Software Defined Networking
OpenFlow and NOX

ECE/CS598HPN

Radhika Mittal

Acknowledgement for some of the slides:
Yashar Ganjali, Univ. of Toronto
Software Defined Network (SDN)
Abs#1: Forwarding Abstraction

- Express intent independent of implementation
  - Don’t want to deal with proprietary HW and SW

- OpenFlow is a standardized interface to switch.
Software Defined Network (SDN)

Network OS

Open interface to packet forwarding

Packet Forwarding

Packet Forwarding

Packet Forwarding

Packet Forwarding
OpenFlow

**Initial objective:** Enable experimentation and innovation within universities.

- Vendors do not want expose their switch control plane (software interface) for experimentation.
- Another alternative: programmable/flexible switches:
  - do not meet performance requirements (standard PCs)
  - or are too expensive (a research prototype)
  - or have limited port density (NetFPGA)

**What minimal support would vendors be comfortable to provide, in a way that allows control plane experimentation and innovation?**

- Can compromise on generality to meet performance/cost requirements and vendors’ constraints, and provide *some reasonable degree of flexibility*.

**Supported by various companies (Cisco, Juniper, HP, NEC, …)**

**Now being used world-wide in industries.**
Traditional Switch

Ethernet Switch
Control path adds rules to the forwarding tables (flow tables) implemented in the data path.
OpenFlow Switch

- Control Program A
- Control Program B

Network OS

OpenFlow Protocol (SSL)

Provides a standard interface to program flow tables in a switch from an external (centralized) software controller.
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”
Match-Action Primitive

Match arbitrary bits in headers:  

```
1000x01xx0101001x
```

- Match on any of the supported header fields
- Allows any flow granularity

Action

- Forward to port(s)
- Encapsulate and send to controller
- Drop
- Rewrite packet headers, map to a particular priority level
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>Packet + byte counters</td>
<td></td>
<td></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

<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>

+ mask what fields to match
Flow Table Entry

- OpenFlow Protocol Version 1.0

<table>
<thead>
<tr>
<th>Rule</th>
<th>Action</th>
<th>Stats</th>
</tr>
</thead>
</table>

- Packet + byte counters

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

Switch Port  MAC src  MAC dst  Eth type  VLAN ID  VLAN prio  IP Src  IP Dst  IP Prot  IP ToS  TCP sport  TCP dport

+ 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>00:2e..</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>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>*</td>
<td>*</td>
<td></td>
<td></td>
<td>port6</td>
</tr>
</tbody>
</table>

5.6.7.8 * *

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>

* * * * vlan1 * * * * *
# Supported Header Fields

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th># Headers</th>
</tr>
</thead>
<tbody>
<tr>
<td>OF 1.0</td>
<td>Dec 2009</td>
<td>12</td>
</tr>
<tr>
<td>OF 1.1</td>
<td>Feb 2011</td>
<td>15</td>
</tr>
<tr>
<td>OF 1.2</td>
<td>Dec 2011</td>
<td>36</td>
</tr>
<tr>
<td>OF 1.3</td>
<td>Jun 2012</td>
<td>40</td>
</tr>
<tr>
<td>OF 1.4</td>
<td>Oct 2013</td>
<td>41</td>
</tr>
</tbody>
</table>
OpenFlow Switches

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

And more....
OpenFlow Usage Example

- Dedicated OpenFlow Network

![Diagram of OpenFlow network with switches and a controller](image)
OpenFlow Usage Example

- Dedicated OpenFlow Network

Controller

Peter’s code

OpenFlow Protocol

Peter
OpenFlow Usage Example

• Dedicated OpenFlow Network

Controller

Peter’s code

OpenFlow Protocol

Peter
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

- **Research VLANs**
- **Production VLANs**

Flow Table

Normal L2/L3 Processing

Controller
Virtualize OpenFlow Switch

- Normal L2/L3 Processing
- Flow Table
- Flow Table
- Flow Table
- Researcher A VLANs
- Researcher B VLANs
- Researcher C VLANs
- Production VLANs
- Controller A
- Controller B
- Controller C
Virtualizing OpenFlow

OpenFlow Switch

A’s Controller

B’s Controller

C’s Controller

OpenFlow FlowVisor & Policy Control

OpenFlow Protocol

OpenFlow Protocol

OpenFlow Protocol
Virtualizing OpenFlow

OpenFlow Switch

Broadcast

Multicast

http Load-balancer

OpenFlow
FlowVisor & Policy Control

OpenFlow Protocol

OpenFlow Protocol

OpenFlow Switch

OpenFlow Switch

OpenFlow Switch

OpenFlow Switch
Discuss!

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

• What are the opportunities?
Software Defined Network (SDN)

Network OS

Feature

Packet Forwarding

Feature

Open interface to packet forwarding

Packet Forwarding

Packet Forwarding

Packet Forwarding

Packet Forwarding
Design choices for scalability

• Granularity of network view
  • Topology (switches, hosts, middleboxes)
  • Bindings between names and addresses
  • *Exclude network traffic state.*

• Granularity of control
  • Per-packet control will not scale.
  • Prefix-based control too coarse-grained.
  • Use *flow-based* control.
Scalability Argument

- **0 - 10/s**: Modification of Control Program
  - Strong Consistency

- **$10^1$ – $10^3$/s**: Per Network Event
  - Eventual Consistency

- **$10^3$ – $10^6$/s**: Per Flow
  - No Consistency

- **$10^6$ – $10^8$/s**: Per Packet
  - No Consistency
Implication

• Can replicate controllers.

• Each replica can independently handle flow initiations.

• With network change events being less frequent, a consistent network view can be maintained across replicas.
Discuss!

• Do you buy the scalability argument?

• Are there any other concerns?
NOX was just the beginning...

• Support different languages
  • POX: Python
  • OpenDaylight, Floodlight, ONOS, Beacon, Maestro: Java
  • Onix: C++
  • ....

• Improved APIs/flexibility/scalability:
  • Maestro: exploit mutli-core parallelism.
  • Onix: richer state (network information base), that is replicated and distributed across instances.
  • Many many more.....