jiashanshan521的博客

个人资料

jiashanshan521

原创	粉丝	喜欢	评论
3	3	1	4

等级： 博客 2　　访问量： 1万+

积分： 203　　排名： 36万+

文章搜索

文章分类

计算机 (2)

文章存档

2017年4月 (1)
2016年10月 (2)
2016年9月 (1)
2016年8月 (5)
2016年5月 (2)

展开

阅读排行

OVS初级教程：使用open vs... (3631)
navicat 如何设置外键 (2714)
整理的最完整的OFPT_STATS... (1813)
基于floodlight开发SDN应用... (1793)
navicat连接mysql (1204)
java用NLPIR对本地txt进行分... (1117)
基于 Open vSwitch 的 Open... (838)

基于 Open vSwitch 的 OpenFlow 实践

2016年08月22日 14:30:56　　844人阅读　　评论(0)　　收藏　　举报

目录(?) [+]

基于 Open vSwitch 的 OpenFlow 实践

Open vSwitch 是运行在虚拟化平台上的虚拟交换机，同时也提供了对 OpenFlow 协议的支持。本文介绍了 Open vSwitch 的基础概念，并举例说明如何使用 Open vSwitch 自带的工具创建 OpenFlow 规则，最后演示了如何使用 Floodlight 连接并管理 Open vSwitch。

3 评论

赵 祎, 软件工程师, IBM
罗 俊, 软件工程师, IBM
陈 玺, 软件工程师, IBM

2014 年 1 月 13 日

内容

Open vSwitch 概述

Open vSwitch（下面简称为 OVS）是由 Nicira Networks 主导的，运行在虚拟化平台（例如 KVM，Xen）上的虚拟交换机。在虚拟化平台上，OVS 可以为动态变化的端点提供 2 层交换功能，很好的控制虚拟网络中的访问策略、网络隔离、流量监控等等。

OVS 遵循 Apache 2.0 许可证，能同时支持多种标准的管理接口和协议。OVS 也提供了对 OpenFlow 协议的支持，用户可以使用任何支持 OpenFlow 协议的控制器对 OVS 进行远程管理控制。

Open vSwitch 概述

在 OVS 中，有几个非常重要的概念：

- Bridge: Bridge 代表一个以太网交换机（Switch），一个主机中可以创建一个或者多个 Bridge 设备。
- Port: 端口与物理交换机的端口概念类似，每个 Port 都隶属于一个 Bridge。
- Interface: 连接到 Port 的网络接口设备。在通常情况下，Port 和 Interface 是一对一的关系，只有在配置 Port 为 bond 模式后，Port 和 Interface 是一对多的关系。
- Controller: OpenFlow 控制器。OVS 可以同时接受一个或者多个 OpenFlow 控制器的管理。
- datapath: 在 OVS 中，datapath 负责执行数据交换，也就是把从接收端口收到的数据包在流表中进行匹配，并执行匹配到的动作。
- Flow table: 每个 datapath 都和一个“flow table”关联，当 datapath 接收到数据之后， OVS 会在 flow table 中查找可以匹配的 flow，执行对应的操作，例如转发数据到另外的端口。

Open vSwitch 实验环境配置

OVS 可以安装在主流的 Linux 操作系统中，用户可以选择直接安装编译好的软件包，或者下载源码进行编译安装。

在我们的实验环境中，使用的操作系统是 64 位 Ubuntu Server 12.04.3 LTS，并通过源码编译的方式安装了 Open vSwitch 1.11.0

Scapy (681)
如何升级Mininet的Open vS... (609)
把jar文件放在一个文件夹里 (523)

最新评论

整理的最完整的OFPT_STATS...
he11onate ：楼主你好，向您请教2个问题，第一：控制器如何向交换机发送stats_request消息呢？第二：又...

基于floodlight开发SDN...
pengcong0201 ：您好，请问下您这个例子用的是floodlight哪个版本呢？谢谢！

基于floodlight开发SDN...
qq_27950951 ：nice

java用NLPIR对本地txt进...
dlrousi ：关键词提取结果是： ??? 8??????????????#??????????????????#...

```
$ lsb_release -a
No LSB modules are available.
Distributor ID:Ubuntu
Description:Ubuntu 12.04.3 LTS
Release:12.04
Codename:precise
```

OVS 的源码编译安装方式可以参考官方文档 How to Install Open vSwitch on Linux, FreeBSD and NetBSD。

安装完毕后，检查 OVS 的运行情况：

```
$ ps -ea | grep ovs
12533 ?        00:00:00 ovs_workq
12549 ?        00:00:04 ovsdb-server
12565 ?        00:00:48 ovs-vswitchd
12566 ?        00:00:00 ovs-vswitchd
```

查看 OVS 的版本信息, 我们安装版本的是 1.11.0

```
$ ovs-appctl --version
ovs-appctl (Open vSwitch) 1.11.0
Compiled Oct 28 2013 14:17:16
```

查看 OVS 支持的 OpenFlow 协议的版本

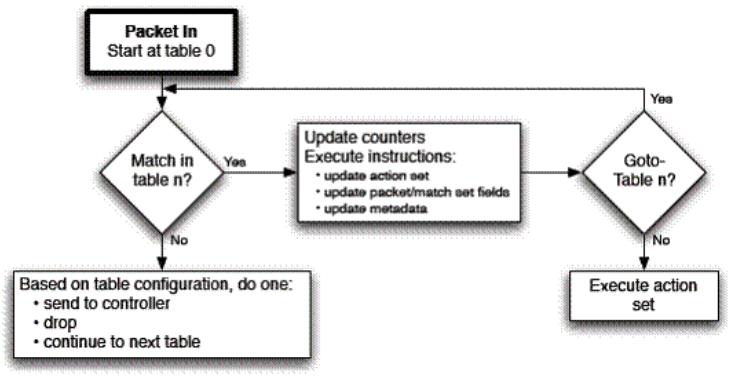
```
$ ovs-ofctl --version
ovs-ofctl (Open vSwitch) 1.11.0
Compiled Oct 28 2013 14:17:17
OpenFlow versions 0x1:0x4
```

回页首

基于 Open vSwitch 的 OpenFlow 实践

OpenFlow 是用于管理交换机流表的协议，ovs-ofctl 则是 OVS 提供的命令行工具。在没有配置 OpenFlow 控制器的模式下，用户可以使用 ovs-ofctl 命令通过 OpenFlow 协议去连接 OVS，创建、修改或删除 OVS 中的流表项，并对 OVS 的运行状况进行动态监控。

图 1. OpenFlow 的匹配流程



Flow 语法说明

在 OpenFlow 的白皮书中，Flow 被定义为某个特定的网络流量。例如，一个 TCP 连接就是一个 Flow，或者从某个 IP 地址发出来的数据包，都可以被认为是一个 Flow。支持 OpenFlow 协议的交换机应该包括一个或者多个流表，流表中的条目包含：数据包头的信息、匹配成功后要执行的指令和统计信息。

当数据包进入 OVS 后，会将数据包和流表中的流表项进行匹配，如果发现了匹配的流表项，则执行该流表项中的指令集。相反，如果数据包在流表中没有发现任何匹配，OVS 会通过控制通道把数据包发到 OpenFlow 控制器中。

在 OVS 中，流表项作为 ovs-ofctl 的参数，采用如下的格式：字段=值。如果有多个字段，可以用逗号或者空格分开。一些常用的字段列举如下：

表 1. 流表常用字段

字段名称	说明
in_port=port	传递数据包的端口的 OpenFlow 端口编号

字段名称	说明
dl_vlan=vlan	数据包的 VLAN Tag 值，范围是 0-4095，0xffff 代表不包含 VLAN Tag 的数据包
dl_src=<MAC> dl_dst=<MAC>	匹配源或者目标的 MAC 地址 01:00:00:00:00:00/01:00:00:00:00:00 代表广播地址 00:00:00:00:00:00/01:00:00:00:00:00 代表单播地址
dl_type=ethertype	匹配以太网协议类型，其中： dl_type=0x0800 代表 IPv4 协议 dl_type=0x086dd 代表 IPv6 协议 dl_type=0x0806 代表 ARP 协议 完整的的类型列表可以参见以太网协议类型列表
nw_src=ip[/netmask] nw_dst=ip[/netmask]	当 dl_typ=0x0800 时，匹配源或者目标的 IPv4 地址，可以使 IP 地址或者域名
nw_proto=proto	和 dl_type 字段协同使用。 当 dl_type=0x0800 时，匹配 IP 协议编号 当 dl_type=0x086dd 代表 IPv6 协议编号 完整的 IP 协议编号可以参见IP 协议编号列表
table=number	指定要使用的流表的编号，范围是 0-254。在不指定的情况下，默认值为 0。通过使用流表编号，可以创建或者修改多个 Table 中的 Flow
reg<idx>=value[/mask]	交换机中的寄存器的值。当一个数据包进入交换机时，所有的寄存器都被清零，用户可以通过 Action 的指令修改寄存器中的值

对于 add-flow，add-flows 和 mod-flows 这三个命令，还需要指定要执行的动作：actions=[target][,target...]

一个流规则中可能有多个动作，按照指定的先后顺序执行。

常见的操作有：

- output:port: 输出数据包到指定的端口。port 是指端口的 OpenFlow 端口编号
- mod_vlan_vid: 修改数据包中的 VLAN tag
- strip_vlan: 移除数据包中的 VLAN tag
- mod_dl_src/ mod_dl_dest: 修改源或者目标的 MAC 地址信息
- mod_nw_src/mod_nw_dst: 修改源或者目标的 IPv4 地址信息
- resubmit:port: 替换流表的 in_port 字段，并重新进行匹配
- load:value->dst[start..end]: 写数据到指定的字段

实践操作 **OpenFlow** 命令

在本例中, 我们会创建一个不连接到任何控制器的 OVS 交换机，并演示如何使用 ovs-octl 命令操作 OpenFlow 流表。

创建一个新的 OVS 交换机

```
$ ovs-vsctl add-br ovs-switch
```

创建一个端口 p0，设置端口 p0 的 OpenFlow 端口编号为 100（如果在创建端口的时候没有指定 OpenFlow 端口编号，OVS 会自动生成一个）。

```
$ ovs-vsctl add-port ovs-switch p0 -- set Interface p0 ofport_request=100
```

设置网络接口设备的类型为“internal”。对于 internal 类型的的网络接口，OVS 会同时在 Linux 系统中创建一个可以用来收发数据的模拟网络设备。我们可以为这个网络设备配置 IP 地址、进行数据监听等等。

```
$ ovs-vsctl set Interface p0 type=internal
$ ethtool -i p0
driver: openvswitch
version:
firmware-version:
bus-info:
```

```
supports-statistics: no
supports-test: no
supports-eeprom-access: no
supports-register-dump: no
```

为了避免网络接口上的地址和本机已有网络地址冲突，我们可以创建一个虚拟网络空间 ns0，把 p0 接口移入网络空间 ns0，并配置 IP 地址为 192.168.1.100

```
$ ip netns add ns0
$ ip link set p0 netns ns0
$ ip netns exec ns0 ip addr add 192.168.1.100/24 dev p0
$ ip netns exec ns0 ifconfig p0 promisc up
```

使用同样的方法创建端口 p1、p2

表 **2.** 创建的端口信息

端口	说明
p0	IP 地址: 192.168.1.100/24 网络名称空间: ns0 网络接口 MAC 地址: 66:4e:cc:ae:4d:20 OpenFlow Port Number: 100
p1	IP 地址: 192.168.1.101/24 网络名称空间: ns1 网络接口 MAC 地址: 46:54:8a:95:dd:f8 OpenFlow Port Number: 101
p2	IP 地址: 192.168.1.102/24, 网络名称空间: ns2 网络接口 MAC 地址: 86:3b:c8:d0:44:10 OpenFlow Port Number: 102

创建所有的端口之后， 查看 OVS 交换机的信息

```
$ ovs-vsctl show
30282710-d401-4187-8e13-52388f693df7
    Bridge ovs-switch
        Port "p0"
            Interface "p0"
                type: internal
        Port "p2"
            Interface "p2"
                type: internal
        Port "p1"
            Interface "p1"
                type: internal
        Port ovs-switch
            Interface ovs-switch
                type: internal
```

使用 ovs-ofctl 创建并测试 OpenFlow 命令

1. 查看 Open vSwitch 中的端口信息。从输出结果中，可以获得交换机对应的 datapath ID （dpid），以及每个端口的 OpenFlow 端口编号，端口名称，当前状态等等。

```
$ ovs-ofctl show ovs-switch
OFPT_FEATURES_REPLY (xid=0x2): dpid:00001232a237ea45
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
 100(p0): addr:54:01:00:00:00:00
     config:     PORT_DOWN
     state:      LINK_DOWN
     speed: 0 Mbps now, 0 Mbps max
 101(p1): addr:54:01:00:00:00:00
     config:     PORT_DOWN
     state:      LINK_DOWN
     speed: 0 Mbps now, 0 Mbps max
 102(p2): addr:54:01:00:00:00:00
     config:     PORT_DOWN
     state:      LINK_DOWN
     speed: 0 Mbps now, 0 Mbps max
 LOCAL(ovs-switch): addr:12:32:a2:37:ea:45
     config:     0
```

```
     state:      0
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
```

如果想获得网络接口的 OpenFlow 编号，也可以在 OVS 的数据库中查询

```
$ ovs-vsctl get Interface p0 ofport
100
```

查看 datapath 的信息

```
$ ovs-dpctl show
system@ovs-system:
lookups: hit:12173 missed:712 lost:0
flows: 0
port 0: ovs-system (internal)
port 1: ovs-switch (internal)
port 2: p0 (internal)
port 3: p1 (internal)
port 4: p2 (internal)
```

2. 屏蔽数据包

   屏蔽所有进入 OVS 的以太网广播数据包

```
$ ovs-ofctl add-flow ovs-switch "table=0, dl_src=01:00:00:00:00:00/01:00:00:00:00:00, actio
ns=drop"
```

   屏蔽 STP 协议的广播数据包

```
$ ovs-ofctl add-flow ovs-switch "table=0, dl_dst=01:80:c2:00:00:00/ff:ff:ff:ff:ff:f0, actio
ns=drop"
```

3. 修改数据包

   添加新的 OpenFlow 条目，修改从端口 p0 收到的数据包的源地址为 9.181.137.1

```
$ ovs-ofctl add-flow ovs-switch "priority=1 idle_timeout=0,\
    in_port=100,actions=mod_nw_src:9.181.137.1,normal"
```

   从端口 p0（192.168.1.100）发送测试数据到端口 p1（192.168.1.101）

```
$ ip netns exec ns0 ping 192.168.1.101
```

   在接收端口 p1 监控数据，发现接收到的数据包的来源已经被修改为 9.181.137.1

```
$ ip netns exec ns1 tcpdump -i p1 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on p1, link-type EN10MB (Ethernet), capture size 65535 bytes
15:59:16.885770 IP 9.181.137.1 > 192.168.1.101: ICMP echo request, id 23111, seq 457, lengt
h 64
15:59:17.893809 IP 9.181.137.1 > 192.168.1.101: ICMP echo request, id 23111, seq 458, lengt
h 64
```

4. 重定向数据包

   添加新的 OpenFlow 条目，重定向所有的 ICMP 数据包到端口 p2

```
$ ovs-ofctl add-flow ovs-switch idle_timeout=0,dl_type=0x0800,nw_proto=1,actions=output:102
```

   从端口 p0 （192.168.1.100）发送数据到端口 p1（192.168.1.101）

```
$ ip netns exec ns0 ping 192.168.1.101
```

   在端口 p2 上监控数据，发现数据包已被转发到端口 p2

```
$ ip netns exec ns3 tcpdump -i p2 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on p2, link-type EN10MB (Ethernet), capture size 65535 bytes
16:07:35.677770 IP 192.168.1.100 > 192.168.1.101: ICMP echo request, id 23147, seq 25, leng
th 64
16:07:36.685824 IP 192.168.1.100 > 192.168.1.101: ICMP echo request, id 23147, seq 26, leng
th 64
```

5. 修改数据包的 VLAN Tag

   除了使用“ping”、“tcpdump”和“iperf” 等 Linux 命令以外，我们也可以使用 OVS 提供的 ovs-appctl ofproto/trace 工具来测试 OVS 对数据包的转发状况。ovs-appctl ofproto/trace 可以用来生成测试用的模拟数据包，并一步步的展示 OVS 对数据包的流处理过程。在以下的例子中，我们演示一下如何使用这个命令：

   修改端口 p1 的 VLAN tag 为 101，使端口 p1 成为一个隶属于 VLAN 101 的端口

```
$ ovs-vsctl set Port p1 tag=101
```

   现在由于端口 p0 和 p1 属于不同的 VLAN，它们之间无法进行数据交换。我们使用 ovs-appctl ofproto/trace 生成一个从端口 p0 发送到端口 p1 的数据包，这个数据包不包含任何 VLAN tag，并观察 OVS 的处理过程

```
$ ovs-appctl ofproto/trace ovs-switch in_port=100,dl_src=66:4e:cc:ae:4d:20,
dl_dst=46:54:8a:95:dd:f8 -generate
Flow:metadata=0,in_port=100,vlan_tci=0x0000,dl_src=66:4e:cc:ae:4d:20,
 dl_dst=46:54:8a:95:dd:f8,dl_type=0x0000
Rule: table=0 cookie=0 priority=0
OpenFlow actions=NORMAL
no learned MAC for destination, flooding

Final flow: unchanged
Relevant fields: skb_priority=0,in_port=100,vlan_tci=0x0000/0x1fff,\
```

```
dl_src=66:4e:cc:ae:4d:20,dl_dst=46:54:8a:95:dd:f8,dl_type=0x0000,nw_frag=no
Datapath actions: 4,1
```

在第一行输出中，“Flow:”之后的字段描述了输入的流的信息。由于我们没有指定太多信息，所以多数字段 （例如 dl_type 和 vlan_tci）被 OVS 设置为空值。

在第二行的输出中，“Rule:” 之后的字段描述了匹配成功的流表项。

在第三行的输出中，“OpenFlow actions”之后的字段描述了实际执行的操作。

最后一段以”Final flow”开始的字段是整个处理过程的总结，“Datapath actions: 4,1”代表数据包被发送到 datapath 的 4 和 1 号端口。

创建一条新的 Flow：对于从端口 p0 进入交换机的数据包，如果它不包含任何 VLAN tag，则自动为它添加 VLAN tag 101

```
$ ovs-ofctl add-flow ovs-switch "priority=3,in_port=100,dl_vlan=0xffff,\
actions=mod_vlan_vid:101,normal"
```

再次尝试从端口 p0 发送一个不包含任何 VLAN tag 的数据包，发现数据包进入端口 p0 之后, 会被加上 VLAN tag101, 同时转发到端口 p1 上

```
$ ovs-appctl ofproto/trace ovs-switch in_port=100,dl_src=66:4e:cc:ae:4d:20,
dl_dst=46:54:8a:95:dd:f8 -generate
Flow: metadata=0,in_port=100,vlan_tci=0x0000,dl_src=66:4e:cc:ae:4d:20,
dl_dst=46:54:8a:95:dd:f8,dl_type=0x0000
Rule: table=0 cookie=0 priority=3,in_port=100,vlan_tci=0x0000
OpenFlow actions=mod_vlan_vid:101,NORMAL
forwarding to learned port

Final flow: metadata=0,in_port=100,dl_vlan=101,dl_vlan_pcp=0,dl_src=66:4e:cc:ae:4d:20,
dl_dst=46:54:8a:95:dd:f8,dl_type=0x0000
Relevant fields: skb_priority=0,in_port=100,vlan_tci=0x0000/0xffff,dl_src=66:4e:cc:ae:4d:2
0,
dl_dst=46:54:8a:95:dd:f8,dl_type=0x0000,nw_frag=no
Datapath actions: 3
```

反过来从端口 p1 发送数据包，由于 p1 现在是带有 VLAN tag 101 的 Access 类型的端口，所以数据包进入端口 p1 之后，会被 OVS 添加 VLAN tag 101 并发送到端口 p0

```
$ ovs-appctl ofproto/trace ovs-switch in_port=101,dl_dst=66:4e:cc:ae:4d:20,
dl_src=46:54:8a:95:dd:f8 -generate
Flow: metadata=0,in_port=101,vlan_tci=0x0000,dl_src=46:54:8a:95:dd:f8,
dl_dst=66:4e:cc:ae:4d:20,dl_type=0x0000
Rule: table=0 cookie=0 priority=0
OpenFlow actions=NORMAL
forwarding to learned port

Final flow: unchanged
Relevant fields: skb_priority=0,in_port=101,vlan_tci=0x0000,dl_src=46:54:8a:95:dd:f8,
dl_dst=66:4e:cc:ae:4d:20,dl_type=0x0000,nw_frag=no
Datapath actions: push_vlan(vid=101,pcp=0),2
```

6. 其他 OpenFlow 常用的操作

   查看交换机中的所有 Table

   ```
   ovs-ofctl dump-tables ovs-switch
   ```

   查看交换机中的所有流表项

   ```
   ovs-ofctl dump-flows ovs-switch
   ```

   删除编号为 100 的端口上的所有流表项

   ```
   ovs-ofctl del-flows ovs-switch "in_port=100"
   ```

   查看交换机上的端口信息

   ```
   ovs-ofctl show ovs-switch
   ```

回页首

通过 **Floodlight** 管理 **OVS**

一方面，OpenFlow 控制器可以通过 OpenFlow 协议连接到任何支持 OpenFlow 的交换机，控制器通过和交换机交换流表规则来控制数据流向。另一方面， OpenFlow 控制器向用户提供的界面或者接口，用户可以通过界面对网络架构进行动态的修改，修改交换机的流表规则等等。Floodlight 是一个基于 Apache 协议，使用 Java 开发的企业级 OpenFlow 控制器。我们在下面的例子中演示如何安装 Floodlight，并连接管理 OVS 的过程。

Floodlight 的安装过程非常简单，在另外一台机器上, 下载 Floodlight 源码并编译

```
$ git clone git://github.com/floodlight/floodlight.git
$ cd floodlight/
$ ant
$ java -jar target/floodlight.jar
```

运行 Floodlight

```
$ java -jar floodlight.jar
```

在安装了 OVS 交换机的节点上，配置 OVS 交换机 ovs-switch，使用 Floodlight 作为控制器。默认情况下，Floodlight 在端口 6633 上进行监听，我们使用 ovs-vsctl 命令配置 OVS 交换机使用 TCP 协议连接到 Floodlight（IP 地址为 9.181.137.182，端口号 6633）。对于一个 OVS 交换机来说，可以同时配置一个或者多个控制器

$ ovs-vsctl set-controller ovs-switch tcp:9.181.137.182:6633

当 OVS 交换机连接到 Floodlight 控制器后，理论上所有的流表规则应该交给控制器来建立。由于 OVS 交换机和控制器之间是通过网络通讯来传递数据的，所以网络连接失败会影响到 Flow 的建立。针对这种情况，OVS 提供了两种处理模式：

- standlone: 默认模式。如果 OVS 交换机超过三次无法正常连接到 OpenFlow 控制器，OVS 交换机自己会负责建立流表。在这种模式下，OVS 和常见的 L2 交换机相似。与此同时，OVS 也会继续尝试连接控制器，一旦网络连接恢复，OVS 会再次切换到使用控制器进行流表管理。
- secure: 在 secure 模式下，如果 OVS 无法正常连接到 OpenFlow 控制器，OVS 会不停的尝试与控制器重新建立连接，而不会自己负责建立流表。

设置 OVS 的连接模式为 secure 模式

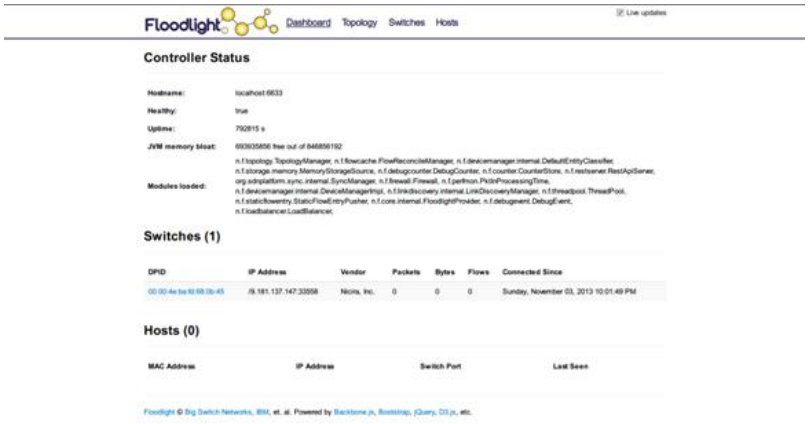
```
$ ovs-vsctl set Bridge ovs-switch fail-mode=secure
```

查看 OVS 的状态，“is_connected:true”代表 OVS 已经成功连接到了 Floodlight

```
$ ovs-vsctl show
30282710-d401-4187-8e13-52388f693df7
    Bridge ovs-switch
        Controller "tcp:9.181.137.182:6633"
            is_connected: true
        Port ovs-switch
            Interface ovs-switch
                type: internal
        Port "p0"
            Interface "p0"
                type: internal
        Port "p1"
            tag: 101
            Interface "p1"
                type: internal
        Port "p2"
            Interface "p2"
                type: internal
```

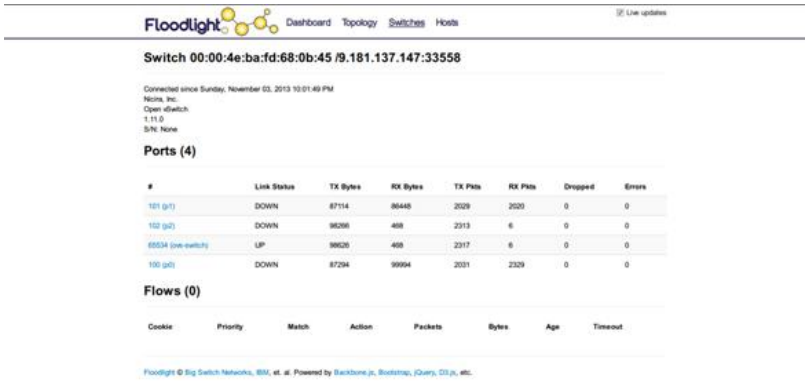
通过访问 Floodlight 提供的 Web 管理界面 http://<Host Address>:8080/ui/index.html，我们可以查看 Floodlight 控制器的状态以及所有连接到 Floodlight 的交换机列表

图 **2. Floodlight** 主界面



选中某个 OpenFlow 交换机， 查看其中的端口列表和流表信息

图 **3.** 查看 **OpenFlow** 交换机的详细信息



通过 Floodlight 的 RESTAPI，添加两条新的规则让端口 p0 和 p1 可以相互通讯。注意：替换命令行中的 switch 的 ID 为交换机的 datapath ID

```
curl -d '{"switch": "00:00:0e:f9:05:6b:7c:44", "name":"my-flow1", "cookie":"0","priority":"32768",
"ingress-port":"100","active":"true", "actions":"output=flood"}'
http://9.181.137.182:8080/wm/staticflowentrypusher/json

curl -d '{"switch": "00:00:0e:f9:05:6b:7c:44", "name":"my-flow2", "cookie":"0","priority":"32768",
"ingress-port":"101","active":"true", "actions":"output=flood"}'
http://9.181.137.182:8080/wm/staticflowentrypusher/json
```

验证是否能从端口 p0 发送数据包到 p1

```
$ ip netns exec ns0 ping -c4 192.168.1.101
PING 192.168.1.101 (192.168.1.101) 56(84) bytes of data.
64 bytes from 192.168.1.101: icmp_req=1 ttl=64 time=0.027 ms
64 bytes from 192.168.1.101: icmp_req=2 ttl=64 time=0.018 ms
64 bytes from 192.168.1.101: icmp_req=3 ttl=64 time=0.023 ms
64 bytes from 192.168.1.101: icmp_req=4 ttl=64 time=0.022 ms

--- 192.168.1.101 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2998ms
rtt min/avg/max/mdev = 0.018/0.022/0.027/0.005 ms
```

在 OVS 端也可以看到，流表规则已经被 OVS 同步到本地。

```
$ ovs-ofctl dump-flows ovs-switch
NXST_FLOW reply (xid=0x4):
 cookie=0xa0000000000000, duration=335.122s, table=0, n_packets=347, n_bytes=28070,
  idle_age=1, in_port=100 actions=FLOOD
 cookie=0xa0000000000000, duration=239.892s, table=0, n_packets=252, n_bytes=24080,
  idle_age=0, in_port=101 actions=FLOOD
```

通过 Floodlight 的 RestAPI，查看交换机上的流表规则

```
curl http://9.181.137.182:8080/wm/staticflowentrypusher/list/00:00:0e:f9:05:6b:7c:44/json
```

通过 Floodlight 的 RestAPI，删除交换机上的流表规则

```
curl http://9.181.137.182:8080/wm/staticflowentrypusher/clear/00:00:0e:f9:05:6b:7c:44/json
```

回页首

总结

通过本文的讲述和实验，我们了解了 Open vSwitch 以及 OpenFlow 的基本概念，以及通过 OpenFlow 协议修改 Open vSwitch 中的流表项，最后演示了如何使用 Floodlight 连接 Open vSwitch 并进行管理。

参考资料

学习

- 参考 Open vSwitch 项目主页，查看 Open vSwitch 的最新信息以及下载源码。
- 查看文章 “Open vSwitch Advanced Features Tutorial”，了解更多的 Open vSwitch 使用技巧。
- Floodlight 是 Floodlight 项目的主页，里面详细介绍了 Floodlight 的最新信息以及开发使用的 RestAPI。
- OpenFlow 是 OpenFlow 协议的主页，里面提供了 OpenFlow 协议的白皮书以及 OpenFlow 交换机的实现规范。
- developerWorks 云计算站点 提供了有关云计算的更新资源，包括
  - 云计算 简介。
  - 更新的 技术文章和教程，以及网络广播，让您的开发变得轻松，专家研讨会和录制会议 帮助您成为高效的云开发人员。
  - 连接转为云计算设计的 IBM 产品下载和信息。
  - 关于 社区最新话题 的活动聚合。

讨论

- 加入 developerWorks 中文社区。查看开发人员推动的博客、论坛、组和维基，并与其他 developerWorks 用户交流。

上一篇 如何升级Mininet的Open vSwitch版本

下一篇 Scapy

0

您还没有登录,请[登录]或[注册]

OVS常用命令与使用总结

rocson0 2017年06月13日 12:01 5754

在平时使用ovs中，经常用到的ovs命令，参数，与举例总结。

ovs+floodlight

tuoyahan 2015年04月29日 13:27 1120

配置 ubuntu14.04 LTE, ovs 2.0.2 1. apt-cache search openvswitch 开始安装 2. apt-get install openvswitch-s...

openstack网络模型

jincm13 2015年10月27日 09:58 1794

一、OpenStack网络设备的命名规律： 1、TenantA的router和Linux网络命名空间qrouter名称 root@controller:~# neutron --os-tenan...

[Cocoa]XCode的一些调试技巧

kesalin 2012年01月31日 16:02 29233

XCode的一些调试技巧 罗朝辉 (http://blog.csdn.net/kesalin/) CC 许可，转载请注明出处 XCode 内置GDB，我们可以在命令行中使用 GDB 命令来调...

Open vSwitch的相关原理与配置

jk19920523 2014年10月16日 15:40 1862

一、前言： 在前面我们介绍了传统设备与SDN设备的

基于 Open vSwitch 的 OpenFlow 实践

jincm13 2014年07月11日 14:12 609

Open vSwitch 概述 Open vSwitch（下面简称为 OVS）是由 Nicira Networks 主导的，运行在虚拟化平台（例如 KVM，Xen）上的虚拟交换机。在虚拟化平台上，O...

.NET中的幕后英雄：MSCOREE.DLL

 ATField 2007年08月19日 23:46 21706

现在做.NET Framework的开发的朋友应该是越来越多了，但是可能并非人人都对MSCOREE.DLL非常了解。而事实上，毫不夸张地说，MSCOREE.DLL是.NET Framework中最为核...

整理了一下 发个关于端口的

 isuker 2016年08月05日 09:45 45

http://bbs.blueidea.com/thread-1541935-1-1.html 关于端口的有 可能不全 系统服务及木马默认端口表 https.blueidea.co...

基于open vSwitch，floodlight的openflow实践

基于 Open vSwitch 的 OpenFlow 实践 是用UnitedStack的UOS，创建一个虚拟机来完成全部的试验。 和IBM的文档不一样的地方是：我使用的是ubuntu 14.04，...

 Tomstrong_369 2014年11月27日 17:49 2615

在路由器上进行基于OpenVswitch的openflow实验

1.简介 本篇文章主要是在刷写了OpenWRT的路由器上的OpenVswitch penflow流表的测试，实际部署的测试过程会遇到许多仿真所碰不见的问题。2.实验环境 两台Ubuntu14.0...

 qq_20448859 2017年04月13日 16:06 612

Java对图片Base64转码--HTML对Base64解码 [Java加强版]

Java对图片Base64编码 package base64; import java.awt.image.BufferedImage; import java.io.Byte...

 u012495182 2014年10月11日 14:39 2605

wpa_supplicant 的配置说明文件 wpa_supplicant.conf 译文

wpa_supplicant 配置文件的例子 这个配置文件描述的格式和列表都是可用的选项 请阅读“examples”子目录下简单的配置例子 空行和空字符以及以“#”开头的字符都会被忽略 注意：这个文件...

 qq_22716879 2016年05月18日 08:39 13970

Open vSwitch 与 OpenFlow

 u011956172 2017年07月05日 09:47 246

1 Open Vswitch 参考： http://blog.csdn.net/lizheng2300/article/details/54582310

openVswitch（OVS）源代码分析之简介

云计算是个全世界的话题，所以也有全世界的能人异士来为实现这个云计算而奋斗。我现阶段遇到的有关云计算的技术就是openVswitch和docker技术。那就先从openVswitch开始介绍起，我会用一...

 YuZhiHui_No1 2014年09月09日 23:39 11284

汇编语言基本概念汇总

 CHENYUFENG1991 2015年07月28日 00:14 3968

汇编语言应该是我们现在学的最“低级”的语言了，因为现在不会再有人去学机器语言了。而汇编语言还在一些硬件或者嵌入式设备上使用并开发着。以下资料是为了大学的汇编考试整理的资料，现在与大家分享，希望能给大家...

ubuntu14.04编译android源码

 coloriy 2015年10月15日 15:20 549

Initializing a Build Environment IN THIS DOCUMENT Choosing a BranchSetting up a Linux ...

open vswitch研究：openflow I

 majieyue 2012年11月03日 23:18 8950

关于openflow的规范不是本文讨论范畴，这篇主要讨论OVS对openflow的支持，代码基本都在ofproto/目录下 struct ofproto代表了一个openflow switch的模型...

0

OpenStack, OpenDaylight, OpenFlow and Ope Switch

PS：在Quora上看到的，将openstack，openflow，opendaylight解释的 。学习一下。 OpenStack is a Cloud Management System w...

 u010455041 2015年12月03日 16:27 255

DllMain中不当操作导致死锁问题的分析--Disable adLibraryCalls对Dll...

（转载于breaksoftware的csdn博客）《windows核心编程》作者在讨论DllMain执行序列化的时候，曾说过一个他的故事：他试图通过调用DisableThreadLibrary...

 ProgrammingRing 2014年03月18日 15:38 667

mongoDB配制及学习mongoDB配制及学习

转自：http://www.cnblogs.com/cxd4321/archive/2012/05/18/2507777.html 第一部分 基础篇 第一章 走进MongoDBMongoDB 是一...

 tywei2012 2013年09月22日 15:05 1984