Nguyễn Thanh Quân - ntquan@fit.hcmus.edu.vn

ARP – OpenFlow

Mục tiêu:

1. Tạo gói tin ARP reply khi nhận được gói tin ARP request từ controller với điều kiện controller biết được thông tin của host.

Tạo Topology trong Mininet:

1. Tạo topology với mininet

```
$ ssh -l mininet 172.16.42.130
mininet@mininet-vm:~$ sudo mn --topo single,2 --mac --controller remote,ip=172.16.42.1 --switch ovsk
*** Creating network
*** Adding controller
Unable to contact the remote controller at 172.16.42.1:6633
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

2. Trên máy chạy RYU, vào thư mục /ryu/ryu/app/, copy file "simpe_switch.py" thanh "arp_simple_switch.py", sau đó edit file này

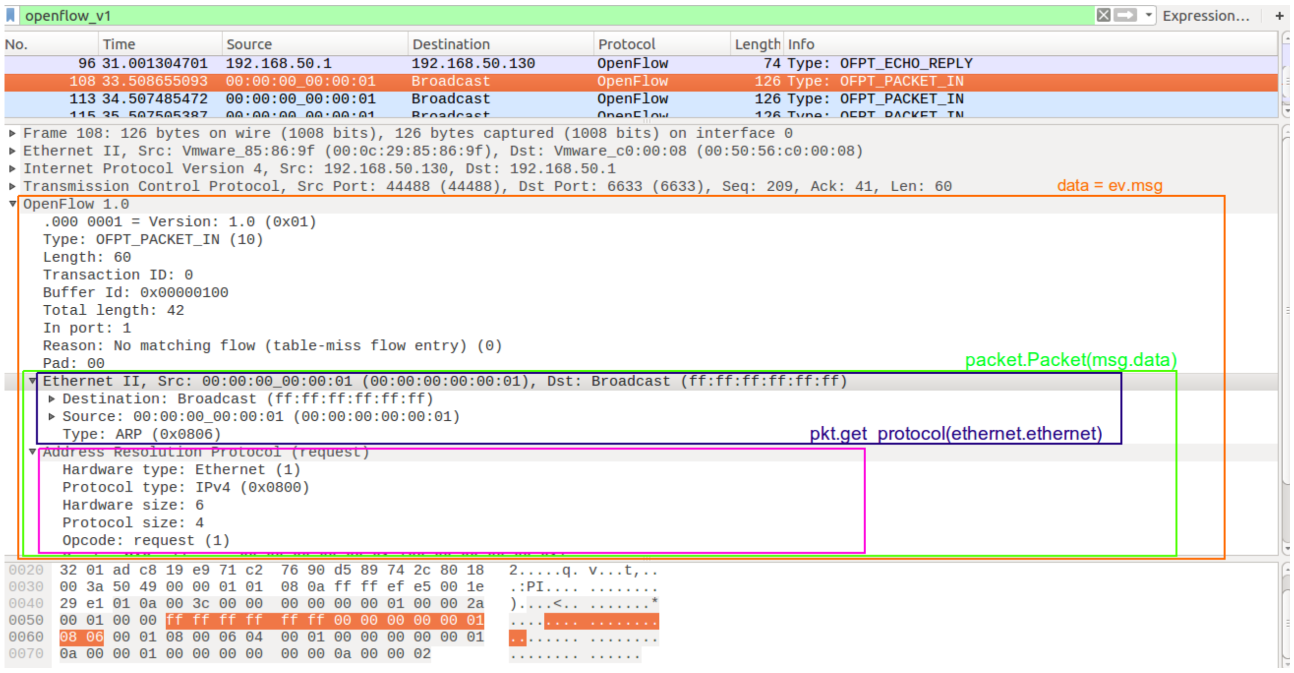
```
$ cd ryu/ryu/app/
$ cp simple_switch.py arp_simple_switch.py
```

3. Chỉnh sửa file này như sau:

```
# Copyright (C) 2011 Nippon Telegraph and Telephone Corporation.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
# implied.
# See the License for the specific language governing permissions and
# limitations under the License.

"""
```

```
An OpenFlow 1.0 L2 learning switch implementation.
"""


from ryu.base import app_manager
from ryu.controller import ofp_event
from ryu.controller.handler import MAIN_DISPATCHER
from ryu.controller.handler import set_ev_cls
from ryu.ofproto import ofproto_v1_0
from ryu.lib.mac import haddr_to_bin
from ryu.lib.packet import packet
from ryu.lib.packet import ethernet
from ryu.lib.packet import ether_types

from ryu.lib.packet import arp


class SimpleSwitch(app_manager.RyuApp):
    OFP_VERSIONS = [ofproto_v1_0.OFP_VERSION]

    def __init__(self, *args, **kwargs):
        super(SimpleSwitch, self).__init__(*args, **kwargs)
        self.mac_to_port = {}

    def add_flow(self, datapath, in_port, dst, actions):
        ofproto = datapath.ofproto

        match = datapath.ofproto_parser.OFPMatch(
            in_port=in_port, dl_dst=haddr_to_bin(dst))

        mod = datapath.ofproto_parser.OFPFlowMod(
            datapath=datapath, match=match, cookie=0,
            command=ofproto.OFPFC_ADD, idle_timeout=0, hard_timeout=0,
            priority=ofproto.OFP_DEFAULT_PRIORITY,
            flags=ofproto.OFPFF_SEND_FLOW_REM, actions=actions)
        datapath.send_msg(mod)

    def receive_arp (self, datapath, pkt, eth, inPort):
        arpPacket = pkt.get_protocol(arp.arp) # Lay ra thong tin ARP trong goi tin

        if arpPacket.opcode ==1:
            arp_srcIp = arpPacket.src_ip #Lay cac field trong goi ARP
            arp_dstIp = arpPacket.dst_ip #Lay cac filed trong goi ARP
            self.logger.info("receive ARP request %s => %s (port %d)" % (eth.src, eth.dst, inPort))
            self.logger.info("Info ARP with IP from %s ask %s" %(arp_srcIp, arp_dstIp))
            self.logger.info("Info ARP with MAC from %s ask %s" %(arpPacket.src_mac, arpPacket.dst_mac))

            self.reply_arp(datapath, eth, arpPacket, inPort) #Goi ham reply_arp
        elif arpPacket.opcode == 2:
            pass

    def reply_arp (self, datapath, eth, arpPacket, inPort):
        dstIP = arpPacket.src_ip
        srcIP = arpPacket.dst_ip
        if srcIP == "10.0.0.2":
            srcMac = "00:00:00:00:00:02"
        elif srcIP == "10.0.0.1":
            srcMac = "00:00:00:00:00:01"
        #create packet arp reply
        e = ethernet.ethernet(dst=arpPacket.src_mac,
                    src=srcMac,
                    ethertype=ether_types.ETH_TYPE_ARP) # Tao goi tin Layer 2

        """
        e = ethernet.ethernet(dst='ff:ff:ff:ff:ff:ff',
                    src='08:60:6e:7f:74:e7',
                    ethertype=ether.ETH_TYPE_ARP)
        """
        """
        e = ethernet.ethernet(dst=arpPacket.src_mac,
                    src="ff:ff:ff:ff:ff:ff",
                    ethertype=ether_types.ETH_TYPE_ARP)
        """
        a = arp.arp(hwtype=1, proto=0x0800, hlen=6, plen=4, opcode=2,
            src_mac=srcMac, src_ip=srcIP,
            dst_mac=arpPacket.src_mac, dst_ip=dstIP) # Tao goi tin ARP
        p = packet.Packet() #Tao ra mot Packet
        p.add_protocol(e) #Dua goi tin Layer 2 vao Packet
        p.add_protocol(a) #Dua goi tin ARP vao Packet
        p.serialize()

        actions = [datapath.ofproto_parser.OFPActionOutput(inPort)] #Dinh nghia action với gói tin PACKET_OUT

        out = datapath.ofproto_parser.OFPPacketOut(
            datapath=datapath, buffer_id=0xffffffff, in_port=datapath.ofproto.OFPP_CONTROLLER,
            actions=actions, data=p.data) #Dinh nghia goi tin PACKOUT voi action
        datapath.send_msg(out) #Gui thong tin den OF switch


    @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
    def _packet_in_handler(self, ev):
        msg = ev.msg # Lay ra msg ma OF switch chuyen cho controller thong qua packet_in
        datapath = msg.datapath #Lay ra thong tin OF switch
        ofproto = datapath.ofproto #Trich ra nhung thong tin giao thuc ma OF switch ho tro

        pkt = packet.Packet(msg.data) #Parse goi tin ma OF switch nhan dc tu controller. Trong vi du la goi tin ARP
        eth = pkt.get_protocol(ethernet.ethernet) #Lay ra thong tin layer 2 cua goi tin

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

        #Phan code them: begin
        if eth.ethertype == ether_types.ETH_TYPE_ARP: #Neu ra goi tin ARP thi xu ly
            self.logger.info("Receive ARP packet")
            inPort = msg.in_port
            self.receive_arp(datapath, pkt, eth, inPort) #Goi ham receive_arp
            return
        else:
            self.logger.info("Drop packet")
            return
        #phan code them: end

        dst = eth.dst
        src = eth.src

        dpid = datapath.id
        self.mac_to_port.setdefault(dpid, {})

        self.logger.info("packet in %s %s %s %s", dpid, src, dst, msg.in_port)
        # learn a mac address to avoid FLOOD next time.
        self.mac_to_port[dpid][src] = msg.in_port

        if dst in self.mac_to_port[dpid]:
            out_port = self.mac_to_port[dpid][dst]
        else:
            out_port = ofproto.OFPP_FLOOD

        actions = [datapath.ofproto_parser.OFPActionOutput(out_port)]

        # install a flow to avoid packet_in next time
        if out_port != ofproto.OFPP_FLOOD:
            self.add_flow(datapath, msg.in_port, dst, actions)

        data = None
        if msg.buffer_id == ofproto.OFP_NO_BUFFER:
            data = msg.data

        out = datapath.ofproto_parser.OFPPacketOut(
            datapath=datapath, buffer_id=msg.buffer_id, in_port=msg.in_port,
            actions=actions, data=data)
        datapath.send_msg(out)

    @set_ev_cls(ofp_event.EventOFPPortStatus, MAIN_DISPATCHER)
    def _port_status_handler(self, ev):
        msg = ev.msg
        reason = msg.reason
        port_no = msg.desc.port_no

        ofproto = msg.datapath.ofproto
        if reason == ofproto.OFPPR_ADD:
            self.logger.info("port added %s", port_no)
        elif reason == ofproto.OFPPR_DELETE:
            self.logger.info("port deleted %s", port_no)
        elif reason == ofproto.OFPPR_MODIFY:
            self.logger.info("port modified %s", port_no)
        else:
            self.logger.info("Illeagal port state %s %s", port_no, reason)
```

3. Gói tin ARP request gửi lên controller



```
No.  Time           Source              Destination      Protocol   Length Info
  96 31.001304701  192.168.50.1        192.168.50.130   OpenFlow      74 Type: OFPT_ECHO_REPLY
 108 33.508655093  00:00:00_00:00:01   Broadcast        OpenFlow     126 Type: OFPT_PACKET_IN
 113 34.507485472  00:00:00_00:00:01   Broadcast        OpenFlow     126 Type: OFPT_PACKET_IN
 115 35.507505387  00:00:00_00:00:01   Broadcast        OpenFlow     126 Type: OFPT_PACKET_IN

▸ Frame 108: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits) on interface 0
▸ Ethernet II, Src: Vmware_85:86:9f (00:0c:29:85:86:9f), Dst: Vmware_c0:00:08 (00:50:56:c0:00:08)
▸ Internet Protocol Version 4, Src: 192.168.50.130, Dst: 192.168.50.1
▸ Transmission Control Protocol, Src Port: 44488 (44488), Dst Port: 6633 (6633), Seq: 209, Ack: 41, Len: 60
▾ OpenFlow 1.0
    .000 0001 = Version: 1.0 (0x01)
    Type: OFPT_PACKET_IN (10)
    Length: 60
    Transaction ID: 0
    Buffer Id: 0x00000100
    Total length: 42
    In port: 1
    Reason: No matching flow (table-miss flow entry) (0)
    Pad: 00
  ▾ Ethernet II, Src: 00:00:00_00:00:01 (00:00:00:00:00:01), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
    ▸ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
    ▸ Source: 00:00:00_00:00:01 (00:00:00:00:00:01)
      Type: ARP (0x0806)
  ▾ Address Resolution Protocol (request)
      Hardware type: Ethernet (1)
      Protocol type: IPv4 (0x0800)
      Hardware size: 6
      Protocol size: 4
      Opcode: request (1)

0020  32 01 ad c8 19 e9 71 c2  76 90 d5 89 74 2c 80 18   2.....q. v...t,..
0030  00 3a 50 49 00 00 01 01  08 0a ff ff ef e5 00 1e   .:PI.... ........
0040  29 e1 01 0a 00 3c 00 00  00 00 00 00 01 00 00 2a   )....<.. .......*
0050  00 01 00 00 ff ff ff ff  ff ff 00 00 00 00 00 01   ........ ........
0060  08 06 00 01 08 00 06 04  00 01 00 00 00 00 00 01   ........ ........
0070  0a 00 00 01 00 00 00 00  00 00 0a 00 00 02         ........ ......
```

4. Trên máy RYU, chạy file arp_simple_switch.py, trên máy Mininet dùng h1 ping h2

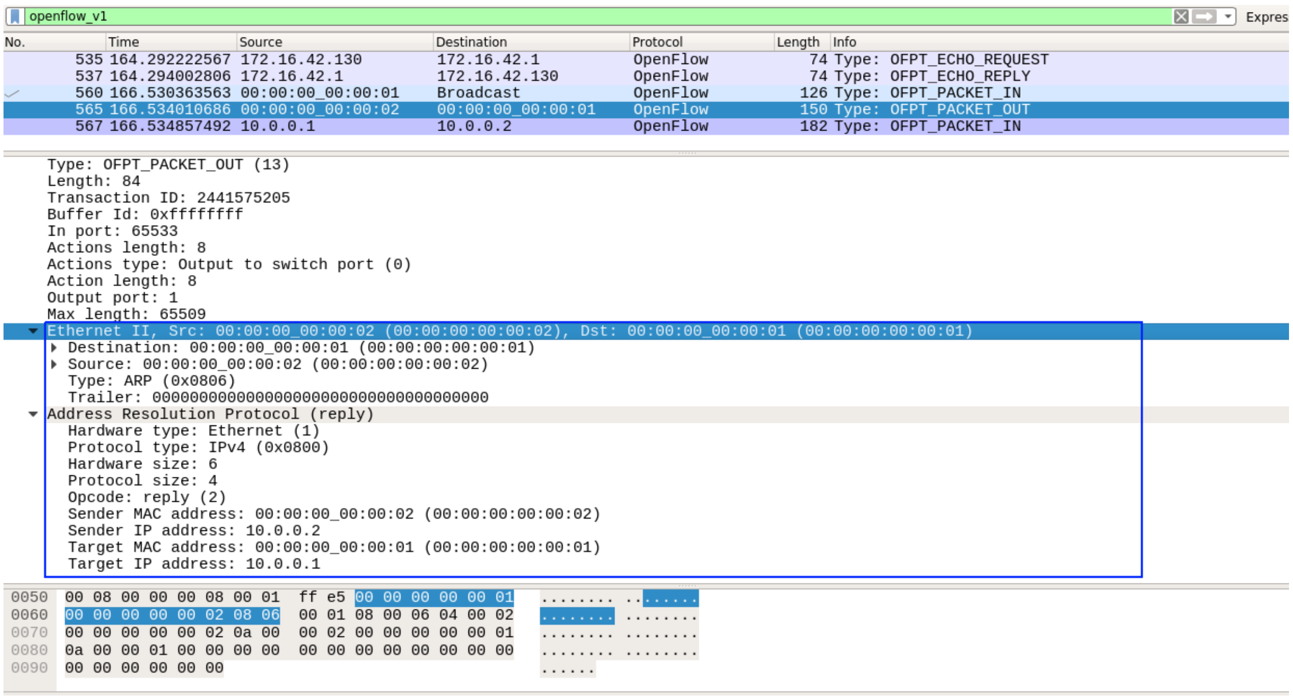
RYU

```
$ ryu-manager ~/ryu/ryu/app/arp_simple_switch.py
loading app /home/bapbap/ryu/ryu/app/arp_simple_switch.py
loading app ryu.controller.ofp_handler
instantiating app /home/bapbap/ryu/ryu/app/arp_simple_switch.py of SimpleSwitch
instantiating app ryu.controller.ofp_handler of OFPHandler
Receive ARP packet
receive ARP request 00:00:00:00:00:01 => ff:ff:ff:ff:ff:ff (port 1)
Info ARP with IP from 10.0.0.1 ask 10.0.0.2
Info ARP with MAC from 00:00:00:00:00:01 ask 00:00:00:00:00:00
Drop packet
```

Mininet

```
mininet> h1 ping h2 -c1
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
^C
--- 10.0.0.2 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

mininet> h1 arp -a
? (10.0.0.2) at 00:00:00:00:00:02 [ether] on h1-eth0
mininet>
```

5. Gói tin ARP reply



```
openflow_v1

No.    Time            Source                 Destination            Protocol   Length  Info
   535 164.292222567 172.16.42.130          172.16.42.1            OpenFlow       74 Type: OFPT_ECHO_REQUEST
   537 164.294002806 172.16.42.1            172.16.42.130          OpenFlow       74 Type: OFPT_ECHO_REPLY
   560 166.530363563 00:00:00_00:00:01      Broadcast              OpenFlow      126 Type: OFPT_PACKET_IN
   565 166.534010686 00:00:00_00:00:02      00:00:00_00:00:01      OpenFlow      150 Type: OFPT_PACKET_OUT
   567 166.534857492 10.0.0.1               10.0.0.2               OpenFlow      182 Type: OFPT_PACKET_IN

    Type: OFPT_PACKET_OUT (13)
    Length: 84
    Transaction ID: 2441575205
    Buffer Id: 0xffffffff
    In port: 65533
    Actions length: 8
    Actions type: Output to switch port (0)
    Action length: 8
    Output port: 1
    Max length: 65509
  Ethernet II, Src: 00:00:00_00:00:02 (00:00:00:00:00:02), Dst: 00:00:00_00:00:01 (00:00:00:00:00:01)
    Destination: 00:00:00_00:00:01 (00:00:00:00:00:01)
    Source: 00:00:00_00:00:02 (00:00:00:00:00:02)
    Type: ARP (0x0806)
    Trailer: 000000000000000000000000000000000000
  Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: 00:00:00_00:00:02 (00:00:00:00:00:02)
    Sender IP address: 10.0.0.2
    Target MAC address: 00:00:00_00:00:01 (00:00:00:00:00:01)
    Target IP address: 10.0.0.1

0050  00 08 00 00 00 08 00 01  ff e5 00 00 00 00 00 01   ........ ........
0060  00 00 00 00 00 02 08 06  00 01 08 00 06 04 00 02   ........ ........
0070  00 00 00 00 00 02 0a 00  00 02 00 00 00 00 00 01   ........ ........
0080  0a 00 00 01 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0090  00 00 00 00 00 00                                  ......
```

Tham khảo:

1) https://github.com/ttsubo/simpleRouter/blob/master/ryu-app/blog/article_01/simpleArp.py
2) https://github.com/ttsubo/simpleRouter/blob/master/ryu-app/blog/article_01/test_SimpleArp.py
3) http://ryu.readthedocs.io/en/latest/library_packet.html