

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：洋溢	
学院	计算机	专业（方向）	计算机科学与技术		
学号	221220142	姓名	欧阳瑞泽		
Email	221220142@smail.nju.edu.cn	开始/完成日期	2024.3.22		

Lab 3: Forwarding Packets

1. 实验目的

实现router的主要功能，包括对ARP请求的回复，用ARP请求MAC地址，对转发表和ARP表的维护，对数据包的转发，对转发队列的处理等。

2. 实验内容

Step 1: IP Forwarding Table Lookup

构建IP转发表有两种方法，一种是静态地从配置好的转发表中读取，另一种是动态地扫描端口。由于本实验中网络结构不会发生变化，所以只需在创建对象时建立转发表即可。扫描端口时，将下一跳IP设置为None。使用时，通常需要用IP地址与IP转发表表项做前缀匹配，找到最大前缀匹配。使用手册中给出的API来操作。

Step 2: Cached ARP Table

Router工作的流程如下：

- 更新各种表项，移除超时的ARP表项，对时间间隔大于1s、请求次数小于5此的ARP请求重新发送，移除请求次数大于5次的ARP请求以及同一目标IP的待发送队列
- 接收数据包，判断以太头的目标地址是否为广播地址或者端口地址，若不是则丢弃。
- 判断数据包类型，分为ARP包和非ARP包，ARP

包采用 Lab3 中的处理方式：在判断 ARP 包是否合法时，首先要判断以太头的目的地址是否为传入端口，其次要判断查询 IP 是否为 router 的端口之一。要判断 ARP 包是否仅仅有两个头，而不是有 ARP 头则合法。根据 RFC1812，仅当 ARP 包是 Reply 类型且源 MAC 地址为广播地址时不更新 ARP 转发表，其他情况都选择相信并更新 ARP 表，并且所有的情况都需要 Reply。与 Lab3 不同的是当 ARP 表更新时，需要发送得知目标地址的数据包队列，顺序与进入 router 时相同。

非 ARP 包需要路由器进行转发，工作流程如下：

1. 若 IP 头的目标地址是 router 的端口，在本次 Lab 中丢弃此数据包，否则数据包的 TTL 减一。

2. 查找 IP 转发表中的最长前缀匹配，得知下一跳 IP 与端口，若下一条 IP 为 None，即说明此端口与目标地址在同一子网，下一跳地址即为数据包目标 IP 地址。

3. 查询 ARP 表，若未知下一条地址的 MAC，否则查看是否已发送目标 MAC 的 ARP 查询，若已发送但仍未响应，则将数据包加入待发送队列，若未发送，则发送第一个 ARP 询问，并打上时间戳。

4. 已知下一条地址的 MAC，需要修改数据包的以太头，即源地址为发送的端口地址，目标地址为下一跳 MAC。

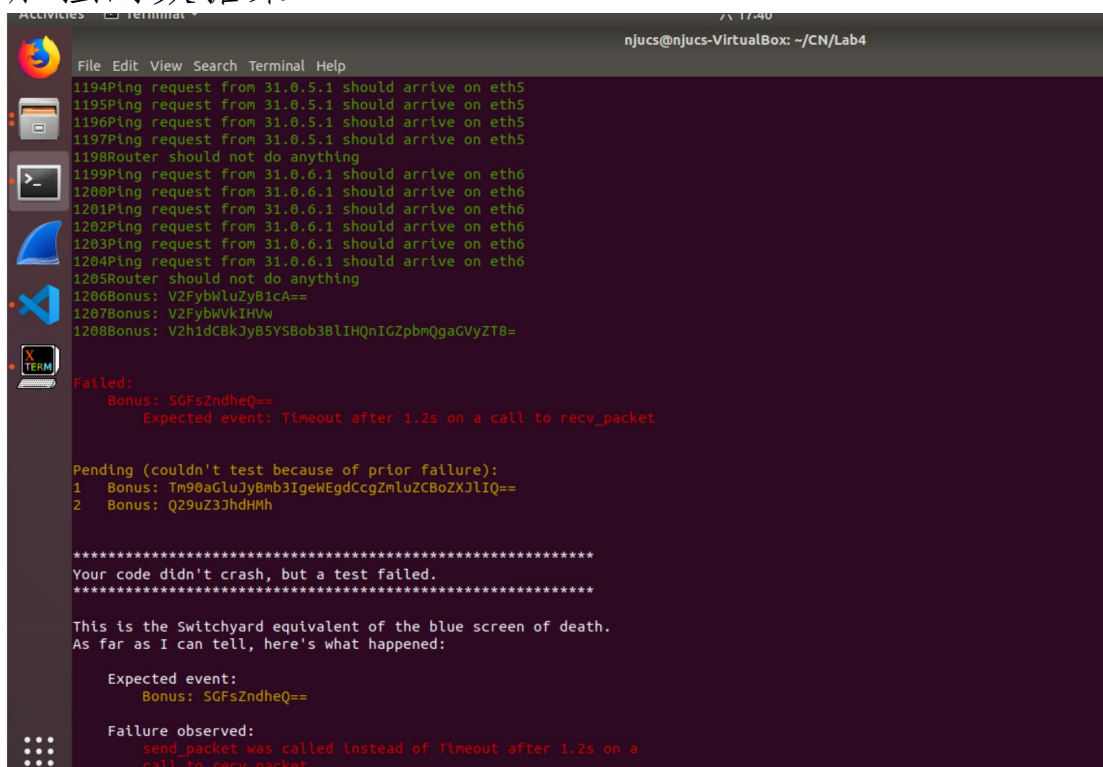
3. 实验结果

基本的数据集

```
njucs@njucs-virtualbox:
File Edit View Search Terminal Help
eth1
15 Application should try to receive a packet, but then timeout
16 Router should send another an 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 192.168.1.239 for 10.200.1.1 should arrive
    on router-eth0. No forwarding table entry should match.
20 An IP packet from 192.168.1.239 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!
```

加强的数据集



```
File Edit View Search Terminal Help
njucs@njucs-VirtualBox: ~/CN/Lab4
1194Ping request from 31.0.5.1 should arrive on eth5
1195Ping request from 31.0.5.1 should arrive on eth5
1196Ping request from 31.0.5.1 should arrive on eth5
1197Ping request from 31.0.5.1 should arrive on eth5
1198Router should not do anything
1199Ping request from 31.0.6.1 should arrive on eth6
1200Ping request from 31.0.6.1 should arrive on eth6
1201Ping request from 31.0.6.1 should arrive on eth6
1202Ping request from 31.0.6.1 should arrive on eth6
1203Ping request from 31.0.6.1 should arrive on eth6
1204Ping request from 31.0.6.1 should arrive on eth6
1205Router should not do anything
1206Bonus: V2FybWVudG91bnQ=
1207Bonus: V2FybWVudG91bnQ=
1208Bonus: V2h1dCBkcyB5YSB0b3B1IHQnIGZpbmQgaGVyZT8=

Failed:
  Bonus: SGFsZndheQ==
  Expected event: Timeout after 1.2s on a call to recv_packet

Pending (couldn't test because of prior failure):
1 Bonus: Tm90aGludG91bnQnIGVldCcgZmluZCB0ZXJlIQ==
2 Bonus: Q29uZ3JhdHMh

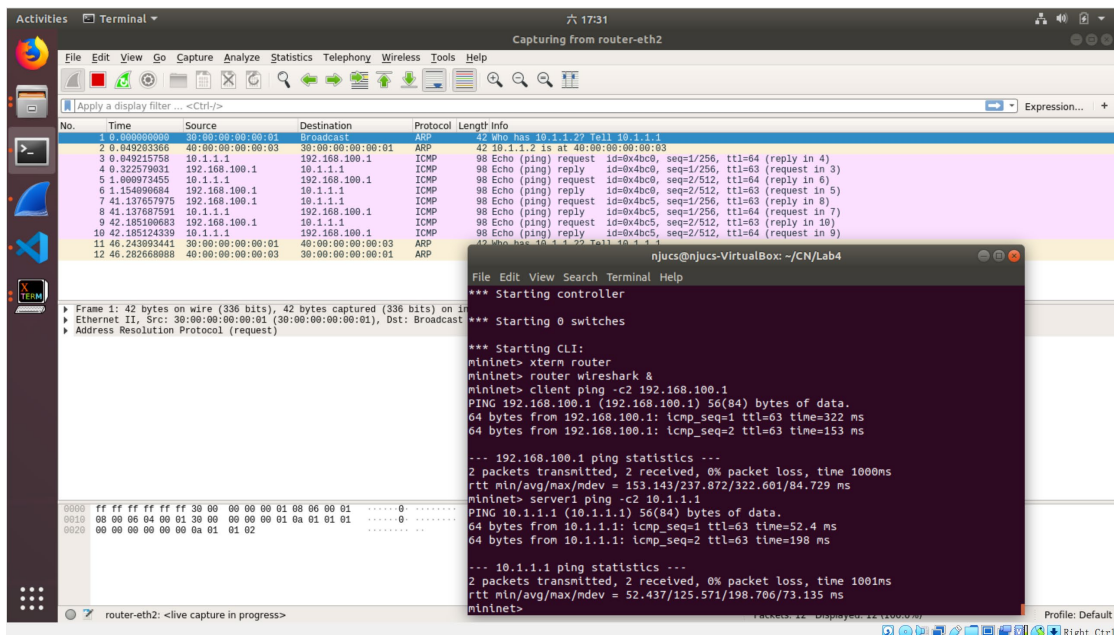
*****
Your code didn't crash, but a test failed.
*****

This is the Switchyard equivalent of the blue screen of death.
As far as I can tell, here's what happened:

  Expected event:
    Bonus: SGFsZndheQ==

  Failure observed:
    send_packet was called instead of Timeout after 1.2s on a
    call to recv_packet
```

在 mininet 中部署，用 wireshark 监测 router 的 eth2 接口，先让 client ping server1 两次，可以看见 client 先询问了网关的 MAC 地址，得到地址后发送两次 ICMP 数据包。可见 IP 头的源地址是 10.1.1.1，目标地址是 192.168.100.1，以太头的源地址是 30:00:00:00:00:01，目标地址是 40:00:00:00:00:03。再让 server1 ping client 两次，由于 router 在第一次 ping 时已知了 client 的 Mac，缓存了 ARP 表，所以直接发送 ICMP 包到 client。可见 IP 头的源地址是 192.168.100.1，目标地址是 10.1.1.1，以太头的源地址是 40:00:00:00:00:03，目标地址是 30:00:00:00:00:01。



4. 核心代码

```

elif packet.has_header(IPv4):
    return #just for Lab3
    IPv4_header = packet.get_header(IPv4)
    if IPv4_header.dst in self.myip:
        #for router, teckle in Lab5
        #assert False
        return
    else:
        IPv4_header.ttl -= 1
        if IPv4_header.ttl == 0:
            #teckle in Lab5
            #assert False
            return

    #find minmax prefix
    entry = self.prefix_match(IPv4_header.dst)

    if entry is None:
        #not match, teckle in Lab5
        return

    network_address, subnet_mask, next_hop_address, interface = entry

    Oface = self.net.interface_by_name(interface)
    eth_header = packet.get_header(Ethernet)

    if next_hop_address is None:
        if self.Arp_table.get(IPv4_header.dst) is None:
            arp_request_packet = create_ip_arp_request(Oface.ethaddr, Oface.ipaddr, IPv4_header.dst)
            if self.arp_dict.get(IPv4_header.dst) is None:
                self.arp_dict[IPv4_header.dst] = {'trytime': 1, 'timestamp': time.time(), 'intf': Oface, 'packet': arp_request_packet}
                self.net.send_packet(Oface, arp_request_packet)
                self.packet_queue[IPv4_header.dst] = [(Oface, packet)]
            else:
                if self.packet_queue.get(IPv4_header.dst):
                    self.packet_queue[IPv4_header.dst].append((Oface, packet))
                else:
                    self.packet_queue[IPv4_header.dst] = [(Oface, packet)]
        else:
            eth_header.dst = self.Arp_table[IPv4_header.dst]['MAC']
            eth_header.src = Oface.ethaddr
            packet[0] = eth_header

```

```

        eth_header.dst = self.Arp_table[IPv4_header.dst]['MAC']
        eth_header.src = Oface.ethaddr
        packet[0] = eth_header
        self.net.send_packet(Oface, packet)
    else:
        if self.Arp_table.get(next_hop_address) is None:
            arp_request_packet = create_ip_arp_request(Oface.ethaddr, Oface.ipaddr, next_hop_address)
            if self.arp_dict.get(next_hop_address) is None:
                self.arp_dict[next_hop_address] = {'trytime': 1, 'timestamp': time.time(), 'intf': Oface, 'packet': arp_request_packet}
                self.net.send_packet(Oface, arp_request_packet)
                self.packet_queue[next_hop_address] = [(Oface, packet)]
            else:
                if self.packet_queue.get(next_hop_address):
                    self.packet_queue[next_hop_address].append((Oface, packet))
                else:
                    self.packet_queue[next_hop_address] = [(Oface, packet)]
        else:
            eth_header.dst = self.Arp_table[next_hop_address]['MAC']
            eth_header.src = Oface.ethaddr
            packet[0] = eth_header
            self.net.send_packet(Oface, packet)

def update(self):
    for ip, info in list(self.Arp_table.items()):
        if time.time() - info['timestamp'] > ARP_TABLE_TIME_OUT:
            print(f"Arp table {ip} has timed out.")
            del self.Arp_table[ip]
            print("")
            self.print_arp_table()

    for dstip, info in list(self.arp_dict.items()):
        trytime, timestamp, packet, Oface = info['trytime'], info['timestamp'], info['packet'], info['intf']
        if trytime == MAX_TRY and time.time() - timestamp >= ARP_TIME_OUT:
            del self.packet_queue[dstip]
            del self.arp_dict[dstip]
        elif trytime < MAX_TRY and time.time() - timestamp >= ARP_TIME_OUT:
            self.arp_dict[dstip] = {'trytime': trytime + 1, 'timestamp': time.time(), 'intf': Oface, 'packet': packet}
            self.net.send_packet(Oface, packet)

```

用字典维护待发送的队列，key 为目标 Ip 地址，对应的为数据包队列，每次主循环 loop 接收数据包之前更新表项。

5. 总结与感想

Router 规则比较复杂，在 Lab3ARP 表的基础上增加了很多，加强数据中要判断的情况很多，比如 ARP 包的合法性，替换以太头时的处理。没有做 bonus 的部分。