Lab 6: RYU Open Flow Controller

鲁君一 517261910033

简介

本项目基于RYU实现了轮询算法（Question1），负载均衡算法（Question2），容灾恢复算法（Question3），并在基于mininet搭建的虚拟网络拓扑上验证了这些算法。

目录结构描述

```
abigail@ubuntu:~/Desktop/RYU$ tree
.
├── fastfailover.py #容灾恢复算法（Question3）
├── multi_path.py   #负载均衡算法（Question2）
├── one_path.py     #只允许使用一条线路h1-s1-s3-s2-h1,从而避免了arp风暴
├── setup.py        #设置mininet网络拓扑
└── switch_path.py  #轮询算法（Question1）

0 directories, 5 files
```

使用

- ubuntu 20.04 LTS
- python 3.8.10
- mininet 2.3.0
- ryu 4.34
- OpenFlow 1.3

将`fastfailover.py`,`multi_path.py`,`switch_path.py`复制到`ryu/ryu/app`文件夹下，分别在终端运行：

```
ryu-manager switch_path.py gui_topology/gui_topology.py --observe-links

ryu-manager multi_path.py gui_topology/gui_topology.py --observe-links

ryu-manager fastfailover.py gui_topology/gui_topology.py --observe-links
```

(可以在弹出的网页上查看交换机拓扑，点击交换机即可查看该交换机当前的流表)

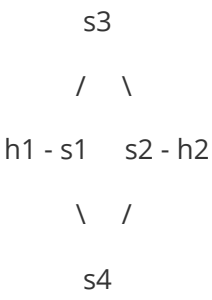
然后在setup.py所在文件夹内开启一个终端，运行：

```
sudo python3 setup.py
```

实验结果

Q1.Set up the network

通过setup.py实现了下面的网络：



网络基本信息如下所示

```
mininet> nodes
available nodes are:
c1 h1 h2 s1 s2 s3 s4
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth1
s1 lo:  s1-eth1:h1-eth0 s1-eth2:s3-eth1 s1-eth3:s4-eth1
s2 lo:  s2-eth1:h2-eth0 s2-eth2:s3-eth2 s2-eth3:s4-eth2
s3 lo:  s3-eth1:s1-eth2 s3-eth2:s2-eth2
s4 lo:  s4-eth1:s1-eth3 s4-eth2:s2-eth3
c1
mininet> links
h1-eth0<->s1-eth1 (OK OK)
h2-eth0<->s2-eth1 (OK OK)
s1-eth2<->s3-eth1 (OK OK)
s1-eth3<->s4-eth1 (OK OK)
s2-eth2<->s3-eth2 (OK OK)
s2-eth3<->s4-eth2 (OK OK)
mininet> ports
s1 lo:0 s1-eth1:1 s1-eth2:2 s1-eth3:3
s2 lo:0 s2-eth1:1 s2-eth2:2 s2-eth3:3
s3 lo:0 s3-eth1:1 s3-eth2:2
s4 lo:0 s4-eth1:1 s4-eth2:2
```

Q2.Switches paths

代码

通过每隔五秒自动更改流表实现，关键代码如下：

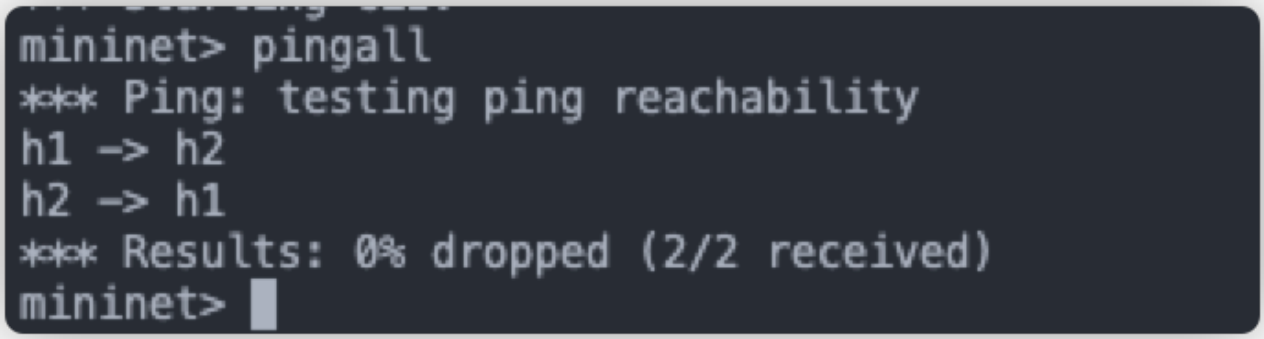
```
    def _change_flow(self):
        flag = 1

        while True:
            print('datapath:', self.datapaths)
            try:
                if flag == 1:
                    print('flag:', flag)
                    flag = 2
                    self.match_flow(dpid=1, in_port=1, out_port=2, priority=1,
add_del=0)
                    self.match_flow(dpid=1, in_port=2, out_port=1, priority=1,
add_del=0)

                    self.match_flow(dpid=1, in_port=1, out_port=3, priority=1,
add_del=1)
                    self.match_flow(dpid=1, in_port=3, out_port=1, priority=1,
add_del=1)

                    self.match_flow(dpid=2, in_port=1, out_port=2, priority=1,
add_del=0)
                    self.match_flow(dpid=2, in_port=2, out_port=1, priority=1,
add_del=0)
```

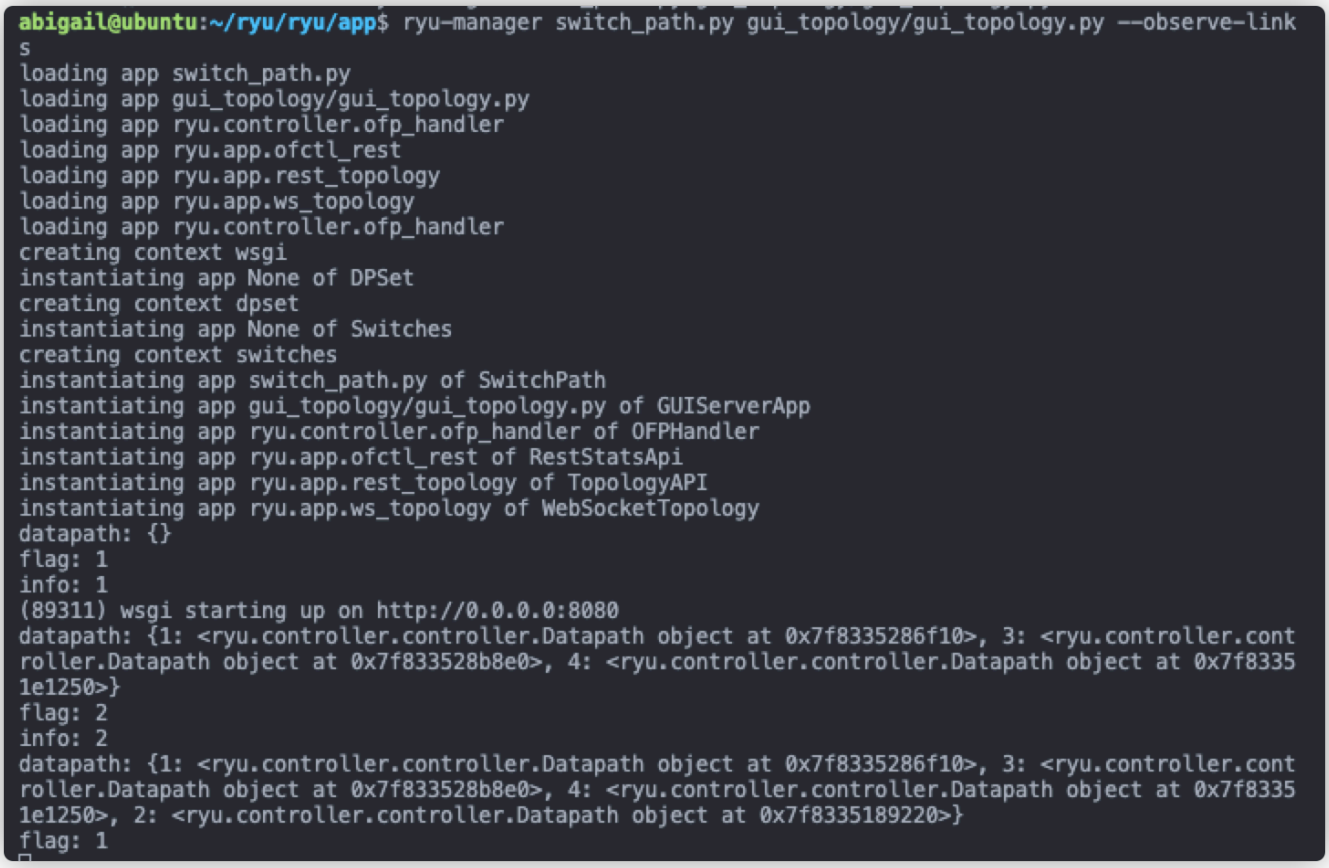
```
                    self.match_flow(dpid=2, in_port=1, out_port=3, priority=1,
add_del=1)
                    self.match_flow(dpid=2, in_port=3, out_port=1, priority=1,
add_del=1)

                    self.match_flow(dpid=3, in_port=1, out_port=2, priority=1,
add_del=0)
                    self.match_flow(dpid=3, in_port=2, out_port=1, priority=1,
add_del=0)

                    self.match_flow(dpid=4, in_port=1, out_port=2, priority=1,
add_del=1)
                    self.match_flow(dpid=4, in_port=2, out_port=1, priority=1,
add_del=1)


                elif flag == 2:
                    print('flag:', flag)
                    flag = 1
                    self.match_flow(dpid=1, in_port=1, out_port=2, priority=1,
add_del=1)
                    self.match_flow(dpid=1, in_port=2, out_port=1, priority=1,
add_del=1)

                    self.match_flow(dpid=1, in_port=1, out_port=3, priority=1,
add_del=0)
                    self.match_flow(dpid=1, in_port=3, out_port=1, priority=1,
add_del=0)

                    self.match_flow(dpid=2, in_port=1, out_port=2, priority=1,
add_del=1)
                    self.match_flow(dpid=2, in_port=2, out_port=1, priority=1,
add_del=1)

                    self.match_flow(dpid=2, in_port=1, out_port=3, priority=1,
add_del=0)
                    self.match_flow(dpid=2, in_port=3, out_port=1, priority=1,
add_del=0)

                    self.match_flow(dpid=3, in_port=1, out_port=2, priority=1,
add_del=1)
                    self.match_flow(dpid=3, in_port=2, out_port=1, priority=1,
add_del=1)

                    self.match_flow(dpid=4, in_port=1, out_port=2, priority=1,
add_del=0)
                    self.match_flow(dpid=4, in_port=2, out_port=1, priority=1,
add_del=0)
```

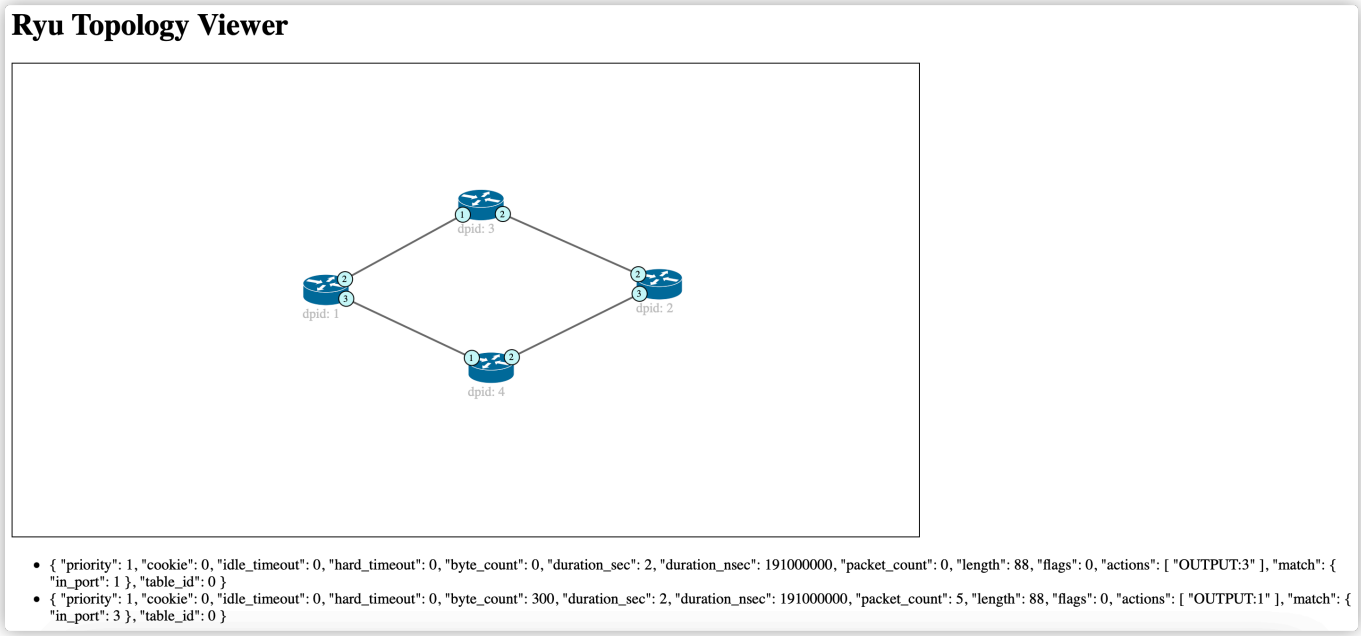
```
            except Exception as info:
                print('info:', info)

            hub.sleep(5)
```

其中，`match_flow`函数为流表项增删函数，add_del=0时候表示添加该条目, add_del=1时候表示删除该条目。

实验结果







Q3.Both paths

代码

通过在`send_group_mod`函数中使用`group table`和其中的`OFPGT_SELECT`功能实现，并将两条线路权重设置为相同。

然后在`flow_table_initial`中调用`send_group_mod`函数负责处理需要组表的交换机接口。

主要代码如下：

```
    def send_group_mod(self,datapath):
        parser =datapath.ofproto_parser
        ofp=datapath.ofproto

        port = 3
        actions_1 = [parser.OFPActionOutput(port)]
        port = 2
        actions_2 = [parser.OFPActionOutput(port)]

        weight_1 = 50
        weight_2 = 50

        watch_port = ofproto_v1_3.OFPP_ANY
        watch_group = ofproto_v1_3.OFPQ_ALL

        buckets = [
            parser.OFPBucket(weight_1, watch_port, watch_group, actions_1),
            parser.OFPBucket(weight_2, watch_port, watch_group, actions_2)]

        group_id = 50
        req = parser.OFPGroupMod(datapath, ofp.OFPFC_ADD,ofp.OFPGT_SELECT, group_id,
buckets)
```

```
        datapath.send_msg(req)

        actions = [parser.OFPActionGroup(group_id=group_id)]
        match = parser.OFPMatch(in_port=1)
        self.add_flow(datapath, 10, match, actions)
```

```
    def flow_table_initial(self,datapath,parser):
        if datapath.id==1:
            self.send_group_mod(datapath)

            actions = [parser.OFPActionOutput ( 1 )]
            match = parser.OFPMatch ( in_port=3 )
            self.add_flow( datapath, 10, match, actions )

            actions = [parser.OFPActionOutput ( 1 )]
            match = parser.OFPMatch ( in_port=2 )
            self.add_flow( datapath, 10, match, actions )

        if datapath.id==2:
            self.send_group_mod(datapath)

            actions = [parser.OFPActionOutput ( 1 )]
            match = parser.OFPMatch ( in_port=2 )
            self.add_flow( datapath, 10, match, actions )

            actions = [parser.OFPActionOutput ( 1 )]
            match = parser.OFPMatch ( in_port=3 )
            self.add_flow( datapath, 10, match, actions )

        if datapath.id==3:
            actions = [parser.OFPActionOutput ( 1 )]
            match = parser.OFPMatch ( in_port=2 )
            self.add_flow( datapath, 10, match, actions )

            actions = [parser.OFPActionOutput ( 2 )]
            match = parser.OFPMatch ( in_port=1 )
            self.add_flow( datapath, 10, match, actions )

        if datapath.id==4:
            actions = [parser.OFPActionOutput ( 1 )]
            match = parser.OFPMatch ( in_port=2 )
            self.add_flow( datapath, 10, match, actions )

            actions = [parser.OFPActionOutput ( 2 )]
            match = parser.OFPMatch ( in_port=1 )
            self.add_flow( datapath, 10, match, actions )
```

实验结果

```
abigail@ubuntu:~/ryu/ryu/app$ ryu-manager multi_path.py gui_topology/gui_topology.py --observe-links
loading app multi_path.py
loading app gui_topology/gui_topology.py
loading app ryu.controller.ofp_handler
loading app ryu.app.ws_topology
loading app ryu.app.rest_topology
loading app ryu.app.ofctl_rest
loading app ryu.controller.ofp_handler
creating context wsgi
instantiating app None of Switches
creating context switches
instantiating app None of DPSet
creating context dpset
instantiating app multi_path.py of MultiPath
instantiating app gui_topology/gui_topology.py of GUIServerApp
instantiating app ryu.controller.ofp_handler of OFPHandler
instantiating app ryu.app.ws_topology of WebSocketTopology
instantiating app ryu.app.rest_topology of TopologyAPI
instantiating app ryu.app.ofctl_rest of RestStatsApi
(87721) wsgi starting up on http://0.0.0.0:8080
hello!!
Successfully add group/flow table for:  2
hello!!
Successfully add group/flow table for:  4
hello!!
Successfully add group/flow table for:  3
hello!!
Successfully add group/flow table for:  1
(87721) accepted ('127.0.0.1', 33310)
127.0.0.1 - - [14/Dec/2021 01:10:23] "GET / HTTP/1.1" 200 515 0.005901
(87721) accepted ('127.0.0.1', 33320)
127.0.0.1 - - [14/Dec/2021 01:10:28] "GET /stats/flow/1 HTTP/1.1" 200 1184 0.001951
```

mininet结果和Q1一样，可以运行`ryu-manager multi_path.py gui_topology/gui_topology.py --observe-links`后查看网页上交换机流表

另外，通过Wireshark抓包分析可以看出，两条线路均有被使用：

Q4.Fast Failover

代码

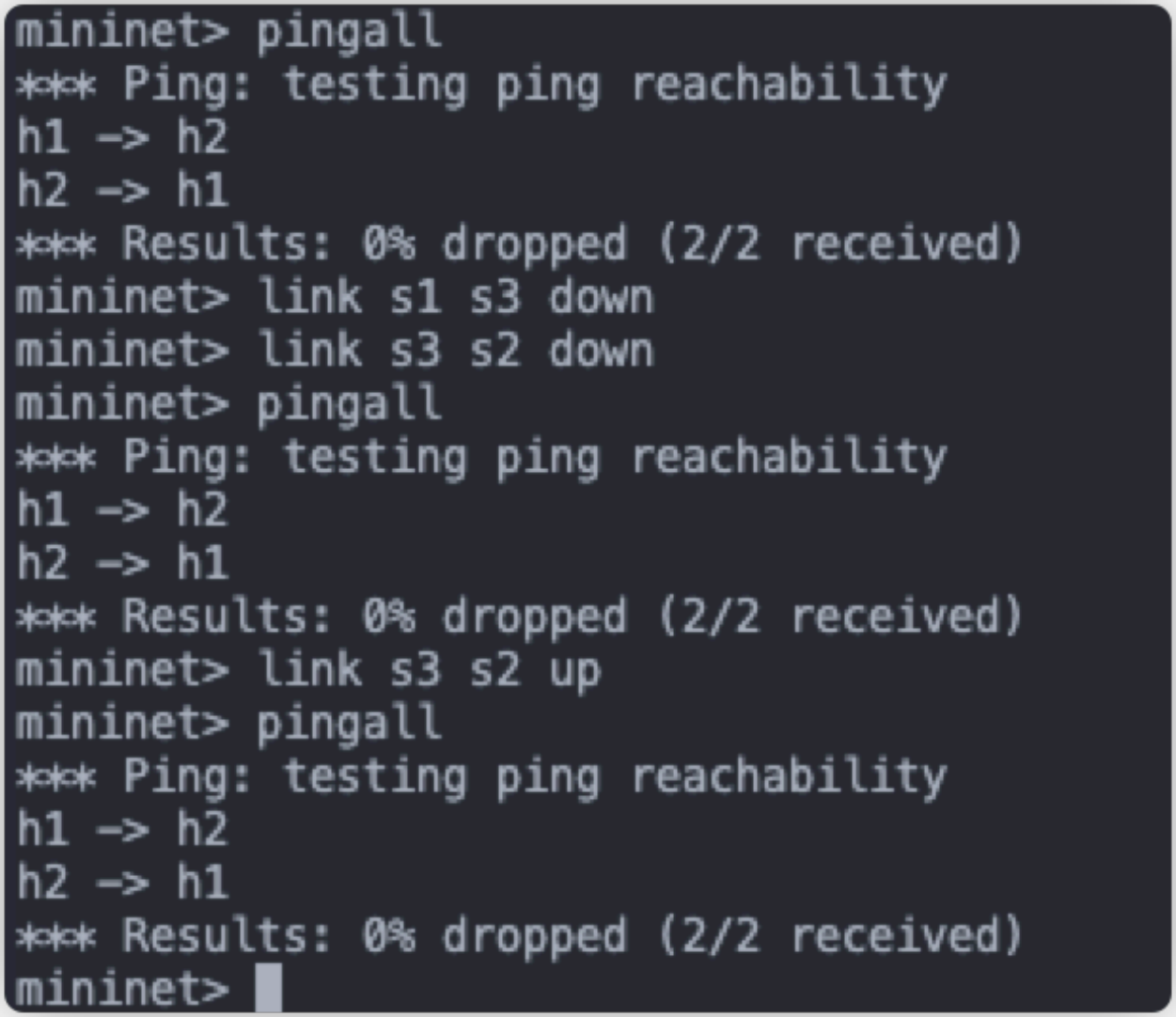
仅需在`multi_path.py`的基础上修改`send_group_mod`函数，使用group table的`OFPGT_FF`功能

```
    def send_group_mod(self,datapath):
        parser =datapath.ofproto_parser
        ofp=datapath.ofproto

        port = 3
        actions_1 = [parser.OFPActionOutput(port)]
        port = 2
        actions_2 = [parser.OFPActionOutput(port)]
```

```
        buckets = [
                    parser.OFPBucket (watch_port=3, actions=actions_1 ),
                    parser.OFPBucket (watch_port=2, actions=actions_2 )]


        group_id = 0
        req = parser.OFPGroupMod(datapath, ofp.OFPFC_ADD,ofp.OFPGT_FF, group_id,
buckets)
        datapath.send_msg(req)

        actions = [parser.OFPActionGroup(group_id=group_id)]
        match = parser.OFPMatch(in_port=1)
        self.add_flow(datapath, 10, match, actions)
```

实验结果

```
abigail@ubuntu:~/ryu/ryu/app$ ryu-manager fastfailover.py gui_topology/gui_topology.py --observe-lin
ks
loading app fastfailover.py
loading app gui_topology/gui_topology.py
loading app ryu.controller.ofp_handler
loading app ryu.app.ofctl_rest
loading app ryu.app.rest_topology
loading app ryu.app.ws_topology
loading app ryu.controller.ofp_handler
creating context wsgi
instantiating app None of DPSet
creating context dpset
instantiating app None of Switches
creating context switches
instantiating app fastfailover.py of FastFailOver
instantiating app gui_topology/gui_topology.py of GUIServerApp
instantiating app ryu.controller.ofp_handler of OFPHandler
instantiating app ryu.app.ofctl_rest of RestStatsApi
instantiating app ryu.app.rest_topology of TopologyAPI
instantiating app ryu.app.ws_topology of WebSocketTopology
(130736) wsgi starting up on http://0.0.0.0:8080
hello!!
Successfully add group/flow table for:  3
hello!!
Successfully add group/flow table for:  2
hello!!
Successfully add group/flow table for:  1
hello!!
Successfully add group/flow table for:  4
(130736) accepted ('127.0.0.1', 47652)
```


mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet> link s1 s3 down
mininet> link s3 s2 down
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet> link s3 s2 up
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet>