CSCI-1680
Software-Defined Networking

Rodrigo Fonseca

Most content from lecture notes by Scott Shenker

SDN

- **For now: a new paradigm for network management**
- **SDN widely accepted as "future of networking"**
  - ~1000 engineers at latest Open Networking Summit
  - Commercialized, in production use (few places)
    - E.g., controls Google's WAN; NTT moving to deploy
  - *Much more acceptance in industry than in academia*
- **An insane level of SDN hype, and backlash…**
  - SDN doesn't work miracles, merely makes things easier
- **If SDN is the solution, what is the problem?**

The Problem with Networking

- **So, what is the problem that justified such excitement?**
- **The management of networks**
  - Loosely, everything related to the *control plane*

- **The real problem: networking as a discipline is built on weak foundations**

Building an Artifact, Not a Discipline

- **Other fields in "systems": OS, DB, etc.**
  - Teach basic principles
  - Are easily managed
  - Continue to evolve

- **Networking:**
  - Study of an artifact: the Internet
  - Teach (mostly) big bag of protocols
  - Notoriously difficult to manage
  - Evolves very slowly

- ***Networks are much more primitive and less understood than other computer systems***

What is Network Management?

- **Recall the two "planes"**
- **Data plane: forwarding packets**
  - Based on local forwarding state
- **Control plane: computing that forwarding state**
  - Involves coordination with rest of system
- **Broad definition of "network management":**
  - ***Everything having to do with the control plane***

Original goals for the control plane

- **Basic connectivity: route packets to destination**
  - Local state computed by routing protocols
  - Globally distributed algorithms
- **Interdomain policy: find policy-compliant paths**
  - Done by fully distributed BGP
- **For long time, these were the only relevant goals!**
  - What other goals are there in running a network?

Also

- **Isolation**
- **Access Control**
- **Traffic Engineering**
- **…**

Isolation

- **Want multiple LANs on single physical network**
- **Packets on LAN don't pass through routers**
  - But routers used to impose various controls (later)
- **Use VLANs (virtual LANs) tags in L2 headers**
  - Controls where broadcast packets go
  - Gives support for logical L2 networks
  - Routers connect these logical L2 networks
- **No universal method for setting VLAN state**

Access Control

- **Operators want to limit access to various hosts**
  - Don't let laptops access backend database machines
- **This can be imposed by routers using ACLs**
  - ACL: Access control list
- **Example entry in ACL: <header template; drop>**

Traffic Engineering

- **Want to avoid persistent overloads on links**
- **Choose routes to spread traffic load across links**
- **Two main methods:**
  - Setting up MPLS tunnels
  - Adjusting weights in OSPF
- **Often done with centralized computation**
  - Take snapshot of topology
  - Compute appropriate MPLS/OSPF state
  - Send to network

Summarizing

- **Network management has many goals**
- **Achieving these goals is job of the control plane…**
- **…which currently involves many mechanisms**

Control Plane Mechanisms

- **Many different control plane mechanisms**
- **Designed from scratch for specific goal**
- **Variety of implementations**
  - **Globally distributed:** routing algorithms
  - **Manual/scripted configuration:** ACLs, VLANs
  - **Centralized computation:** Traffic engineering
- **Network control plane is a complicated mess!**

How Have We Managed To Survive?

- **Net. admins miraculously master this complexity**
  - Understand all aspects of networks
  - Must keep myriad details in mind
- **This ability to master complexity is both a blessing**
  - **…and a curse!**

Mastering Complexity versus Extracting Simplicity

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

- **Each has its role:**
  - When first getting systems to work, *master complexity*
  - When making system easy to use, *extract simplicity*

- **You will never succeed in extracting simplicity**
  - If you don't recognize it is a different skill set than mastering complexity

JOURNEY LATIN AMERICA

MANUAL PHOTOGRAPHY CHEAT SHEET

By Tom Parrott

EXPOSURE

+ | . . I . . 0 . . I . . | - Try to keep your light meter at 0.

+ | . . I . . 0 . . I . . | - OVEREXPOSED

+ | . . I . . 0 . . I . . | - UNDEREXPOSED

APERTURE

f/1.4 | f/2 | f/2.8 | f/4 | f/5.6 | f/8 | f/11 | f/16 ❯

SHALLOW DEPTH OF FIELD — DEEP DEPTH OF FIELD

BRIGHTER — DARKER

SHUTTER

10" | 2" | 1" | 1/25 | 1/30 | 1/50 | 1/100 | 1/125 | 1/250 | 1/320 | 1/500

❮ ❯

LONGER EXPOSURE
To capture things that don't move or leave streaks of light if they do.

SHORTER EXPOSURE
To capture movement.

BRIGHTER PHOTOGRAPH — DARKER PHOTOGRAPH

ISO (Film Speed)

❮ 100 200 400 800 1600 3200 HI2 ❯

LOW SENSITIVITY TO LIGHT
USE DURING DAY TIME & SUNSHINE
HIGHER QUALITY (SMOOTH)

HIGH SENSITIVITY TO LIGHT
USE DURING NIGHT & LOW LIGHT INDOORS
LOWER QUALITY (GRAINY)

Mastering Complexity versus Extracting Simplicity

- **Networking has never made the distinction…**
  - And therefore has never made the transition from mastering complexity to extracting simplicity
- **Still focused on mastering complexity**
  - Networking "experts" are those that know all the details
- ***Extracting simplicity lays intellectual foundations***
  - This is why networking has weak foundation
  - We are ***still*** building the artifact, not the discipline

Why make the transition

- **Complexity has increased to "unmanageable" levels**
- **Consider datacenters:**
  - 100,000s machines, 10,000s switches
  - 1000s of customers
    - Each with their own logical networks: ACLs, VLANs, etc
- **Way beyond what we can handle**
  - Leads to brittle, ossified configurations
  - Probably inefficient too

An Example Transition: Programming

- **Machine languages: no abstractions**
  - Had to deal with low-level details
  - 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 About Networking Abstractions?

- **Consider the data and control planes separately**
- **Different tasks, so naturally different abstractions**

Abstractions for Data Plane: Layers

Applications

…built on…

Reliable (or unreliable) transport

…built on…

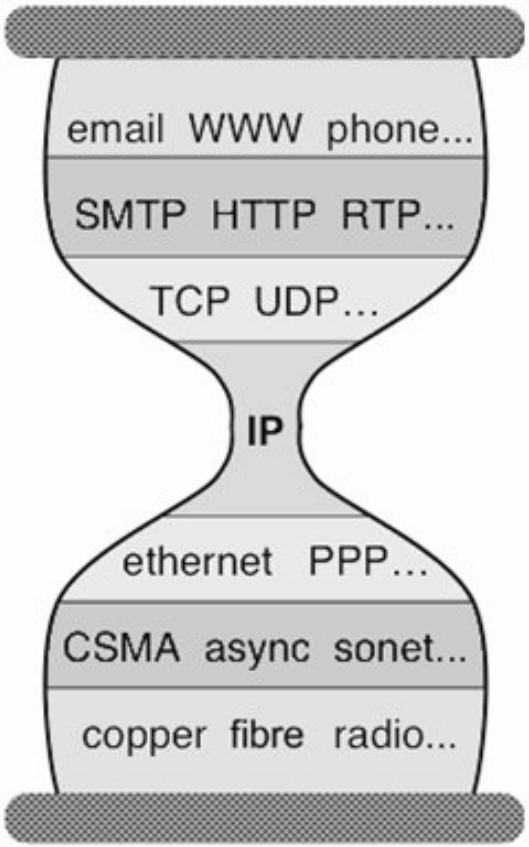
Best-effort global packet delivery

…built on…

Best-effort local packet delivery

…built on…

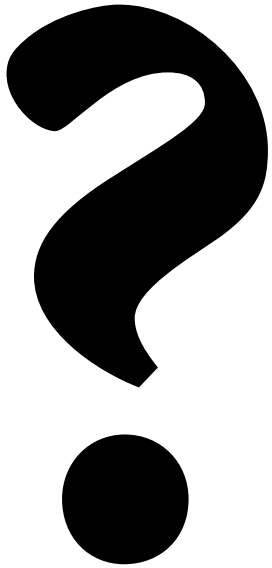
Physical transfer of bits



The Importance of Layering

- **Decomposed delivery into basic components**
- **Independent, compatible innovation at each layer**
  - Clean "separation of concerns"
  - Leaving each layer to solve a tractable problem
- **Responsible for the success of the Internet!**
  - Rich ecosystem of independent innovation

Control Plane Abstractions



(Too) Many Control Plane Mechanisms

- **Variety of goals, no modularity:**
  - **Routing:** distributed routing algorithms
  - **Isolation**: ACLs, VLANs, Firewalls,…
  - **Traffic engineering**: adjusting weights, MPLS,…

- **Control Plane: mechanism without abstraction**
  - ***Too many mechanisms, not enough functionality***

Finding Control Plane Abstractions

How do you find abstractions?

- You first decompose the problem….
- …and define abstractions for each subproblem
- So what is the control plane problem?

Task: Compute forwarding state:

- **Consistent with low-level hardware/software**
  - Which might depend on particular vendor
- **Based on entire network topology**
  - Because many control decisions depend on topology
- **For all routers/switches in network**
  - Every router/switch needs forwarding state

Our current approach

- **Design one-off mechanisms that solve all three**
  - A sign of how much we love complexity
- **No other field would deal with such a problem!**
- **They would define abstractions for each subtask**
- **…and so should we!**

Separate Concerns with Abstractions

1. **Be compatible with low-level hardware/software**

   Need an abstraction for general **forwarding model**

2. **Make decisions based on entire network**

   Need an abstraction for **network state**

3. **Compute configuration of each physical device**

   Need an abstraction that **simplifies configuration**

Abs#1: Forwarding Abstraction

- **Express intent independent of implementation**
  - Don't want to deal with proprietary HW and SW
- **OpenFlow is current proposal for forwarding**
  - Standardized interface to switch
  - Configuration in terms of flow entries: <header, action>
- **Design details concern exact nature of:**
  - Header matching
  - Allowed actions

Two Important Facets to OpenFlow

- **Switches accept external control messages**
  - Not closed, proprietary boxes
- **Standardized flow entry format**
  - So switches are interchangeable

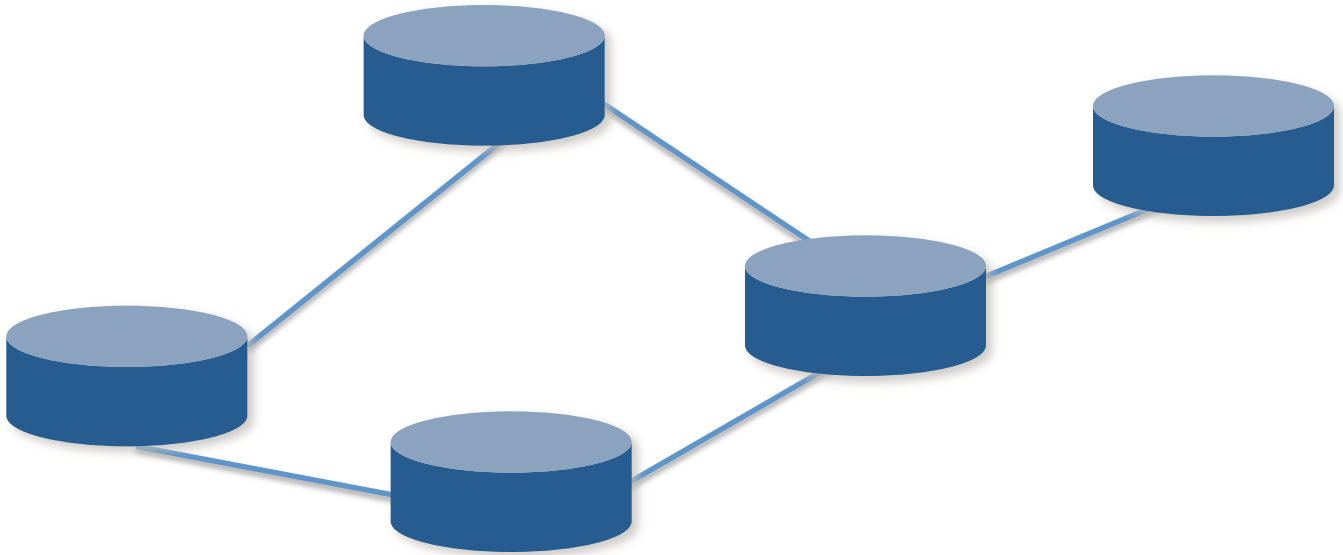
Abs#2: Network State Abstraction

- **Abstract away various distributed mechanisms**
- **Abstraction: global network view**
  - Annotated network graph provided through an API
- **Implementation: "Network Operating System"**
  - Runs on servers in network ("controllers")
  - Replicated for reliability
- **Information flows both ways**
  - Information *from* routers/switches to form "view"
  - Configurations *to* routers/switches to control forwarding

Network Operating System

- **Think of it as a centralized link-state algorithm**
- **Switches send connectivity info to controller**
- **Controller computes forwarding state**
  - Some control program that uses the topology as input
- **Controller sends forwarding state to switches**
- **Controller is replicated for resilience**
  - System is only "logically centralized"

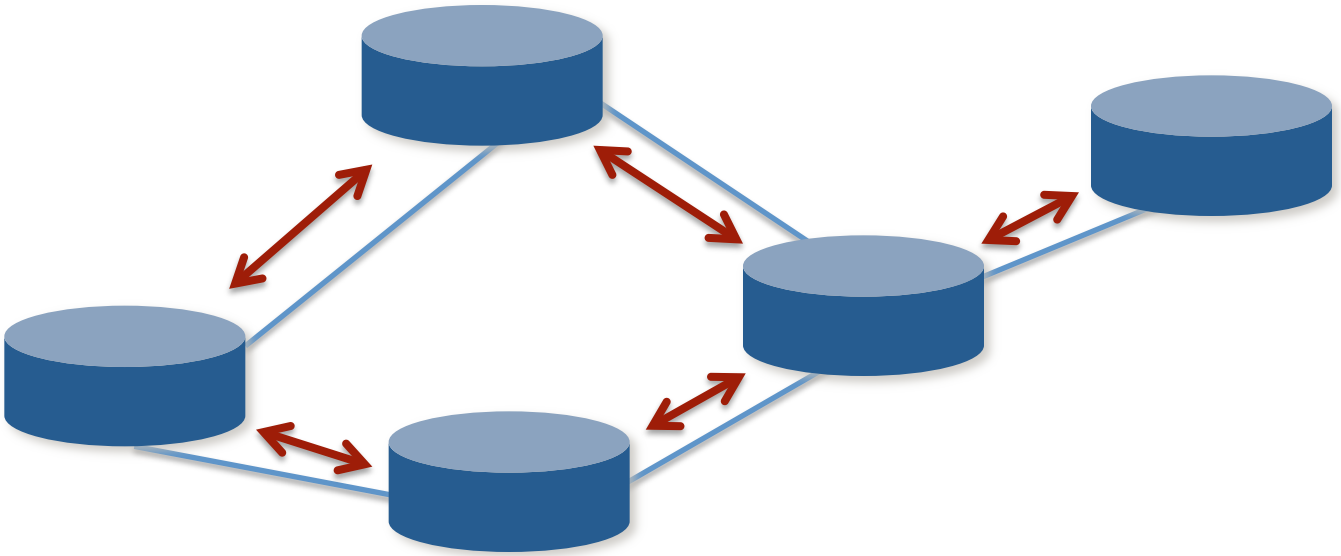
Network of Switches and/or Routers



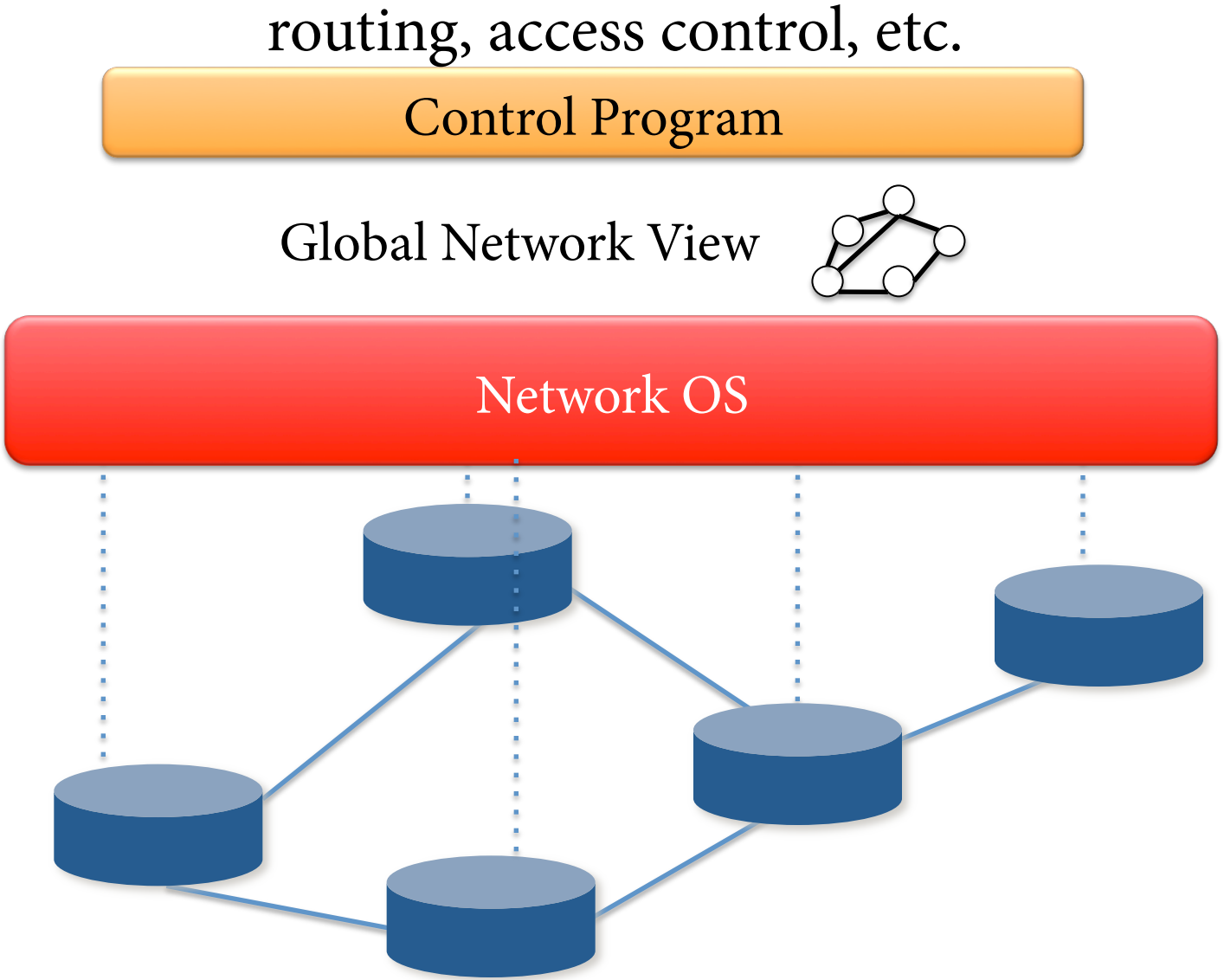
Traditional Control Mechanisms

Distributed algorithm running between neighbors

Complicated task-specific distributed algorithm



Software Defined Network (SDN)



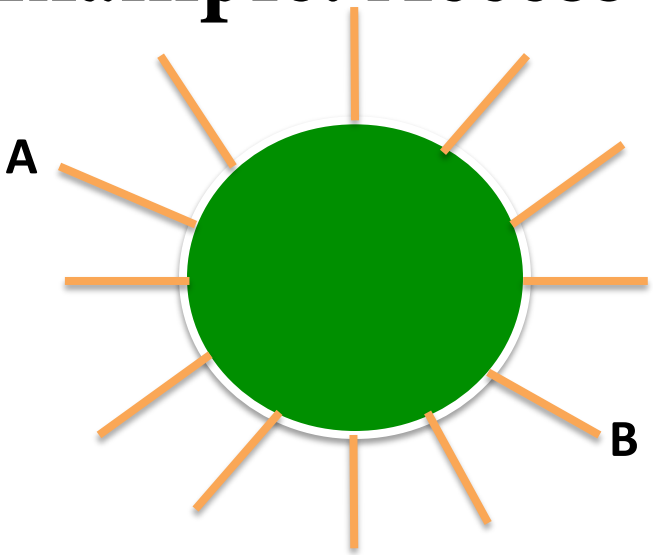
Major Change in Paradigm

- **Control program:**
  - **Configuration = Function(view)**
- **Control mechanism now program using NOS API**
- **Not a distributed protocol, just a graph algorithm**

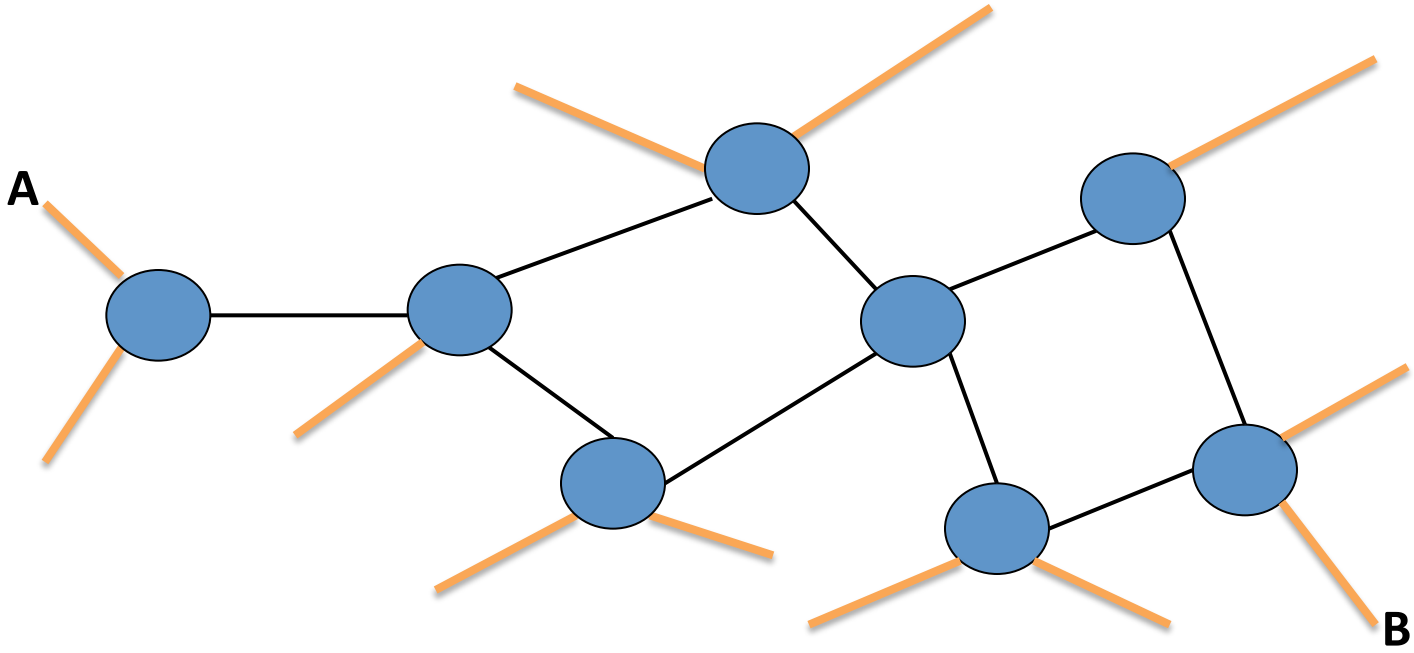
Abs#3: Specification Abstraction

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

Simple Example: Access Control



Abstract
Network
View



Global
Network
View

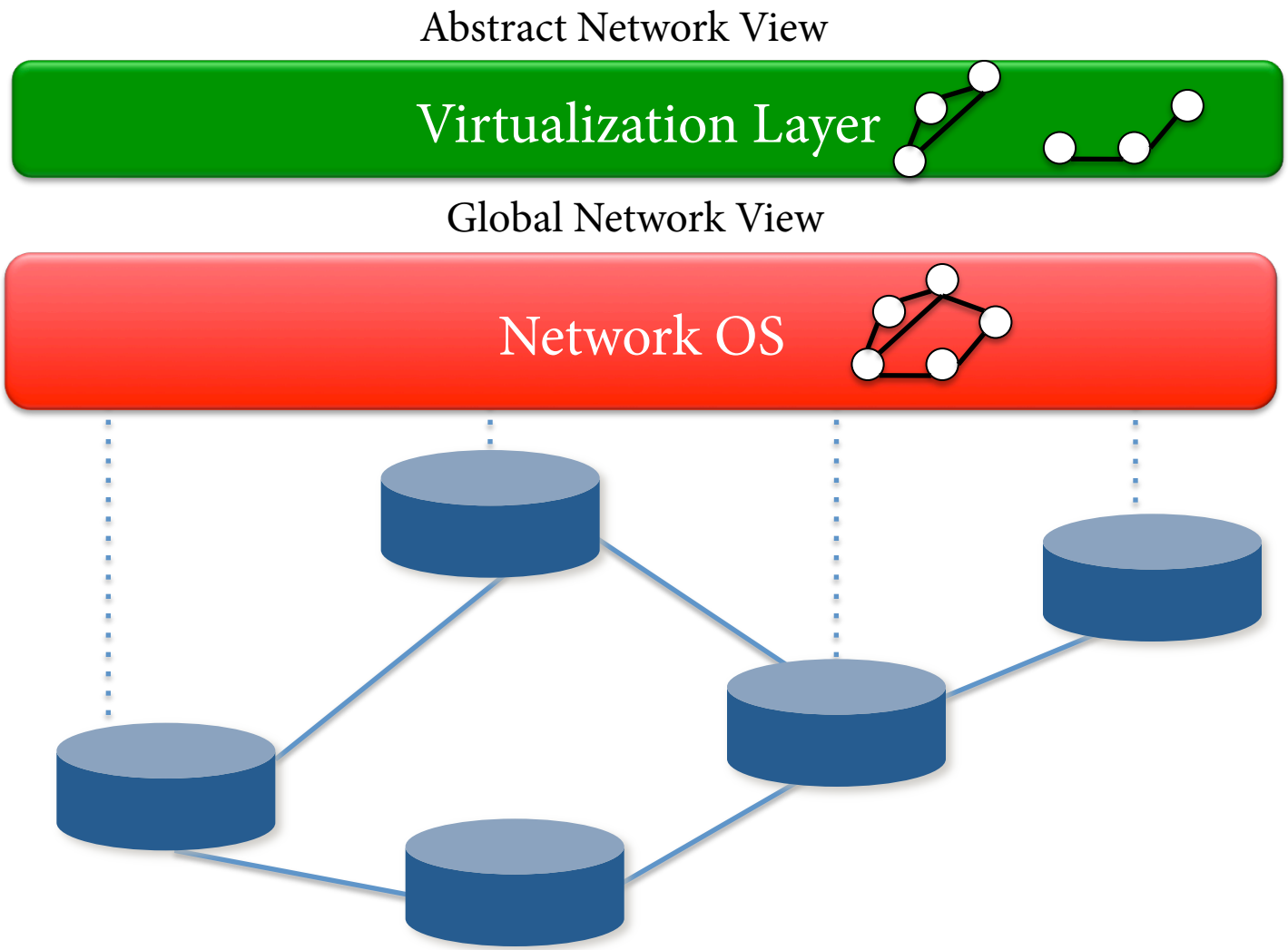
Routing

- Look at graph of network
- Compute routes
- Give to SDN platform, which passes on to switches

Access Control

- **Control program decides who can talk to who**
- **Pass this information to SDN platform**
- **Appropriate ACL flow entries are added to network**
  - In the right places (based on the topology)

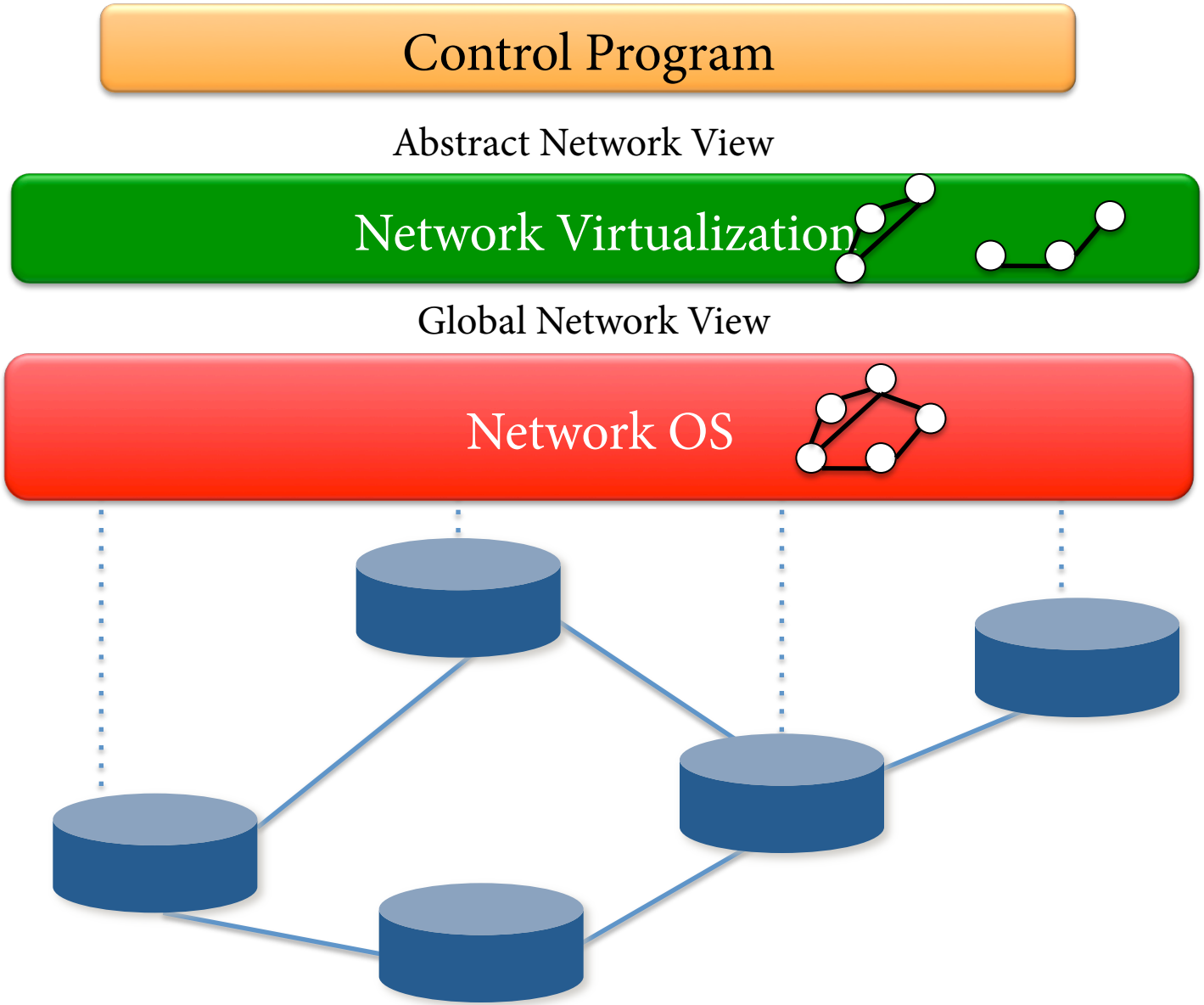
Software Defined Network



Clean Separation of Concerns

- **Control program: express goals on abstract view**
  - Driven by **Operator Requirements**
- **Virtualization Layer: abstract view ⟷ global view**
  - Driven by **Specification Abstraction** for particular task
- **NOS: global view ⟷ physical switches**
  - API: driven by **Network State Abstraction**
  - Switch interface: driven by **Forwarding Abstraction**

SDN: *Layers* for the Control Plane



Abstractions for Control Plane

Expression of Intent

…built on…

Abstract Network View

…built on…

Global Network View

…built on…

Physical Topology

Abstractions Don't Remove Complexity

- **NOS, Virtualization are complicated pieces of code**
- **SDN merely localizes the complexity:**
  - Simplifies interface for control program (user-specific)
  - Pushes complexity into ***reusable*** code (SDN platform)
- **This is the big payoff of SDN: modularity!**
  - The core distribution mechanisms can be reused
  - Control programs only deal with their specific function
- **Note that SDN separates control and data planes**
  - SDN platform does control plane, switches do data plane

What This Really Means

Separation of Control/Data Plane

- **Today, routers implement both**
  - They forward packets
  - And run the control plane software
- **SDN networks**
  - Data plane implemented by switches
    - Switches act on local forwarding state
  - Control plane implemented by controllers
    - All forwarding state computed by SDN platform
- **This is a technical change, with broad implications**

Changes

- **Less vendor lock-in**
  - Can buy HW/SF from different vendors
- **Changes are easier**
  - Can test components separately
    - *HW has to forward*
    - *Can simulate controller*
    - *Can do verification on logical policy*
  - Can change topology and policy independently
  - Can move from private net to cloud and back!
  - Greater rate of innovation

Current Status of SDN

- **SDN widely accepted as "future of networking"**
  - ~1000 engineers at latest Open Networking Summit
  - Commercialized, in production use (few places)
    - E.g., controls Google's WAN; NTT moving to deploy
  - *Much more acceptance in industry than in academia*
- **Insane level of SDN hype, and backlash…**
  - SDN doesn't work miracles, merely makes things easier
- **Open Networking Foundation (72 members)**
  - **Board**: Google, Yahoo, Verizon, DT, Msoft, F'book, NTT
  - **Members**: Cisco, Juniper, HP, Dell, Broadcom, IBM,…
- **Watch out for upcoming chapters!**

To learn more…

- **Scott Shenker's talk "The Future of Networking, and the Past of Protocols"**
  - http://www.youtube.com/watch?v=YHeyuD89n1Y
  - Keynote at the 2011 Open Networking Summit
- **Take my graduate seminar next semester**
  - Advanced Networking: SDNs and Datacenter Networking