OpenVSwitch

Dean Pemberton – NSRC
Andy Linton – NSRC
Sam Russell – REANNZ

UNIVERSITY OF OREGON

NSRC
Network Startup Resource Center

Why Open vSwitch…

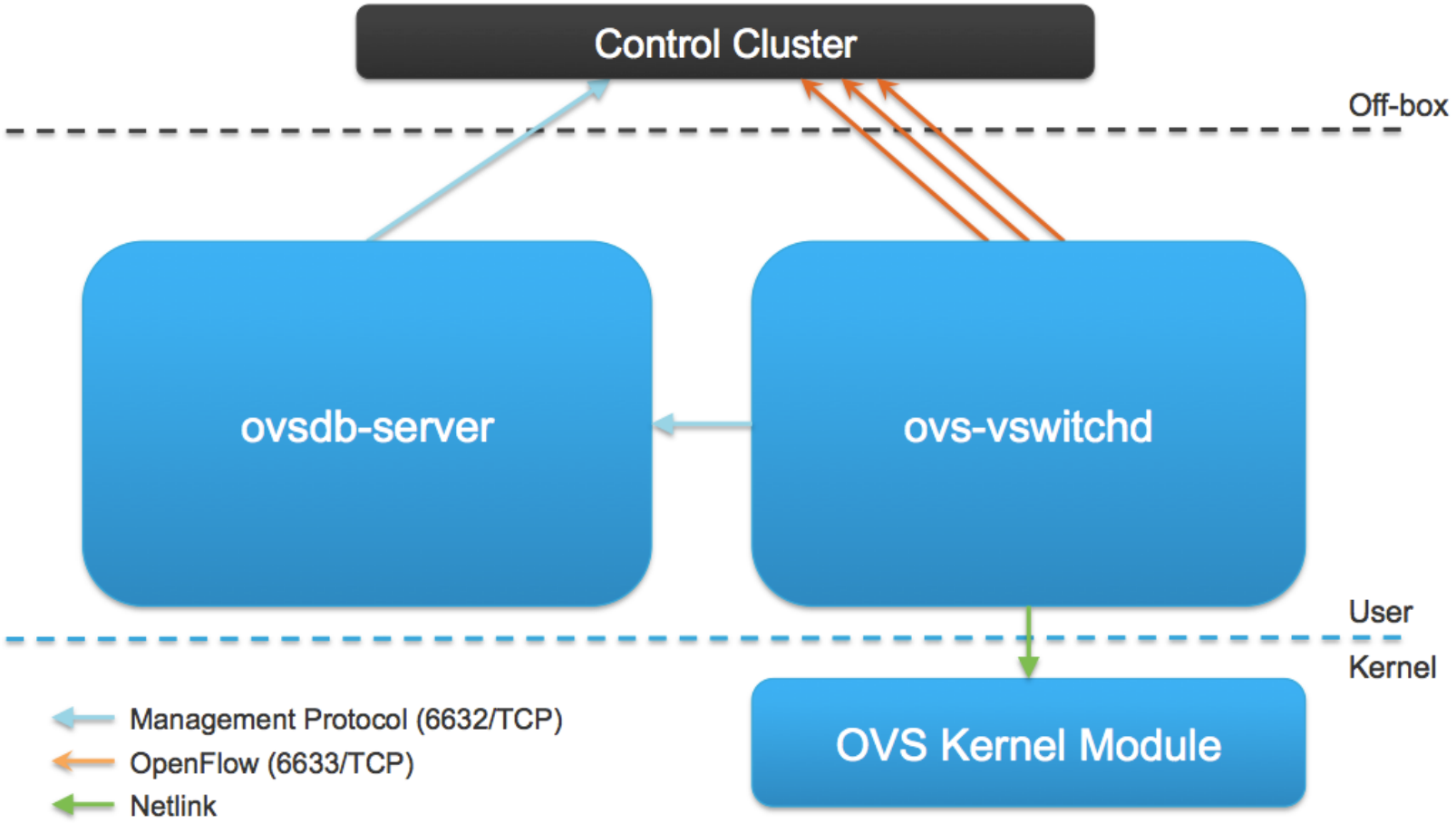
Open vSwitch's forwarding path (the in-kernel datapath) is designed to be amenable to "offloading" packet processing to hardware chipsets, whether housed in a classic hardware switch chassis or in an end-host NIC.

This allows for the Open vSwitch control path to be able to both control a pure software implementation or a hardware switch.

…Why Open vSwitch

The advantage of hardware integration is not only performance within virtualized environments. If physical switches also expose the Open vSwitch control abstractions, both bare-metal and virtualized hosting environments can be managed using the same mechanism for automated network control.

Components

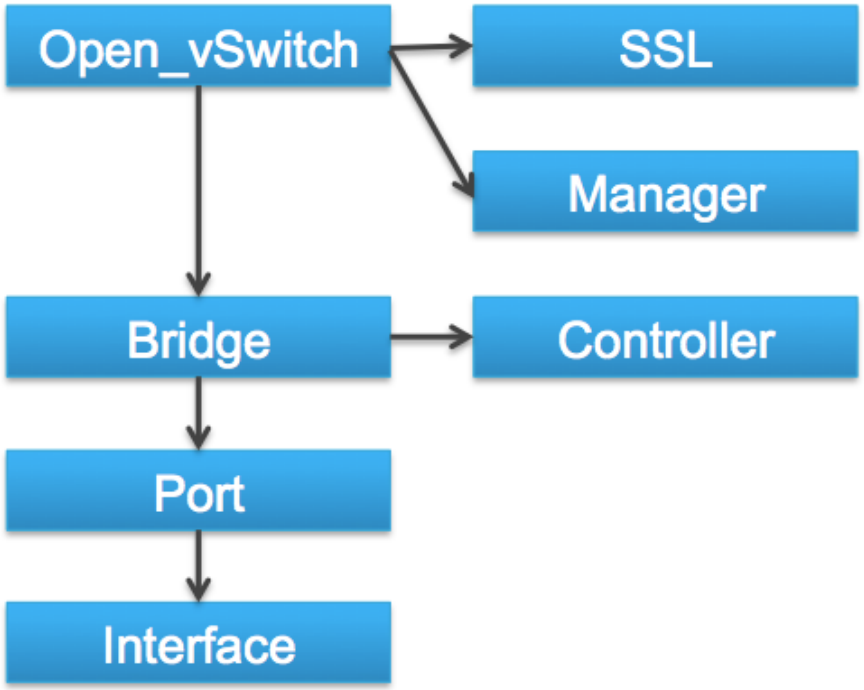

Control Cluster
Off-box
ovsdb-server
ovs-vswitchd
User
Kernel
OVS Kernel Module
Management Protocol (6632/TCP)
OpenFlow (6633/TCP)
Netlink

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

Core Tables

"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.

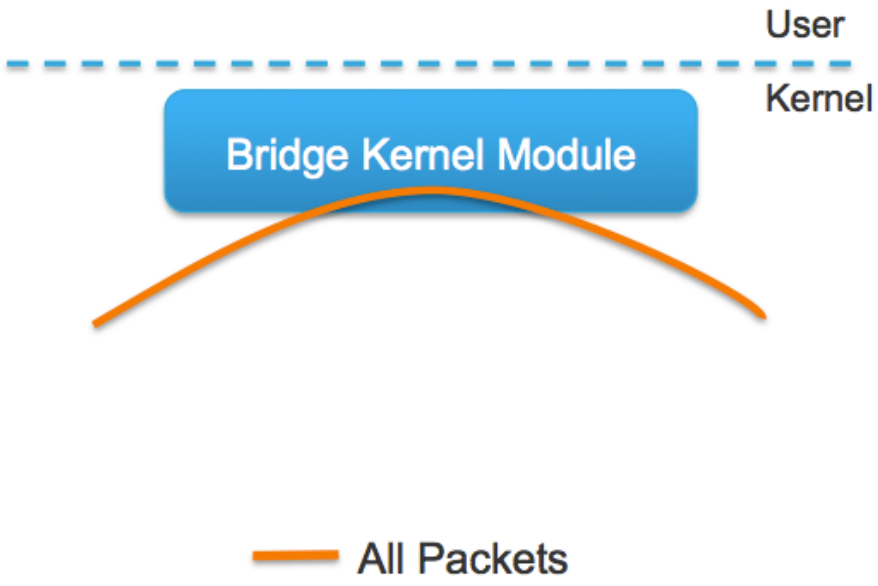


Debugging the Database

- 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>

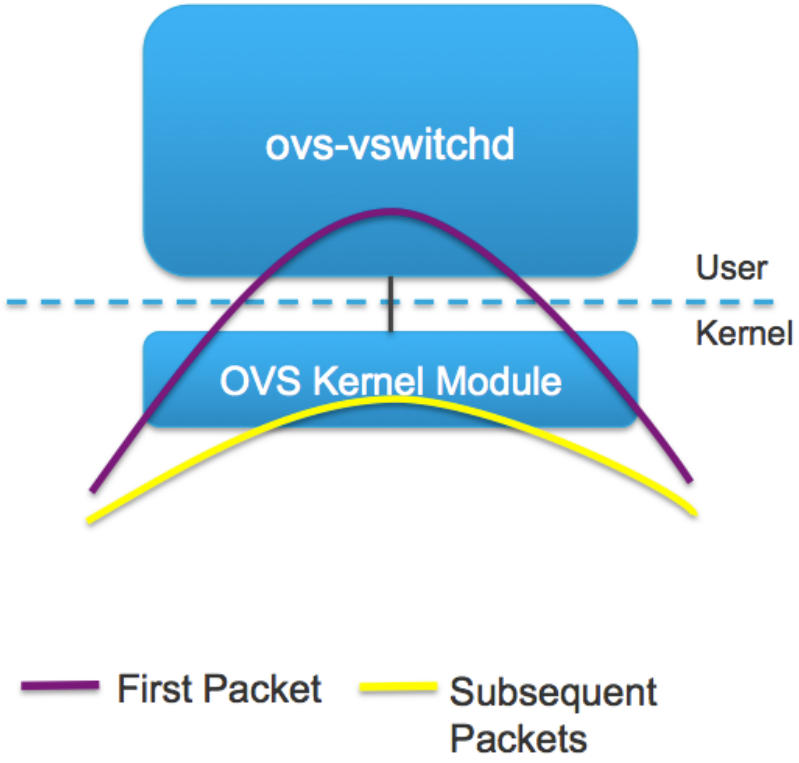
Linux Bridge Design

- Simple forwarding
- Matches destinatior 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

OVS and 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 dump-flows

- The default flow table includes a single entry that does “normal” processing:

```
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
```

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

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=0x0000/0x1fff,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=50:5
4:00:00:00:03),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=5
0:54:00:00:00:03,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

    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

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
```

First applied OpenFlow rule

Second applied OpenFlow rule

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=0x1
0,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
    Resubmitted  odp: drop
    No match

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

First applied OpenFlow Rule

No matching second flow, so implicit drop

Datapath flow description

Datapath action

UNIVERSITY OF OREGON

NSRC
Network Startup Resource Center

Links

- http://openvswitch.org/
- Examples and diagrams from - http://openvswitch.org/slides/OpenStack-131107.pdf