南开大学

Nankai University

南　开　大　学

网络空间安全学院

网络技术与应用课程报告

简单路由器程序的设计

学号：2011897

姓名：任薏霖

年级：2020 级

专业：物联网工程

2022 年 12 月 23 日

第 1 节 实验内容说明

1. 简单路由器设计实验

防火墙实验在虚拟仿真环境下完成，要求如下：

(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。

第 2 节 实验准备

一、环境配置

（一）虚拟环境

本次实验在四台虚拟机下完成，分别是 1、2、3 和 4；

编号	IP	NetMask	说明
1	206.1.1.2	255.255.255.0	终端设备
2	206.1.1.1,206.1.2.1	255.255.255.0	路由器
3	206.1.2.2,206.1.3.1	255.255.255.0	路由器
4	206.1.3.2	255.255.255.0	终端设备

其中，1 号设备和 4 号设备为终端设备，2 号设备为你需要运行路由程序的设备，3 号设备为另一台路由器。所有 4 台设备均已安装 x86 的 VC++运行环境。

具体结构如下所示：

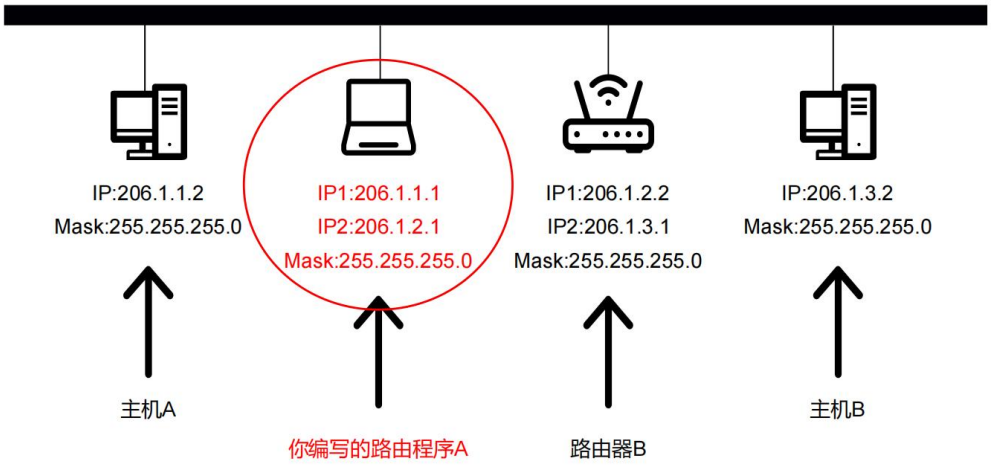

图 1 网络拓扑图

（二）关于 Visual Studio 的配置

由于虚拟环境使用的是 Windows Server2003，版本较老，因此应该尽量减少一些新的库函数的使用，具体配置如下：

- 依次点击【项目->属性->链接器->系统->所需最低版本】最低版本填写 5.01
- 编译选项为 Release 和 x86

- 配置 npcap 等环境（如以往实验）

由此，程序可以在虚拟环境中成功运行；

二、实验思路

在以往实验的基础上，本次实验的设计思路可以具体分成如下三部分：

（一）准备工作

- 打开网卡获取双 IP
- 伪造 ARP 报文获取本机 MAC
- 自动添加默认路由表项，手动添加&删除路由表项，显示路由表

（二）接收消息并准备转发

- 捕获报文
- 捕获 IP 报文的处理；
- 捕获 ARP 报文的处理

（三）转发

- MAC 地址的修改
- TTL 的修改
- 重新设置校验和

具体流程图如下所示：

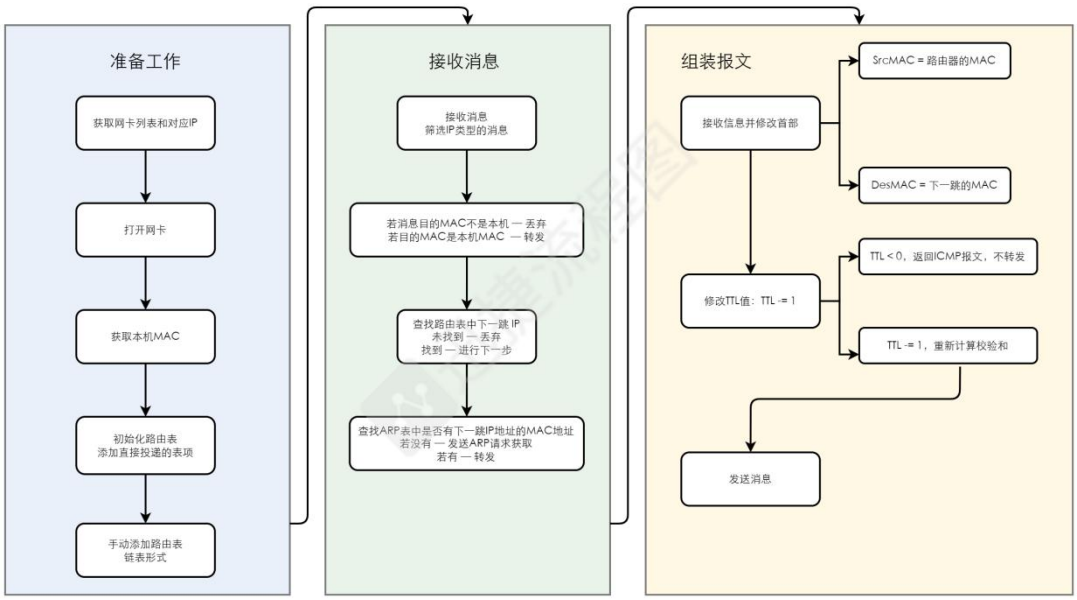

图 2 流程图

第 3 节 实验过程

一、自定义数据结构

本次实验在前期基础上主要添加数据结构有路由表项、路由表、ARP 表项、ARP 表、日志等，其具体关系图如下所示：

图 3 自定义数据结构

二、初步准备

（一）打开网卡获取双 IP

本次实验中对于第一部分进行简化（相较以往实验）；

1. 调用 Npcap 接口通过 find_alldevs()函数获取网络接口设备列表，根据用户输入设备序号信息，遍历列表打印对应的列表信息，并存储其 IP 地址/MAC 地址等相关信息；

```
void find_alldevs()    {
    if (pcap_findalldevs_ex(pcap_src_if_string, NULL, &alldevs, errbuf) == -1){
        ……}
    else{
        int i = 0;
        //  获取该网络接口设备的ip地址信息
        for (d = alldevs; d != NULL; d = d->next){
            if (i == index){
                ……
                for (a = d->addresses; a != nullptr; a = a->next){
                    if (((struct sockaddr_in*)a->addr)->sin_family == AF_INET &&
a->addr){
                        printf("%d ", i);
                        ……
                        //  存储对应IP地址与MAC地址
                        ……}    }
                //  打开该网卡
                ahandle = open(d->name);
            }
            i++;
        }
    }
    pcap_freealldevs(alldevs);
}
```

2. 调用 Npcap 接口，通过 open()函数打开该网卡，而 IP 将用于构造 ARP 请求分组；

```
pcap_t* open(char* name){
    pcap_t* temp = pcap_open(name,65536,PCAP_OPENFLAG_PROMISCUOUS,100,NULL,errbuf);
    if (temp == NULL)
        printf("==========  error  ==========");
    return temp;
```

（二）伪造 ARP 报文获取本机 MAC

1. 构建 ARP 请求包

分别将以太网帧中 APRFrame.FrameHeader.DesMAC 设置为广播地址；APRFrame.FrameHeader.SrcMAC、ARPFrame.SendHa 设置为本机网卡的 MAC 地址；将 ARPFrame.RecvHa 设置为 0，具体如下：

```
ARPFrame_t ARPFrame;
for (int i = 0; i < 6; i++) {
    //  将APRFrame.FrameHeader.DesMAC设置为广播地址
    ARPFrame.FrameHeader.DesMAC[i] = 0xff;
    //  将APRFrame.FrameHeader.SrcMAC设置为本机网卡的MAC地址
    ARPFrame.FrameHeader.SrcMAC[i] = 0x0f;
    //  将ARPFrame.SendHa设置为本机网卡的MAC地址
    ARPFrame.SendHa[i] = 0x0f;
    //  将ARPFrame.RecvHa设置为0
    ARPFrame.RecvHa[i] = 0;
}
```

2. 初始化

将以太网帧设置为 ARP 帧，并对 ARP 帧各参数进行初始化；其中 SendIP 可任意设定，而目的 IP 为本机网卡所绑定的 IP 地址，具体如下：

```
ARPFrame.FrameHeader.FrameType = htons(0x806);    //帧类型为ARP
ARPFrame.HardwareType = htons(0x0001);            //硬件类型为以太网
ARPFrame.ProtocolType = htons(0x0800);            //协议类型为IP
ARPFrame.HLen = 6;                                //硬件地址长度为6
ARPFrame.PLen = 4;                                //协议地址长为4
ARPFrame.Operation = htons(0x0001);               //操作为ARP请求
//  将ARPFrame.SendIP设置为本机网卡上绑定的IP地址
ARPFrame.SendIP = inet_addr("122.122.122.122");
ARPFrame.RecvIP = ip;
```

3. 发送 ARP 请求

通过 pcap_sendpacket()函数发送 ARP 请求，具体如下：

```
/*
adhandle —— 指定通过哪块接口网卡发送数据包，通常是调用 pcap_open()函数成功后返回的值
ARPFrame —— 指向需要发送的数据包，其中包含各层头部信息
size —— 指定大小
        */
if (pcap_sendpacket(ahandle, (u_char*)&ARPFrame, sizeof(ARPFrame_t)) != 0){
    //  发送错误处理
    printf("senderror\n");
}
else{
    ……
}
```

4. 过滤信息

当发送成功后，设置过滤条件对本机网卡收到的数据包进行捕获，确保是本机网卡的响应包；当满足过滤条件时，将响应报文的发送 MAC 拷贝到用于储存本机 MAC 地址的数组中，具体如下：

```
//  发送成功
while (1){
    ......
    if (pcap_next_ex(ahandle, &pkt_header, &pkt_data) == 1){
        ARPFrame_t* IPPacket = (ARPFrame_t*)pkt_data;
        //  输出目的MAC地址
        if (ntohs(IPPacket->FrameHeader.FrameType) == 0x806){
            if (!compare(IPPacket->FrameHeader.SrcMAC, ARPFrame.FrameHeader.SrcMAC) &&
compare(IPPacket->FrameHeader.DesMAC, ARPFrame.FrameHeader.SrcMAC)){
                ltable.write2log_arp(IPPacket);
            //  输出源MAC地址，源MAC地址即为所需MAC地址
                for (int i = 0; i < 6; i++)
                    selfmac[i] = IPPacket->FrameHeader.SrcMAC[i];
                    break;
            }
        }
    }
}
```

（三）初始化路由表项并手动添加路由表

1. 路由表初始化

路由表项均以链表形式储存在路由表当中，因此首先应对链表进行初始化

2. 添加默认路由

- 将路由器的网卡双 IP 对应的三元组（掩码，目的网络，下一跳）添加如路由表；
- 其中路由器 IP 所在网络 —— 可通过本机网卡 IP 与掩码按位与而获得
- 设置默认路由的类型为 0 —— 表示该路由表项不可删除

具体代码如下：

```
//  初始化路由表，添加默认路由
Route_table::Route_table(){
    head = new Route_item;
    tail = new Route_item;
    head->nextitem = tail;
    num = 0;
    for (int i = 0; i < 2; i++){
        Route_item* temp = new Route_item;
        //  本机网卡的ip 和掩码进行按位与即为所在网络
        temp->net = (inet_addr(ip[i])) & (inet_addr(mask[i]));
        temp->mask = inet_addr(mask[i]);
        temp->type = 0;        //0表示直接投递的网络，不可删除
        this->add(temp);  //添加表项
    }
}
```

3. 手动添加路由表项

针对 exe 文件提示，分别输入掩码、目的网络、下一跳地址等；

具体添加路由表项过程在 add 函数中进行处理：

- 默认路由表项 —— 直接添加在路由表链表头部
- 其余路由表项 —— 根据最长匹配原则，在插入时对掩码进行排序
- 添加完成 —— 对各路由表项索引进行重新排序

```
//  添加路由表项
void Route_table::add(Route_item* item){
    Route_item* pointer;
    // 默认路由添加在路由表链表头部
    if (item->type == 0){
        ……
    }
    //其它，按照最长匹配原则
    else{
        for (pointer = head->nextitem; pointer != tail && pointer->nextitem != tail;
pointer = pointer->nextitem){
            if (item->mask < pointer->mask && item->mask >= pointer->nextitem->mask ||
pointer->nextitem == tail)
                break;
        }
        //插入到合适位置
        item->nextitem = pointer->nextitem;
        pointer->nextitem = item;
    }
```

```
    //  更新索引
    Route_item* p = head->nextitem;
    for (int i = 0; p != tail; p = p->nextitem, i++){
        p->index = i;
    }
    num++;
}
```

4. 删除路由表项

- 手动输入想要删除的路由表项序号
- 判断序号的特征值是否为 0（即判断是否为默认路由） —— 若为 0，则不可删除
- 遍历链表找到对应序号的路由表项将其移出链表

具体代码如下：

```
//  删除路由表项
void Route_table::remove(int index){
    for (Route_item* t = head; t->nextitem != tail; t = t->nextitem){
        if (t->nextitem->index == index){
            //  直接投递的路由表项不可删除
            if (t->nextitem->type == 0){
                printf("==========  该项不可删除  ==========\n");
                return;
            }
            else{
                t->nextitem = t->nextitem->nextitem;
                return;
            }
        }
    }
    printf("无该表项\n");
}
```

三、接收消息并准备转发

对报文的捕获与处理分为以下几个过程，其中为使消息转发和路由表添加、删除、打印等操作可以同时进行，使用线程函数对报文进行处理；

（一）接收消息

通过 pcap_next_ex()函数对本机网卡接收到的数据包进行循环捕获；

```
int iprecv(pcap_pkthdr* pkt_header, const u_char* pkt_data){
    int rtn = pcap_next_ex(ahandle, &pkt_header, &pkt_data);
    return rtn;
}
```

（二）捕获报文的处理

1. 对捕获报文 MAC 与 IP 的判断

- 如果捕获报文的目的 MAC 不是本机 MAC —— 丢弃
- 如果捕获报文的目的 MAC 是本机 MAC
  - 在路由表中查找目的 IP 未指向本机 IP —— 递交上层
  - 在路由表中查找目的 IP 指向本机 IP —— 转发

其中涉及到 MAC 地址的比较函数 Compare_MAC()，具体如下：

```
bool Compare_MAC(BYTE a[6], BYTE b[6]){
    for (int i = 0; i < 6; i++){
        if (a[i] != b[i])
            return false;
    }
    return true;
}
```

2. 查找路由表对应的下一跳 IP

把目的 IP 地址和子网掩码做“与”运算，得到网络号，把网络号和路由表上的网络号比较（从上到下），匹配到的就是下一跳地址；

- 未找到下一跳 IP —— 丢弃
- 找到下一跳 IP —— 进行下一步

其中涉及到路由表项的查找函数 lookup(),具体如下：

```
//  查找路由表对应表项 —— 并给出下一跳的ip地址
DWORD Route_table::lookup(DWORD ip){
    Route_item* t = head->nextitem;
    for (; t != tail; t = t->nextitem){
        if ((t->mask & ip) == t->net)
            return t->nextip;
    }
    return -1;
}
```

3. 查找 ARP 表

- 存在下一跳 IP 地址的 MAC 地址
- 不存在下一跳 IP 地址的 MAC 地址 —— 伪造 ARP 获取远程 MAC，发送 ARP 请求获取

其中涉及到对 ARP 表项的查找与添加函数，分别为 insert(),lookup(),具体如下：

```
void Arp_table::insert(DWORD ip, BYTE mac[6]){
    atable[num].ip = ip;
    getothermac(ip, atable[num].mac);
    memcpy(mac, atable[num].mac, 6);
    num++;
}
int Arp_table::lookup(DWORD ip, BYTE mac[6]){
    memset(mac, 0, 6);
    for (int i = 0; i < num; i++){
        if (ip == atable[i].ip){
            memcpy(mac, atable[i].mac, 6);
            return 1;
        }
    }
    return 0;
}
```

四、转发

（一）修改 MAC 地址

- 将源 MAC 地址修改为路由器 MAC 地址
- 将目的 MAC 地址修改为下一跳 MAC 地址

（二）修改 TTL 值

- TTL = TTL-1 —— 若 TTL<0，则返回 ICMP 超时报文，不转发

（三）重新设置校验和

校验和设置与检验的具体代码如下所示：

```
//  设置校验和
void setchecksum(Data_t* temp){
    temp->IPHeader.Checksum = 0;
    unsigned int sum = 0;
    WORD* t = (WORD*)&temp->IPHeader;
    for (int i = 0; i < sizeof(IPHeader_t) / 2; i++){
        sum += t[i];
        while (sum >= 0x10000){
            int s = sum >> 16;
            sum -= 0x10000;
            sum += s;
        }
    }
    temp->IPHeader.Checksum = ~sum;//取反
}
```

```
//  检查校验和
bool Check_checksum(Data_t* temp){
    unsigned int sum = 0;
    WORD* t = (WORD*)&temp->IPHeader;
    for (int i = 0; i < sizeof(IPHeader_t) / 2; i++){
        sum += t[i];
        //包含原有校验和相加
        while (sum >= 0x10000){
            int s = sum >> 16;
            sum -= 0x10000;
            sum += s;
        }
    }
    if (sum == 65535)
        return 1;
    return 0;
}
```

转发过程的具体代码如下所示：

```
//  数据报转发
void resend(ICMP_t data, BYTE dmac[]){
    Data_t* temp = (Data_t*)&data;
    memcpy(temp->FrameHeader.SrcMAC, temp->FrameHeader.DesMAC, 6);//源MAC为本机MAC
    memcpy(temp->FrameHeader.DesMAC, dmac, 6);                    //目的MAC为下一跳MAC
    temp->IPHeader.TTL -= 1;                //TTL-1
    if (temp->IPHeader.TTL < 0)
        return;//丢弃
    setchecksum(temp);                                            //重新设置校验和
    int rtn = pcap_sendpacket(ahandle, (const u_char*)temp, 74);  //发送数据报
    if (rtn == 0)
        ltable.write2log_ip("[forward IP]", temp);                //写入日志
}
```

五、日志记录

路由器日志结构体具体如图 所示，其中日志记录大致过程分为打开文件、写入文件、文件关闭等；

接下来将通过日志打印 Print_file()函数，进行简单介绍：

```
void Log_file::Print_file(){
    int i;
    if (num > 50)
        i = (num + 1) % 50;
```

```
    else i = 0;
    for (; i < num % 50; i++){
        ……
        if (!strcmp(diary[i].type, "ARP")){
            in_addr addr;
            addr.s_addr = diary[i].arp.ip;
            char* pchar = inet_ntoa(addr);
            printf("%s\t", pchar);
            for (int i = 0; i < 5; i++){
                printf("%02X.", diary[i].arp.mac[i]);
            }
            printf("%02X\n", diary[i].arp.mac[5]);
        }
        else if (!strcmp(diary[i].type, "IP")){
            in_addr addr;
            addr.s_addr = diary[i].ip.sip;
            char* pchar = inet_ntoa(addr);
            ……
            for (int i = 0; i < 5; i++){
                printf("%02X.", diary[i].ip.smac[i]);
            }
            printf("%02X\t", diary[i].ip.smac[5]);
            printf("DesMAC: ");
            for (int i = 0; i < 5; i++){
                printf("%02X.", diary[i].ip.dmac[i]);
            }
            printf("%02X\n", diary[i].ip.dmac[5]);
        }
    }
}
```

六、实验结果

- 将路由器程序添加到虚拟机 2 和 3 中，运行 exe 文件可以查看到两个虚拟机的双 IP、MAC 地址和默认路由表，如下图所示：

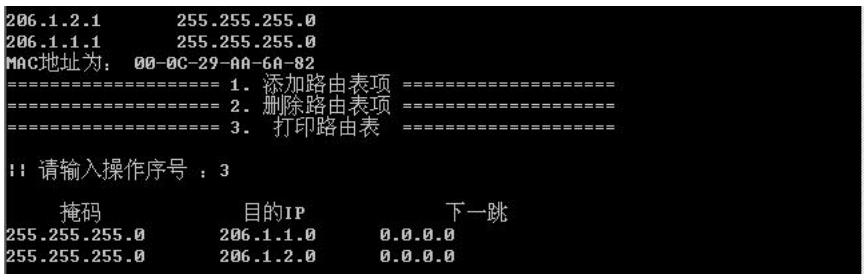

图 4 路由器 2 初始化

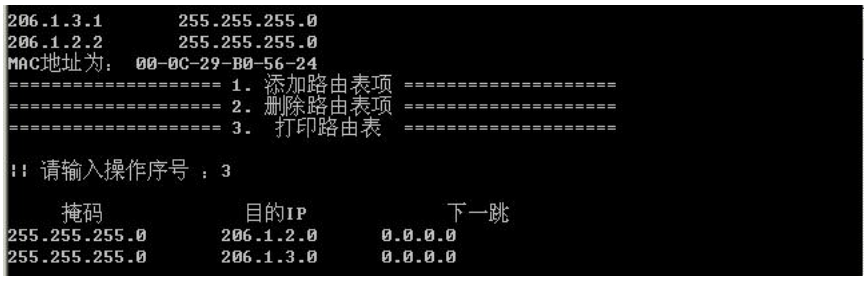

图 5 路由器 3 初始化

- 分别在两个路由器中手动添加路由表项，具体如下：

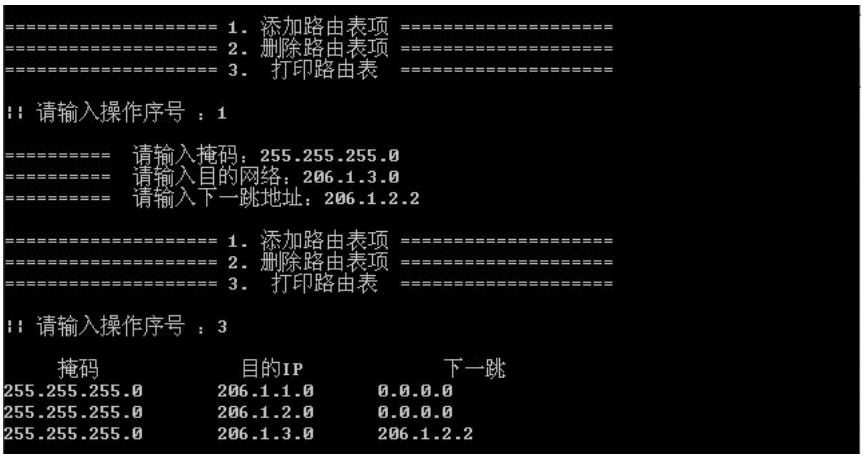

图 6 路由器 2 手动添加路由表项

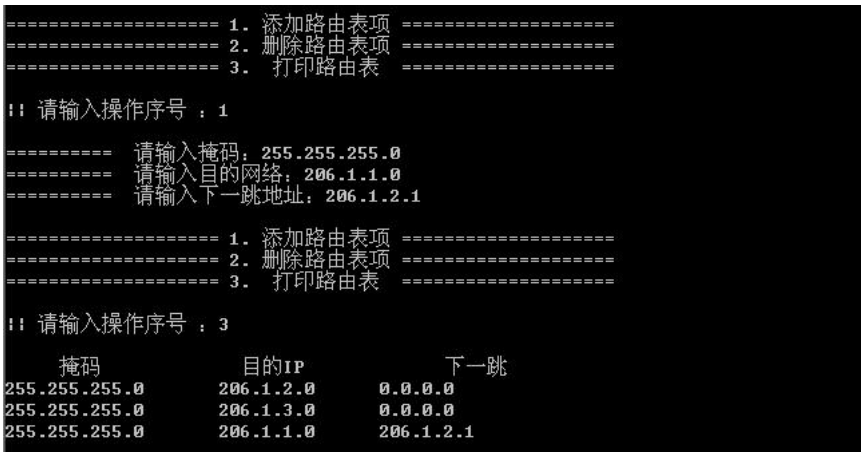

图 7 路由器 3 手动添加路由表项

- 使用虚拟机 1 ping 虚拟机 4，观察是否可以 ping 通，如下图所示：

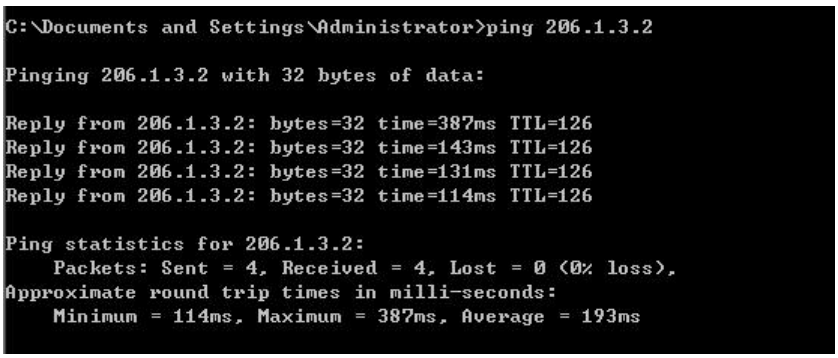

图 8 查看连通性

- 检验路由表的删除功能，具体如下：

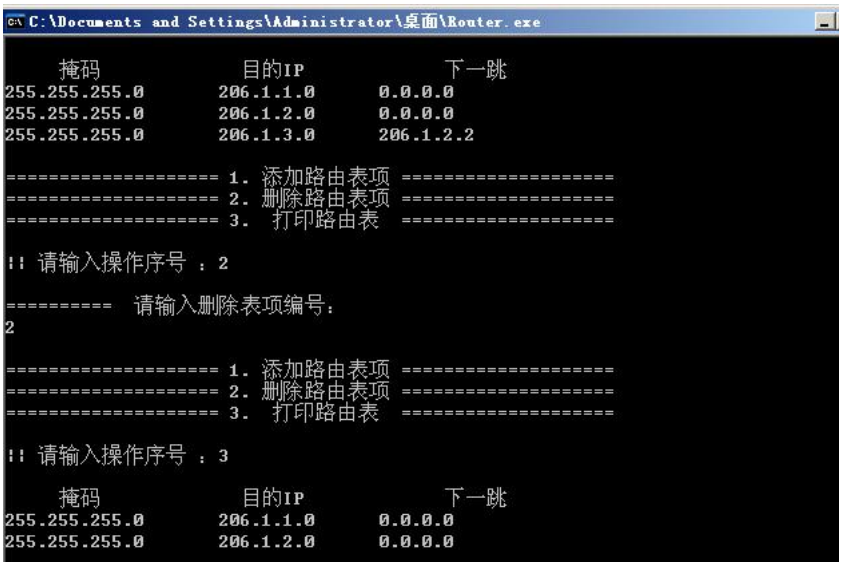

图 9 删除路由表项

- 使用 wireshark 进行抓包，具体如下图所示：

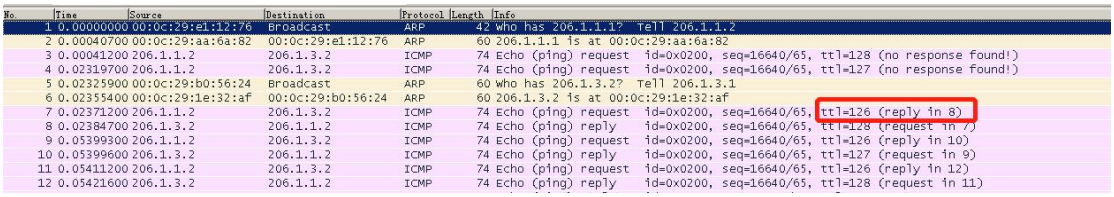

图 10 wireshark 抓包信息

- 观察记录日志，主要有 ARP 广播信息、接收 IP 信息、转发 IP 信息等内容；

Log_File.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
[ ARP ] DesIP：  206.1.2.1  DesMAC：  00-0C-29-AA-6A-82
[receive IP]  SrcIP：  206.1.1.2  DesIP：  206.1.3.2  SrcMAC：  00-0C-29-E1-12-76  DesMAC：  00-0C-29-AA-6A-82
[ ARP ] DesIP：  206.1.2.2  DesMAC：  00-0C-29-B0-56-24
[forward IP]  SrcIP：  206.1.1.2  DesIP：  206.1.3.2  SrcMAC：  00-0C-29-AA-6A-82  DesMAC：  00-0C-29-B0-56-24
[receive IP]  SrcIP：  206.1.3.2  DesIP：  206.1.1.2  SrcMAC：  00-0C-29-B0-56-24  DesMAC：  00-0C-29-AA-6A-82S
[ ARP ] DesIP：  206.1.1.2  DesMAC：  00-0C-29-E1-12-76
[forward IP]  SrcIP：  206.1.3.2  DesIP：  206.1.1.2  SrcMAC：  00-0C-29-AA-6A-82  DesMAC：  00-0C-29-E1-12-76
[receive IP]  SrcIP：  206.1.1.2  DesIP：  206.1.3.2  SrcMAC：  00-0C-29-E1-12-76  DesMAC：  00-0C-29-AA-6A-82
[forward IP]  SrcIP：  206.1.1.2  DesIP：  206.1.3.2  SrcMAC：  00-0C-29-AA-6A-82  DesMAC：  00-0C-29-B0-56-24
[receive IP]  SrcIP：  206.1.3.2  DesIP：  206.1.1.2  SrcMAC：  00-0C-29-B0-56-24  DesMAC：  00-0C-29-AA-6A-82
[forward IP]  SrcIP：  206.1.3.2  DesIP：  206.1.1.2  SrcMAC：  00-0C-29-AA-6A-82  DesMAC：  00-0C-29-E1-12-76
[receive IP]  SrcIP：  206.1.3.2  DesIP：  206.1.1.2  SrcMAC：  00-0C-29-B0-56-24  DesMAC：  00-0C-29-AA-6A-82
[forward IP]  SrcIP：  206.1.3.2  DesIP：  206.1.1.2  SrcMAC：  00-0C-29-AA-6A-82  DesMAC：  00-0C-29-E1-12-76
[receive IP]  SrcIP：  206.1.1.2  DesIP：  206.1.3.2  SrcMAC：  00-0C-29-E1-12-76  DesMAC：  00-0C-29-AA-6A-82
[forward IP]  SrcIP：  206.1.1.2  DesIP：  206.1.3.2  SrcMAC：  00-0C-29-AA-6A-82  DesMAC：  00-0C-29-B0-56-24
[receive IP]  SrcIP：  206.1.3.2  DesIP：  206.1.1.2  SrcMAC：  00-0C-29-B0-56-24  DesMAC：  00-0C-29-AA-6A-82
[forward IP]  SrcIP：  206.1.3.2  DesIP：  206.1.1.2  SrcMAC：  00-0C-29-AA-6A-82  DesMAC：  00-0C-29-E1-12-76
[receive IP]  SrcIP：  206.1.3.2  DesIP：  206.1.1.2  SrcMAC：  00-0C-29-B0-56-24  DesMAC：  00-0C-29-AA-6A-82
[forward IP]  SrcIP：  206.1.3.2  DesIP：  206.1.1.2  SrcMAC：  00-0C-29-AA-6A-82  DesMAC：  00-0C-29-E1-12-76
[receive IP]  SrcIP：  206.1.1.2  DesIP：  206.1.3.2  SrcMAC：  00-0C-29-E1-12-76  DesMAC：  00-0C-29-AA-6A-82
[forward IP]  SrcIP：  206.1.1.2  DesIP：  206.1.3.2  SrcMAC：  00-0C-29-AA-6A-82  DesMAC：  00-0C-29-B0-56-24
[receive IP]  SrcIP：  206.1.3.2  DesIP：  206.1.1.2  SrcMAC：  00-0C-29-B0-56-24  DesMAC：  00-0C-29-AA-6A-82
[forward IP]  SrcIP：  206.1.3.2  DesIP：  206.1.1.2  SrcMAC：  00-0C-29-AA-6A-82  DesMAC：  00-0C-29-E1-12-76
[receive IP]  SrcIP：  206.1.3.2  DesIP：  206.1.1.2  SrcMAC：  00-0C-29-B0-56-24  DesMAC：  00-0C-29-AA-6A-82
[forward IP]  SrcIP：  206.1.3.2  DesIP：  206.1.1.2  SrcMAC：  00-0C-29-AA-6A-82  DesMAC：  00-0C-29-E1-12-76
[ ARP ] DesIP：  206.1.2.1  DesMAC：  00-0C-29-AA-6A-82
[receive IP]  SrcIP：  206.1.1.2  DesIP：  206.1.3.2  SrcMAC：  00-0C-29-E1-12-76  DesMAC：  00-0C-29-AA-6A-82
[ ARP ] DesIP：  206.1.2.2  DesMAC：  00-0C-29-B0-56-24
[forward IP]  SrcIP：  206.1.1.2  DesIP：  206.1.3.2  SrcMAC：  00-0C-29-AA-6A-82  DesMAC：  00-0C-29-B0-56-24
[receive IP]  SrcIP：  206.1.3.2  DesIP：  206.1.1.2  SrcMAC：  00-0C-29-B0-56-24  DesMAC：  00-0C-29-AA-6A-82
[ ARP ] DesIP：  206.1.1.2  DesMAC：  00-0C-29-E1-12-76
[forward IP]  SrcIP：  206.1.3.2  DesIP：  206.1.1.2  SrcMAC：  00-0C-29-AA-6A-82  DesMAC：  00-0C-29-E1-12-76
[receive IP]  SrcIP：  206.1.3.2  DesIP：  206.1.1.2  SrcMAC：  00-0C-29-B0-56-24  DesMAC：  00-0C-29-AA-6A-82
[forward IP]  SrcIP：  206.1.3.2  DesIP：  206.1.1.2  SrcMAC：  00-0C-29-AA-6A-82  DesMAC：  00-0C-29-E1-12-76
[receive IP]  SrcIP：  206.1.3.2  DesIP：  206.1.1.2  SrcMAC：  00-0C-29-B0-56-24  DesMAC：  00-0C-29-AA-6A-82
[forward IP]  SrcIP：  206.1.3.2  DesIP：  206.1.1.2  SrcMAC：  00-0C-29-AA-6A-82  DesMAC：  00-0C-29-E1-12-76
[receive IP]  SrcIP：  206.1.3.2  DesIP：  206.1.1.2  SrcMAC：  00-0C-29-B0-56-24  DesMAC：  00-0C-29-AA-6A-82
[forward IP]  SrcIP：  206.1.3.2  DesIP：  206.1.1.2  SrcMAC：  00-0C-29-AA-6A-82  DesMAC：  00-0C-29-E1-12-76
[receive IP]  SrcIP：  206.1.3.2  DesIP：  206.1.1.2  SrcMAC：  00-0C-29-B0-56-24  DesMAC：  00-0C-29-AA-6A-82
[forward IP]  SrcIP：  206.1.3.2  DesIP：  206.1.1.2  SrcMAC：  00-0C-29-AA-6A-82  DesMAC：  00-0C-29-E1-12-76
[receive IP]  SrcIP：  206.1.1.2  DesIP：  206.1.3.2  SrcMAC：  00-0C-29-E1-12-76  DesMAC：  00-0C-29-AA-6A-82
[forward IP]  SrcIP：  206.1.1.2  DesIP：  206.1.3.2  SrcMAC：  00-0C-29-AA-6A-82  DesMAC：  00-0C-29-B0-56-24
[receive IP]  SrcIP：  206.1.3.2  DesIP：  206.1.1.2  SrcMAC：  00-0C-29-B0-56-24  DesMAC：  00-0C-29-AA-6A-82
[forward IP]  SrcIP：  206.1.3.2  DesIP：  206.1.1.2  SrcMAC：  00-0C-29-AA-6A-82  DesMAC：  00-0C-29-E1-12-76
[receive IP]  SrcIP：  206.1.3.2  DesIP：  206.1.1.2  SrcMAC：  00-0C-29-B0-56-24  DesMAC：  00-0C-29-AA-6A-82
```

图 11 日志信息