

SDN 第二次实验报告

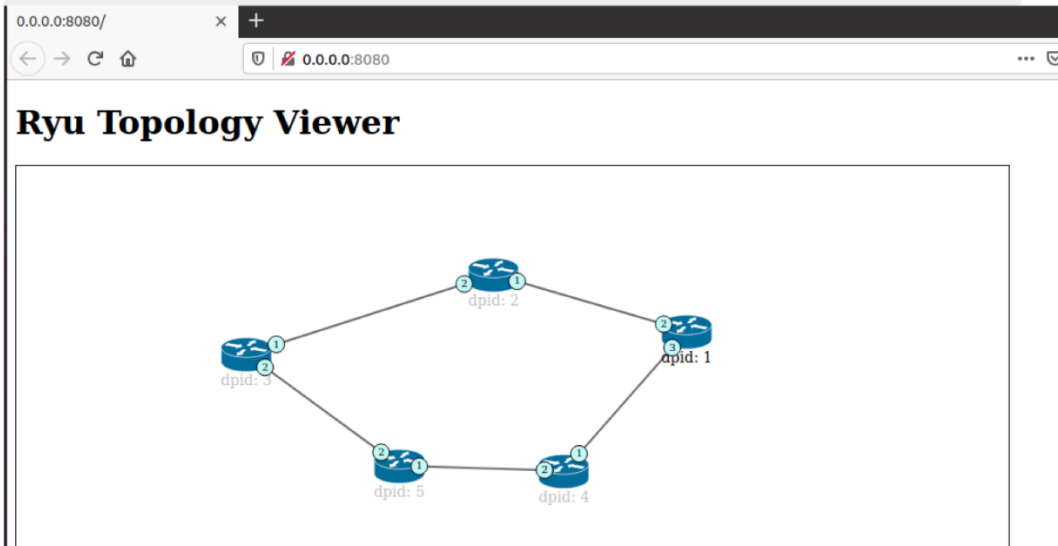
计算机86常傲2182212573

```
from mininet.topo import Topo

class MyTopo( Topo ):
    def __init__( self ):
        "Create custom topo."
        # Initialize topology
        Topo.__init__( self )

        h1 = self.addHost('h1')
        h2 = self.addHost('h2')
        s1 = self.addSwitch('s1')
        s2 = self.addSwitch('s2')
        s3 = self.addSwitch('s3')
        s4 = self.addSwitch('s4')
        s5 = self.addSwitch('s5')
        self.addLink(s1,s2)
        self.addLink(s2,s3)
        self.addLink(s3,s5)
        self.addLink(s1,s4)
        self.addLink(s4,s5)
        self.addLink(h1,s1)
        self.addLink(s5,h2)

topos = { 'mytopo': ( lambda: MyTopo() ) }
```



一、 动态改变转发规则

思路以及代码：

1. 由于包含两条路径：s1-s4-s5 与 s1-s2-s3-s5，可以视为最短路径和最长路径。因此，利用给出的最短路径函数 `short_path()`，修改得到最长路径函数 `long_path()`

```

def long_path(self, src, dst, bw=0):
    if src == dst:
        return []
    result = defaultdict(lambda: defaultdict(lambda: None))
    distance = defaultdict(lambda: None)
    # the node is checked
    seen = [src]

    # the distance to src
    distance[src] = 0

    w = 1 # weight
    while len(seen) < len(self.src_links):
        node = seen[-1]
        if node == dst:
            break
        for (temp_src, temp_dst) in self.src_links[node]:
            if temp_dst not in seen:
                temp_src_port = self.src_links[node][(temp_src,
temp_dst)][0]
                temp_dst_port = self.src_links[node][(temp_src,
temp_dst)][1]
                if (distance[temp_dst] is None) or (distance[temp_dst] <
distance[temp_src] + w):
                    distance[temp_dst] = distance[temp_src] + w
                    # result = {"dpid":(link_src, src_port, link_dst,
dst_port)}
                    result[temp_dst] = (temp_src, temp_src_port,
temp_dst, temp_dst_port)
                    max_node = None
                    max_path = -999
                    # get the max_path node
                    for temp_node in distance:
                        if (temp_node not in seen) and (distance[temp_node] is not
None):
                            if distance[temp_node] > max_path:
                                max_node = temp_node
                                max_path = distance[temp_node]
                    if max_node is None:
                        break
                    seen.append(max_node)

        path = []

    if dst not in result:
        return None

    while (dst in result) and (result[dst] is not None):
        path = [result[dst][2:4]] + path
        path = [result[dst][0:2]] + path
        dst = result[dst][0]
    #self.logger.info("path : %s", str(path))
    return path

```

2. 利用 `hard_timeout` 参数，设置每 5 秒硬删除相关流表项

```
self.add_flow(datapath_path, 100, match, actions, idle_timeout=5,
hard_timeout=5)
```

3. 同时利用全局变量 `self.pathmod`，初始化为 0，在每次下发流表时，判断：

(1) 若 `pathmod` 为偶数，下发最短路径，`pathmod + 1`

(2) 若 `pathmod` 为奇数，下发最长路径，`pathmod + 1`

```
def install_path(self, src_dpid, dst_dpid, src_port, dst_port, ev, src,
dst, pkt_ipv4, pkt_tcp):
    msg = ev.msg
    datapath = msg.datapath
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser

    mid_path = None
    if(self.pathmod % 2 == 0) :
        mid_path = self.short_path(src=src_dpid, dst=dst_dpid)
        self.pathmod = self.pathmod + 1
    else:
        mid_path = self.long_path(src=src_dpid, dst=dst_dpid)
        self.pathmod = self.pathmod + 1

    if mid_path is None:
        return
    self.path = None
    self.path = [(src_dpid, src_port)] + mid_path + [(dst_dpid,
dst_port)]

    self.logger.info("path : %s", str(self.path))

    for i in xrange(len(self.path) - 2, -1, -2):
        datapath_path = self.datapaths[self.path[i][0]]
        match = parser.OFPMatch(in_port=self.path[i][1], eth_src=src,
eth_dst=dst, eth_type=0x0800,
                                ipv4_src=pkt_ipv4.src,
                                ipv4_dst=pkt_ipv4.dst)

        if i < (len(self.path) - 2):
            actions = [parser.OFPACTIONOutput(self.path[i + 1][1])]
        else:
            actions =
[parser.OFPACTIONSetField(eth_dst=self.ip_to_mac.get(pkt_ipv4.dst)),
parser.OFPACTIONOutput(self.path[i + 1][1])]

        self.add_flow(datapath_path, 100, match, actions, idle_timeout=5,
hard_timeout=5)
```

```

path : [(5, 3), (5, 1), (3, 2), (3, 1), (2, 2), (2, 1), (1, 1), (1, 3)]  FIN 10.0.0.2 (10.0.0.2) 58184 bytes of data.
path : [(1, 3), (1, 2), (4, 1), (4, 2), (5, 2), (5, 3)]          64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.263 ms
path : [(5, 3), (5, 2), (4, 2), (4, 1), (1, 2), (1, 3)]          64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.094 ms
path : [(1, 3), (1, 1), (2, 1), (2, 2), (3, 1), (3, 2), (5, 1), (5, 3)]  64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.532 ms
path : [(5, 3), (5, 1), (3, 2), (3, 1), (2, 2), (2, 1), (1, 1), (1, 3)]  64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.753 ms
path : [(1, 3), (1, 2), (4, 1), (4, 2), (5, 2), (5, 3)]          64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.091 ms
path : [(5, 3), (5, 2), (4, 2), (4, 1), (1, 2), (1, 3)]          64 bytes from 10.0.0.2: icmp_seq=14 ttl=64 time=0.072 ms
path : [(1, 3), (1, 1), (2, 1), (2, 2), (3, 1), (3, 2), (5, 1), (5, 3)]  64 bytes from 10.0.0.2: icmp_seq=15 ttl=64 time=0.050 ms
path : [(5, 3), (5, 1), (3, 2), (3, 1), (2, 2), (2, 1), (1, 1), (1, 3)]  64 bytes from 10.0.0.2: icmp_seq=17 ttl=64 time=0.095 ms
path : [(1, 3), (1, 2), (4, 1), (4, 2), (5, 2), (5, 3)]          64 bytes from 10.0.0.2: icmp_seq=19 ttl=64 time=0.111 ms
path : [(5, 3), (5, 2), (4, 2), (4, 1), (1, 2), (1, 3)]          64 bytes from 10.0.0.2: icmp_seq=20 ttl=64 time=0.128 ms
path : [(1, 3), (1, 1), (2, 1), (2, 2), (3, 1), (3, 2), (5, 1), (5, 3)]  64 bytes from 10.0.0.2: icmp_seq=22 ttl=64 time=0.083 ms
path : [(5, 3), (5, 2), (4, 2), (4, 1), (1, 2), (1, 3)]          64 bytes from 10.0.0.2: icmp_seq=24 ttl=64 time=0.080 ms
path : [(1, 3), (1, 1), (2, 1), (2, 2), (3, 1), (3, 2), (5, 1), (5, 3)]  64 bytes from 10.0.0.2: icmp_seq=25 ttl=64 time=0.068 ms
path : [(5, 3), (5, 1), (3, 2), (3, 1), (2, 2), (2, 1), (1, 1), (1, 3)]  64 bytes from 10.0.0.2: icmp_seq=27 ttl=64 time=0.121 ms
path : [(1, 3), (1, 2), (4, 1), (4, 2), (5, 2), (5, 3)]          64 bytes from 10.0.0.2: icmp_seq=29 ttl=64 time=0.080 ms
path : [(5, 3), (5, 2), (4, 2), (4, 1), (1, 2), (1, 3)]          64 bytes from 10.0.0.2: icmp_seq=30 ttl=64 time=0.063 ms
path : [(1, 3), (1, 1), (2, 1), (2, 2), (3, 1), (3, 2), (5, 1), (5, 3)]  64 bytes from 10.0.0.2: icmp_seq=31 ttl=64 time=0.100 ms
path : [(5, 3), (5, 2), (4, 2), (4, 1), (1, 2), (1, 3)]          64 bytes from 10.0.0.2: icmp_seq=34 ttl=64 time=0.098 ms
path : [(1, 3), (1, 1), (2, 1), (2, 2), (3, 1), (3, 2), (5, 1), (5, 3)]  64 bytes from 10.0.0.2: icmp_seq=35 ttl=64 time=0.075 ms
path : [(5, 3), (5, 1), (3, 2), (3, 1), (2, 2), (2, 1), (1, 1), (1, 3)]  64 bytes from 10.0.0.2: icmp_seq=37 ttl=64 time=0.049 ms
path : [(1, 3), (1, 2), (4, 1), (4, 2), (5, 2), (5, 3)]          64 bytes from 10.0.0.2: icmp_seq=40 ttl=64 time=0.058 ms
path : [(5, 3), (5, 2), (4, 2), (4, 1), (1, 2), (1, 3)]          64 bytes from 10.0.0.2: icmp_seq=42 ttl=64 time=0.125 ms
path : [(1, 3), (1, 1), (2, 1), (2, 2), (3, 1), (3, 2), (5, 1), (5, 3)]
path : [(5, 3), (5, 1), (3, 2), (3, 1), (2, 2), (2, 1), (1, 1), (1, 3)] ^C

```

思路以及代码：

Value	Explanation
OFFPFC_ADD	Adds new flow entries.
OFFPFC_MODIFY	Updates flow entries.
OFFPFC_MODIFY_STRICT	Update strictly matched flow entries
OFFPFC_DELETE	Deletes flow entries.
OFFPFC_DELETE_STRICT	Deletes strictly matched flow entries.

[illegible]


```

        instructions=inst)
    else:
        mod = parser.OFPFlowMod(datapath=datapath, priority=priority,
                                idle_timeout=idle_timeout,
                                hard_timeout=hard_timeout,
                                command=ofproto.OFPFC_DELETE,
                                match=match, instructions=inst)
        datapath.send_msg(mod)

```

3.构造 `get_OFPPortStatus_msg()` 函数。利用事件包 `EventOFPPortStatus` 中 `reason` 参数可以进行判断。对链路的up和down操作，会使得端口信息变化，一旦检测到 `reason==ofproto.OFPPR_MODIFY`，即删除流表项

```

@set_ev_cls(ofp_event.EventOFPPortStatus, [CONFIG_DISPATCHER,
MAIN_DISPATCHER, DEAD_DISPATCHER, HANDSHAKE_DISPATCHER])
def get_OFPPortStatus_msg(self, ev):
    msg=ev.msg
    datapath=ev.msg.datapath
    dpid = msg.datapath.id
    ofproto=datapath.ofproto
    parser=datapath.ofproto_parser
    port=msg.desc.port_no
    reason=msg.reason
    if (reason==ofproto.OFPPR_MODIFY):
        self.delete_flow(datapath=datapath,dst_port=port)

```

实验结果：

反复执行 `link s1 s4 down` 与 `link s1 s4 up`，如图：

```

instantiating app dynamic_rules_2.py of dynamic_rules
instantiating app ryu.controller.ofp_handler.OFPHandler
instantiating app ryu.topology.switches.Switches
(7982) wsgi starting up on http://0.0.0.0:8080
path : [(1, 3), (1, 2), (4, 1), (4, 2), (5, 2), (5, 3)]
path : [(5, 3), (5, 2), (4, 2), (4, 1), (1, 2), (1, 3)]
(4,1) delete flow!
(1,2) delete flow!
(1,2) delete flow!
(4,1) delete flow!
path : [(1, 3), (1, 1), (2, 1), (2, 2), (3, 1), (3, 2), (5, 1), (5, 3)]
path : [(5, 3), (5, 1), (3, 2), (3, 1), (2, 2), (2, 1), (1, 1), (1, 3)]
(1,2) delete flow!
(4,1) delete flow!
(1,2) delete flow!
(4,1) delete flow!
(1,2) delete flow!
path : [(1, 3), (1, 2), (4, 1), (4, 2), (5, 2), (5, 3)]
path : [(5, 3), (5, 2), (4, 2), (4, 1), (1, 2), (1, 3)]
(4,1) delete flow!
(4,1) delete flow!
(1,2) delete flow!
path : [(1, 3), (1, 1), (2, 1), (2, 2), (3, 1), (3, 2), (5, 1), (5, 3)]

Node: h1
64 bytes from 10.0.0.2: icmp_seq=67 ttl=64 time=0.126 ms
64 bytes from 10.0.0.2: icmp_seq=68 ttl=64 time=0.095 ms
64 bytes from 10.0.0.2: icmp_seq=69 ttl=64 time=0.093 ms
64 bytes from 10.0.0.2: icmp_seq=70 ttl=64 time=0.133 ms
64 bytes from 10.0.0.2: icmp_seq=71 ttl=64 time=0.102 ms
64 bytes from 10.0.0.2: icmp_seq=72 ttl=64 time=0.113 ms
64 bytes from 10.0.0.2: icmp_seq=73 ttl=64 time=0.093 ms
64 bytes from 10.0.0.2: icmp_seq=74 ttl=64 time=0.093 ms
64 bytes from 10.0.0.2: icmp_seq=75 ttl=64 time=0.091 ms
64 bytes from 10.0.0.2: icmp_seq=76 ttl=64 time=0.041 ms
64 bytes from 10.0.0.2: icmp_seq=77 ttl=64 time=0.096 ms
64 bytes from 10.0.0.2: icmp_seq=78 ttl=64 time=0.098 ms
64 bytes from 10.0.0.2: icmp_seq=79 ttl=64 time=0.073 ms
64 bytes from 10.0.0.2: icmp_seq=80 ttl=64 time=0.108 ms
64 bytes from 10.0.0.2: icmp_seq=81 ttl=64 time=0.090 ms
64 bytes from 10.0.0.2: icmp_seq=82 ttl=64 time=0.114 ms
64 bytes from 10.0.0.2: icmp_seq=83 ttl=64 time=0.096 ms
64 bytes from 10.0.0.2: icmp_seq=84 ttl=64 time=0.092 ms
64 bytes from 10.0.0.2: icmp_seq=85 ttl=64 time=0.078 ms
64 bytes from 10.0.0.2: icmp_seq=86 ttl=64 time=0.152 ms
64 bytes from 10.0.0.2: icmp_seq=87 ttl=64 time=0.115 ms
64 bytes from 10.0.0.2: icmp_seq=88 ttl=64 time=0.090 ms
64 bytes from 10.0.0.2: icmp_seq=89 ttl=64 time=0.097 ms
64 bytes from 10.0.0.2: icmp_seq=90 ttl=64 time=0.282 ms
[]
mininet> xterm h1
mininet> link s1 s4 down
mininet> link s1 s4 up
mininet> link s1 s4 down

```

三、 ping 的界面不停顿地执行，但会有延迟，为什么？

流表被删除后需要向控制器发送信息重新下发流表，因此会有延迟