Announcements

Office Hours
3-4pm today in Upson 4137

Homework #1
- Goes out today (on CMS)
- Due in two weeks
- Topic: Basic OpenFlow programming
Vertically integrated
Closed, proprietary
Slow innovation
Small industry

Specialized Applications
Specialized Operating System
Specialized Hardware

Horizontal
Open interfaces
Rapid innovation
Huge industry

Microprocessor

Open Interface

Windows (OS) or Linux or Mac OS
Vertically integrated
Closed, proprietary
Slow innovation

Horizontal
Open interfaces
Rapid innovation
Today
Closed Boxes, Fully Distributed Protocols
Software Defined Network (SDN)

1. Open interface to packet forwarding
2. At least one Network OS
   - Open- and closed-source

Global Network View
Software Defined Network (SDN)

Abstract Network View

Network Virtualization

Global Network View

Network OS

Packet Forwarding

Control Programs

Control logics

Control platform
Control Logic

Runs on one or more controllers

Manages computation of forwarding state and perhaps coordination among instances

Control platform provides basic services to ease the latter (e.g., state distribution mechanisms)

Logic must decide how to partition computation, deal with failover, and implement the consistency model
Control Platform

Schedule computations over the network graph

Store network state and support for different consistency models

Most control platforms today run a single application

- Not clear yet how to resolve interference (e.g., policy routing vs. traffic engineering)
- We’re trying to answer some of these questions in the Frenetic project
State Distribution Abstraction

Control program should not have to handle all distributed-state details

Proposed abstraction: global network view

Control program operates on network view

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

Network OS provides network view
Forwarding Abstraction

Forwarding behavior specified by a control program.
Possibilities: x86, MPLS, OpenFlow
Example

OSPF
Distributed System

IS-IS
Distributed System

OS

Custom Hardware

Distributed System

Network OS

OSPF (Dijkstra)

IS-IS

New!

Packet Forwarding

Packet Forwarding

Packet Forwarding

Packet Forwarding
OpenFlow Forwarding Abstraction

Control Program A

Control Program B

Network OS

Packet Forwarding

Flow Table(s)

“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”
How does OpenFlow work?
OpenFlow Controller

OpenFlow Protocol (SSL/TCP)

Control Path

OpenFlow

Data Path (Hardware)
OpenFlow Forwarding Abstraction

**Patterns**

- Allows any flow granularity

**Example:** 1000x01xx0101001x

**Actions**

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

OpenFlow Client

Flow Table

<table>
<thead>
<tr>
<th>MAC src</th>
<th>MAC dst</th>
<th>IP Src</th>
<th>IP Dst</th>
<th>TCP sport</th>
<th>TCP dport</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>5.6.7.8</td>
<td>*</td>
<td>*</td>
<td>port 1</td>
</tr>
</tbody>
</table>

Controller

PC

Software Layer

Hardware Layer
OpenFlow Flow Tables

<table>
<thead>
<tr>
<th>Rule</th>
<th>Action</th>
<th>Stats</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Packet + byte counters</td>
</tr>
</tbody>
</table>

1. Forward packet to zero or more ports
2. Encapsulate and forward to controller
3. Send to normal processing pipeline
4. Modify Fields
5. Any extensions you add!

+ mask what fields to match
## Examples

### Switching

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>VLAN ID</th>
<th>IP Src</th>
<th>IP Dst</th>
<th>IP Prot</th>
<th>TCP sport</th>
<th>TCP dport</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>*</td>
<td>00:1f:..</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>port6</td>
</tr>
</tbody>
</table>

### Flow Switching

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>VLAN ID</th>
<th>IP Src</th>
<th>IP Dst</th>
<th>IP Prot</th>
<th>TCP sport</th>
<th>TCP dport</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>port3</td>
<td>00:20..</td>
<td>00:1f..0800</td>
<td>vlan1</td>
<td>1.2.3.4</td>
<td>5.6.7.8</td>
<td>4</td>
<td>17264</td>
<td>80</td>
<td>port6</td>
<td></td>
</tr>
</tbody>
</table>

### Firewall

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>VLAN ID</th>
<th>IP Src</th>
<th>IP Dst</th>
<th>IP Prot</th>
<th>TCP sport</th>
<th>TCP dport</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>22</td>
<td>drop</td>
<td></td>
</tr>
</tbody>
</table>
OpenFlow Forwarding Abstraction

Protocol Independent
- Construct Ethernet, IPv4, VLAN, MPLS, …
- Construct new forwarding methods

Backward Compatible
- Run in existing networks

Technology Independent
- Switches, routers, WiFi APs
- Cellular basestations
- WDM/TDM circuits
Things to Note about Forwarding

Common OpenFlow model is to use first packets of flows to compute and push state

Flows vs state: SDN allows more general models of forwarding state management independent of traffic
- Events trigger changes, e.g., failures, control traffic
- Managing inconsistencies is critical

Fabrics vs switches: Control logics don’t have to deal with switches
- They essentially program a fabric that looks like one large switch and supports end-to-end connectivity by default
- Complex logic pushed to the edge
Virtual Data Paths

Thinking in terms of fabrics essentially means control logics have to deal with simpler topologies

- Topology captured by “virtual data paths”
  - Depending on control logic, can be very simple: for access controls it is just data path through a single switch

- Every virtual element uses familiar forwarding abstractions, e.g., L2, L3 and ACLs

Control platform responsible for mapping virtual data path to the physical network
### SDN in development

#### Domains
- Data centers
- Public clouds
- Enterprise/campus
- Cellular backhaul
- Enterprise WiFi
- WANs
- Home networks

#### Products
- Switches, routers: About 15 vendors
- Software: 8-10 vendors and startups

New startups. Lots of hiring in networking.
The SDN Stack

Controller
- NOX
- Beacon
- Trema
- Maestro
- ...)

Slicing Software
- FlowVisor
- FlowVisor Console

Applications
- Simple Switch
- CloudNaaS
- Stratos
- ...

Commercial Switches
- HP, NEC, Pronto, Juniper...
- and many more

OpenFlow Switches
- Software Ref. Switch
- NetFPGA
- Broadcom Ref. Switch
- OpenWRT
- PCEngine WiFi AP
- Open vSwitch
OpenFlow Progression

OF v1.0: released end of 2009: “Into the Campus”

OF v1.1: released March 1 2011: “Into the WAN”
- multiple tables: leverage additional tables
- tags and tunnels: MPLS, VLAN, virtual ports
- multipath forwarding: ECMP, groups

- extensible match
- extensible actions
- IPv6
- multiple controllers

OF v1.3: approved May 17 2012
The SDN Stack

Controller

Commercial Switches
- HP, NEC, Pronto, Juniper.. and many more
- Software Ref. Switch
- OpenWRT

OpenFlow Switches
- NetFPGA
- Broadcom Ref. Switch
- PCEngine WiFi AP
- Open vSwitch
<table>
<thead>
<tr>
<th>Vendor</th>
<th>Model(s)</th>
<th>Virtualize?</th>
<th>Notes</th>
<th>Image</th>
</tr>
</thead>
</table>
| HP ProCurve  | 5400zl, 6600, +| 1 OF instance per VLAN | -LACP, VLAN and STP processing before OF  
- Wildcard rules or non-IP pkts processed in s/w  
- Header rewriting in s/w  
- CPU protects mgmt during loop | ![Image](image1.png) |
| Pronto/      | 3290, 3780, 3920, + | 1 OF instance per switch | - No legacy protocols (like VLAN and STP)  
- Most actions processed in hardware  
- MAC header rewriting in h/w | ![Image](image2.png) |

<table>
<thead>
<tr>
<th>Name</th>
<th>Lang</th>
<th>Platform(s)</th>
<th>Original Author</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenFlow Reference</td>
<td>C</td>
<td>Linux</td>
<td>Stanford/Nicira</td>
<td>not designed for extensibility</td>
</tr>
<tr>
<td>Open vSwitch</td>
<td>C/Python</td>
<td>Linux/BSD?</td>
<td>Ben Pfaff/Nicira</td>
<td>In Linux kernel 3.3+</td>
</tr>
<tr>
<td>Indigo</td>
<td>C/Lua</td>
<td>Linux-based Hardware Switches</td>
<td>Dan Talayco/BigSwitch</td>
<td>Bare OpenFlow switch</td>
</tr>
</tbody>
</table>
The SDN Stack

Controller

NOX  Beacon  Trema  Maestro  ...

Commercial Switches

HP, NEC, Pronto, Juniper.. and many more

Software Ref. Switch  NetFPGA  Broadcom Ref. Switch

OpenWRT  PCEngine WiFi AP  Open vSwitch

OpenFlow Switches
Controllers

<table>
<thead>
<tr>
<th>Name</th>
<th>Lang</th>
<th>Original Author</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenFlow Reference</td>
<td>C</td>
<td>Stanford/Nicira</td>
<td>not designed for extensibility</td>
</tr>
<tr>
<td>NOX</td>
<td>Python, C++</td>
<td>Nicira</td>
<td>actively developed</td>
</tr>
<tr>
<td>Beacon</td>
<td>Java</td>
<td>David Erickson (Stanford)</td>
<td>runtime modular, web UI framework, regression test framework</td>
</tr>
<tr>
<td>Maestro</td>
<td>Java</td>
<td>Zheng Cai (Rice)</td>
<td></td>
</tr>
<tr>
<td>Trema</td>
<td>Ruby, C</td>
<td>NEC</td>
<td>includes emulator, regression test framework</td>
</tr>
<tr>
<td>RouteFlow</td>
<td>?</td>
<td>CPqD (Brazil)</td>
<td>virtual IP routing as a service</td>
</tr>
<tr>
<td>POX</td>
<td>Python</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Floodlight</td>
<td>Java</td>
<td>BigSwitch, based on Beacon</td>
<td></td>
</tr>
</tbody>
</table>

Too many to easily keep track of…

http://yuba.stanford.edu/~casado/of-sw.html
The SDN Stack

NOX | Beacon | Trema | Maestro | ... | Controller

FlowVisor Console

FlowVisor

FlowVisor

Commercial Switches

HP, NEC, Pronto, Juniper.. and many more

Software Ref. Switch

NetFPGA

Broadcom Ref. Switch

OpenWRT

PCEngine WiFi AP

Open vSwitch

OpenFlow Switches
How SDN will shape networking

1. Empower network owners and operators
   - Customize networks to local needs
   - Eliminate unneeded features
   - Creation of virtual, isolated networks

2. Increase the pace of innovation
   - Innovation at software speed
   - Standards (if any) will follow software deployment
   - Technology exchange with partners
   - Technology transfer from universities
Summary

Networks becoming

- More programmatic
- Defined by owners and operators, not vendors
- Faster changing, to meet operator needs
- Lower opex, capex and power

Abstractions

- Will shield programmers from complexity
- Make behavior formally verifiable
- “Will take us places we can’t yet imagine”