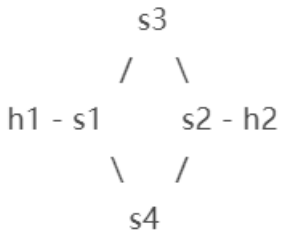
Lab：Write SDN Controller

叶增渝 519030910168

1.构建如下拓扑结构的文件为 test.py（可放在任意文件下执行）



2.符合要求的 remote controller 文件为 6_1.py，文件放在~/ryu/app 文件夹下执行如下命令开启 controller

```
spoilvoid@ubuntu:~/ryu/ryu/app$ ryu-manager --verbose 6_1.py
```

然后我们再使用如下命令构建网络拓扑

```
spoilvoid@ubuntu:~/ryu/ryu/app$ sudo python3 test.py
```

在 mininet 中使用如下命令测试网络转发

```
mininet> h1 ping h2
```

首先我们可以看到是能够 ping 通的

```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=115 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=51.8 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=52.1 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=52.3 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=53.3 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=51.5 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=51.7 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=52.0 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=52.1 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=52.6 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=50.8 ms
s64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=51.2 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=49.6 ms
```

然后我们查看 packet_in_handler 发出的信息，我们可以看到在 flag 为 True 时，我们使用经过 s3 的线路进行转发，在时间到达后，我们的 flag 被取反，此时使用经过 s4 的线路进行转发，说明达到目标。其中计时我们使用 `self.time_start = time.time()`作为计时开始，每达到 5s 更改 flag 切换线路（由于频繁切换线路，我们不添加流表项）

```
time_length = time.time() - self.time_start
if((int(time_length) // 5) % 2 != 0):
    self.flag = not self.flag
self.time_start += (int(time_length) // 5) * 5
```

p.s.这里输出的信息按照顺序分别为 16 位 dp_id、source、destination、in_port、out_port（4294967291 为 OFPP_FLOOD 的输出）、flag

```
packet in 0000000000000002 0a:32:e7:9a:5a:c9 32:c5:81:a0:4e:5c 2 1 True
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000002 32:c5:81:a0:4e:5c 0a:32:e7:9a:5a:c9 1 2 True
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000003 32:c5:81:a0:4e:5c 0a:32:e7:9a:5a:c9 2 4294967291 True
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 32:c5:81:a0:4e:5c 0a:32:e7:9a:5a:c9 2 1 True
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 0a:32:e7:9a:5a:c9 32:c5:81:a0:4e:5c 1 2 True
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000003 0a:32:e7:9a:5a:c9 32:c5:81:a0:4e:5c 1 4294967291 True
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000002 0a:32:e7:9a:5a:c9 32:c5:81:a0:4e:5c 2 1 True
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000002 32:c5:81:a0:4e:5c 0a:32:e7:9a:5a:c9 1 2 True
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000003 32:c5:81:a0:4e:5c 0a:32:e7:9a:5a:c9 2 4294967291 True
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 32:c5:81:a0:4e:5c 0a:32:e7:9a:5a:c9 2 1 True
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 0a:32:e7:9a:5a:c9 32:c5:81:a0:4e:5c 1 3 False
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000004 0a:32:e7:9a:5a:c9 32:c5:81:a0:4e:5c 1 4294967291 False
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000002 0a:32:e7:9a:5a:c9 32:c5:81:a0:4e:5c 3 1 False
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000002 32:c5:81:a0:4e:5c 0a:32:e7:9a:5a:c9 1 3 False
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000004 32:c5:81:a0:4e:5c 0a:32:e7:9a:5a:c9 2 4294967291 False
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 32:c5:81:a0:4e:5c 0a:32:e7:9a:5a:c9 3 1 False
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 0a:32:e7:9a:5a:c9 32:c5:81:a0:4e:5c 1 3 False
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000004 0a:32:e7:9a:5a:c9 32:c5:81:a0:4e:5c 1 4294967291 False
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000002 0a:32:e7:9a:5a:c9 32:c5:81:a0:4e:5c 3 1 False
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000002 32:c5:81:a0:4e:5c 0a:32:e7:9a:5a:c9 1 3 False
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000004 32:c5:81:a0:4e:5c 0a:32:e7:9a:5a:c9 2 4294967291 False
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 32:c5:81:a0:4e:5c 0a:32:e7:9a:5a:c9 3 1 False
```

3. 符合要求的 remote controller 文件为 6_2.py，文件放在~/ryu/app 文件夹下执行如下命令开启 controller

```
spoilvoid@ubuntu:~/ryu/ryu/app$ ryu-manager --verbose 6_2.py
```

然后我们再使用如下命令构建网络拓扑

```
spoilvoid@ubuntu:~/ryu/ryu/app$ sudo python3 test.py
```

在 mininet 中使用如下命令测试网络转发

```
mininet> h1 ping h2
```

首先我们可以看到是能够 ping 通的

```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=101 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=48.9 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=48.3 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=48.3 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=49.7 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=48.2 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=49.0 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=49.0 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=49.6 ms
```

然后我们查看 packet_in_handler 发出的信息，我们可以看到在 ping 测试时，同时能接收到来自 s3 与 s4 的消息，说明两路同时在转发， 由于 ping 通，我们可以知道是各 50%的两路转发

```
packet in 0000000000000004 7a:15:a3:9e:37:16 fa:8a:77:1b:ef:f2 1 2
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000002 7a:15:a3:9e:37:16 fa:8a:77:1b:ef:f2 3 1
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000004 fa:8a:77:1b:ef:f2 7a:15:a3:9e:37:16 2 1
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 fa:8a:77:1b:ef:f2 7a:15:a3:9e:37:16 3 1
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000003 7a:15:a3:9e:37:16 fa:8a:77:1b:ef:f2 1 2
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000002 7a:15:a3:9e:37:16 fa:8a:77:1b:ef:f2 2 1
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000003 fa:8a:77:1b:ef:f2 7a:15:a3:9e:37:16 2 1
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 fa:8a:77:1b:ef:f2 7a:15:a3:9e:37:16 2 1
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000002 5e:39:c2:49:d6:47 33:33:00:00:00:02 2 1
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000004 7a:15:a3:9e:37:16 fa:8a:77:1b:ef:f2 1 2
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000002 7a:15:a3:9e:37:16 fa:8a:77:1b:ef:f2 3 1
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000004 fa:8a:77:1b:ef:f2 7a:15:a3:9e:37:16 2 1
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 fa:8a:77:1b:ef:f2 7a:15:a3:9e:37:16 3 1
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000004 7a:15:a3:9e:37:16 fa:8a:77:1b:ef:f2 1 2
```

4. 符合要求的 remote controller 文件为 6_3.py，文件放在~/ryu/app 文件夹下执行如下命令开启 controller（这里在 s1 与 s3 使用了 OFPGT_FF，当 link 断开时，s1 自然地选择下方通路，而已经从 s2 传至 s3 的 packet 会改为流向 s2，s2 根据记录的 h2 的 eth 与 packet 的 source 进行比较，将返回的包重新发往 s4）

```
spoilvoid@ubuntu:~/ryu/ryu/app$ ryu-manager --verbose 6_3.py
```

然后我们再使用如下命令构建网络拓扑

```
spoilvoid@ubuntu:~/ryu/ryu/app$ sudo python3 test.py
```

在 mininet 中使用如下命令测试网络转发

```
mininet> h1 ping h2
```

首先我们可以看到是能够 ping 通的

```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=102 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=50.6 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=49.3 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=49.9 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=51.6 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=48.1 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=47.8 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=48.8 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=49.8 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=49.6 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=50.7 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=48.4 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=64 time=50.2 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=48.9 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=49.1 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=64 time=51.4 ms
64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=49.3 ms
```

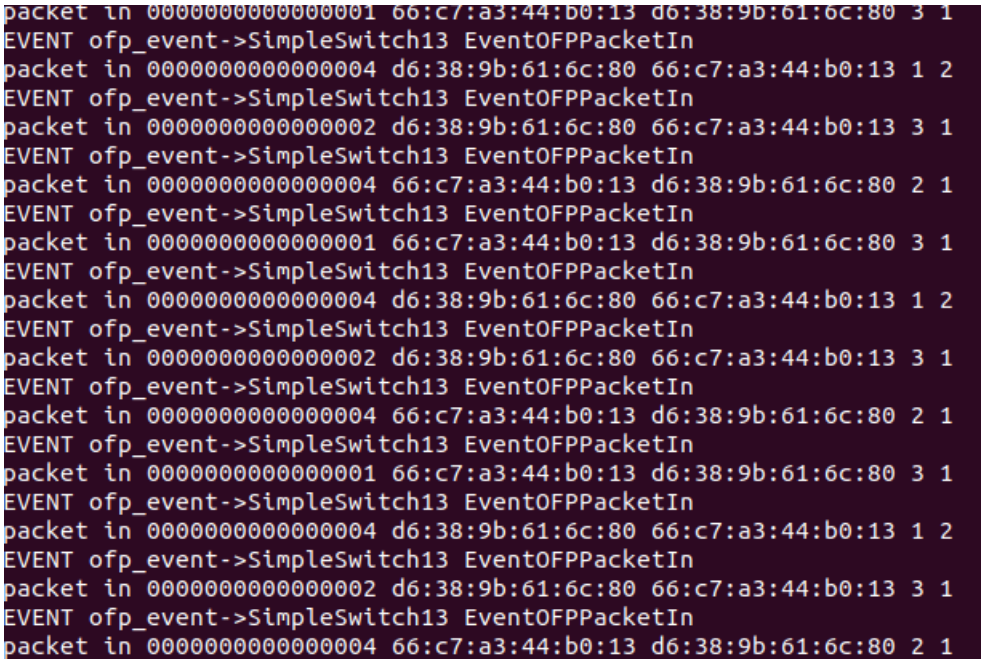
此时的输出信息如下

```
packet in 0000000000000002 62:13:f2:3d:c5:63 33:33:00:00:00:02 2 1
packet in 0000000000000003 32:73:d9:ca:6c:46 33:33:00:00:00:02 1 2
packet in 0000000000000001 4a:3d:fd:73:56:3e 33:33:00:00:00:02 2 1
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000002 32:73:d9:ca:6c:46 33:33:00:00:00:02 2 1
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000003 fe:cb:0c:31:e5:ae f2:de:15:32:62:68 1 2
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000002 fe:cb:0c:31:e5:ae f2:de:15:32:62:68 2 1
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000004 f2:de:15:32:62:68 fe:cb:0c:31:e5:ae 2 1
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000001 f2:de:15:32:62:68 fe:cb:0c:31:e5:ae 3 1
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000003 fe:cb:0c:31:e5:ae f2:de:15:32:62:68 1 2
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 0000000000000002 fe:cb:0c:31:e5:ae f2:de:15:32:62:68 2 1
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
```

我们使用如下命令将连接 s3 的端口 2 断开

```
spoilvoid@ubuntu:~$ sudo ovs-ofctl -O OpenFlow13 mod-port s1 2 down
```

此时我们可以看到依然每一个 packet 都 ping 通，但是此时，我们只能看到含 s4 的线路转发



说明完成了 fast failover 操作