SDN for Cloud Networks

Building the Foundation for “Networking as a Service”
Outline

- Software Defined Networking and Cloud
- Openflow
- Virtual Switching
- SDN Controllers
Software Defined Networking and Cloud
Networking as a Service

- Compute and Storage as a Service
  - What AWS, Azure, and Google Compute Engine provide
  - Pay only for as much compute and storage as you use

- What Networking as a Service is
  - Only connect to my VMs
    - Looks like a LAN
  - Only bring up network when VMs need it
    - Dynamic
  - Only pay for bandwidth when I use it
  - Isolate my traffic from other customers

- Goal: Offer full Infrastructure as a Service
  - EnterpriseLAN but entirely in the cloud

Source: http://www.golime.co
Recall:

- Need for isolation
  - Broadcast/multicast
  - Different tenants/IP addresses from same address space
  - Between different tenant VMs on the same server
- Multiple tenants/dynamic creation and destruction of tenant networks
  - Tenant network resource usage is highly dynamic
- VM movement between different physical servers
  - DC operator manages VMs by “hot swapping” them between servers

Solution is Virtual Networks!
Key Properties of Virtual Networks

- **Partitioning:**
  - Each physical resource can be used concurrently by multiple virtual network instances
    - Also true of statistical multiplexing

- **Isolation:**
  - The clear isolation of any virtual network from all others

- **Abstraction:**
  - Control and management of a virtual resource need not require complexity of directly manipulating underlying physical resources

- **Aggregation:**
  - Multiple instances of physical resource grouped to obtain increased capacity

Source: https://mountainss.files.wordpress.com/2012/07/virtual-networks.png
Wide Area Network Characteristics

- Huge physical area
  - 5,000 km from SFO to NYC
  - 9,000 km from SFO to FRA
  - 15-30 ms speed of light delay

- Sparse and minimal bandwidth between routers/switches
  - 100 Mbps corporate VPN connections are typical

- Bandwidth is expensive
  - 1G fiber connection from San Jose to Montreal - $3K per month!

- Simple network and edge controlled by enterprise/retail customers
  - But complex operator edge is trending

- Slow resource allocation/fast reaction to failure
  - Slow: routing protocol convergence times
  - Fast: MPLS protection and failover time

Source: http://www.wikipedia.com
But in Data Centers:

- Small physical area
  - Order of thousand m2
  - Minimal speed of light delay
- Lots of bandwidth
  - 1G leaf/10G backbone links common
  - 10G leaf/40G backbone links trending
- Bandwidth is cheap
  - Commodity switches inexpensive

- Complex edge/simple core
  - Virtual switch/virtual edge bridge (VEB) on server
  - Simple L2/L3 fabric out to Internet router

- Fast dynamic, centralized control of compute and storage resources
  - Tenant resource allocation takes place through dedicated orchestration software
    - Like an operating system on a single server

Source: Bilal, et. al.
In Addition:

- Natural centralization of control in data center architecture through the Network Virtualization Authority (NVA)
- Software trumps hardware for fast implementation and deployment (VEPA example)
Traditional Computer Networks: Data Plane

Data plane:
Packet streaming
Fast but Dumb

Forward, filter, buffer, mark, rate-limit, and measure packets
Traditional Computer Networks: Control Plane

Control plane:
Distributed algorithms
Slow but Smart

Track topology changes, compute routes, install forwarding rules
Traditional Computer Networks: Management Plane

Management plane:
   Human time scale

   Slow but Really Smart

Collect measurements and configure the equipment
Software Defined Networking (SDN)

- Logically-centralized controller
- Slow but Smart
- API to the data plane (e.g., OpenFlow)
- Fast but Dumb
- Switches
SDN Design Principles

- Logically centralized but physically distributed control/management plane
  - Merge control and management plane
  - Controller replicas, distributed DB protocols for scale-out and reliability
- Northbound API exposes useful abstractions for network services programmers
  - Hide details of:
    - Individual vendor equipment
    - Control protocols that do similar jobs
    - ... 
- Southbound protocols allow full control of forwarding and network configuration
  - Example forwarding control: OpenFlow (up next)
  - Example network management: OF-Config (up next)
- Switches and routers implement forwarding and network configuration as programmed from the controller

SDN is a Key Technological Approach to Implementing Virtual Networks in Cloud
Generalized SDN Architecture

SDN Application
SDN App Logic
NBI Driver

SDN Application
SDN App Logic
NBI Driver

SDN Application
SDN App Logic
NBI Driver

Control + Management Plane Application Plane

SDN Controller
NBI Agent
SDN Control Logic
SBI Driver

SDN Northbound Interfaces (NBIs)

Data Plane

Network Element
SDN Datapath
SBI Agent
Forwarding Engine / Processing Function

Network Element
SDN Datapath
SBI Agent
Forwarding Engine / Processing Function

Source: Open Network Foundation

+ indicates one or more instances  |  * indicates zero or more instances
Openflow
OpenFlow

Control Path (Software)

Data Path (Hardware)
OpenFlow

OpenFlow Controller

OpenFlow Protocol (SSL/TCP)

Control Path

OpenFlow

Data Path (Hardware)
OpenFlow Switching

OpenFlow Client

<table>
<thead>
<tr>
<th>MAC src</th>
<th>MAC dst</th>
<th>IP Src</th>
<th>IP Dst</th>
<th>TCP sport</th>
<th>TCP dport</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>5.6.7.8</td>
<td>*</td>
<td>*</td>
<td>port 1</td>
</tr>
</tbody>
</table>

Controller

PC

OpenFlow Flow Table

OpenFlow Flow Table 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>

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>

<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>5.6.7. 8</td>
<td></td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></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>

<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>drop</td>
</tr>
</tbody>
</table>

OpenFlow/SDN tutorial, Srini Seetharaman, Deutsche Telekom, Silicon Valley Innovation Center
Development of OpenFlow Specification

OF 1.0: Single flow table, Ethernet circa 1994, IP circa 1994

OF 1.1: Multiple flow tables, Group tables, MPLS (but broken)

OF 1.2: Extensible match, Queue extension, IPv6

OF 1.3: IPv6 hdr, MPLS fixed, PBB tagging, Tunnel control

OF 1.4: Optical ports, Bundled messages

OF 1.5: Meter Table, Multiple Controllers

OF v0.9 developed at Stanford in CleanSlate Program

Standards

OpenFlow 1.5 Architecture

OpenFlow 1.5 Switch Model

Ingress processing:
- Packet In
- Set Ingress Port
- Flow Table 0
- Flow Table 1
- ...
- Flow Table n
- Execute Action Set
- Group Table

Egress processing:
- Set Output Port
- Action Set = {output}
- Flow Table e
- Flow Table e+1
- ...
- Flow Table e+m
- Execute Action Set
- Packet Out
- Output Port

Multiple Table Example: Port Based VLAN Tagging

- Most NICs don’t handle VLAN tags
  - Packets are sent untagged
- VLAN tags are inserted at the first hop switch
  - Based on the source port
- Implementing with one table results in a combinatorial explosion
  - With one table: Nport x NMAC
  - With two tables: Nport + NMAC

### Rules

<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>

**Table 0:**
7  *  *  *  *  0  *  *  *  *  *  *
Push VLAN ID 42;
Send to Table 1

**Table 1:**
*  *  *  *  *  42  *  *  *  *  *
Packet out Port 42
Group Table Detail

Group Table: A table supporting entries with *action buckets*, which can be executed all or in part depending on the group type

**Group Table Entry:**

- **Group identifier:** 32 bit integer identifying the group on the OpenFlow switch
- **Group type:** a type identifier defining the group semantics
- **Counters:** for tracking group statistics
- **Action Buckets:** an ordered list of action buckets, where each bucket contains a set of actions and associated parameters that are always executed as an action set.
Action Bucket Semantics

- **Indirect**
  - Execute the one bucket in this group
  - Only a single bucket
  - Allows multiple flow entries to point to a single group
  - Example use: IP next hop forwarding

- **All**
  - Execute all buckets in the group
  - Example use: broadcast or multicast

- **Select**
  - Execute one bucket in the group
  - Bucket selected through a switch-specific algorithm
  - Example use: Equal Cost MultiPath (ECMP) forwarding

- **Fast Failover**
  - Execute the first live bucket in the group
  - Allows switch to change forwarding without contacting the controller
  - Switch must implement a liveness mechanism
  - Example use: fast failover
Meter Table Details

Meter Table: a collection of per flow meters used for rate limiting that can be combined with Queues to implement QoS strategies

Meter Table Entry:
- Meter identifier: 32 bit unsigned integer identifying this meter
- Meter bands: unordered list of bands, where each band specifies a rate and a processing method
- Counters: keep track of number of packets processed by this meter

<table>
<thead>
<tr>
<th>Meter Identifier</th>
<th>Meter Bands</th>
<th>Counters</th>
</tr>
</thead>
</table>

Band Entry:
- Band type: defines how the packet is processed
- Rate: lowest rate at which the band can apply
- Counters: keep track of the number of packets processed by this band
- Type specific arguments: optional arguments depending on type

<table>
<thead>
<tr>
<th>Band Type</th>
<th>Rate</th>
<th>Counters</th>
<th>Type specific arguments</th>
</tr>
</thead>
</table>

Network Management for OpenFlow

- OF-Config is the standard Network Management protocol for OpenFlow
  - Others are also in widespread use

- Job of OF-Config
  - Configures various switch “hard” state
  - Query switch for capabilities

- Extension of IETF Netconf protocol
  - OF-Config schema in XML

- Examples of hard state configuration:
  - Virtual ports
    - Tunnels
    - Aggregated links
  - Queues
  - Certificates
OF-Config Architecture

OF-Config Tunnel UML Model

OpenFlow Control Plane for VxLAN Data Center Virtualization

OF-Switch Sets Up Flow Processing for VNI-1

OpenFlow Controller/OF-Config Management Point

ToR1

ToR2

ToR3

VNI-1

VM1
VM2
VM3
VM4

Hypervisor

OF Virtual Switch
Server Hardware

NIC1
NIC2

VNI-1

VM

MAC src
MAC dst
Eth type
VLAN ID
IP Src
IP Dst
IP Prot
TCP sport
TCP dport

Action

VNI-1 vPort out

Flow Table

Port

vPort

Port

Port

Port

MAC src
MAC dst
Eth type
VLAN ID
IP Src
IP Dst
IP Prot
TCP sport
TCP dport

vPort out
OpenFlow and Hardware: Pica8

- Example Hardware:
  - P-3922 (top) and P-3930 (bottom)

- Ports
  - 64 x 10GbE
  - 4 x 40GbE uplink

- Either fiber (P-3922) or copper (P-3930)

- Nonblocking switch fabric with low latency
  - 1.2 Tb throughput
  - 960 Mpps

- Suitable for top of rack switch

- Conventional networking:
  - Maximum MAC addresses: 128K
  - Maximum VLANs: 4094
  - Maximum routes: 12,000

- OpenFlow networking:
  - Trident+ chip: maximum 8,000

Limited Flow Table Size is the Problem with OpenFlow and Conventional Hardware
Virtual Switching
What is a Virtual Switch?

- System software that emulates a physical switch
  - Conceptually sits in the hypervisor
  - Downlink: Emulates NICs toward VMs (vNICs)
  - Uplink: Connects to hardware NICs through hypervisor
- Provides virtualized networking support to VMs
  - Bridges traffic between VMs on the physical server
  - Switches off-server traffic from the ToR to the VMs
Example Virtual Switch: Open Virtual Switch (OVS)

- **History**
  - 2007 - First released by Nicira
  - 2010 - OpenFlow 1.0 support
  - 2011 - Ported to hardware (Pica8)
  - 2012 - VMWare acquires Nicira, commits to maintaining OVS
  - 2012 - OVS becomes part of Linux release
  - 2014 - OpenFlow 1.5 support

- **Open source, maintained by VMWare**
  - Included with Xen, KVM open source hypervisors

- **Proprietary competitors**
  - Cisco 1000V (number one)
  - VMWare vDS (but trending downward)
  - IBM DVS 5000V
  - Microsoft Hyper-V vswitch
Supported Functionality

- IPv4 and IPv6, full 802.1, 802.3 Ethernet
- Standard 802.1Q VLANs with trunking
- Spanning Tree Protocol (STP)
- Multiple IP layer tunneling protocols
  - GRE, VxLAN, and more
- OpenFlow 1.x, where x depends on OVS version
  - But not OF-Config
    - Older OVS-DB protocol used for configuration
- Many, many other features that hardware switches support
Software Architecture

User space
- Configuration
- Control Plane

Kernel space
- Data path (data plane)
- Included in Linux kernel release since 3.3

Source: http://www.slideshare.net/janghoonsim/virtualized-network-with-openv-switch?related=1
How OVS Looks to a VM

Source: http://www.slideshare.net/janghoonsim/virtualized-network-with-openv-switch?related=1
Performance v.s. Linux Bridge

![Graph showing performance comparison between OVS and Linux Bridge](http://www.slideshare.net/janghoonsim/virtualized-network-with-openv-switch?related=1)
SDN Controllers
What Should I Look for in an SDN Controller?

- What applications/services come with the controller?
- What protocols does it support on the southbound (network equipment side) interface?
- What kinds of abstractions does the northbound API support?
- Is there support for scale-out, failover, replication, reliability, high availability, in-service upgrade?
- Does the controller support a Web GUI, command line interface (CLI), and/or API for controller configuration?
- Does the controller support peering with other controllers?

Example Controller: OpenDaylight
OpenDaylight: History

- April 2013 – Formed as an open source project under the Linux Foundation
  - Cisco was the driver
  - Other supporters were Ericsson, IBM, HP, Juniper, Arista Networks. …
- December 2013 – Hydrogen release
  - Three “editions”
    - Basic
    - Data Center
    - Service Provider
- October 2014 – Helium release
Other Features Not Shown

- Apache Karaf used for OSGI support
  - In service upgrade
  - Allows installation of a new module without shutting down the controller
  - Essential feature for telcom software

- GUI and CLI both provided
  - CLI is through Karaf

- Infinispan key-value store used for storing state
  - ACID (Atomicity, Consistency, Isolation, Durability) “hard” transaction model
  - But doesn’t support persistence

- No explicit support for scale-out, failover, replication, reliability, high availability
Service Abstraction Layer (SAL)

- SAL is the major difference between OpenDaylight and other controllers
- **Objective:**
  - Write an abstract layer that can be used by various services northbound and various protocols southbound
  - Allow different southbound protocols and northbound applications to reuse components
  - Eliminates the problem of duplicated code for different protocols
- **Basic approach: model driven development**
  - Design model using UML
  - Write YANG code to implement the model
    - YANG: Yet Another Next Generation
    - Data modeling language for NetConf
  - Compile YANG code to Java
- **Controversial**
  - Not fully realized in the Hydrogen release
    - Some services including cloud networking were released outside the SAL
  - More of a research project than production ready
Summary

- SDN lays the foundation for Networking as a Service (NaaS)
  - SDN technology is a natural match for deploying virtual networks in a cloud
- OpenFlow is an example of a protocol that supports the deployment of a Software Defined Network
  - But it is not the only such protocol
    - Example: XMPP used by Juniper in its Contrail controller
- Virtual switching locates complex virtual network configuration on the end host
  - Where the computational resources support it
- SDN controllers support northbound interface to cloud management/orchestration software simplifying virtual network provisioning
Acknowledgements

- Xenofontas Dimitropoulos, ETH Zurich
- Jennifer Rexford, Princeton