Multicasting on SDN

Prof. Sunyoung Han
Konkuk University
syhan@cclab.konkuk.ac.kr
23 July 2015
Contents

1. Software Defined Networking (SDN)
2. OpenFlow
3. Multicasting
4. Open vSwitch
5. OpenFlow Protocol
6. SDN Multicasting Testbed
7. Conclusion
1. Software Defined Networking (SDN)
SDN Definition

“SDN is a **refactoring** of the relationship between **network devices** and the **software** that **controls** them”

- **Directly programmable**
  - Network control is directly programmable
    - Because it is decoupled from forwarding functions
  - Dynamic and automatically configure, manage, secure, and optimize network resources via SDN programs

- **Centrally managed**
  - Network intelligence is (logically) centralized in software-based SDN controllers
  - SDN controllers maintain a global view of the network

- **Open standards-based and vendor-neutral**
  - SDN simplifies network design and operation
  - Instructions are provided by SDN controllers instead of multiple vendor-specific devices and protocols.
Autonomous Network vs SDN

Distributed Autonomous Network

Software Defined Network

Source: ONF Seminar (2012.3)
SDN Architecture

- **Application Layer**
  - Applications and services

- **Control Layer**
  - Control entities in infrastructure layer
  - Isolated from data plane
  - Control & manage the entire network

- **Infrastructure Layer**
  - Hardware components for forwarding packets
  - L2/L3 switching for data transmission
  - Implemented by switches and routers
Software-Defined Network with key Abstractions in the Control Plane

Well-defined API

Routing
Traffic Engineering
Other Applications

Network Virtualization
Network Map Abstraction

Separation of Data and Control Plane

Forwarding
Forwarding
Forwarding
Forwarding
Forwarding
Forwarding
Open Networking Foundation: Dedicated to SDN

• A user-driven organization dedicated to the promotion and the adoption of SDN

• Developing open standards
  – Such as the OpenFlow Standard, OpenFlow Configuration and Management Protocol Standard

• The OpenFlow Standard is the first vendor-neutral standard for the communication between the control and forwarding layer
2. OpenFlow
What is OpenFlow?

- “OpenFlow is based on an Ethernet switch, with an internal flow-table, and a standardized interface to add and remove flow entries.”
Basic idea of OpenFlow

• Separate Control function and Forwarding engine
  – OpenFlow Controller
  – OpenFlow Switch

• Provide standard interface to control forwarding engine
OpenFlow Switching

OpenFlow Switch specification

OpenFlow Switch

Secure Channel

Flow Table

Controller

PC

OpenFlow Protocol

SSL

sw

hw

The Stanford Clean Slate Program
http://cleanslate.stanford.edu
OpenFlow Switching

**OpenFlow Switch**

**OpenFlow Table**

<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>

- Port 1: 5.6.7.8
- Port 2: 1.2.3.4

Controller

Hardware Layer

Software Layer

3. Multicasting
Review major types of casting

- Anycasting
- Multicasting
- Broadcasting
- Unicasting
Multicasting

- Group communication.
- One to Many distribution.
- Message from one will be addressed to a group of destination nodes simultaneously.
Multicast – Efficient Data Distribution

Unicast approach

Multicast approach
Is Multicast Useful?

- Better bandwidth utilization.
- Less host/router processing (reduces server and network load)
- Multicast is beneficial to many applications
  - Internet Live Broadcasting
  - Network Game
  - File Distribution for large number of Users
**IP multicast**

- Implements multicast service at IP routing level.
- Individual packet sent from the source, duplicated at the router and delivered to multiple destination simultaneously.
- Also called **native multicast**.
IP Multicast Addresses

• Class D IP addresses
  – 224.0.0.0 – 239.255.255.255

| 1 1 1 0 | Group ID |

• How to allocate these addresses?
  – Well-known addresses: IANA
  – Transient addresses: e.g., by “SDR” program
    • Assigned and reclaimed dynamically,
Failure of IP Multicast

• Real world:
  – Not widely deployed even after 15 years!
  – Use carefully – e.g., on LAN or campus, rarely over WAN
  – Largest deployment: MBONE, using IP-tunnels to connect domains

• IP Multicast failings
  – Scalability of routing protocols
    • Extra router state required
  – Hard to manage
    • Who gets to set up groups and when?
  – Hard to implement TCP equivalent
    • As we just saw with SRM
  – Chicken-egg: No real applications
    • Hard to get applications to use IP Multicast without existing wide deployment
  – Economics, policy: Hard to get inter-domain support
    • Who pays for packet duplication?
Alternative Solution: Overlay Multicast

- Focus connectivity from end point to end point in logical network.
- Host will participate multicast.
Overlay Multicast

- **Unicast will be implement among pairs of hosts for data dissemination.**
  - Between source and destination network, unicast techniques will be used.

- **The hosts in overlay multicast exclusively handle group management, routing, and tree construction, without any support from Internet routers.**

- **Also known as:**
  - Application Layer Multicast (ALM)
  - End System Multicast (ESM)
Emerging Solution: Multicasting in SDN?

- SDN is a complete new solution
  - Redefines the idea of networking from scratch
  - A fully open system – easy and free to implement
  - Programmability enables your own multicast distribution policy
  - Centralized approach enables central management of multicast group members
SDN-based Multicast

- OVSs establish unicast tunnels between OVS and OVS
- OVS performs multicast for local network
- Topology built based on multicast tree
SDN-based Multicast Advantages

• Easy to manage SDN network
  – Network status detection and load balance
• Efficient routing mechanism
• Less overhead with no header encapsulation

• SDN-based multicast approach is implemented by Open vSwitch
4. Open vSwitch
Open vSwitch

- Software-based virtual switch
- Assign VMs with elastic and secure network configurations
- Flexible Controller in User-Space
- Fast Datapath in Kernel

Basic Design of Open vSwitch

http://www.slideshare.net/teyenliu/the-basic-introduction-of-open-vswitch
Features of Open vSwitch (1/2)

• Visibility into inter-VM communication
  – via NetFlow, sFlow, etc.
  – **NetFlow** is a protocol for exporting aggregated IP flow totals.
  – **sFlow** is a general purpose network traffic measurement system technology.

• Standard 802.1Q VLAN model

• BFD and 802.1ag link monitoring
  – Bidirectional Forwarding Detection (BFD) is a network protocol used to detect faults between two forwarding engines connected by a link.

• **STP** – Spanning Tree Protocol
  – STP prevents loops from being formed when switches or bridges are interconnected via multiple paths.

• Fine-grained QoS control
Features of Open vSwitch (1/2)

• OpenFlow protocol support
  – Including many extensions for virtualization
• IPv6 support
• Multiple tunneling protocols
  – GRE, VXLAN, IPsec, GRE and VXLAN over IPsec
• Remote configuration protocol with C and Python
The Main Components

Controller

ovsdb-server

ovs-vswitchd

OVS Kernel Module

Management Protocol (6632/TCP)
OpenFlow (6633/TCP)
Netlink
Component: ovsdb-server

- Database that holds switch level configuration
- Speaks OVSDB management protocol (JSON-RPC) to manager and ovs-vswitchd
- Based on JSON-RPC 1.0, which is an agreed-upon format for using JSON as a remote procedure call markup language.
Component: ovs-vswitchd

- Core component in the system:
  - Communicates with outside world using OpenFlow
  - Communicates with ovsdbserver using management protocol (OVSDB)
  - Communicates with kernel module over netlink
  - Communicates with the system through netdev abstract interface
- Supports multiple independent datapaths (bridges)
- Implements mirroring, bonding, and VLANs through modifications of the same flow table exposed through OpenFlow.
Component: openvswitch_mod.ko

- OVS Kernel Module
- Handles switching and tunneling
- Exact match flow tables
- Fast and simple
  - Packet comes in, associated actions executed and counters updated.
5. OpenFlow Protocol
OpenFlow Table Entry

<table>
<thead>
<tr>
<th>Rule</th>
<th>Action</th>
<th>Stats</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch Port</td>
<td>MAC src</td>
<td>MAC dst</td>
</tr>
<tr>
<td></td>
<td>Eth type</td>
<td>VLAN ID</td>
</tr>
<tr>
<td></td>
<td>IP Src</td>
<td>IP Dst</td>
</tr>
<tr>
<td></td>
<td>IP Prot</td>
<td>TCP sport</td>
</tr>
<tr>
<td></td>
<td>TCP dport</td>
<td></td>
</tr>
</tbody>
</table>

+ mask

Packet + byte counters

1. Forward packet to port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline
5. ….
### Examples of Flow Table (1/2)

<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; Eth Type == ARP</td>
<td></td>
<td>forward NORMAL</td>
<td>40000</td>
</tr>
</tbody>
</table>
Examples of Flow Table (2/2)

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>

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>5.6.7.8</td>
<td>*</td>
<td>*</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>22</td>
<td>drop</td>
</tr>
</tbody>
</table>
OpenFlow Usage

» Alice’s code:
  > Simple learning switch
  > Per Flow switching
  > Network access control/firewall
  > Static “VLANs”
  > Her own new routing protocol: unicast, multicast, multipath
  > Home network manager
  > Packet processor (in controller)
  > IPvAlice
OpenFlow Protocol

- Controller with Switches
- Maintains flow tables in OFS
  - Create
  - Delete
  - Modify
- Notify unknown packet to OFC
- Gather information and statistics
OpenFlow Specification

- Current OpenFlow specification is v1.4
- Components
  - Secure Channel
  - Controller
  - Switch
    - Flow table
Components

• Switch
  – Forwarding received packets according to flow tables
  – If unmatched packets are received, send `packet-in` message to controller

• Controller
  – Handles packet-in event from switches
  – Communicates via TCP port 6633
  – Create flow table
  – Modify flow table
  – Delete flow table
## Modify fields (1/2)

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set VLAN ID</td>
<td>Insert VLAN header with specified VID, priority zero if no VLAN header is present. If already exists VLAN header, replace the VID.</td>
</tr>
<tr>
<td>Set VLAN Priority</td>
<td>Insert VLAN header with specified priority, VID of zero if no VLAN header is present. If already exists VLAN header, replace the priority.</td>
</tr>
<tr>
<td>Strip VLAN header</td>
<td>Strip VLAN header if present.</td>
</tr>
<tr>
<td>Modify Ether source address</td>
<td>Replace the Ether source address with specified value.</td>
</tr>
<tr>
<td>Modify Ether destination address</td>
<td>Replace the Ether destination address with specified value.</td>
</tr>
</tbody>
</table>
## Modify fields (2/2)

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modify IPv4 source address</td>
<td>Replace the IPv4 source address with specified value, and update checksum (IP, TCP, UDP)</td>
</tr>
<tr>
<td>Modify IPv4 destination address</td>
<td>Replace the IPv4 destination address with specified value, and update checksum (IP, TCP, UDP)</td>
</tr>
<tr>
<td>Modify IPv4 ToS bit</td>
<td>Replace the existing IPv4 ToS bit with new value.</td>
</tr>
<tr>
<td>Modify transport source port</td>
<td>Replace the existing TCP/UDP source port with specified value, and update checksum</td>
</tr>
<tr>
<td>Modify transport destination port</td>
<td>Replace the existing TCP/UDP destination port with specified value, and update checksum</td>
</tr>
</tbody>
</table>
Packet Flow Processing

(a) Packets are matched against multiple tables in the pipeline

① Find highest-priority matching flow entry

② Apply instructions:
   i. Modify packet & update match fields (apply actions instruction)
   ii. Update action set (clear actions and/or write actions instructions)
   iii. Update metadata

③ Send match data and action set to next table

(b) Per-table packet processing
## Flow Table (1.4)

<table>
<thead>
<tr>
<th>Match Fields</th>
<th>Priority</th>
<th>Counters</th>
<th>Instructions</th>
<th>Timeouts</th>
<th>Cookie</th>
</tr>
</thead>
</table>

- **Match fields**: to match against packets.
  - Ingress port, packet headers and optionally metadata specified by a previous table.
- **Priority**: matching precedence of the flow entry.
- **Counters**: updated when packets are matched.
- **Instructions**: to modify the action set or pipeline processing.
- **Timeouts**: flow expiration time by the switch.
- **Cookie**: opaque data value chosen by the controller.
  - May be used by the controller to filter flow statistics, flow modification and flow deletion.
  - Not used when processing packets.
Matching

Packet In
Start at table 0

- Match in table n?
  - Yes
    - Update counters
      - Execute instructions:
        - update action set
        - update packet/match set fields
        - update metadata
  - No
- Table-miss flow entry exists?
  - Yes
  - No
- Drop packet
- Goto-Table n?
  - Yes
  - Execute action set
  - No
Priority

• Packets are matched against flow entries based on prioritization
  – 16 bits
  – Each entry has priority

• **Higher numbers have higher priorities**
  – 65535: highest
  – 0: lowest

• Exact match entry has always high priority
OpenFlow Message Header

- Common header

- Version
- Type
- Length

- Transmission ID

- Transaction ID to identify request and reply pair

- Specify type of following message

- Length of total OpenFlow message
# OpenFlow Message Types

<table>
<thead>
<tr>
<th>name</th>
<th>value</th>
<th>name</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HELLO</td>
<td>0</td>
<td>FLOW_REMOVED</td>
<td>11</td>
</tr>
<tr>
<td>ERROR</td>
<td>1</td>
<td>PORT_STATUS</td>
<td>12</td>
</tr>
<tr>
<td>ECHO_REQUEST</td>
<td>2</td>
<td>PACKET_OUT</td>
<td>13</td>
</tr>
<tr>
<td>ECHO_REPLY</td>
<td>3</td>
<td>FLOW_MOD</td>
<td>14</td>
</tr>
<tr>
<td>VENDOR</td>
<td>4</td>
<td>PORT_MOD</td>
<td>15</td>
</tr>
<tr>
<td>FEATURES_REQUEST</td>
<td>5</td>
<td>STATS_REQUEST</td>
<td>16</td>
</tr>
<tr>
<td>FEATURES_REPLY</td>
<td>6</td>
<td>STATS_REPLY</td>
<td>17</td>
</tr>
<tr>
<td>GET_CONFIG_REQUEST</td>
<td>7</td>
<td>BARRIER_REQUEST</td>
<td>18</td>
</tr>
<tr>
<td>GET_CONFIG_REPLY</td>
<td>8</td>
<td>BARRIER_REPLY</td>
<td>19</td>
</tr>
<tr>
<td>SET_CONFIG</td>
<td>9</td>
<td>QUEUE_GET_CONFIG_REQUEST</td>
<td>20</td>
</tr>
<tr>
<td>PACKET_IN</td>
<td>10</td>
<td>QUEUE_GET_CONFIG_REPLY</td>
<td>21</td>
</tr>
</tbody>
</table>
OpenFlow Data Structures (Wildcard Field)
# Flow Match

<table>
<thead>
<tr>
<th>name</th>
<th>description</th>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wildcards</td>
<td>Specify wildcard</td>
<td>nw_tos</td>
<td>IPv4 ToS</td>
</tr>
<tr>
<td>in_port</td>
<td>Input port</td>
<td>nw_proto</td>
<td>IPv4 protocol / ARP opcode</td>
</tr>
<tr>
<td>dl_src</td>
<td>Ethernet source address</td>
<td>nw_src</td>
<td>IPv4 source address</td>
</tr>
<tr>
<td>dl_dst</td>
<td>Ethernet destination address</td>
<td>nw_dst</td>
<td>IPv4 destination address</td>
</tr>
<tr>
<td>dl_vlan</td>
<td>VLAN ID</td>
<td>tp_src</td>
<td>TCP / UDP source port</td>
</tr>
<tr>
<td>dl_vlan_pcp</td>
<td>VLAN Priority</td>
<td>tp_dst</td>
<td>TCP / UDP destination port</td>
</tr>
<tr>
<td>dl_type</td>
<td>Ethernet type</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Data Structures for Flow Match

- 13 actions are defined

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFPAT_OUTPUT</td>
<td>0</td>
<td>Output to switch port</td>
</tr>
<tr>
<td>OFPAT_SET_VLAN_VID</td>
<td>1</td>
<td>Set VLAN ID</td>
</tr>
<tr>
<td>OFPAT_SET_VLAN_PCP</td>
<td>2</td>
<td>Set VLAN priority</td>
</tr>
<tr>
<td>OFPAT_STRIP_VLAN</td>
<td>3</td>
<td>Strip VLAN header</td>
</tr>
<tr>
<td>OFPAT_SET_DL_SRC</td>
<td>4</td>
<td>Set Ethernet source address</td>
</tr>
<tr>
<td>OFPAT_SET_DL_DST</td>
<td>5</td>
<td>Set Ethernet destination address</td>
</tr>
<tr>
<td>OFPAT_SET_NW_SRC</td>
<td>6</td>
<td>Set IPv4 source address</td>
</tr>
<tr>
<td>OFPAT_SET_NW_DST</td>
<td>7</td>
<td>Set IPv4 destination address</td>
</tr>
<tr>
<td>OFPAT_SET_NW_TOS</td>
<td>8</td>
<td>Set IPv4 ToS field</td>
</tr>
<tr>
<td>OFPAT_SET_TP_SRC</td>
<td>9</td>
<td>Set TCP / UDP source port</td>
</tr>
<tr>
<td>OFPAT_SET_TP_DST</td>
<td>10</td>
<td>Set TCP / UDP destination port</td>
</tr>
<tr>
<td>OFPAT_ENQUEUE</td>
<td>11</td>
<td>Output to queue</td>
</tr>
<tr>
<td>OFPAT_VENDOR</td>
<td>0xffffffff</td>
<td>Vendor defined action</td>
</tr>
</tbody>
</table>
6. SDN Multicasting Testbed
Objective

• Build SDN multicasting testbed based on national R&E networks
  – Korea (KOREN), China (CERNET), Thailand (ThaiREN)
  – Through TEIN/APAN
  – SDN/OpenFlow/Open vSwitch (OVS)

• QoS/QoE measurement and analysis of multimedia data transmission on APAN/TEIN
Participants

- Prof. Sunyoung Han
  - Konkuk University, Seoul, Korea
  - syhan@konkuk.ac.kr

- Prof. Dongsu Han
  - KAIST, Deajeon, Korea
  - Dongsu.han@gmail.com

- Prof. Junfeng Wang
  - Sichuan University, Chengdu, China
  - wangjf@scu.edu.cn

- Prof. Sinchai (Hatayai), Prof. Wasimon (Phuket)
  - Prince of Songkla University, Thailand
  - ksinchai@gmail.com, wasimon@coe.phuket.psu.ac.th
SDN Multicasting Testbed

1. OVS Controller: Network management
SDN Multicasting Testbed on National Research Networks
SDN Multicasting Testbed

- Sichuan University & Konkuk University

Sichuan University

Konkuk University
SDN Multicasting Testbed

- Sichuan University & Konkuk University

Sichuan University

Konkuk University
Test with PSU Phuket
Test Results

Remote OVS
Remote media server

Local OVS
Local media client 1
Local media client 2
Local media client 3
QoS/QoE Measurement Testbed

- Server Side
  - Media Server
  - Switch
  - OVS

- Network
  - Network State Control server
  - Network information
  - QoS/QoE measured data
  - Information Collection Server

- Client Side
  - Terminal
  - QoE Information Collection Tool
  - Host1
  - Host2
  - Switch
Test Results

- Konkuk Unvi. ↔ KAIST (KOREN)
  - Bandwidth: more than 90Mbps
  - Delay: around 3ms

- Korea ↔ Thailand (KOREN ↔ APAN ↔ ThaiREN)
  - Bandwidth: more than 20Mbps
  - Delay: around 160ms

Korea ↔ China (KOREN ↔ APAN ↔ CERNET)
- Bandwidth: around 49Mbps in the morning and 620kbps in the afternoon
- Delay: around 117ms
Workshop at Thailand 2014

1. Workshop
   - Annual Workshop
   - Venue
     Bangkok, Phuket...
   - Participants
     PSU, SCU, Konkuk U.
     South Asian Countries
   - Sponsor
     Project
     TEIN-CC Program
     Feb. 2015 or later.
     Venue: Bangkok/Phuket
     Theme: SDN, ICN, or FI issues, us$10,000
Conclusion
Conclusion

- OpenFlow is more simple and efficient to manage multicast:
  - Centralized approach:
    - Complete view of the network, thus optimal tree creation
  - Easy multicast group management
  - Immediate re-routing in case of sender failure
  - Easy to implement new customized solutions
References

• Overlay Networks, Reading: 9.4, COS 461: Computer Networks, Jennifer Rexford
• http://en.wikipedia.org/wiki/Overlay_network
• http://web.cse.ohio-state.edu/hpcs/WWW/HTML/internet-p2p.html
• Open vSwitch: http://openvswitch.org/
Thank You!