实验 5：简单路由器程序的设计

2112065 李金霖 密码科学与技术

一、实验要求

（1）设计和实现一个路由器程序，要求完成的路由器程序能和现有的路由器产品（如思科路由器、华为路由器、微软的路由器等）进行协同工作。
（2）程序可以仅实现 IP 数据报的获取、选路、投递等路由器要求的基本功能。可以忽略分片处理、选项处理、动态路由表生成等功能。
（3）需要给出路由表的手工插入、删除方法。
（4）需要给出路由器的工作日志，显示数据报获取和转发过程。
（5）完成的程序须通过现场测试，并在班（或小组）中展示和报告自己的设计思路、开发和实现过程、测试方法和过程。

二、实验内容

路由表结构体如下：

```
typedef struct router_table {
    ULONG netmask;          //网络掩码
    ULONG desnet;           //目的网络
    ULONG nexthop;          //下一站路由
}router_table;
```

帧首部与 ip 首部如下：

```
typedef struct FrameHeader_t//帧首部
{
    BYTE DesMac[6];
    BYTE SrcMac[6];
    WORD FrameType;
}FrameHeader_t;

typedef struct IPHeader_t {     //IP首部
    BYTE    Ver_HLen;   //版本与协议类型
    BYTE    TOS;        //服务类型
    WORD    TotalLen;   //总长度
    WORD    ID;         //标识
    WORD    Flag_Segment; //标志和片偏移
    BYTE    TTL;        //生存周期
    BYTE    Protocol;   //协议
    WORD    Checksum;   //校验和
    ULONG   SrcIP;      //源IP地址
    ULONG   DstIP;      //目的IP地址
} IPHeader_t;
```

数据包内容与 ARP 帧如下：

```
typedef struct IPData_t {    //包含帧首部和IP首部的数据包
    FrameHeader_t   FrameHeader;
    IPHeader_t      IPHeader;
} IPData_t;

typedef struct ARPFrame_t//ARP帧
{
    FrameHeader_t FrameHeader;
    WORD HardwareType;
    WORD ProtocolType;
    BYTE HLen;
    BYTE PLen;
    WORD Operation;
    BYTE SendHa[6];
    DWORD SendIP;
    BYTE RecvHa[6];
    DWORD RecvIP;
}ARPFrame_t;
```

选路函数采用最长匹配原则

遍历路由表，将网络掩码与目的 IP 相与和目的网络比较，选择最长匹配然后保存表项下标。

```
ULONG search(router_table* t, int tLength, ULONG DesIP)//返回下一跳步的IP
{
    ULONG best_desnet = 0;  //最优匹配的目的网络
    int best = -1;   //最优匹配路由表项的下标
    for (int i = 0; i < tLength; i++)
    {
        if ((t[i].netmask & DesIP) == t[i].desnet) //目的IP和网络掩码相与后和目的网络比较
        {
            if (t[i].desnet >= best_desnet)//最长匹配
            {
                best_desnet = t[i].desnet;  //保存最优匹配的目的网络
                best = i;    //保存最优匹配路由表项的下标
            }
        }
    }
    if (best == -1)
        return 0xffffffff;      //没有匹配项
    else
        return t[best].nexthop;  //获得匹配项
}
```

路由表项添加与删除函数如下：

```
//向路由表中添加项（没有做插入时排序的优化）
bool additem(router_table* t, int& tLength, router_table item)
{
    if (tLength == RT_TABLE_SIZE)  //路由表满则不能添加
        return false;
    for (int i = 0; i < tLength; i++)
        if ((t[i].desnet == item.desnet) && (t[i].netmask == item.netmask) && (t[i].nexthop == item.nexthop))   //路由表中已存在该项，则不能添加
            return false;
    t[tLength] = item;   //添加到表尾
    tLength = tLength + 1;
    return true;
}

//从路由表中删除项
bool deleteitem(router_table* t, int& tLength, int index)
{
    if (tLength == 0)   //路由表空则不能删除
        return false;
    for (int i = 0; i < tLength; i++)
        if (i == index)   //删除以index索引的表项
        {
            for (; i < tLength - 1; i++)
                t[i] = t[i + 1];
            tLength = tLength - 1;
            return true;
        }
    return false;   //路由表中不存在该项则不能删除
}
```

重要的结构体有两个：路由表链表以及 ARP 表

```
router_table* rt = new router_table[RT_TABLE_SIZE];//把路由表项用链表串联起来
int rt_length = 0;//路由表的初始长度
```

```
int ARPtablesize = 0;
ARPFrame_t* ARPFrame3 = new ARPFrame_t[256];
```

ARP 表存储已经获取到的路由程序下一跳的目的 mac 地址和 ip 地址对应映射，这样就不用每次都根据路由表项中下一跳的 ip 地址重新构建 ARP 帧以截获下一跳的 mac 地址

路由器转发数据包具体流程如下：

首先，遍历输出所有网络接口卡信息，选择网络接口卡，然后构建虚拟的 ip 地址和 mac 地址给自己发包以截获自身的 mac 地址：

```
//////////////////////////////////////////////////////////////////////////////////////
    //向自己发送arp包，获取本机的MAC
BYTE scrMAC[6];
ULONG scrIP;
for (i = 0; i < 6; i++)
{
    scrMAC[i] = 0x66;
}
scrIP = inet_addr("112.112.112.112");//虚拟IP


for (d = alldevs, i = 0; i < in; i++, d = d->next);
for (a = d->addresses; a != NULL; a = a->next)
{
    if (a->addr->sa_family == AF_INET)
    {
        targetIP = inet_addr(inet_ntoa(((struct sockaddr_in*)(a->addr))->sin_addr));
        my_ip = targetIP;
    }
}
```

同时规定过滤器规则：只要 ARP 和 IP 包

```
u_int net_mask;
char packet_filter[] = "ip or arp";
struct bpf_program fcode;
net_mask = ((sockaddr_in*)(d->addresses->netmask))->sin_addr.S_un.S_addr;
if (pcap_compile(p, &fcode, packet_filter, 1, net_mask) < 0)
    //使用pcap_compile函数编译一个数据包过滤表达式（packet_filter）。如果编译成功，编译后的过滤程序将存储在fcode结构中
{
    printf("Unable to compile the packet filter.Check the syntax.\n");
    pcap_freealldevs(alldevs);
    return 0;
}
if (pcap_setfilter(p, &fcode) < 0)
    //将之前使用 pcap_compile 编译得到的过滤程序应用到一个 pcap 会话
{
    printf("Error setting the filter.\n");
    pcap_freealldevs(alldevs);
    return 0;
}
```

然后截获目的 ip 不是自身 ip 地址，目的 mac 为自身 mac 地址的包：

```
//////////////////////////////////////////////////////////////////////////////////////

    //获取目的mac为本机mac，目的ip非本机ip的ip数据报

    ULONG nextIP;//路由的下一站
    flag = 0;

    IPData_t* IPPacket;


    pcap_pkthdr* pkt_header = new pcap_pkthdr[1500];
    const u_char* pkt_data;

    int ARPtablesize = 0;
    ARPFrame_t* ARPFrame3 = new ARPFrame_t[256];
```

其目的是对这些包进行转发

```
//查arp表
for (int i = 0; i < ARPtablesize; i++) {
    if (ARPFrame3[i].SendIP == nextIP)
    {
        arpflag = 1;
        for (int j = 0; j < 6; j++) {
            its_mac[j] = ARPFrame3[i].SendHa[j];
        }

    }
}
```

查询 ARP 表内有无这样的映射关系，如果有说明之前已经对这样的包进行了转发，本次直接转发即可，如果没有的话就需要构建 ARP 帧去截获 mac 地址然后向 mac 地址发送数据包

```
targetIP = nextIP;

//组装ARP包
for (int i = 0; i < 6; i++)
{
    ARPFrame.FrameHeader.DesMac[i] = 0xff;
    ARPFrame.FrameHeader.SrcMac[i] = scrMAC[i];
    ARPFrame.SendHa[i] = scrMAC[i];
    ARPFrame.RecvHa[i] = 0;
}
ARPFrame.FrameHeader.FrameType = htons(0x0806);
ARPFrame.HardwareType = htons(0x0001);
ARPFrame.ProtocolType = htons(0x0800);
ARPFrame.HLen = 6;
ARPFrame.PLen = 4;
ARPFrame.Operation = htons(0x0001);
ARPFrame.SendIP = scrIP;
cout << "sendIP:";
printIP(ARPFrame.SendIP);
cout << endl;
ARPFrame.RecvIP = targetIP;
cout << "recvIP:";
printIP(ARPFrame.RecvIP);
cout << endl;
int send_ret = pcap_sendpacket(p, (u_char*)&ARPFrame, sizeof(ARPFrame_t));
```

截获 mac 地址后进行转发并添加至 ARP 表中

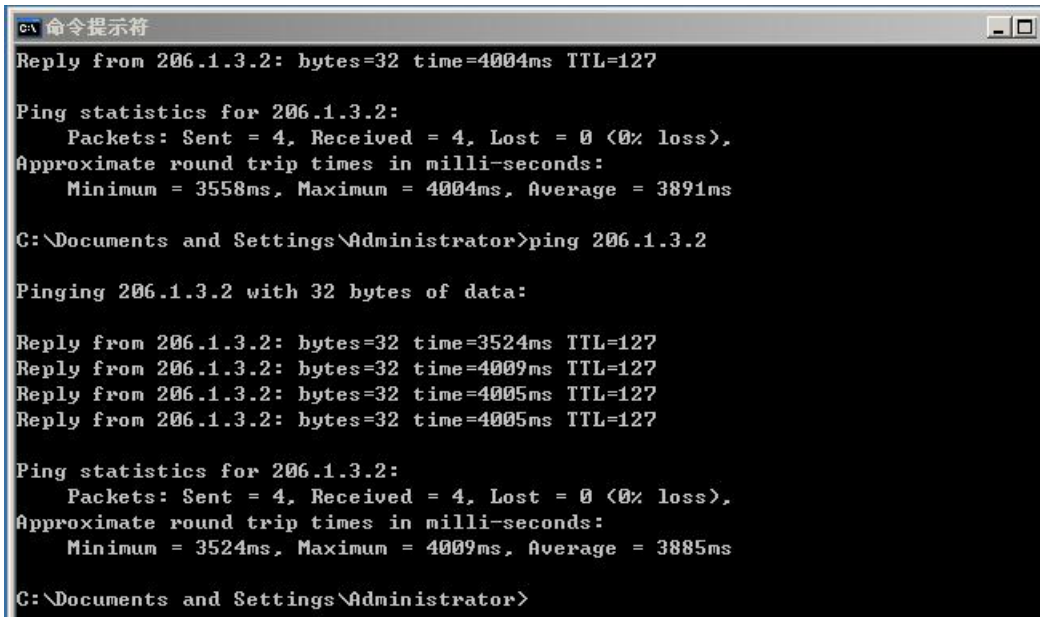
```
pcap_pkthdr* pkt_header2 = new pcap_pkthdr[1500];
const u_char* pkt_data2;

int res;
ARPFrame_t* ARPFrame2;

int flag1 = 0;
while (!flag1)
{
    res = pcap_next_ex(p, &pkt_header2, &pkt_data2);



    if ((res == 0))
    {
        continue;

    }
    if (res == 1)
    {
        ARPFrame2 = (ARPFrame_t*)pkt_data2;

        //新添加arp表项
        for (int i = 0; i < ARPtablesize; i++)
        {
            if (ARPFrame3[i].SendIP == ARPFrame2->SendIP && ARPFrame3[i].SendHa == ARPFrame2->SendHa)
            {
                cout << "ARP表已有";
                break;
            }
            ARPFrame3[ARPtablesize] = *ARPFrame2;
```

三、实验结果

```
1:
rpcap://\Device\NPF_{BF9BBE09-9BC0-4644-B6F2-E7D91703E342
Network adapter 'Microsoft' on local host

2:
rpcap://\Device\NPF_{C26EBE5B-5A9E-44C5-8FCA-3B82A5A8FD22
Network adapter 'Oracle' on local host
IP地址: 206.1.2.1
子网掩码: 255.255.255.0
广播地址: 255.255.255.255
IP地址: 206.1.1.1
子网掩码: 255.255.255.0
广播地址: 255.255.255.255

3:
rpcap://\Device\NPF_{875A8B2E-4C5C-4362-A34C-BFB684BB0FB2
Network adapter 'VMware Virtual Ethernet Adapter' on loca
IP地址: 192.168.83.1
子网掩码: 255.255.255.0
```

```
2
IP地址：206.1.2.1
子网掩码：255.255.255.0
广播地址：255.255.255.255
IP地址：206.1.1.1
子网掩码：255.255.255.0
广播地址：255.255.255.255
是否修改路由表项（y / n）
y
add or delete
add
请输入路由表，输入顺序为：目的网络号，子网掩码，下一跳步
206.1.3.0 255.255.255.0 206.1.2.2
continue or not?   y/n
n
路由表如下：
        网络掩码        目的网络        下一站路由
0  255.255.255.0 206.1.2.0 0.0.0.0
        网络掩码        目的网络        下一站路由
1  255.255.255.0 206.1.1.0 0.0.0.0
        网络掩码        目的网络        下一站路由
2  255.255.255.0 206.1.3.0 206.1.2.2
向自己发包成功

本机IP:206.1.1.1  本机MAC:a-0-27-0-0-13

正为该包进行转发
version=4 headlen=5 tos=0 totallen=60 id=0x138 ttl=128 protocol=1
数据包源地址：206.1.1.2   数据包目的地址：206.1.3.2
nextIP:206.1.2.2  sendIP:206.1.1.1  recvIP:206.1.2.2 发包成功
```

```
正为该包进行转发
version=4 headlen=5 tos=0 totallen=60 id=0x142 ttl=128 protocol=1
数据包源地址：206.1.1.2   数据包目的地址：206.1.3.2
nextIP:206.1.2.2  sendIP:206.1.1.1  recvIP:206.1.2.2 发包成功
NextIP的MAC地址:0-c-29-2a-30-e3
NextIP的IP:206.1.2.2
转发成功

源IP地址：206.1.1.2     目的IP地址：206.1.3.2
目的mac：0-c-29-2a-30-e3        源mac：a-0-27-0-0-13

正为该包进行转发
version=4 headlen=5 tos=0 totallen=60 id=0x140 ttl=127 protocol=1
数据包源地址：206.1.3.2   数据包目的地址：206.1.1.2
nextIP:206.1.1.2  sendIP:206.1.1.1  recvIP:206.1.1.2 发包成功
NextIP的MAC地址:0-c-29-b9-5e-1d
NextIP的IP:206.1.1.2
转发成功

源IP地址：206.1.3.2     目的IP地址：206.1.1.2
目的mac：0-c-29-b9-5e-1d        源mac：a-0-27-0-0-13
```



四、实验心得

通过本次实验加强了对于网技知识的理解，然后对于编程内容部分有了自己的理解，收获很大