实验四 静态路由编程实现

161130118 尹浚宇

实验目的

深入了解静态路由机制的工作方式
学会设计和实现一个简单的静态路由机制
加深对二三层协议衔接及静态路由的理解

数据结构说明

本次实验自定义的数据结构如下:

```
struct route_item
{
    char destination[16]; //目的子网
    char gateway[16]; //网关
    char netmask[16]; //子网掩码
    char interface[16]; //网卡名称
} route_table[MAX_ROUTE_SIZE];
int route_item_nr = 0;
```

route_table 对应于主机中的路由表，其中每一项对应于一个路由表项的简化版，因为这里只需要保证实现静态路由机制即可. route_item_nr 记录了路由表的表项数量. 该表在程序中的作用有，查询一个目的 ip 与本机是否处于同一子网，查询下一跳 ip，确定转发的源端口号.

```
struct arp_item
{
    char ip_addr[16]; //ip 地址
    char mac_addr[18]; //mac 地址
} arp_table[MAX_ARP_SIZE];
int arp_item_nr = 0;
```

arp_table 对应于主机中的 arp 表，该表在程序中的作用有，根据确定好的下一跳 ip 地址确定目的 mac 地址.

```
struct device_item
{
    char interface[14]; //网卡名称
    char mac_addr[18]; //mac 地址
    char ip_addr[16]; //ip 地址
    char netmask[16]; //子网掩码
} device_table[MAX_DEVICE_SIZE];
int device_item_nr = 0;
```

device_table 对应于主机中的设备表，其表项对应于 ifconfig 命令给出的参数，在程序中的作用有，根据源端口号确定源 ip 地址和源 mac 地址.

本次实验用到的系统库提供的结构体如下:

```
struct ip
{
    u_int ip_v:4; //版本
    u_int ip_hl:4; //报头长度
    u_char ip_tos; //服务类型
    u_short ip_len; //ip 包长度
    u_short ip_id; //标识号
    u_short ip_off; //分片标志
    u_char ip_ttl; //寿命
    u_char ip_p; //上层协议
    u_short ip_sum; //ip 头检验和
    struct in_addr ip_src; //源 ip 地址
    struct in_addr ip_dst; //目的 ip 地址
};
```

该结构体对应于 ip 头结构，用于解析和填充以太网帧的 ip 部分.

```
struct icmp
{
    u_char icmp_type; //报文类型
    u_char icmp_code; //报文类型子码
    u_short icmp_cksum; //icmp 头检验和
    u_short icmp_id; //标识号
    u_short icmp_seq; //顺序号
    char icmp_data[1]; //icmpdata
};
```

该结构体对应于 icmp 头结构，用于解析和填充以太网帧的 icmp 部分.

```
struct sockaddr_ll
{
    unsigned short int sll_family; /* 一般为 AF_PACKET */
    unsigned short int sll_protocol; /* 上层协议 */
    int sll_ifindex; /* 接口类型 */
    unsigned short int sll_hatype; /* 报头类型 */
    unsigned char sll_pkttype; /* 包类型 */
    unsigned char sll_halen; /* 地址长度 */
    unsigned char sll_addr[8]; /* MAC 地址 */
};
```

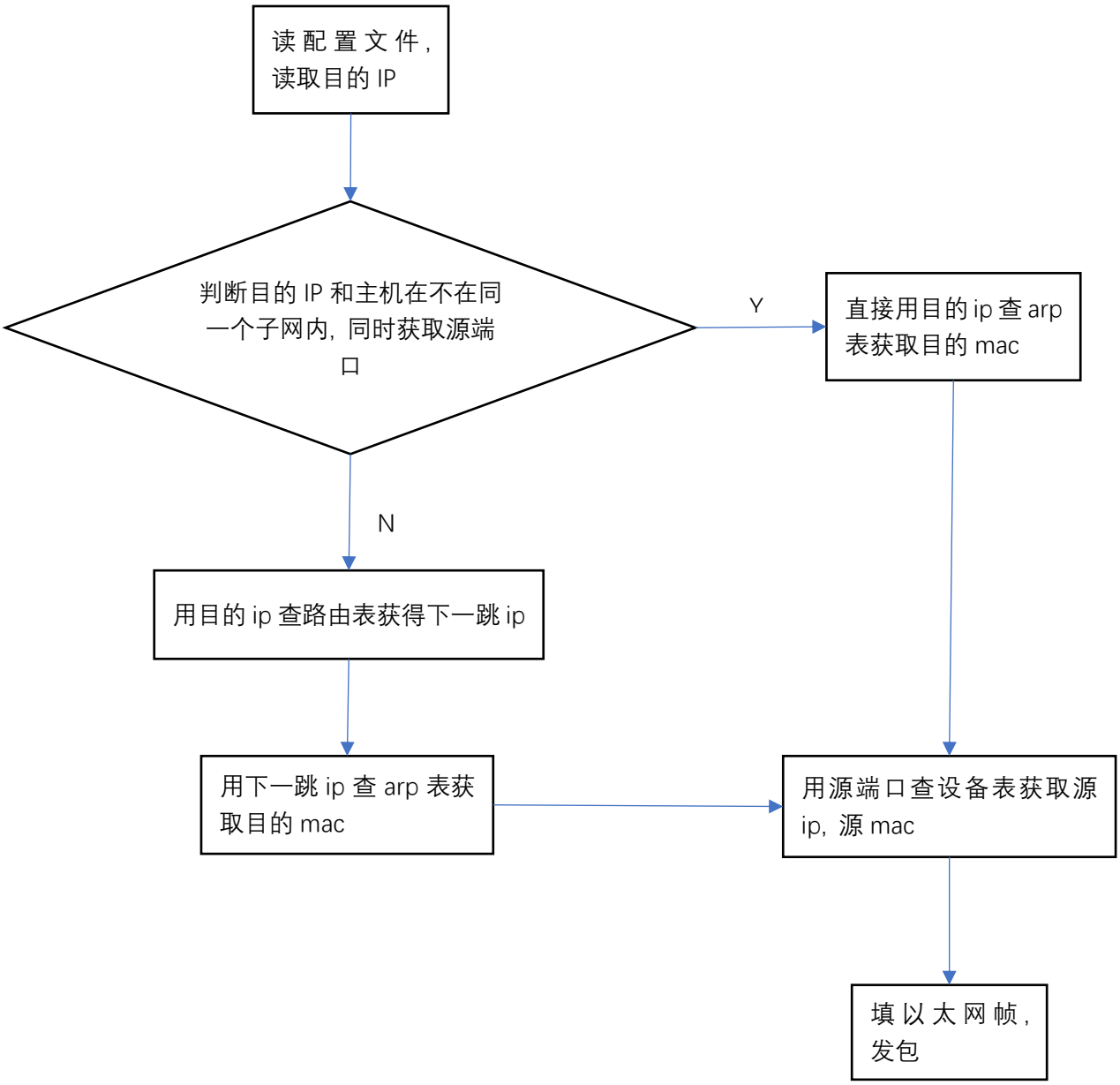
因为本次实验中需要发送和接受的都是以太网帧，所以这里使用该结构体配合 sendto 函数从而实现在数据链路层收发包的功能.

配置文件说明

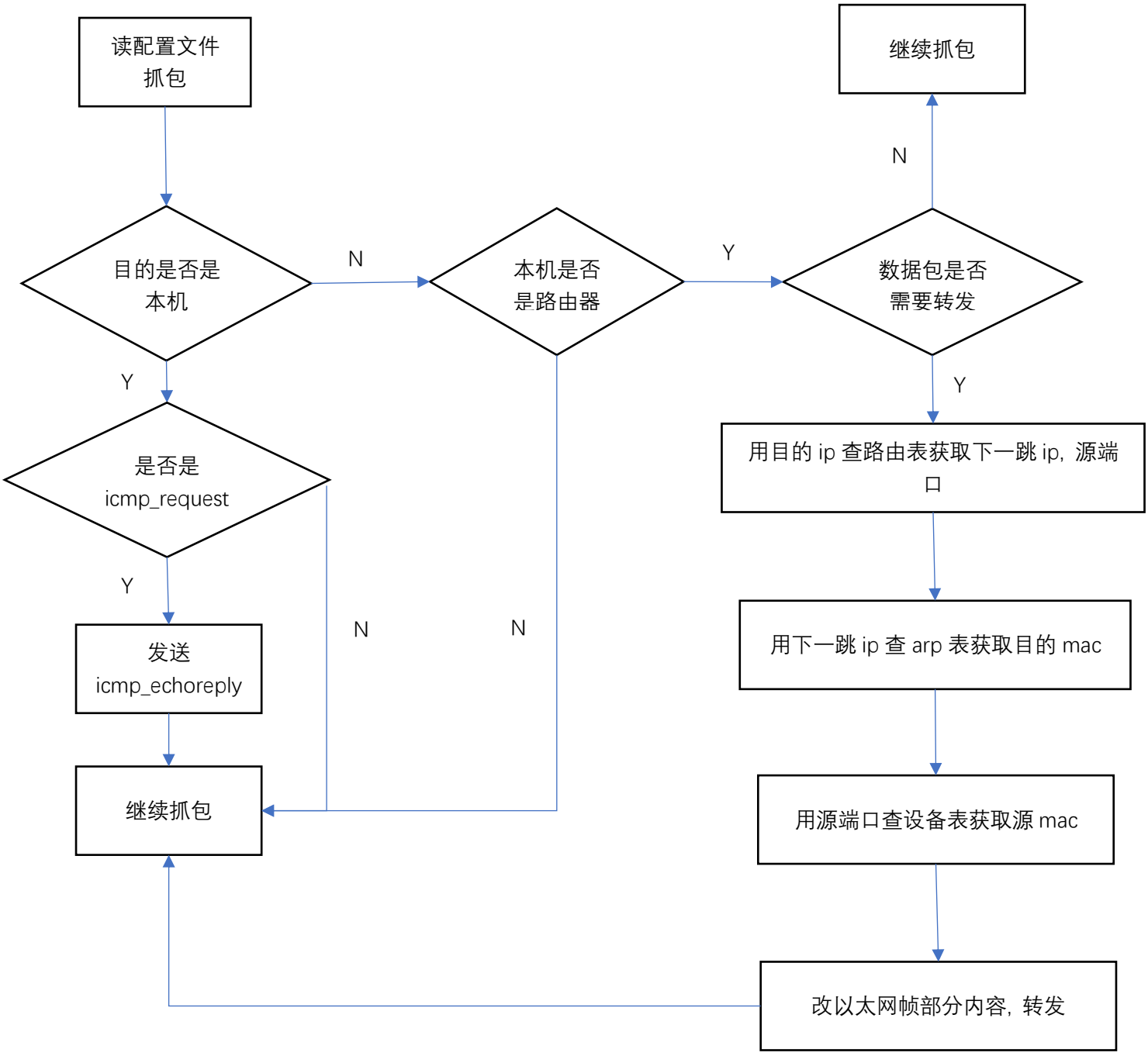
配置文件存放在 config 文件夹中，每台虚拟机有自己的配置文件，配置文件的读取已经硬编码到程序中，只需要将放有配置文件的文件夹放入可执行程序的同目录下，并且将文件夹更名为 config，运行程序后即可读取.

程序设计思路及运行流程

本次实验的静态路由机制通过两个程序完成，一个 send 程序运行在一个主机上负责发包，一个 recv 程序运行在每个主机上负责接受该主机收到的包，同时判断是否需要转发及回复.
send 程序流程图如下:

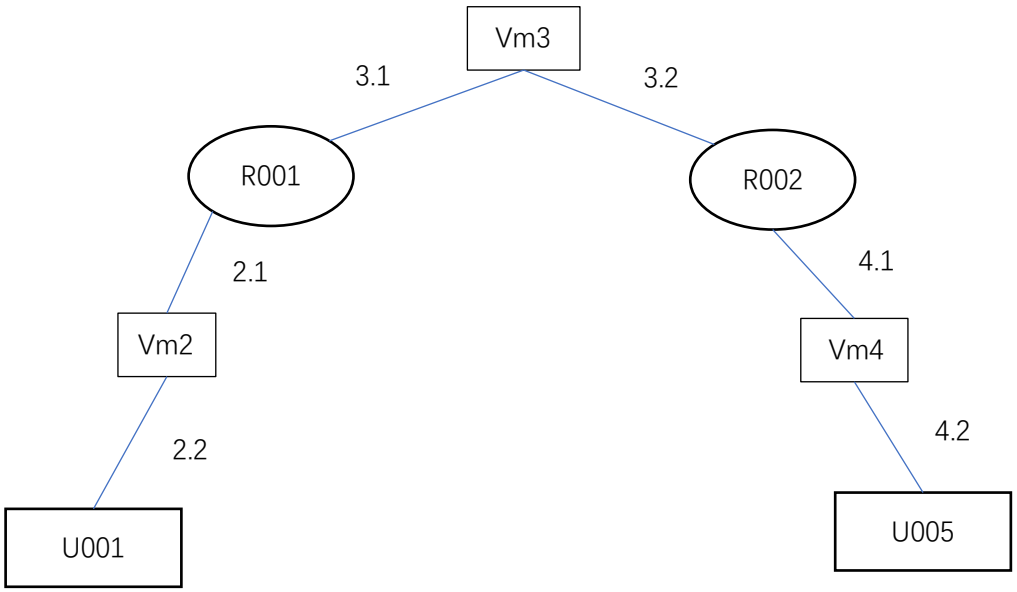


recv 程序的流程图如下:



运行结果截图

下面先给出网络拓扑图:



在 U001 上运行 send 的截图:

```
user1@ubuntu:~/lab4$ sudo ./send 192.168.4.2
[sudo] password for user1:
reading device table...          done
reading arp table...             done
reading route table...           done

device table has 1 items
ens33 00:0c:29:cb:88:0d 192.168.2.2 255.255.255.0

arp table has 1 items
192.168.2.1 00:0c:29:7a:d9:d4

route table has 3 items
192.168.2.0 0.0.0.0 255.255.255.0 ens33
192.168.3.0 192.168.2.1 255.255.255.0 ens33
192.168.4.0 192.168.2.1 255.255.255.0 ens33

not in same net, search ip_addr of next hop
ip_next_hop is 192.168.2.1
interface is ens33
dst mac is 00:0c:29:7a:d9:d4
src mac is 00:0c:29:cb:88:0d
send 0 times
send 1 times
send 2 times
send 3 times
```

在 U001 上运行 recv 的截图:

```
type: IP (0x0800)
receive a packet from 127.0.0.1 to 127.0.0.53
host is not a router, drop the packet

type: IP (0x0800)
receive a packet from 192.168.4.2 to 192.168.2.2
dst is myself
receive a icmp for reply

type: IP (0x0800)
receive a packet from 192.168.4.2 to 192.168.2.2
dst is myself
receive a icmp for reply

type: IP (0x0800)
receive a packet from 127.0.0.1 to 127.0.0.53
host is not a router, drop the packet
```

在 R001 上运行 recv 的截图:

```
type: IP (0x0800)
receive a packet from 192.168.2.2 to 192.168.4.2
dst is not myself, need forwarding
ip_next_hop is 192.168.3.2
interface is ens38
dst mac is 00:0c:29:7e:a4:6e
src mac is 00:0c:29:7a:d9:de

type: IP (0x0800)
receive a packet from 192.168.4.2 to 192.168.2.2
dst is not myself, need forwarding
ip_next_hop is 192.168.2.2
interface is ens33
dst mac is 00:0c:29:cb:88:0d
src mac is 00:0c:29:7a:d9:d4

type: IP (0x0800)
receive a packet from 192.168.2.2 to 192.168.4.2
dst is not myself, need forwarding
ip_next_hop is 192.168.3.2
interface is ens38
dst mac is 00:0c:29:7e:a4:6e
src mac is 00:0c:29:7a:d9:de
```

在 R002 上运行 recv 的截图:

```
type: IP (0x0800)
receive a packet from 192.168.2.2 to 192.168.4.2
dst is not myself, need forwarding
ip_next_hop is 192.168.4.2
interface is ens38
dst mac is 00:0c:29:c0:cd:2e
src mac is 00:0c:29:7e:a4:78

type: IP (0x0800)
receive a packet from 192.168.4.2 to 192.168.2.2
dst is not myself, need forwarding
ip_next_hop is 192.168.3.1
interface is ens33
dst mac is 00:0c:29:7a:d9:de
src mac is 00:0c:29:7e:a4:6e

type: IP (0x0800)
receive a packet from 192.168.2.2 to 192.168.4.2
dst is not myself, need forwarding
ip_next_hop is 192.168.4.2
interface is ens38
dst mac is 00:0c:29:c0:cd:2e
src mac is 00:0c:29:7e:a4:78
```

在 U005 上运行 recv 的截图:

```
type: IP (0x0800)
receive a packet from 192.168.2.2 to 192.168.4.2
dst is myself
receive a icmp for request
sending reply

type: IP (0x0800)
receive a packet from 192.168.2.2 to 192.168.4.2
dst is myself
receive a icmp for request
sending reply

type: IP (0x0800)
receive a packet from 192.168.2.2 to 192.168.4.2
dst is myself
receive a icmp for request
sending reply
```

从截图中可以看出, R001 和 R002 成功转发了从 U001 到 U005 和从 U005 到 U001 的数据包. 同时 U005 接收到 U001 的数据包后成功做出了回复, 在 U001 上也收到了 reply 的 ICMP 包.

相关参考资料

实验教材, ppt, QQ 群里大佬的发言, linux 手册中关于结构体的说明

对比样例程序

无

代码个人创新以及思考

分为两个模块执行，分工明确且更符合底层的运行逻辑. 将转发和回复综合在一个程序中执行，集成化程度更高. 同时代码中函数分工明确，注释详细.

该程序的应用场景

抓包程序能够捕捉以太网帧，而以太网帧里又包含 mac 地址信息，且一张网卡的 mac 地址是独一无二的，那么可以通过丢弃某个 mac 地址的所有数据包，来达到封禁某台主机的通信的作用.