

Lab4

171830635 俞星凯

实验目的	<ol style="list-style-type: none">1. 设计和实现 IP 包的静态路由转发程序。2. 加深对链路层和网络层协议衔接及静态路由的理解。
数据结构说明	<ol style="list-style-type: none">1. 路由配置相关数据结构<pre>struct route_item{ char destination[16]; char gateway[16]; char netmask[16]; char interface[16]; }route_info[MAX_ROUTE_INFO]; int route_item_index=0; struct arp_table_item{ char ip_addr[16]; char mac_addr[18]; }arp_table[MAX_ARP_SIZE]; int arp_item_index =0; struct device_item{ char interface[14]; char ip_addr[16]; char mac_addr[18]; }device[MAX_DEVICE]; int device_index=0;</pre>上述数据结构分别定义了静态路由表，ARP 缓存和设备信息。2. 路由转发相关数据结构<pre>char recv_buf[256]; char send_buf[256]; struct sockaddr_ll src_ll; struct sockaddr_ll dest_ll;</pre>两个缓冲区用于存储收发的数据。两个 <code>sockaddr_ll</code> 类型变量用于处理收发的源 MAC 地址和目的 MAC 地址。

配置文件说明	<div>192.168.1.0 Router1 的配置文件如左。</div> <div>0.0.0.0</div> <div>255.255.255.0</div> <div>eth0</div> <div>192.168.2.0 可以分为 3 个部分：</div> <div>0.0.0.0</div> <div>255.255.255.0</div> <div>eth1 第一部分是 3 条路由规则，每条有 4 个表项。</div> <div>192.168.3.0</div> <div>192.168.2.2</div> <div>255.255.255.0</div> <div>eth1</div> <div>192.168.1.2</div> <div>00:0c:29:25:7f:51 第二部分是 2 条 ARP 缓存，每条有 2 个表项。</div> <div>192.168.2.2</div> <div>00:0c:29:1f:86:54</div> <div>eth0</div> <div>192.168.1.1</div> <div>00:0c:29:82:6c:74</div> <div>eth1 第三部分是 2 条设备信息，每条有 3 个表项。</div> <div>192.168.2.1</div> <div>00:0c:29:82:6c:7e</div>
程序设计的思路以及运行流程	<div><pre>graph TD; A[路由程序初始状态] --> B[读取配置文件并初始化路由程序中的数据结构]; B --> C[监听所有的网口]; C --> D[捕获到一个来自于本地网口的数据包]; D --> E[解析包头并判断是否转发]; E -- 否 --> C; E -- 是 --> F[更改包头并转发到响应的接口]; F --> C;</pre></div> <div>1. 运行 PC1 时创建 SOCK_DGRAM 类型套接字，则发送/接收的数据包会自动添加/去除以太网帧头部。从 argv[1] 获取目的 IP 地址，根据静态路由表获取对应网关和接口，在 ARP 缓存中查找网关对应的 MAC 地址，在设备信息中查找接口对应的设备号，对 sockaddr_ll 类型变量 dest_ll 做好赋值。在发送缓冲区内填入 IP，ICMP 的必要信息，即可发送 ICMP_ECHO 报文。</div>

	<p>2. 运行 Route1 和 Router2 是一样的。在接收缓冲区中判断目的 IP 是否为本地 IP，若是，则发送 ICMP_ECHOREPLY 报文，若否，则根据静态路由表转发。转发操作同上。需要注意的是，在本实验中 ICMP_ECHOREPLY 采用了简化实现，由于在 recvfrom 时操作系统为 sockaddr_ll 类型变量 src_ll 进行了赋值，故直接将接收缓冲区的内容拷贝到发送缓冲区并利用 src_ll 发送即可。</p> <p>3. 运行 PC2 与 Router 类似，只不过其不具备转发功能，只有 ICMP_ECHOREPLY 功能。</p>
运行结果 截图	<p>1. 在每个 PC 和 router 上运行 ifconfig，确保 IP 地址为空。</p> <p>PC1:</p> <pre>eth0 Link encap:Ethernet HWaddr 00:0c:29:25:7f:51 inet addr:192.168.1.106 Bcast:192.168.1.255 Mask:255.255.255.0 inet6 addr: fe80::20c:29ff:fe25:7f51/64 Scope:Link UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:441 errors:0 dropped:0 overruns:0 frame:0 TX packets:106 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:56557 (56.5 KB) TX bytes:16495 (16.4 KB) Interrupt:19 Base address:0x2024</pre> <p>PC2:</p> <pre>eth0 Link encap:Ethernet HWaddr 00:0c:29:bf:13:a5 inet addr:192.168.1.109 Bcast:192.168.1.255 Mask:255.255.255.0 inet6 addr: fe80::20c:29ff:febf:13a5/64 Scope:Link UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:512 errors:0 dropped:0 overruns:0 frame:0 TX packets:242 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:63224 (63.2 KB) TX bytes:23570 (23.5 KB) Interrupt:19 Base address:0x2024</pre> <p>Router1:</p> <pre>eth0 Link encap:Ethernet HWaddr 00:0c:29:82:6c:74 inet addr:192.168.1.105 Bcast:192.168.1.255 Mask:255.255.255.0 inet6 addr: fe80::20c:29ff:fe82:6c74/64 Scope:Link UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:148 errors:0 dropped:0 overruns:0 frame:0 TX packets:167 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:22803 (22.8 KB) TX bytes:17968 (17.9 KB) Interrupt:19 Base address:0x2024 eth1 Link encap:Ethernet HWaddr 00:0c:29:82:6c:7e inet addr:192.168.1.104 Bcast:192.168.1.255 Mask:255.255.255.0 inet6 addr: fe80::20c:29ff:fe82:6c7e/64 Scope:Link UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1 RX packets:77 errors:0 dropped:0 overruns:0 frame:0 TX packets:48 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:11347 (11.3 KB) TX bytes:8675 (8.6 KB) Interrupt:19 Base address:0x20a4</pre>

Router2:

```
eth0      Link encap:Ethernet  HWaddr 00:0c:29:1f:86:54
          inet addr:192.168.1.107  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe1f:8654/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:640 errors:0 dropped:0 overruns:0 frame:0
          TX packets:344 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:80363 (80.3 KB)  TX bytes:34718 (34.7 KB)
          Interrupt:19 Base address:0x2024

eth1      Link encap:Ethernet  HWaddr 00:0c:29:1f:86:5e
          inet addr:192.168.1.108  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe1f:865e/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:448 errors:0 dropped:0 overruns:0 frame:0
          TX packets:62 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:48381 (48.3 KB)  TX bytes:10572 (10.5 KB)
          Interrupt:19 Base address:0x20a4
```

2. 用 PC1 向 PC2 发送 ICMP_ECHO 报文，路由器转发报文，PC2 回复 ICMP_ECHOREPLY 报文。

PC1:

```
user@ubuntu:~$ sudo ./PC1 192.168.3.2
send ICMP request packets to 192.168.1.1(00:0c:29:82:6c:74)
src: 192.168.1.2, dst: 192.168.3.2

receive an ICMP reply packet from 00:0c:29:82:6c:74
src: 192.168.3.2, dst: 192.168.1.2

receive an ICMP reply packet from 00:0c:29:82:6c:74
src: 192.168.3.2, dst: 192.168.1.2

receive an ICMP reply packet from 00:0c:29:82:6c:74
src: 192.168.3.2, dst: 192.168.1.2

receive an ICMP reply packet from 00:0c:29:82:6c:74
src: 192.168.3.2, dst: 192.168.1.2

receive an ICMP reply packet from 00:0c:29:82:6c:74
src: 192.168.3.2, dst: 192.168.1.2
```

Router1:


```
receive an ICMP reply packet from 00:0c:29:1f:86:54
src: 192.168.3.2, dst: 192.168.1.2
forward it to (00:0c:29:25:7f:51)

receive an ICMP request packet from 00:0c:29:25:7f:51
src: 192.168.1.2, dst: 192.168.3.2
forward it to (00:0c:29:1f:86:54)

receive an ICMP reply packet from 00:0c:29:1f:86:54
src: 192.168.3.2, dst: 192.168.1.2
forward it to (00:0c:29:25:7f:51)

receive an ICMP request packet from 00:0c:29:25:7f:51
src: 192.168.1.2, dst: 192.168.3.2
forward it to (00:0c:29:1f:86:54)

receive an ICMP reply packet from 00:0c:29:1f:86:54
src: 192.168.3.2, dst: 192.168.1.2
forward it to (00:0c:29:25:7f:51)
```

Router2:

```
receive an ICMP request packet from 00:0c:29:82:6c:7e
src: 192.168.1.2, dst: 192.168.3.2
forward it to (00:0c:29:bf:13:a5)

receive an ICMP reply packet from 00:0c:29:bf:13:a5
src: 192.168.3.2, dst: 192.168.1.2
forward it to (00:0c:29:82:6c:7e)

receive an ICMP request packet from 00:0c:29:82:6c:7e
src: 192.168.1.2, dst: 192.168.3.2
forward it to (00:0c:29:bf:13:a5)

receive an ICMP reply packet from 00:0c:29:bf:13:a5
src: 192.168.3.2, dst: 192.168.1.2
forward it to (00:0c:29:82:6c:7e)

receive an ICMP request packet from 00:0c:29:82:6c:7e
src: 192.168.1.2, dst: 192.168.3.2
forward it to (00:0c:29:bf:13:a5)
```

PC2:

	<pre> receive an ICMP request packet from 00:0c:29:1f:86:5e src: 192.168.1.2, dst: 192.168.3.2 reply to it receive an ICMP request packet from 00:0c:29:1f:86:5e src: 192.168.1.2, dst: 192.168.3.2 reply to it receive an ICMP request packet from 00:0c:29:1f:86:5e src: 192.168.1.2, dst: 192.168.3.2 reply to it receive an ICMP request packet from 00:0c:29:1f:86:5e src: 192.168.1.2, dst: 192.168.3.2 reply to it </pre>
相关参考资料	仅参考实验教材独立完成。
代码个人创新以及思考	本实验使用 SOCK_DGRAM 类型套接字，编程成功的关键有三：对 sockaddr_ll 类型变量的正确使用，静态路由表的查找以及 IP，IPICMP 报文的设置。本实验用传统的 C 语言实现，简洁高效。代码方面创新包括 ICMP_ECHOREPLY 的简化处理以及断言机制的应用。