Handout # 18: 
Software-Defined Networking

Professor Yashar Ganjali
Department of Computer Science
University of Toronto

yganjali@cs.toronto.edu
http://www.cs.toronto.edu/~yganjali
Announcements

• Programming assignment 2
  • Due: Fri. Dec. 2\(^{\text{th}}\) at 5pm
  • Don’t leave it to the last minute …

• Grading for PA2 (18% of your final mark)
  • 6% testing simple router (PA1)
  • 12% for the NAT

• Problem Set 2
  • Due: Nov. 18\(^{\text{th}}\) at 5pm

• This week’s tutorial:
  • Friday, and Tuesday: Programming assignment 2
The Story So Far

- **Layering**
  - **Link layer**
    - Media, framing, error detection/correction, switches, hubs, ...
  - **Network layer**
    - Addressing (CIDR, subnet), routing and forwarding, DNS, BGP, ...
  - **Transport layer**
    - TCP, UDP, flow control, congestion control, queue management, ...
- **Misc**: Queueing Mechanisms, Middleboxes
- **Next**: Software-defined networks
Innovation – Computers vs. Networks

• How difficult is it to create/modify a computer application?
• How difficult is it to create/modify a network feature?
• What is the difference?
• What are the tools available for each?
Innovation in Applications

OS abstracts hardware substrate

→ Innovation in applications
Innovation in OS and Applications

Simple, common, stable, hardware substrate below
+ Programmability
+ Competition
→ Innovation in OS and applications
Innovation in Infrastructure

Simple, common, stable, hardware substrate below
+ Programmability
+ Strong isolation model
+ Competition above
→ Innovation in infrastructure
Vertically integrated
Closed, proprietary
Slow innovation
Small industry

Horizontal
Open interfaces
Rapid innovation
Huge industry
We Have Lost Our Way

Routing, management, mobility management, access control, VPNs, …

- Million of lines of source code
- 6,000 RFCs
- Billions of gates
- Bloated
- Power Hungry
- Vertically integrated, complex, closed, proprietary
- Networking industry with “mainframe” mind-set
• Lack of competition means glacial innovation
• Closed architecture means blurry, closed interfaces
Vertically integrated: Closed, proprietary, slow innovation.

Horizontal: Open interfaces, rapid innovation.

Specialized Features
Specialized Control Plane
Specialized Hardware

Open Interface
Control Plane
Merchant Switching Chips
Control Plane
Control Plane
Control Plane

App App App App App App App App App

CSC 458/CSC 2209 – Computer Networks
University of Toronto – Fall 2016
What we need ...
1) Separate Intelligence from Datapath

Operators, users, 3rd party developers, researchers, …

New function!
2) Cache Decisions

- In minimal flow-based datapath

“If header = $x$, send to port 4”
“If header = $y$, overwrite header with $z$, send to ports 5,6”
“If header = $?$, send to me”
How Can We Do This?

Network OS

Feature

OS
Custom Hardware

Feature

OS
Custom Hardware

Feature

OS
Custom Hardware

Feature

OS
Custom Hardware

Feature

OS
Custom Hardware

Feature

OS
Custom Hardware

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
Consequences

• More innovation in network services
  • Owners, operators, 3rd party developers, researchers can improve the network
  • E.g. energy management, data center management, policy routing, access control, denial of service, mobility

• Lower barrier to entry for competition
  • Healthier market place, new players

• Lower cost
  • Infrastructure
  • Management
Example: Routing

- OSPF
  - RFC 2328: 245 pages
- Distributed System
  - Builds consistent, up-to-date map of the network: 101 pages
- Dijkstra’s Algorithm
  - Operates on map: 4 pages
Example: Routing

OSPF = Dijkstra

IS-IS

Distributed System

Network OS

Packet Forwarding

Packet Forwarding

Packet Forwarding

Custom Hardware

OS

Distributed System

Distributed System
Back to the story ...
Software Defined Network (SDN)

Control Program A

Control Program B

Network OS

Packet Forwarding

Packet Forwarding

Packet Forwarding

Packet Forwarding

Packet Forwarding

Packet Forwarding

Packet Forwarding
Network OS

**Network OS**: distributed system that creates a consistent, up-to-date network view

- Runs on servers (controllers) in the network
- NOX, ONIX, HyperFlow, Kandoo, Floodlight, Trema, Beacon, Maestro, BeeHive, OpenDayLight, ... + more

Uses forwarding abstraction to:

- Get state information **from** forwarding elements
- Give control directives **to** forwarding elements
Software Defined Network (SDN)
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
Software Defined Network (SDN)

Control Program A

Control Program B

Network OS

Packet Forwarding

Packet Forwarding

Packet Forwarding

Packet Forwarding
Forwarding Abstraction

**Purpose**: Abstract away forwarding hardware

- **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
Forwarding Substrate

- Flow-based
- Small number of actions for each flow
  - Plumbing: Forward to port(s)
  - Control: Forward to controller
  - Routing between flow-spaces: Rewrite header
  - Bandwidth isolation: Min/max rate
- External open API to flow-table
What is a flow?

- Application flow
- All http
- Jim’s traffic
- All packets to Canada
- ...

Types of action

- Allow/deny flow
- Route & re-route flow
- Isolate flow
- Make flow private
- Remove flow
Substrate: “Flowspace”

Payload

Ethernet
DA, SA, etc
IP
DA, SA, etc
TCP
DP, SP, etc

Collection of bits to plumb flows (of different granularities) between end points

Header
User-defined flowspace
OpenFlow

• Open standard to run experimental protocols in production networks
  • API between the forwarding elements and the network OS
• Based in Stanford, supported by various companies (Cisco, Juniper, HP, NEC, ...)
• Used by universities to deploy innovative networking technology
Traditional Switch

Ethernet Switch
Traditional Switch

Control Path (Software)

Data Path (Hardware)
OpenFlow Switch

Control Program A

Control Program B

Network OS

OpenFlow Protocol (SSL)

Ethernet Switch
OpenFlow Rules

Control Program A

Control Program B

Network OS

Packet Forwarding

Packet Forwarding

Flow Table(s)

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”
Plumbing Primitives

\(<Match, Action>\)

**Match** arbitrary bits in headers: Match: 1000x01xx0101001x

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

**Action**

- Forward to port(s), drop, send to controller
- Overwrite header with mask, push or pop
- Forward at specific bit-rate
**OpenFlow Rules – Cont’d**

- Exploit the flow table in switches, routers, and chipsets

<table>
<thead>
<tr>
<th>Flow 1.</th>
<th>Rule (exact &amp; wildcard)</th>
<th>Action</th>
<th>Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow 2.</td>
<td>Rule (exact &amp; wildcard)</td>
<td>Action</td>
<td>Statistics</td>
</tr>
<tr>
<td>Flow N.</td>
<td>Rule (exact &amp; wildcard)</td>
<td>Default Action</td>
<td>Statistics</td>
</tr>
</tbody>
</table>
# Flow Table Entry

- **OpenFlow Protocol Version 1.0**

<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 port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline

+ mask what fields to match

<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>
</tr>
</thead>
</table>

[Image of flow table entry with fields and actions]
## 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>*</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>*</td>
<td>*</td>
<td>00:1f:..</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>*</td>
<td>00:2e..</td>
<td>00:1f..</td>
<td>*</td>
<td>0800</td>
<td>vlan1</td>
<td>1.2.3.4</td>
<td>5.6.7.8</td>
<td>4</td>
<td>17264</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>*</td>
<td>*</td>
<td>22</td>
</tr>
</tbody>
</table>

---
### Examples

**Routing**

<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>5.6.7.8</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>port6</td>
</tr>
</tbody>
</table>

**VLAN**

<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>*</td>
<td>*</td>
<td>port6, port7, port9</td>
</tr>
</tbody>
</table>
OpenFlow Hardware

- Juniper MX-series
- NEC IP8800
- WiMax (NEC)
- HP Procurve 5400
- Cisco Catalyst 6k
- PC Engines
- Quanta LB4G

More coming soon...
OpenFlow Usage Example

- Dedicated OpenFlow Network

Diagram showing OpenFlow switches connected to a controller via OpenFlow Protocol. Each switch has sections for Rule, Action, and Statistics. Peter’s code is shown on the controller.
Usage examples

- Peter’s code:
  - Static “VLANs”
  - His own new routing protocol: unicast, multicast, multipath, load-balancing
  - Network access control
  - Home network manager
  - Mobility manager
  - Energy manager
  - Packet processor (in controller)
  - IPvPeter
  - Network measurement and visualization
  - …
Research/Production VLANS

Flow Table

Normal L2/L3 Processing

Controller

Research VLANs

Production VLANs
Virtualize OpenFlow Switch

- Researcher A VLANs
- Researcher B VLANs
- Researcher C VLANs
- Production VLANs

Normal L2/L3 Processing

Controller A
Controller B
Controller C
Virtualizing OpenFlow

A’s Controller

B’s Controller

C’s Controller

OpenFlow Protocol

OpenFlow FlowVisor & Policy Control

OpenFlow Switch

OpenFlow Switch

OpenFlow Switch
Virtualizing OpenFlow

- Broadcast
- Multicast
- http Load-balancer

OpenFlow Protocol

OpenFlow Switch

FlowVisor & Policy Control

OpenFlow Protocol

OpenFlow Switch

OpenFlow Switch

OpenFlow Switch
Food for Thought

- What are the challenges in switching from traditional networks to OpenFlow networks?

- What are the opportunities?