CSC 458/2209 – Computer Networks

Handout # 16:
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
  - To be completed individually.
  - Due: Friday, Nov. 29th at 5pm

- No tutorials this week
Announcements

- **Final Exam**
  - **Time:** Tue. December 10\(^{th}\), 2019; 14:00-16:00
  - **Location:**
    - A-KE: GB304
    - KI-OM: MS2170
    - OU-ZZ: MS3153
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**: Queuing Mechanisms, Middleboxes
- **Next**: Software-defined networking
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?
OS abstracts hardware substrate
\[ \rightarrow \text{Innovation in applications} \]
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

- Vertically integrated, complex, closed, proprietary
- Networking industry with “mainframe” mind-set

Routing, management, mobility management, access control, VPNs, ...

Million of lines of source code
6,000 RFCs

Billions of gates
Bloated
Power Hungry

Feature
Feature

OS
Custom Hardware
Reality is Even Worse

- Lack of competition means glacial innovation
- Closed architecture means blurry, closed interfaces
Vertically integrated
Closed, proprietary
Slow innovation

Horizontal
Open interfaces
Rapid innovation
What we need ...
1) Separate Intelligence from Datapath

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

New function!
2) Cache Decisions

- In minimal flow-based datapath

obody

- 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?
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

Packet Forwarding
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
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 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
Software Defined Network (SDN)
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”

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

“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

\(<\text{Match, Action}>\)

**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>

---

CSC 458/CSC 2209 – Computer Networks  
University of Toronto – Fall 2019
## 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>00:2e..</td>
<td>00:1f..</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>
<td>80</td>
<td>port6</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>Forward</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>vlan1</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 ...
OpenFlow Usage Example

- Dedicated OpenFlow Network

Controller

Peter’s code

PC

OpenFlow Protocol

OpenFlow Switch

Rule Action Statistics

Rule Action Statistics

Rule Action Statistics

Rule Action Statistics

Peter

CSC 458/CSC 2209 – Computer Networks

University of Toronto – Fall 2019
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)
  - IPv6
  - Network measurement and visualization
  - …
Research/Production VLANS

- **Research VLANs**
- **Production VLANs**
- **Flow Table**
- **Normal L2/L3 Processing**

Controller
Virtualize OpenFlow Switch

Normal L2/L3 Processing

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

Controller A
Controller B
Controller C

Flow Table
Virtualizing OpenFlow

OpenFlow Switch

A’s Controller

OpenFlow Protocol

B’s Controller

OpenFlow FlowVisor
& Policy Control

C’s Controller

OpenFlow Protocol

OpenFlow Switch

OpenFlow Switch

OpenFlow Switch
Virtualizing OpenFlow

OpenFlow Switch

Broadcast

OpenFlow Switch

Multicast

http Load-balancer

OpenFlow Protocol

OpenFlow FlowVisor & Policy Control

OpenFlow Protocol

OpenFlow Switch

OpenFlow Switch
Food for Thought

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

• What are the opportunities?