构建SDN网络

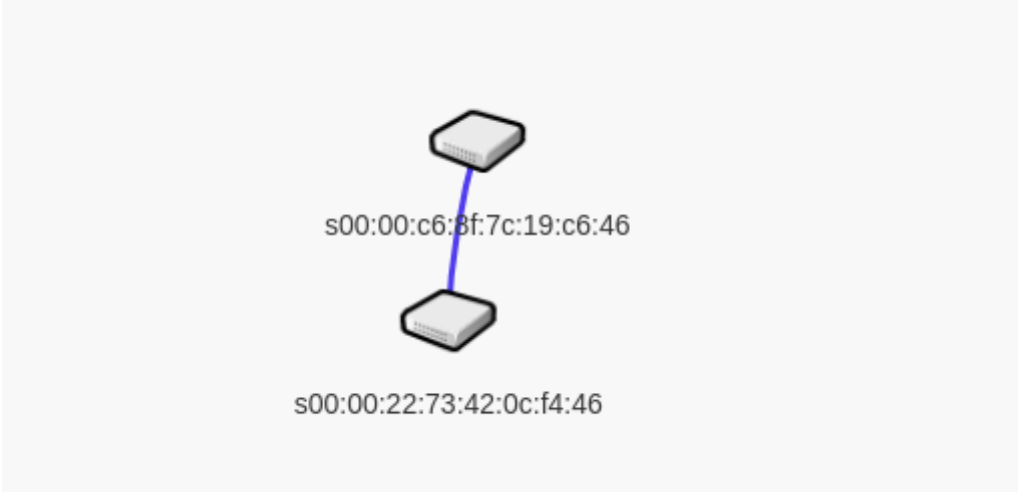
1. 包含两个OVS和两个主机，每个主机链接到一个OVS上，OVS之间互联

```
创建两个OVS交换机
ovs-vsctl add-br s1
ovs-vsctl add-br s2

查看
ovs-vsctl show

s1添加端口p1, p2
ovs-vsctl add-port s1 p1
ovs-vsctl set Interface p1 ofport_request=10
ovs-vsctl set Interface p1 type=internal
ethtool -i p1

ovs-vsctl add-port s1 p2
ovs-vsctl set Interface p2 ofport_request=11
ovs-vsctl set Interface p2 type=internal
ethtool -i p2

s2添加端口p3, p4
ovs-vsctl add-port s2 p3
ovs-vsctl set Interface p3 ofport_request=1
ovs-vsctl set Interface p3 type=internal
ethtool -i p3

ovs-vsctl add-port s2 p4
ovs-vsctl set Interface p4 ofport_request=2
ovs-vsctl set Interface p4 type=internal
ethtool -i p4


创建虚拟主机h1, h2
ip netns add h1
ip link set p1 netns h1
ip netns exec h1 ip addr add 192.168.10.10/24 dev p1
ip netns exec h1 ifconfig p1 promisc up

ip netns add h2
ip link set p4 netns h2
ip netns exec h2 ip addr add 192.168.10.11/24 dev p4
ip netns exec h2 ifconfig p4 promisc up

建立交换机之间的链路
对应端口设置为patch
ovs-vsctl set interface p2 type=patch
ovs-vsctl set interface p3 type=patch

创建p2和p3之间的链路
ovs-vsctl set interface p2 options:peer=p3
```

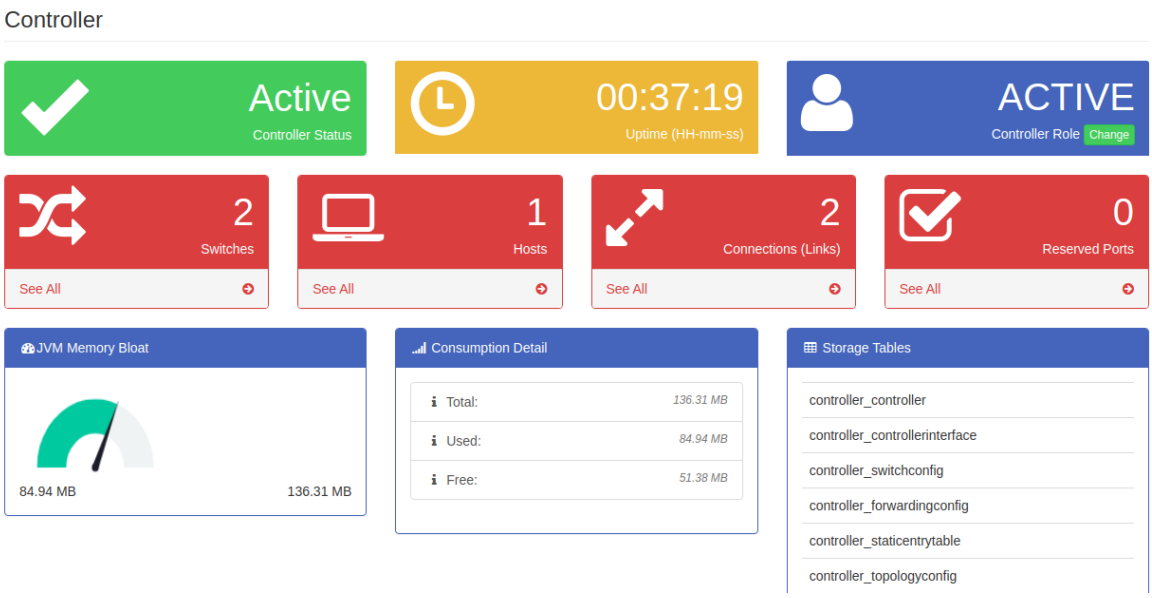
```
ovs-vsctl set interface p3 options:peer=p2

添加流表项
ovs-ofctl add-flow s1 "in_port=10, actions=output:11"
ovs-ofctl add-flow s1 "in_port=11, actions=output:10"

ovs-ofctl add-flow s2 "in_port=1, actions=output:2"
ovs-ofctl add-flow s2 "in_port=2, actions=output:1"

测试
ip netns exec h1 ping 192.168.10.11
ip netns exec h2 ping 192.168.10.10

s1, s2连接到控制器
ovs-vsctl set-controller s1 tcp:127.0.0.1:6653
ovs-vsctl set-controller s2 tcp:127.0.0.1:6653
```

```
root@oval:/home/zen# ip netns exec h1 ping 192.168.10.11
PING 192.168.10.11 (192.168.10.11) 56(84) bytes of data.
64 bytes from 192.168.10.11: icmp_seq=1 ttl=64 time=0.808 ms
64 bytes from 192.168.10.11: icmp_seq=2 ttl=64 time=0.084 ms
64 bytes from 192.168.10.11: icmp_seq=3 ttl=64 time=0.091 ms
64 bytes from 192.168.10.11: icmp_seq=4 ttl=64 time=0.098 ms
64 bytes from 192.168.10.11: icmp_seq=5 ttl=64 time=0.091 ms
64 bytes from 192.168.10.11: icmp_seq=6 ttl=64 time=0.092 ms
64 bytes from 192.168.10.11: icmp_seq=7 ttl=64 time=0.088 ms
64 bytes from 192.168.10.11: icmp_seq=8 ttl=64 time=0.037 ms
64 bytes from 192.168.10.11: icmp_seq=9 ttl=64 time=0.089 ms
64 bytes from 192.168.10.11: icmp_seq=10 ttl=64 time=0.089 ms
^C
--- 192.168.10.11 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9197ms
rtt min/avg/max/mdev = 0.037/0.156/0.808/0.218 ms
root@oval:/home/zen# ip netns exec h2 ping 192.168.10.10
PING 192.168.10.10 (192.168.10.10) 56(84) bytes of data.
64 bytes from 192.168.10.10: icmp_seq=1 ttl=64 time=0.062 ms
64 bytes from 192.168.10.10: icmp_seq=2 ttl=64 time=0.080 ms
64 bytes from 192.168.10.10: icmp_seq=3 ttl=64 time=0.077 ms
64 bytes from 192.168.10.10: icmp_seq=4 ttl=64 time=0.060 ms
^C
--- 192.168.10.10 ping statistics ---
```

拓扑图



控制器

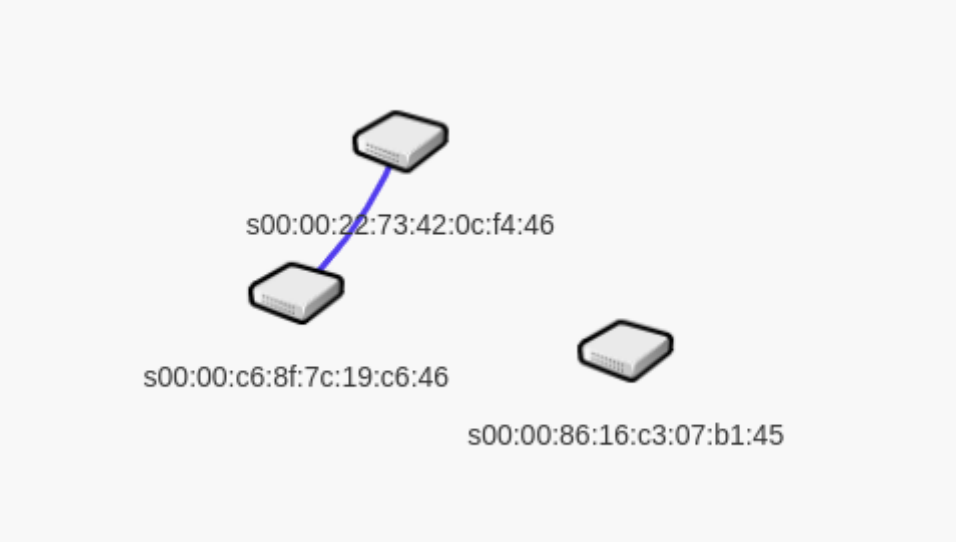


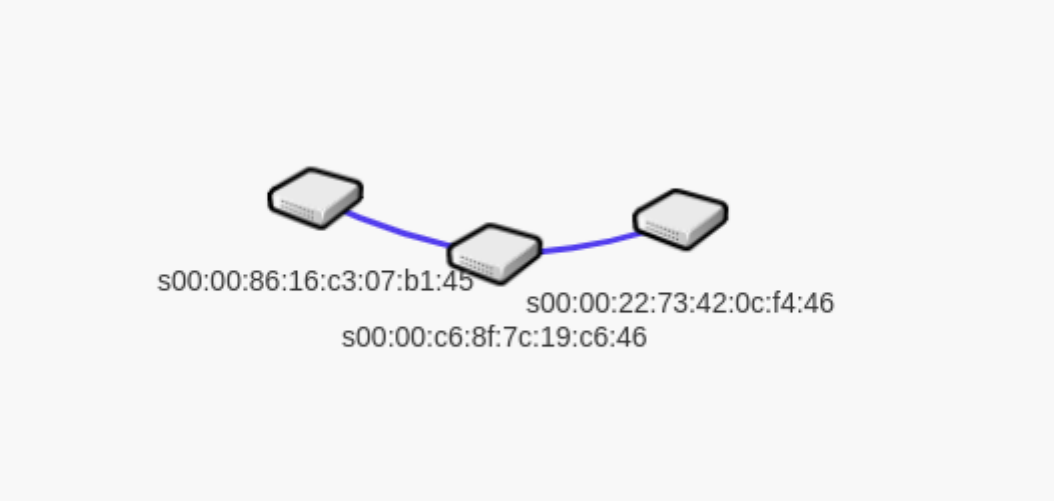
2. 包含3个OVS，3个主机，每个主机和一个OVS相连，3个OVS之间互联

```
创建三个OVS交换机
ovs-vsctl add-br s1
ovs-vsctl add-br s2
ovs-vsctl add-br s3

查看
ovs-vsctl show

s1添加端口p1, p2
ovs-vsctl add-port s1 p1
ovs-vsctl set Interface p1 ofport_request=10
ovs-vsctl set Interface p1 type=internal
ethtool -i p1

ovs-vsctl add-port s1 p2
ovs-vsctl set Interface p2 ofport_request=11
ovs-vsctl set Interface p2 type=internal
ethtool -i p2

s2添加端口p3, p4
ovs-vsctl add-port s2 p3
ovs-vsctl set Interface p3 ofport_request=1
ovs-vsctl set Interface p3 type=internal
ethtool -i p3

ovs-vsctl add-port s2 p4
ovs-vsctl set Interface p4 ofport_request=2
ovs-vsctl set Interface p4 type=internal
ethtool -i p4

s3添加端口p5, p6
ovs-vsctl add-port s3 p5
ovs-vsctl set Interface p5 ofport_request=10
ovs-vsctl set Interface p5 type=internal
ethtool -i p5
```

```
ovs-vsctl add-port s3 p6
ovs-vsctl set Interface p6 ofport_request=11
ovs-vsctl set Interface p6 type=internal
ethtool -i p6


创建虚拟主机h1, h2, h3
ip netns add h1
ip link set p1 netns h1
ip netns exec h1 ip addr add 192.168.10.10/24 dev p1
ip netns exec h1 ifconfig p1 promisc up

ip netns add h2
ip link set p4 netns h2
ip netns exec h2 ip addr add 192.168.10.11/24 dev p4
ip netns exec h2 ifconfig p4 promisc up

ip netns add h3
ip link set p5 netns h3
ip netns exec h3 ip addr add 192.168.10.12/24 dev p5
ip netns exec h3 ifconfig p5 promisc up

建立交换机之间的链路
对应端口设置为patch
ovs-vsctl set interface p2 type=patch
ovs-vsctl set interface p3 type=patch
ovs-vsctl set interface p6 type=patch

创建p2和p3之间的链路
ovs-vsctl set interface p2 options:peer=p3
ovs-vsctl set interface p3 options:peer=p2

创建p2和p6之间的链路
ovs-vsctl set interface p2 options:peer=p6
ovs-vsctl set interface p6 options:peer=p2

创建p3和p6之间的链路
ovs-vsctl set interface p3 options:peer=p6
ovs-vsctl set interface p6 options:peer=p3



添加流表项
s1的流表项
ovs-ofctl add-flow s1 "in_port=10, actions=output:11"
ovs-ofctl add-flow s1 "in_port=11, actions=output:10"

s2的流表项
ovs-ofctl add-flow s2 "in_port=1, actions=output:2"
ovs-ofctl add-flow s2 "in_port=2, actions=output:1"

s3的流表项
ovs-ofctl add-flow s3 "in_port=10, actions=output:11"
ovs-ofctl add-flow s3 "in_port=11, actions=output:10"

测试
ip netns exec h1 ping 192.168.10.11
```

```
ip netns exec h2 ping 192.168.10.10
```





代码下发流表

```java
@Override
public void startUp(FloodlightModuleContext context) throws FloodlightModuleException {
    System.out.println("新模块");
    new Thread() {
        @Override
        public void run() {
            try {
                //Thread.sleep(20000);  //交换机与控制器链接需要一定的时间，因此要先睡眠一段时间
            } catch (Exception e) {
                e.printStackTrace();
            }
            //addFlow();
        }
    }.start();
}

public void addFlow() {
    //前期准备
    //给哪个交换机下发流表，必须要先拿到对应交换机的对象
    Set<DatapathId> ids = switchService.getAllSwitchDpids();
    DatapathId swid = null;
    for (DatapathId id: ids) {
        swid = id;
        break;
```

```java
IOFSwitch sw = switchService.getSwitch(swid); //拿到交换机对象
//封装flowmod消息
OFFlowMod.Builder fmb = sw.getOFFactory().buildFlowAdd();
//匹配域
Match.Builder mb = sw.getOFFactory().buildMatch();
mb.setExact(MatchField.ETH_SRC, MacAddress.of("F4:4d:d4:cc:00:2f"));
mb.setExact(MatchField.ETH_TYPE, EthType.IPv4);
mb.setExact(MatchField.IPV4_SRC, IPv4Address.of("192.168.1.3"));
mb.setExact(MatchField.IP_PROTO, IpProtocol.UDP);
mb.setExact(MatchField.IN_PORT, OFPort.of(54)); //交换机的物理端口

//指令
List<OFAction> actions = new ArrayList<>();
actions.add(sw.getOFFactory().actions().output(OFPort.of(55), Integer.MAX_VALUE));
// 封装flowmod
U64 cookie = AppCookie.makeCookie( application: 2,  user: 0);
fmb.setCookie(cookie)
        .setHardTimeout(0)
        .setIdleTimeout(0)
        .setBufferId(OFBufferId.NO_BUFFER)
        .setPriority(5)
        .setMatch(mb.build());
FlowModUtils.setActions(fmb, actions,sw);
```