几个你也许不知道的ovs知识
组播侦听如何与VLAN配合使用
OVS不缓冲数据包
流表dump命令区别
创建ovs bridge流程分析：
backer的作用
网桥重配置的作用
revalidate
handler

几个你也许不知道的ovs知识

组播侦听如何与VLAN配合使用

- ovs会维护每个vlan的侦听表

OVS不缓冲数据包

- Open vSwitch 快速路径数据包处理使用“运行至完成”模型，其中每个数据包都在一次通过中完全处理。因此，在数据包只是通过快速路径的常见情况下，Open vSwitch 本身并不缓冲数据包。接收和稍后传输数据包所涉及的操作系统和网络驱动程序通常包括缓冲。Open vSwitch 只是它们之间的中间人，不能直接访问或影响它们的缓冲区。
- 当 OVS 快速路径处理与其兆流缓存中的任何流都不匹配的数据包时，它会将该数据包传递给 Open vSwitch 慢速路径。此过程将数据包的副本排队到处理它的 Open vSwitch 用户空间，并在必要时将其传递回内核(或者是dpdk收发包)模块。

流表dump命令区别

- `ovs-ofctl dump-flows` dumps OpenFlow flows, excluding hidden flows. This is the most commonly useful form of flow dump. (Unlike the other commands, this should work with any OpenFlow switch, not just Open vSwitch.)
- `ovs-appctl bridge/dump-flows` dumps OpenFlow flows, including hidden flows. This is occasionally useful for troubleshooting suspected issues with in-band control.
- `ovs-dpctl dump-flows [dp]` dumps the datapath flow table entries for a Linux kernel-based datapath. In Open vSwitch 1.10 and later, ovs-vswitchd merges multiple switches into a single datapath, so it will show all the flows on all your kernel-based switches. This command can occasionally be useful for debugging. It doesn't dump flows that was offloaded to hardware.
- `ovs-appctl dpif/dump-flows`, new in Open vSwitch 1.10, dumps datapath flows for only the specified bridge, regardless of the type. Supports dumping of HW offloaded flows. See ovs-vswitchd(8) for details.

创建ovs bridge流程分析：

1.通过ovs-vsctl 创建网桥，将创建参数发送给ovsdb-server，ovsdb-server将数据写入数据库。

2.ovs-vswitchd从ovsdb-server中读取创建网桥的信息，在ovs-vswithd层创建一个bridge结构体信息。

3.然后将brdige信息应用到ofproto层，在ofproto层通过ofproto_create创建网桥，ofproto_create通过用户指定的网桥类型查找包含该类型的ofproto provider(目前只支持一个ofproto provider)。
查找后创建ofproto结构体(该结构体也表示一个bridge)，并通过ofproto provider 构造函数创建ofproto provider的私有信息。

4.ofproto-dpif 层，构造函数完成如下：ofproto-dpif会为相同类型的ofproto创建一个backer结构体，所有类型的ofproto的backer使用全局列表表示。ofproto-dpif通过backer关联dpif。同时backer关联upcall处理线程
netdev没有实现upcall注册函数，所以对应的backer线程实际上不做任何处理，但依然会有该处理线程。netlink 通过backer启动的线程实现处理upcall数据包的处理。

backer的作用

- netlink 通过backer启动的线程实现处理upcall数据包的处理。

网桥重配置的作用

bridge_reconfigure 主要完成根据数据以及当前进程信息，创建、更新、删除必要的网桥、接口、端口以及其他协议的配置等。最终这些操作为应用到ofproto 层、ofproto dpif 层、run_stats_update、run_status_update、run_system_stats 更新openvswitch数据库状态信息netdev_run netdev_linux_run监控网卡状态并更新。

以STP 某个功能为例，例如之前的一个bug，OvS 不会检查stp port 所处的状态。如果已经加入的port开启的stp功能，用户使用ifconfig 禁止网卡状态，那么理论上OvS 应该停止通过该端口发送数据包括BPDU。但是OvS 并没有做这方便的检查。同时如果ifconfig up了该网卡，那么重新进入stp 状态，以及stp 各个状态的改变。

社区最后接收了该bug的修复，patch 如下：https://github.com/openvswitch/ovs/commit/52182c5f50198d0f985b10677e47a9ac49ee709b

revalidate

```
1   udpif_start_threads函数负责启动revalidate与handler线程
2
3
4
5   switch (backer->need_revalidate) {
6    case REV_RECONFIGURE:    COVERAGE_INC(rev_reconfigure);    break;
7    case REV_STP:            COVERAGE_INC(rev_stp);            break;
8    case REV_RSTP:           COVERAGE_INC(rev_rstp);           break;
9    case REV_BOND:           COVERAGE_INC(rev_bond);           break;
10   case REV_PORT_TOGGLED:   COVERAGE_INC(rev_port_toggled);   break;
11   case REV_FLOW_TABLE:     COVERAGE_INC(rev_flow_table);     break;
12   case REV_MAC_LEARNING:   COVERAGE_INC(rev_mac_learning);   break;
13   case REV_MCAST_SNOOPING: COVERAGE_INC(rev_mcast_snooping); break;
14    }
```

```
    backer->need_revalidate = 0;
```

handler