

计算机网络实验四：IPv4 路由器实验报告

一、实验目的

1. 理解IPv4路由器的基本工作原理
2. 实现一个基本的IPv4路由器，包括：
 - 静态路由表的加载和查询
 - ARP缓存表的维护
 - IP数据包的转发
 - ARP请求和响应的处理

二、实验环境

- 编程语言：Python3
- 网络模拟框架：Switchyard
- 网络拓扑：

```
server1
  \
   router----client
  /
server2
```

三、实验内容与实现

3.1 路由器基本结构

路由器类（Router）的主要组件包括：

1. 接口信息管理

```
self.interfaces = self.net.interfaces()
self.ip_mac_map = {}
for intf in self.interfaces:
    self.ip_mac_map[intf.ipaddr] = intf.ethaddr
```

2. ARP缓存表

```
self.arp_table = {} # 格式: {ip: (mac, timestamp)}
self.arp_timeout = 100 # ARP表项超时时间 (秒)
self.arp_retry_count = 5 # ARP请求重试次数
self.arp_request_timeout = 1.0 # ARP请求超时时间 (秒)
```

3. 转发表管理

```
self.forwarding_table = [] # 存储转发表项
```

3.2 关键功能实现

3.2.1 转发表加载与查询

1. 加载转发表:

```
def load_forwarding_table(self):
    with open('forwarding_table.txt', 'r') as f:
        for line in f:
            network, netmask, nexthop, interface = line.strip().split()
            entry = {
                'network': IPv4Address(network),
                'netmask': IPv4Address(netmask),
                'nexthop': IPv4Address(nexthop) if nexthop != '-' else None,
                'interface': interface
            }
            self.forwarding_table.append(entry)
```

2. 最长前缀匹配查询:

```
def lookup_forwarding_table(self, ip_addr):
    longest_match = None
    longest_prefix_len = -1

    for entry in self.forwarding_table:
        network_bits = IPv4Address(int(entry['netmask']))._ip
        prefix_len = bin(network_bits).count('1')

        if int(ip_addr) & int(entry['netmask']) == int(entry['network']):
            if prefix_len > longest_prefix_len:
                longest_prefix_len = prefix_len
                longest_match = entry

    return longest_match
```

3.2.2 ARP表维护

1. ARP表更新:

```
def update_arp_table(self):
    current_time = time.time()
    to_delete = []
    for ip, (mac, timestamp) in self.arp_table.items():
        if current_time - timestamp > self.arp_timeout:
            to_delete.append(ip)
    for ip in to_delete:
        del self.arp_table[ip]
```

2. ARP请求发送:

```
def send_arp_request(self, target_ip, interface):
    arp_request = create_ip_arp_request(
        self.ip_mac_map[interface.ipaddr],
        interface.ipaddr,
        target_ip
    )
    self.net.send_packet(interface.name, arp_request)
```

3.2.3 数据包处理

1. ARP包处理:

- 更新ARP缓存表

- 处理ARP请求并发送响应
- 定期清理过期表项

2. IP包转发：

- 检查TTL
- 查找转发表
- 确定下一跳
- 获取MAC地址（缓存或ARP请求）
- 转发数据包

3.3 关键算法说明

1. 最长前缀匹配算法：

- 遍历所有转发表项
- 计算网络前缀长度
- 检查IP地址是否匹配当前表项
- 更新最长匹配结果

2. ARP缓存更新策略：

- 使用时间戳记录每个表项的创建时间
- 定期检查并删除超时表项
- 接收到ARP响应时更新表项

四、实验结果与分析

4.1 功能验证

路由器实现了以下功能：

1. 正确加载和解析转发表
2. 维护ARP缓存表
3. 处理ARP请求和响应
4. 转发IP数据包
5. 实现最长前缀匹配

4.2 性能分析

1. ARP缓存:

- 超时时间设置为100秒
- 使用哈希表存储, 查询效率 $O(1)$
- 定期清理机制避免内存占用过大

2. 转发表查询:

- 使用线性查找实现最长前缀匹配
- 时间复杂度 $O(n)$, n 为转发表项数
- 可通过前缀树等数据结构优化

五、问题与解决方案

1. ARP请求超时处理:

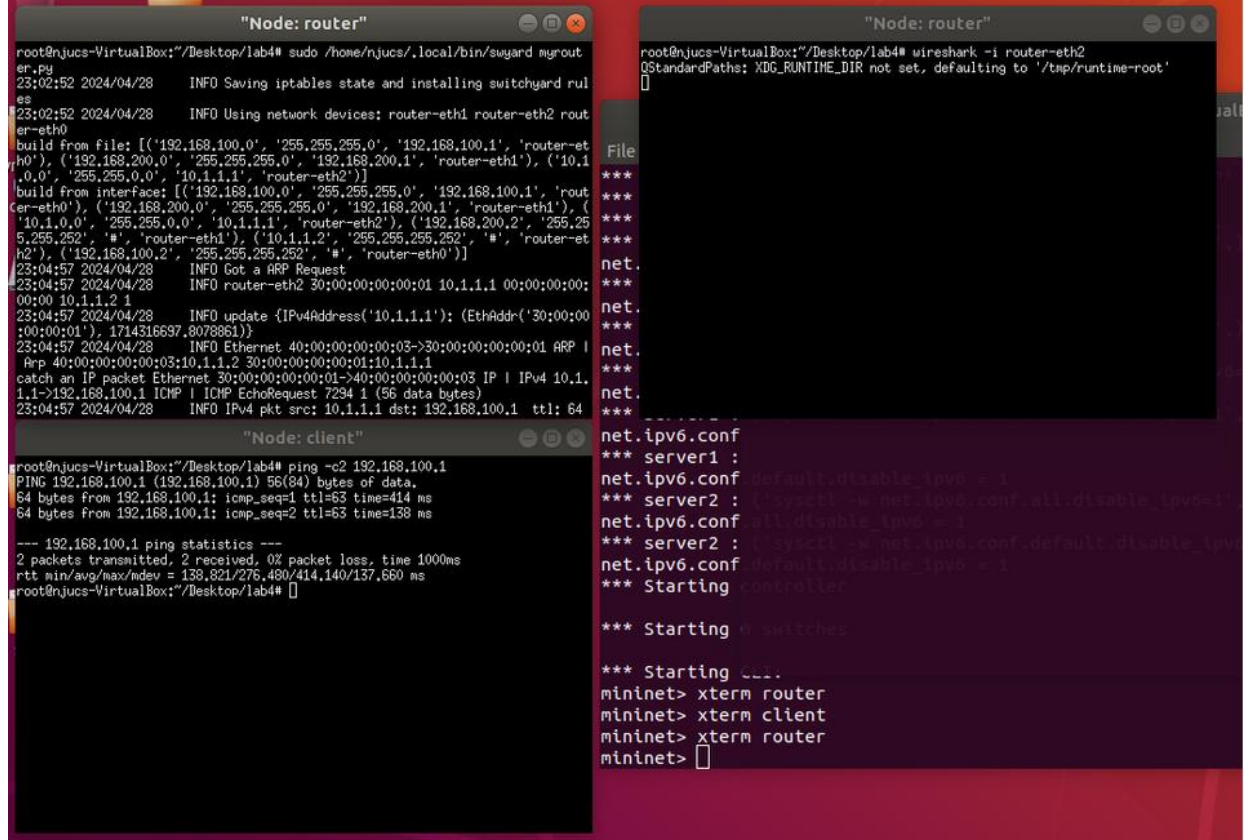
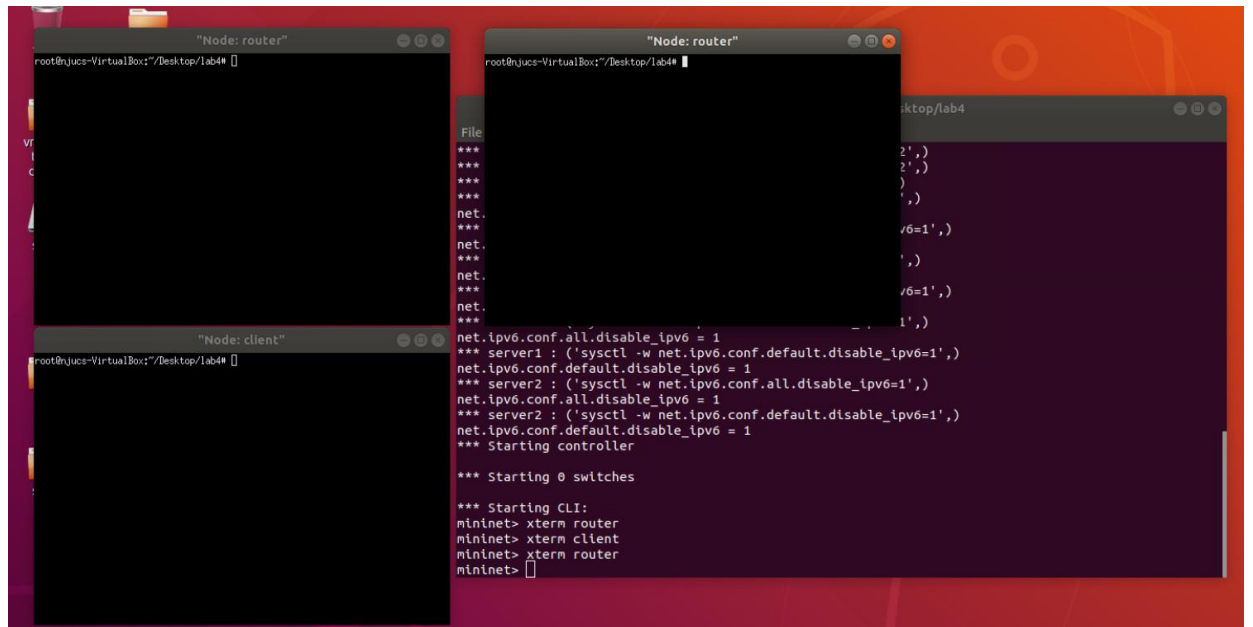
- 问题: ARP请求可能失败
- 解决: 实现重试机制, 最多重试5次

2. 转发表查询效率:

- 问题: 线性查找效率较低
- 解决: 可以考虑使用前缀树优化

```
njucs@njucs-VirtualBox: ~/Desktop/lab4
File Edit View Search Terminal Help
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!
```



| Apply a display filter ... <Ctrl-/> | | | | | |
|-------------------------------------|-------------|-------------------|-------------------|----------|---|
| No. | Time | Source | Destination | Protocol | Length Info |
| 1 | 0.00000000 | 30:00:00:00:00:01 | Broadcast | ARP | 42 Who has 10.1.1.2? Tell 10.1.1.1 |
| 2 | 0.108937330 | 40:00:00:00:00:03 | 30:00:00:00:00:01 | ARP | 42 10.1.1.2 is at 40:00:00:00:00:03 |
| 3 | 0.108946535 | 10.1.1.1 | 192.168.100.1 | ICMP | 98 Echo (ping) request id=0xc7e, seq=1/256, ttl=64 (reply in 4) |
| 4 | 0.414081566 | 192.168.100.1 | 10.1.1.1 | ICMP | 98 Echo (ping) reply id=0xc7e, seq=1/256, ttl=63 (request in 3) |
| 5 | 1.006610850 | 10.1.1.1 | 192.168.100.1 | ICMP | 98 Echo (ping) request id=0xc7e, seq=2/512, ttl=64 (reply in 6) |
| 6 | 1.139405798 | 192.168.100.1 | 10.1.1.1 | ICMP | 98 Echo (ping) reply id=0xc7e, seq=2/512, ttl=63 (request in 5) |

六、总结与展望

6.1 实验总结

成功实现了一个基本的IPv4路由器，包括：

- 静态路由表管理
- ARP协议处理
- IP数据包转发
- 最长前缀匹配

6.2 可能的改进

1. 性能优化：

- 使用前缀树优化路由表查询
- 优化ARP缓存管理

2. 功能扩展：

- 支持动态路由协议
- 实现ICMP处理
- 添加访问控制功能