南 开 大 学

计算机学院

网络技术与应用课程报告

期末大作业实验报告

学号：1911406
姓名：丁彦添
年级：2019 级
专业：计科

2021 年 12 月 23 日

第1节　实验内容说明

简单路由器程序设计实验的具体要求为：

(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

结合课堂知识，了解利用 NPcap 开发包实现简单路由程序，该路由程序应该包括以下功能：

(1) IP 数据包捕获和转发；

(2) ARP 请求与解析；

(3) 重新计算 IP 数据包的头部校验和；

(4) 处理 IP 数据包的头部校验和；处理 IP 数据包的 TTL 值；

(5) 静态路由表维护。

IP 互联网采用表驱动的路由选择算法

1. 需要路由选择的设备保存一张 IP 路由表

2. 路由表存储有关目的地址及怎样到达目的地址的信息

3. 通过查询路由表决定把数据报发往何处

相关的 ICMP 报文

当路由器不能 为数据包找到路由器或主机交付数据包时 ，就丢弃该数据包 ，然后向源主机发出 ICMP 目的不可达报文。

超时报文

路由器在转发数据包时，如果生存周期 TTL 值减 1 后为 0，就丢弃这个数据包。当丢弃这个数据包时，路由器向源主机发出 ICMP 超时报文。

当计时器超时，而目的主机还没有接收到一个数据包时，它就会将数据包丢弃并向源主机发出 ICMP 超时报文。

第3节　实验过程

项目设计思路和关键代码分析

使用 VSCode 创建一个 MFC 工程，在 VC++目录、链接器、编译选项等地方设置相应 npcap 目录，添加相应的选项。

采用线性表构建路由表

初始化过程：

获得本机的设备列表→根据设备列表获取各接口及各接口的所有 IP 信息→打开每个设备接口→获取各接口的 MAC 地址→初始化路由表→设置过滤规则，仅接收 ARP 响应帧和需要路由的帧→释放设备列表→开启数据包处理线程

获得本机设备列表

```
    if (pcap_findalldevs_ex(PCAP_SRC_IF_STRING, NULL, &m_alldevs, errbuf) == -
1)
    {
        sprintf_s(strbuf, "pcap_findalldevs_ex 错误: %s", errbuf);
        // 如果错误，返回错误信息
        PostMessage(WM_QUIT, 0, 0);
    }
```

获得接口信息和 IP 地址信息

```
    // 获取IP地址信息
    for(d = alldevs; d != NULL; d = d->next)
    {
        if(d->addresses != NULL)// 排除集成modem的影响（没有IP地址）
        {
            // 得到一个有效的接口和其IP地址列表
            IfInfo[i].DeviceName = d->name;
            IfInfo[i].Description = d->description;
            for(a = d->addresses; a; a = a->next)
            {
                if (a->addr->sa_family == AF_INET)
                {
                    chk[i]=1;
                    ipaddr.IPAddr = (((struct sockaddr_in
*)a->addr)->sin_addr.s_addr);
                    ipaddr.IPMask = (((struct sockaddr_in
*)a->netmask)->sin_addr.s_addr);
                    IfInfo[i].ip.Add(ipaddr);
                    j++;
                }
            }
            if (i==MAX_IF)  // 最多处理MAX_IF个接口
                break;
            else i++;
        }
    }
```

初始化路由表

```
    RouteTable_t rt;
    for (i = 0; i < IfCount; i++)
    {

        if(chk[i]){
        for (j = 0; j < IfInfo[i].ip.GetSize(); j++)
        {
```

```
            rt.IfNo = i;
            rt.DstIP = IfInfo[i].ip[j].IPAddr  & IfInfo[i].ip[j].IPMask;
            rt.Mask  = IfInfo[i].ip[j].IPMask;
            rt.NextHop = 0; // 直接投递
            RouteTable.AddTail(rt);
            m_RouteTable.InsertString(-1, IPntoa(rt.Mask) + " -- " +
IPntoa(rt.DstIP) + " -- " + IPntoa(rt.NextHop) + "  (直接投递)");
        }}
    }
```

设置过滤规则，仅接收 ARP 和需要路由的数据帧

```
    CString Filter, Filter0, Filter1;
    Filter0 = "(";
    Filter1 = "(";
    for (i = 0; i < IfCount; i++)
    {
        Filter0 += "(ether dst " + MACntoa(IfInfo[i].MACAddr) + ")";
        for (j = 0; j < IfInfo[i].ip.GetSize(); j++)
        {
            Filter1 += "(ip dst host " + IPntoa(IfInfo[i].ip[j].IPAddr) + ")";
            if (((j == (IfInfo[i].ip.GetSize() -1))) && (i == (IfCount-1)))
                Filter1 += ")";
            else Filter1 += " or ";
        }
        if (i == (IfCount-1)) Filter0 += ")";
        else Filter0 += " or ";


    }
    Filter = Filter0 + " and ((arp and (ether[21]=0x2)) or (not" + Filter1 +
"))";
```

之后就是对路由表项的处理

添加路由表

```
// 添加路由表项
void CmeinrouterDlg::OnAddRouterButton()
{
    bool flag;
    int i, j;
    DWORD ipaddr;
    RouteTable_t rt;
    m_NextHop.GetAddress(ipaddr);
    ipaddr = htonl(ipaddr);
    // 检查合法性
```

```
    flag = false;
    for (i=0; i < IfCount; i++)
    {
        for (j = 0; j < IfInfo[i].ip.GetSize(); j++)
        {
            if (((IfInfo[i].ip[j].IPAddr) & (IfInfo[i].ip[j].IPMask)) ==
((IfInfo[i].ip[j].IPMask) & ipaddr))
            {
                rt.IfNo = i;
                // 记录子网掩码
                m_Mask.GetAddress(ipaddr);
                rt.Mask = htonl(ipaddr);
                // 记录目的IP
                m_Destination.GetAddress(ipaddr);
                rt.DstIP = htonl(ipaddr);
                // 记录下一跳
                m_NextHop.GetAddress(ipaddr);
                rt.NextHop = htonl(ipaddr);
                // 把该条路由表项添加到路由表
                RouteTable.AddTail(rt);

                // 在路由表窗口中显示该路由表项
                m_RouteTable.InsertString(-1, IPntoa(rt.Mask) + " -- "
                    + IPntoa(rt.DstIP) + " -- " + IPntoa(rt.NextHop));
                flag = true;
            }
        }
    }
}
```

删除路由表

```
// 删除路由表项
void CRouterDlg::OnBnClickedDel()
{
    //……
    // 取得子网掩码选项
    strncat_s(ipaddr, str, 15);
    mask = inet_addr(ipaddr);
    // 取得目的地址选项
    ipaddr[0] = 0;
    strncat_s(ipaddr, &str[19], 15);
    destination = inet_addr(ipaddr);
    // 取得下一跳选项
    ipaddr[0] = 0;
```

```
    strncat_s(ipaddr, &str[38], 15);
    nexthop = inet_addr(ipaddr);
    if (nexthop == 0)
    {
        MessageBox(L"直连路由，不允许删除！");
        return;
    }
    // 路由表中没有需要处理的内容，则返回
    if (RouteTable.IsEmpty())
    {
        MessageBox(L"路由表为空！");
        return;
    }
    // 遍历路由表,把需要删除的路由表项从路由表中删除
    pos = RouteTable.GetHeadPosition();//pos=0
    int ii = 0;
    while (true) {
        rt = RouteTable.GetAt(pos);
        if (ii = i || (rt.Mask == mask) && (rt.DstIP == destination) &&
(rt.NextHop == nexthop)) {
            RouteTable.RemoveAt(pos);
            m_RouteTable.DeleteString(i);// 把该路由表项从路由表窗口中删除
            MessageBox(L"成功删除路由表项！");
            return;
        }
        RouteTable.GetNext(pos);
    }
    MessageBox(L"未能成功删除路由表项！");
}
```

数据包捕获与处理

捕获流经本路由器的数据包并按照路由协议进行处理。

数据包捕获

```
    // 开始正式接收并处理帧
    while (true)
    {
        res = pcap_next_ex( pIfInfo->adhandle, &header, &pkt_data);
        if (res == 1)
        {
            FrameHeader_t   *fh;
            fh = (FrameHeader_t *) pkt_data;
            switch (ntohs(fh->FrameType))
```

```
        {
            case 0x0806:

                ARPFrame_t *ARPf;
                ARPf = (ARPFrame_t *)pkt_data;
                // ARP包，转到ARP包处理函数
                ARPPacketProc(header, pkt_data);
                break;

            case 0x0800:

                IPFrame_t *IPf;
                IPf = (IPFrame_t*) pkt_data;
                // IP包，转到IP包处理函数
                IPPacketProc(pIfInfo, header, pkt_data);
                break;
            default:
                break;
        }
    }
```

处理 IP 数据包

捕获到 IP 数据包，将其 TTL-1，看是否小于等于 0，如果是则向源主机发送 ICMP 超时报文，如果不是，查询路由表。如果找不到路由表对应项，则发送 ICMP 目的不可达报文。如果找到了，先计算校验和，再查询 IP-MAC 表，如果找到了就直接转发，如果找不到，就看缓存队列是否已满，如果满了，则丢弃报文；如果没满，将报文存入缓存队列并设置定时器，向下一跳发送 ARP 查询 IP-MAC 对应关系。

```
    pDlg->Logger.InsertString(-1, ("收到IP数据包:" + IPntoa(IPf->IPHeader.SrcIP)
+ "->"
        + IPntoa(IPf->IPHeader.DstIP)));

    // ICMP超时
    if (IPf->IPHeader.TTL <= 0)
    {
        ICMPPacketProc(pIfInfo, 11, 0, pkt_data);
```

```
            return;
        }

        IPHeader_t *IpHeader = &(IPf->IPHeader);
        // ICMP差错
        if (IsChecksumRight((char *)IpHeader) == 0)
        {
            // 日志输出信息
            pDlg->Logger.InsertString(-1, "   IP数据包校验和错误，丢弃数据包");
            return;
        }

        // 路由查询
        DWORD nextHop;       // 经过路由选择算法得到的下一站目的IP地址
        UINT ifNo;           // 下一跳的接口序号
        // 路由查询
        if((nextHop = RouteLookup(ifNo, IPf->IPHeader.DstIP, &RouteTable)) == -1)
        {
            // ICMP目的不可达
            ICMPPacketProc(pIfInfo, 3, 0, pkt_data);
            return;
        }
        else
        {
            sPacket.IfNo = ifNo;
            sPacket.TargetIP = nextHop;

            cpyMAC(IPf->FrameHeader.SrcMAC, IfInfo[sPacket.IfNo].MACAddr);

            // TTL减1
            IPf->IPHeader.TTL -= 1;

            unsigned short check_buff[sizeof(IPHeader_t)];
            // 设IP头中的校验和为0
            IPf->IPHeader.Checksum = 0;

            memset(check_buff, 0, sizeof(IPHeader_t));
            IPHeader_t * ip_header = &(IPf->IPHeader);
            memcpy(check_buff, ip_header, sizeof(IPHeader_t));

            // 计算IP头部校验和
            IPf->IPHeader.Checksum = ChecksumCompute(check_buff,
    sizeof(IPHeader_t));
```

```
        // IP-MAC地址映射表中存在该映射关系
        if (IPLookup(sPacket.TargetIP, IPf->FrameHeader.DesMAC))
        {
            memcpy(sPacket.PktData, pkt_data, header->len);
            sPacket.len = header->len;
            if(pcap_sendpacket(IfInfo[sPacket.IfNo].adhandle, (u_char *)
sPacket.PktData, sPacket.len) != 0)
            {
                // 错误处理
                AfxMessageBox("发送IP数据包时出错!");
                return;
            }

            // 日志输出信息
            pDlg->Logger.InsertString(-1,"   转发IP数据包：");
            pDlg->Logger.InsertString(-1,("   "+IPntoa(IPf->IPHeader.SrcIP) +
"->"
                + IPntoa(IPf->IPHeader.DstIP) + "  " +
MACntoa(IPf->FrameHeader.SrcMAC )
                + "->" + MACntoa(IPf->FrameHeader.DesMAC)));
        }
        // IP-MAC地址映射表中不存在该映射关系
        else
        {
            if (SP.GetCount() < 65530)      // 存入缓存队列
            {
                sPacket.len = header->len;
                // 将需要转发的数据报存入缓存区
                memcpy(sPacket.PktData, pkt_data, header->len);

                // 在某一时刻只允许一个线程维护链表
                mMutex.Lock(INFINITE);

                sPacket.n_mTimer = TimerCount;
                if (TimerCount++ > 65533)
                {
                    TimerCount = 1;
                }
                pDlg->SetTimer(sPacket.n_mTimer, 10000, NULL);
                SP.AddTail(sPacket);

                mMutex.Unlock();

                // 日志输出信息
```

```
                pDlg->Logger.InsertString(-1, "   缺少目的MAC地址，将IP数据包存入
转发缓冲区");
                pDlg->Logger.InsertString(-1, ("   存入转发缓冲区的数据包为：
"+IPntoa(IPf->IPHeader.SrcIP)
                    + "->" + IPntoa(IPf->IPHeader.DstIP) + "  " +
MACntoa(IPf->FrameHeader.SrcMAC)
                    + "->xx:xx:xx:xx:xx:xx"));
                pDlg->Logger.InsertString(-1, "   发送ARP请求");

                // 发送ARP请求
                ARPRequest(IfInfo[sPacket.IfNo].adhandle,
IfInfo[sPacket.IfNo].MACAddr,
                    IfInfo[sPacket.IfNo].ip[1].IPAddr, sPacket.TargetIP);
            }
            else// 如缓存队列太长，抛弃该报
            {
                // 日志输出信息
                pDlg->Logger.InsertString(-1, "   转发缓冲区溢出，丢弃IP数据包");
                pDlg->Logger.InsertString(-1, ("   丢弃的IP数据包为：" +
IPntoa(IPf->IPHeader.SrcIP) + "->"
                    + IPntoa(IPf->IPHeader.DstIP) + "  " +
MACntoa(IPf->FrameHeader.SrcMAC)
                    + "->xx:xx:xx:xx:xx:xx"));
            }
        }
    }
```

处理 ARP 数据包

先看是否是 ARP 应答，如果不是的话直接返回。如果是 ARP 应答，看该 ARP 包含的 IP-MAC 映射关系是否已经存了，如果是，直接返回；如果不是，将该映射关系存入 IP-MAC 表中，该表用 List 存储。然后看报文缓存队列中是否有目的 MAC 地址是这个 MAC 地址的 IP 数据包，如果有的话，发送这些 IP 数据包。

```
    ARPf = *(ARPFrame_t *)pkt_data;

    if (ARPf.Operation == ntohs(0x0002))
    {
        pDlg->Logger.InsertString(-1, "收到ARP响应包");
        pDlg->Logger.InsertString(-1, ("   ARP "+(IPntoa(ARPf.SendIP))+" -- "
```

```
                +MACntoa(ARPf.SendHa)));
        // IP－MAC地址映射表中已经存在该对应关系
        if (IPLookup(ARPf.SendIP, macAddr))
        {
            pDlg->Logger.InsertString(-1, "    该对应关系已经存在于IP－MAC地址映射
表中");
            return;
        }
        else
        {
            ip_mac.IPAddr = ARPf.SendIP;
            memcpy(ip_mac.MACAddr, ARPf.SendHa, 6);
            // 将IP-MAC映射关系存入表中
            IP_MAC.AddHead(ip_mac);

            // 日志输出信息
            pDlg->Logger.InsertString(-1, "    将该对应关系存入IP－MAC地址映射表中
");
        }

        mMutex.Lock(INFINITE);
        //转发IP数据包的处理
        。。。。。。
                    // 日志输出信息
                    pDlg->Logger.InsertString(-1, "    转发缓存区中目的地址是该MAC
地址的IP数据包");
                    pDlg->Logger.InsertString(-1, ("      发送IP数据包：
"+IPntoa(IPf->IPHeader.SrcIP) + "->"
                        + IPntoa(IPf->IPHeader.DstIP) + "  " +
MACntoa(IPf->FrameHeader.SrcMAC )
                        +"->"+MACntoa(IPf->FrameHeader.DesMAC)));

                    flag = true;
                    break;
                }
            }
        } while(flag);

        mMutex.Unlock();
    }
```

IP 数据包的发送

遍历转发缓存区，发送 IP 数据包

```
        do{
            // 查看是否能转发缓存中的IP数据报
            flag = false;

            // 没有需要处理的容
            if (SP.IsEmpty())
            {
                break;
            }

            // 遍历转发缓存区
            pos = SP.GetHeadPosition();
            for (int i=0; i < SP.GetCount(); i++)
            {
                CurrentPos = pos;
                sPacket = SP.GetNext(pos);

                if (sPacket.TargetIP == ARPf.SendIP)
                {
                    IPf = (IPFrame_t *) sPacket.PktData;
                    cpyMAC(IPf->FrameHeader.DesMAC, ARPf.SendHa);

                    for(int t=0; t<6; t++)
                    {
                        IPf->FrameHeader.SrcMAC[t] =
IfInfo[sPacket.IfNo].MACAddr[t];
                    }
                    // 发送IP数据包
                    pcap_sendpacket(IfInfo[sPacket.IfNo].adhandle, (u_char *)
sPacket.PktData, sPacket.len);

                    SP.RemoveAt(CurrentPos);
```

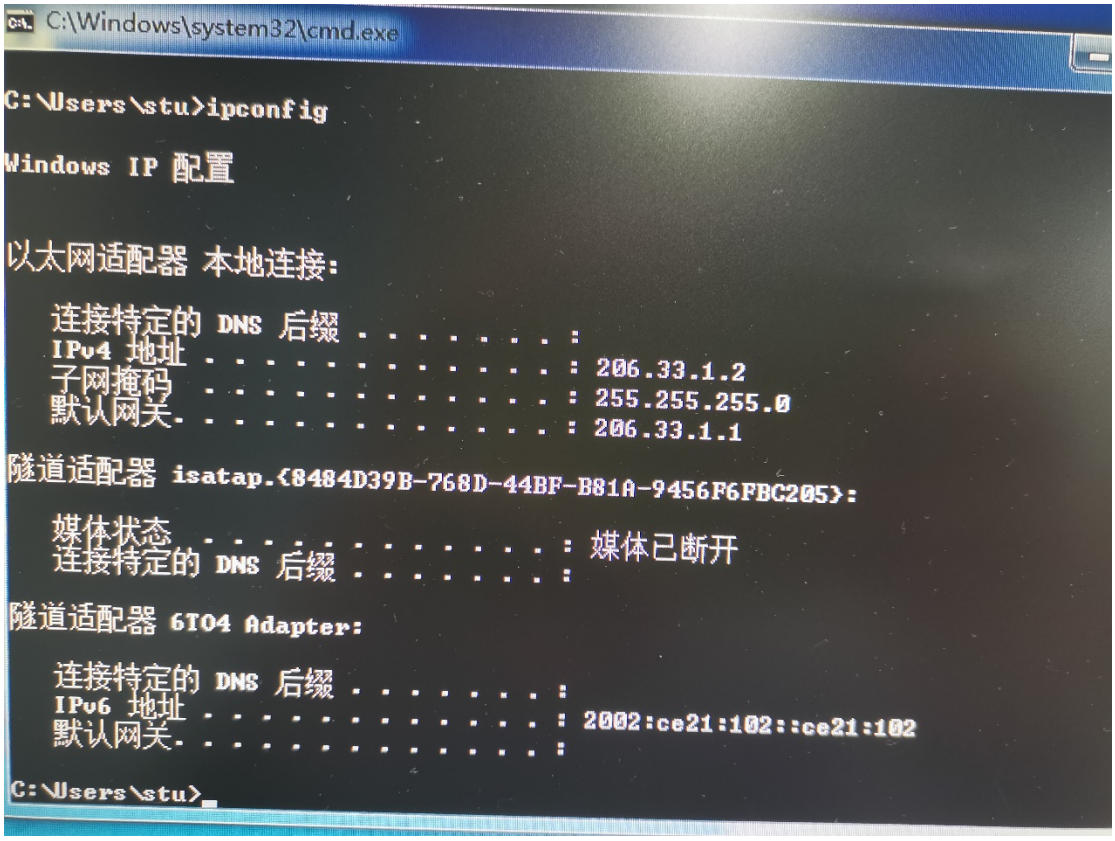
第4节　实验结果

实验测试配置：

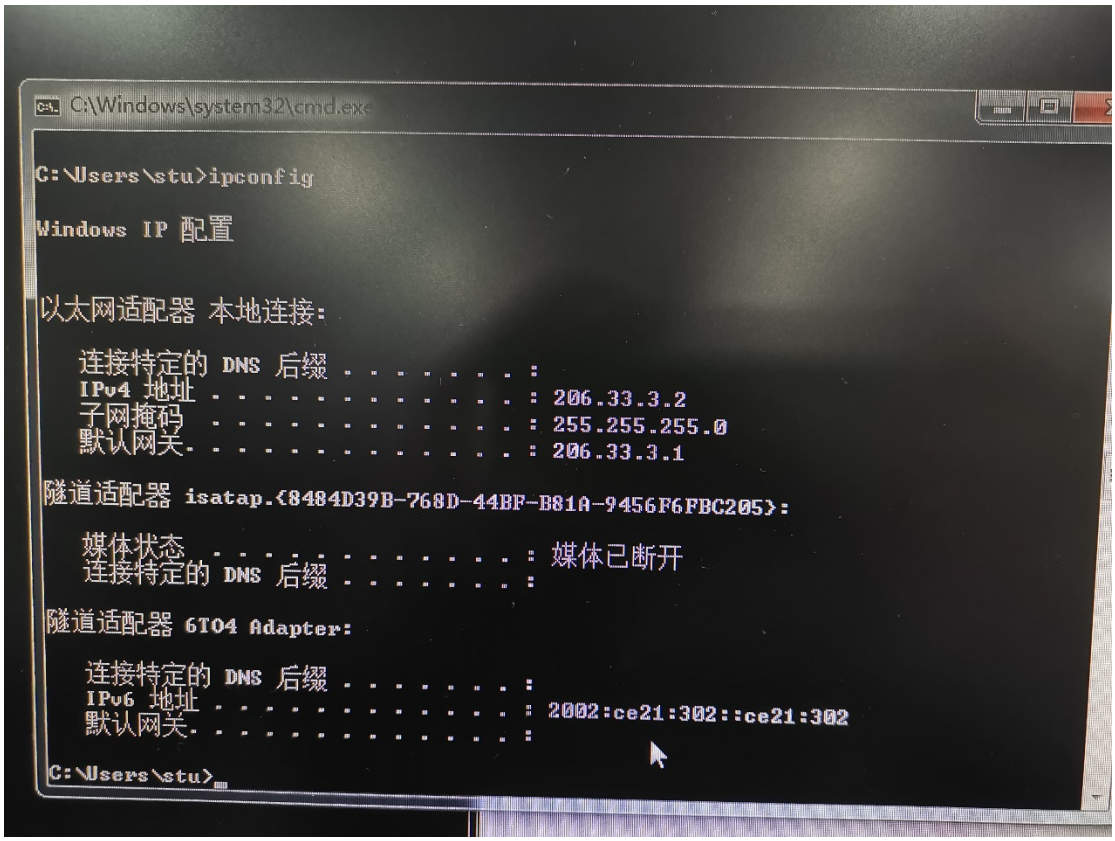
使用四台计算机，分别作为主机一、主机二、路由器一、路由器二。

具体 IP 和网关配置如下图：

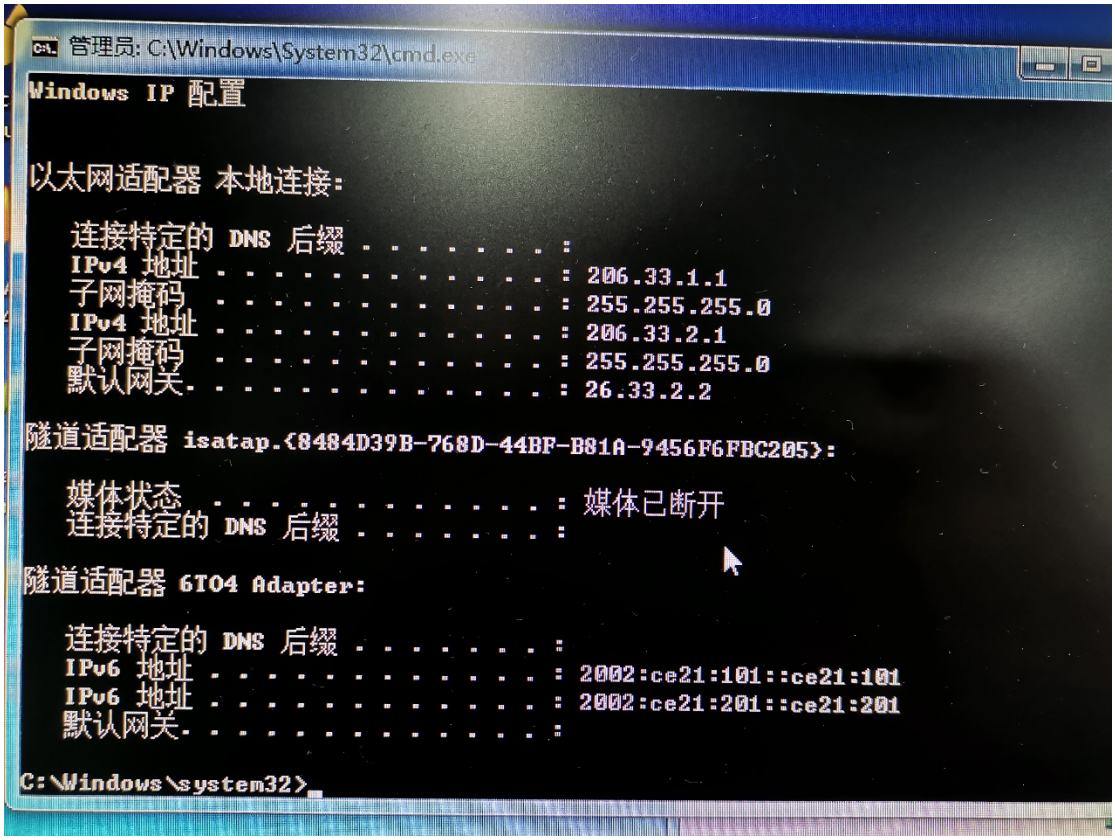
主机一 IP 网关设置


C:\Windows\system32\cmd.exe
C:\Users\stu>ipconfig
Windows IP 配置
以太网适配器 本地连接:
连接特定的 DNS 后缀 . . . . . :
IPv4 地址 . . . . . : 206.33.1.2
子网掩码 . . . . . : 255.255.255.0
默认网关. . . . . : 206.33.1.1
隧道适配器 isatap.{8484D39B-768D-44BF-B81A-9456F6FBC205}:
媒体状态 . . . . . : 媒体已断开
连接特定的 DNS 后缀 . . . . . :
隧道适配器 6TO4 Adapter:
连接特定的 DNS 后缀 . . . . . :
IPv6 地址 . . . . . : 2002:ce21:102::ce21:102
默认网关. . . . . :
C:\Users\stu>

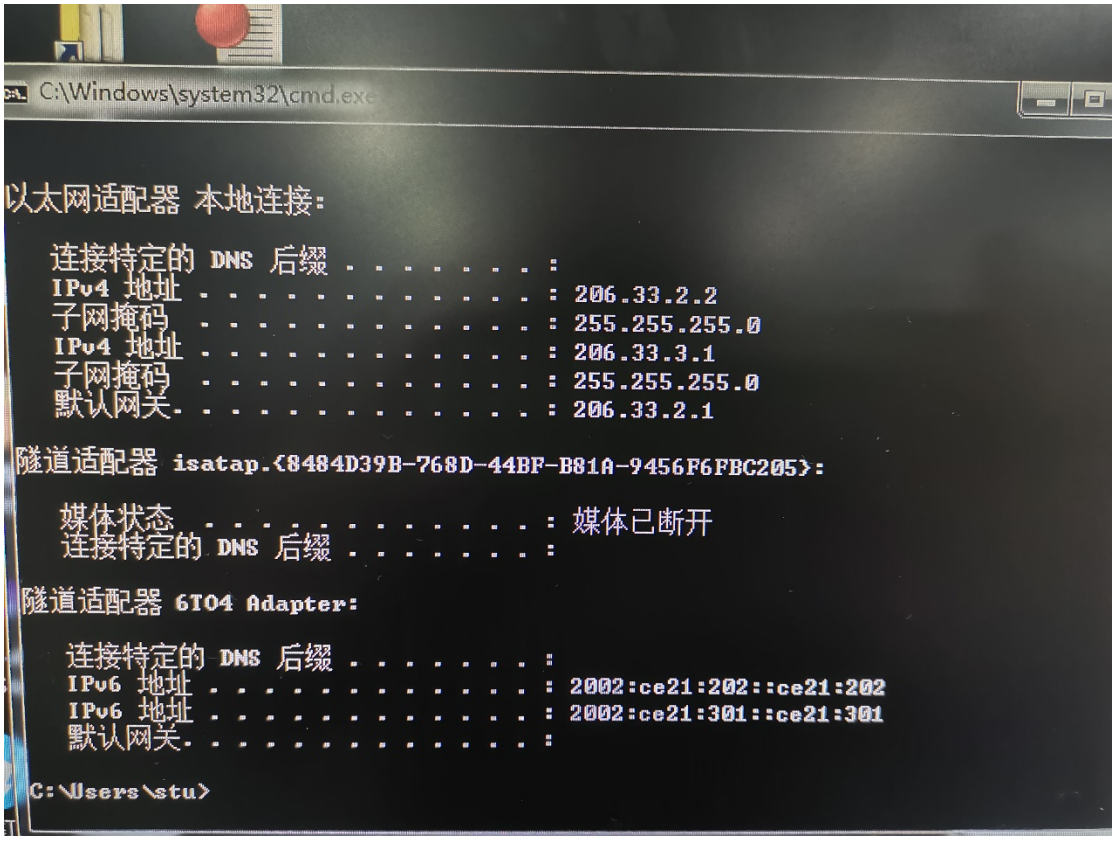
主机二 IP 网关设置


C:\Windows\system32\cmd.exe
C:\Users\stu>ipconfig
Windows IP 配置
以太网适配器 本地连接:
连接特定的 DNS 后缀 . . . . . :
IPv4 地址 . . . . . : 206.33.3.2
子网掩码 . . . . . : 255.255.255.0
默认网关. . . . . : 206.33.3.1
隧道适配器 isatap.{8484D39B-768D-44BF-B81A-9456F6FBC205}:
媒体状态 . . . . . : 媒体已断开
连接特定的 DNS 后缀 . . . . . :
隧道适配器 6TO4 Adapter:
连接特定的 DNS 后缀 . . . . . :
IPv6 地址 . . . . . : 2002:ce21:302::ce21:302
默认网关. . . . . :
C:\Users\stu>

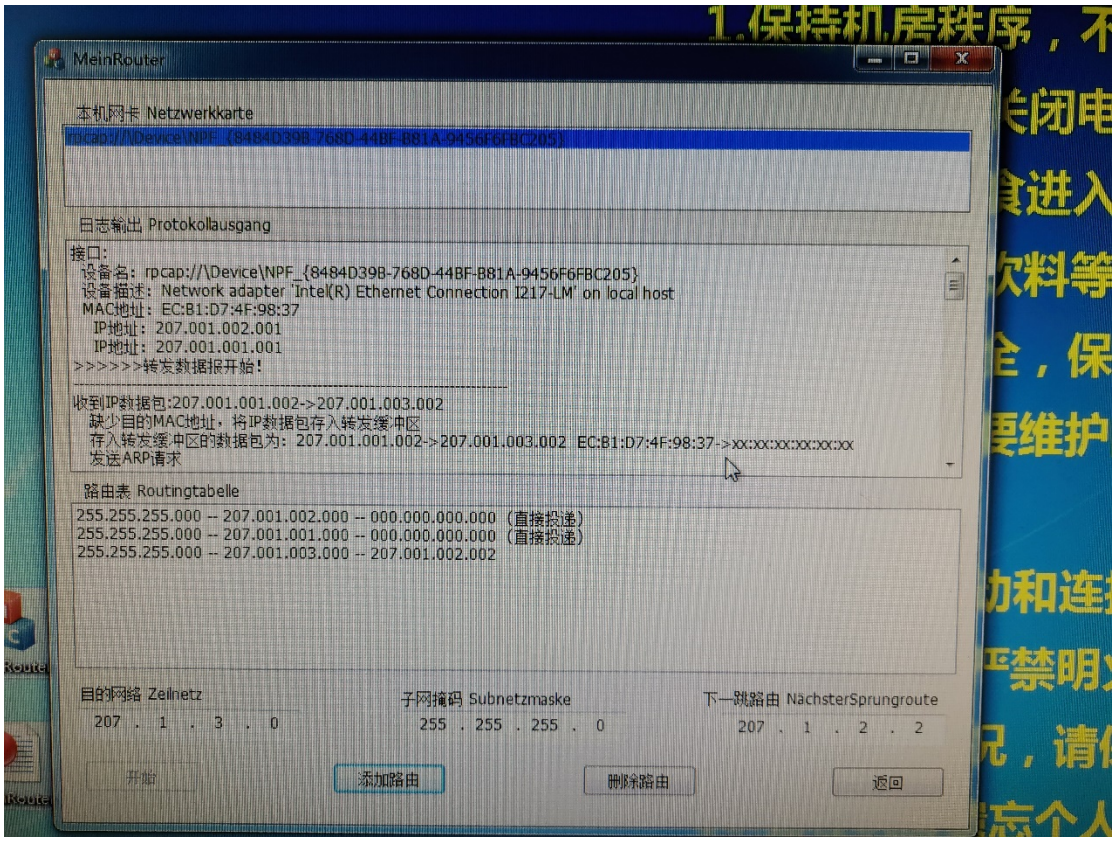
路由器一 IP 网关设置，需要打开服务中的 Route and Remoting Access。



路由器二 IP 网关设置，不要打开 Route and Remoting Access。运行我的路由器软件。



路由器程序运行界面，已添加相关表项。



使用主机一 ping 主机二，结果能 ping 通。