OpenStack: OVS Deep Dive

Justin Pettit

Eric Lopez

07 November 2013
Overview

- Visibility (NetFlow, IPFIX, sFlow, SPAN/RSPAN)
- Fine-grained ACLs and QoS policies
- Centralized control through OpenFlow and OVSDB
- Port bonding, LACP, tunneling
- Works on FreeBSD and Linux-based hypervisors
  - Xen, XenServer, KVM, VirtualBox
- Open source, commercial-friendly Apache 2 license
- Multiple ports to physical switches
Visibility

- **Number of subscribers to mailing lists:**
  - discuss: 1371
  - announce: 716
  - dev: 651
  - git: 158

- **OpenStack Summit User Survey showed 48% of deployments use Open vSwitch for their networking solution**
(Partial) List of Contributors
Main Components

- **OVS Kernel Module**
- **ovsdb**-server
- **ovs-vswitchd**

Diagram:
- **User Kernel**
- **Off-box**
- **Management Protocol (6632/TCP)**
- **OpenFlow (6633/TCP)**
- **Netlink**

Diagram shows the components and their connections:
- Control Cluster
- ovsdb-server
- ovs-vswitchd
- OVS Kernel Module
CONFIGURATION DATABASE
ovsdb-server

- Database that holds switch-level configuration
  - Bridge, interface, tunnel definitions
  - OVSDB and OpenFlow controller addresses
- Configuration is stored on disk and survives a reboot
- Custom database with nice properties:
  - Value constraints
  - Weak references
  - Garbage collection
- Log-based (fantastic for debugging!)
- Speaks OVSDB protocol to manager and ovs-vswitchd
- The OVSDB protocol is in the process of becoming an Informational RFC

Tools: ovs-vsctl, ovsdb-tool, ovsdb-client, ovs-appctl
“Open_vSwitch” is the root table and there is always only a single row. The tables here are the ones most commonly used; a full entity-relationship diagram is available in the ovs-vswitchd.conf.db man page.
Debugging the Database

- **ovs-vsctl**: Configures ovs-vswitchd, but really a high-level interface for database
  - ovs-vsctl add-br <bridge>
  - ovs-vsctl list-br
  - ovs-vsctl add-port <bridge> <port>
  - ovs-vsctl list-ports <bridge>
  - ovs-vsctl get-manager <bridge>
  - ovs-vsctl get-controller <bridge>
  - ovs-vsctl list <table>

- **ovsdb-tool**: Command-line tool for managing database file
  - ovsdb-tool show-log [-mmm] <file>
ovsdb-tool show-log

Record number  Time of Change  Caller's comment

root@vm-vswitch:~# ovsdb-tool show-log -m
...
record 3: 2011-04-13 16:03:52 "ovs-vsctl: /usr/bin/ovs-vsctl --timeout=20 -- --with-iface --if-exists del-port eth0 -- --may-exist add-br xenbr0 -- --may-exist add-port xenbr0 eth0 -- set Bridge xenbr0 "other-config:hwaddr="00:c:29:ab:f1:e9"" -- set Bridge xenbr0 fail_mode=standalone -- remove Bridge xenbr0 other_config disable-in-band -- br-set-external-id xenbr0 xs-network-uuids 9ae8bc91-cfb8-b873-1947-b9c4098e4f4b"

table Port insert row "xenbr0":
table Port insert row "eth0":
table Interface insert row "eth0":
table Interface insert row "xenbr0":
table Open_vSwitch row a1863ada:
table Bridge insert row "xenbr0":
...
FORWARDING PATH
Linux Bridge Design

- Simple forwarding
- Matches destination MAC address and forwards
- Packet never leaves kernel
Open vSwitch Design

- Decision about how to process packet made in userspace
- First packet of new flow goes to ovs-vswitchd, following packets hit cached entry in kernel
ovs-vswitchd

- **Core component in the system:**
  - Communicates with outside world using OpenFlow
  - Communicates with ovsdb-server using OVSDB protocol
  - Communicates with kernel module over netlink
  - Communicates with the system through netdev abstract interface

- **Supports multiple independent datapaths (bridges)**

- **Packet classifier supports efficient flow lookup with wildcards and “explodes” these (possibly) wildcard rules for fast processing by the datapath**

- **Implements mirroring, bonding, and VLANs through modifications of the same flow table exposed through OpenFlow**

- **Checks datapath flow counters to handle flow expiration and stats requests**

Tools: ovs-ofctl, ovs-appctl
OVS Kernel Module

- Kernel module that handles switching and tunneling
- Fast cache of non-overlapping flows
- Designed to be fast and simple
  - Packet comes in, if found, associated actions executed and counters updated. Otherwise, sent to userspace
  - Does no flow expiration
  - Knows nothing of OpenFlow
- Implements tunnels

Tools: ovs-dpctl
Userspace Processing

- Packet received from kernel
- Given to the classifier to look for matching flows accumulates actions
- If “normal” action included, accumulates actions from “normal” processing, such as L2 forwarding and bonding
- Actions accumulated from configured modules, such as mirroring
- Prior to 1.11, an exact match flow is generated with the accumulated actions and pushed down to the kernel module (along with the packet)
Kernel Processing

- Packet arrives and header fields extracted
- Header fields are hashed and used as an index into a set of large hash tables
- If entry found, actions applied to packet and counters are updated
- If entry is not found, packet sent to userspace and miss counter incremented
Megaflows

- Version 1.11 added support for wildcarding in the datapath
- ovs-vswitchd dynamically determines how much wildcarding can be done:
  - Flow table
  - Actions from matching flow
  - General switch configuration (e.g., bonding)
- With megaflows, “normal” performance close to Linux bridge
Tunnels

- Tunnels in OVS are just virtual ports with own OpenFlow port number
- Keys set statically at creation time or dynamically through OpenFlow action
- Types:
  - GRE
  - VXLAN
  - LISP
- Visible in kernel datapath:
  - ovs-dpctl show
UTILITIES
OpenFlow

- **ovs-ofctl speaks to OpenFlow module**
  - ovs-ofctl show <bridge>
  - ovs-ofctl dump-flows <bridge>
  - ovs-ofctl add-flow <bridge> <flow>
  - ovs-ofctl del-flows <bridge> [flow]
  - ovs-ofctl snoop <bridge>

- **OpenFlow plus extensions**
  - Resubmit Action: Simulate multiple tables in a single table
  - NXM: Extensible match
  - Registers: Eight 32-bit metadata registers
  - Fine-grained control over multiple controllers

- **See “hidden” flows (in-band, fail-open, etc):**
  - ovs-appctl bridge/dump-flows <bridge>
ovs-ofctl show <br>

root@vm-vswitch:~# ovs-ofctl show br0
OFPT_FEATURES_REPLY (xid=0x2): dpid:0000505400000005
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS ARP_MATCH_IP
actions: OUTPUT SET_VLAN_VID SET_VLAN_PCP STRIP_VLAN SET_DL_SRC SET_DL_DST SET_NW_SRC
SET_NW_DST SET_NW_TOS SET_TP_SRC SET_TP_DST ENQUEUE
1(eth0): addr:50:54:00:00:00:05
   config: 0
   state: 0
   current: 1GB-FD COPPER AUTO_NEG
   advertised: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG
   supported: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG
   speed: 1000 Mbps now, 1000 Mbps max
2(eth1): addr:50:54:00:00:00:06
   config: 0
   state: 0
   current: 1GB-FD COPPER AUTO_NEG
   advertised: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG
   supported: 10MB-HD 10MB-FD 100MB-HD 100MB-FD 1GB-FD COPPER AUTO_NEG
   speed: 1000 Mbps now, 1000 Mbps max
LOCAL(br0): addr:50:54:00:00:00:05
   config: 0
   state: 0
   speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
The default flow table includes a single entry that does “normal” processing:

```bash
root@vm-vswitch:~# ovs-ofctl dump-flows br0
NXST_FLOW reply (xid=0x4):
  cookie=0x0, duration=4.05s, table=0, n_packets=8, n_bytes=784,
  idle_age=0, priority=0 actions=NORMAL
```
Hidden Flows

- There are flows that OVS uses for its own purpose that are higher priority than can be configured from outside.

- Types
  - In-band control (priority >= 180000): Allow control traffic to pass regardless of configured flows.
  - Fail-open (priority = 0xf0f0f0): Allow all traffic to pass when a connection to the controller fails.

- They are hidden from controllers and commands like “ovs-ofctl dump-flows” due to being higher priority than OpenFlow allows (>65535).

- Only visible with “ovs-appctl bridge/dump-flows <bridge>”
Kernel Datapath

- ovs-dpctl speaks to kernel module
- See datapaths and their attached interfaces:
  - ovs-dpctl show
- See flows cached in datapath:
  - ovs-dpctl dump-flows
**ovs-dpctl show**

```
root@vm-vswitch:~# ovs-dpctl show
system@ovs-system:
  lookups: hit:188056 missed:7722 lost:0
  flows: 2
  masks: hit:199268 total:1 hit/pkt:1.02
  port 0: ovs-system (internal)
  port 1: br0 (internal)
  port 2: eth0
  port 3: eth1
```

- **hit**: Packets hit existing entry
- **missed**: Packets sent to userspace
- **lost**: Dropped before getting to userspace

**Datapath port number**

**Interface name**

**Interface type**
ovs-dpctl dump-flows

- Flows used to be exact-match:

  in_port(2), eth(src=50:54:00:00:00:01, dst=50:54:00:00:00:03), eth_type(0x0800), ipv4(src=192.168.0.1, dst=192.168.0.2, proto=1, tos=0, ttl=64, frag=no), icmp(type=8, code=0), packets:3, bytes:294, used:0.185s, actions:3

  in_port(3), eth(src=50:54:00:00:00:03, dst=50:54:00:00:00:01), eth_type(0x0800), ipv4(src=192.168.0.2, dst=192.168.0.1, proto=1, tos=0, ttl=64, frag=no), icmp(type=0, code=0), packets:3, bytes:294, used:0.205s, actions:2

- Starting in OVS 1.11, may contain wildcards:

  in_port(3), eth(src=50:54:00:00:00:03, dst=50:54:00:00:00:01), eth_type(0x0800), ipv4(src=192.168.0.2/0.0.0.0, dst=192.168.0.1/0.0.0.0, proto=1/0, tos=0/0, ttl=64/0, frag=no/0x2), icmp(type=0/0, code=0/0), packets:95, bytes:9310, used:0.425s, actions:2

  in_port(2), eth(src=50:54:00:00:00:01, dst=50:54:00:00:00:03), eth_type(0x0800), ipv4(src=192.168.0.1/0.0.0.0, dst=192.168.0.2/0.0.0.0, proto=1/0, tos=0/0, ttl=64/0, frag=no/0x2), icmp(type=8/0, code=0/0), packets:95, bytes:9310, used:0.525s, actions:3
**ovs-appctl**

- Utility to invoke runtime control and query facilities in most OVS daemons
- The “-t <target>” option specifies the daemon name (default is ovs-vswitchd)
- All daemons support the following commands:
  - help – Lists the commands supported by the target
  - version – Displays the version and compilation date of the target
  - vlog/list – List the known logging modules and their current levels
  - vlog/set [spec] – Sets logging levels
- Many interesting features supported, which are defined in the targets’ man pages
Flow Debugging

- Flow tables can become incredibly complex, but OVS has tools to make it easier to debug

- Here is a set of rules to (poorly) implement a firewall (with an unnecessary resubmit) to block all TCP traffic except port 80:

  # Move TCP traffic arriving on port 1 to next stage of “pipeline”
  priority=100, tcp, in_port=1 actions=resubmit:4000

  # Allow port TCP port 80 traffic (and implicitly drop all others)
  priority=100, tcp, in_port=4000, tp_dst=80 actions=NORMAL

  # Allow all non-TCP traffic arriving on port 1
  priority=90, in_port=1 actions=NORMAL

  # Allow all traffic arriving on port 2
  priority=100, in_port=2 actions=NORMAL
Tracing Flow (ICMP Allowed)

```
root@vm-vswitch:~# ovs-appctl ofproto/trace
"skb_priority(0),in_port(2),skb_mark(0),eth(src=50:54:00:00:00:01,dst=50:54
:00:00:00:03),eth_type(0x0800),ipv4(src=192.168.0.1,dst=192.168.0.2,proto=1
,tos=0,ttl=64,frag=no),icmp(type=8,code=0)"
Bridge: br0
Flow: icmp,metadata=0,in_port=1,vlan_tci=0x0000,dl_src=50:54:00:00:00:01,dl_dst=5
0:54:00:00:00:03,nw_src=192.168.0.1,nw_dst=192.168.0.2,nw_tos=0,nw_ecn=0,nw
_ttl=64,icmp_type=8,icmp_code=0
Rule: table=0 cookie=0 priority=90,in_port=1
OpenFlow actions=NORMAL
forwarding to learned port

Final flow: unchanged
Relevant fields: skb_priority=0,icmp,in_port=1,vlan_tci=0x0000/0xffff,dl_src=50:54:00:00:00:01,dl_dst=50:54:00:00:00:03,nw_frag=no,icmp_code=0
Datapath actions: 3
```

Applied OpenFlow rule
Datapath flow description
Datapath action
Tracing Flow (TCP allowed)

root@vm-vswitch:~# ovs-appctl ofproto/trace
"skb_priority(0), in_port(2), skb_mark(0), eth(src=50:54:00:00:00:01, dst=50:54:00:00:00:03), eth_type(0x0800), ipv4(src=192.168.0.1, dst=192.168.0.2, proto=6, tos=0x10, ttl=64, frag=no), tcp(src=56176, dst=80), tcp_flags(0x002)"

Bridge: br0

Flow:
tcp, metadata=0, in_port=1, vlan_tci=0x0000, dl_src=50:54:00:00:00:01, dl_dst=50:54:00:00:00:03, nw_src=192.168.0.1, nw_dst=192.168.0.2, nw_tos=16, nw_ecn=0, nw_ttl=64, tp_src=56176, tp_dst=80, tcp_flags=0x002

Rule: table=0 cookie=0 priority=100, tcp, in_port=1
OpenFlow actions=resubmit:4000

First applied OpenFlow rule

Resubmitted flow: unchanged
Resubmitted regs: reg0=0x0 reg1=0x0 reg2=0x0 reg3=0x0 reg4=0x0
reg5=0x0 reg6=0x0 reg7=0x0
Resubmitted odp: drop
Rule: table=0 cookie=0 priority=100, tcp, in_port=4000, tp_dst=80
OpenFlow actions=NORMAL
forwarding to learned port

Second applied OpenFlow rule

Final flow: unchanged

Relevant fields:
skb_priority=0, tcp, in_port=1, vlan_tci=0x0000/0x1fff, dl_src=50:54:00:00:00:01, dl_dst=50:54:00:00:00:03, nw_frag=no, tp_dst=80
Datapath actions: 3

Datapath flow description

Datapath action
Tracing Flow (TCP denied)

root@vm-vswitch:~# ovs-appctl ofproto/trace
"skb_priority(0),in_port(2),skb_mark(0),eth(src=50:54:00:00:00:01,dst=50:54:00:00:00:03),eth_type(0x0800),ipv4(src=192.168.0.1,dst=192.168.0.2,proto=6,tos=0x10,ttl=64,frag=no),tcp(src=56177,dst=100),tcp_flags(0x002)"
Bridge: br0
Flow:
tcp,metadata=0,in_port=1,vlan_tci=0x0000,dl_src=50:54:00:00:00:01,dl_dst=50:54:00:00:00:03,nw_src=192.168.0.1,nw_dst=192.168.0.2,nw_tos=16,nw_ecn=0,nw_ttl=64,tp_src=56177,tp_dst=100,tcp_flags=0x002
Rule: table=0 cookie=0 priority=100 tcp,in_port=1
OpenFlow actions=resubmit:4000

Resubmitted flow: unchanged
Resubmitted regs: reg0=0x0 reg1=0x0 reg2=0x0 reg3=0x0 reg4=0x0 reg5=0x0 reg6=0x0 reg7=0x0
No match

No matching second flow, so implicit drop

Datapath flow description

Final flow: unchanged
Relevant fields: skb_priority=0,tcp,in_port=1,nw_frag=no,tp_dst=100
Datapath actions: drop

Select the relevant parts of the text for each section as needed.
Logging

- ovs-appctl configures running OVS daemons
- Most common use is to modify logging levels
- By default configures ovs-vswitchd, but “-t” option changes target
- Default level for log files is “info”, only thing lower is “dbg”

<table>
<thead>
<tr>
<th>bridge</th>
<th>console</th>
<th>syslog</th>
<th>file</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EMER</td>
<td>ERR</td>
<td>INFO</td>
</tr>
<tr>
<td>vswitchd</td>
<td>EMER</td>
<td>ERR</td>
<td>INFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
root@vm-vswitch:~# ovs-appctl vlog/list

<table>
<thead>
<tr>
<th>console</th>
<th>syslog</th>
<th>file</th>
</tr>
</thead>
<tbody>
<tr>
<td>--------</td>
<td>--------</td>
<td>------</td>
</tr>
</tbody>
</table>

bridge    | EMER   | ERR   | INFO |
```

```
root@vm-vswitch:~# ovs-appctl vlog/set ofproto:file:dbg
```
Log Files

- **Open vSwitch logs:** /var/log/openvswitch/*
  - ovs-vswitchd.log
  - ovsdb-server.log
- **System:** /var/log/messages
- **Configuration Database:** /etc/openvswitch/conf.db
Questions?

- Try the documentation, we strive to make it thorough and up to date
- Look at the FAQ:
  - http://openvswitch.org/faq/
- Ask questions on the mailing list:
  - discuss@openvswitch.org