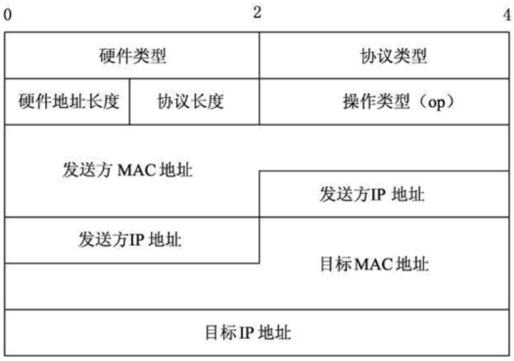
Computer Network* Lab 3

- **徐思源 191220133**
- Department of Computer Science and Technology
- Nanjing University
- 1357307497@qq.com

Task2

- **判断收到的数据包是否是ARP包，并且对属于自己的ARP包进行回复**
  - **ARP结构如下:**
    - 

ARP 报文总长度为 28 字节，MAC 地址长度为 6 字节，IP 地址长度为 4 字节。

其中，每个字段的含义如下。
- 硬件类型：指明了发送方想知道的硬件接口类型，以太网的值为 1。
- 协议类型：表示要映射的协议地址类型。它的值为 0x0800，表示 IP 地址。
- 硬件地址长度和协议长度：分别指出硬件地址和协议的长度，以字节为单位。对于以太网上 IP 地址的ARP请求或应答来说，它们的值分别为 6 和 4。
- 操作类型：用来表示这个报文的类型，ARP 请求为 1，ARP 响应为 2，RARP 请求为 3，RARP 响应为 4。
- 发送方 MAC 地址：发送方设备的硬件地址。
- 发送方 IP 地址：发送方设备的 IP 地址。
- 目标 MAC 地址：接收方设备的硬件地址。
- 目标 IP 地址：接收方设备的IP地址。

  - **判断是否是ARP包在指导文件中已经给出了，只需要检查ARP头是否存在**
    - ```
      arp = packet.get_header(Arp)#get the ARP header
              if arp is not None:
                  #handle
      ```
  - **对ARP包的处理只需要判断是否属于自己并且针对属于自己的包进行发送就可以,利用函数creat_ip_arp_reply构造回复的包**
      ```

- **根据ARP结构**

  ARP (address resolution protocol) header

  class switchyard.lib.packet.**Arp**(***kwargs*)

  hardwaretype

  operation ¶

  protocoltype

  senderhwaddr

  senderprotoaddr

  targethwaddr

  targetprotoaddr

- **我们提取其中的senderhwaddr,targetprotoaddr,senderprotoaddr作为creat的后三个参数，而第一个参数是我们当前的接口地址**
- ```
  reply = create_ip_arp_reply(interface.ethaddr,arp.senderhwaddr,arp.targetprotoaddr,arp.senderprotoaddr)
  ```
- ```
  try:
      interface=self.net.interface_by_ipaddr(arp.targetprotoaddr)
  except KeyError:
      interface=None

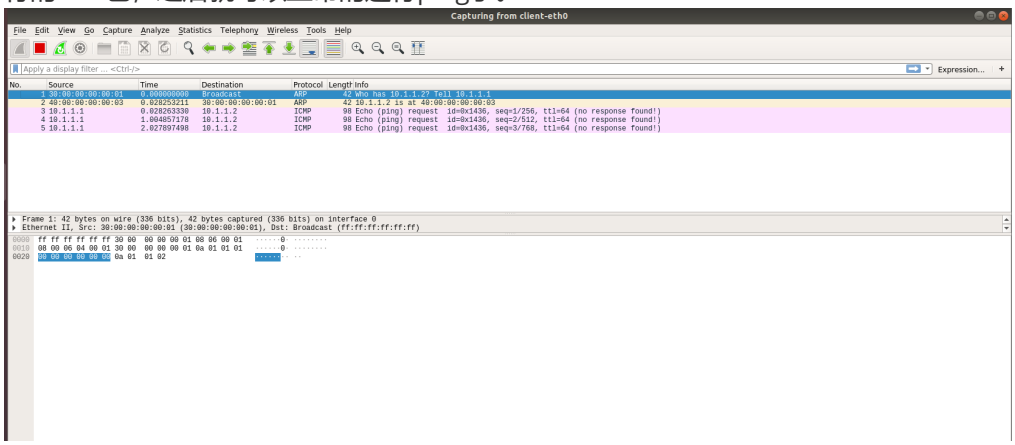
  if interface is not None:
      reply=create_ip_arp_reply(...)
      self.net.send_packet(ifaceName, reply)
  ```

- 实验结果及说明如下：
  - test：
    - ```
      uter1_testscenario.srpy myrouter.py
      10:55:28 2021/04/14     INFO Starting test scenario testcases/myrouter1_testscen
      ario.srpy

      Results for test scenario ARP request: 6 passed, 0 failed, 0 pending


      Passed:
      1    ARP request for 192.168.1.1 should arrive on router-eth0
      2    Router should send ARP response for 192.168.1.1 on router-
           eth0
      3    An ICMP echo request for 10.10.12.34 should arrive on
           router-eth0, but it should be dropped (router should only
           handle ARP requests at this point)
      4    ARP request for 10.10.1.2 should arrive on router-eth1, but
           the router should not respond.
      5    ARP request for 10.10.0.1 should arrive on on router-eth1
      6    Router should send ARP response for 10.10.0.1 on router-eth1


      All tests passed!
      ```
      ```

- mininet运行：
  - 按照指导文件中的步骤进行运行，抓包结果如下，抓包结果显示的是，当client需要对路由器router发送ICMP包的时候，首先需要获得router的mac地址，所以先广播一个ARP包询问router的地址，当router接收到这个ARP包的时候，遍历自己的interface发现这个ARP包是发给自己的，所以做出回复，而这个回复的包被wireshark拦截到，就是第二行的ARP包，之后就可以正常的进行ping了。
  - 

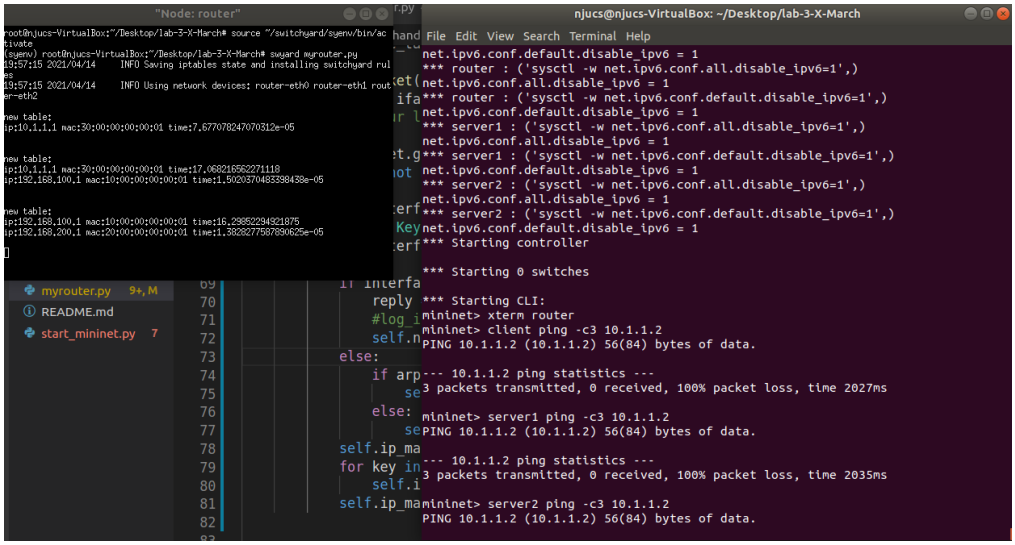
Task3

- **这一步中我们需要实现ip地址和mac地址对组成的储存表，并且需要首先timeout机制，因为**

  You can also see here that the IP address is **unique** in the table.

  所以这里我们将ip地址作为key值来索引，timestamp和mac地址作为value值，table结构用dict来实现，timestamp和mac地址组成一个类结构

  - **代码实现和说明如下：**
    - ```
      if interface is not None:
          ...
      else:
          #reply packet
          if arp.targethwaddr != "00:00:00:00:00:00": #target`s mac has
      been assigned
              self.ip_mac_table.set(arp.targetprotoaddr,arp.targethwaddr)
      self.ip_mac_table.set(arp.senderprotoaddr,arp.senderhwaddr) #update
      timestamp
          for key in list(self.ip_mac_table.table.keys()): #check timeout
              self.ip_mac_table.get(key)
          self.ip_mac_table.print()#check the table
      ```
      ```

- **mininet运行结果如下：**
  - 打开router路由器，并且client，server1，server2分别对router进行ping，然后我们可以得到以下的table结果，因为ping的时间比较长，所以为了结果更加清晰，将timeout标准定为30秒，提交的版本是10秒的，输出中的时间是entry已经存在的时间：
    - 

    - **可以看到第一次client对router ping的时候记录了client的信息，之后server1对router ping的时候记录了server1的信息，此时client的entry已经存在了17秒多，所以接下来server2 ping的时候client的entry已经被删除了，server1的还在，符合timeout的规则，并且可以看到我们会记录发送方的ip和mac数据到table当中**