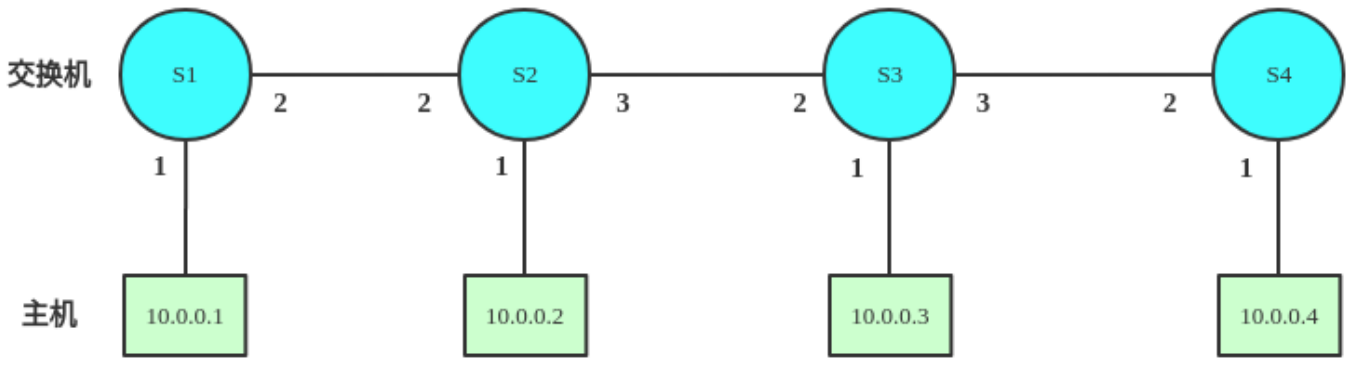
Ryu的应用开发（五）网络拓扑发现

一：实验简介

（一）网络拓扑信息：



其中1,2,3表示该交换机对应的端口号！！！

（二）用邻接矩阵展示

入 \ 出	S1	S2	S3	S4
S1	0	(2,0)	0	0
S2	(2,0)	0	(3,0)	0
S3	0	(2,0)	0	(3,0)
S4	0	0	(2,0)	0

其中左侧列S1,S2,S3,S4表示出节点，----->，上面S1,S2,S3,S4表示入节点。（m,0），m表示出节点的端口--->入节点，0暂时表示两个节点之间的时延信息！

（三）主机信息展示

```
    IP          交换机
['10.0.0.1']------s1
['10.0.0.2']------s2
['10.0.0.3']------s3
['10.0.0.4']------s4
```

二：代码实现

（一）导入模块

Python | 复制代码

```python
from ryu.base import app_manager

from ryu.ofproto import ofproto_v1_3

from ryu.controller import ofp_event
from ryu.controller.handler import MAIN_DISPATCHER,CONFIG_DISPATCHER,DEAD_DISPATCHER   #只是表示datapath数据路径的状态
from ryu.controller.handler import set_ev_cls

from ryu.lib import hub
from ryu.lib.packet import packet,ethernet

from ryu.topology import event,switches
from ryu.topology.api import get_switch,get_link,get_host

import threading     #需要设置线程锁
```

（二）数据结构

Python | 复制代码

```python
DELAY_MONITOR_PERIOD = 5
LOCK = threading.RLock() #实现线程锁

class TopoDetect(app_manager.RyuApp):
    OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]

    def __init__(self,*args,**kwargs):
        super(TopoDetect,self).__init__(*args,**kwargs)
        self.topology_api_app = self  #用于保持对象本身，后面get_switch等方法需要（我们也可以直接传入self）
        self.link_list = None         #保存所有的link信息，由get_link获得
        self.switch_list = None       #保存所有的switch信息，由get_switch获得
        self.host_list = None         #保存所有的host信息，由get_host获得

        self.dpid2id = {}             #获取交换机dpid,以及自定义id--->{dpid:id}
        self.id2dpid = {}             #对应上面的self.dpid2id,翻转即可，因为我们使用id进行建立邻接矩阵，这两个结构方便查找
        self.dpid2switch = {}         #保存dpid和对应的交换机全部信息---->通过矩阵获得id,然后获得dpid,最后获得交换机对象信息

        self.ip2host = {}             #根据ip,保存主机对象信息--->{ip:host}
        self.ip2switch = {}           #根据ip,获取当前主机是连接到哪个交换机--->{ip:dpid}

        self.net_size = 0             #记录交换机个数（网络拓扑大小）
        self.net_topo = []            #用于保存邻接矩阵

        self.net_flag = False         #标识：用于表示拓扑网络拓扑self.net_topo是否已经更新完成
        self.net_arrived = 0          #标识：用于表示网络中交换机消息到达，每当一个交换机到达以后，我们设置+1
        self.monitor_thread = hub.spawn(self._monitor)  #协程实现定时检测网络拓扑
```

（三）实现基本openflow消息处理

Python | 复制代码

```python
    @set_ev_cls(ofp_event.EventOFPSwitchFeatures,CONFIG_DISPATCHER)
    def switch_feature_handle(self,ev):
        """
        datapath中有配置消息到达
        """
        LOCK.acquire()
        self.net_arrived += 1 #表示有1个交换机到达
        LOCK.release()
        #print("------XXXXXXXXXX------%d------XXXXXXXXXX----------switch_feature_handle"%self.net_arrived)
        msg = ev.msg
        datapath = msg.datapath
        ofproto = datapath.ofproto
        ofp_parser = datapath.ofproto_parser

        match = ofp_parser.OFPMatch()

        actions = [ofp_parser.OFPActionOutput(ofproto.OFPP_CONTROLLER,ofproto.OFPCML_NO_BUFFER)]

        self.add_flow(datapath=datapath,priority=0,match=match,actions=actions,extra_info="config infomation arrived!!")


    def add_flow(self,datapath,priority,match,actions,idle_timeout=0,hard_timeout=0,extra_info=None):
        #print("-----------------add_flow:")
        if extra_info != None:
            print(extra_info)
        ofproto = datapath.ofproto
        ofp_parser = datapath.ofproto_parser

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

        mod = ofp_parser.OFPFlowMod(datapath=datapath,priority=priority,
                                    idle_timeout=idle_timeout,
                                    hard_timeout=hard_timeout,
                                    match=match,instructions=inst)
        datapath.send_msg(mod);

    @set_ev_cls(ofp_event.EventOFPPacketIn,MAIN_DISPATCHER)
    def packet_in_handler(self,ev):
        #print("-----------------packet_in_handler")
        msg = ev.msg
```

```
        datapath = msg.datapath
        ofproto = datapath.ofproto
        ofp_parser = datapath.ofproto_parser

        dpid = datapath.id
        in_port = msg.match['in_port']

        pkt = packet.Packet(msg.data)
        eth_pkt = pkt.get_protocol(ethernet.ethernet)
        dst = eth_pkt.dst
        src = eth_pkt.src

        #self.logger.info("------------------Controller %s get packet, Mac address from: %s send to: %s , send from datapath: %s,in port is: %s"
        #                  ,dpid,src,dst,dpid,in_port)
        #self.get_topology(None)
```

注意：对于packet_in消息，我们没有处理，所以整个网络（交换机之间的链路）是无法工作通信的，

但是各个交换机可以与控制器通信（switch_feature_handle实现），主机也可以和边缘交换机通信，

所以控制器可以获取网络拓扑信息！！！

（四）实现拓扑发现功能

Python | 复制代码

```python
    def _monitor(self):
        """
        协程实现伪并发，探测拓扑状态
        """
        while True:
            #print("------------------_monitor")
            self._host_add_handler(None) #主机单独提取处理
            self.get_topology(None)
            if self.net_flag:
                try:
                    self.show_topology()
                except Exception as err:
                    print("Please use cmd: pingall to detect topology and wait a moment")
                hub.sleep(DELAY_MONITOR_PERIOD) #5秒一次
```

Python | 复制代码

```python
    @set_ev_cls([event.EventHostAdd])
    def _host_add_handler(self,ev):    #主机信息单独处理，不属于网络拓扑
        self.host_list = get_host(self.topology_api_app)  #3.需要使用pingall,主机通过与边缘交换机连接，才能告诉控制器
        #获取主机信息字典ip2host{ipv4:host object}  ip2switch{ipv4:dpid}
        for i,host in enumerate(self.host_list):
            self.ip2switch["%s"%host.ipv4] = host.port.dpid
            self.ip2host["%s"%host.ipv4] = host


    events = [event.EventSwitchEnter, event.EventSwitchLeave,
              event.EventSwitchReconnected,
              event.EventPortAdd, event.EventPortDelete,
              event.EventPortModify,
              event.EventLinkAdd, event.EventLinkDelete]
    @set_ev_cls(events)
    def get_topology(self,ev):
        if not self.net_arrived:
            return

        LOCK.acquire()
        self.net_arrived -= 1
        if self.net_arrived < 0:
            self.net_arrived = 0
        LOCK.release()

        self.net_flag = False
        self.net_topo = []

        print("-----------------get_topology")
        #获取所有的交换机、链路
        self.switch_list = get_switch(self.topology_api_app)  #1.只要交换机与控制器联通，就可以获取
        self.link_list = get_link(self.topology_api_app)      #2.在ryu启动时，加上--observe-links即可用于拓扑发现

        #获取交换机字典id2dpid{id:dpid} dpid2switch{dpid:switch object}
        for i,switch in enumerate(self.switch_list):
            self.id2dpid[i] = switch.dp.id
            self.dpid2id[switch.dp.id] = i
            self.dpid2switch[switch.dp.id] = switch


        #根据链路信息，开始获取拓扑信息
        self.net_size = len(self.id2dpid) #表示网络中交换机个数
```

```python
        for i in range(self.net_size):
            self.net_topo.append([0]*self.net_size)

        for link in self.link_list:
            src_dpid = link.src.dpid
            src_port = link.src.port_no

            dst_dpid = link.dst.dpid
            dst_port = link.dst.port_no

            try:
                sid = self.dpid2id[src_dpid]
                did = self.dpid2id[dst_dpid]
            except KeyError as e:
                print("--------------Error:get KeyError with link infomation(%s)"%e)
                return
            self.net_topo[sid][did] = [src_port,0] #注意：这里0表示存在链路，后面可以修改为时延
            self.net_topo[did][sid] = [dst_port,0] #注意：修改为列表，不要用元组，元组无法修改，我们后面要修改时延


        self.net_flag = True #表示网络拓扑创建成功
```

Python | 复制代码

```python
def show_topology(self):
    print("-----------------show_topology")
    print("----------switch network----------")
    line_info = "         "
    for i in range(self.net_size):
        line_info+="        s%-5d        "%self.id2dpid[i]
        print(line_info)
        for i in range(self.net_size):
            line_info = "s%d      "%self.id2dpid[i]
            for j in range(self.net_size):
                if self.net_topo[i][j] == 0:
                    line_info+="%-22d"%0
                else:
                    line_info+="(%d,%.12f)    "%tuple(self.net_topo[i][j])
                    print(line_info)

                    print("----------host 2 switch----------")
                    for key,val in self.ip2switch.items():
            print("%s---s%d"%(key,val))
```

（五）全部代码

Python | 复制代码

```python

from ryu.topology import event,switches
from ryu.topology.api import get_switch,get_link,get_host

import threading,time,random

DELAY_MONITOR_PERIOD = 5

class TopoDetect(app_manager.RyuApp):
    OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]

    def __init__(self,*args,**kwargs):
        super(TopoDetect,self).__init__(*args,**kwargs)
        self.topology_api_app = self
        self.name = "topology"
        self.link_list = None
        self.switch_list = None
        self.host_list = None

        self.dpid2id = {}
        self.id2dpid = {}
        self.dpid2switch = {}

        self.ip2host = {}
        self.ip2switch = {}

        self.net_size = 0
        self.net_topo = []

        self.net_flag = False
        self.net_arrived = 0

        self.monitor_thread = hub.spawn(self._monitor)


    def _monitor(self):
        """
        协程实现伪并发，探测拓扑状态
        """
        while True:
            #print("-----------------_monitor")
            self._host_add_handler(None) #主机单独提取处理
            self.get_topology(None)
            if self.net_flag:
                try:
                    self.show_topology()
```

```
                except Exception as err:
                    print("Please use cmd: pingall to detect topology and wait a moment")
            hub.sleep(DELAY_MONITOR_PERIOD) #5秒一次


    @set_ev_cls(ofp_event.EventOFPSwitchFeatures,CONFIG_DISPATCHER)
    def switch_feature_handle(self,ev):
        """
        datapath中有配置消息到达
        """
        #print("------XXXXXXXXXX------%d------XXXXXXXXXX----------switch_feature_handle"%self.net_arrived)
        #print("----%s----------",ev.msg)
        msg = ev.msg
        datapath = msg.datapath
        ofproto = datapath.ofproto
        ofp_parser = datapath.ofproto_parser

        match = ofp_parser.OFPMatch()

        actions = [ofp_parser.OFPActionOutput(ofproto.OFPP_CONTROLLER,ofproto.OFPCML_NO_BUFFER)]

        self.add_flow(datapath=datapath,priority=0,match=match,actions=actions,extra_info="config infomation arrived!!")


    def add_flow(self,datapath,priority,match,actions,idle_timeout=0,hard_timeout=0,extra_info=None):
        #print("-----------------add_flow:")
        if extra_info != None:
            print(extra_info)
        ofproto = datapath.ofproto
        ofp_parser = datapath.ofproto_parser


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

        mod = ofp_parser.OFPFlowMod(datapath=datapath,priority=priority,
                                    idle_timeout=idle_timeout,
                                    hard_timeout=hard_timeout,
                                    match=match,instructions=inst)
        datapath.send_msg(mod);

    @set_ev_cls(ofp_event.EventOFPPacketIn,MAIN_DISPATCHER)
    def packet_in_handler(self,ev):
```

```
        #print("------------------packet_in_handler")
        msg = ev.msg
        datapath = msg.datapath
        ofproto = datapath.ofproto
        ofp_parser = datapath.ofproto_parser

        dpid = datapath.id
        in_port = msg.match['in_port']

        pkt = packet.Packet(msg.data)
        eth_pkt = pkt.get_protocol(ethernet.ethernet)
        dst = eth_pkt.dst
        src = eth_pkt.src

        #self.logger.info("------------------Controller %s get packet, Mac address from: %s send to: %s , send from datapath: %s,in port is: %s"
        #                    ,dpid,src,dst,dpid,in_port)
        self.get_topology(None)


    @set_ev_cls([event.EventHostAdd])
    def _host_add_handler(self,ev):    #主机信息单独处理，不属于网络拓扑
        self.host_list = get_host(self.topology_api_app) #3.需要使用pingall,主机通过与边缘交换机连接，才能告诉控制器
        #获取主机信息字典ip2host{ipv4:host object}  ip2switch{ipv4:dpid}
        for i,host in enumerate(self.host_list):
            self.ip2switch["%s"%host.ipv4] = host.port.dpid
            self.ip2host["%s"%host.ipv4] = host


    events = [event.EventSwitchEnter, event.EventSwitchLeave,
              event.EventSwitchReconnected,
              event.EventPortAdd, event.EventPortDelete,
              event.EventPortModify,
              event.EventLinkAdd, event.EventLinkDelete]
    @set_ev_cls(events)
    def get_topology(self,ev):
        #print("------++++++++++++------%d------++++++++++------------get_topology"%self.net_arrived)

        self.net_flag = False
        self.net_topo = []

        #print("-----------------get_topology")
        #获取所有的交换机、链路
        self.switch_list = get_switch(self.topology_api_app) #1.只要交换机与控制器联通，就可以获取
        self.link_list = get_link(self.topology_api_app) #2.在ryu启动时，加
```

```
上--observe-links即可用于拓扑发现

        #获取交换机字典id2dpid{id:dpid} dpid2switch{dpid:switch object}
        for i,switch in enumerate(self.switch_list):
            self.id2dpid[i] = switch.dp.id
            self.dpid2id[switch.dp.id] = i
            self.dpid2switch[switch.dp.id] = switch


        #根据链路信息，开始获取拓扑信息
        self.net_size = len(self.id2dpid) #表示网络中交换机个数
        for i in range(self.net_size):
            self.net_topo.append([0]*self.net_size)

        for link in self.link_list:
            src_dpid = link.src.dpid
            src_port = link.src.port_no

            dst_dpid = link.dst.dpid
            dst_port = link.dst.port_no

            try:
                sid = self.dpid2id[src_dpid]
                did = self.dpid2id[dst_dpid]
            except KeyError as e:
                #print("--------------Error:get KeyError with link infomation(%s)"%e)
                return
            self.net_topo[sid][did] = [src_port,0] #注意：这里0表示存在链路，后面可以修改为时延
            self.net_topo[did][sid] = [dst_port,0] #注意：修改为列表，不要用元组，元组无法修改，我们后面要修改时延


        self.net_flag = True #表示网络拓扑创建成功

    def show_topology(self):
        print("------------------show_topology")
        print("----------switch network----------")
        line_info = "         "
        for i in range(self.net_size):
            line_info+="        s%-5d        "%self.id2dpid[i]
        print(line_info)
        for i in range(self.net_size):
            line_info = "s%d      "%self.id2dpid[i]
            for j in range(self.net_size):
                if self.net_topo[i][j] == 0:
                    line_info+="%-22d"%0
```

```
                else:
                    line_info+="(%d,%.12f)    "%tuple(self.net_topo[i][j]
)
            print(line_info)

        print("----------host 2 switch----------")
        for key,val in self.ip2switch.items():
```

三：实验验证

（一）启动Ryu控制器

Shell | 复制代码

```
ryu-manager TopoDetect.py --verbose --observe-links
```

其中--observe-links用于拓扑发现，添加之后用于链路的信息获取！！

```
ld@ld-Lenovo-Product:~/RyuSCP$ ryu-manager TopoDetect.py --verbose --observe-links
loading app TopoDetect.py
require_app: ryu.topology.switches is required by TopoDetect
loading app ryu.topology.switches
loading app ryu.controller.ofp_handler
instantiating app ryu.topology.switches of Switches
instantiating app TopoDetect.py of TopoDetect
instantiating app ryu.controller.ofp_handler of OFPHandler
BRICK TopoDetect
  CONSUMES EventPortModify
  CONSUMES EventSwitchReconnected
  CONSUMES EventLinkDelete
```

（二）启动mininet

Shell | 复制代码

```
sudo mn --topo=linear,4 --switch=ovsk --controller=remote --link=tc
```

```
ld@ld-Lenovo-Product:~/openvswitch/openvswitch-2.11.4$ sudo mn --topo=linear,4 -
-switch=ovsk --controller=remote --link=tc
[sudo] password for ld:
*** Creating network
*** Adding controller
Connecting to remote controller at 127.0.0.1:6653
```

注意：需要在mininet中使用pingall，才能使得交换机获得host存在，从而使得控制器获取host消息！！

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X
h2 -> X X X
h3 -> X X X
h4 -> X X X
*** Results: 100% dropped (0/12 received)
```

（三）结果显示

```
----------switch network----------
              s1                  s2                  s3                  s4
s1      0                   (2,0.000000000000)  0                   0
s2      (2,0.000000000000)  0                   (3,0.000000000000)  0
s3      0                   (2,0.000000000000)  0                   (3,0.000000000000)
s4      0                   0                   (2,0.000000000000)  0
```

```
----------host 2 switch----------
['10.0.0.1']---s1
['10.0.0.3']---s3
['10.0.0.2']---s2
['10.0.0.4']---s4
```