Software Defined Networking
OpenFlow and NOX

ECE/CS598HPN

Radhika Mittal

Acknowledgements: 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)

Feature

Network OS

Open interface to packet forwarding

Packet Forwarding

Packet Forwarding

Packet Forwarding

Packet Forwarding

Packet Forwarding
OpenFlow

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

• Developed at Stanford.

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

• Now being used world-wide in industries.
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

Flow Table(s)

Packet Forwarding

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”
Match-Action Primitive

**Match** arbitrary bits in headers: Match: 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

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
# Flow Table Entry

• OpenFlow Protocol Version 1.0

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>VLAN ID</th>
<th>VLAN prio</th>
<th>IP Src</th>
<th>IP Dst</th>
<th>IP Prot</th>
<th>IP ToS</th>
<th>TCP sport</th>
<th>TCP dport</th>
</tr>
</thead>
</table>

+ mask what fields to match

- Rule
- Action
- Stats

- Packet + byte counters

1. Forward packet to port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline
## 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>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>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</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>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.6.7.8</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### VLAN

<table>
<thead>
<tr>
<th>Switch</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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>port6,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>port7,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>port9</td>
</tr>
</tbody>
</table>
## 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

More coming soon...
OpenFlow Usage Example

- Dedicated OpenFlow Network

![Diagram of OpenFlow network with Controller, PCs, and OpenFlow switches]
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

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

Flow Table

Normal L2/L3 Processing

Controller A
Controller B
Controller C
Virtualizing OpenFlow

OpenFlow Switch

A’s Controller

OpenFlow Protocol

B’s Controller

OpenFlow FlowVisor & Policy Control

C’s Controller

OpenFlow Switch

OpenFlow Switch

OpenFlow Switch
Virtualizing OpenFlow

- OpenFlow Switch
- OpenFlow Switch
- OpenFlow Switch
- Broadcast
- Multicast
- http Load-balancer
- OpenFlow Protocol
- OpenFlow FlowVisor & Policy Control
Discuss!

• What are the challenges in switching from traditional networks to OpenFlow networks?
  • Performance
  • Security or DoS
  • Dealing with very large network, scalability

• What are the opportunities?
  • Test network without disrupting production
  • Functionality within switches, middleboxes (caching… )
OpenFlow -- your opinions

• Pros:
  • concrete, clear workflow, comprehensive examples, achievable
  • flexible packet format (somewhat)
  • use existing switch mechanisms -- flow tables
  • Not overly ambitious – first focus on campus networks
OpenFlow -- your opinions

Cons:
• Reliability of performance (?)
• Security (?)
• Performance (?)
  • Latency to the controller
• Size of flow table
• Incentive for vendors
• Impact on production traffic
• More details on controller
• Sharing resources across multiple OpenFlow users
• How to support multiple controller instances?
OpenFlow -- your opinions

Ideas:
• QoS for production and experimental traffic
• ML + controller for network resource regulation (?)
• Make OpenFlow more flexible and expressive
• Refactoring middlebox functionality using OpenFlow
• Evaluate scalability
• Use OpenFlow to handle link failures
• Can it really be deployed at large scale?
Software Defined Network (SDN)
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 multi-core parallelism.
  • Onix: richer state (network information base), that is replicated and distributed across instances.
  • Many many more.....
NOX -- your opinions

• Pros:
  • “flow” granularity – trade-off flexibility and scalability
  • OS-like abstraction -- multiple applications
  • Functional prototype
  • Good motivation, examples
NOX -- your opinions

Cons:

• Controller energy consumption
• No experimental results
• What are the pitfalls?
• How well can it scale?
  • Costly to maintain network view?
• Performance issues?
• Security issues?
• How to handle packet losses?
NOX -- your opinions

Ideas:
• What level of consistency is required for network state?
• More functionality
• Evaluation performance and scalability
• What if network topology changes very rapidly?
• More powerful distributed algorithm?