Software Defined Networking (SDN)
OpenFlow and OpenStack

Vivek Dasgupta
Principal Software Maintenance Engineer
Red Hat
• Introduction – SDN and components
• SDN Architecture, Components
• SDN Controller - OpenDayLight
• OpenFlow architecture
• Open vSwitch
• OpenStack Neutron
• OpenStack SDN
• Future Trends
Introduction – SDN and related technologies

- SDN is a technology enabling **programmable networks**
- Using software running on general purpose OS/Hardware
- SDN – Separation of control and data plane
The need for SDN

- Traditional Router / Switch
VTEP Edge Appears Directly Connected

Evolving Hardware Opens New Opportunities

OpenFlow v1.3
OVSDB

OpenStack

Daylight Cluster
Evolution to SDN

Router/Switch

Applications Layer
APP
APP
APP
APP

Control Plane
Data Plane

Control Layer

Infrastructure Layer / Data plane

Router/Switch

APP
APP
APP
APP

APP
APP
APP
APP

Control Plane
Data Plane

Router/Switch
SDN Architecture

- **SDN Applications**
- **SDN Control Plane**
  - Controller
  - NOS
- **SDN Data Plane**
  - Devices
  - OpenFlow
SDN Components - (Ecosystem)

- Cloud Orchestration
- Network Virtualization
- Network Functions Virtualization (NFV)
SDN Controller
OpenDayLight

- Open Source
- Southbound - OF
- Northbound -> Neutron
- Plugin -> Red Hat Enterprise Linux OpenStack Platform

Base Network Service Functions
- Topology Manager
- Stats Manager
- Switch Manager
- Host Tracker
- Shortest Path Forwarding
- DOVE Manager
- Affinity Services
- Traffic Redirect
- LLSP Services
- VTN Manager

Service Abstraction Layer (SAL)
(Plugin Manager, Capability Abstraction, Flow Programming, Inventory etc.)

Controller Platform
- Management GUI
- NTN Coordinator
- OpenStack Neutron
- DCOS Protection
- Network Application Orchestration & Services

Southbound Interfaces & Protocol Plugins
- OpenFlow 1.0, 1.3
- OVSDB
- NETCONF
- LISP
- BCP
- PCEP
- SNMP

Dataplane Elements
- OpenFlow Enabled Devices
- Open vSwitches
- Additional Virtual & Physical Devices
- Dataplane Elements (Virtual Switches, Physical Device Interfaces)
OpenFlow / Open vSwitch
OpenFlow introduction

- **Openflow** - standard for interacting with forwarding behaviours of switches
- Control the behaviour of switches dynamically and programmatically

- **Flow tables**, Group tables and OpenFlow Channel (**TCP port 6633**)
OpenFlow protocol - Messages

• **Controller to Switch** :: Switch / Flow table config, *Packet out*, Barrier, Role Req, Bundle [Controller to Switch messages]

• **Asynchronous** :: *Packet-in*, Flow-removed, Port-status, Controller Role status, Table status, Request forward [Async messages]

• **Symmetric messages** :: Hello, Echo Req/Reply, Error, Experimenter [Symmetric messages]
OpenFlow – FlowTables and Routing

• Packet-in message for table miss
• Controller sends a packet-out mesg specifying action
• Buffer id -> packet
• Flow modification
OpenFlow – FlowTables and Routing

• Flow Table Components
  
  **Match fields**, Priority, Counters, Instructions, Timeouts, Cookie

• Flow table
  
  match vs miss

• Flow removal

<table>
<thead>
<tr>
<th>Match</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Src. Port= 10</td>
<td>Fwd to Port 20</td>
</tr>
<tr>
<td>Src. Port= 20</td>
<td>Fwd to Port 10</td>
</tr>
</tbody>
</table>
OpenFlow – FlowTables

- Group table
- Meter table
- Counters
- Instructions
- Actions

**OpenFlow-enabled Network Device**
FlowTable compared to an instruction set

<table>
<thead>
<tr>
<th>MAC Src</th>
<th>MAC dst</th>
<th>IP Src</th>
<th>IP dst</th>
<th>TCP dport</th>
<th>....</th>
<th>Action</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10.20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Port 1</td>
<td>250</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5.6.7.8</td>
<td></td>
<td></td>
<td>Port 2</td>
<td>300</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>25</td>
<td></td>
<td>drop</td>
<td>892</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>local</td>
<td>120</td>
</tr>
<tr>
<td></td>
<td></td>
<td>192.</td>
<td></td>
<td></td>
<td></td>
<td>controller</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
• OpenFlow Packet Capture
Open vSwitch

• Networking in Software
• In 2012 – total # of virtual ports surpassed physical ports
• A opensource software switch
• High performance forwarding using Linux Kernel Module
• OpenFlow Compliant
• Advanced switching features
Open vSwitch

- Kernel Datapath
- Userspace daemon
- Configuration database
- Since RHEL 6.4

Datapath

Management

ovs-ofctl

ovsdb-tool

ovs-vsctl

Open vSwitch Diagram:

- vSwitchd
- ovsdb
- ovs-ofctl
- ovs-vsctl
- Datapath
- User Space
- Kernel
- Netlink
- UpCall
- Reinject
- From NetDevice
- To NetDevice
- Flow Table
Open vSwitch operations

• Normal mode vs Flow mode
• Flow Table Match based on L2/L3/L4
• Forward, Drop Modify headers
• Systemtap Probe

0 swapper(0): -> ovs_netdev_frame_hook
7 swapper(0): -> ovs_vport_receive
11 swapper(0): -> ovs_dp_process_received_packet
15 swapper(0): -> ovs_flow_extract
20 swapper(0): <- ovs_dp_process_received_packet
24 swapper(0): -> ovs_flow_tbl_lookup
28 swapper(0): -> ovs_flow_hash
32 swapper(0): <- ovs_flow_tbl_lookup
35 swapper(0): -> find_bucket
38 swapper(0): <- ovs_flow_tbl_lookup
41 swapper(0): <- ovs_dp_process_received_packet
45 swapper(0): -> ovs_flow_used
48 swapper(0): <- ovs_dp_process_received_packet
52 swapper(0): -> ovs_execute_actions
56 swapper(0): -> do_execute_actions

0 ovs-vswitchd(1302): -> ovs_vport_cmd_get
4 ovs-vswitchd(1302): -> lookup_vport
8 ovs-vswitchd(1302): -> ovs_vport_locate
12 ovs-vswitchd(1302): -> hash_bucket
15 ovs-vswitchd(1302): <- ovs_vport_locate
19 ovs-vswitchd(1302): -> ovs_netdev_get_name
22 ovs-vswitchd(1302): <- ovs_vport_locate
25 ovs-vswitchd(1302): <- lookup_vport
28 ovs-vswitchd(1302): <- ovs_vport_cmd_get

OpenStack (SDN)
Nova Networking

- Early days of Openstack networking
- Flat, Flat-DHCP, VLAN
- No router, firewalls etc
Neutron

- Neutron is at the northbound side of the SDN framework
- Neutron provides network services to the Nova compute
Neutron plugin architecture

- Neutron services
- Various plugins connect to controllers or OpenFlow Switches

**Neutron API**

**API Extensions**

**Neutron Service**
- L2 Network Abstraction
- Device and Service framework
- Does NOT do any actual Implementation of abstraction

**Neutron Plug-in API**

**Vendor User Plug-in**
- Maps Abstraction to Implementation on Physical Network
- Makes all decisions about how a network is implemented
- Can provide additional features through API extension
OpenStack Network Topology
OpenStack SDN

• Various Components
• Iptables
• Open vSwitch
• Overlay networks
• Tunnels – GRE/VXLAN
• Network Namespace
• Netfilter NAT for Floating IP addresses
OpenStack Floating IP

Red Hat Enterprise Linux
OpenStack Platform
OpenStack SDN

- A, B, C :: Tap, Fw Bridge, Iptables
- D, E :: VLAN tagging
- F, G :: Tunnels
- Open vSwitch, GRE
- O, P :: DHCP
- M, N :: Router
- Netfilter NAT
Future Trends

• Need to stick to open standards
• Need to have a stable SDN ecosystem
• Standardization for various components
• Allows for various vendor solutions (open/closed source)
• Scope for Innovation at each layer - Apps, Controller, Protocols, Devices (Physical/Virtual)
• Possible scope for Hardware acceleration products in SDN space
• Evolution of NOS
Thank You ! Questions ?

<vdasgupt@redhat.com>

Please complete this session survey available on the mobile app :: Complete 8 surveys – Win Exciting Prizes !!

Slides in PDF will be available later at www.redhat.com/summit/2014/presentations