Open vSwitch 安装及配置

N.J.C.H

一、Open vSwitch 简介

1.1 概述

Open vSwitch 是一个高质量的、多层虚拟交换机，使用开源 Apache 2.0 许可协议，由 Nicira Networks 开发，主要实现代码为可移植的 C 代码。

它的目的是让大规模网络自动化可以通过编程扩展,同时仍然支持标准的管理接口和协议（例如 NetFlow, sFlow, SPAN, RSPAN, CLI, LACP, 802.1ag）。此 外,它被设计位支持跨越多个物理服务器的分布式环境，类似于 VMware 的 vNetwork 分布式 vswitch 或 Cisco Nexus 1000 V。

Open vSwitch 支持多种 linux 虚拟化技术，包括 Xen/XenServer， KVM，和 VirtualBox。

1.2 模块介绍

当前最新代码包主要包括以下模块和特性：
ovs-vswitchd 主要模块，实现 switch 的 daemon，包括一个支持流交换的 Linux 内核模块；
ovsdb-server 轻量级数据库服务器，提供 ovs-vswitchd 获取配置信息；
ovs-brcompatd 让 ovs-vswitch 替换 Linux bridge，包括获取 bridge ioctls 的 Linux 内核模块；
ovs-dpctl 用来配置 switch 内核模块；
一些 Scripts and specs 辅助 OVS 安装在 Citrix XenServer 上，作为默认 switch；
ovs-vsctl 查询和更新 ovs-vswitchd 的配置；
ovs-appctl 发送命令消息，运行相关 daemon；
ovsdbmonitor GUI 工具，可以远程获取 OVS 数据库和 OpenFlow 的流表。

此外，OVS 也提供了支持 OpenFlow 的特性实现，包括
ovs-openflowd：一个简单的 OpenFlow 交换机；
ovs-controller：一个简单的 OpenFlow 控制器；
ovs-ofctl 查询和控制 OpenFlow 交换机和控制器；
ovs-pki ：OpenFlow 交换机创建和管理公钥框架；
ovs-tcpundump：tcpdump 的补丁，解析 OpenFlow 的消息；

1.3 运行原理

内核模块实现了多个“数据路径”（类似于网桥），每个都可以有多个“vports”（类似于桥内的

端口）。每个数据路径也通过关联一下流表（flow table）来设置操作，而这些流表中的流都是用户空间在报文头和元数据的基础上映射的关键信息，一般的操作都是将数据包转发到另一个 vport。当一个数据包到达一个 vport，内核模块所做的处理是提取其流的关键信息并在流表中查找这些关键信息。当有一个匹配的流时它执行对应的操作。如果没有匹配，它会将数据包送到用户空间的处理队列中（作为处理的一部分，用户空间可能会设置一个流用于以后碰到相同类型的数据包可以在内核中执行操作）。

二、准备工作

2.1 说明

在 Open vSwitch 分布中编译 userspace 程序，需要下面软件

```
- A make program, e.g. GNU make.  BSD make should also work.

- The GNU C compiler.  We generally test with version 4.1, 4.2, or 4.3.

- pkg-config.  We test with version 0.22.

- libssl, from OpenSSL, is optional but recommended if you plan to
  connect the Open vSwitch to an OpenFlow controller.  libssl is
  required to establish confidentiality and authenticity in the
  connections from an Open vSwitch to an OpenFlow controller.  If
  libssl is installed, then Open vSwitch will automatically build
  with support for it.
```

在 linux 编译内核模块，还需要安装下面的软件。如果你不能建造或安装内核模块，你可以使用 userspace-only 实现。

```
The Open vSwitch datapath requires bridging support
 (CONFIG_BRIDGE) to be built as a kernel module.  (This is common
 in kernels provided by Linux distributions.)  The bridge module
 must not be loaded or in use.  If the bridge module is running
```

```
    (check with "lsmod | grep bridge"), you must remove it ("rmmod
    bridge") before starting the datapath.
```

2.2 操作

安装 Open vSwitch 之前确保安装了下面的软件

```
  - libc compatible with the libc used for build.
  - libssl compatible with the libssl used for build, if OpenSSL was
    used for the build.
  - On Linux, the same kernel version configured as part of the build.
  - For optional support of ingress policing on Linux, the "tc" program
    from iproute2 (part of all major distributions and available at
    http://www.linux-foundation.org/en/Net:Iproute2).
```

执行

```
apt-get install build-essential

apt-get install openssl
```

On Linux you should ensure that /dev/urandom exists. To support TAP devices, you must also ensure that /dev/net/tun exists.

```
root@ubuntu229:~# ls -l /dev/urandom
crw-rw-rw- 1 root root 1, 9 Nov  5 08:52 /dev/urandom
```

```
root@ubuntu229:~# ls -l /dev/net/tun
crw-rw-rwT 1 root root 10, 200 Nov  5 08:52 /dev/net/tun
```

三、安装 Open vSwitch

3.1 下载

Openvswitch包下载地址：http://openvswitch.org/download/

```
root@ubuntu229:~# wget
http://openvswitch.org/releases/openvswitch-1.9.0.tar.gz
```

3.2 解压

```
root@ubuntu229:~# tar -xzf openvswitch-1.9.0.tar.gz
```

3.3 安装

生成内核模式的 Open vSwitch 时需指定的内核源码编译目录，基本步骤如下

```
cd openvswitch-1.9.0/

./configure --with-linux=/lib/modules/`uname -r`/build

make && make install

insmod datapath/linux/openvswitch.ko
```

补充一下 DataPath 的概念，OpenFlow 就是控制和数据转发分离，而控制端，就叫做 Controller Path,比如 floodlight 等；负责转发数据的数据转发端，就叫做 DataPath，也就是支持 OpenFlow 的硬件或者软件交换机 Switch。

四、配置 Open vSwitch

建立 OpenVSwitch 配置文件和数据库

```
root@ubuntu229:~# mkdir -p /usr/local/etc/openvswitch
```

```
root@ubuntu229:~# ovsdb-tool create /usr/local/etc/openvswitch/conf.db
/usr/local/share/openvswitch/vswitch.ovsschema
```

启动配置数据库

```
root@bogon:~# ovsdb-server
--remote=punix:/usr/local/var/run/openvswitch/db.sock
--remote=db:Open_vSwitch,manager_options  --private-key=db:SSL,private_key
--certificate=db:SSL,certificate --bootstrap-ca-cert=db:SSL,ca_cert
--pidfile --detach
```

如果你 built Open vSwitch 没有加 SSL 支持，省略 --private-key, --certificate, and --bootstrap-ca-cert

查启动情况

```
root@ubuntu229:~# ps -ef |grep ovsdb-server
root     13245     1  0 13:27 ?        00:00:00 ovsdb-server --remote=punix:/usr/local/var/run/o
penvswitch/db.sock --remote=db:Open_vSwitch,manager_options --private-key=db:SSL,private_key --c
ertificate=db:SSL,certificate --bootstrap-ca-cert=db:SSL,ca_cert --pidfile --detach
root     13253  7631  0 13:28 pts/0    00:00:00 grep --color=auto ovsdb-server
```

初始化数据库

```
root@ubuntu229:~# ovs-vsctl --no-wait init

//仅需在第一次创建数据库时运行，但是每次都运行也没问题
```

启动 Open vSwitch daemon，连接到同样的 Unix domain socket 上

```
root@ubuntu229:~# ovs-vswitchd --pidfile --detach
```

成功后，会有三个进程，一 ovs_个 workeq 进程，一个 ovs-vswitchd 进程，一个 ovsdb-server 进程

```
root@ubuntu229:~# ps -ef |grep ovs
root      5415     2  0 Nov07 ?        00:00:00 [ovs_workq]
root     11157     1  0 Nov07 ?        00:00:07 ovsdb-server --remote=punix:/usr/local/var/run/o
penvswitch/db.sock --remote=db:Open_vSwitch,manager_options --private-key=db:SSL,private_key --c
ertificate=db:SSL,certificate --bootstrap-ca-cert=db:SSL,ca_cert --pidfile --detach
root     11160     1  0 Nov07 ?        00:00:48 ovs-vswitchd --pidfile --detach
root     14090 11925  0 10:54 pts/0    00:00:00 grep --color=auto ovs
```

五、升级 Open vSwitch

升级 Open vSwitch 的时候，也需要升级数据库架构，步骤如下

5.1 停止 Open vSwitch daemons

```
kill `cd /usr/local/var/run/openvswitch && cat ovsdb-server.pid
ovs-vswitchd.pid`
```

5.2 安装新的版本

方法同上，下载，解压，configure make make install

5.3 升级数据库

两种情况如下
如果数据库没有重要的信息,可以直接删除数据库文件,然后用 ovsdb-tool 命令重新创建，方法同上。

如果要保留数据库的内容，首先把它备份一下，然后使用 ovsdb-tool convert 命令升级，如下

```
ovsdb-tool convert /usr/local/etc/openvswitch/conf.db
/usr/local/share/openvswitch/vswitch.ovsschema
```

5.4 启动 Open vSwitch daemon

方法同上。

六、使用 Open vSwitch

6.1 网桥管理

6.1.1 非 ovsdb 数据库操作

添加名为 br0 的网桥

```
root@ubuntu229:~# ovs-vsctl add-br br0
```

列出所有网桥

```
root@ubuntu229:~# ovs-vsctl list-br

br0
```

判断网桥 br0 是否存在

```
root@ubuntu229:~# ovs-vsctl br-exists br0

echo $?  //0 表示存在，否则不存在
```

将网络接口 eth0 挂接到网桥 br0 上

```
root@ubuntu229:~# ovs-vsctl add-port br0 eth0
```

列出挂接到网桥 br0 上的所有网络接口

```
root@ubuntu229:~# ovs-vsctl list-ports br0

eth0
```

列出已挂接 eth0 网络接口的网桥

```
root@ubuntu229:~# ovs-vsctl port-to-br eth0

br0
```

查看结果

```
root@ubuntu229:~# ovs-vsctl show

131648b5-f7a6-4949-9a39-273ed62c0922
```

```
    Bridge "br0"
        Port "br0"
            Interface "br0"
                type: internal
        Port "eth0"
            Interface "eth0"
```

删除网桥 br0 上挂接的 eth0 网络接口

```
root@ubuntu229:~# ovs-vsctl del-port br0 eth0
```

删除名为 br0 的网桥

```
root@ubuntu229:~# ovs-vsctl del-br br0
```

6.1.2 ovsdb 数据库操作

ovsdb 是一个非常轻量级的数据库，与其说它是一个数据库，不如说它是一个提供增删查改等功能的临时配置缓存，之所以这么说，是因为 ovsdb 数据库的根本就未使用多少数据库技术，如 SQL 语言查询、存储过程等等。ovsdb 数据库通过模式文件“openvswitch-1.1.0pre2/vswitchd/vswitch.ovsschema”，如要定制 ovsdb 数据库，可通过更改 vswitch.ovsschema 文件实现。

数据库操作的一般格式为

```
ovs-vsctl list/set/get/add/remove/clear/destroy table record column [value]
```

默认情况下 ovsdb 中有以下数据表

```
bridge,
controller,interface,mirror,netflow,open_vswitch,port,qos,queue,ssl,sflow
```

查看 bridge 数据表中的所有记录

```
ovs-vsctl list bridge
```

```
root@ubuntu229:~# ovs-vsctl list bridge
_uuid               : 7298956e-9f6d-4633-86df-93b5e62e1eab
controller          : []
datapath_id         : []
datapath_type       : ""
external_ids        : {}
fail_mode           : []
flood_vlans         : []
flow_tables         : {}
mirrors             : []
name                : "br0"
netflow             : []
other_config        : {}
ports               : [51e40707-bba5-4803-bc65-c6e3bc8e40ac]
sflow               : []
status              : {}
stp_enable          : false
```

获取 bridge 的_uuid 字段值

```
ovs-vsctl get bridge br0 _uuid
```

```
root@ubuntu229:~# ovs-vsctl get bridge br0 _uuid
7298956e-9f6d-4633-86df-93b5e62e1eab
```

设置 bridge 数据表 datapath_type 字段的值

```
ovs-vsctl set bridge br0 datapath_type="system"
```

```
root@ubuntu229:~# ovs-vsctl set bridge br0 datapath_type="system"
root@ubuntu229:~# ovs-vsctl get bridge br0 datapath_type
system
```

清除 bridge 数据表 flood_vlans 字段的值

```
ovs-vsctl clear bridge br0 flood_vlans  or

ovs-vsctl remove bridge br0 flow_tables 23
```

删除 uuid 为 69ee0c09-9e52-4236-8af6-037a98ca704d 的 qos 记录

```
ovs-vsctl destroy qos 69ee0c09-9e52-4236-8af6-037a98ca704d
```

6.2 流规则管理

6.2.1 流规则组成

每条流规则由一系列字段组成，分为基本字段、条件字段和动作字段三部分。

基本字段包括生效时间 duration_sec、所属表项 table_id、优先级 priority、处理的数据包数 n_packets,空闲超时时间 idle_timeout 等,空闲超时时间 idle_timeout 以秒为单位，超过设置的空闲超时时间后该流规则将被自动删除,空闲超时时间设置为 0 表示该流规则永不过期，idle_timeout 将不包含于 ovs-ofctl dump-flows brname 的输出中。

条件字段包括输入端口号 in_port、源目的 mac 地址 dl_src/dl_dst、源目的 ip 地址 nw_src/nw_dst、数据包类型 dl_type、网络层协议类型 nw_proto 等，可以为这些字段的任意组合，但在网络分层结构中底层的字段未给出确定值时上层的字段不允许给确定值，即一条流规则中允许底层协议字段指定为确定值，高层协议字段指定为通配符(不指定即为匹配任何值)，而不允许高层协议字段指定为确定值，而底层协议字段却为通配符(不指定即为匹配任何值)，否则，ovs-vswitchd 中的流规则将全部丢失，网络无法连接。

动作字段包括正常转发 normal、定向到某交换机端口 output：port、丢弃 drop、更改源目的 mac 地址 mod_dl_src/mod_dl_dst 等，一条流规则可有多个动作，动作执行按指定的先后顺序依次完成。

6.2.2 基本操作

查看虚拟交换机的信息

```
root@bogon:~# ovs-ofctl show br0

OFPT_FEATURES_REPLY (xid=0x1): dpid:00004a662add9d41

n_tables:255, 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

 LOCAL(br0): addr:4a:66:2a:dd:9d:41

     config:     PORT_DOWN

     state:      LINK_DOWN

     speed: 100 Mbps now, 100 Mbps max

OFPT_GET_CONFIG_REPLY (xid=0x3): frags=normal miss_send_len=0
```

查看 br0 上各交换机端口的状态

```
root@bogon:~# ovs-ofctl dump-ports br0

OFPST_PORT reply (xid=0x1): 1 ports

  port 65534: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0

           tx pkts=0, bytes=0, drop=0, errs=0, coll=0
```

注：输出的结果中包含了各网络接口上收到的数据包数，字节数，丢包数，错误数据包数等信息

添加一条流规则：丢弃从 2 号端口发来的所有数据包

```
root@bogon:~# ovs-ofctl add-flow br0 idle_timeout=120,in_port=2,actions=drop
```

查看 br0 上所有的流规则

```
root@bogon:~# ovs-ofctl dump-flows br0

NXST_FLOW reply (xid=0x4):

 cookie=
```

```
0x0, duration=56.824s, table=0, n_packets=0, n_bytes=0, idle_timeout=120,
idle_age=56, in_port=2 actions=drop
```

删除一条流规则：删除条件字段中包含 in_port=2 的所有流规则

```
root@bogon:~# ovs-ofctl del-flows br0 in_port=2
```

6.3 应用场景设置

6.3.1 QoS 设置

针对网络接口的设置：
设置网络接口 eth4 的带宽为 1000±100kbps

```
ovs-vsctl set interface eth4 ingress_policing_rate=1000

ovs-vsctl set interface eth4 ingress_policing_burst=100
```

注：
ingress_policing_rate：最大发送速率(单位均为 kbps)
ingress_policing_burst：超过 ingress_policing_rate 的最大浮动值

针对交换机端口的设置：
创建在 vif0.0 端口上的 linux-htb QoS，linux-htb QoS 可以针对具有指定特征的数据包流设置最大最小带宽，且在最大带宽范围内，某一特征的数据包流可以借用其他特征数据包流未用完的带宽。

```
ovs-vsctl -- set port vif0.0 qos=@newqos -- --id=@newqos create qos type=linux-htb
other-config:

max-rate=100000000 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=500000000
```

6.3.2 端口映射

将发往 eth0 端口和从 eth1 端口发出的数据包全部定向到 eth2 端口
用 **ovs-vsctl list port** 命令查看 eth0、eth1、eth2 端口的 uuid 分别为：
a27e5ec3-5d78-437e-8e36-d6f81679132a
be52eece-1f03-4ccf-a4c6-b0b68cb25f8a
bc38e1c3-60a1-468e-89d7-e4b45585b533

命令如下

```
ovs-vsctl --set bridge br0 mirrors=@m-- --id=@m create mirror name=mymirror
select-dst-port=a27e5ec3-5d78-437e-8e36-d6f81679132a
select-src-port=be52eece-1f03-4ccf-a4c6-b0b68cb25f8a
output-port=bc38e1c3-60a1-468e-89d7-e4b45585b533
```

6.3.3 其他设置

网站屏蔽
屏蔽由 Open vSwitch 管理的任何主机对主机 119.75.213.50 的访问,但只屏蔽 ip 数据包(由 dl_type=0x0800 指定)，即所有主机将无法访问该主机上所有基于 IP 协议的服务，如万维网服务、FTP 访问等

```
ovs-ofctl add-flow br0
idle_timeout=0,dl_type=0x0800,nw_src=119.75.213.50,actions=drop
```

注：简写形式为将字段组简写为协议名，目前支持的简写有 ip，arp，icmp，tcp，udp，与流规则条件字段的对应关系如下
dl_type=0x0800 <=> ip
dl_type=0x0806 <=> arp
dl_type=0x0800，nw_proto=1 <=> icmp
dl_type=0x0800，nw_proto=6 <=> tcp
dl_type=0x0800，nw_proto=17 <=> udp

数据包重定向
将交换机中所有的 icmp 协议包（由 dl_type=0x0800,nw_proto=1 指定）全部转发到 4 号端口，包括 4 号端口自己发出的 icmp 包，该流规则将导致由 Open vSwitch 管理的主机间以及与外部网络间都将访问 ping 通，但可以使用万维网、FTP 等服务。

```
ovs-ofctl add-flow br0
idle_timeout=0,dl_type=0x0800,nw_proto=1,actions=output:4
```

去除 VLAN tag

去除从 3 号端口发来的所有 VLAN 数据包中的 tag，然后转发

```
ovs-ofctl add-flow br0 idle_timeout=0,in_port=3,actions=strip_vlan,normal
```

更改数据包源 IP 地址后转发

将从 3 号端口收到的所有 IP 包的源 IP 字段更改为 192.168.28.225

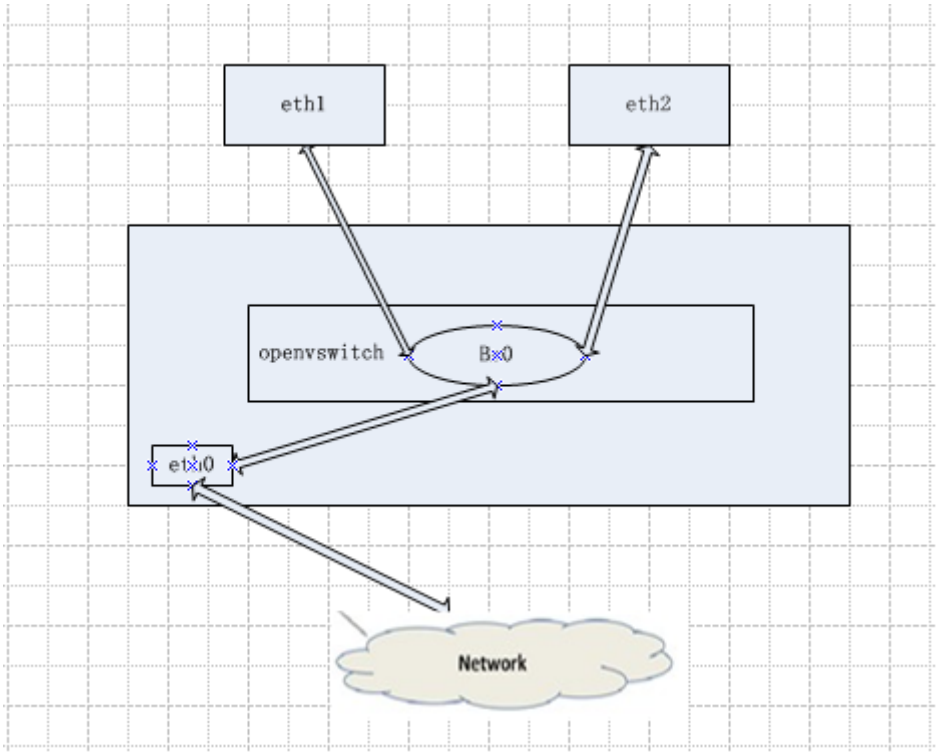
```
ovs-ofctl add-flow br0
idle_timeout=0,in_port=3,actions=mod_nw_src:192.168.28.225,normal
```

6.4 综合应用

6.4.1 构建简单的交换机

将一台有五个网卡的主机做成交换机，然后接两台一个网卡的主机到交换机上进行上网测试。

结果图如下



说明：

eth1 后面接一台主机，IP：192.168.28.43

eth2 后面接一台主机，IP：192.168.28.209

eth0、eth1、eth2 都属于一台主机上的网卡

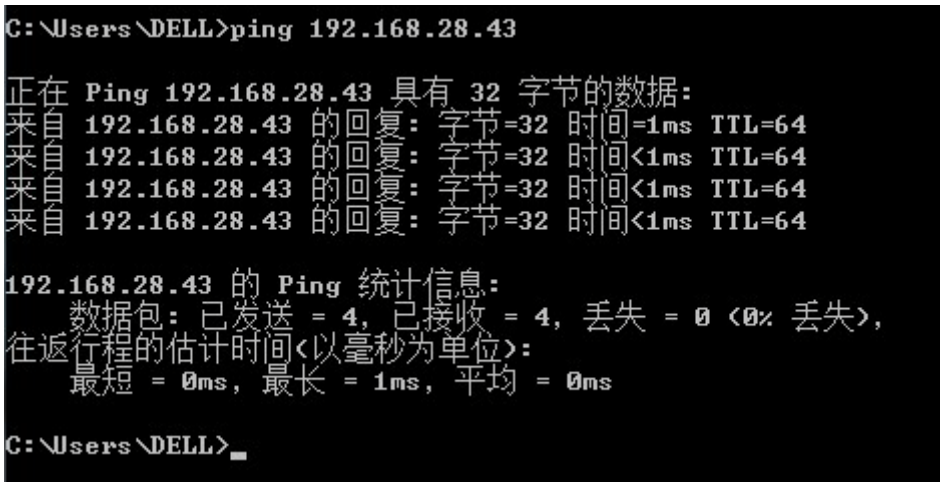
执行命令

```
root@ubuntu229:~# ovs-vsctl add-br br0 //建立一個 OpenvSwitch 的 Bridge

root@ubuntu229:~# ovs-vsctl add-port br0 eth0 //eth0 加到 br0 中

root@ubuntu229:~# ovs-vsctl add-port br0 eth1 //eth1 加到 br0

root@ubuntu229:~# ovs-vsctl add-port br0 eth2  // eth2 加到 br0
```

验证：

192.168.28.43ping 192.168.28.209

192.168.28.209ping 192.168.28.43

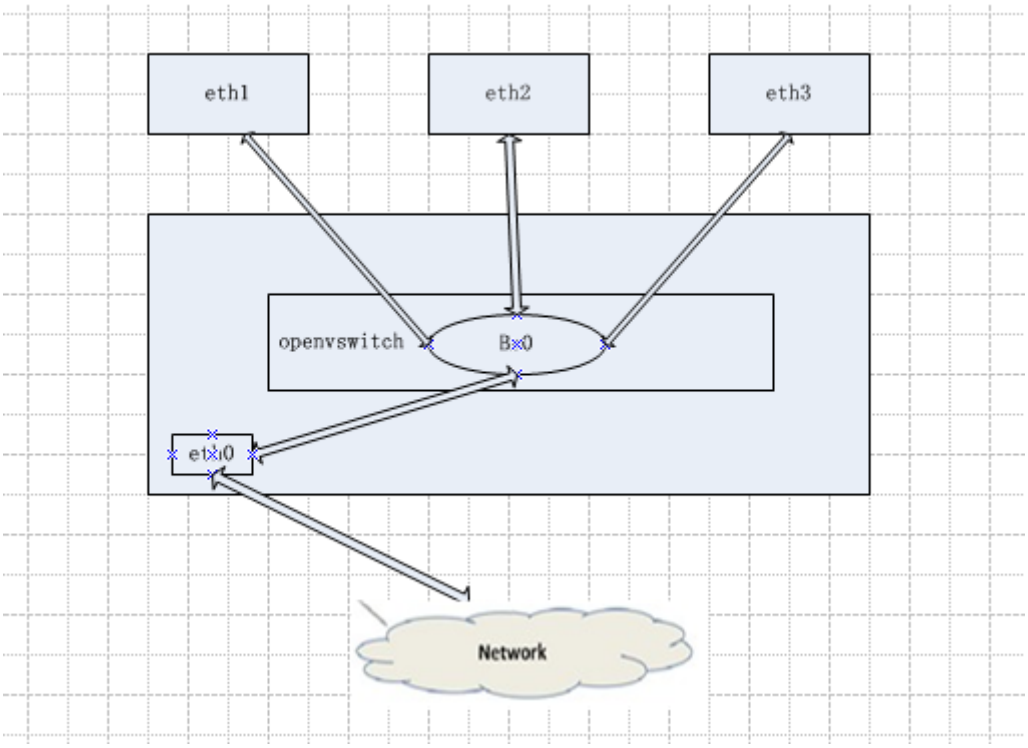


可见两台机器能够相互 ping 通，进一步验证 eth1 和 eth2 后面的主机能够正常连上网络。

6.4.2 交换机 VLAN 划分

这里将同一个网桥的不同网卡分配到不同的 VLAN 上

结构图：



VLAN 分配情况：
VLAN1：eth1，eth3
VLAN2：eth2

执行命令

```
root@ubuntu229:~# ovs-vsctl add-br br0 //建立一個 OpenvSwitch 的 Bridge

root@ubuntu229:~# ovs-vsctl add-port br0 eth0 //eth0 加到 br0 中

root@ubuntu229:~# ovs-vsctl add-port br0 eth1 tag=1 //eth1 加到 VLAN1

root@ubuntu229:~# ovs-vsctl add-port br0 eth2 tag=2 // eth2 加到 VLAN2

root@ubuntu229:~# ovs-vsctl add-port br0 eth3 tag=1 // eth3 加到 VLAN1

root@ubuntu229:~# ovs-vsctl show

131648b5-f7a6-4949-9a39-273ed62c0922

    Bridge "br0"

        Port "br0"

            Interface "br0"

                type: internal
```

```
        Port "eth0"
            Interface "eth0"
        Port "eth2"
            tag: 2
            Interface "eth2"
        Port "eth1"
            tag: 1
            Interface "eth1"
        Port "eth3"
            tag: 1
            Interface "eth3"
```

注：
tag vlan 基于 IEEE 802.1Q（vlan 标准），用 VID（vlan id）来划分不同的 VLAN。

验证
用 192.168.28.43ping192.168.28.209

用 192.168.28.209 ping 192.168.28.43，也 ping 不通
除此之外，发现两台机器也不能上网，不能上网的原因可以是内核不支持。

进一步验证，把 eth1 和 eth2 加到同一个 VLAN，如下

```
root@ubuntu229:~# ovs-vsctl add-port br0 eth1 tag=1 //eth1 加到 VLAN1
```

```
root@ubuntu229:~# ovs-vsctl add-port br0 eth2 tag=1 // eth2 加到 VLAN
```

相互 Ping 的结果如下

需要注意的是，连接在 eth1 和 eth2 的两台机器仍然不能上网。

结论：

VLAN 技术是基于二层和三层之间的隔离，可以将不同的网络用户与网络资源进行分组并通过支持 VLAN 技术的交换机隔离不同组内网络设备间的数据交换来达到网络安全的目的。同一个 VALN 上的机器可以相互通信，不同的 VLAN 之前不可以通信，因为它们之间在数据链路层上是断开的，只能通过三层路由器才能访问。

6.4.3 ovs-controller 测试

这里使用 OpenvSwitch 自带的控制器做一些操作

部署情况：

主机 192.168.28.229 做 ovs-controller

虚拟机 192.168.1.156 做 vswitch 受 192.168.28.229 控制

6.4.3.1 TCP 方式

首先在 192.168.1.156 上执行
建立控制器

```
root@bogon:~# ovs-vsctl add-br br0

root@bogon:~# ovs-vsctl set-controller br0 tcp:192.168.28.229:6633
```

查看控制器

```
root@bogon:~# ovs-vsctl get-controller br0

tcp:192.168.28.229:6633
```

查看所有

```
root@bogon:~# ovs-vsctl show

05d9e9e9-cb3b-4e7d-900e-e85926e6b7d3

    Bridge "br0"

        Controller "tcp:192.168.28.229:6633"

            is_connected: true

        Port "br0"

            Interface "br0"

                type: internal
```

然后在 192.168.28.229 上执行

```
root@ubuntu229:~# ovs-controller ptcp:6633:192.168.28.229
```

最后在 192.168.1.156 上执行

```
root@bogon:~# ovs-controller tcp:192.168.28.229:6633
```

这样控制器和 vswitch 就连接成功了
验证如下

```
root@ubuntu229:~# netstat -npt |grep 6633
tcp        0      0 192.168.28.229:6633     192.168.28.1:20536      ESTABLISHED 17765/ovs-contro
lle
tcp        0      0 192.168.28.229:6633     192.168.28.1:20533      ESTABLISHED 17765/ovs-contro
lle
```

```
root@bogon:~# netstat -npt |grep 6633
tcp        0      0 192.168.1.156:40765     192.168.28.229:6633     ESTABLISHED 6199/ovs-control
ler
tcp        0      0 192.168.1.156:40764     192.168.28.229:6633     ESTABLISHED 5608/ovs-vswitch
d
```

可见 192.168.28.229 和 192.168.1.156 连接起来了，仔细查看，他们之间建立了两个连接，一个是 229 的 controller 和 156 的 vswitch，还有一个是 229 的 controller 和 156 的 controller。还有一点就是在 229 上查看，192.168.1.156 的连接名是 192.168.28.1，是因为 192.168.28.1 是无线路由的 IP，而它 192.168.1.156 是无线路由器分配的 IP，采用方式是 NAT。

6.4.3.1 SSL 方式

当 ovs-vswitchd 被配置为通过 SSL 进行连接管理器或控制器连接，下面的参数是必需的：

private-key 私有密钥
指定 PEM 文件，其中包含的私有密钥用于 SSL 连接到控制器的虚拟交换机的身份。

certificate 证书
指定一个的 PEM 文件包含一个证书签名的证书颁发机构（CA）的使用，证明了虚拟交换机的私有密钥，确定一个值得信赖的开关控制器和管理者。

ca-certCA-证书
指定 PEM 文件，其中包含的 CA 证书用于验证的虚拟交换机连接到一个值得信赖的控制器

6.5 代码测试

这里对安装包文件~/openvswitch-1.7.1/tests 目录下的部分代码进行简单的了解。

6.5.1 test-uuid

test-uuid.c 主要产生 uuid 或者检查 uuid 是否正确，创建一个设备或者获取一个设备的

uuid，都会调用和它类似的函数，根据后面的参数个数来决定产生 uuid 还是检查 uuid。
uuid 格式如下
＊＊＊＊＊＊＊＊-＊＊＊＊-＊＊＊＊-＊＊＊＊-＊＊＊＊＊＊＊＊＊＊＊＊
总共 32 位，如：1e08ebdf-2005-4fc5-826a-5fe16acc0acc

6.5.2 test-stp

STP — Spanning Tree Protocol（生成树协议）逻辑上断开环路，防止二层网络的广播风暴的产生当线路出现故障，断开的接口被激活，恢复通信，起备份线路的作用。

STP 的原理
通过阻断冗余链路，使一个有回路的桥接网络修剪成一个无回路的树形拓扑结构。

STP 的算法，即 STP 将一个环形网络生成无环拓朴的步骤：
选择根网桥（Root Bridge）
选择根端口（Root Ports）
选择指定端口（Designated Ports）

依据分别如下：
根据网桥 ID；
在非根网桥上选择一个到根网桥最近的端口作为根端口，依据是根路径成本最低，直连（上游）的网桥 ID 最小，端口（上游）ID 最小；
由 STP 计算，在每个网段选择 1 个指定端口（DP）；

6.5.3 test-netflow

网络流量采集器测试工具
用法：test-netflow [OPTIONS] PORT[:IP]
监听 UDP 端口，IP 可选。

6.5.3 test-json

读取文件中的 josn 信息，执行文件后面跟文件名，文件里面要有相应的 josn 格式信息，如果 json 格式不对，执行会报错，只有在文件存在并且里面的值是 josn 格式的时候，才会打印出文件里面的 json 值。