SDN Overview for UCAR IT meeting
19-March-2014

Presenter Steven Wallace
(ssw@iu.edu)

Support by the GENI Program Office!
Patterns (here, there, everywhere)
Patterns (here, there, everywhere)
Today’s Internet

A very few Patterns

Conceptually configured per device

Devices exchange information (e.g., routing info) that, along with their static configuration, determines their behavior
Software Defined Networking (SDN)

Network devices dynamically configured by a central “controller”

Example of “southbound” SDN protocols include: OpenFlow, OVSDB, Puppet, NetConf, etc.
How Might SDN be useful...

Multi-tenant data center (e.g., Amazon Web Services)

On-demand provisioning (aka *orchestration*) of services, including network as a service

Can be implemented as an overlay (e.g., VXLAN)
Q: How to implement load balancing
A: schedule a meeting, or fill out web form

AWS Load balancer, bringing the idea of SDN home!
What is OpenFlow?

- It's a protocol for control the forwarding behavior of Ethernet switches in a **Software Defined Network**
- Initially released by the **Clean Slate Program** at Stanford, its specification is now maintained by the **Open Networking Forum**
- Most of today's material is based on the **OpenFlow 1.0** specification
- In April 2012, **OpenFlow 1.3** was approved (see also 4/2012 ONF **white paper**
### Ethernet Switch

- **Features**: 
  - CLI, SNMP, TFTP

- **Value Add**: 
  - Embedded Operating System

- **Data Plane**: 
  - Table-based (e.g., TCAM/CAM) high-speed forwarding engine

- **Control Plane**: 
  - Embedded Operating System
OpenFlow Controller
Table-based (e.g., TCAM/CAM) high-speed forwarding engine
Embedded Operating System implements OpenFlow

Features
Value Add

Control Plane

OpenFlow Protocol

Embedded Operating System implements OpenFlow

Data Plane

Table-based (e.g., TCAM/CAM) high-speed forwarding engine
OpenFlow Controller

Features

OpenFlow Protocol
Each switch connects directly with OF Controller

Value Add
# Flow Table

<table>
<thead>
<tr>
<th>Header Fields</th>
<th>Counters</th>
<th>Actions</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ingress Port</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ethernet Source Addr</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ethernet Dest Addr</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ethernet Type</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VLAN id</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VLAN Priority</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IP Source Addr</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IP Dest Addr</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IP Protocol</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IP ToS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICMP type</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICMP code</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Per Flow Counters**
- Received Packets
- Received Bytes
- Duration seconds
- Duration nanoseconds

**Forward**
- (All, Controller, Local, Table, IN_port, Port# Normal, Flood)

**Enqueue**
- Drop

**Modify-Field**
<table>
<thead>
<tr>
<th>Header Fields</th>
<th>Counters</th>
<th>Actions</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>If ingress port == 2</td>
<td></td>
<td>Drop packet</td>
<td>32768</td>
</tr>
<tr>
<td>if IP_addr == 129.79.1.1</td>
<td></td>
<td>re-write to 10.0.1.1, forward port 3</td>
<td>32768</td>
</tr>
<tr>
<td>if Eth Addr == 00:45:23</td>
<td></td>
<td>add VLAN id 110, forward port 2</td>
<td>32768</td>
</tr>
<tr>
<td>if ingress port == 4</td>
<td></td>
<td>forward port 5, 6</td>
<td>32768</td>
</tr>
<tr>
<td>if Eth Type == ARP</td>
<td></td>
<td>forward CONTROLLER</td>
<td>32768</td>
</tr>
<tr>
<td>If ingress port == 2 &amp;&amp; Eth Type == ARP</td>
<td></td>
<td>forward NORMAL</td>
<td>40000</td>
</tr>
</tbody>
</table>
Special Ports

Controller (sends packet to the controller)

Normal (sends packet to non-openflow function of switch)

Local (can be used for in-band controller connection)

Flood (flood the packet using normal pipeline)
### Flow Table

<table>
<thead>
<tr>
<th>Header Fields</th>
<th>Counters</th>
<th>Actions</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>If ingress port == 2</td>
<td></td>
<td>Drop packet</td>
<td>32768</td>
</tr>
<tr>
<td>if IP_addr == 129.79.1.1</td>
<td></td>
<td>re-write to 10.0.1.1, forward port 3</td>
<td>32768</td>
</tr>
</tbody>
</table>

Each Flow Table entry has two timers:

- **idle_timeout**
  - seconds of no matching packets
  - after which the flow is removed
  - zero means never timeout

- **hard_timeout**
  - seconds after which the flow is removed
  - zero means never timeout

If both **idle_timeout** and **hard_timeout** are set, then the flow is removed when the first of the two expires.
Populating the Flow Table

Proactive

Rules are relatively static, controller places rules in switch before they are required.

Reactive

Rules are dynamic. Packets which have no match are sent to the controller (packet in). Controller creates appropriate rule and sends packet back to switch (packet out) for processing.
Controller and Switch Communication

- Mode - Controller vs. Listener
  - TCP Communication, who initiates conversation
- Mode and Populating Flow Table independent
Example application: topology discovery

OpenFlow Controller

Diagram showing a network topology with a central OpenFlow controller connected to multiple devices.
Bootstrapping a new switch

Switch requires minimal initial configuration (e.g., IP address, default GW, and OpenFlow controller)

Switch connects to controller. Controller requests things like a list of ports, etc.

Controller proceeds to determine the switch's location.
Bootstrapping a new switch

Controller *proactively* places a rule in the switch.

If $\text{ether\_type} = \text{LLDP}$, $\text{actions}=$output:controller

Then the controller creates an LLDP packet, sends it to the switch, and instructs the switch to send it out a port (repeat for all ports).

Since all switches in the controller's network have a rule to send LLDP packets to the controller, the controller is able to determine the topology.
What is OpenFlow leaving on the table?

Remember those patterns, here, there, everywhere

“Pure” OF requires that you idle the silicon built to implement those patterns

OpenFlow TTPs (table typing patterns) may solve this
Internet2 and OF

OESS is OF-based and configures services, and it’s opensource

Soon to support “Flowvisor” function, allowing researchers to control “slice” of productoin network

What is a slice?