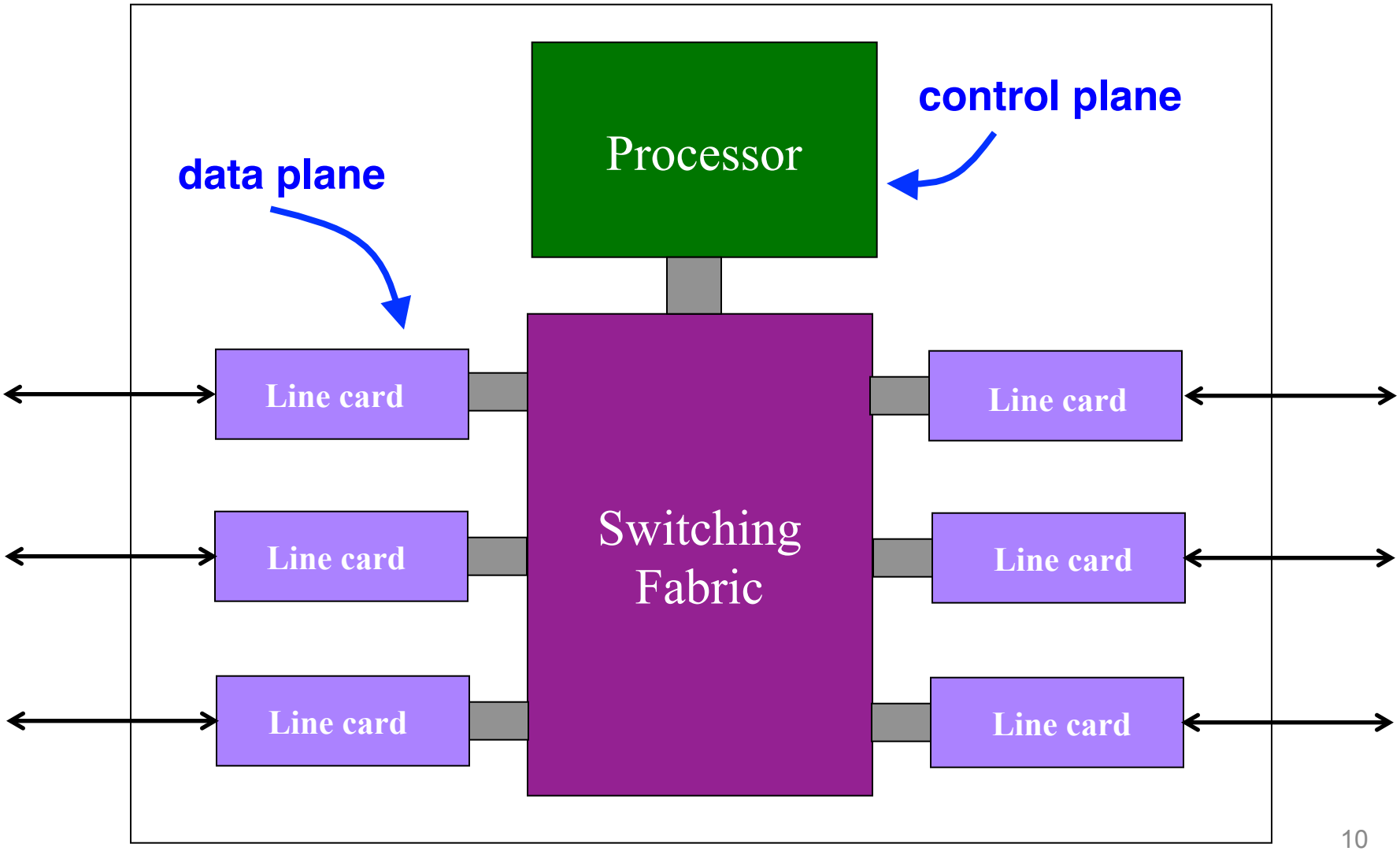
Management plane: configuring and tuning the network

- Traffic engineering, ACL config, device provisioning, …

Timescales

	Data	Control	Management
Time-scale	Packet (nsec)	Event (10 msec to sec)	Human (min to hours)
Location	Linecard hardware	Router software	Humans or scripts

Data and Control Planes

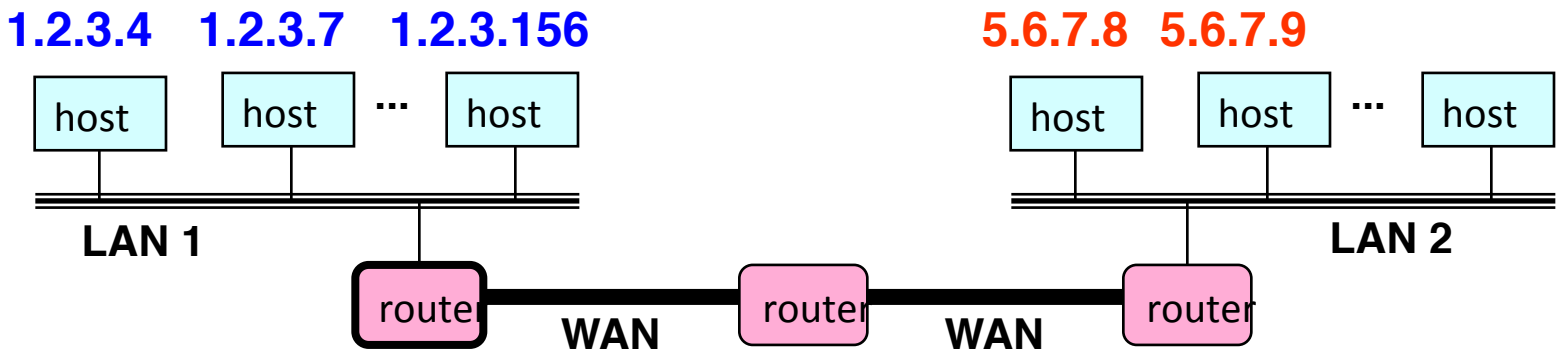


Data Plane

Streaming algorithms on packets

- Matching on some header bits
- Perform some actions

Example: IP Forwarding



1.2.3.0/24	←
5.6.7.0/24	→

forwarding table

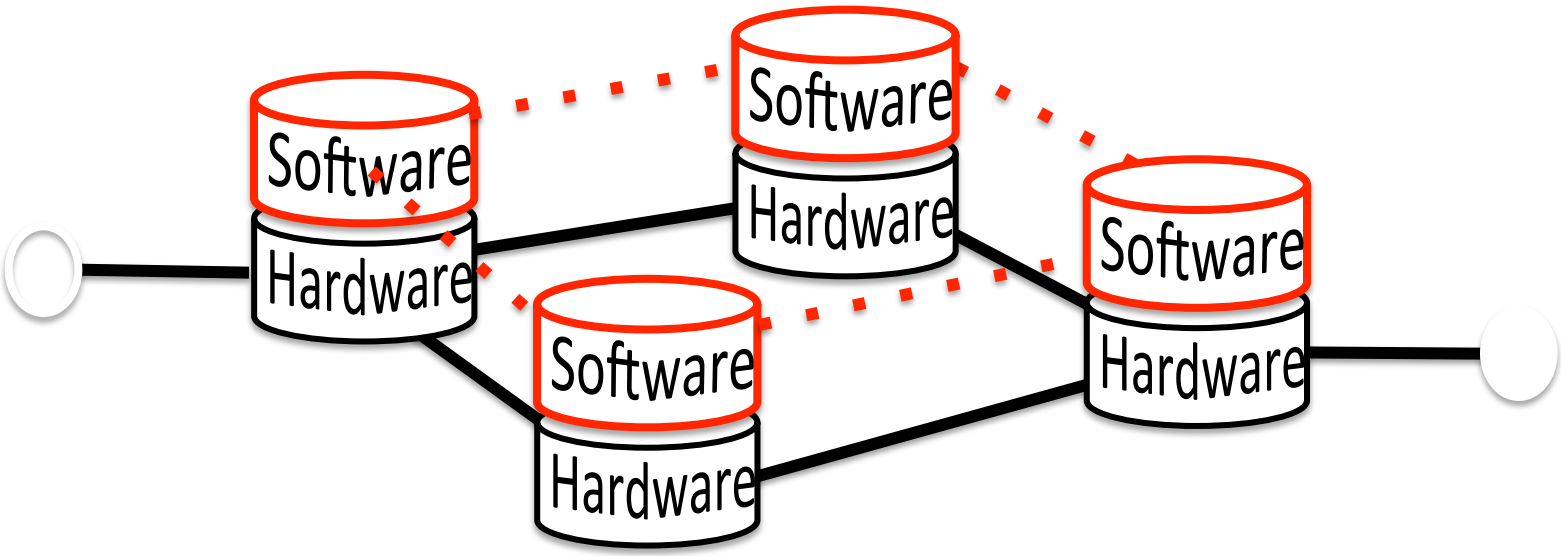
Control Plane

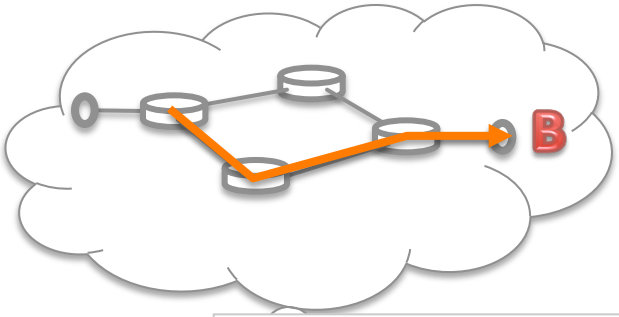
Compute paths the packets will follow

- Populate forwarding tables
- Traditionally, a distributed protocol

Example: Link-state routing (OSPF, IS-IS)

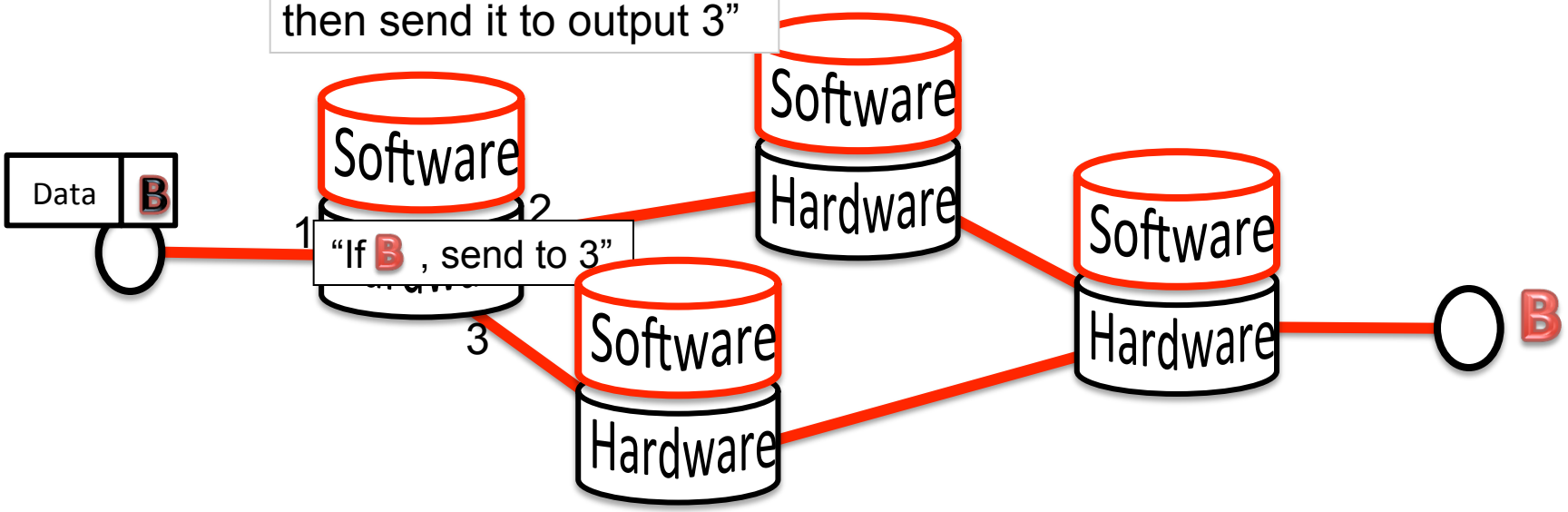
- Flood the entire topology to all nodes
- Each node computes shortest paths
- Dijkstra's algorithm


Software
Hardware
Software
Hardware
Software
Hardware
Software
Hardware



1. Figure out which routers and links are present.
2. Run Dijkstra's algorithm to find shortest paths.

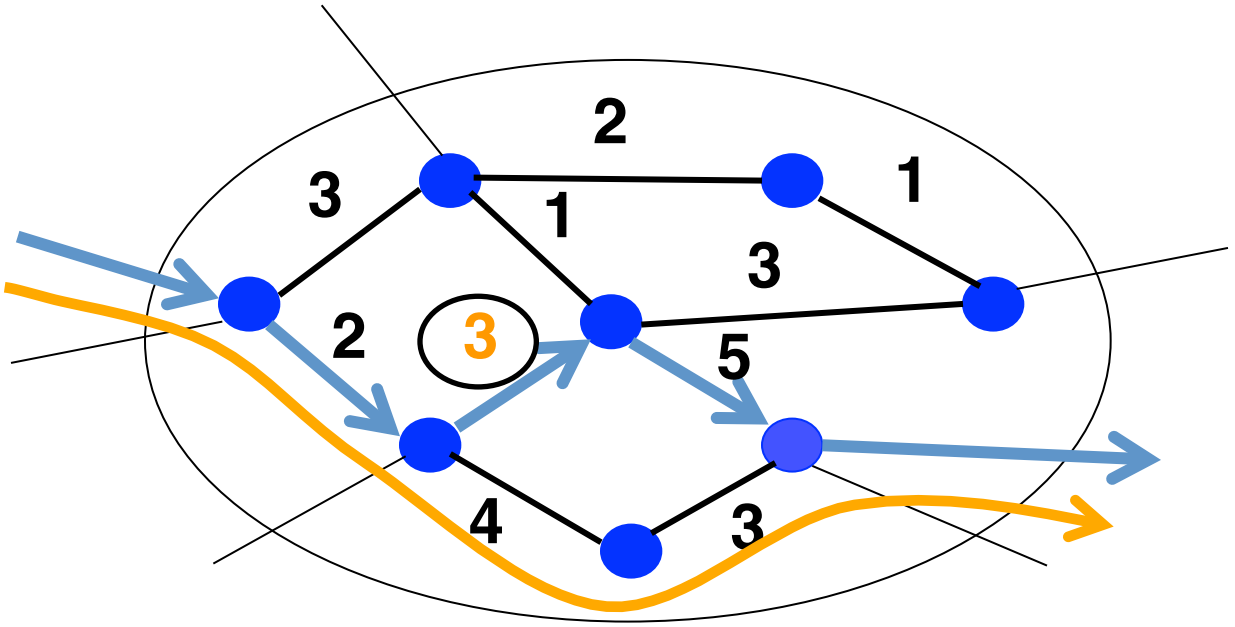
"If a packet is going to B, then send it to output 3"



Management Plane

Traffic Engineering: setting the weights

- Inversely proportional to link capacity?
- Proportional to propagation delay?
- Network-wide optimization based on traffic?



Challenges

(Too) many task-specific control mechanisms

- No modularity, limited functionality

Indirect

- Mu ou want
- Ex.

Uncoord

- Ca

Interacting protocols and mechanisms

- Routing, addressing, access control, QoS

The network is

- Hard to reason about
- Hard to evolve
- Expensive

Example 1: Inter-domain Routing

Today's inter-domain routing protocol, BGP, artificially constrains routes

- Routing only on **destination IP address blocks**
- Can only influence **immediate neighbors**
- Very difficult to incorporate other information

Application-specific peering

- Route video traffic one way, and non-video another

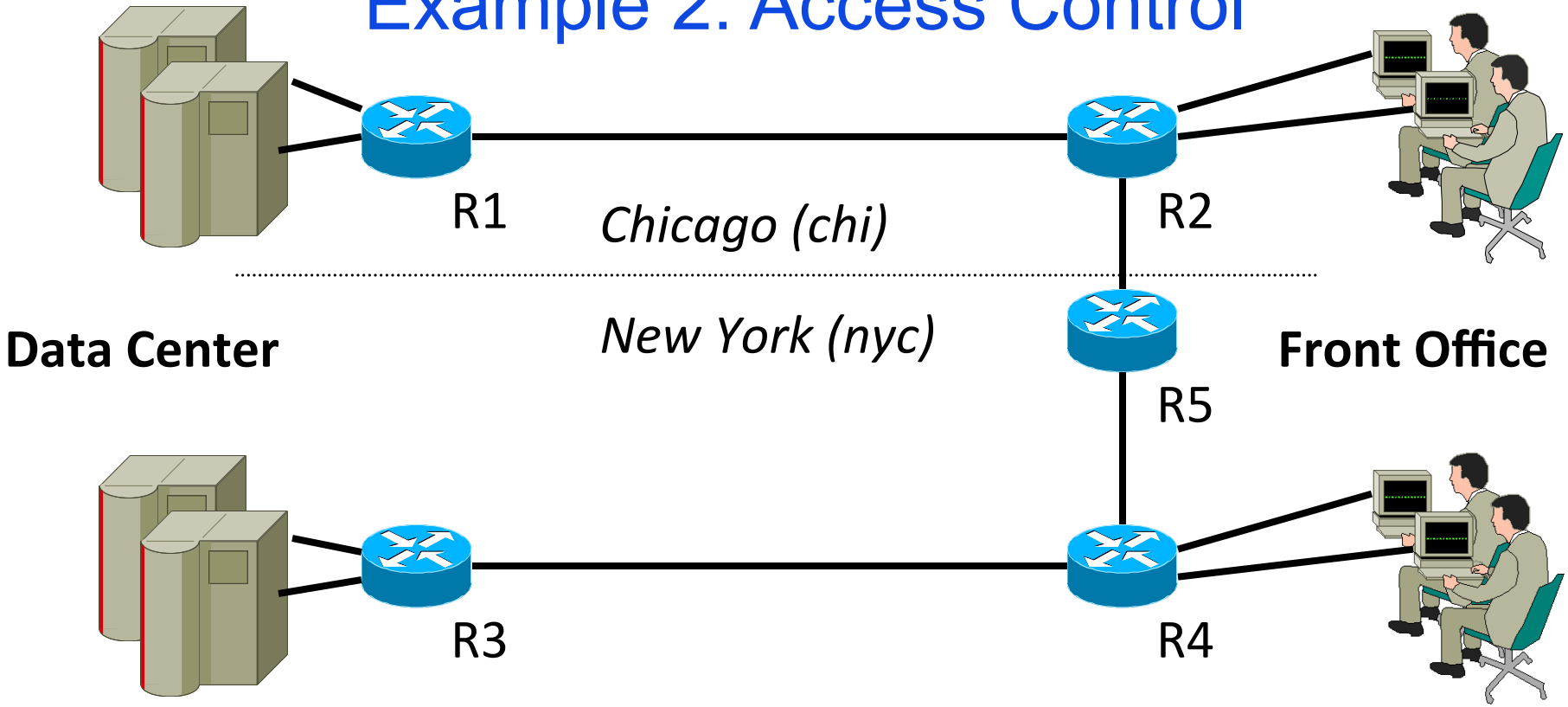
Blocking denial-of-service traffic

- Dropping unwanted traffic further upstream

Inbound traffic engineering

- Splitting incoming traffic over multiple peering links

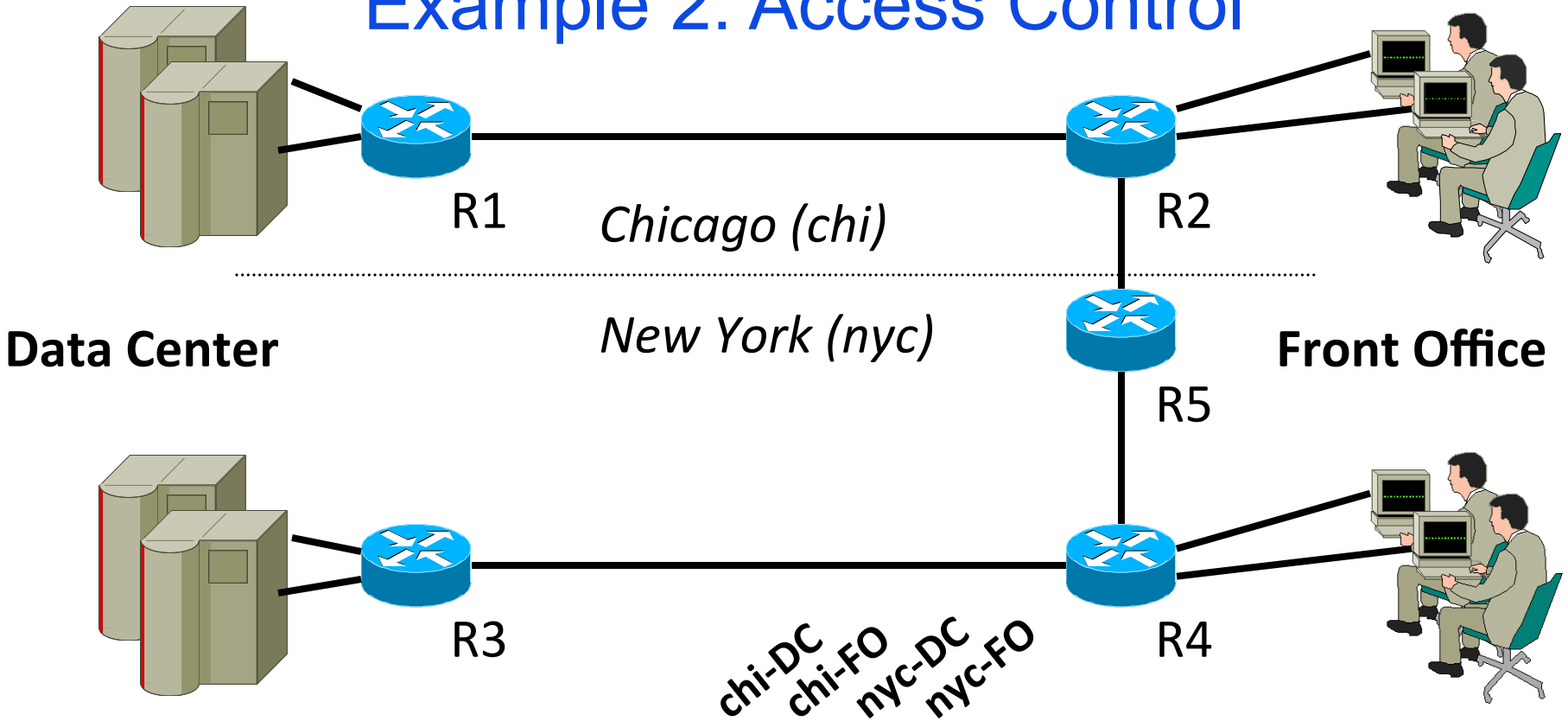
Example 2: Access Control



Two locations, each with data center & front office

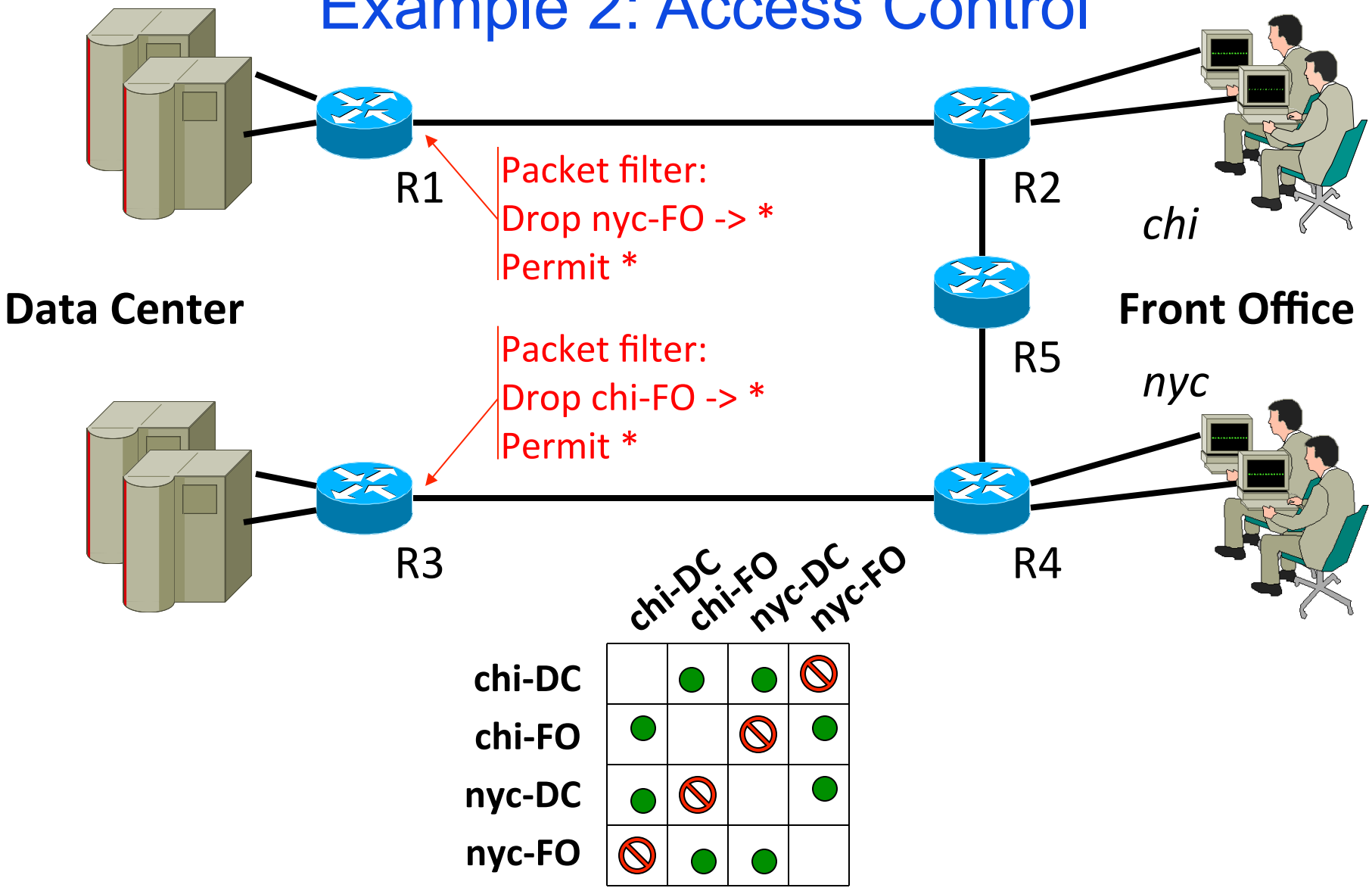
All routers exchange routes over all links

Example 2: Access Control

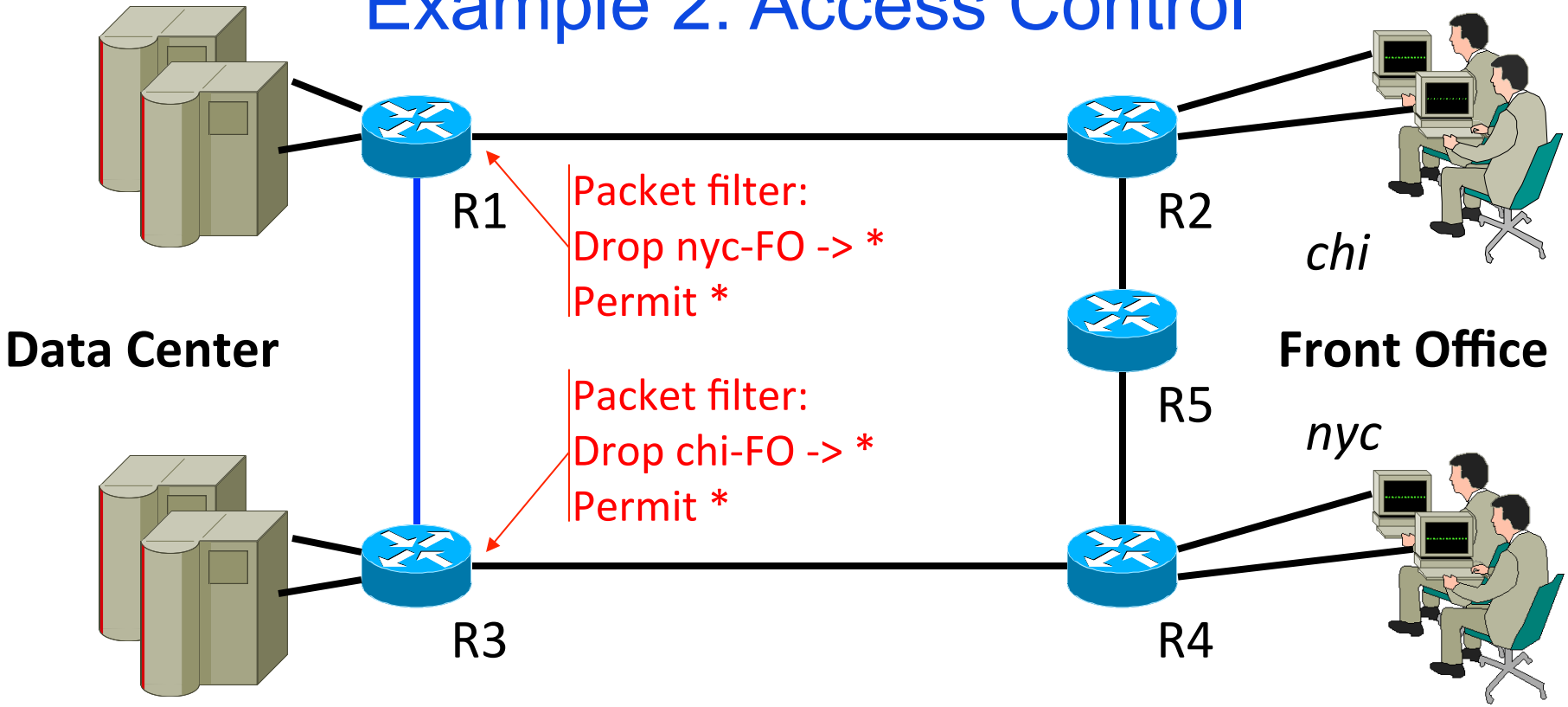


	chi-DC	chi-FO	nyc-DC	nyc-FO
chi-DC		●	●	⊘
chi-FO	●		⊘	●
nyc-DC	●	⊘		●
nyc-FO	⊘	●	●	

Example 2: Access Control



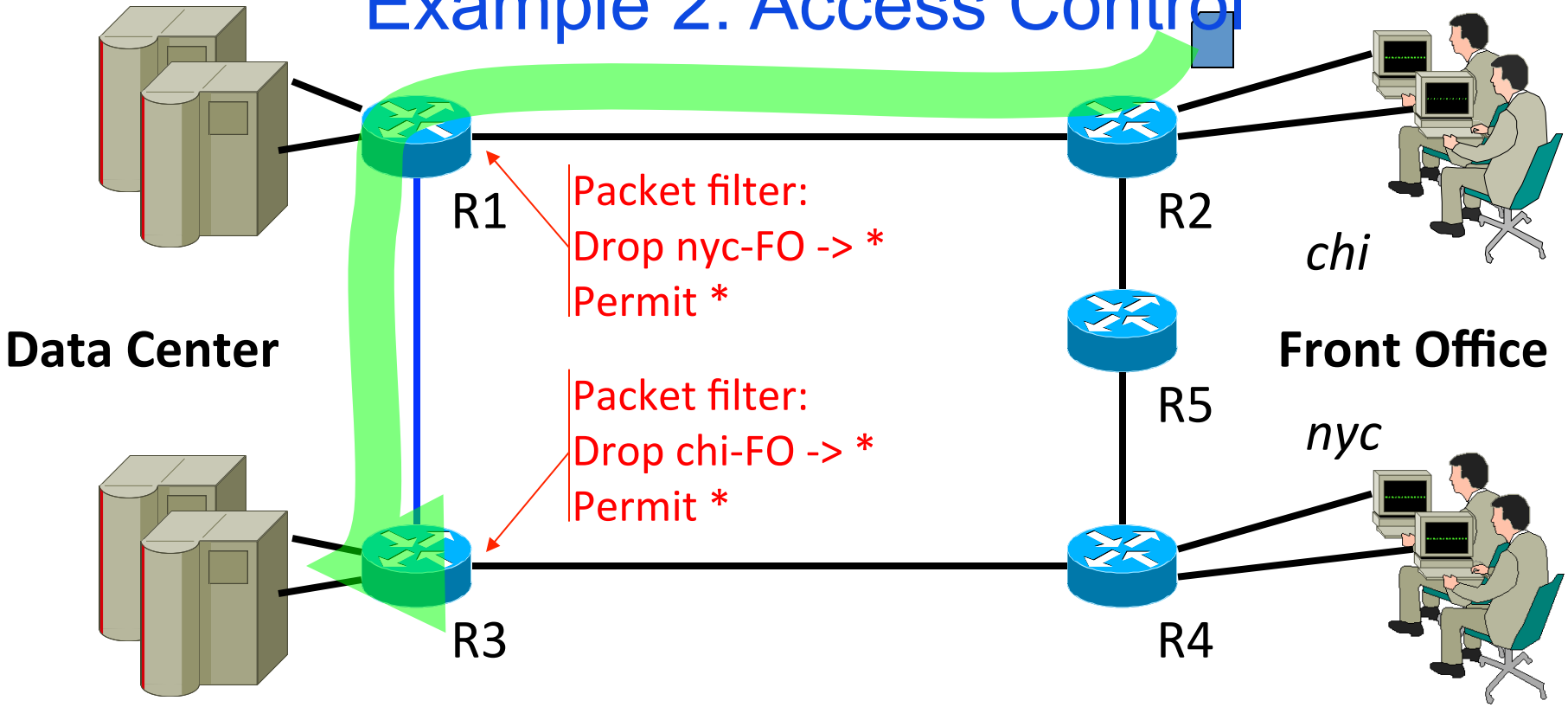
Example 2: Access Control



A new short-cut link added between data centers

Intended for backup traffic between centers

Example 2: Access Control

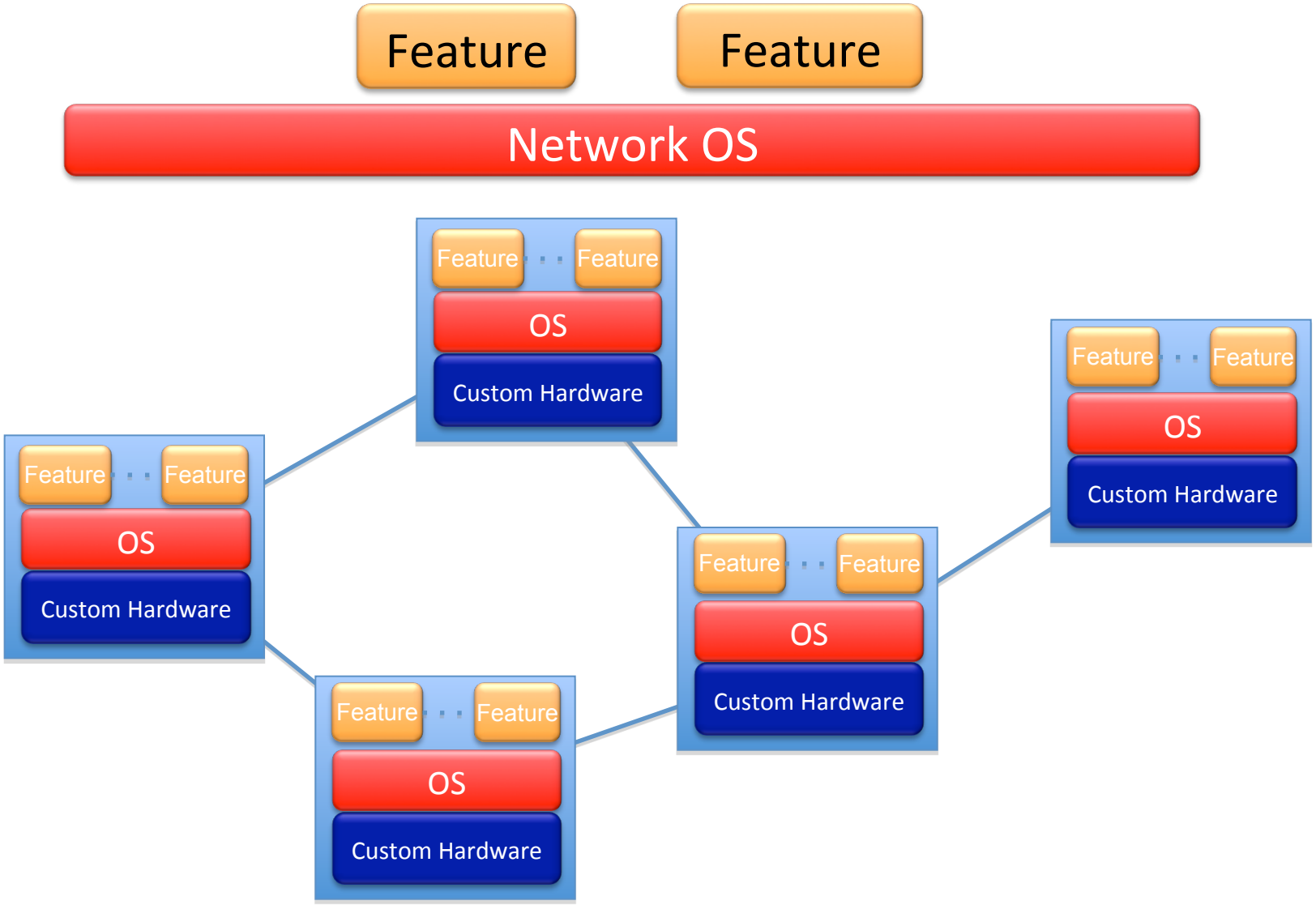


Oops – new link lets packets violate ***access control policy***!

Routing changed, but

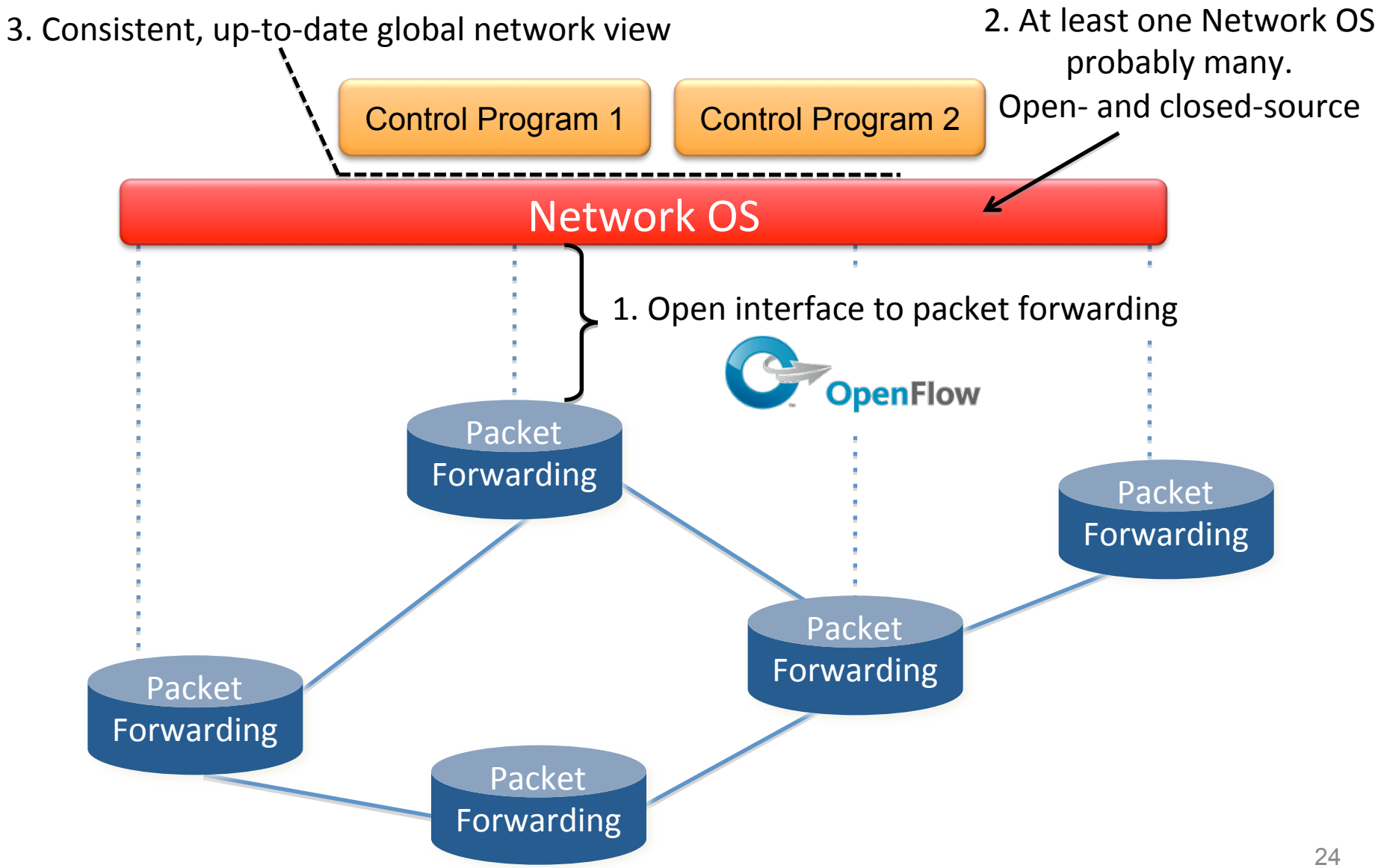
Packet filters don't update automatically

How SDN Changes the Network



23

Software Defined Network (SDN)



24

Network OS

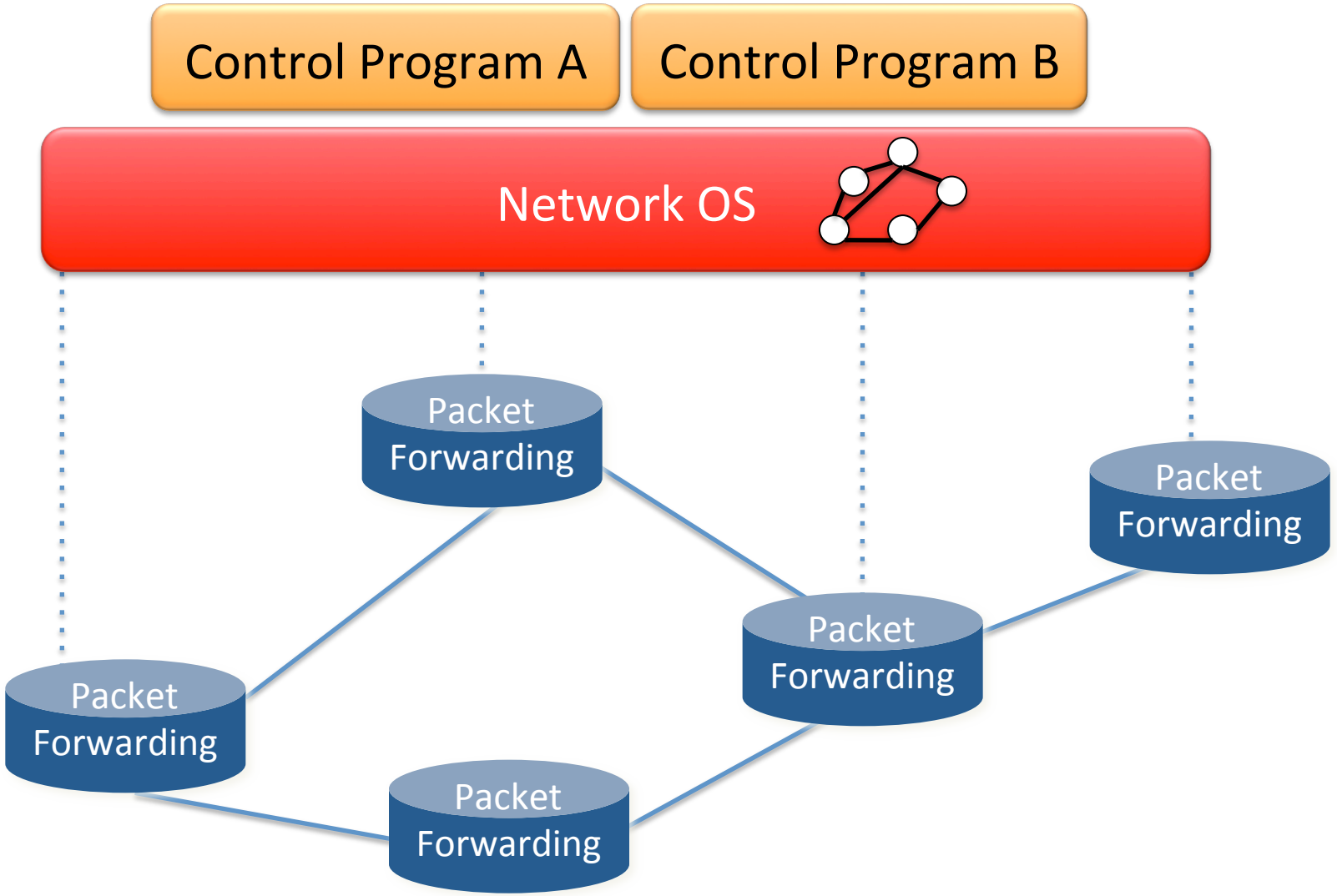
Network OS: distributed system that creates a consistent, up-to-date network view

- Runs on servers (controllers) in the network
- NOX, ONIX, Floodlight, Trema, OpenDaylight, HyperFlow, Kandoo, Beehive, Beacon, Maestro, … + more

Uses **forwarding abstraction** to:

- Get state information **from** forwarding elements
- Give control directives **to** forwarding elements

Software Defined Network (SDN)



Control Program

Control program operates on view of network

- **Input**: global network view (graph/database)
- **Output**: configuration of each network device

Control program is not a distributed system

- Abstraction hides details of distributed state

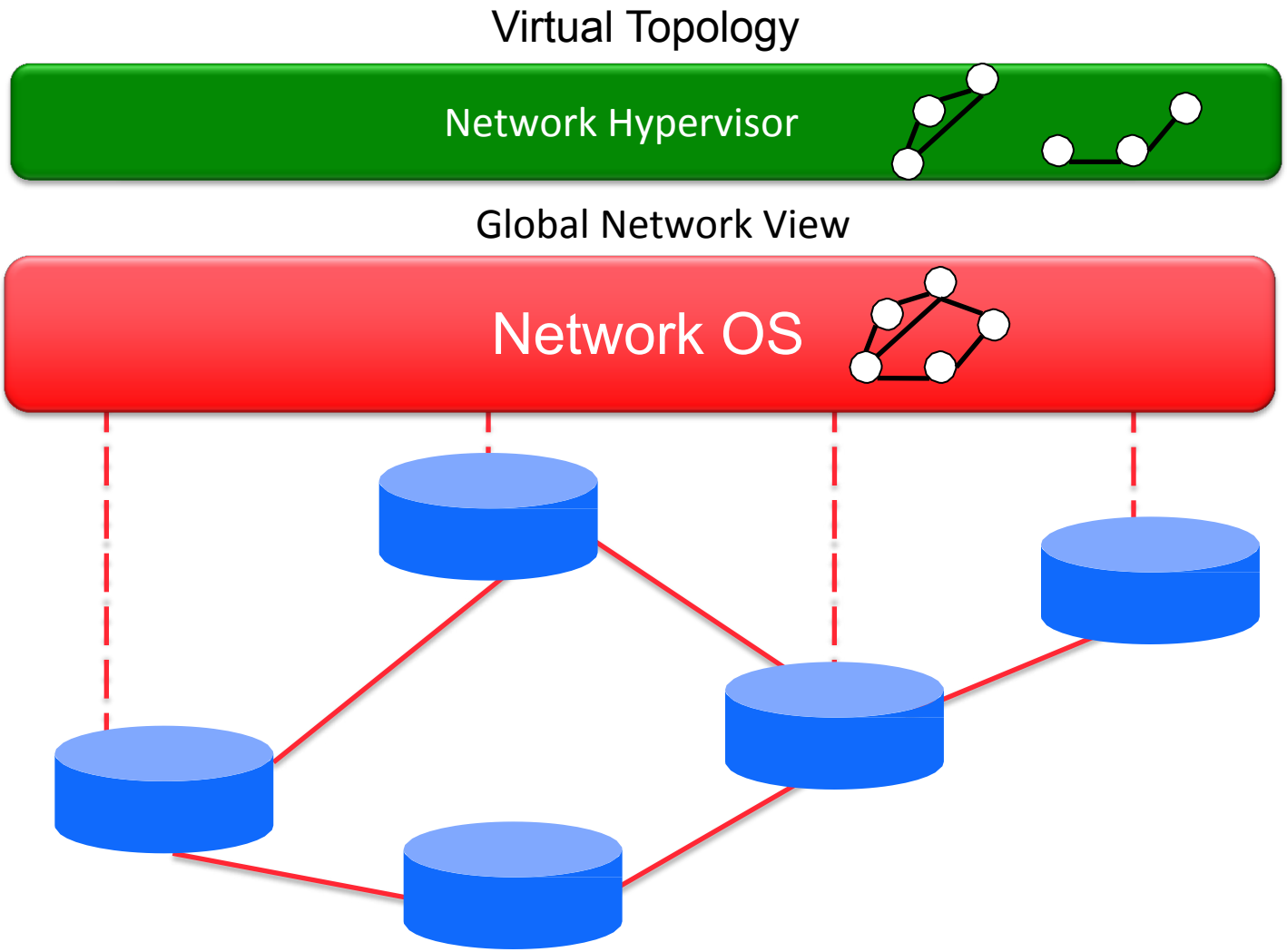
Forwarding Abstraction

Purpose: Standard way of defining forwarding state

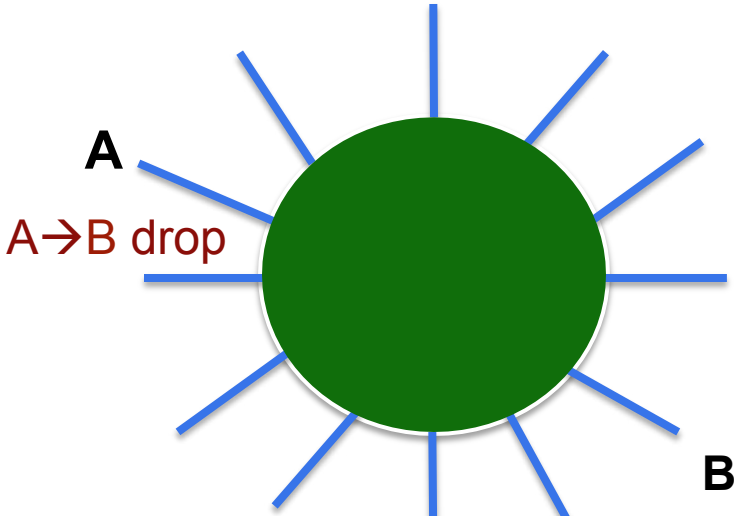
- Flexible
  - Behavior specified by control plane
  - Built from basic set of forwarding primitives
- Minimal
  - Streamlined for speed and low-power
  - Control program not vendor-specific

OpenFlow is an example of such an abstraction

Software Defined Network

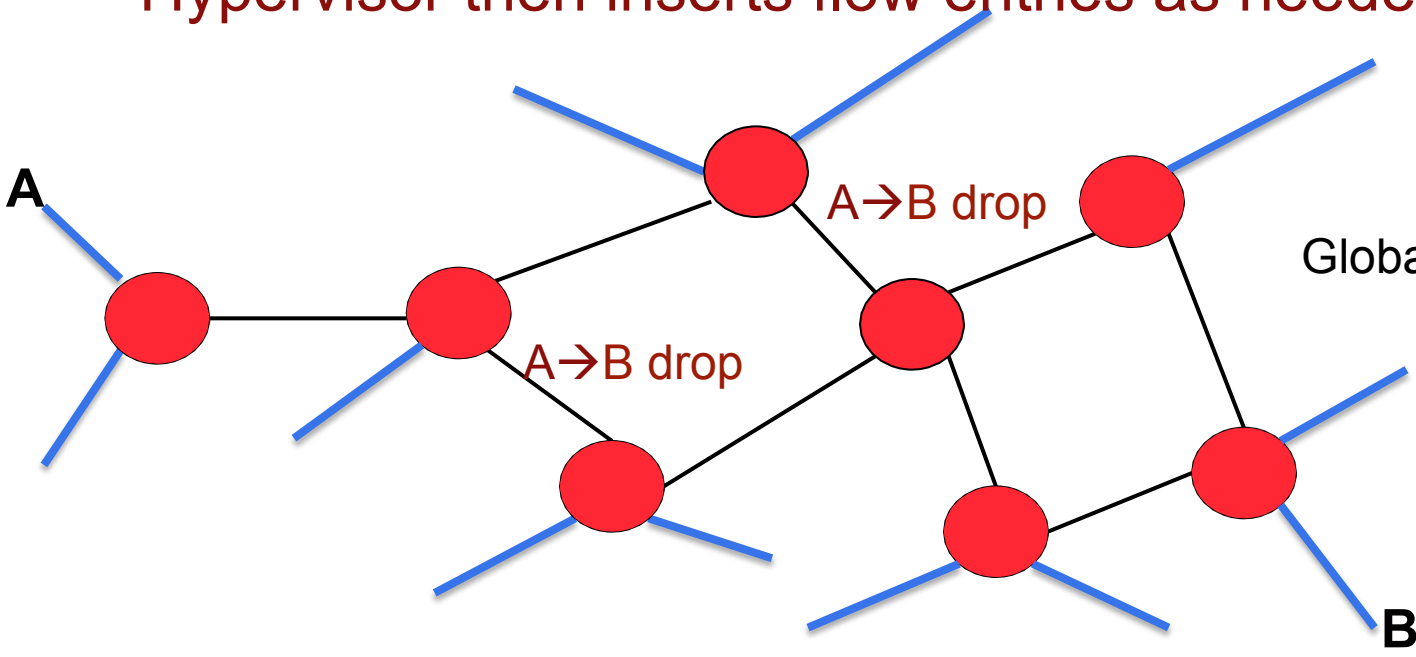


Virtualization Simplifies Control Program



Abstract Network View

Hypervisor then inserts flow entries as needed



Global Network View

Does SDN Simplify the Network?

What You Said

“However, I remain skeptical that such an approach will actually simplify much in the long run. That is, the basic paradigm in networks (layers) is in fact a simple model. However, the ever-changing performance and functionality goals have forced more complexity into network design. I'm not sure if SDN will be able to maintain its simplified model as goals continue to evolve.”

Does SDN Simplify the Network?

Abstraction doesn't eliminate complexity

- NOS, Hypervisor are still complicated pieces of code

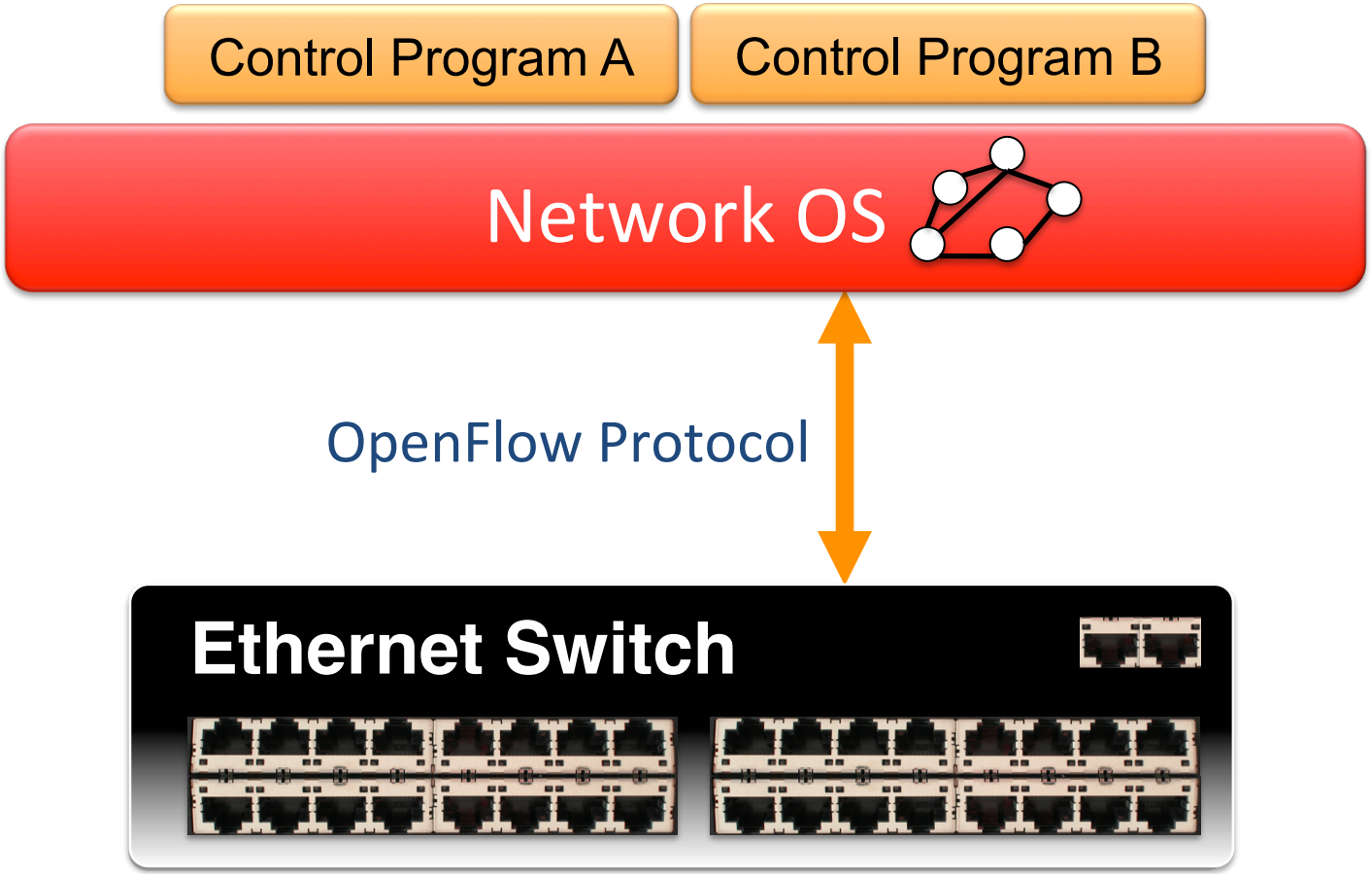
SDN main achievements

- Simplifies interface for control program (user-specific)
- Pushes complexity into reusable code (SDN platform)

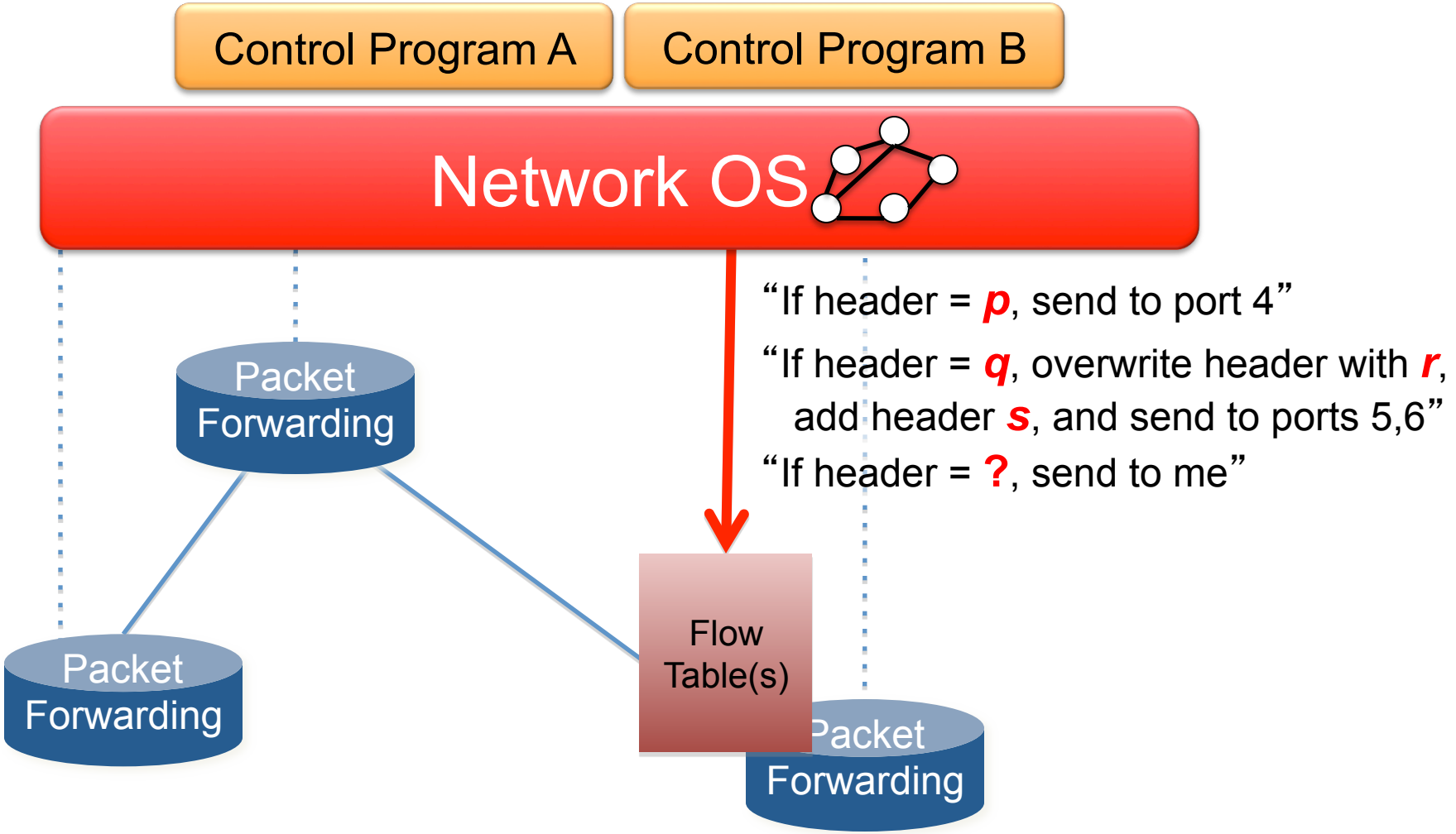
Just like compilers….

OpenFlow Basics

OpenFlow Basics

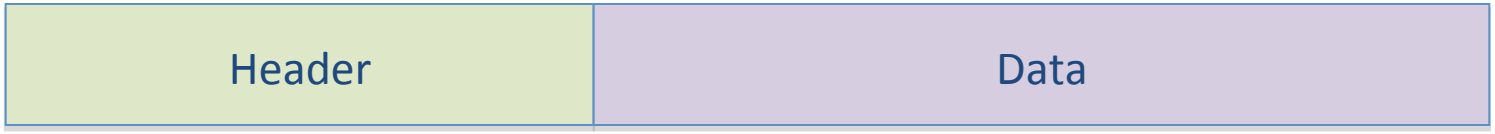


OpenFlow Basics



Primitives <Match, Action>

Match arbitrary bits in headers:



Match: 1000x01xx0101001x

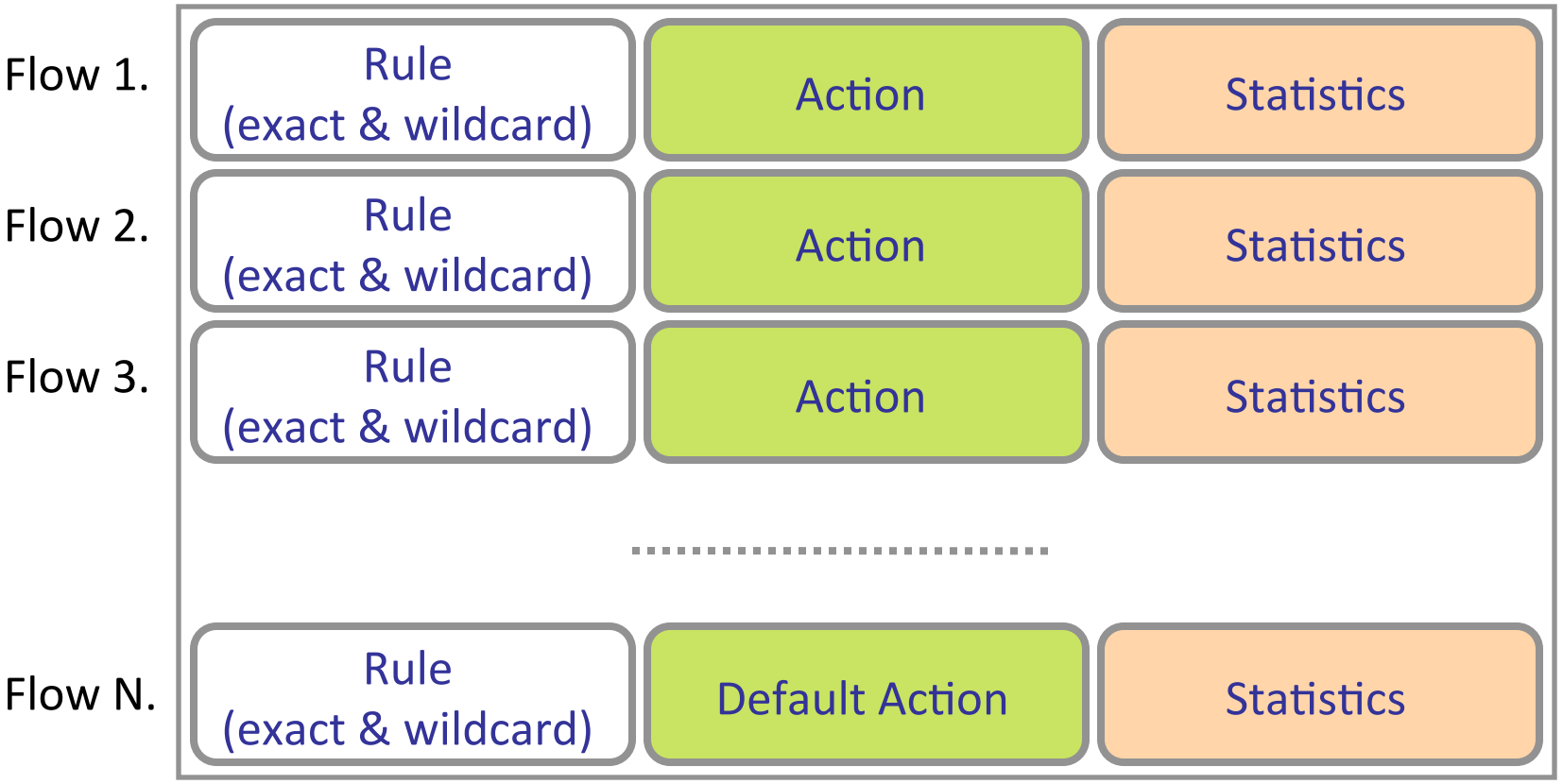
- Match on any header, or new header
- Allows any flow granularity

Action

- Forward to port(s), drop, send to controller
- Overwrite header with mask, push or pop
- Forward at specific bit-rate

OpenFlow Rules

Exploit the flow table in switches, routers, and chipsets



Why is SDN happening now?

The Road to SDN

Active Networking: 1990s

- First attempt make networks programmable
- Demultiplexing packets to software programs, network virtualization, …

Control/Dataplane Separation: 2003-2007

- ForCes [IETF],
  RCP, 4D [Princeton, CMU],
  SANE/Ethane [Stanford/Berkeley]
- Open interfaces between data and control plane, logically centralized control

OpenFlow API & Network Oses: 2008

- OpenFlow switch interface [Stanford]
- NOX Network OS [Nicira]

N. Feamster et al., "The Road to SDN: An Intellectual History of Programmable Networks", ACM SIGCOMM CCR 2014.

SDN Drivers

Rise of merchant switching silicon

- Democratized switching
- Vendors eager to unseat incumbents

Cloud / Data centers

- Operators face real network management problems
- Extremely cost conscious; desire a lot of control

The right balance between vision & pragmatism

- OpenFlow compatible with existing hardware

A “killer app”: Network virtualization

Virtualization is Killer App for SDN

Consider a multi-tenant datacenter

- Want to allow each tenant to specify virtual topology
- This defines their individual policies and requirements

Datacenter's network hypervisor compiles these virtual topologies into set of switch configurations

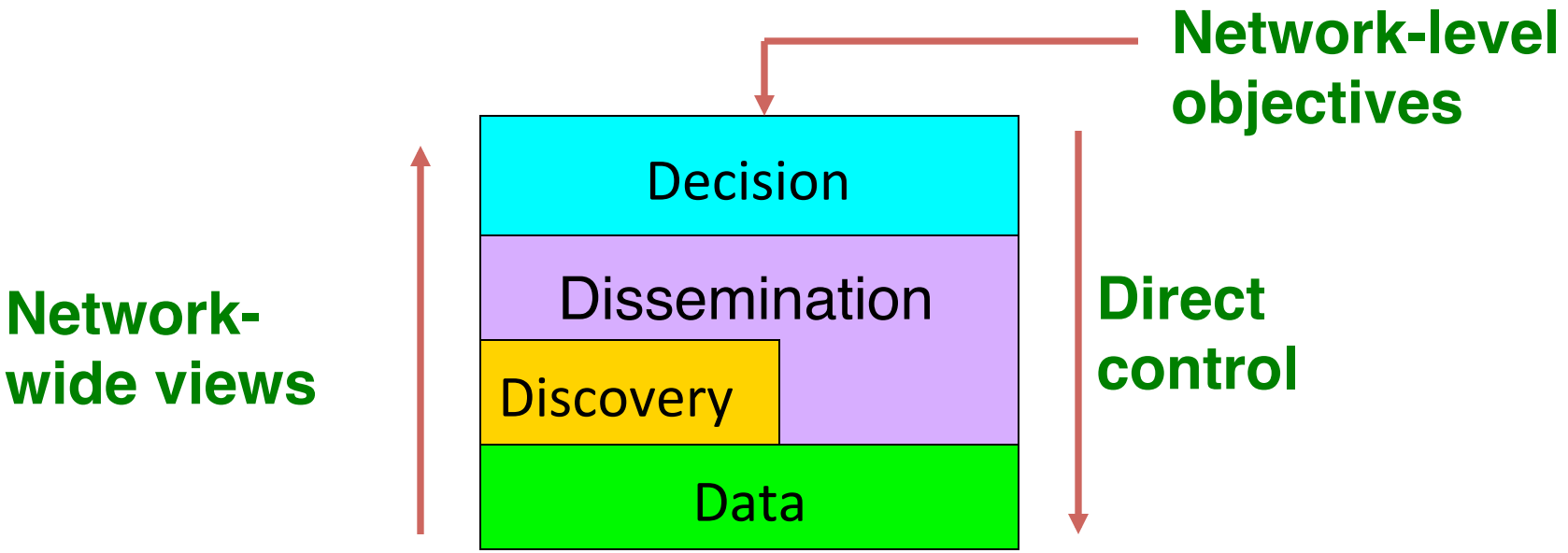
- Takes 1000s of individual tenant virtual topologies
- Computes configurations to implement all simultaneously

This is what people are paying money for….

- ***Enabled by SDN's ability to virtualize the network***

4D

4D



Decision: all management and control logic

Dissemination: communicating with routers

Discovery: topology and traffic monitoring

Data: packet handling

routers

What You Said

“The paper reads more like a thought-exercise or meta discussion of the future SDN field than a presentation of research. I am surprised sigcomm published it.”

“some good things about the way the paper was structured was that it mentioned that it had a lot of future work to do and didn't think it was a final solution. By at least addressing that it needs to continue to expand, the authors acknowledge they don't know the merits behind their solution…”

What You Said

"The most compelling aspect of SDN and of the 4D Approach proposed, in my opinion, is the ability to enable innovation. However, SDN taken to the extreme proposed in the 4D approach seems to me to significantly limit scalability and increase complexity."

What You Said

“My concern is that, previous designs that is aware of the delay of updating network view, take the consideration right on their control (they have control rules and protocol that touch this directly). But SDN tries to hide this nature from the programmers. I am not sure if the design of the software, in the absence of these concerns, will end up with expected results.”

Practical Challenges

Scalability
- Decision elements responsible for many routers

Reliability
- Surviving failures of decision elements and routers

Response time
- Delays between decision elements and routers

Consistency
- Ensuring multiple decision elements behave consistently

Security
- Network vulnerable to attacks on decision elements

Interoperability
- Legacy routers and neighboring domains

Next Time…

Onix: A Distributed Control Platform for Large-scale Production Networks

Teemu Koponen*, Martin Casado*, Natasha Gude*, Jeremy Stribling*, Leon Poutievski†, Min Zhu†, Rajiv Ramanathan†, Yuichiro Iwata‡, Hiroaki Inoue‡, Takayuki Hama‡, Scott Shenker§

Abstract

Computer networks lack a general control paradigm, as traditional networks do not provide any network-wide management abstractions. As a result, each new function (such as routing) must provide its own state distribution, element discovery, and failure recovery mechanisms. We believe this lack of a common control platform has significantly hindered the development of flexible, reliable and feature-rich network control planes.

To address this, we present Onix, a platform on top of which a network control plane can be implemented as a distributed system. Control planes written within Onix operate on a global view of the network, and use basic state distribution primitives provided by the platform. Thus Onix provides a general API for control plane implementations, while allowing them to make their own trade-offs among consistency, durability, and scalability.

and distributing the appropriate control state to them, as well as coordinating the state among the various platform servers – and provides a programmatic interface upon which developers can build a wide variety of management applications. (The term "management application" refers to the control logic needed to implement management features such as routing and access control.)[3] For the purposes of this paper, we refer to this paradigm for network control as *Software-Defined Networking* (SDN).

This is in contrast to the traditional network control model in which state distribution is limited to link and reachability information and the distribution model is fixed. Today a new network control function (*e.g.*, scalable routing of flat intra-domain addresses [21]) requires its own distributed protocol, which involves first solving a hard, low-level design problem and then later overcoming the difficulty of deploying this design on switches. As a result, networking gear today supports