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	**计算机科学与技术**	专业（方向）	**计算机科学与技术**
学号	**211220062**	姓名	**孔繁林**
Email	**2015098887@qq.com**	开始/完成日期	**11.6-11.7**

1. 实验名称

转发数据包

2. 实验目的

接收和转发到达链路并发往其他主机的数据包。转发过程的一部分是在转发表中执行地址查找（“最长前缀匹配”查找）。我们将在路由器中仅使用“静态”路由，而不是 RIP 或 OSPF 等动态路由协议。

对没有已知以太网 MAC 地址的 IP 地址发出 ARP 请求。路由器通常必须将数据包发送到其他主机，并且需要以太网 MAC 地址才能执行此操作。

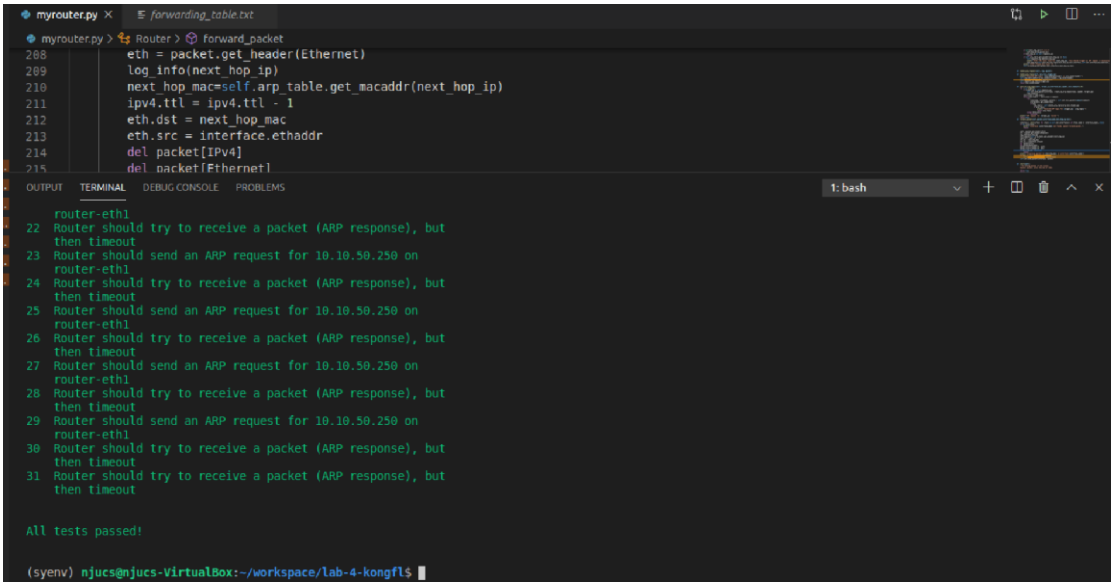
3. 实验内容

IP 转发表查找

转发数据包和 ARP

4. 实验结果

swyard -t testcases/testscenario2.srpy myrouter.py：



构建转发表

```
class RoutingTableEntry:…

class RoutingTable:…

class Router(object):
    def __init__(self, net: switchyard.llnetbase.LLNetBase):
        self.net = net
        self.arp_table=ARPTable(timeout=100)
        self.routing_table=RoutingTable()
        self.pending_arp_requests = []
        self.requests_ip=set()
        for interface in self.net.interfaces():
            self.routing_table.addEntry(interface.ipaddr, IPv4Address(interface.netmask), IPv4Address('0.0.0.0'), interface.name)
            print(f"Adding entry: {interface.ipaddr} {'255.255.255.0'}")
        table_file = open('forwarding_table.txt')
        try:
            for line in table_file:
                table_entry = line.split()
                self.routing_table.addEntry( IPv4Address(table_entry[0]), IPv4Address(table_entry[1]), IPv4Address(table_entry[2]
        finally:
            table_file.close()
        # other initialization stuff here
```

将目标 IP 地址与转发表匹配

```
class RoutingTable:
    def __init__(self):
        self.entries = []

    def addEntry(self,network,netmask,next_hop,interface):
        entry = RoutingTableEntry(network, netmask, next_hop, interface)
        self.entries.append(entry)

    def lookup(self,ip_address):
        print("begin match.")
        next_hop_ip=IPv4Address('0.0.0.0')
        next_interface='None'
        best_prefix_len=0
        ip=IPv4Address(ip_address)

        for entry in self.entries:
            if (int(ip_address) & int(entry.netmask)) == (int(entry.network)& int(entry.netmask)) and entry.prefixlen>best_prefix
                next_hop_ip=entry.next_hop
                next_interface=entry.interface
                best_prefix_len=entry.prefixlen

        log_info(next_hop_ip)
        log_info(next_interface)
        return next_hop_ip,next_interface
```

发送 ARP 请求并转发数据包

```
if packet.has_header(IPv4):
    ip_packet = packet.get_header(IPv4)
    next_hop_ip,next_interface = self.routing_table.lookup(ip_packet.dst)
    if next_interface =='None':
        self.pending_arp_requests.remove(recv)
        return
    if ip_packet.ttl <= 0:
        print(f"IPv4:::Dropping packet with TTL=0 from {ip_packet.src}")
        return

    print(f"IPv4:::Received packet destined for {ip_packet.dst}")

    if ip_packet.dst in [interface.ipaddr for interface in self.net.interfaces()]:
        print(f"IPv4:::Packet destined for router itself, dropping packet: {ip_packet.dst}")
        return

    if str(next_hop_ip)=='0.0.0.0':
        next_hop_ip=ip_packet.dst
    if next_hop_ip in self.requests_ip:
        return
    if self.arp_table.get_macaddr(next_hop_ip) == None:
        self.requests_ip.add(next_hop_ip)
        print(f"IPv4:::No route found for {next_hop_ip}. This should trigger an ARP request if necessary.")
        next_hop_mac=self.wait_for_arp_reply(next_hop_ip,next_interface,self.net.interface_by_name(next_interface).ethadd
    print(f"IPv4:::Found best r next_interface: str
    self.forward_packet(packet,next_interface,next_hop_ip,recv)
```

处理 ARP 查询

```
def wait_for_arp_reply(self, target_ip,interface,mac,ipaddr,recv,timeout=1.0):
    for i in range(5):
        if target_ip in self.requests_ip:
            self.net.send_packet(interface, create_ip_arp_request(mac, ipaddr, target_ip))
            time.sleep(1)
        start_time = time.time()
        while time.time() - start_time < timeout:
            try:
                timestamp, ifaceName, packets = self.net.recv_packet(timeout=timeout)
                arp=packets.get_header(Arp)
                if arp:
                    arp_reply = self.handle_arp_reply(arp,recv,target_ip)
                    if arp_reply:
                        print(f"Received ARP reply for {target_ip}: {arp_reply}")
                        return arp_reply
            except NoPackets:
                continue
    print(f"ARP request for {target_ip} failed ")
    return None
```

转发 IP 数据包

```python
def forward_packet(self,packet,interface_name,next_hop_ip,recv):

    interface = next((iface for iface in self.net.interfaces() if iface.name == interface_name), None)
    if not interface:
        print(f"Interface {interface_name} not found, cannot forward packet.")
        return

    ipv4 = packet.get_header(IPv4)
    eth = packet.get_header(Ethernet)
    log_info(next_hop_ip)
    next_hop_mac=self.arp_table.get_macaddr(next_hop_ip)
    ipv4.ttl = ipv4.ttl - 1
    eth.dst = next_hop_mac
    eth.src = interface.ethaddr
    del packet[IPv4]
    del packet[Ethernet]
    packet.insert_header(0, ipv4)
    packet.insert_header(0, eth)
    if next_hop_mac ==None:
        return
    print(f"Forwarding packet to {next_hop_mac} on interface {interface_name}")
    if recv in self.pending_arp_requests:
        self.pending_arp_requests.remove(recv)
    self.net.send_packet(interface, packet)
```

Client ping -c 2 192.168.100.1