

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：

学院	计算机学院	专业（方向）	计算机科学与技术
学号	211840016	姓名	姚宇斌
Email	2427482058@qq.com	开始/完成日期	

1. 实验名称 Forwarding Packets

2. 实验目的 实现路由器的两个功能，接收并转发目的地是其他主机的数据包和为没有已知 mac 地址的 ip 地址发送 arp 请求。

3. 实验内容

Task2: IP Forwarding Table Lookup, 模拟路由器转发表, 实现复合最长前缀匹配的查询。

Task3: Forwarding the Packet and ARP, 根据目的 mac 地址发送 Arp Request, 转发部分数据包

4. 实验结果

建立 forwarding_table, 表项为 dest, mask, gateway, interface
对 table 按前缀长度从大到小排序, 在匹配时从前向后遍历第一个匹配的就是长度最大的

建 表 代 码 :

```
class ForwardingTableEntry:
    def __init__(self, dest, mask, gateway, interface):
        self.dest = dest
        self.mask = mask
        self.gateway = gateway
        self.interface = interface
        self.prefixnet = IPv4Network(f'{dest}/{mask}')
```

重定义表的比较函数，用于按照前缀长度排序

```
def __lt__(self, other):  
    return self.prefixnet.prefixlen > other.prefixnet.prefixlen
```

最长前缀匹配：

```
def get_forwarding_entry(self, ipaddr):  
    for entry in self.forwarding_table:  
        if ipaddr in entry.prefixnet:  
            return entry  
    return None
```

IPv4 包处理：

如果包长错误或者发给自己则不处理，在 `forwarding_table` 中查找对应表项，没有则不处理，确定下一跳地址，为给定的下一跳或者就是目的地 `ip` 地址

```
def handle_ipv4_packet(self, recv):  
    timestamp, ifaceName, packet = recv  
    ipv4 = packet.get_header(IPv4)  
    eth = packet.get_header(Ethernet)  
    dst_ip = ipv4.dst  
    if len(eth) + ipv4.total_length != packet.size():  
        return  
    if dst_ip in self.ip_list:  
        return  
    entry = self.get_forwarding_entry(dst_ip)  
    if entry is None:  
        return  
  
    if entry.gateway is None:  
        next_hop_ip = dst_ip  
    else:  
        next_hop_ip = ip_address(entry.gateway)  
    next_hop_mac = self.arp_table.get(next_hop_ip)
```

若表中已有 mac 地址，则直接发包，否则发送 arp 请求进行询问
(若已经发过了则不用再发)

```
if next_hop_mac is not None:
    packet[Ethernet].src = self.net.interface_by_name(entry)
    packet[Ethernet].dst = next_hop_mac
    packet[IPv4].ttl -= 1
    self.net.send_packet(entry.interface, packet)
else:
    if next_hop_ip not in self.waiting_ip.keys():
        self.waiting_ip[next_hop_ip] = (time.time(), 1)
        arp_request = create_ip_arp_request(self.net.interface)
        self.net.send_packet(entry.interface, arp_request)
    if next_hop_ip not in self.waiting_packet.keys():
        self.waiting_packet[next_hop_ip] = []
    self.waiting_packet[next_hop_ip].append(packet)
```

Arp packet 处理:

若不是发送给自己的包或者源地址是广播地址，不处理

若是 arp 请求，创立 arp 回复并发回，

若是 arp 回复，将等待队列中对应的等待包发送出去后删除。

```

dst_ip = arp.targetprotoaddr
if dst_ip not in self.ip_list:
    return
if arp.operation == ArpOperation.Request:
    self.arp_table[src_ip] = src_mac
    arp_reply = create_ip_arp_reply(self.net.interface_by_ip
    self.net.send_packet(ifaceName,arp_reply)

elif arp.operation ==ArpOperation.Reply:
    if eth.src == 'ff:ff:ff:ff:ff:ff':
        return
    self.arp_table[src_ip] = src_mac
    if src_ip in self.waiting_ip.keys():
        for packet in self.waiting_packet[src_ip]:
            packet[Ethernet].src = self.net.interface_by_name
            packet[Ethernet].dst = src_mac
            packet[IPv4].ttl -=1
            self.net.send_packet(ifaceName,packet)
        del self.waiting_packet[src_ip]
    del self.waiting_ip[src_ip]

```

Timeout: 对于发送的 arp 请求，没收到回复的每 1s 重发一次，重发 5 次还未收到的则将其对应的包和等待 ip 表删除。

```

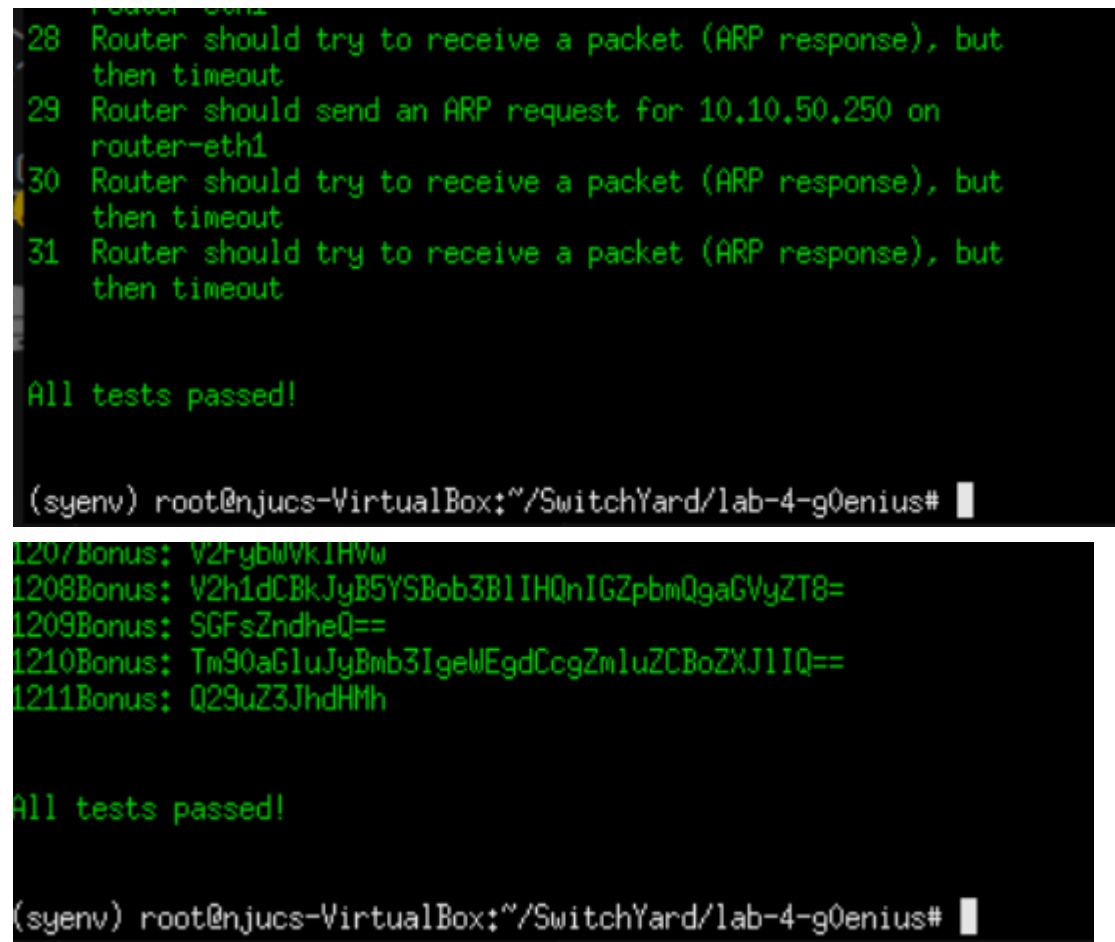
def handle_timeout(self):
    for ip in list(self.waiting_ip.keys()):
        timestamp = self.waiting_ip[ip][0]
        retries = self.waiting_ip[ip][1]
        if time.time() -timestamp >1:
            if retries >=5:
                del self.waiting_ip[ip]
                del self.waiting_packet[ip]
            else:
                self.waiting_ip[ip] =(time.time(),retries+1)
                arp_request = create_ip_arp_request(
                self.net.interface_by_name(self.get_forwarding_
                self.net.interface_by_name(self.get_forwarding_
                self.net.send_packet(s选择编码 forwarding_entry(

```

Handle_packet:

检查以太网地址，分为 ipv4 和 arp 情况进行处理，其他情况不处理。

测 试 通 过 截 图 :



```
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!

(syenv) root@njucs-VirtualBox:~/SwitchYard/lab-4-g0enius#
```

```
1207Bonus: V2FygbWVKIHVw
1208Bonus: V2h1dCBkJyB5YSBob3B1IHQnIGZpbmQgaGVyZT8=
1209Bonus: SGFsZndheQ==
1210Bonus: Tm90aGluJyBmb3IgeWEgdCcgZmluZCBoZXJlIQ==
1211Bonus: Q29uZ3JhdHMh

All tests passed!

(syenv) root@njucs-VirtualBox:~/SwitchYard/lab-4-g0enius#
```

抓包结果:

使用指令 server1 ping -c 2 10.1.1.1，抓取到数据包如下:

lab_4_eth2.pcapng						
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help						
Apply a display filter ... <Ctrl-/> Expression...						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	40:00:00:00:00:03	Broadcast	ARP	42	Who
2	0.000019326	30:00:00:00:00:01	40:00:00:00:00:03	ARP	42	10.
3	0.100905062	192.168.100.1	10.1.1.1	ICMP	98	Ech
4	0.100931000	10.1.1.1	192.168.100.1	ICMP	98	Ech
5	0.996000510	192.168.100.1	10.1.1.1	ICMP	98	Ech
6	0.996017592	10.1.1.1	192.168.100.1	ICMP	98	Ech
7	5.113630644	30:00:00:00:00:01	40:00:00:00:00:03	ARP	42	Who
8	5.152522615	40:00:00:00:00:03	30:00:00:00:00:01	ARP	42	10.

Protocol	Length	Info
ARP	42	Who has 10.1.1.1? Tell 10.1.1.2
ARP	42	10.1.1.1 is at 30:00:00:00:00:01
ICMP	98	Echo (ping) request id=0x0957, seq=1/256, ttl=63 (reply in
ICMP	98	Echo (ping) reply id=0x0957, seq=1/256, ttl=64 (request i
ICMP	98	Echo (ping) request id=0x0957, seq=2/512, ttl=63 (reply in
ICMP	98	Echo (ping) reply id=0x0957, seq=2/512, ttl=64 (request i
ARP	42	Who has 10.1.1.2? Tell 10.1.1.1
ARP	42	10.1.1.2 is at 40:00:00:00:00:03

首先 server1 向 router 的 eth0 发送 ArpRequest, 试图获取 eth0 的 mac 地址, router 对其进行了回复, 随后 server1 向 router 发送 ICMP 数据包, 因为不知道 client 的 mac 地址, 所以路由器向 client 发送了 ArpRequest, client 向路由器回复, 这时在等待队列中的 ICMP 包即可正常发送在 server1、client 之间正常传递, 观察其 ttl 值, 发现都被正常减 1, 结果基本正确。