An Introduction to Open vSwitch

LinuxCon Japan, Yokohama

Simon Horman <simon@horms.net>
Horms Solutions Ltd., Tokyo

2nd June 2011

Contents

Open vSwitch

- Multi-Layer Virtual Switch
- Flexible Controller in User-Space
- Fast Datapath in Kernel
- An implementation of Open Flow

Open vSwitch Availability

- Available from openvswitch.org
- Development code is available in git
- Announce, discussion and development mailing lists
- User-space (controller and tools) is under the Apache license
- Kernel (datapath) is under the GPLv2
- Shared headers are dual-licensed

Open vSwitch Concepts

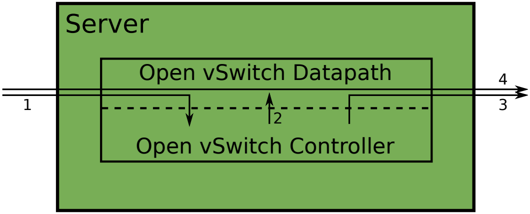
- A switch contains ports
- A port may have one or more interfaces
  - Bonding allows more than once interface per port
- Packets are forward by flow

Identifying Flows

- A flow may be identified by any combination of
  - Tunnel ID
  - IPv6 ND target
  - IPv4 or IPv6 source address
  - IPv4 or IPv6 destination address
  - Input port
  - Ethernet frame type
  - VLAN ID (802.1Q)
  - TCP/UDP source port
  - TCP/UDP destination port
  - Ethernet dource address
  - Ethernet destination address
  - IP Protocol or lower 8 bits of ARP ppcode
  - IP ToS (DSCP field)
  - ARP/ND source hardware address
  - ARP/ND destination hardware address

Forwarding Flows

1. The first packet of a flow is sent to the controller
2. The controller programs the datapath's actions for a flow
   - Usually one, but may be a list
   - Actions include:
     - Forward to a port or ports, mirror
     - Encapsulate and forward to controller
     - Drop
3. And returns the packet to the datapath
4. Subsequent packets are handled directly by the datapath



Open vSwitch Management

- Open vSwitch controller is configured via a JSON database
- Database and thus configuration is persistent across reboots
- Database actions won't return until the controller is reconfigured
- JSON database may be controlled locally using a UNIX socket or remotely using TLS (SSL)

Basic Configuration

1. Ensure that Open vSwitch is running

   `/etc/init.d/openvswitch-switch start`

2. Create a bridge

   `ovs-vsctl -- --may-exist add-br br0`

3. Add port to a bridge

   `ovs-vsctl -- --may-exist add-port br0 eth0`

Basic De-Configuration

1. Ensure that Open vSwitch is running

   `/etc/init.d/openvswitch-switch start`

2. Remove a port from a bridge

   `ovs-vsctl -- --if-exists del-port br0 eth0`

3. Remove a bridge

   `ovs-vsctl -- --if-exists del-br br0`

Examples of Advanced Configuration

- Port Mirroring (SPAN)
- QoS

Port Mirroring (SPAN)

- Allows frames sent to or recieved on one or more ports to be duplicated on a different port
- Useful for debugging

Port Mirroring Configuration (Preparation)

1. Create a dummy interface that will recieve mirrored packets

```
modprobe dummy
ip link set up dummy0
modprobe dummy
```

2. Add the dummy interface to the bridge in use

```
ovs-vsctl add-port br0 dummy0
```

Port Mirroring Configuration (Target)

1. Create a mirror

```
ovs-vsctl \
    -- --id=@m create mirror name=mirror0 \
    -- add bridge br0 mirrors @m
```

2. Find the UUID of the target interface

```
ovs-vsctl list port dummy0
_uuid               : 4d5ed382-a0c3-4453-ab3c-58e1e7f603b0
...
```

3. Configure the mirror to output mirrored packets to the target interface

```
ovs-vsctl set mirror mirror0 \
    output_port=4d5ed382-a0c3-4453-ab3c-58e1e7f603b0
```

Port Mirroring Configuration (Selected Source)

- All packets sent to or received from tap0 will be mirrored on dummy0
- All flooded packets will go to dummy0

1. Find the UUID of the port or ports whose packets should be be mirrored

```
ovs-vsctl list port tap0
_uuid                : d624f5b1-f5e3-4f85-a907-bd209b5463aa
...
```

2. Mirror packets sent to and received from the interface of interest

```
ovs-vsctl set mirror mirror0 \
    select\_dst\_port=d624f5b1-f5e3-4f85-a907-bd209b5463aa
ovs-vsctl set mirror mirror0 \
    select\_src\_port=d624f5b1-f5e3-4f85-a907-bd209b5463aa
```

Port Mirroring Configuration (All Sources)

- All switch packets will go to dummy0

```
ovs-vsctl set mirror mirror0 select_all=1
```

QoS

Open vSwitch QoS capabilities

- Interface rate limiting
- Port QoS policy

QoS: Interface rate limiting

- A rate and burst can be assigned to an Interface
- Conceptually similar to Xen's netback credit scheduler
- Utilises the Kernel tc framework's ingress polycing
- Simple
- Configuration example. 100Mbit/s rate with 10Mbit/s burt:

```
# ovs-vsctl set Interface tap0 ingress_policing_rate=100000
# ovs-vsctl set Interface tap0 ingress_policing_burst=10000
```

QoS: Control: No interface rate limiting

```
# netperf -4 -t UDP_STREAM -H 172.17.50.253 -- -m 8972
UDP UNIDIRECTIONAL SEND TEST from 0.0.0.0 (0.0.0.0)...
Socket  Message  Elapsed      Messages
Size    Size     Time         Okay Errors   Throughput
bytes   bytes    secs            #      #   10^6bits/sec

120832    8972   10.01      146797      0     1052.60
109568           10.01      146620            1051.33
```

- tap networking used
- jumbo frames required to reach line speed ($\approx$210Mbits/s with 1500 byte frames)
- virtio does much better

QoS: Interface rate limiting result

```
# netperf -4 -t UDP_STREAM -H 172.17.50.253
UDP UNIDIRECTIONAL SEND TEST from 0.0.0.0 (0.0.0.0)...
Socket  Message  Elapsed      Messages
Size    Size     Time         Okay Errors   Throughput
bytes   bytes    secs            #      #   10^6bits/sec

120832    8972   10.01      149735      0    1073.66
109568           10.01       14684            105.29
```

- Difference in sent and received packets indicates that excess packets are dropped – no backpressure
- This is an inherent problem when using ingress policying

QoS: Port QoS policy

- A port may be assigned one ore more QoS policy
- Each QoS policy consists of a class and a qdisc
- Classes and qdisc use the Linux kernel's tc implementation
- Only HTB and HFSC classes are supported at this time
- The class of a flow is chosen by the controller
- The QoS policy (i.e. class) of a flow is chosen by the controller
- Operates as an egress filter

QoS: Port QoS policy example

Programming the Datapath

```
# ovs-vsctl set port eth1 qos=@newqos \
 -- --id=@newqos create qos type=linux-htb \
      other-config:max-rate=200000000 queues=0=@q0,1=@q1 \
 -- --id=@q0 create queue \
      other-config:min-rate=100000000 \
      other-config:max-rate=100000000 \
 -- --id=@q1 create queue \
      other-config:min-rate=50000000 \
      other-config:max-rate=50000000
```

Line numbers added for clarity

QoS: Port QoS policy example

Hard-coding the controller

```
# ovs-ofctl add-flow br0 "in_port=2 ip nw_dst=172.17.50.253 \
     idle_timeout=0 actions=enqueue:1:0"
# ovs-ofctl add-flow br0 "in_port=3 ip nw_dst=172.17.50.253 \
     idle_timeout=0 actions=enqueue:1:1"
```

Only suitable for testing

QoS: Port QoS policy example

```
Guest 0:
# netperf -4 -t TCP_STREAM -H 172.17.50.253 -l 30 -- -m 8972
TCP STREAM TEST from 0.0.0.0 (0.0.0.0)...
Recv   Send    Send
Socket Socket  Message  Elapsed
Size   Size    Size     Time     Throughput
bytes  bytes   bytes    secs.    10^6bits/sec
87380  16384   8972    30.01      99.12

Guest 1:
# netperf -4 -t TCP_STREAM -H 172.17.50.253 -l 30 -- -m 8972
...
87380  16384   8972    30.14      49.56
```

QoS: Port QoS policy controller improvements

- Add a default queue to the Port table
- Add enqueue to the FLOOD and NORMAL ports

Questions

Bonus Topic: VLAN Extensions

- Per-Customer VLANs are desirable for security reasons
- But there is a limit of 4094 VLANs

More VLANs

Two, apparently competing, approaches

- IETF / Cisco
  - RFC5517 — Private VLANs
- IEEE
  - 802.1ad — Provider Bridges (Q-in-Q)
  - 802.1ah — Provider Backbone Brides (MAC-in-MAC)

RFC5517 — Private VLANs

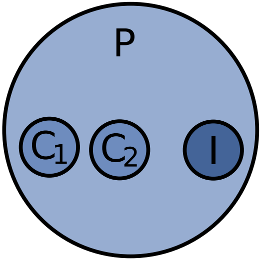
- Uses existing 802.1Q framing
  - Simple to implement (in software/firmware)
- Makes use of pairs of VIDs
  - Requires all switches to support of Private VLANs otherwise switch tables may not merge
- Provides L2 broadcast isolation
  - Forwarding may occur at L3
  - Requires the router to perform proxy ARP
- Currently not supported by Open vSwitch

RFC5517 — Private VLANs

Three VLAN classifications

- Promiscuous
  - May communicate with endpoints on any port
  - e.g.: Gateway, Management Host
- Community
  - May only communicate with endpoints on promiscuous ports or ports belonging to the same comunity
  - e.g.: Different hosts belonging to the same customer
- Isolated
  - May only communicate with endpoints on promiscuous ports
  - e.g.: Hosts that only require access to the gateway

Private VLANs — Domain View



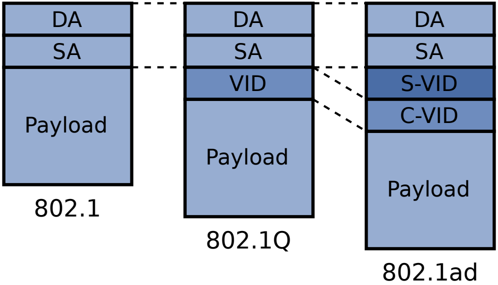
- Promiscous domain (P)
  - May communicate with endpoints in the same domain and sub-domains
- Two community sub-domains (C_1, C_2)
  - May communicate with endpoints in the same domain and parent-domain
- Isolated sub-domain (I)
  - May communicate with endpoints in the parent domain
  - May *not* communicate with endpoints in the same domain

802.1ad — Provider Bridges (Q-in-Q)

- Current standard is 802.1ad-2005, Approved December 2005
- Builds on 802.1Q
- New Framing
  - C-VID (inner)
    - Renamed 802.1Q VID
    - There may be more than one C-VID (inner-inner, ...)
  - S-VID (outer)
    - Different ether-type to C-VID
    - May be translated
- Currently not supported by Linux Kernel / Open vSwitch

802.1ad Framing — Provider Bridges

DA	Destination MAC address
SA	Source MAC addresss
S-VID	Service VLAN ID
C-VID	Customer VLAN ID
VID	VLAN ID

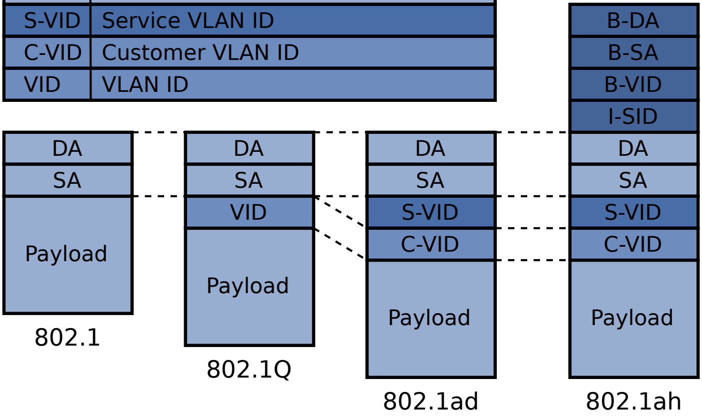


802.1ah — Provider Backbone Bridges (MAC-in-MAC)

- Current standard is 802.1ah-2008, Approved August 2008
- Builds on 802.1ad
- New Framing
  - MAC encapsulation provides full Client VLAN isolation
    - Inner MAC is unknown outside of its scope
  - I-SID: Up to $2^{24} \approx 16$million backbone services
  - I-VID semantics are the same as the S-VLAN
    - Only edge switches need to be Provider Backbone Bridge aware
    - Core switches need only be Provider Bridge (802.1ad) aware
- Currently not supported by Linux Kernel / Open vSwitch

802.1ah Framing — Provider Backbone Bridges

B-DA	Backbone Destination MAC address
B-SA	Backbone Source MAC address
B-VID	Backbone VLAN ID
I-SID	Service ID
DA	Destination MAC address
SA	Source MAC addresss
S-VID	Service VLAN ID
C-VID	Customer VLAN ID
VID	VLAN ID



Questions