CompSci 514: Computer Networks

Lecture 11: Software Defined Networking

Xiaowei Yang
Overview

• Introducing SDN

• A real-world application of SDN
  – Google’s B4 network
Software Defined Networking

Slides adapted from Mohammad Alizadeh (MIT)’s SDN lecture
Outline

• Networking before SDN

• What is SDN?

• OpenFlow basics

• Why is SDN happening now? (a brief history)
Networking before SDN
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”
The Networking “Planes”

- **Data plane**: processing and delivery of packets with local forwarding state
  - Forwarding state + packet header $\rightarrow$ 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

<table>
<thead>
<tr>
<th>Time-scale</th>
<th>Data</th>
<th>Control</th>
<th>Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet (nsec)</td>
<td>Event (10 msec to sec)</td>
<td>Human (min to hours)</td>
<td></td>
</tr>
<tr>
<td>Linecard hardware</td>
<td>Router software</td>
<td>Humans or scripts</td>
<td></td>
</tr>
</tbody>
</table>
Data and Control Planes

- Processor
- Switching Fabric
- Line card
  - Data plane
  - Control plane
Data Plane

• Streaming algorithms on packets
  – Matching on some header bits
  – Perform some actions

• Example: IP Forwarding

```
1.2.3.4  1.2.3.7  1.2.3.156
  host   host    ...  host
  LAN 1

      route

1.2.3.0/24

5.6.7.8  5.6.7.9
  host   host    ...  host
  LAN 2

      route

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
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 control
  - Must invert protocol behavior, "coax" it to do what you want
  - Ex. Changing weights instead of paths for TE

Uncoordinated control
  - Cannot control which router updates first

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

Chicago (chi)

Data Center

New York (nyc)

Front Office

<table>
<thead>
<tr>
<th></th>
<th>chi-DC</th>
<th>chi-FO</th>
<th>nyc-DC</th>
<th>nyc-FO</th>
</tr>
</thead>
<tbody>
<tr>
<td>chi-DC</td>
<td>![Green Circle](Green Circle)</td>
<td>![Red Circle](Red Circle)</td>
<td>![Green Circle](Green Circle)</td>
<td></td>
</tr>
<tr>
<td>chi-FO</td>
<td>![Red Circle](Red Circle)</td>
<td>![Green Circle](Green Circle)</td>
<td>![Red Circle](Red Circle)</td>
<td></td>
</tr>
<tr>
<td>nyc-DC</td>
<td>![Green Circle](Green Circle)</td>
<td>![Red Circle](Red Circle)</td>
<td>![Green Circle](Green Circle)</td>
<td></td>
</tr>
<tr>
<td>nyc-FO</td>
<td>![Red Circle](Red Circle)</td>
<td>![Green Circle](Green Circle)</td>
<td>![Red Circle](Red Circle)</td>
<td></td>
</tr>
</tbody>
</table>
Example 2: Access Control

Packet filter: Drop nyc-FO -> *
Permit *

Packet filter: Drop chi-FO -> *
Permit *

<table>
<thead>
<tr>
<th>chi-DC</th>
<th>chi-FO</th>
<th>nyc-DC</th>
<th>nyc-FO</th>
</tr>
</thead>
<tbody>
<tr>
<td>√</td>
<td>√</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>
- A new short-cut link added between data centers
- Intended for backup traffic between centers

Example 2: Access Control

Packet filter:
Drop nyc-FO -> *
Permit *

Packet filter:
Drop chi-FO -> *
Permit *
• Oops – new link lets packets violate access control policy!
• Routing changed, but
• Packet filters don’t update automatically
Software Defined Network

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

\textit{and}

A single (logically centralized) control plane controls several forwarding devices.
Software Defined Network (SDN)

Global Network Map

Control Plane

Control Program

Control Program

Control Program

Control
Forwarding

Control
Forwarding

Control
Forwarding

Control
Forwarding

Control
Forwarding
A Major Trend in Networking

Entire backbone runs on SDN

Bought for $1.2 billion (mostly cash)
How SDN Changes the Network

Network OS

Custom Hardware

OS

Feature

Feature
Software Defined Network (SDN)

1. Open interface to packet forwarding
2. At least one Network OS probably many.
   - Open- and closed-source
3. Consistent, up-to-date global network view
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 A

Control Program B

Network OS

Packet Forwarding

Packet Forwarding

Packet Forwarding

Packet Forwarding
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

Virtual Topology
Network Hypervisor

Global Network View
Network OS
Virtualization Simplifies Control Program

Hypervisor then inserts flow entries as needed.

Abstract Network View

Global Network View
Does SDN Simplify the Network?
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

Network OS

Control Program A

Control Program B

OpenFlow Protocol

Ethernet Switch
OpenFlow Basics

Control Program A | Control Program B

Network OS

Packet Forwarding

Flow Table(s)

Packet Forwarding

Packet Forwarding

"If header = p, send to port 4"
"If header = q, overwrite header with r, add header s, and send to ports 5,6"
"If header = ?, send to me"
Primitives <Match, Action>

**Match** arbitrary bits in headers:

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

![Header and Data Table]

Match: 1000x01xx0101001x

**Action**

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

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
B4: Experience with a Globally-Deployed Software Defined WAN

Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jonathan Zolla, Urs Hölzle, Stephen Stuart, and Amin Vahdat (Google)
What’s B4?

- Google’s private WAN
- Delivering traffic between its own datacenters

Figure 1: B4 worldwide deployment (2011).
Why SDN?

• Traffic engineering
  – Control of paths
  – Control of priority
B4 Architecture

Figure 2: B4 architecture overview.
B4 switch

Figure 3: A custom-built switch and its topology.

• Custom built with merchant silicon
Interactions between routing and open flow

• Initially, all routing protocols run on switches
• Later OFA forwards all routing packets to OFC
Centralized Traffic Engineering
Experience from an outage

• A switch is configured with duplicate ID

• Leads to route flaps

• Routing messages explode

• No new paths can be established
Summary

• Software Defined Networking
  – A standard interface for exercising control in the network

• A case study
  – B4