CSCI-1680
Software-Defined Networking

Rodrigo Fonseca

With content from Scott Shenker, Nick McKeown
SDN

• For now: a new paradigm for network management
• SDN enables management at cloud scales
  – ~1000 engineers at latest Open Networking Summit
  – Commercialized, in production use
    • Controls Google’s WAN; Microsoft and Google cloud offerings
    • VMWare’s main networking product
    • Drives many OpenStack network deployments
  – Strong acceptance in industry and academia

• High level of SDN hype, and backlash…
  – Nicira bought by VMWare in 2012 for $1.2B
  – 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
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
- ...

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!
### MANUAL PHOTOGRAPHY CHEAT SHEET

**Journey Latin America**

**By Tom Parrott**

#### EXPOSURE

- Try to keep your light meter at 0.
- OVEREXPOSED
- UNDEREXPOSED

#### APERTURE

<table>
<thead>
<tr>
<th>f/1.4</th>
<th>f/2</th>
<th>f/2.8</th>
<th>f/4</th>
<th>f/5.6</th>
<th>f/8</th>
<th>f/11</th>
<th>f/16</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMALL DEPTH OF FIELD</td>
<td>DEEP DEPTH OF FIELD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### SHUTTER

<table>
<thead>
<tr>
<th>10&quot;</th>
<th>2&quot;</th>
<th>1&quot;</th>
<th>1/25</th>
<th>1/50</th>
<th>1/100</th>
<th>1/250</th>
<th>1/1250</th>
<th>1/500</th>
</tr>
</thead>
<tbody>
<tr>
<td>LONGER EXPOSURE</td>
<td>SHORTER EXPOSURE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To capture things that don’t move or leave streaks of light if they do.  To capture movement.

#### ISO (Film Speed)

<table>
<thead>
<tr>
<th>100</th>
<th>200</th>
<th>400</th>
<th>800</th>
<th>1600</th>
<th>3200</th>
<th>HI2</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOW SENSITIVITY TO LIGHT</td>
<td>HIGH SENSITIVITY TO LIGHT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Linux Observability tools by Brendan Gregg, brendangregg.com
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
Number of published Internet Standards

Graph from Nick McKeown
Cisco Stock Price

1991 - 1999

200x
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

• Control Plane: mechanisms without abstraction
  – Too many mechanisms, not enough functionality

• Variety of goals, no modularity:
  – Routing: distributed routing algorithms
  – Isolation: ACLs, VLANs, Firewalls,…
  – Traffic engineering: adjusting weights, MPLS,…
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!
Example

- OSPF:
  - 5% for Djikstra’s algorithm,
  - 95% to find and maintain the state of the network
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 fields, action>

• **P4** is a much more general language for forwarding
  – Programmable switches

• **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)

- Control Program
- Global Network View
- Network OS

Routing, access control, etc.

Software

Very simple hardware
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)
Clean Separation of Concerns

• Control program: express goals on abstract view
  – Driven by Operator Requirements

• Virtualization Layer: abstract view $\leftrightarrow$ global view
  – Driven by Specification Abstraction for particular task

• NOS: global view $\leftrightarrow$ physical switches
  – API: driven by Network State Abstraction
  – Switch interface: driven by Forwarding Abstraction
SDN: *Layers* for the Control Plane

- **Control Program**
- **Network Virtualization**
- **Network OS**
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
Computer Industry

Specialized Applications
Specialized Operating System
Specialized Hardware

Open Interface
Windows (OS) or Linux or Mac OS

Open Interface
Microprocessor
Dell Stock Price

$42 → $14

2005 → 2013

Google Finance
Networking Industry

- Switch Chips
- Specialized Operating System
- Specialized Hardware
- Open Interface
- NOX
- Beacon
- ONI X
- POX
- ONOS
- Flood light
- Trema
- ODL
- Ryu

Switch Chips

- Broadcom
- Stratix 3100

Open Interface
Current Status of SDN

- SDN widely accepted as “future of networking”
  - Commercial use inter-datacenter (Google), intra-datacenter (Microsoft)
  - Network virtualization is current killer app
    - VMWare’s NSX, OpenStack network management

- A lot of SDN hype, and backlash…
  - SDN doesn’t work miracles, merely makes things easier

- Open Networking Foundation (100+ 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

• NEC SDN Reading List
  – http://www.nec-labs.com/~lume/sdn-reading-list.html

• The Road to SDN
  – http://queue.acm.org/detail.cfm?id=2560327
OpenFlow

- Simple API between switches and centralized controller
- Basic abstraction: flow match / action
  - E.g., if a packet matches this IP dest, ETH protocol type, forward on port 3
  - If a packet matches ARP, send to controller
  - If a packet comes from evil IP address, drop