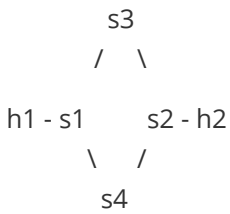
Lab6: RYU Open Flow Controller

This lab is about using ryu to manage network, and it mainly focuses on switch management. It is hard at first for I have never come across such kind of APIs, but with the help of ta and guides on the internet, I finally figure out it.

Topology

It requests to set up the following network first:



and the final result of my topology is:

```
mininet> nodes
available nodes are:
c1 h1 h2 s1 s2 s3 s4
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)
```

That's great!

ARP Storm

It's easy to know that this topology has a circle which may result in arp storm. So, we need to take measures to avoid arp storm. My method is: every switch only allows one inport to ask mac address. If an arp request comes from this allowing inport, firstly the switch looks at the arp table which stores the mac address of requesting ip, if we know the mac address, reply it to that port; if not, the switch just spreads this arp request out. If the arp request comes from other ports, firstly the switch looks at the arp table too, if we know the mac address, reply it to that port; if not, just drop this arp request.

The codes to perform an arp reply is:

```
if arp_dst_ip in self.arp_table:    # arp reply
    #actions = [parser.OFPActionOutput(ofproto.OFPP_FLOOD)]
    actions = [parser.OFPActionOutput(in_port)]
    ARP_Reply = packet.Packet()

    ARP_Reply.add_protocol(ethernet.ethernet(ethertype=eth.ethertype,
                                             dst=eth.src,
                                             src=self.arp_table[arp_dst_ip]))
    ARP_Reply.add_protocol(arp.arp(opcode=arp.ARP_REPLY,
                                   src_mac=self.arp_table[arp_dst_ip],
                                   src_ip=arp_dst_ip,
                                   dst_mac=eth_src, dst_ip=arp_src_ip))
```

```
    ARP_Reply.serialize()


    out = datapath.ofproto_parser.OFPPacketOut(datapath=datapath,

 buffer_id=datapath.ofproto.OFP_NO_BUFFER,

 in_port=datapath.ofproto.OFPP_CONTROLLER,
                                                 actions=actions,
data=ARP_Reply.data)
    datapath.send_msg(out)
```

Data Struct and Topology Learning in RYU Controller

The data struct used in ryu controller is:

```
def __init__(self, *args, **kwargs):
    super(ARP_PROXY_13, self).__init__(*args, **kwargs)
    self.mac_to_port = {} # out_port = self.mac_to_port[dpid][dst]
    self.arp_table = {} # arp_dst_ip in self.arp_table ---- store the mac address
of dst ip
    self.sw = {} # (datapath.id, arp_src_ip, arp_dst_ip) in self.sw
    self.dp = {} # switch id : datapath
```

The meanings are written in the notes.

In order to keep the global information, ryu contrller should perform several kinds of learning.

Outport of forwarding learning:

```
self.mac_to_port.setdefault(dpid, {})
# learn a mac address to avoid FLOOD next time.
if src not in self.mac_to_port[dpid]:
    self.mac_to_port[dpid][src] = in_port
```

Mac address learning:

```
# if there is a arp packet
arp_pkt = pkt.get_protocol(arp.arp) # when arp request happens, avoid arp storm

if arp_pkt:
    self.arp_table[arp_pkt.src_ip] = src        #ARP learning
```

To learn the fixed inport to ask arp request in a switch:

```
self.sw[(datapath.id, arp_src_ip, arp_dst_ip)] = in_port
```

Datapath learning(which denotes the switch identifier):

```
datapath = ev.msg.datapath
if datapath.id not in self.dp:
    self.dp[datapath.id] = datapath
```

Dynamic path change

This question requests switching paths (h1-s1-s3-s2-h2 or h1-s1-s4-s2-h2) between h1 and h2 every 5 seconds. At first, `CONFIG_DISPATCHER, MAIN_DISPATCHER` are two states in the process of connecting between ryu controller and switches. ryu passes down default flow table in state `CONFIG_DISPATCHER`, which allows switches pass forward packets to ryu controller when flow table miss happens. After state `CONFIG_DISPATCHER`, the connection steps into state `MAIN_DISPATCHER` in which ryu controller handles flow table miss, gives instructions to switches and add new flow table in switches.

Actions happen in state `CONFIG_DISPATCHER`:

```
@set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
def switch_features_handler(self, ev):
    datapath = ev.msg.datapath
    if datapath.id not in self.dp:
        self.dp[datapath.id] = datapath
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser

    # install table-miss flow entry
    #
    # we specify NO BUFFER to mac_len of the output action due to
    # OVS bug. At this moment, if we specify a lesser number, e.g.,
    # 128, OVS will send Packet-In with invalid buffer_id and truncated packet
data.
    # In that case, we cannot output packets correctly.
    match = parser.OFPMatch()
    actions = [parser.OFPActionOutput(ofproto.OFPP_CONTROLLER,
                                      ofproto.OFPCML_NO_BUFFER)]
    self.add_flow(datapath, 0, match, actions)
    self.logger.info("switch: %s connected", datapath.id) # datapath includes the
id and other information of switch
```

When ryu handles flow table miss, firstly, it should identify the kind of coming packet:

```
if eth.ethertype == ether_types.ETH_TYPE_LLDP:
    # ignore lldp packet
    return

if pkt.get_protocol(ipv6.ipv6):  # drop the ipv6 packets
    match = parser.OFPMatch(eth_type=eth.ethertype)
    actions = []
    self.add_flow(datapath, 1, match, actions)
    return None

# if there is a arp packet
arp_pkt = pkt.get_protocol(arp.arp) # when arp request happens, avoid arp storm

if arp_pkt:
    self.arp_table[arp_pkt.src_ip] = src        #ARP learning
```

```
    #self.logger.info(" ARP: %s -> %s", arp_pkt.src_ip, arp_pkt.dst_ip)
    if self.arp_handler(msg):               #answer or drop
        return None
```

Next, if ryu don't know which port to transimit, it asks the switch to flood. But if ryu knows where to transmit, it must install a flow table in the switch to avoid flood next time. The method to add flow is:

```
def add_flow(self, datapath, priority, match, actions, hard_timeout = 0):
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser

    inst = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS, actions)]

    if hard_timeout:
        mod = parser.OFPFlowMod(datapath=datapath,
                                priority=priority, match=match, hard_timeout =
hard_timeout,
                                instructions=inst)
    else:
        mod = parser.OFPFlowMod(datapath=datapath, priority=priority,
                                match=match, instructions=inst)
    datapath.send_msg(mod)
```

The parameter `hard_timeout` is used to shift forwarding pathes every 5 seconds. If `hard_timeout` is larger than zero, it means this flow table can just exist for `hard_timeout` seconds. So, we can set this value to five in switches `s1` and `s2`. After every five seconds, this flow table will become invalid, and thus table miss happens, ryu controller can install another path in `s1` and `s2`.

The method to perform different flow table installing is:

```
ipv_4 = pkt.get_protocol(ipv4.ipv4)
if ipv_4:
    if dpid == 3 or dpid == 4:
        out_port = 3 - in_port

    if dpid == 2:
        if in_port == 1:
            hard_timeout = 5
            if up_path:
                out_port = 2
                up_path = 0
                self.add_flow(self.dp[1], 1, parser.OFPMatch(in_port=in_port,
eth_dst=src), [parser.OFPActionOutput(out_port)], hard_timeout)
                self.logger.info("install flow_mod in datapath %s in switch %s:
%s -> %s",self.dp[1], 1, in_port, out_port)
            else:
                out_port = 3
                up_path = 1
                self.add_flow(self.dp[1], 1, parser.OFPMatch(in_port=in_port,
eth_dst=src), [parser.OFPActionOutput(out_port)], hard_timeout)
                self.logger.info("install flow_mod in datapath %s in switch %s:
%s -> %s",self.dp[1], 1, in_port, out_port)
        else:
            out_port = 1
```

```
    if dpid == 1:
        if in_port == 1:
            hard_timeout = 5
            if up_path:
                out_port = 2
                up_path = 0
                self.add_flow(self.dp[2], 1, parser.OFPMatch(in_port=in_port,
eth_dst=src), [parser.OFPActionOutput(out_port)], hard_timeout)
                self.logger.info("install flow_mod in datapath %s in switch %s:
%s -> %s",self.dp[2], 2, in_port, out_port)
            else:
                out_port = 3
                up_path = 1
                self.add_flow(self.dp[2], 1, parser.OFPMatch(in_port=in_port,
eth_dst=src), [parser.OFPActionOutput(out_port)], hard_timeout)
                self.logger.info("install flow_mod in datapath %s in switch %s:
%s -> %s",self.dp[2], 2, in_port, out_port)
        else:
            out_port = 1

actions = [parser.OFPActionOutput(out_port)]

# install a flow to avoid packet_in next time.
if out_port != ofproto.OFPP_FLOOD:
    self.logger.info("install flow_mod in datapath %s in switch %s: %s ->
%s",datapath, dpid, in_port, out_port)
    match = parser.OFPMatch(in_port=in_port, eth_dst=dst)
    self.add_flow(datapath, 1, match, actions, hard_timeout)

data = None
if msg.buffer_id == ofproto.OFP_NO_BUFFER:
    data = msg.data
out = parser.OFPPacketOut(datapath=datapath, buffer_id=msg.buffer_id,
                          in_port=in_port, actions=actions, data=data)

datapath.send_msg(out)
```

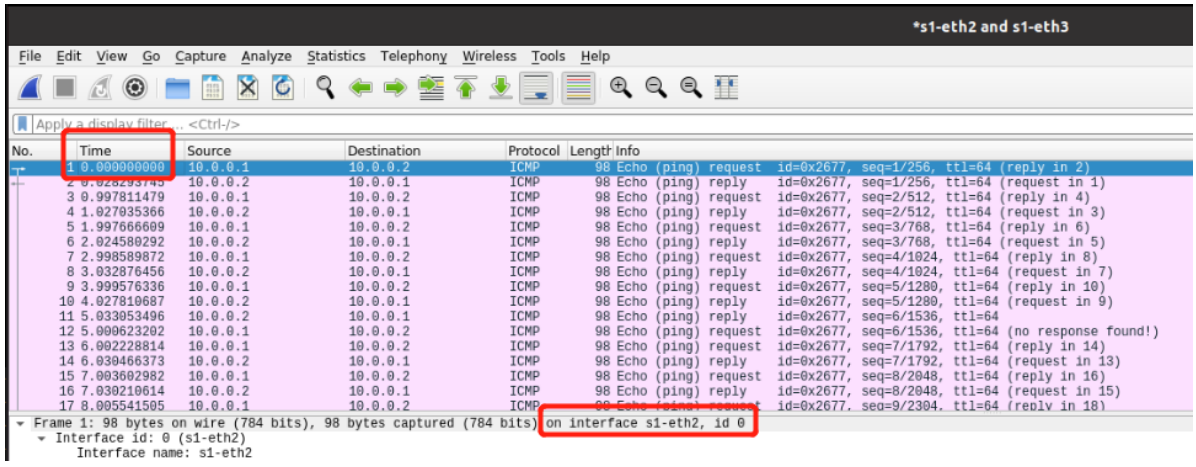
The value of `dpid` denotes switch number, so we can construct different flow table in different switches.

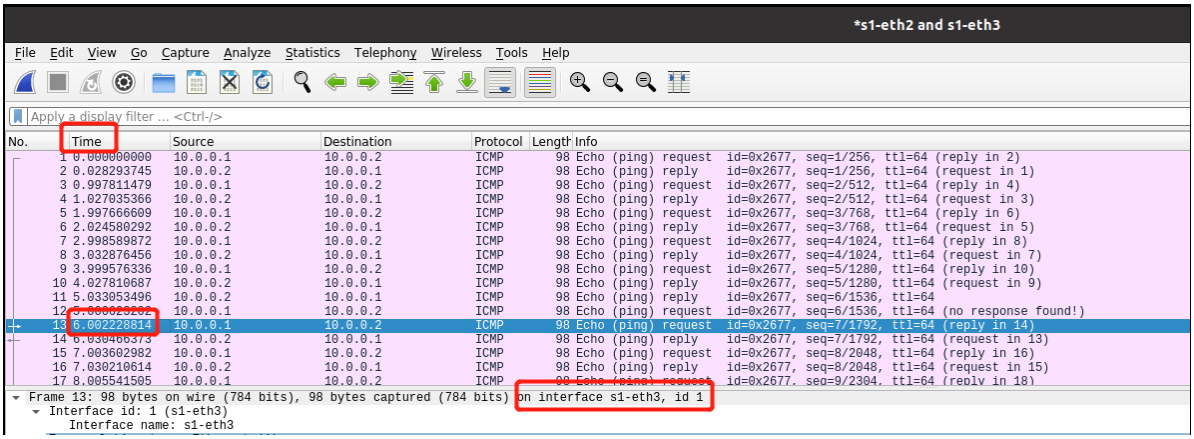
The result of flow table installing is:

```
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7fad3c0387c0> in switch
2: 1 -> 2
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7fad3c038970> in switch
1: 1 -> 2
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:3 source:a6:3a:f2:1e:d6:b2 destination:ff:ff:ff:ff:ff:ff in port:1
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7fad3c026730> in switch
3: 1 -> 2
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:2 source:a6:3a:f2:1e:d6:b2 destination:ff:ff:ff:ff:ff:ff in port:2
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7fad3c0387c0> in switch
2: 2 -> 1
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:3 source:62:b5:bb:d8:b7:fe destination:a6:3a:f2:1e:d6:b2 in port:2
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7fad3c026730> in switch
3: 2 -> 1
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:1 source:62:b5:bb:d8:b7:fe destination:a6:3a:f2:1e:d6:b2 in port:2
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7fad3c038970> in switch
1: 2 -> 1
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:1 source:a6:3a:f2:1e:d6:b2 destination:62:b5:bb:d8:b7:fe in port:1
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7fad3c0387c0> in switch
2: 1 -> 3
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7fad3c038970> in switch
1: 1 -> 3
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:4 source:a6:3a:f2:1e:d6:b2 destination:62:b5:bb:d8:b7:fe in port:1
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7fad3c99cac0> in switch
4: 1 -> 2
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:2 source:a6:3a:f2:1e:d6:b2 destination:62:b5:bb:d8:b7:fe in port:3
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7fad3c0387c0> in switch
2: 3 -> 1
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:4 source:62:b5:bb:d8:b7:fe destination:a6:3a:f2:1e:d6:b2 in port:2
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7fad3c99cac0> in switch
4: 2 -> 1
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:1 source:62:b5:bb:d8:b7:fe destination:a6:3a:f2:1e:d6:b2 in port:3
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7fad3c038970> in switch
1: 3 -> 1
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:1 source:a6:3a:f2:1e:d6:b2 destination:62:b5:bb:d8:b7:fe in port:1
```

The red rectangle shows the process of flow table installing.

We can also test it by wireshark:





Load Balance

It's similar with dymanic path change, but load balance needs buckets to store several actions to define different ways to perform load balance. As a result, ryu can not only pass a single flow table to switches, but a group flow table.

The method to install group flow table in switches is:

```
def add_group_flow(self, datapath, priority, match):
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser

    port = 2
    actions1 = [parser.OFPActionOutput(port)]
    port = 3
    actions2 = [parser.OFPActionOutput(port)]

    weight1 = 50
    weight2 = 50
    watch_port = ofproto_v1_3.OFPP_ANY
    watch_group = ofproto_v1_3.OFPQ_ALL

    # to perform different actions to make a load balance
    buckets = [parser.OFPBucket(weight1, watch_port, watch_group,
                                actions1),
               parser.OFPBucket(weight2, watch_port, watch_group,
                                actions2)]

    group_id = 1 # to identify the group flow table

    # add the group flow table
    req = parser.OFPGroupMod(datapath, ofproto.OFPGC_ADD,
                             ofproto.OFPGT_SELECT, group_id, buckets)
    datapath.send_msg(req)

    actions = [parser.OFPActionGroup(group_id=group_id)]
    inst = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,
                                         actions)]
    mod = parser.OFPFlowMod(datapath=datapath, priority=priority,
                            match=match, instructions=inst)
    datapath.send_msg(mod)
```

According to the value of `dpid`, ryu controller can perform different flow table install in different switches:

```
ipv_4 = pkt.get_protocol(ipv4.ipv4)
if ipv_4:
#if dst != "ff:ff:ff:ff:ff:ff":
    if dpid == 3 or dpid == 4:
        out_port = 3 - in_port

    if dpid == 2:
        if in_port == 1:
            self.add_group_flow(datapath, 2, parser.OFPMatch(in_port=in_port,
eth_dst=dst))
            self.add_group_flow(self.dp[1], 2, parser.OFPMatch(in_port=in_port,
eth_dst=src))

            data = None
            if msg.buffer_id == ofproto.OFP_NO_BUFFER:
                data = msg.data
            out = parser.OFPPacketOut(datapath=datapath, buffer_id=msg.buffer_id,
                                      in_port=in_port, actions=[parser.OFPActionOutput(2)],
data=data)
            datapath.send_msg(out)

            return

        else:
            out_port = 1

    if dpid == 1:
        if in_port == 1:
            self.add_group_flow(datapath, 2, parser.OFPMatch(in_port=in_port,
eth_dst=dst))
            self.add_group_flow(self.dp[2], 2, parser.OFPMatch(in_port=in_port,
eth_dst=src))

            data = None
            if msg.buffer_id == ofproto.OFP_NO_BUFFER:
                data = msg.data
            out = parser.OFPPacketOut(datapath=datapath, buffer_id=msg.buffer_id,
                                      in_port=in_port, actions=[parser.OFPActionOutput(2)],
data=data)
            datapath.send_msg(out)

            return

        else:
            out_port = 1
```

And the result is as follows:

```
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:1 source:aa:50:f9:fd:6c:a3 destination:ff:ff:ff:ff:ff:ff in port:1
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:4 source:aa:50:f9:fd:6c:a3 destination:ff:ff:ff:ff:ff:ff in port:1
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:3 source:aa:50:f9:fd:6c:a3 destination:ff:ff:ff:ff:ff:ff in port:1
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:2 source:aa:50:f9:fd:6c:a3 destination:ff:ff:ff:ff:ff:ff in port:2
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:2 source:ea:ae:a4:8b:a8:37 destination:aa:50:f9:fd:6c:a3 in port:1
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7f39db5ca970> in switch
2: 1 -> 2
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:3 source:ea:ae:a4:8b:a8:37 destination:aa:50:f9:fd:6c:a3 in port:2
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7f39db5ec6d0> in switch
3: 2 -> 1
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:1 source:ea:ae:a4:8b:a8:37 destination:aa:50:f9:fd:6c:a3 in port:2
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7f39db5ca9a0> in switch
1: 2 -> 1
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:1 source:aa:50:f9:fd:6c:a3 destination:ea:ae:a4:8b:a8:37 in port:1
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:3 source:aa:50:f9:fd:6c:a3 destination:ea:ae:a4:8b:a8:37 in port:1
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7f39db5ec6d0> in switch
3: 1 -> 2
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:2 source:aa:50:f9:fd:6c:a3 destination:ea:ae:a4:8b:a8:37 in port:2
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7f39db5ca970> in switch
2: 2 -> 1
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:4 source:aa:50:f9:fd:6c:a3 destination:ea:ae:a4:8b:a8:37 in port:1
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7f39db5ecf10> in switch
4: 1 -> 2
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:2 source:aa:50:f9:fd:6c:a3 destination:ea:ae:a4:8b:a8:37 in port:3
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7f39db5ca970> in switch
2: 3 -> 1
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:4 source:ea:ae:a4:8b:a8:37 destination:aa:50:f9:fd:6c:a3 in port:2
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7f39db5ecf10> in switch
4: 2 -> 1
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:1 source:ea:ae:a4:8b:a8:37 destination:aa:50:f9:fd:6c:a3 in port:3
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7f39db5ca9a0> in switch
1: 3 -> 1
```

The red rectangle shows the process of flow table installing.

```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
 cookie=0x0, duration=51.623s, table=0, n_packets=50, n_bytes=7224, priority=1,ipv6 actions=drop
 cookie=0x0, duration=45.627s, table=0, n_packets=39, n_bytes=3822, priority=1,in_port="s1-eth3",dl_d
st=2e:b7:28:92:27:7a actions=output:"s1-eth1"
 cookie=0x0, duration=45.613s, table=0, n_packets=39, n_bytes=3766, priority=1,in_port="s1-eth1",dl_d
st=b6:89:40:e3:47:52 actions=group:1
 cookie=0x0, duration=2.374s, table=0, n_packets=0, n_bytes=0, priority=1,in_port="s1-eth2",dl_dst=2e
:b7:28:92:27:7a actions=output:"s1-eth1"
 cookie=0x0, duration=51.796s, table=0, n_packets=5, n_bytes=310, priority=0 actions=CONTROLLER:65535
*** s2 ------------------------------------------------------------------------
 cookie=0x0, duration=51.436s, table=0, n_packets=48, n_bytes=7048, priority=1,ipv6 actions=drop
 cookie=0x0, duration=45.618s, table=0, n_packets=42, n_bytes=3948, priority=1,in_port="s2-eth1",dl_d
st=2e:b7:28:92:27:7a actions=group:1
 cookie=0x0, duration=45.603s, table=0, n_packets=3, n_bytes=162, priority=1,in_port="s2-eth2",dl_dst
=b6:89:40:e3:47:52 actions=output:"s2-eth1"
 cookie=0x0, duration=44.661s, table=0, n_packets=37, n_bytes=3626, priority=1,in_port="s2-eth3",dl_d
st=b6:89:40:e3:47:52 actions=output:"s2-eth1"
 cookie=0x0, duration=51.725s, table=0, n_packets=6, n_bytes=412, priority=0 actions=CONTROLLER:65535
*** s3 ------------------------------------------------------------------------
 cookie=0x0, duration=51.633s, table=0, n_packets=42, n_bytes=6588, priority=1,ipv6 actions=drop
 cookie=0x0, duration=45.615s, table=0, n_packets=1, n_bytes=42, priority=1,in_port="s3-eth1",dl_dst=
b6:89:40:e3:47:52 actions=output:"s3-eth2"
 cookie=0x0, duration=2.391s, table=0, n_packets=1, n_bytes=42, priority=1,in_port="s3-eth2",dl_dst=2
e:b7:28:92:27:7a actions=output:"s3-eth1"
 cookie=0x0, duration=51.660s, table=0, n_packets=6, n_bytes=352, priority=0 actions=CONTROLLER:65535
*** s4 ------------------------------------------------------------------------
 cookie=0x0, duration=51.482s, table=0, n_packets=42, n_bytes=6580, priority=1,ipv6 actions=drop
 cookie=0x0, duration=45.649s, table=0, n_packets=39, n_bytes=3822, priority=1,in_port="s4-eth2",dl_d
st=2e:b7:28:92:27:7a actions=output:"s4-eth1"
 cookie=0x0, duration=44.679s, table=0, n_packets=37, n_bytes=3626, priority=1,in_port="s4-eth1",dl_d
st=b6:89:40:e3:47:52 actions=output:"s4-eth2"
 cookie=0x0, duration=51.606s, table=0, n_packets=4, n_bytes=272, priority=0 actions=CONTROLLER:65535
```

Flow tables in each switches also denote its correct.

And the bandwidth between `s1` and `s2` is:

```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['8.03 Mbits/sec', '8.90 Mbits/sec']
```

Fast Failover

The question is: write a ryu controller that uses the first path (h1-s1-s3-s2-h2) for routing packets from h1 to h2 and uses the second path for backup. Specifically, when the first path experiences a link failure, the network should automatically switch to the second path without causing packet drop. (hint: consider using OFPGT_FF (FF is short for "fast failover") to construct a group table)

It's similar to load balance, we just need to modify the `add_group_flow` to change the `buckets` and `type`. The codes modified is as follows:

```
def add_group_flow(self, datapath, priority, match, group_id = 1):
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser

    port = 2
    actions1 = [parser.OFPActionOutput(port)]
    port = 3
    actions2 = [parser.OFPActionOutput(port)]

    #watch_port = ofproto_v1_3.OFPP_ANY
    #watch_group = ofproto_v1_3.OFPQ_ALL
    if group_id == 1:
        buckets = [parser.OFPBucket(watch_port = 2,
                                    actions = actions1),
                   parser.OFPBucket(watch_port = 3,
                                    actions = actions2)]
    else:
        buckets = [parser.OFPBucket(watch_port = 3,
                                    actions = actions2),
                   parser.OFPBucket(watch_port = 2,
                                    actions = actions1)]

    req = parser.OFPGroupMod(datapath, ofproto.OFPGC_ADD,
                             ofproto.OFPGT_FF, group_id, buckets)
    datapath.send_msg(req)

    actions = [parser.OFPActionGroup(group_id=group_id)]
    inst = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,
                                         actions)]
    mod = parser.OFPFlowMod(datapath=datapath, priority=priority,
                            match=match, instructions=inst)
    datapath.send_msg(mod)
```

The result of fast failover is as follows:

The order of installing flow table when no link down:

```
packet in: switch:2 source:d2:b4:b6:ae:3f:6a destination:de:9e:36:4e:ad:33 in port:1
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7f4ebc7eca30> in switch
2: 1 -> 3
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:4 source:d2:b4:b6:ae:3f:6a destination:de:9e:36:4e:ad:33 in port:2
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7f4ebc80ef10> in switch
4: 2 -> 1
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:1 source:d2:b4:b6:ae:3f:6a destination:de:9e:36:4e:ad:33 in port:3
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7f4ebc7eca00> in switch
1: 3 -> 1
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:1 source:de:9e:36:4e:ad:33 destination:d2:b4:b6:ae:3f:6a in port:1
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:3 source:de:9e:36:4e:ad:33 destination:d2:b4:b6:ae:3f:6a in port:1
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7f4ebc80e6d0> in switch
3: 1 -> 2
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:2 source:de:9e:36:4e:ad:33 destination:d2:b4:b6:ae:3f:6a in port:2
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7f4ebc7eca30> in switch
2: 2 -> 1
```

The flow table of each switches is:

```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
 cookie=0x0, duration=51.719s, table=0, n_packets=50, n_bytes=7216, priority=1,ipv6 actions=drop
 cookie=0x0, duration=46.969s, table=0, n_packets=45, n_bytes=4298, priority=1,in_port="s1-eth3",dl_d
st=de:9e:36:4e:ad:33 actions=output:"s1-eth1"
 cookie=0x0, duration=46.949s, table=0, n_packets=44, n_bytes=4200, priority=1,in_port="s1-eth1",dl_d
st=d2:b4:b6:ae:3f:6a actions=group:1
 cookie=0x0, duration=51.865s, table=0, n_packets=4, n_bytes=272, priority=0 actions=CONTROLLER:65535
*** s2 ------------------------------------------------------------------------
 cookie=0x0, duration=51.697s, table=0, n_packets=49, n_bytes=7134, priority=1,ipv6 actions=drop
 cookie=0x0, duration=46.954s, table=0, n_packets=45, n_bytes=4298, priority=1,in_port="s2-eth1",dl_d
st=de:9e:36:4e:ad:33 actions=group:2
 cookie=0x0, duration=46.938s, table=0, n_packets=44, n_bytes=4200, priority=1,in_port="s2-eth2",dl_d
st=d2:b4:b6:ae:3f:6a actions=output:"s2-eth1"
 cookie=0x0, duration=51.795s, table=0, n_packets=5, n_bytes=310, priority=0 actions=CONTROLLER:65535
*** s3 ------------------------------------------------------------------------
 cookie=0x0, duration=51.320s, table=0, n_packets=40, n_bytes=6408, priority=1,ipv6 actions=drop
 cookie=0x0, duration=46.952s, table=0, n_packets=44, n_bytes=4200, priority=1,in_port="s3-eth1",dl_d
st=d2:b4:b6:ae:3f:6a actions=output:"s3-eth2"
 cookie=0x0, duration=51.726s, table=0, n_packets=4, n_bytes=272, priority=0 actions=CONTROLLER:65535
*** s4 ------------------------------------------------------------------------
 cookie=0x0, duration=51.521s, table=0, n_packets=42, n_bytes=6584, priority=1,ipv6 actions=drop
 cookie=0x0, duration=46.992s, table=0, n_packets=45, n_bytes=4298, priority=1,in_port="s4-eth2",dl_d
st=de:9e:36:4e:ad:33 actions=output:"s4-eth1"
 cookie=0x0, duration=51.666s, table=0, n_packets=3, n_bytes=174, priority=0 actions=CONTROLLER:65535
```

The bandwidth between `s1` and `s2` without link down is:

```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['8.38 Mbits/sec', '8.85 Mbits/sec']
```

After run `link s1 s3 down` in the command line of mininet, the process of installing flow table adds two things:

```
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:2 source:d2:b4:b6:ae:3f:6a destination:de:9e:36:4e:ad:33 in port:1
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7f4ebc7eca30> in switch
2: 1 -> 3
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:4 source:d2:b4:b6:ae:3f:6a destination:de:9e:36:4e:ad:33 in port:2
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7f4ebc80ef10> in switch
4: 2 -> 1
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:1 source:d2:b4:b6:ae:3f:6a destination:de:9e:36:4e:ad:33 in port:3
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7f4ebc7eca00> in switch
1: 3 -> 1
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:1 source:de:9e:36:4e:ad:33 destination:d2:b4:b6:ae:3f:6a in port:1
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:3 source:de:9e:36:4e:ad:33 destination:d2:b4:b6:ae:3f:6a in port:1
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7f4ebc80e6d0> in switch
3: 1 -> 2
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:2 source:de:9e:36:4e:ad:33 destination:d2:b4:b6:ae:3f:6a in port:2
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7f4ebc7eca30> in switch
2: 2 -> 1
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:4 source:de:9e:36:4e:ad:33 destination:d2:b4:b6:ae:3f:6a in port:1
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7f4ebc80ef10> in switch
4: 1 -> 2
EVENT ofp_event->ARP_PROXY_13 EventOFPPacketIn
packet in: switch:2 source:de:9e:36:4e:ad:33 destination:d2:b4:b6:ae:3f:6a in port:3
install flow_mod in datapath <ryu.controller.controller.Datapath object at 0x7f4ebc7eca30> in switch
2: 3 -> 1
```

And the flow table of each switches changes to:

```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
 cookie=0x0, duration=263.164s, table=0, n_packets=62, n_bytes=8721, priority=1,ipv6 actions=drop
 cookie=0x0, duration=258.414s, table=0, n_packets=3998, n_bytes=266276, priority=1,in_port="s1-eth3"
,dl_dst=de:9e:36:4e:ad:33 actions=output:"s1-eth1"
 cookie=0x0, duration=258.394s, table=0, n_packets=4174, n_bytes=11812196, priority=1,in_port="s1-eth
1",dl_dst=d2:b4:b6:ae:3f:6a actions=group:1
 cookie=0x0, duration=263.310s, table=0, n_packets=4, n_bytes=272, priority=0 actions=CONTROLLER:6553
5
*** s2 ------------------------------------------------------------------------
 cookie=0x0, duration=263.142s, table=0, n_packets=63, n_bytes=8912, priority=1,ipv6 actions=drop
 cookie=0x0, duration=258.399s, table=0, n_packets=3998, n_bytes=266276, priority=1,in_port="s2-eth1"
,dl_dst=de:9e:36:4e:ad:33 actions=group:2
 cookie=0x0, duration=258.383s, table=0, n_packets=2090, n_bytes=5906420, priority=1,in_port="s2-eth2
",dl_dst=d2:b4:b6:ae:3f:6a actions=output:"s2-eth1"
 cookie=0x0, duration=48.683s, table=0, n_packets=2083, n_bytes=5905678, priority=1,in_port="s2-eth3"
,dl_dst=d2:b4:b6:ae:3f:6a actions=output:"s2-eth1"
 cookie=0x0, duration=263.241s, table=0, n_packets=6, n_bytes=408, priority=0 actions=CONTROLLER:6553
5
*** s3 ------------------------------------------------------------------------
 cookie=0x0, duration=262.766s, table=0, n_packets=50, n_bytes=7773, priority=1,ipv6 actions=drop
 cookie=0x0, duration=258.398s, table=0, n_packets=2090, n_bytes=5906420, priority=1,in_port="s3-eth1
",dl_dst=d2:b4:b6:ae:3f:6a actions=output:"s3-eth2"
 cookie=0x0, duration=263.172s, table=0, n_packets=4, n_bytes=272, priority=0 actions=CONTROLLER:6553
5
*** s4 ------------------------------------------------------------------------
 cookie=0x0, duration=262.966s, table=0, n_packets=53, n_bytes=8152, priority=1,ipv6 actions=drop
 cookie=0x0, duration=258.437s, table=0, n_packets=3998, n_bytes=266276, priority=1,in_port="s4-eth2"
,dl_dst=de:9e:36:4e:ad:33 actions=output:"s4-eth1"
 cookie=0x0, duration=48.700s, table=0, n_packets=2083, n_bytes=5905678, priority=1,in_port="s4-eth1"
,dl_dst=d2:b4:b6:ae:3f:6a actions=output:"s4-eth2"
 cookie=0x0, duration=263.111s, table=0, n_packets=4, n_bytes=272, priority=0 actions=CONTROLLER:6553
5
mininet>
```

The bandwidth between `s1` and `s2` after link down is:

```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['8.30 Mbits/sec', '8.79 Mbits/sec']
```

Reflection

My codes aren't plantable because I know the topology and other needed information of the network, so I can directly assign instructions to each switch. But my codes can't run on another network. One way to solve this problem is to write functions to enable ryu controller finds topology by itself. For example:

```
def get_switch(app, dpid=None):
    rep = app.send_request(event.EventSwitchRequest(dpid))
    return rep.switches


def get_link(app, dpid=None):
    rep = app.send_request(event.EventLinkRequest(dpid))
    return rep.links
```

So that ryu controller can write more general forwarding methods.