Lab4 实验报告

一、 实验目的

1.进一步加强在 linux 下进行网络编程的能力，熟练掌握收发包的功能。
2.学会静态路由编程，加深对于二层、三层数据传输的理解。
3.高度模拟路由器的各类表项，更好地理解路由的本质。

二、 数据结构说明

1、以太头

```
struct Ether_head                    //以太头
{
    unsigned char dest_mac[6]; //目的MAC地址
    unsigned char src_mac[6];  //源MAC地址
    unsigned short frame_type; //类型
};
```

2、IP 头

```
struct ipheader                                //IP头
{
    unsigned char headlen:4, version:4; //首部长度、版本
    unsigned char service_type;         //服务类型
    unsigned short total_len;           //总长度
    unsigned short id;                  //标识
    unsigned short flag_offset;         //标志偏移量
    unsigned char ttl;                  //生存时间
    unsigned char proto;                //协议
    unsigned short head_checksum;       //首部校验和
    unsigned char src_ip[4];            //源IP地址
    unsigned char dest_ip[4];           //目的IP地址
};
```

3、ICMP 头

```
struct icmpheader                          //ICMP头
{
    unsigned char icmp_type;               //类型
    unsigned char icmp_code;               //代码
    unsigned short int icmp_cksum;         //校验和
    unsigned short int icmp_id;            //标识符
    unsigned short int icmp_seq;           //序号
};
```

4、路由表、ARP 缓存表、设备信息表

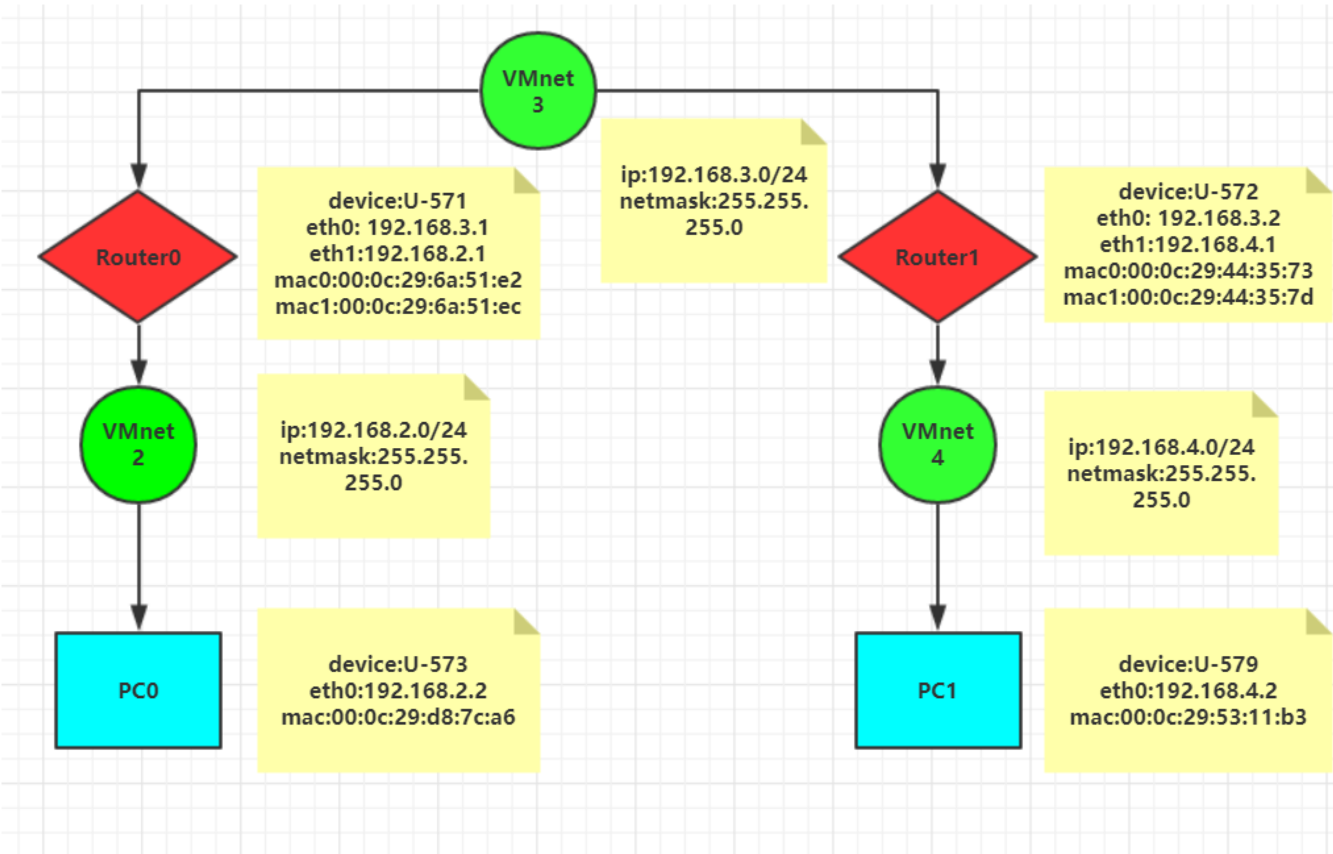
```c
//the information of the static routing table
struct route_item{
    unsigned char destination[4];
    unsigned char gateway[4];
    int if_index;
}route_info[MAX_ROUTE_INFO];

//the informaiton of the " my arp cache"
struct arp_table_item{
    unsigned char ip_addr[4];
    unsigned char mac_addr[6];
}arp_table[MAX_ARP_SIZE];

// the storage of the device , got information from configuration file : if.info
struct device_item{
    unsigned char local_ip_addr[4];
    unsigned char local_mac_addr[6];
}device[MAX_DEVICE];
```

三、 实验思路

1、实验搭建拓扑极其配置：



2、运行结果（截图为 wireshark 抓包）：

ICMP（Request 包）：

PC0(eth0) → Router0(eth1)

```
▶ Ethernet II, Src: Vmware_d8:7c:a6 (00:0c:29:d8:7c:a6), Dst: Vmware_6a:51:ec (00:0c:29:6a:51:ec)
▶ Internet Protocol Version 4, Src: 192.168.2.2 (192.168.2.2), Dst: 192.168.4.2 (192.168.4.2)
```

Router0(eth0) → Router1(eth0)

```
▶ Ethernet II, Src: Vmware_6a:51:e2 (00:0c:29:6a:51:e2), Dst: Vmware_44:35:73 (00:0c:29:44:35:73)
▶ Internet Protocol Version 4, Src: 192.168.2.2 (192.168.2.2), Dst: 192.168.4.2 (192.168.4.2)
```

Router1(eth1) → PC1(eth0)

```
▶ Ethernet II, Src: Vmware_44:35:7d (00:0c:29:44:35:7d), Dst: Vmware_53:11:b3 (00:0c:29:53:11:b3)
▶ Internet Protocol Version 4, Src: 192.168.2.2 (192.168.2.2), Dst: 192.168.4.2 (192.168.4.2)
```

ICMP（Reply 包）：

PC1(eth0) → Router1(eth1)

```
▶ Ethernet II, Src: Vmware_53:11:b3 (00:0c:29:53:11:b3), Dst: Vmware_44:35:7d (00:0c:29:44:35:7d)
▶ Internet Protocol Version 4, Src: 192.168.4.2 (192.168.4.2), Dst: 192.168.2.2 (192.168.2.2)
```

Router1(eth0) → Router0(eth0)

```
▶ Ethernet II, Src: Vmware_44:35:73 (00:0c:29:44:35:73), Dst: Vmware_6a:51:e2 (00:0c:29:6a:51:e2)
▶ Internet Protocol Version 4, Src: 192.168.4.2 (192.168.4.2), Dst: 192.168.2.2 (192.168.2.2)
```

Router0(eth1) → PC0(eth0)

```
▶ Ethernet II, Src: Vmware_6a:51:ec (00:0c:29:6a:51:ec), Dst: Vmware_d8:7c:a6 (00:0c:29:d8:7c:a6)
▶ Internet Protocol Version 4, Src: 192.168.4.2 (192.168.4.2), Dst: 192.168.2.2 (192.168.2.2)
```

3、核心代码分析

由于每个文件的基本功能相似，这里选取了 Router0 的代码进行分析。

接收包：

```c
int recvpack;   //定义的套接字
char recvgram[BUFFER_MAX];  //接收缓存区
//recv
recvpack = socket(AF_PACKET, SOCK_DGRAM, htons(ETH_P_IP));
struct sockaddr_ll addr;    //用来存储发送方的各方面的信息
socklen_t addr_len = sizeof(addr);
while(1)
{
    // addr中保存了链路层发送端的地址信息
    int recv = recvfrom(recvpack, recvgram, BUFFER_MAX, 0, (struct sockaddr *) &addr, &addr_len);
    if(recv < 64)
    {
        printf ("Fail to recv!!! %x\n", errno); //检测错误并且输出错误号
    }

    char *pt = recvgram;
    struct ipheader *iphead = (struct ipheader *)pt;
    if(strncmp(iphead->dest_ip, device[0].local_ip_addr, 4) == 0 || strncmp(iphead->dest_ip, device[1].local_ip_addr, 4) == 0)
    {                                           //通过设备信息表来看目的IP是不是自己以确定是否要转发
        printf ("Succeed to recv!!!\n");
    }
    else
    {
        int i;
        for(i = 0; i < MAX_ROUTE_INFO; i++)
        {
            if(strncmp(iphead->dest_ip, route_info[i].destination, 3) == 0)  //查看路由表对比目的地址
            {
                printf ("Succeed to recv!!!\n");
                int forward_index = route_info[i].if_index;  //得到转发的端口信息
                int j;
                for(j = 0; j < MAX_ARP_SIZE; j++)
                {
                    if(strncmp(arp_table[j].ip_addr, route_info[i].gateway, 4) == 0)  //查看ARP缓存表根据下一条网关找到下一条的MAC地址
                    {
                        unsigned char dest_mac_addr[6];
                        memcpy(dest_mac_addr, arp_table[j].mac_addr, 6);
```

发送包：

```c
int sendpack;   //发送套接字
char sendgram[BUFFER_MAX];  //发送的缓存
int datalen;   //数据区的长度
int packsize;  //包的总大小

sendpack = socket(AF_PACKET, SOCK_DGRAM, htons(ETH_P_IP));
struct sockaddr_ll dest_addr =
{
    .sll_family = AF_PACKET,          //目的地址的类型
    .sll_protocol = htons(ETH_P_IP),  //目的地址的协议
    .sll_halen = ETH_ALEN,            //MAC地址长度
    .sll_ifindex = forward_index,     //发送端口的信息
};

memcpy(&dest_addr.sll_addr, &dest_mac_addr, ETH_ALEN);    //将目的MAC地址拷入结构体

datalen = 64;
packsize = sizeof(struct ipheader) + sizeof(struct icmpheader) + datalen;
memset(sendgram, 0, BUFFER_MAX);
memcpy(sendgram, recvgram, packsize);     //复制包的内容

if (sendto(sendpack, sendgram, packsize, 0, (struct sockaddr *)&dest_addr, sizeof(dest_addr)) < 0) //sendto函数发包
    printf ("Fail to send!!! %x\n", errno);
else
    printf ("Succeed to send!!!\n");
break;
```

4、参考资料

主要参考了实验 PPT 中提供的各种与获取虚拟机信息和收发包有关的函数。

四、 实验的创新点

这次实验其实并没有什么创新点，从代码层面来讲的话，一开始写得相当地凌乱，因为第一要务是使程序能够跑起来，在完全实现实验要求后，对于代码进行了很大程度的封装，使得代码看起来精简和清晰了许多。然后还有一个创新的想法（虽然没有实现，但是很实用），就是在实际的路由器中，它的表项包含的东西一定是非常非常多的，因此可以再查找表项上搞点事情，比如哈希查找等等可以大幅提升程序效率。