Computer Network lab4 实验报告

姓名 曾闻天

学号 231880147

邮箱 2249428377@qq.com

完成日期 2025.5.6

1 实验名称

Lab 4: Forwarding Packets

2 实验目的

1.实现 IPv4 路由器的静态路由功能，支持数据包的接收与转发。
2.构建转发表，通过最长前缀匹配算法确定下一跳地址和出口接口。
3.实现 ARP 请求的发送与缓存管理，支持多次重试机制（最多 5 次）。
4.处理 IPv4 数据包的 TTL 字段修改及以太网头部重构

3 实验进行

3.1 构建转发表

接口信息加载：从路由器接口（net.interfaces()）获取直接连接的子网信息，生成转发表条目。
格式：(网络地址, 子网掩码, 下一跳地址（0.0.0.0）, 接口名称)。

文件加载：读取 forwarding_table.txt，添加静态路由条目。

示例条目：172.16.0.0 255.255.255.0 192.168.1.2 router-eth0。

```
def __init__(self, net: switchyard.llnetbase.LLNetBase):
        self.net = net
        # other initialization stuff here
        self.queue = []
        self.requesting = set()
        self.arptable = ARPCachedTable()
        self.table = set()

        for intf in self.net.interfaces():
            self.table.add( (intf.ipaddr, intf.netmask,
IPv4Address('0.0.0.0'), intf.name) )
        table_file = open('forwarding_table.txt')
        try:
            for line in table_file:
                table_entry = line.split()
                self.table.add( (IPv4Address(table_entry[0]),
IPv4Address(table_entry[1]), IPv4Address(table_entry[2]), table_entry[3]) )
        finally:
            table_file.close()
```

3.2 最长前缀匹配算法

匹配逻辑：对目标 IP 地址进行按位与操作，匹配子网掩码最长的条目。

实现函数：longest_prefix_match(ipaddr)返回下一跳 IP 和出口接口。

```
def longest_prefix_match(self, ipaddr: switchyard.llnetbase.IPv4Address):
        log_info(f"begin match")
        match_lenth = 0
        next_hopip = IPv4Address('0.0.0.0')
        match_itface = 'None'
        for ip, netmask, hopip, itface in self.table:
            if (int(ipaddr) & int(netmask)) == (int(ip) & int(netmask)) and
int(netmask) > match_lenth:
                match_lenth = int(netmask)
                match_itface = itface
                next_hopip = hopip
        log_info(next_hopip)
        log_info(match_itface)
        log_info(f"end match")
        return next_hopip, match_itface
```

3.3 ARP请求与缓存管理

缓存表设计：使用 ARPCachedTable 类管理 ARP 缓存，超时时间为 10 秒。

请求机制：若未命中缓存，发送 ARP 请求（最多 5 次，间隔 1 秒），失败后丢弃数据包。

多线程处理：通过线程实现异步发送请求，避免阻塞主线程。

```
def send_arp_request(self, targetipaddr: switchyard.llnetbase.IPv4Address,
itface, myhwaddr, myipaddr, recv):
        log_info(f"mul-sending {targetipaddr}")
        if targetipaddr in self.requesting:
            self.net.send_packet(itface, create_ip_arp_request(myhwaddr,
myipaddr, targetipaddr))
        for i in range(4):
            time.sleep(1)
            if targetipaddr not in self.requesting:
                return
            self.net.send_packet(itface, create_ip_arp_request(myhwaddr,
myipaddr, targetipaddr))
            log_info(f"mul-sending no.{i+1}")

        log_info(f"{recv} has no arp reply, del it")
        self.queue.remove(recv)
        self.requesting.remove(targetipaddr)
```

3.4 数据包转发

TTL 处理：IPv4 头部 TTL 字段减 1，并重新封装以太网头部。

转发逻辑：

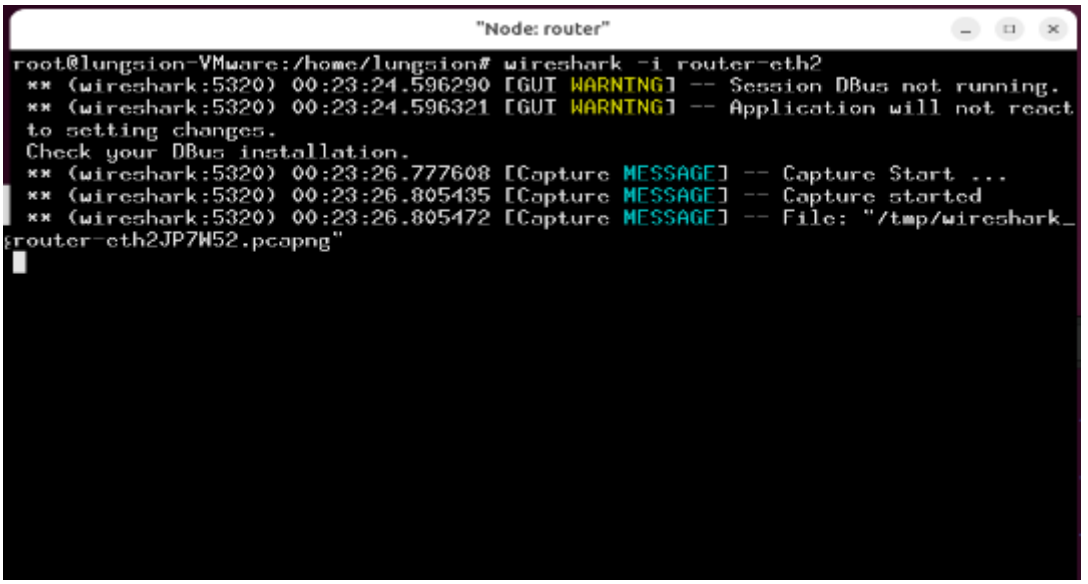
丢弃无效包（目标 MAC 不匹配、无匹配路由条目）。

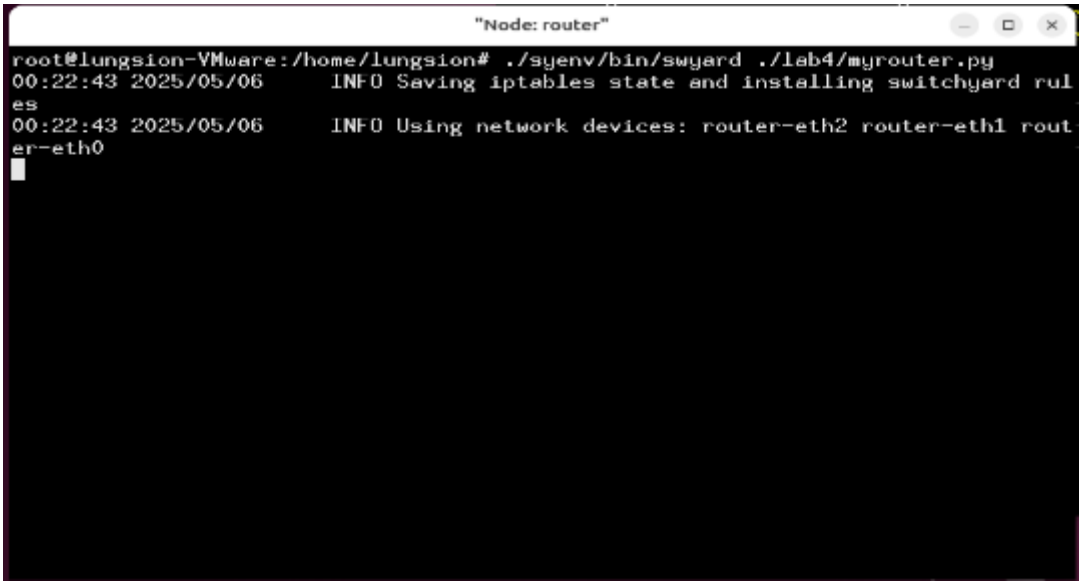
根据 ARP 缓存获取下一跳 MAC 地址，重构以太网头部后转发。

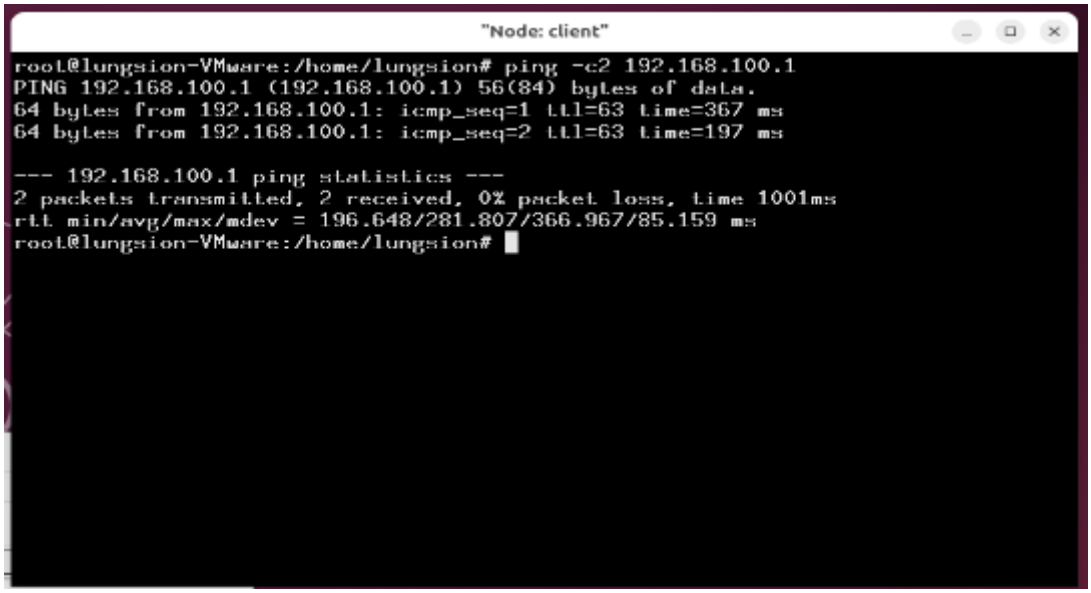
```
def handle_packet(self, recv: switchyard.llnetbase.ReceivedPacket):
        timestamp, ifaceName, packet = recv

        log_info(packet)

        #handle arp
        if 'Arp' in packet.headers():
            arp = packet.get_header(Arp)
            self.arptable.add(arp.senderprotoaddr, arp.senderhwaddr)
            if arp.senderprotoaddr in self.requesting:
                self.requesting.remove(arp.senderprotoaddr)
            if arp.operation == ArpOperation.Request:
                #log_info(f"ARP request received!{arp}")
                try:
                    targethwaddr =
self.net.interface_by_ipaddr(arp.targetprotoaddr).ethaddr
                except KeyError:
```

```
                    #log_info(f"No interface assigned to
{arp.targetprotoaddr}! Ignore it")
                    return
                self.net.send_packet(ifaceName,
create_ip_arp_reply(targethwaddr, arp.senderhwaddr, arp.targetprotoaddr,
arp.senderprotoaddr))
                log_info(f"Send ARP reply asking {arp.targetprotoaddr}")
            return

        #forward packet
        ipv4 = packet.get_header(IPv4)
        eth = packet.get_header(Ethernet)
        nexthop, match_itface = self.longest_prefix_match(ipv4.dst)
        if match_itface == 'None':
            log_info(f"remove packet {recv}")
            self.queue.remove(recv)
            return
        if str(nexthop) == '0.0.0.0':
            nexthop = ipv4.dst
        if nexthop in self.requesting:
            return
        if self.arptable.isintable(nexthop) == False:
            log_info(f"need to arp request")
            self.requesting.add(nexthop)
            t = threading.Thread(target=self.send_arp_request, args=(nexthop,
match_itface, self.net.interface_by_name(match_itface).ethaddr,
self.net.interface_by_name(match_itface).ipaddr, recv, ))
            t.start()
            return
        log_info(f"begin to edit header")
        eth.src = self.net.interface_by_name(match_itface).ethaddr
        eth.dst = self.arptable.get(nexthop)
        ipv4.ttl = ipv4.ttl - 1
        del packet[IPv4]
        del packet[Ethernet]
        packet.insert_header(0, ipv4)
        packet.insert_header(0, eth)
        log_info(f"remove packet {recv}")
        self.queue.remove(recv)
        self.net.send_packet(match_itface, packet)
```

4 实验结果

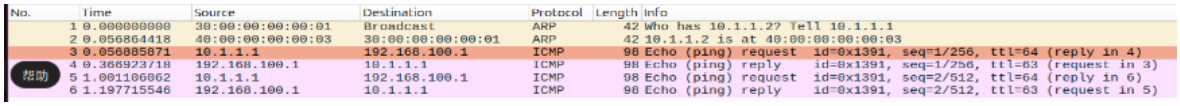

"Node: router"
root@lungsion-VMware:/home/lungsion# wireshark -i router-eth2
** (wireshark:5320) 00:23:24.596290 [GUI WARNING] -- Session DBus not running.
** (wireshark:5320) 00:23:24.596321 [GUI WARNING] -- Application will not react
to setting changes.
Check your DBus installation.
** (wireshark:5320) 00:23:26.777608 [Capture MESSAGE] -- Capture Start ...
** (wireshark:5320) 00:23:26.805435 [Capture MESSAGE] -- Capture started
** (wireshark:5320) 00:23:26.805472 [Capture MESSAGE] -- File: "/tmp/wireshark_
router-eth2JP7W52.pcapng"


"Node: router"
root@lungsion-VMware:/home/lungsion# ./syenv/bin/swyard ./lab4/myrouter.py
00:22:43 2025/05/06 INFO Saving iptables state and installing switchyard rul
es
00:22:43 2025/05/06 INFO Using network devices: router-eth2 router-eth1 rout
er-eth0



如

lab4.pcapng



5 实验总结

ARP 缓存竞争：通过 threading.Lock()保证缓存操作的原子性，避免多线程冲突。

路由表更新：文件 forwarding_table.txt 动态加载后，需重新计算网络地址，使用按位与操作解决。

ICMP 支持：当前丢弃无效包，后续需添加 TTL 超时、目的不可达等 ICMP 消息。

动态路由协议：结合 RIP 或 OSPF 实现动态路由表更新。

深入理解路由器转发逻辑、ARP 协议交互及多线程编程。

掌握最长前缀匹配算法的实现与优化方法。

致谢 在此,我向对本报告的工作给予支持和建议的老师和助教,南京大学南京大学智能软件与工程学院殷亚凤教授及其领导的助教们表示感谢.