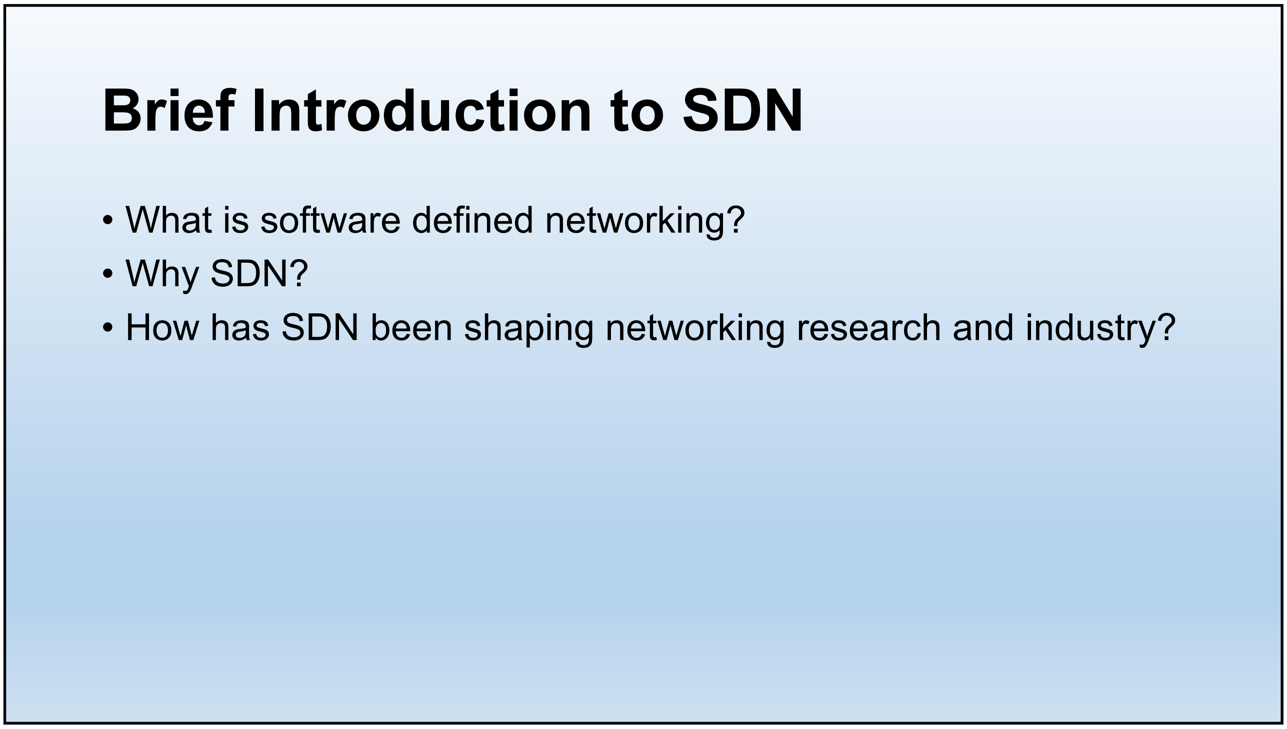
Lecture Goals and Structure

- The lecture will equips you to address the following questions:
  - What is software defined networking?
  - What are the key building blocks?
  - How do we use SDN to solve enterprise, carrier, and data center/cloud networking problems?

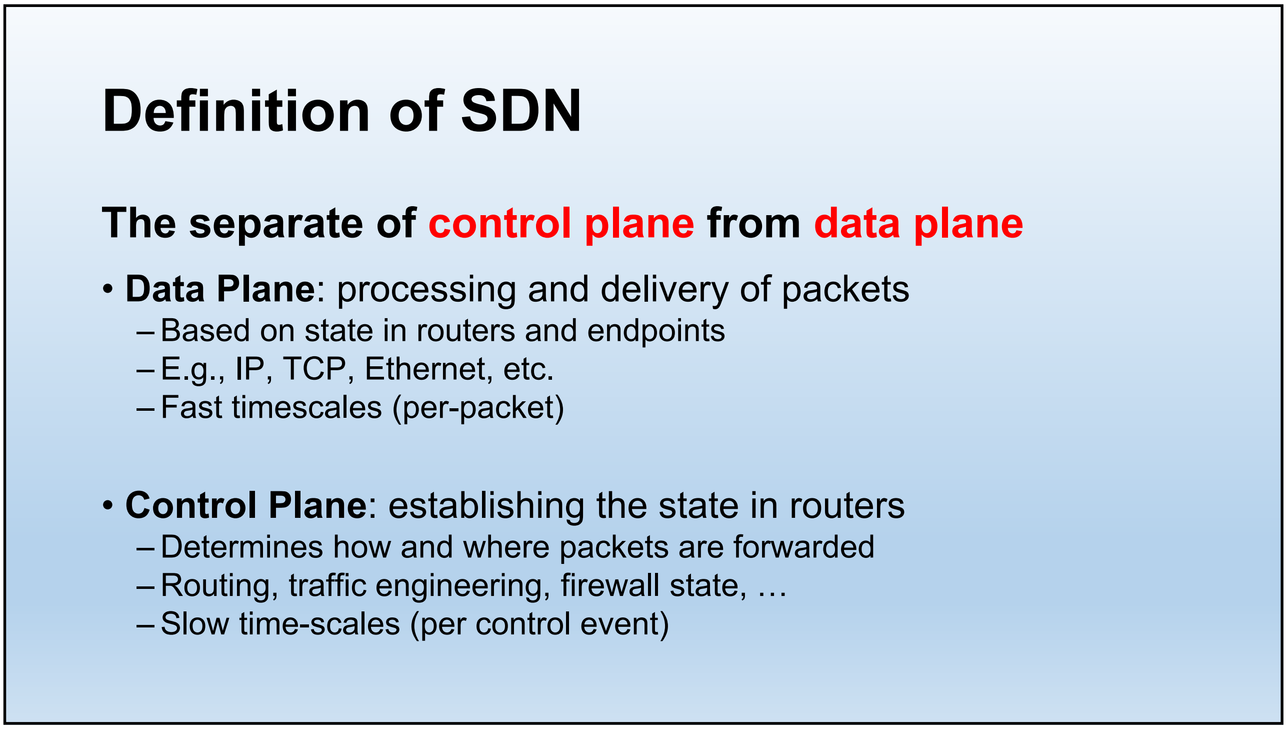
List of Suggested Research Areas

- SDN to improve video applications
- SDN measurement primitives
- SDN testing and debugging
- SDN reliability
- SDN security: mitigate DDoS attacks
- SDN controller design, (e.g. cellular, data centers)
- Cellular network virtualization
- Programming language abstraction for wireless networks/optical networks
- Fast control plane and data plane interaction



Brief Introduction to SDN

- What is software defined networking?
- Why SDN?
- How has SDN been shaping networking research and industry?



Definition of SDN

The separate of control plane from data plane

- **Data Plane**: processing and delivery of packets
  - Based on state in routers and endpoints
  - E.g., IP, TCP, Ethernet, etc.
  - Fast timescales (per-packet)

- **Control Plane**: establishing the state in routers
  - Determines how and where packets are forwarded
  - Routing, traffic engineering, firewall state, …
  - Slow time-scales (per control event)

Software-Defined Networking

- SDN clearly has advantages over status quo
- But is SDN the “right” Solution?
- First we will find out: Not what SDN is, but why SDN is

Key to Internet Success: Layers

Applications

…built on…

Reliable (or unreliable) transport

…built on…

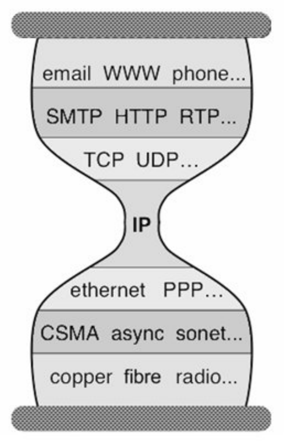
Best-effort global packet delivery

…built on…

Best-effort local packet delivery

…built on…

Physical transfer of bits

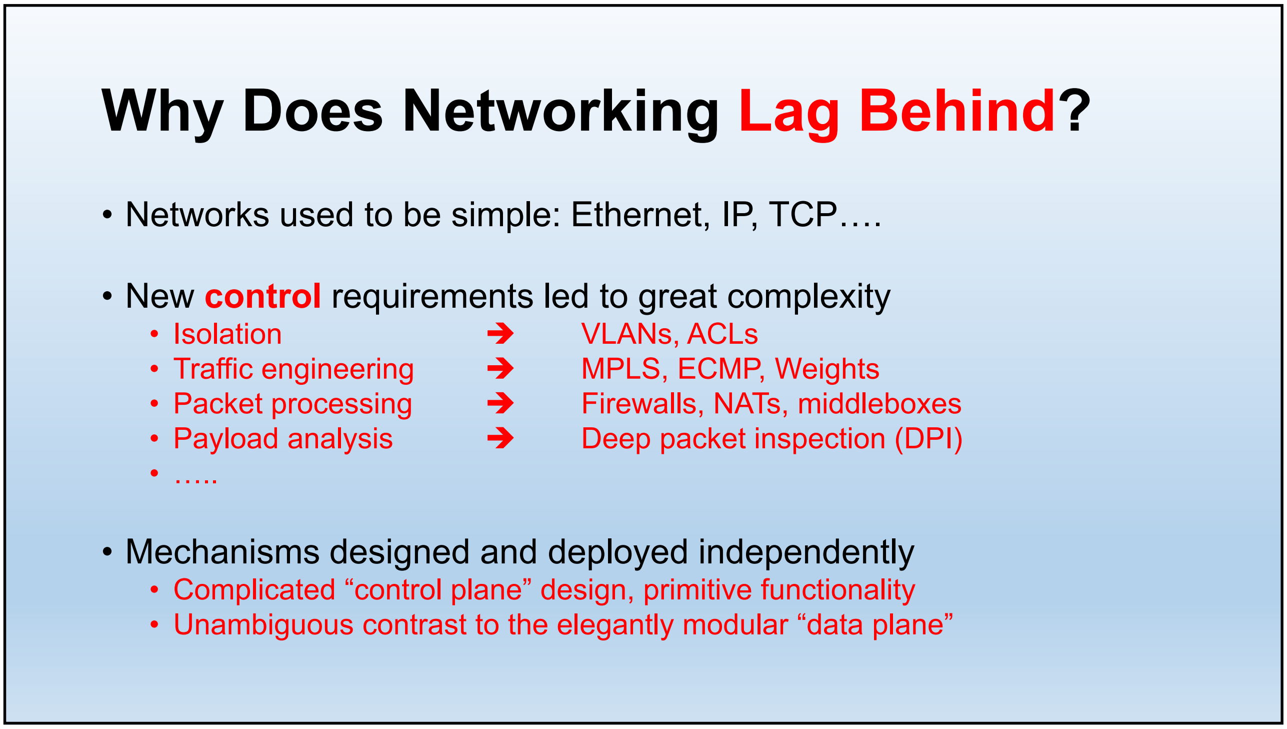


Why Is Layering So Important?

- Decomposed delivery into fundamental components
- Independent but compatible **innovation** at each layer
- A practical success of unprecedented proportions…
- **…but an academic failure**

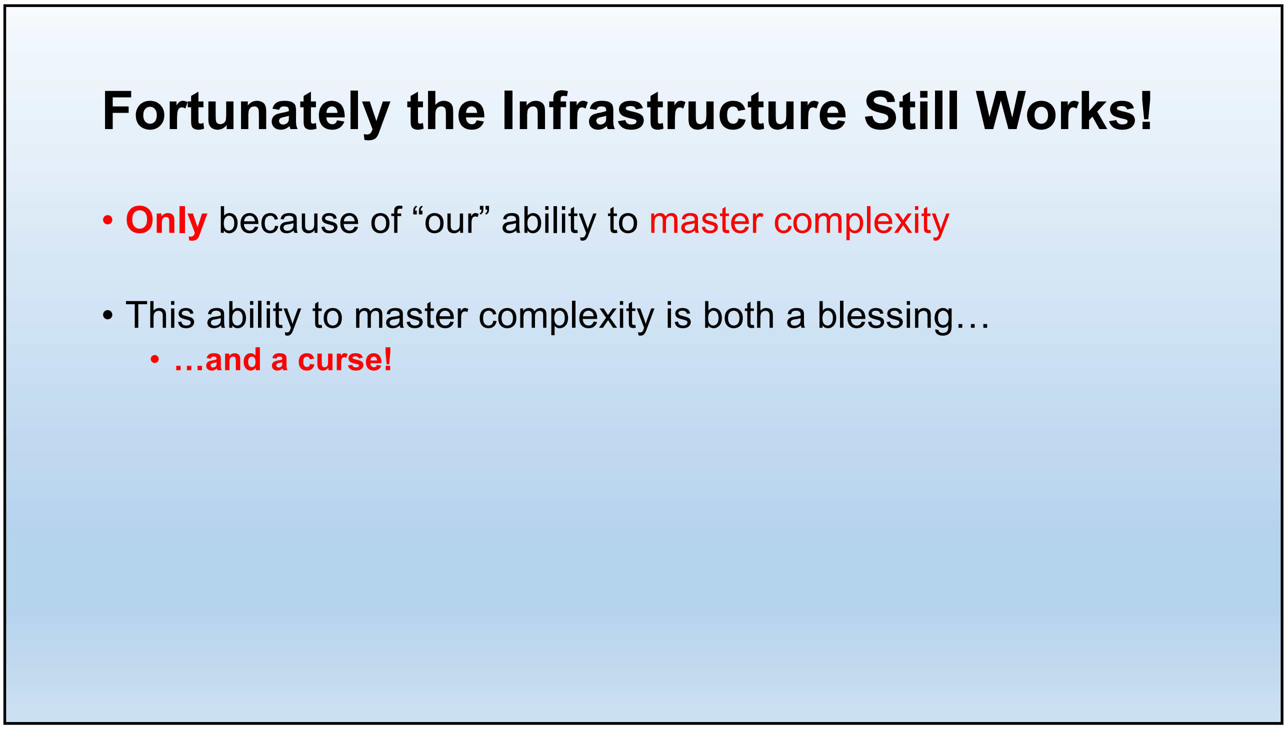
Built an Artifact, Not a Discipline

- Other fields in “systems”: OS, DB, DS, etc.
  - Teach basic principles
  - Are easily managed
  - Continue to evolve
- Networking:
  - Teach big bag of protocols
  - Notoriously difficult to manage
  - Evolves very slowly



Why Does Networking Lag Behind?

- Networks used to be simple: Ethernet, IP, TCP….
- New **control** requirements led to great complexity
  - Isolation ➔ VLANs, ACLs
  - Traffic engineering ➔ MPLS, ECMP, Weights
  - Packet processing ➔ Firewalls, NATs, middleboxes
  - Payload analysis ➔ Deep packet inspection (DPI)
  - …..
- Mechanisms designed and deployed independently
  - Complicated “control plane” design, primitive functionality
  - Unambiguous contrast to the elegantly modular “data plane”



Fortunately the Infrastructure Still Works!

- **Only** because of “our” ability to master complexity
- This ability to master complexity is both a blessing…
  - **…and a curse!**

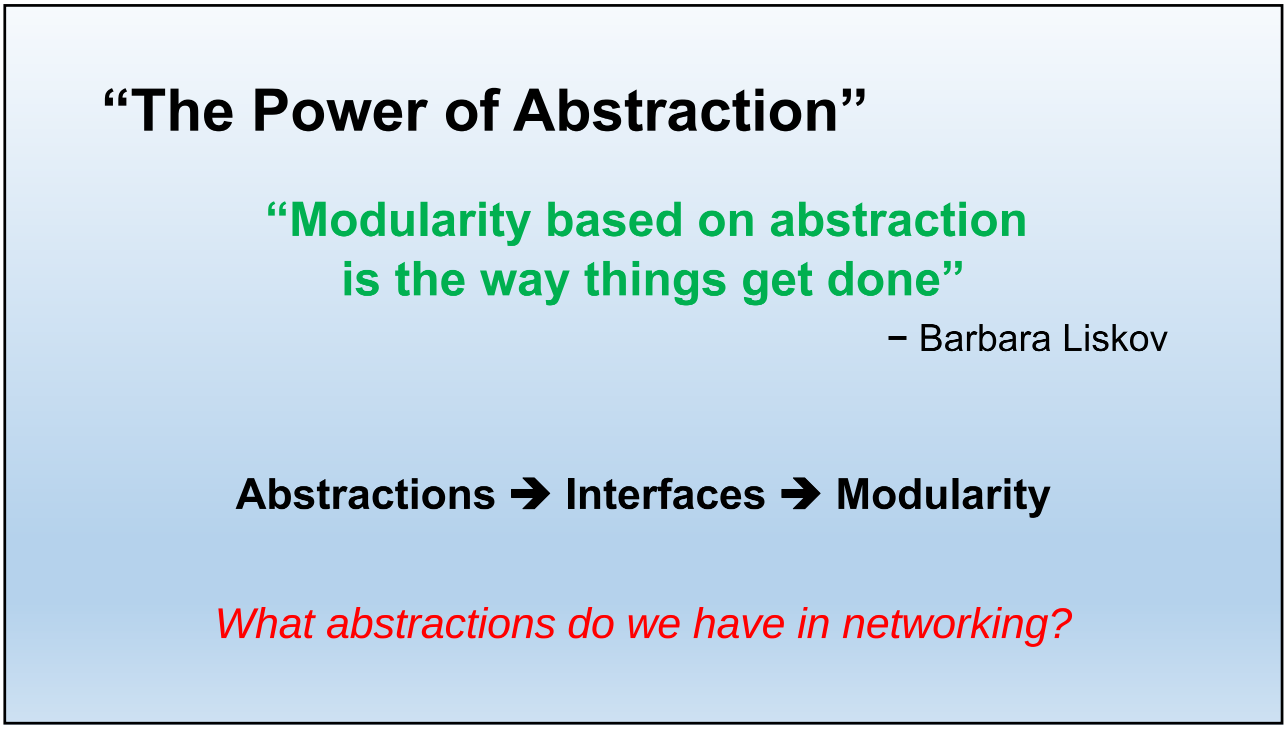
What Is the *problem*?

- Networking still focused on mastering complexity
  - Little emphasis on extracting simplicity from control plane
  - No recognition that there’s a difference….
- Extracting simplicity builds intellectual foundations
  - **Necessary for creating a discipline….**
  - **That’s why networking lags behind**

A Better Example: Programming

- Machine languages: no abstractions
  - Mastering complexity was crucial
- Higher-level languages: OS and other abstractions
  - File system, virtual memory, abstract data types, ...
- Modern languages: even more abstractions
  - Object orientation, garbage collection,…

Abstractions key to extracting simplicity



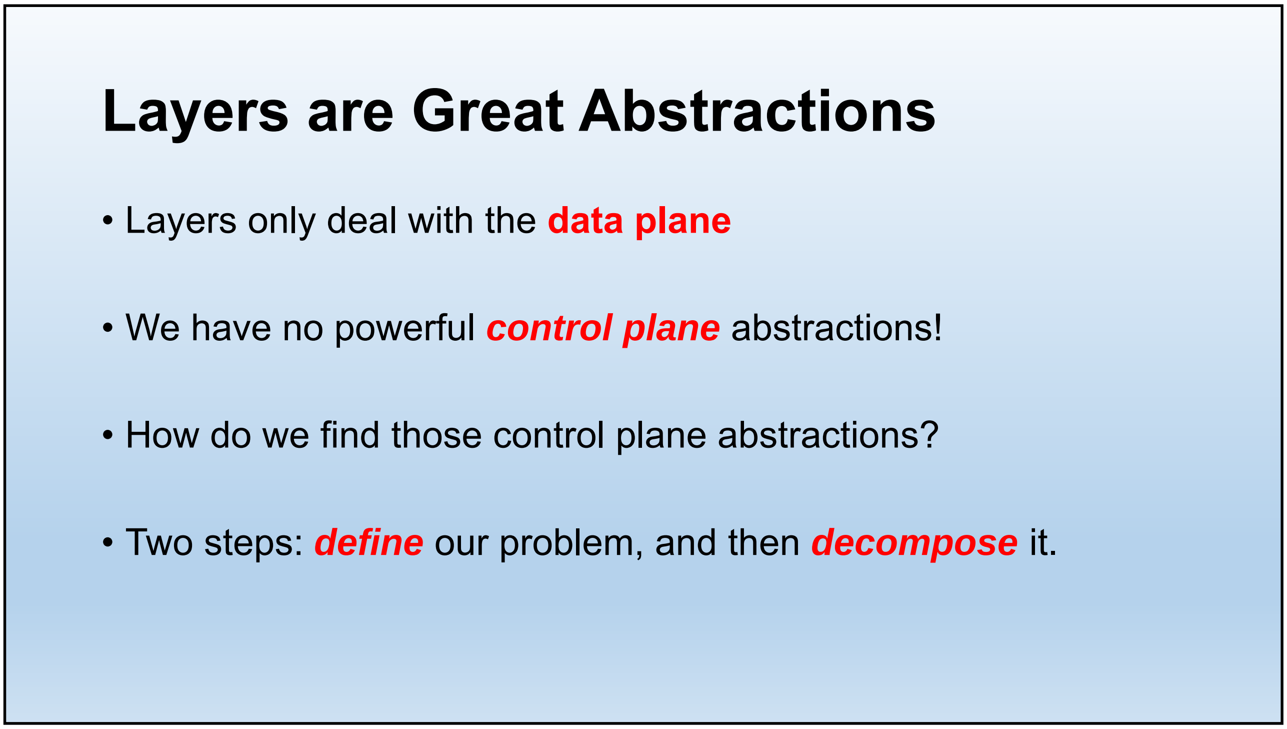
“The Power of Abstraction”

“Modularity based on abstraction is the way things get done”

− Barbara Liskov

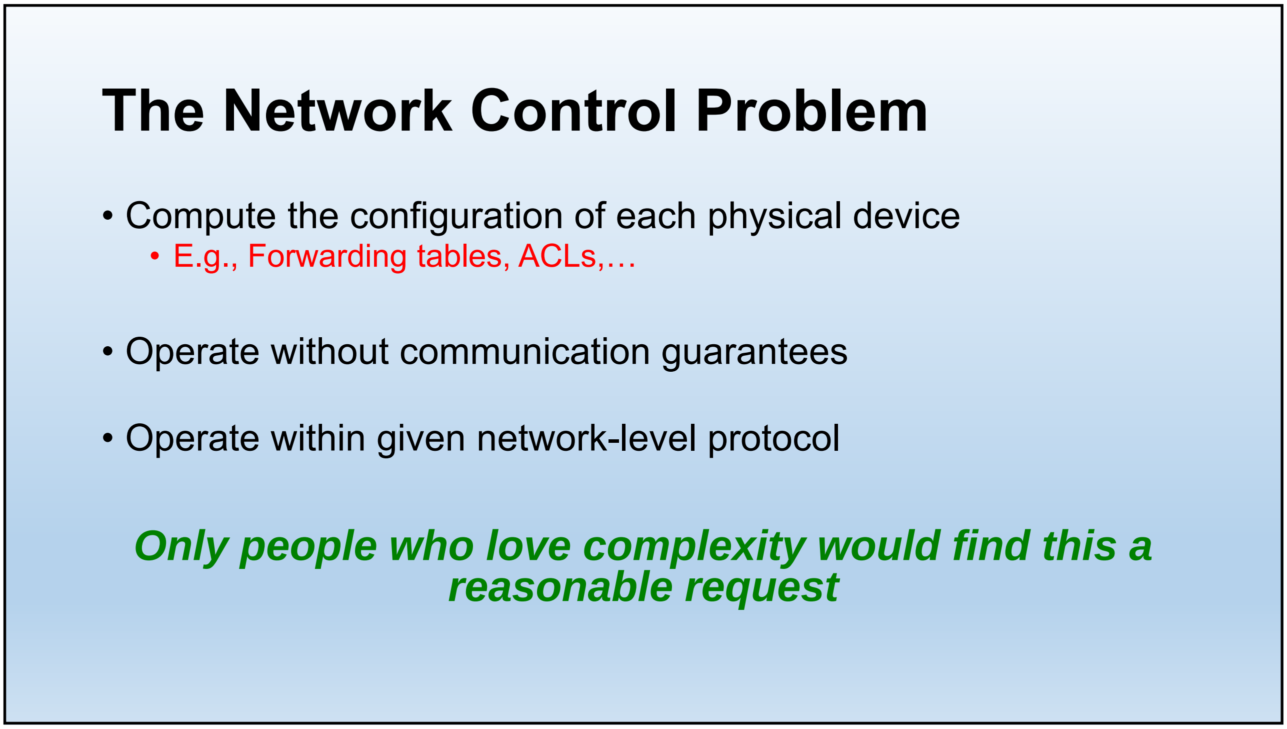
Abstractions → Interfaces → Modularity

What abstractions do we have in networking?



Layers are Great Abstractions

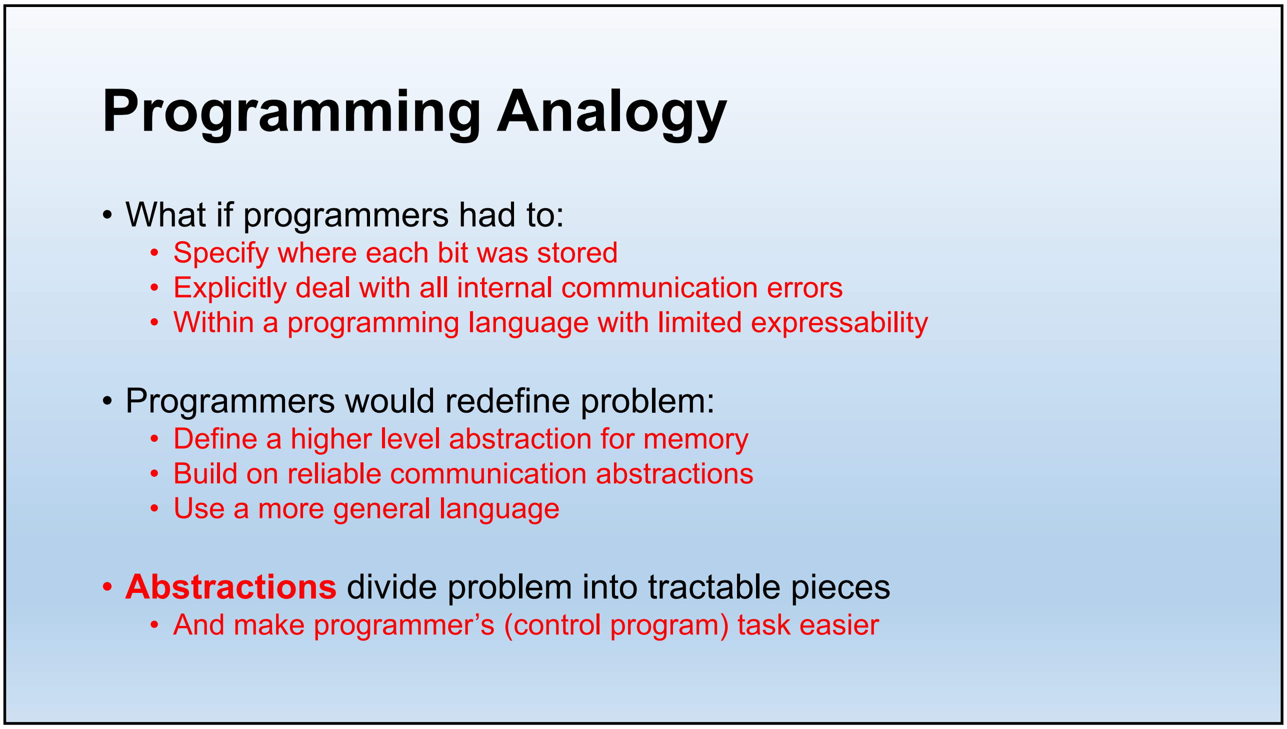
- Layers only deal with the **data plane**
- We have no powerful ***control plane*** abstractions!
- How do we find those control plane abstractions?
- Two steps: ***define*** our problem, and then ***decompose*** it.



The Network Control Problem

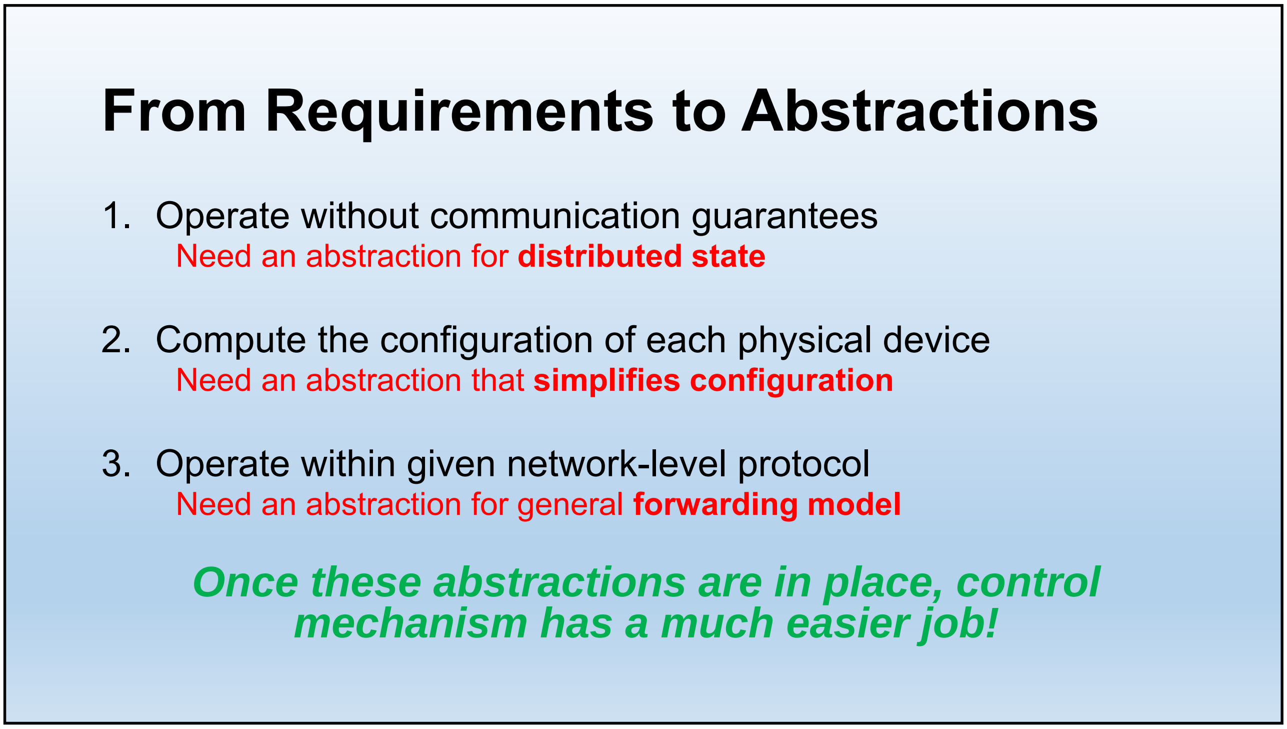
- Compute the configuration of each physical device
  - E.g., Forwarding tables, ACLs,…
- Operate without communication guarantees
- Operate within given network-level protocol

Only people who love complexity would find this a reasonable request



Programming Analogy

- What if programmers had to:
  - Specify where each bit was stored
  - Explicitly deal with all internal communication errors
  - Within a programming language with limited expressability
- Programmers would redefine problem:
  - Define a higher level abstraction for memory
  - Build on reliable communication abstractions
  - Use a more general language
- **Abstractions** divide problem into tractable pieces
  - And make programmer’s (control program) task easier



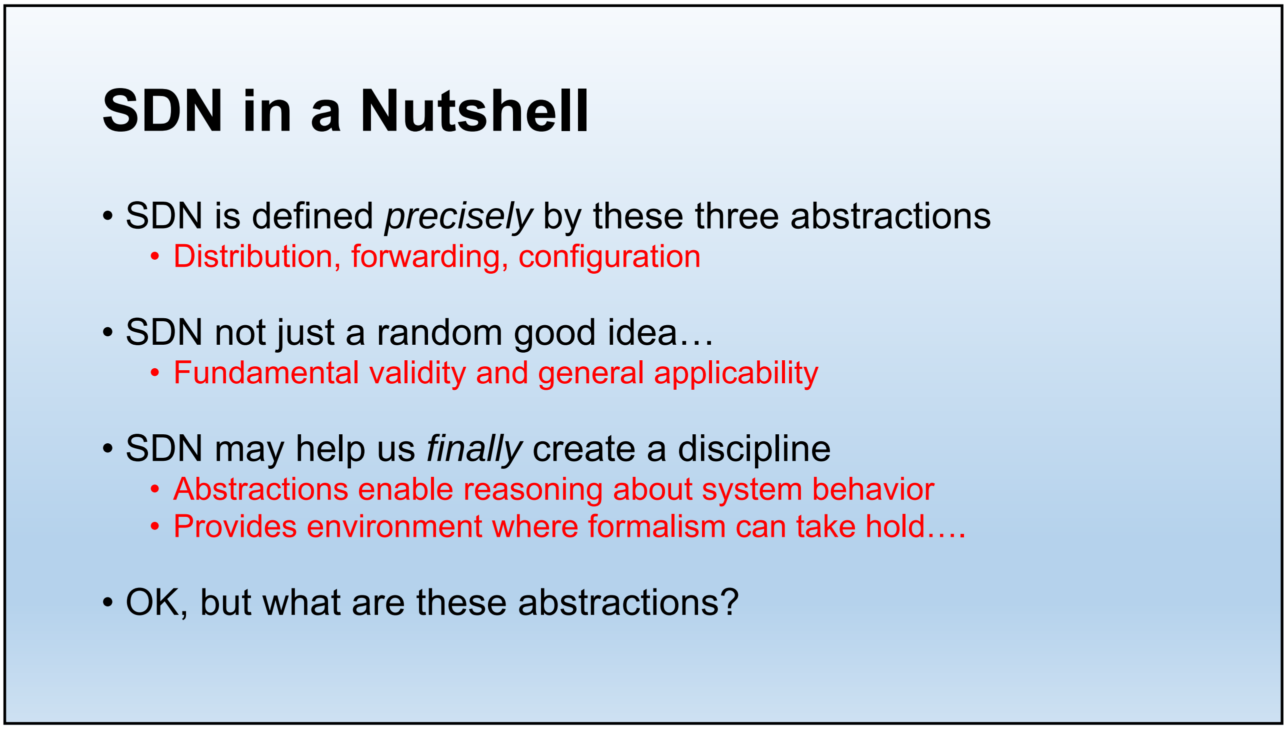
From Requirements to Abstractions

1. Operate without communication guarantees
   Need an abstraction for **distributed state**

2. Compute the configuration of each physical device
   Need an abstraction that **simplifies configuration**

3. Operate within given network-level protocol
   Need an abstraction for general **forwarding model**

Once these abstractions are in place, control mechanism has a much easier job!



SDN in a Nutshell

- SDN is defined *precisely* by these three abstractions
  - Distribution, forwarding, configuration
- SDN not just a random good idea…
  - Fundamental validity and general applicability
- SDN may help us *finally* create a discipline
  - Abstractions enable reasoning about system behavior
  - Provides environment where formalism can take hold….
- OK, but what are these abstractions?

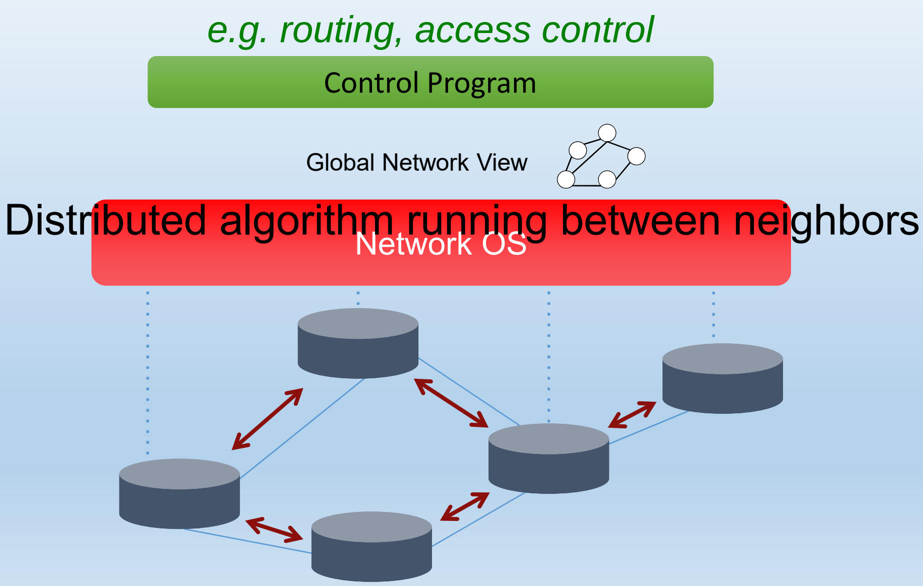
1. Distributed State Abstraction

- Shield control mechanisms from state distribution
  - While allowing access to this state
- Natural abstraction: ***global network view***
  - Annotated network graph provided through an API
- Implemented with “Network Operating System”
- Control mechanism is now program using API
  - No longer a distributed protocol, now just a graph algorithm
  - E.g. Use Dijkstra rather than Bellman-Ford

Software Defined Network (SDN)

Network of Switches and/or Routers

Traditional Control Mechanisms



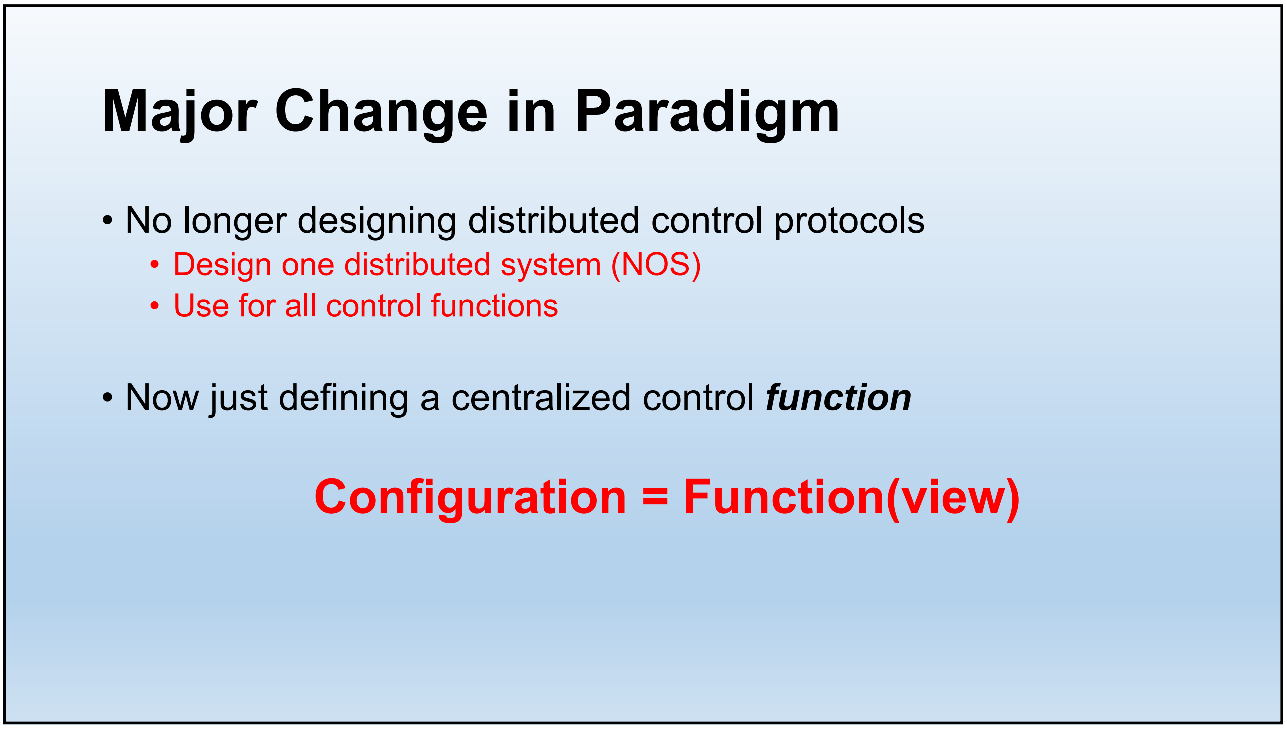
e.g. routing, access control

Control Program

Global Network View

Distributed algorithm running between neighbors

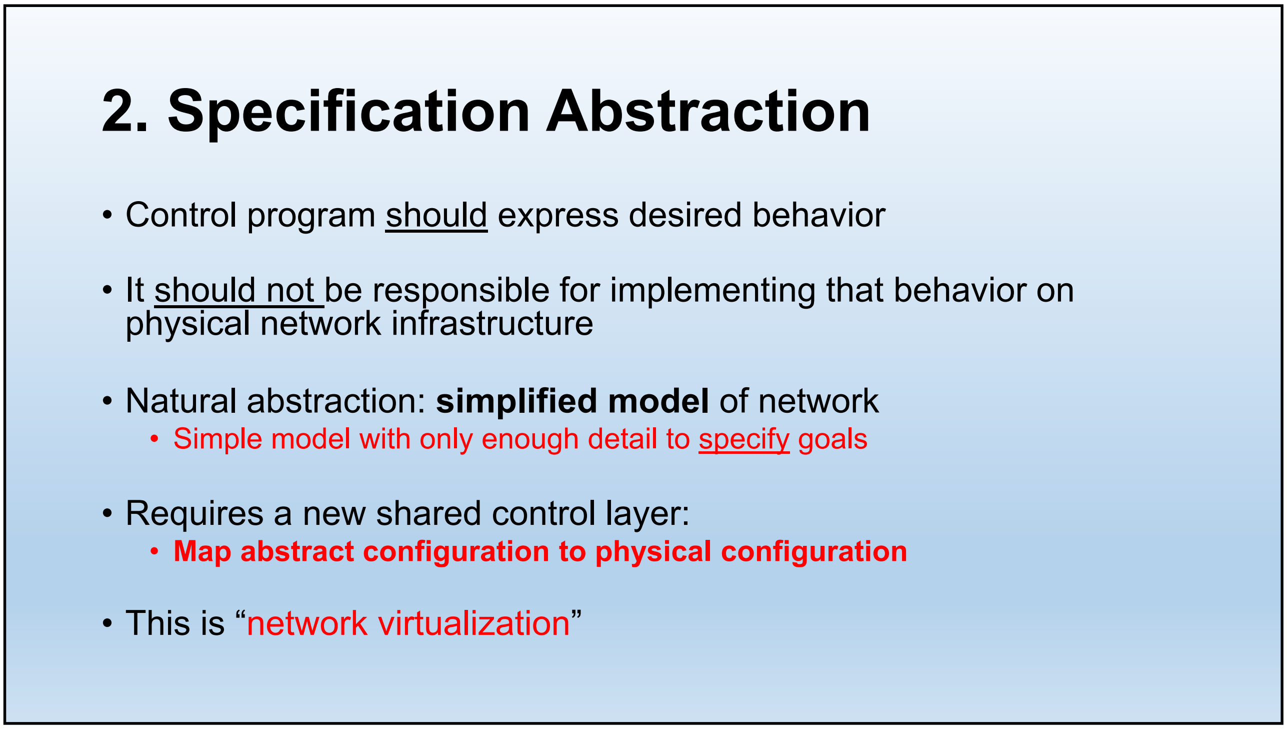
Network OS



Major Change in Paradigm

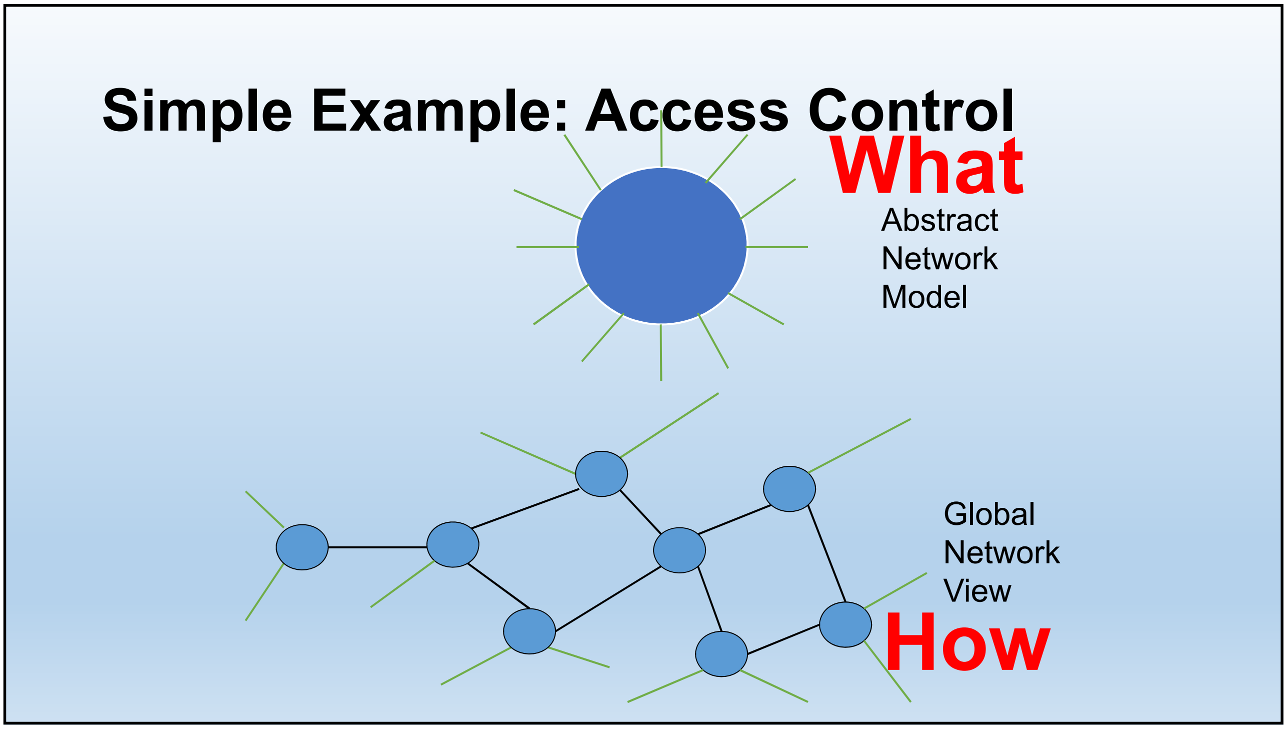
- No longer designing distributed control protocols
  - Design one distributed system (NOS)
  - Use for all control functions
- Now just defining a centralized control ***function***

Configuration = Function(view)



2. Specification Abstraction

- Control program should express desired behavior
- It should not be responsible for implementing that behavior on physical network infrastructure
- Natural abstraction: **simplified model** of network
  - Simple model with only enough detail to specify goals
- Requires a new shared control layer:
  - **Map abstract configuration to physical configuration**
- This is “network virtualization”



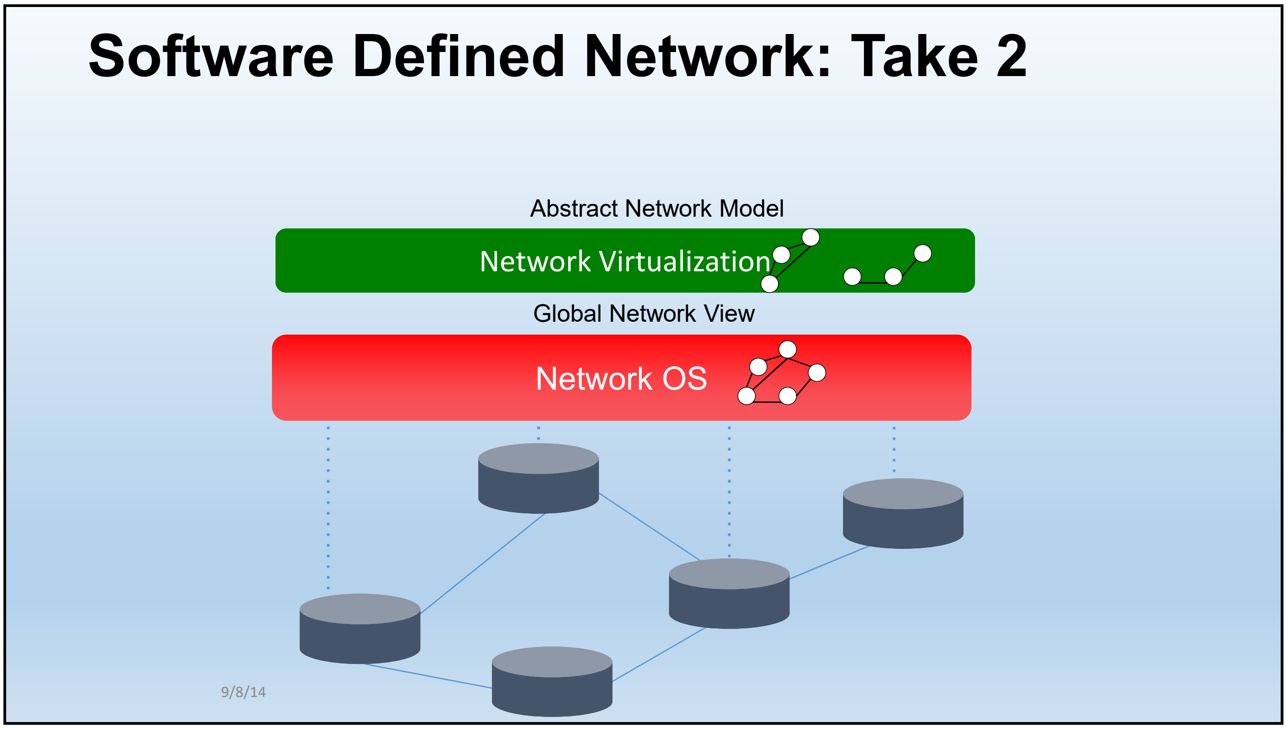
Simple Example: Access Control

What

Abstract
Network
Model

Global
Network
View

How



Software Defined Network: Take 2

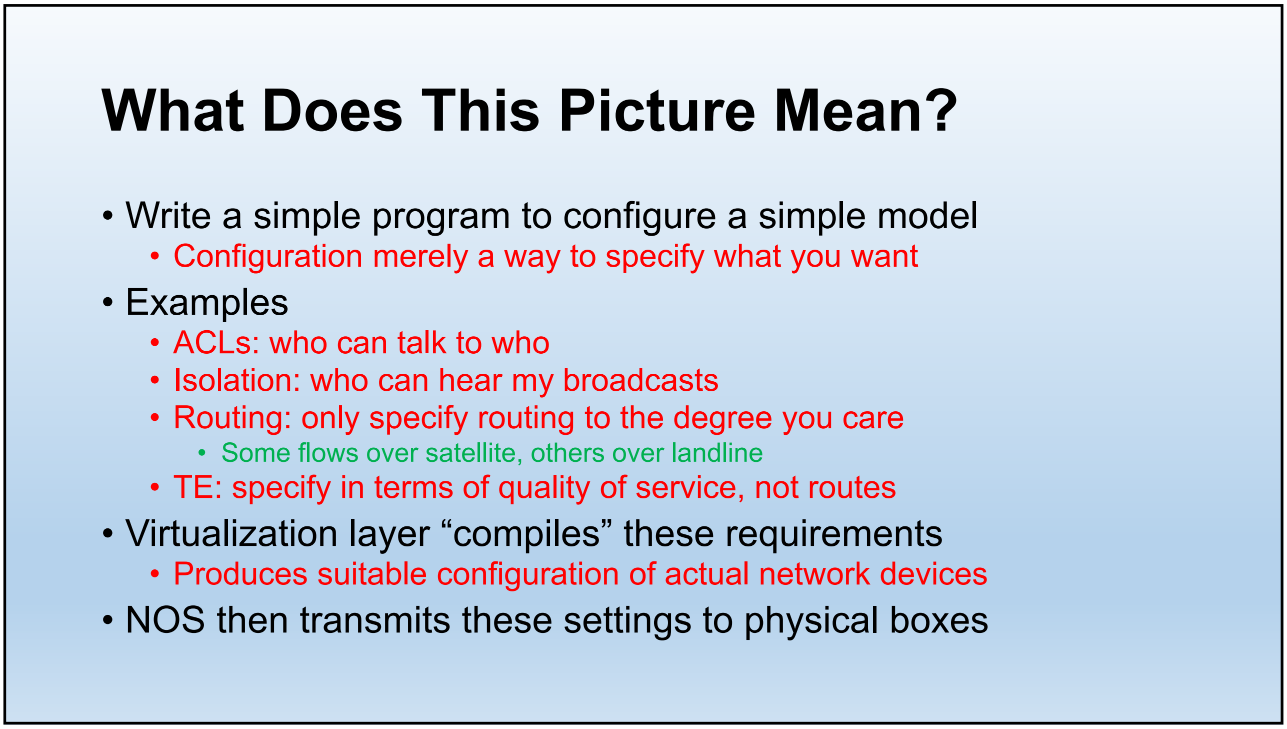
Abstract Network Model

Network Virtualization

Global Network View

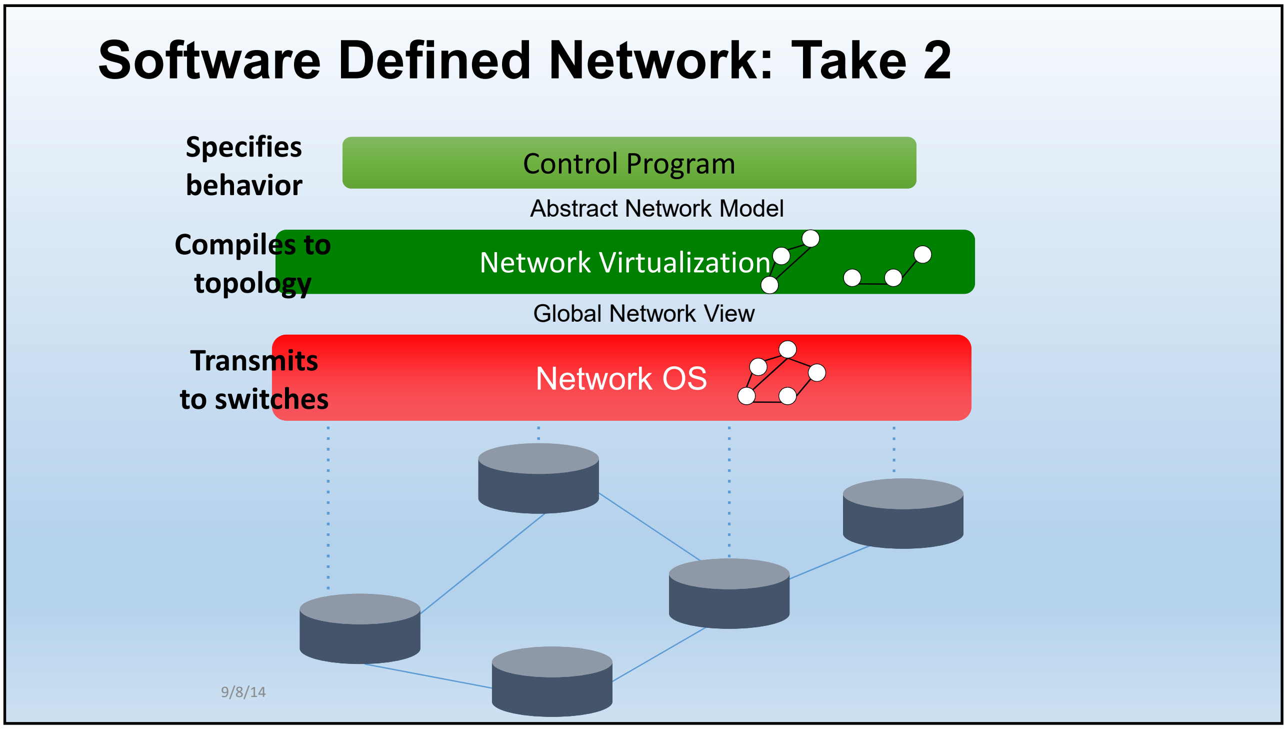
Network OS

9/8/14



What Does This Picture Mean?

- Write a simple program to configure a simple model
    - Configuration merely a way to specify what you want
- Examples
    - ACLs: who can talk to who
    - Isolation: who can hear my broadcasts
    - Routing: only specify routing to the degree you care
        - Some flows over satellite, others over landline
    - TE: specify in terms of quality of service, not routes
- Virtualization layer “compiles” these requirements
    - Produces suitable configuration of actual network devices
- NOS then transmits these settings to physical boxes



Software Defined Network: Take 2

Specifies behavior

Control Program

Abstract Network Model

Compiles to topology

Network Virtualization

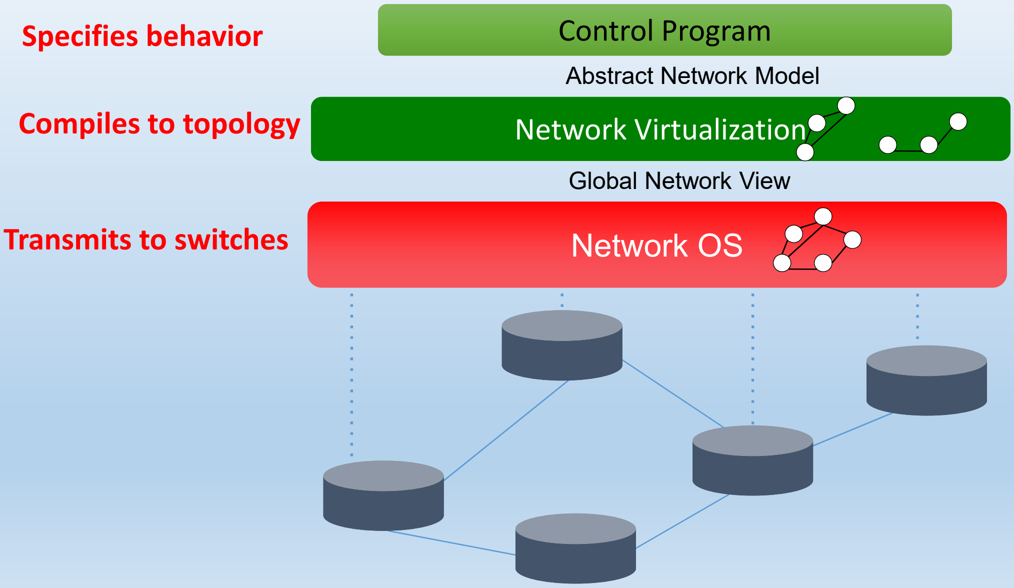
Global Network View

Transmits to switches

Network OS

9/8/14

Software Defined Network: Take 2



Specifies behavior

Control Program

Abstract Network Model

Compiles to topology

Network Virtualization

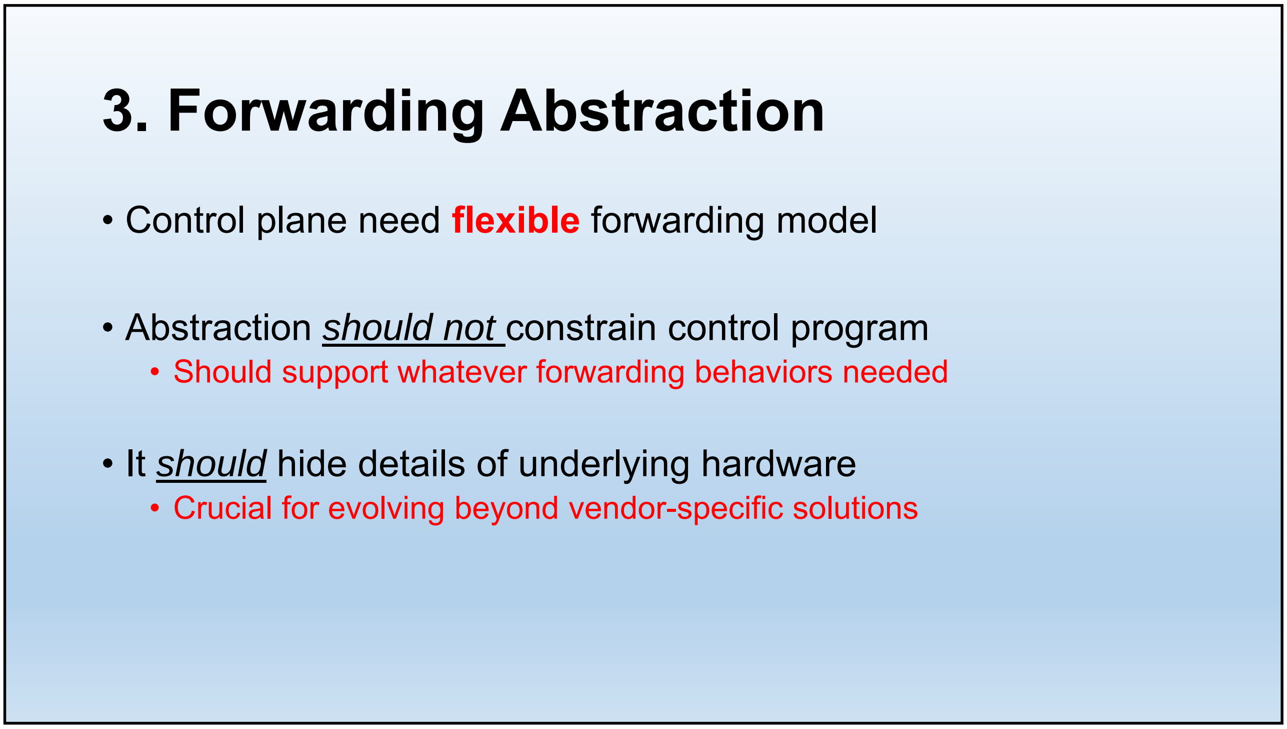
Global Network View

Transmits to switches

Network OS

Two Examples Uses

- Scale-out router:
  - Abstract view is single router
  - Physical network is collection of interconnected switches
  - Allows routers to “scale out, not up”
  - Use standard routing protocols on top
- Multi-tenant networks:
  - Each tenant has control over their “private” network
  - Network virtualization layer compiles all of these individual control requests into a single physical configuration
- **Hard to do without SDN, easy** *(in principle)* **with SDN**

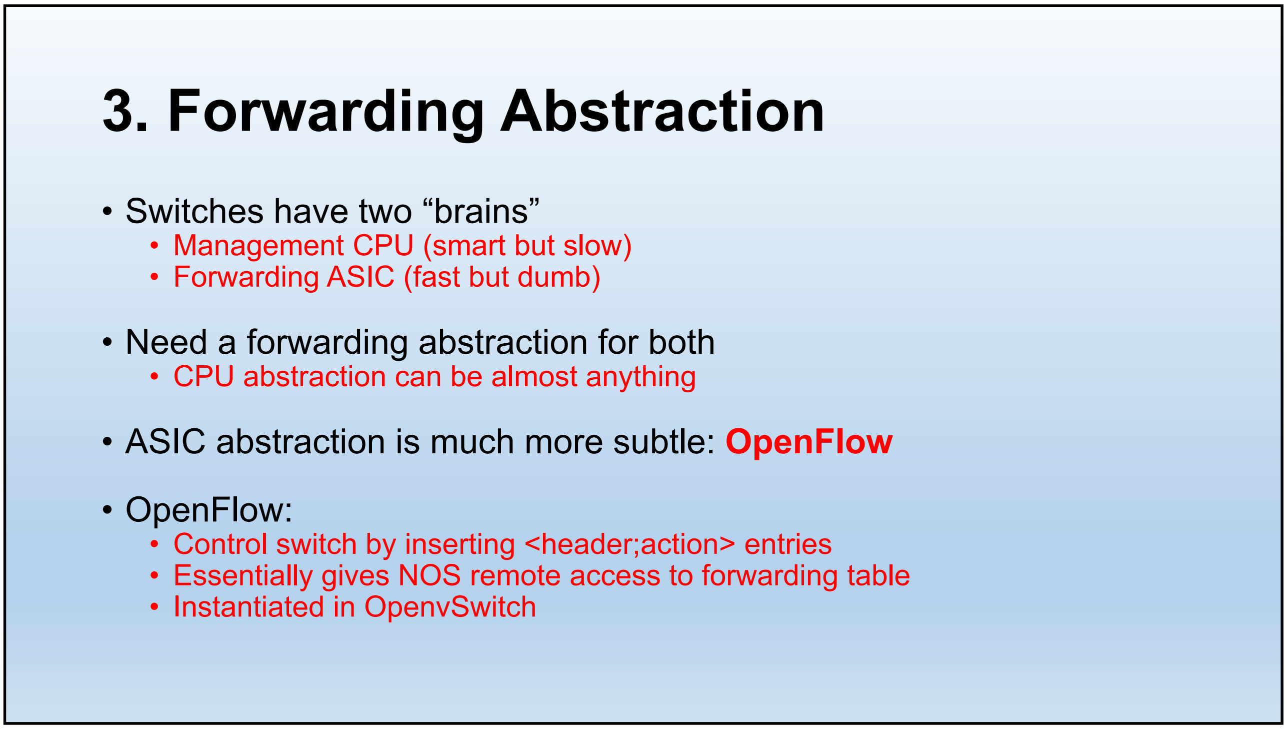


3. Forwarding Abstraction

- Control plane need **flexible** forwarding model

- Abstraction *should not* constrain control program
  - Should support whatever forwarding behaviors needed

- It *should* hide details of underlying hardware
  - Crucial for evolving beyond vendor-specific solutions



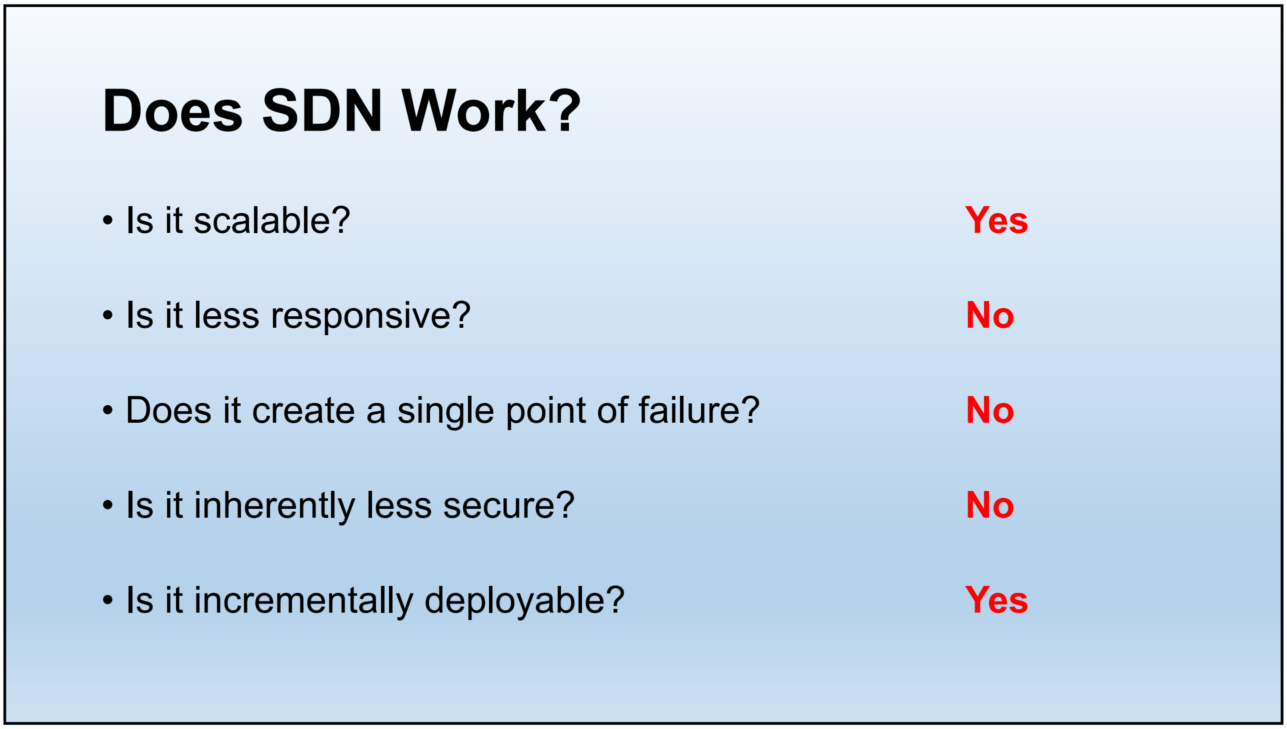
3. Forwarding Abstraction

- Switches have two “brains”
  - Management CPU (smart but slow)
  - Forwarding ASIC (fast but dumb)

- Need a forwarding abstraction for both
  - CPU abstraction can be almost anything

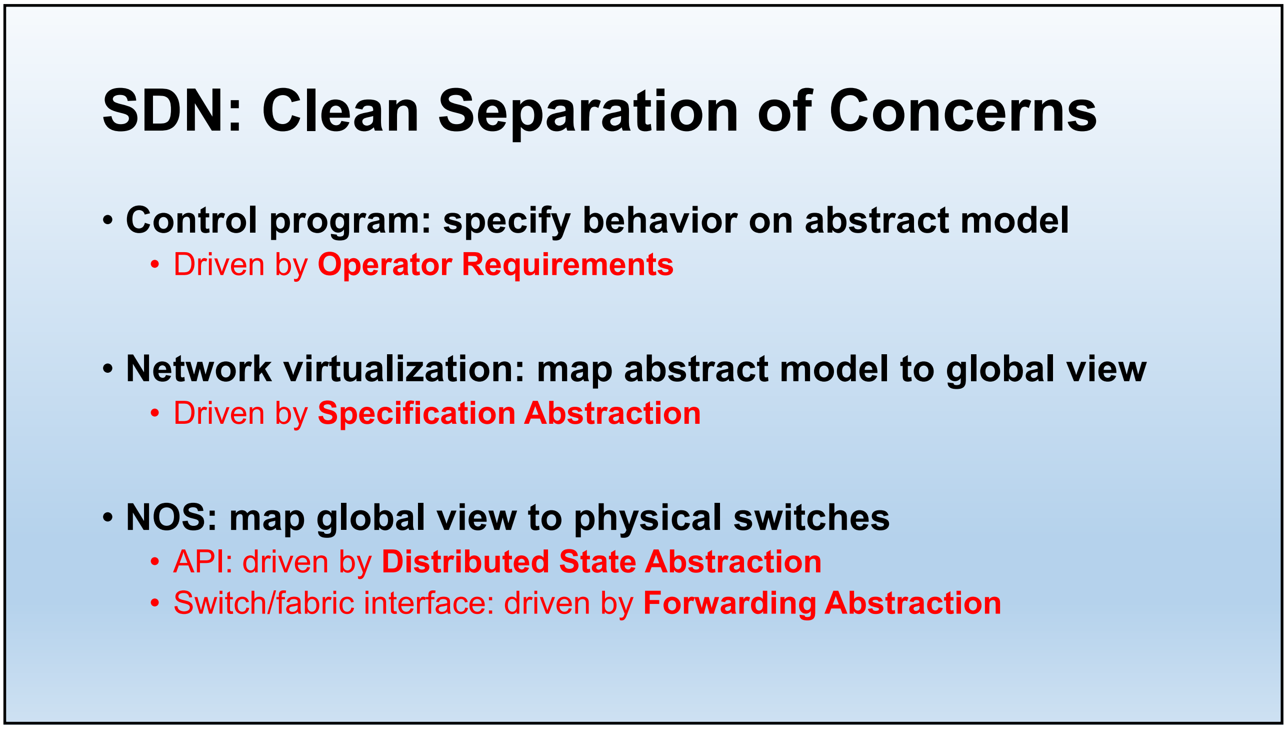
- ASIC abstraction is much more subtle: **OpenFlow**

- OpenFlow:
  - Control switch by inserting <header;action> entries
  - Essentially gives NOS remote access to forwarding table
  - Instantiated in OpenvSwitch



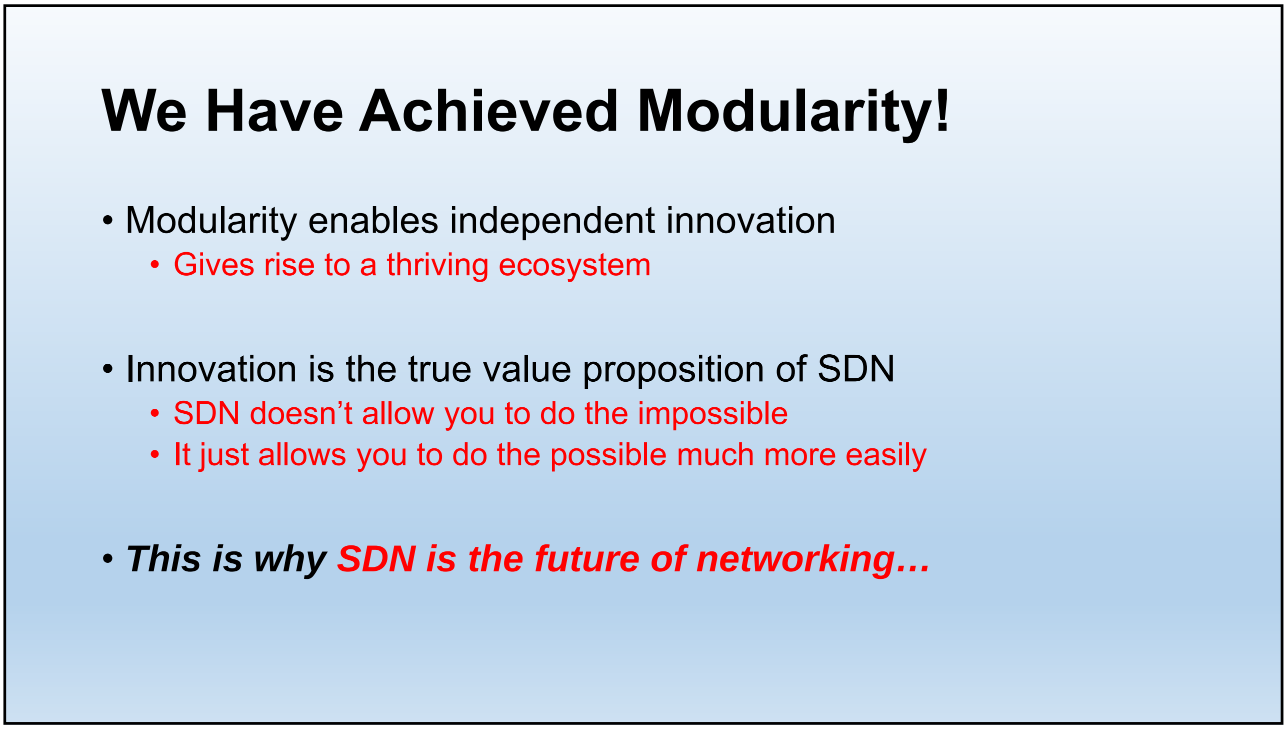
Does SDN Work?

• Is it scalable?	Yes
• Is it less responsive?	No
• Does it create a single point of failure?	No
• Is it inherently less secure?	No
• Is it incrementally deployable?	Yes



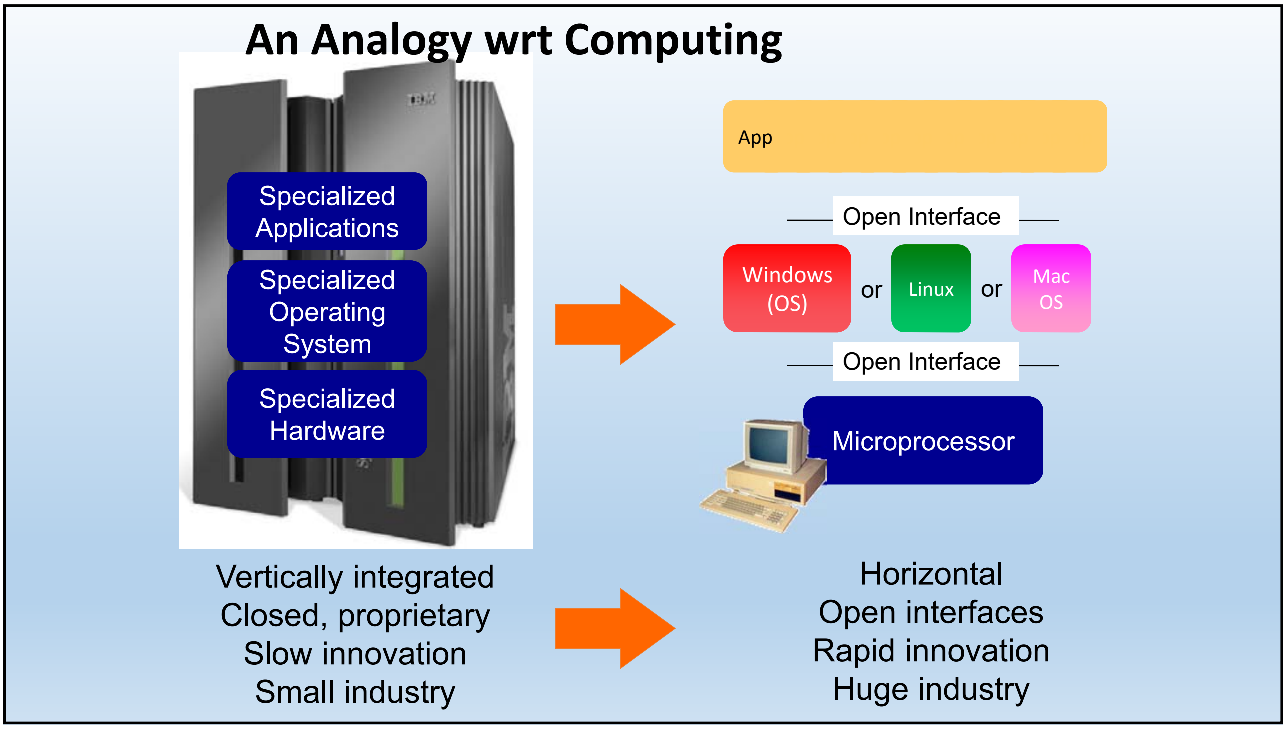
SDN: Clean Separation of Concerns

- **Control program: specify behavior on abstract model**
  - Driven by **Operator Requirements**
- **Network virtualization: map abstract model to global view**
  - Driven by **Specification Abstraction**
- **NOS: map global view to physical switches**
  - API: driven by **Distributed State Abstraction**
  - Switch/fabric interface: driven by **Forwarding Abstraction**



We Have Achieved Modularity!

- Modularity enables independent innovation
  - Gives rise to a thriving ecosystem
- Innovation is the true value proposition of SDN
  - SDN doesn't allow you to do the impossible
  - It just allows you to do the possible much more easily
- ***This is why SDN is the future of networking…***



An Analogy wrt Computing

Specialized Applications

Specialized Operating System

Specialized Hardware

App

Open Interface

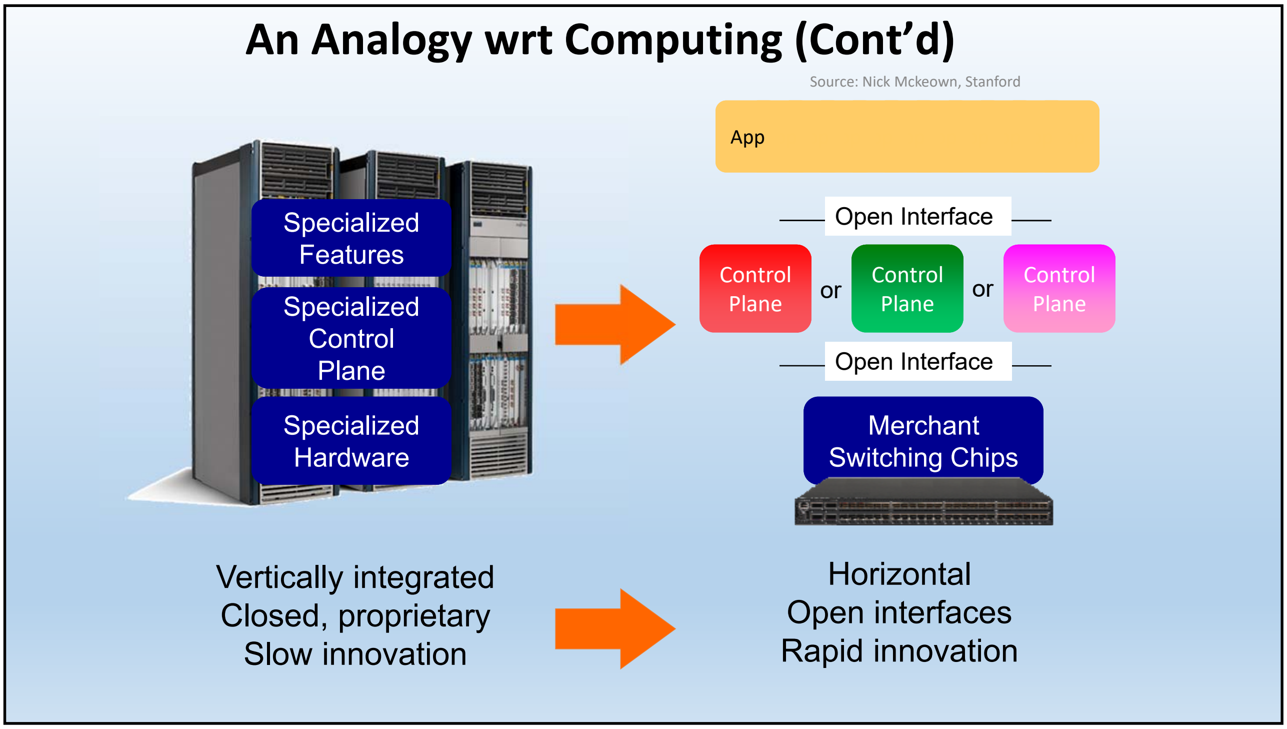
Windows (OS) or Linux or Mac OS

Open Interface

Microprocessor

Vertically integrated
Closed, proprietary
Slow innovation
Small industry

Horizontal
Open interfaces
Rapid innovation
Huge industry



An Analogy wrt Computing (Cont'd)

Source: Nick Mckeown, Stanford

Specialized Features

Specialized Control Plane

Specialized Hardware

App

Open Interface

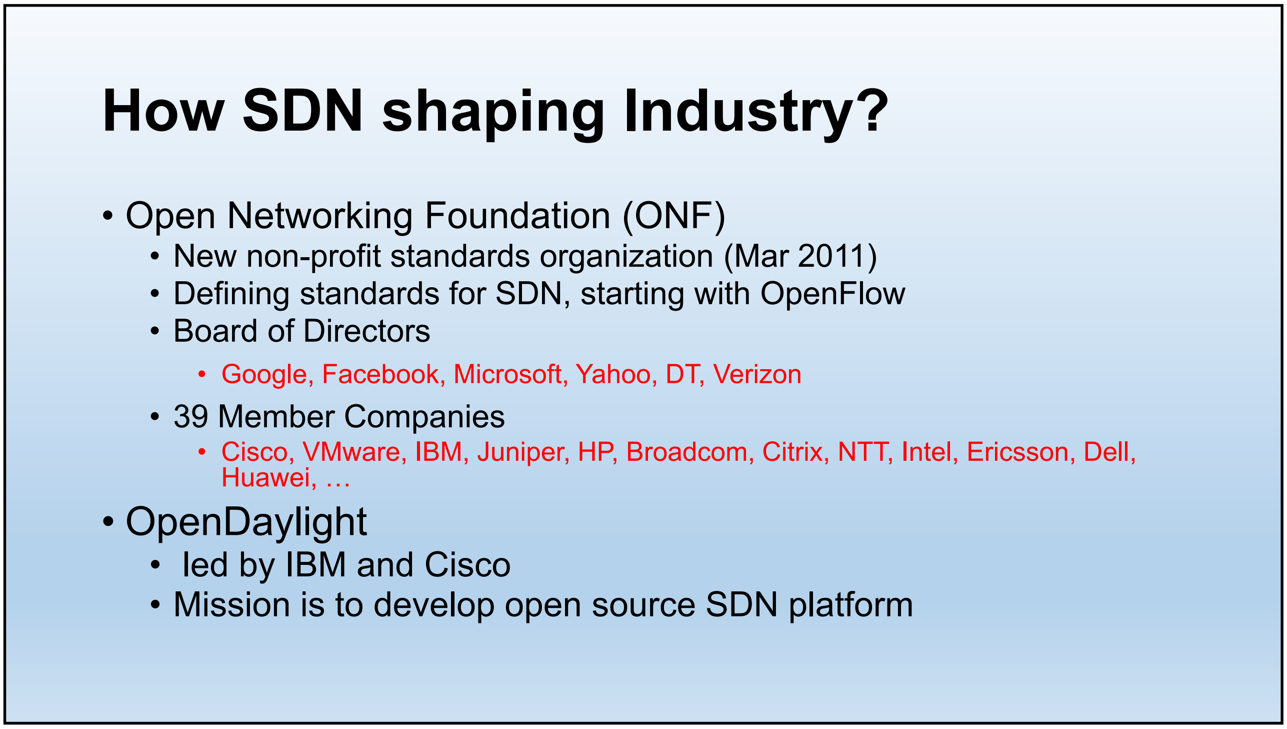
Control Plane or Control Plane or Control Plane

Open Interface

Merchant Switching Chips

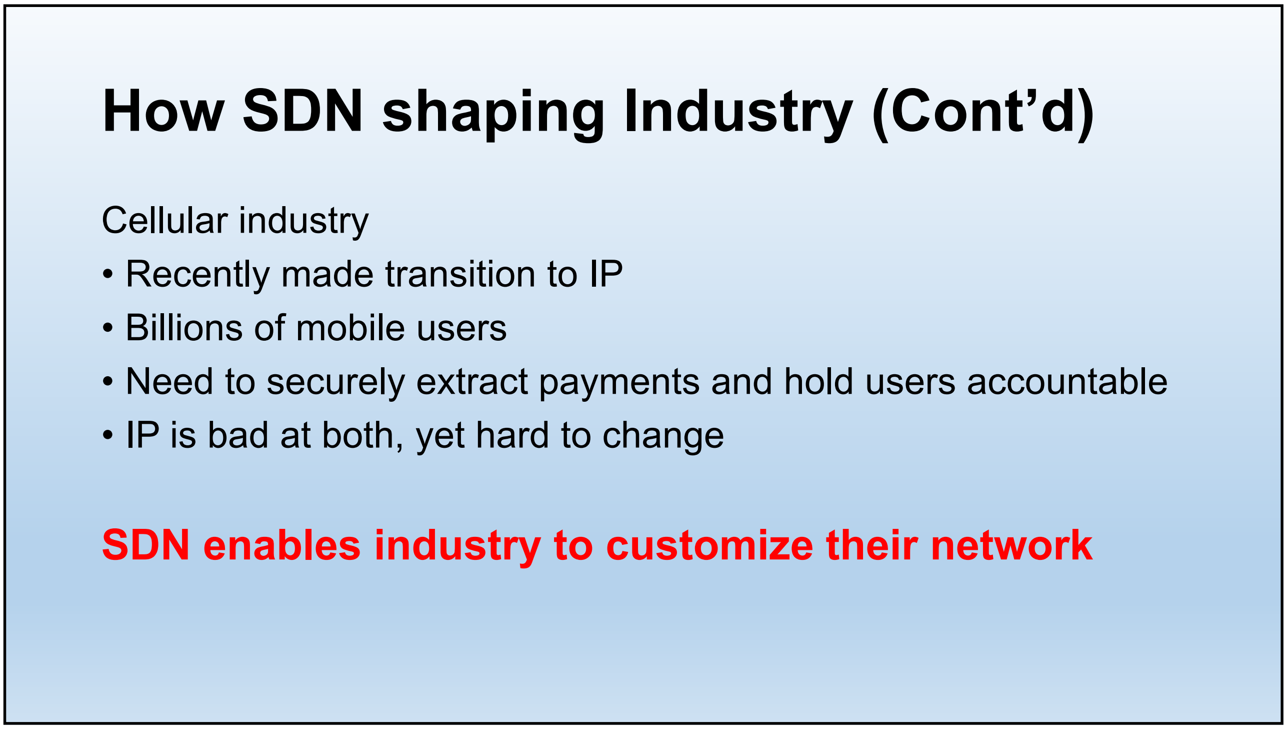
Vertically integrated
Closed, proprietary
Slow innovation

Horizontal
Open interfaces
Rapid innovation



How SDN shaping Industry?

- Open Networking Foundation (ONF)
  - New non-profit standards organization (Mar 2011)
  - Defining standards for SDN, starting with OpenFlow
  - Board of Directors
    - Google, Facebook, Microsoft, Yahoo, DT, Verizon
  - 39 Member Companies
    - Cisco, VMware, IBM, Juniper, HP, Broadcom, Citrix, NTT, Intel, Ericsson, Dell, Huawei, …
- OpenDaylight
  - led by IBM and Cisco
  - Mission is to develop open source SDN platform



How SDN shaping Industry (Cont’d)

Cellular industry

- Recently made transition to IP
- Billions of mobile users
- Need to securely extract payments and hold users accountable
- IP is bad at both, yet hard to change

SDN enables industry to customize their network

How SDN shaping Industry (Cont'd)

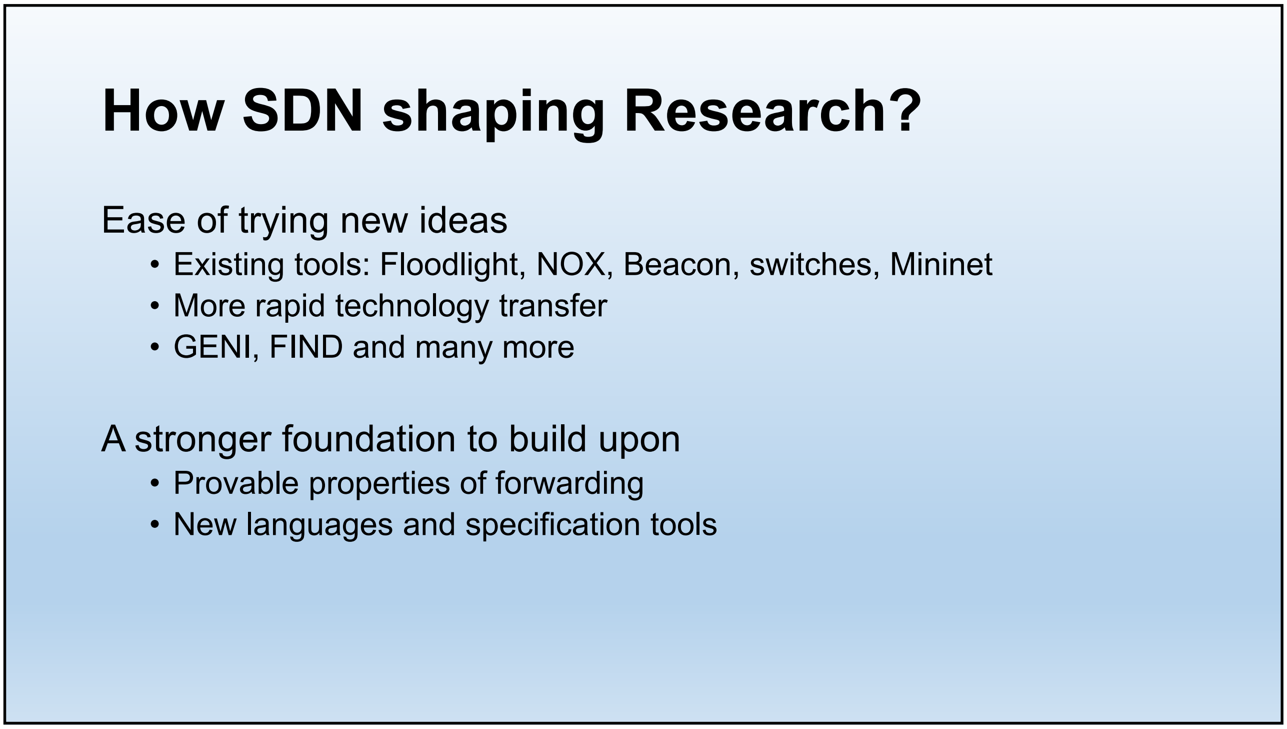
Big companies

- Google B4: deployed SDN to manage cross data center traffic
- Microsoft SWAN: software defined WAN
- Facebook: infrastructure team exploring SDN
- Vmware: Nicira, overlay approach to SDN
- Intel: OpenFlow switch
- Cisco: OpenFlow switch
- AT&T: Domain 2.0
- …

How SDN shaping Industry (Cont'd)

Startups

- Affirmed Networks: virtualized subscriber and content management tools for mobile operators
- Big Switch Networks: OpenFlow-based SDN switches, controllers and monitoring tools
- Embrane: layer 3-7 SDN services to enterprises and service providers
- Accelera: software defined wireless networks funded by Stanford Professor Andrea Goldsmith



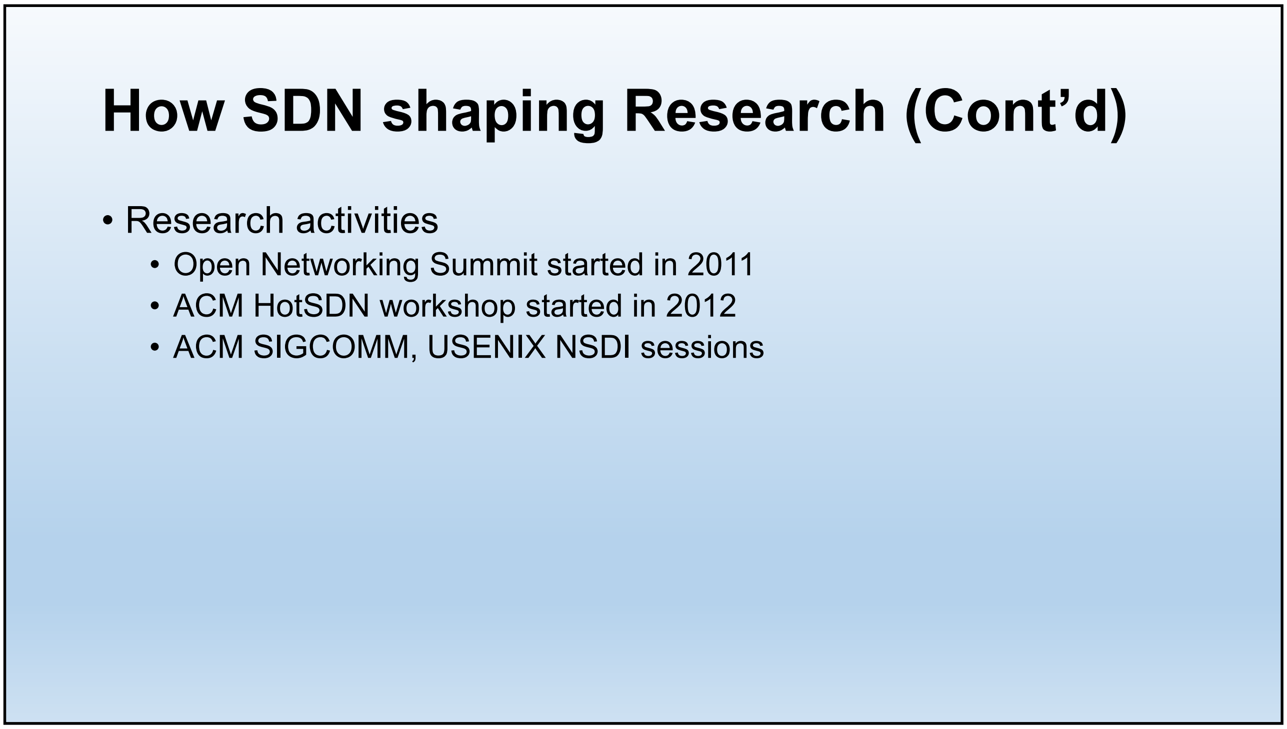
How SDN shaping Research?

Ease of trying new ideas

- Existing tools: Floodlight, NOX, Beacon, switches, Mininet
- More rapid technology transfer
- GENI, FIND and many more

A stronger foundation to build upon

- Provable properties of forwarding
- New languages and specification tools



How SDN shaping Research (Cont'd)

- Research activities
  - Open Networking Summit started in 2011
  - ACM HotSDN workshop started in 2012
  - ACM SIGCOMM, USENIX NSDI sessions