

南京大学本科生实验报告

课程名称：计算机网络

任课教师：田臣/李文中

助教：

学院	匡亚明学院	专业（方向）	计算机科学与技术
学号	221240009	姓名	冯雨桐
Email	221240009@smail.nju.edu.cn	开始/完成日期	10.27-10.28

Task 2

```
def lookup(self, ip):
    intf = None
    dst_ip = None
    max_prefix = 0
    for entry in self.table:
        prefix = entry["ip"]
        mask = entry["mask"]
        prefix_len = IPv4Network(f"192.0.0.0/{str(mask)}").prefixlen
        if (int(mask) & int(ip)) == (int(mask) & int(prefix)):
            if prefix_len > max_prefix:
                max_prefix = prefix_len
                intf = entry["intfname"]
                dst_ip = entry["nxtHop"]
                print(entry)
    if intf == None:
        print(f"Not found: {ip}")
    print(f"[lookup]: {dst_ip} matches {max_prefix}")
    return intf, dst_ip
```

建立一个 ForwardingTable 类，创建一个 lookup 方法

初始化时，从 forwarding_table.txt 和 net.ports()中读取所有表项并保存

Lookup 方法接受 ip 作为参数，遍历所有 forwardingTable 表项，若 $\text{prefixIP} \& \text{mask} == \text{ip} \& \text{mask}$ ，则记录下该表项，最终选择 mask 最长的表项最为最终的匹配项

遍历完成后，如果没有任何表项匹配，则返回 None

如果有匹配，则返回该表项的出端口和下一跳 ip

Router 会检查返回值，如果返回 None，直接丢弃包。否则，如果 nxtHop 是 0.0.0.0，说明直接可达，下一跳 mac 直接等于目的地 ip 的 mac，如果不是 0.0.0.0，说明下一跳需要先经过另一个路由器，因此下一跳的 mac 为该路由器的 mac

无论哪种情况，都需要寻找某一个 ip 对应的 mac。这需要查询 arp 缓存/发送 arp-request。在 Task3 中实现。

Task 3

代码框架如下：

Router 主循环中, 先处理 arp-request 队列. 处理时需要注意删除尝试满五次的表项, 另外, 同一个 ip, 只能每隔一秒请求一次, 即使有多个包对应该 ip. 删除一个表项时, 需要一同删除所有对应该 ip 的其他表项

然后进入 handle_packet()

丢弃 IPv6 头的包, 判断是否有 arp 头, 分别进入 arp/none_arp 处理函数

Arp 处理函数中, 和 Lab3 基本相同, 更新 arp 缓存, 丢弃非法包. 现在需要单独处理 arp-reply. 从 arp-reply 解析得到 src_ip 和 src_mac, 然后去 arp-request 队列中寻找等待 src_ip 的表项. 将这些表项按顺序删除并调用 UnfinishedArp.resolve(后面解释)发送.

None-arp 处理函数中, 也有一些 corner-case 的判断(省略). 主要代码如下:
先通过 lookup 方法, 查询 dst_ip 对应的下一跳 ip 和出端口. 如果返回 None, 丢弃该包. 否则, 判断下一跳 ip 是否为 0.0.0.0, 如果不是, query_ip=下一跳 ip, 否则 query_ip=dst_ip

接下来需要得到 query_ip 的 mac

先查询 arp 缓存, 如果命中, 那么直接构造一个新的 packet 发出

如果不在 arp 缓存中, 用下面的代码构造一个 arp-request 询问 query_ip 的 mac

```
print(f"[Arp miss]: {dst_ip} not in arp")
self.arp_table.show()
ether = Ethernet()
ether.src = intf_mac
ether.dst = "ff:ff:ff:ff:ff:ff"
ether.ethertype = EtherType.ARP
arp = Arp(
    operation=ArpOperation.Request,
    senderhwaddr=intf_mac,
    senderprotoaddr=self.mac2ip[intf_mac],
    targethwaddr="ff:ff:ff:ff:ff:ff",
    targetprotoaddr=dst_ip
)
arp_packet = ether + arp
```

接着, 将当前的包封装为一个 UnfinishedArp 类, 添加到处理队列尾部

这里 UnfinishedArp 包含了原本要发送的包, 需要查询的 ip, 查询次数, 上一次查询的时间戳等信息. 并且定义了 resolve 方法, 可以在得到查询结果后, 将 UnfinishedArp 对象原本要发送的包发送出去:

```
def resolve(self, arp_reply):
    print(self.packet.headers())
    eth = self.packet.get_header(Ethernet)
    eth.dst = EthAddr(arp_reply.senderhwaddr)
    eth.src = EthAddr(self.outIntf_mac)

    ipv4hdr = self.packet.get_header(IPv4)
    ipv4hdr.ttl -= 1

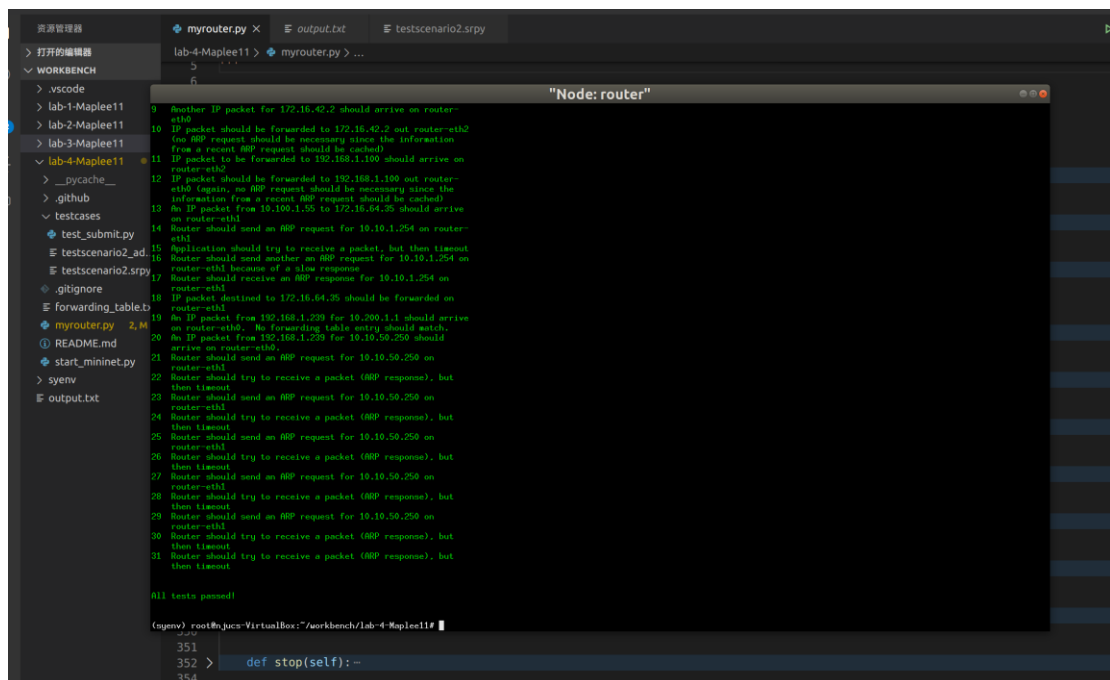
    hdrs = []
    for hdr in self.packet.headers():
        if hdr in ["Ethernet", "IPv4"]:
            continue
        hdrs.append(self.packet.get_header(hdr))

    pkt = eth + ipv4hdr
    for hdr in hdrs:
        pkt += hdr
    return self.outIntf, pkt
```

返回的二元组就是 `net.send` 需要的两个参数

上面构造 `arp_request` 和 `resolve` 中构造包的方法都类似，只要将 `packet.headers()` 中所有协议的头分别取出，分别进行处理，最后加起来即可。注意修改 `ttl` 等信息

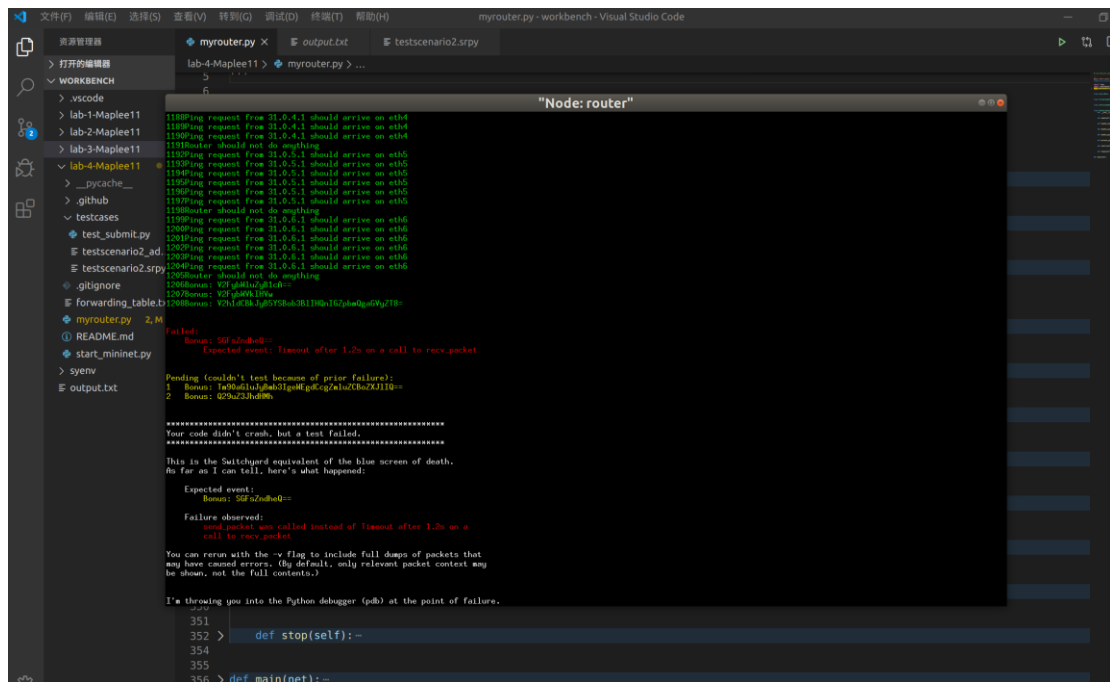
两个 `testcase` 的测试结果：



```
9 Another IP packet for 172.16.42.2 should arrive on router-eth0
10 IP packet should be forwarded to 172.16.42.2 out router-eth0
   (no ARP request should be necessary since the information
   from a recent ARP request should be cached)
11 IP packet to be forwarded to 192.168.1.100 should arrive on
   router-eth0
12 IP packet should be forwarded to 192.168.1.100 out router-
   eth0 (again, no ARP request should be necessary since the
   information from a recent ARP request should be cached)
13 An IP packet from 19.199.1.10 to 172.16.64.25 should arrive
   on router-eth1
14 Router should send an ARP request for 10.10.1.254 on router-
   eth1
15 Application should try to receive a packet, but then timeout
16 Router should send another ARP request for 10.10.1.254 on
   router-eth1 because of a slow response
17 Router should receive an ARP response for 10.10.1.254 on
   router-eth1
18 IP packet destined to 172.16.64.35 should be forwarded on
   router-eth1
19 An IP packet from 192.168.1.229 for 10.200.1.1 should arrive
   on router-eth0. No forwarding table entry should match.
20 An IP packet from 192.168.1.229 for 10.10.50.250 should
   arrive on router-eth0.
21 Router should send an ARP request for 10.10.50.250 on
   router-eth1
22 Router should try to receive a packet (ARP response), but
   then timeout
23 Router should send an ARP request for 10.10.50.250 on
   router-eth1
24 Router should try to receive a packet (ARP response), but
   then timeout
25 Router should send an ARP request for 10.10.50.250 on
   router-eth1
26 Router should try to receive a packet (ARP response), but
   then timeout
27 Router should send an ARP request for 10.10.50.250 on
   router-eth1
28 Router should try to receive a packet (ARP response), but
   then timeout
29 Router should send an ARP request for 10.10.50.250 on
   router-eth1
30 Router should try to receive a packet (ARP response), but
   then timeout
31 Router should try to receive a packet (ARP response), but
   then timeout

All tests passed!

(gnvm) root@hpc:~$ VirtualBox:~/workbench/lab-4-Maple11#
351
352 > def stop(self): ~
354
```



最后三个 Bonus 没有通过

Mininet 测试

