南京大学本科生实验报告

课程名称：**计算机网络**　　任课教师：黄程远　　助教：刘松岳

学院	**人工智能学院**	专业（方向）	**人工智能**
学号	**211300022**	姓名	**刘梦杰**
Email	**2757400745@qq.com**	开始/完成日期	**2024.10.24/2024.11.18**

1. **实验名称　计网 lab3**
2. **实验目的　实现路由器的 ip 转发表的构造、arp 请求发送、转发数据包**
3. **实验内容与核心代码**
   a) **IP 转发表的构造：构造列表变量 self.forwarding_table，通过读取文件和调用 net.interfaces()来向其中加入[子网，下一跳 ip，接口]，并在加入变量时通过函数 insert_by_max 依据最大前缀进行插入排序，使得匹配时匹配到第一个合适的停下即可；**

```
def insert_by_max(l: list, a):
    if len(l) == 0:
        l.append(a)
    else:
        c = 0
        for i in l:
            p1 = i[0].prefixlen
            p2 = a[0].prefixlen
            if p1 >= p2:
                c += 1
            else:
                break
        l.insert(c, a)
```

```
for intf in self.net.interfaces():
    ipad = IPv4Address(int(intf.ipaddr) & int(intf.netmask))
    x = IPv4Network(str(ipad)+'/'+str(intf.netmask))
    x1 = [x, '', intf.name]
    insert_by_max(self.forwarding_table, x1)
with open("forwarding_table.txt", "r") as flies:
    a = flies.readlines()
for i in range(len(a)):
    a[i] = a[i].split(" ")
    if i != len(a)-1:
        a[i][3] = a[i][3].strip("\n")
for i in a:
    x2 = [IPv4Network(i[0]+'/'+i[1]), i[2], i[3]]
    insert_by_max(self.forwarding_table, x2)
```

b) 转发表的匹配：排除特殊情况以后遍历转发表进行匹配，再将匹配到的下一跳 ip 和接口放在类 Waiting_packet 中：

```
elif ipv4:
    ipv4.ttl -= 1
    judging = True
    interface_macs = [intf.ethaddr for intf in self.net.interfaces()]
    ether = packet.get_header(Ethernet)
    if ether.dst != 'ff:ff:ff:ff:ff:ff' and ether.dst not in interface_macs:
        judging = False
    for intf in self.net.interfaces():
        if ipv4.dst == intf.ipaddr:
            judging = False
            break
    if judging:
        fw_index = -1
        for i in range(len(self.forwarding_table)):
            if ipv4.dst in self.forwarding_table[i][0]:
                fw_index = i
                break
        if fw_index != -1:
            if self.forwarding_table[fw_index][1]:
                next_hop_ip = IPv4Address(self.forwarding_table[fw_index][1])
            else:
                next_hop_ip = ipv4.dst
            for intf in self.net.interfaces():
                if intf.name == self.forwarding_table[fw_index][2]:
                    router_intf = intf
                    break
            packet[0].src = router_intf.ethaddr
            self.waiting_queue.append(Waiting_packet(packet, router_intf, next_hop_ip))
```

c) 转发包的等待队列构建与处理：通过类 Waiting_packet 存储

包的下一跳 ip、接口、上次转发时间、转发次数、待转发的包五个变量，接着在 start 函数中处理，列表 without_query 用于存储因为查到了或者次数超过限制而不用再查询而被删除的类，re 列表用于排除重复的 ip（需要注意的是此处理不能放在函数 handle_packet 中，因为在没有收到包的时候也要处理队列，否则会产生 bug）；

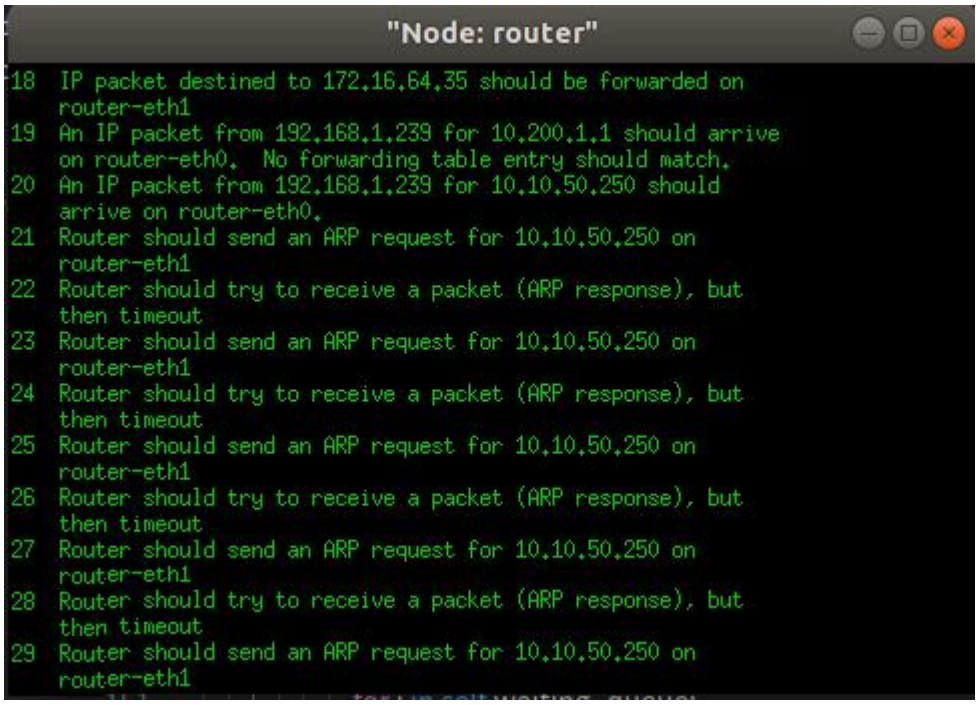
```
class Waiting_packet:
    def __init__(self, pkt, intf, dstip):
        self.packet = pkt
        self.last_send_time = 0
        self.count = 0
        self.router_intf = intf
        self.next_hop_ip = dstip
```

```
while True:
    bb = True
    try:
        recv = self.net.recv_packet(timeout=1.0)
    except NoPackets:
        bb = False
    except Shutdown:
        break
    if bb:
        self.handle_packet(recv)
    re = []
    without_query = []
    for i in self.waiting_queue:
        if i.next_hop_ip in self.my_arptable.keys():
            mac = self.my_arptable[i.next_hop_ip][0]
            i.packet[0].dst = str(mac)
            self.net.send_packet(i.router_intf.name, i.packet)
            without_query.append(i)
        elif time.time()-i.last_send_time>=1 and i.next_hop_ip not in re:
            if i.count<5:
                re.append(i.next_hop_ip)
                self.send_arp_request(i.router_intf, i.next_hop_ip)
                i.count += 1
                i.last_send_time = time.time()
            else:
                nip = i.next_hop_ip
                for j in self.waiting_queue:
                    if j.next_hop_ip == nip:
                        without_query.append(j)
    for i in without_query:
        self.waiting_queue.remove(i)
```

4. **实验测试方式与结果：**

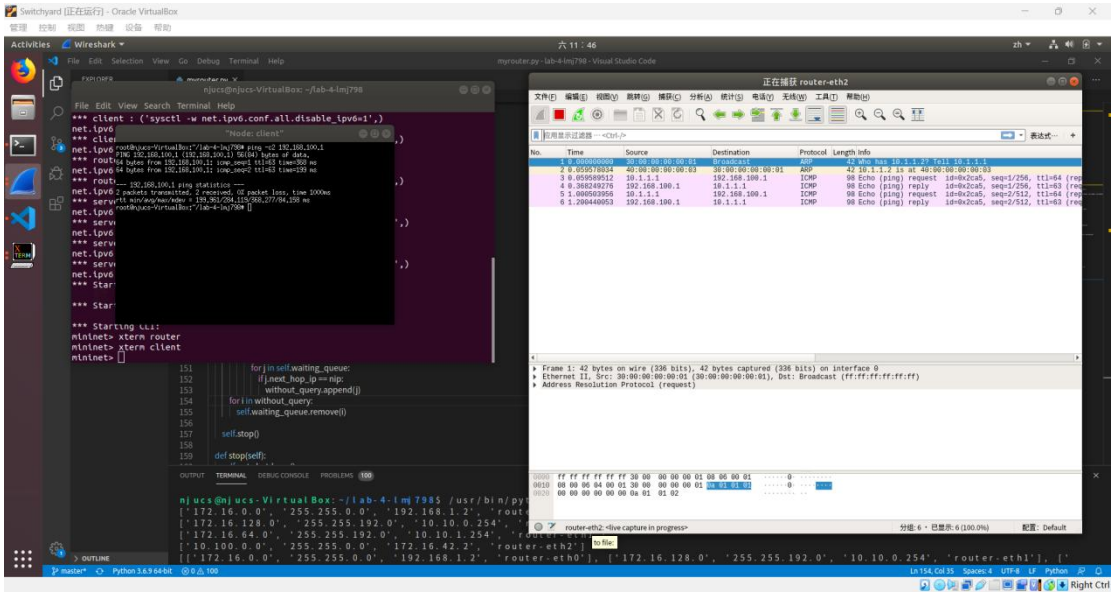
   a) **Test 结果如下：**

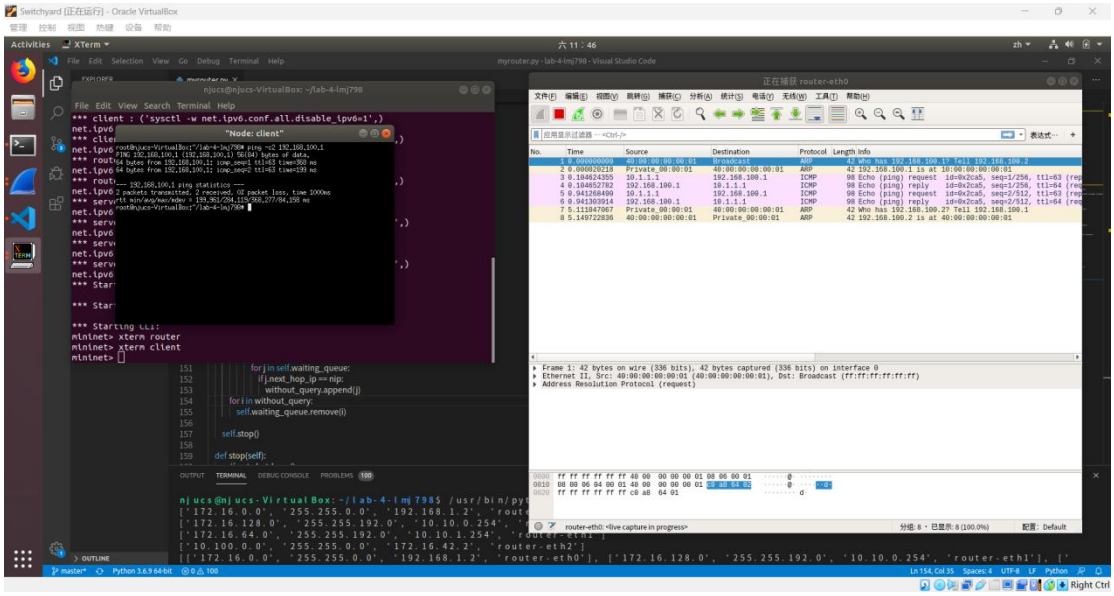
   均通过



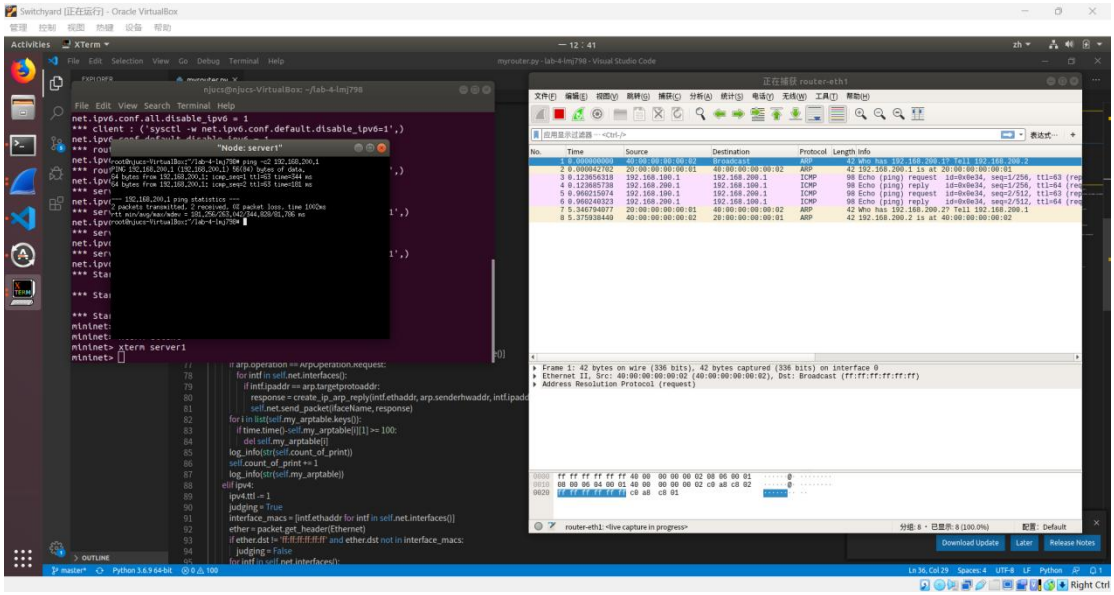
b) Wireshark 与 xterm 输出：

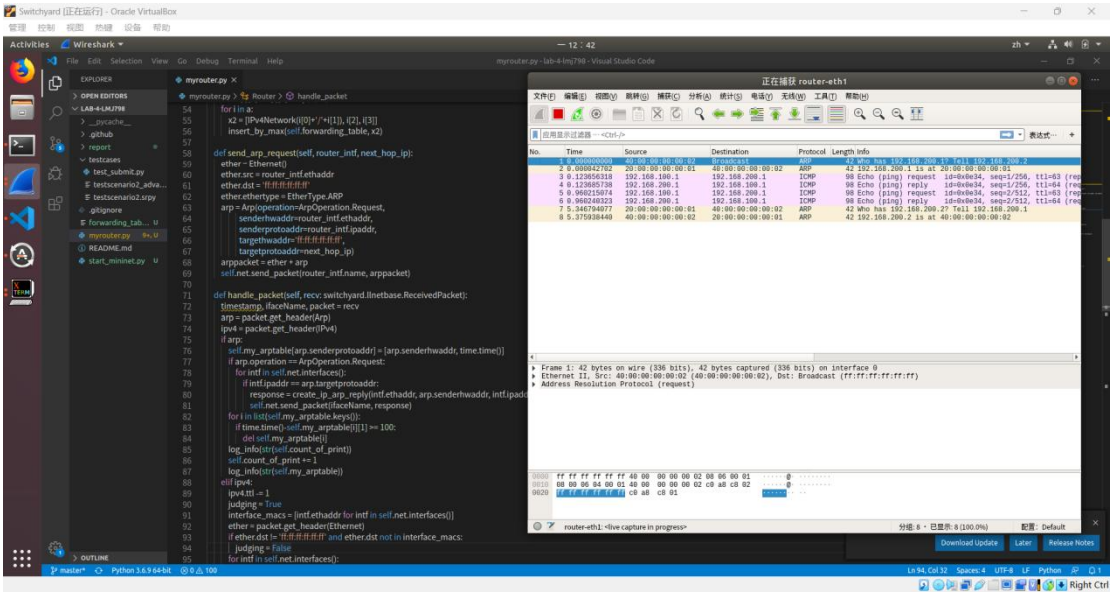
Wireshark 输出如下：





可以看到 arp 包的发送与回复，同样在 sever1 上进行测试，ping -c2 192.168.200.1，结果相同：





5. 总结与感想

a) **Lab3 中没有判断是否是 ArpOperation.Request，虽然 lab3 的 test 通过，但是 lab4 因此出错**