

网络技术与应用第五次实验

- 实验名称：简化路由器程序设计
- 专业：物联网工程
- 姓名：秦泽斌
- 学号：2212005

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. 需要给出路由表的建立和形成方式。
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，并讲解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

1. 报文设计

处理网络数据包的结构体和类，主要用于解析和记录ARP帧和IP帧

```
1  typedef struct FrameHeader_t {           //帧首部
2      BYTE DesMAC[6]; //目的地址
3      BYTE SrcMAC[6]; //源地址
4      WORD FrameType; //帧类型
5  }FrameHeader_t;
6
7  typedef struct IPHeader_t {              //IP首部
8      BYTE Ver_HLen; //IP协议版本和IP首部长度：高4位为版本，低4位为首部的长度
9      BYTE TOS; //服务类型
10     WORD TotalLen; //总长度
11     WORD ID; //标识
12     WORD Flag_Segment; //标志 片偏移
13     BYTE TTL; //生存周期
14     BYTE Protocol; //协议
15     WORD Checksum; //头部校验和
16     u_int SrcIP; //源IP
17     u_int DstIP; //目的IP
18 }IPHeader_t;
19
20 typedef struct ARPFrame_t {
21     FrameHeader_t FrameHeader;
22     WORD HardwareType; //硬件类型
23     WORD ProtocolType; //协议类型
24     BYTE HLen; //硬件地址长度
25     BYTE PLen; //协议地址长度
26     WORD Operation; //操作类型
27     BYTE SendHa[6]; //发送方MAC地址
```

```

28     DWORD SendIP;//发送方IP地址
29     BYTE RecvHa[6]; //接收方MAC地址
30     DWORD RecvIP;//接收方IP地址
31 }ARPFrame_t;
32
33 typedef struct Data_t {      //数据包
34     FrameHeader_t FrameHeader;
35     IPHeader_t IPHeader;
36 }Data_t;
37
38 typedef struct ICMP { //ICMP报文
39     FrameHeader_t FrameHeader;
40     IPHeader_t IPHeader;
41     char buf[0x80];
42 }ICMP_t;

```

2. 路由表设计

`RouterItem` 表示每一条路由表项。通过链表结构存储路由项，并支持添加、删除、打印以及查找路由的功能。

`RouterTable` 类用于管理所有路由表项，支持添加、删除、查找和打印操作。

- `RouterTable()`：构造函数，初始化链表，并添加直接相连的网络。通过本机网卡的IP和子网掩码计算出网络地址，并将这些直接连接的网络作为路由表的一部分添加进去。
- `RouterAdd(RouterItem* a)`：添加路由项。添加时，如果路由项类型为0（直接连接的网络），则直接插入到链表头部；否则，按照掩码的长短（子网掩码更长的优先）插入到合适的位置。
- `RouterRemove(int index)`：删除路由项。根据索引查找路由项，如果该项为直接连接的网络（`type == 0`），则不能删除，程序会输出错误提示。如果是用户添加的路由（`type == 1`），则可以删除。
- `print()`：打印所有路由表项的信息。
- `RouterFind(DWORD ip)`：查找最长前缀匹配的路由项，返回下一跳的IP地址。如果没有匹配的路由项，则返回-1。

```

1  class RouterItem//路由表表项
2  {
3  public:
4      DWORD mask;//掩码
5      DWORD net;//目的网络
6      DWORD nextip;//下一跳
7      BYTE nextmac[6];
8      int index;
9      int type;//0为直接连接，1为用户添加
10     RouterItem* nextitem;//采用链表形式存储
11     RouterItem()
12     {
13         memset(this, 0, sizeof(*this)); //全部初始化为0
14     }
15     void PrintItem()//打印表项内容：掩码、目的网络、下一跳IP、类型
16     {
17         in_addr addr;

```

```

18     printf("%d ", index);
19     addr.s_addr = mask;
20     char* temp = inet_ntoa(addr);
21     printf("%s\t", temp);
22     addr.s_addr = net;
23     temp = inet_ntoa(addr);
24     printf("%s\t", temp);
25     addr.s_addr = nextip;
26     temp = inet_ntoa(addr);
27     printf("%s\t", temp);
28     printf("%d\n", type);
29 }
30 };
31
32 class RouterTable//路由表
33 {
34 public:
35     RouterItem* head, * tail;
36     int num;//条数
37     RouterTable()//初始化, 添加直接相连的网络
38     {
39         head = new RouterItem;
40         tail = new RouterItem;
41         head->nextitem = tail;
42         num = 0;
43         for (int i = 0; i < 2; i++)
44         {
45             RouterItem* temp = new RouterItem;
46             temp->net = (inet_addr(ip[i])) & (inet_addr(mask[i]));//本机网卡的
47             //ip和掩码进行按位与即为所在网络
48             temp->mask = inet_addr(mask[i]);
49             temp->type = 0;//0表示直接连接, 不可删除
50             this->RouterAdd(temp);
51         }
52     }
53     void RouterAdd(RouterItem* a)//路由表的添加
54     {
55         RouterItem* pointer;
56         if (!a->type)
57         {
58             a->nextitem = head->nextitem;
59             head->nextitem = a;
60             a->type = 0;
61         }
62         else
63         {
64             for (pointer = head->nextitem; pointer != tail && pointer->
65             nextitem != tail; pointer = pointer->nextitem)
66             {
67                 if (a->mask < pointer->mask && a->mask >= pointer->
68                 nextitem->mask || pointer->nextitem == tail)
69                 {
70                     break;
71                 }
72             }
73             a->nextitem = pointer->nextitem;
74         }
75     }
76 };

```

```

71         pointer->nextitem = a;
72     }
73     RouterItem* p = head->nextitem;
74     for (int i = 0; p != tail; p = p->nextitem, i++)
75     {
76         p->index = i;
77     }
78     num++;
79 }
80 void RouterRemove(int index)//路由表的删除
81 {
82     for (RouterItem* t = head; t->nextitem != tail; t = t->nextitem)
83     {
84         if (t->nextitem->index == index)
85         {
86             if (t->nextitem->type == 0)
87             {
88                 printf("该项不可删除\n");
89                 return;
90             }
91             else
92             {
93                 t->nextitem = t->nextitem->nextitem;
94                 return;
95             }
96         }
97     }
98     printf("无该表项\n");
99 }
100 void print()
101 {
102     for (RouterItem* p = head->nextitem; p != tail; p = p->nextitem)
103     {
104         p->PrintItem();
105     }
106 }
107 DWORD RouterFind(DWORD ip)
108 {
109     for (RouterItem* t = head->nextitem; t != tail; t = t->nextitem)
110     {
111         if ((t->mask & ip) == t->net)
112         {
113             return t->nextip;
114         }
115     }
116     return -1;
117 }
118 };

```

3. 功能函数

1. GetOtherMac ()

- **功能:** 通过发送伪造的ARP请求来获取指定IP地址的MAC地址。构造一个ARP请求数据包, 源MAC地址为本机的MAC地址, 目标MAC地址为广播地址。然后通过PCAP发送该ARP请求并监听返回的ARP应答包, 提取其中的源MAC地址。

```
1 void GetOtherMac(DWORD ip0, BYTE mac[])
2 {
3     memset(mac, 0, sizeof(mac));
4     ARPFrame_t ARPFrame;
5     //将ARPFrame.FrameHeader.DesMAC设置为广播地址
6     for (int i = 0; i < 6; i++)
7         ARPFrame.FrameHeader.DesMAC[i] = 0xff;
8     //将ARPFrame.FrameHeader.SrcMAC设置为本机网卡的MAC地址
9     for (int i = 0; i < 6; i++)
10    {
11        ARPFrame.FrameHeader.SrcMAC[i] = selfmac[i];
12        ARPFrame.SendHa[i] = selfmac[i];
13    }
14    ARPFrame.FrameHeader.FrameType = htons(0x0806); //帧类型为ARP
15    ARPFrame.HardwareType = htons(0x0001); //硬件类型为以太网
16    ARPFrame.ProtocolType = htons(0x0800); //协议类型为IP
17    ARPFrame.HLen = 6; //硬件地址长度为6
18    ARPFrame.PLen = 4; //协议地址长为4
19    ARPFrame.Operation = htons(0x0001); //操作为ARP请求
20    //将ARPFrame.SendIP设置为本机网卡上绑定的IP地址
21    ARPFrame.SendIP = inet_addr(ip0);
22    //将ARPFrame.RecvHa设置为0
23    for (int i = 0; i < 6; i++)
24    {
25        ARPFrame.RecvHa[i] = 0;
26    }
27    //将ARPFrame.RecvIP设置为请求的IP地址
28    ARPFrame.RecvIP = ip0;
29    if (adhandle == nullptr)
30    {
31        printf("网卡接口打开错误\n");
32    }
33    else
34    {
35        if (pcap_sendpacket(adhandle, (u_char*)&ARPFrame,
36            sizeof(ARPFrame_t)) != 0)
37        {
38            //发送错误处理
39            printf("发送错误\n");
40            return;
41        }
42        else
43        {
44            //发送成功
45            while (1)
46            {
47                pcap_pkthdr* pkt_header;
```

```

47         const u_char* pkt_data;
48         int rtn = pcap_next_ex(adhandle, &pkt_header,
&pkt_data);
49         if (rtn == 1)
50         {
51             ARPFrame_t* IPPacket = (ARPFrame_t*)pkt_data;
52             if (ntohs(IPPacket->FrameHeader.FrameType) ==
0x0806)
53                 { //输出目的MAC地址
54                     if (ntohs(IPPacket->Operation) == 0x0002) //如果帧
类型为ARP并且操作为ARP应答
55                     {
56                         LT.WritelogARP(IPPacket);
57                         //输出源MAC地址
58                         for (int i = 0; i < 6; i++)
59                             mac[i] = IPPacket-
>FrameHeader.SrcMAC[i];
60                         break;
61                     }
62                 }
63             }
64         }
65     }
66 }
67 }

```

2. Checksum ()

- **功能：**计算和验证IP报文的校验和。

```

1  bool CheckSum(Data_t* temp)
2  {
3      unsigned int sum = 0;
4      WORD* t = (WORD*)&temp->IPHeader;
5      for (int i = 0; i < sizeof(IPHeader_t) / 2; i++)
6      {
7          sum += t[i];
8          while (sum >= 0x10000) //如果溢出，则进行回卷
9          {
10             int s = sum >> 16;
11             sum -= 0x10000;
12             sum += s;
13         }
14     }
15     if (sum == 65535)
16     {
17         return 1;
18     }
19     else
20     {
21         return 0;
22     }
23 }

```

3. resent ()

- **功能：**转发数据包，修改源MAC和目标MAC地址后重新发送。接收到的ICMP数据包（包含IP数据报文）被修改源MAC为本机MAC，目标MAC为下一跳的MAC，并将TTL减1后重新发送。

```
1 void resend(ICMP_t data, BYTE desmac[])
2 {
3     printf("已转发数据包:\n");
4     printf("原源MAC: \t");
5     for (int i = 0; i < 6; i++)
6         printf("%02x:", data.FrameHeader.SrcMAC[i]);
7     printf("\n");
8     printf("原目的MAC: \t");
9     for (int i = 0; i < 6; i++)
10        printf("%02x:", data.FrameHeader.DesMAC[i]);
11    printf("\n");
12    Data_t* temp = (Data_t*)&data;
13    memcpy(temp->FrameHeader.SrcMAC, temp->FrameHeader.DesMAC, 6); //源
    MAC为本机MAC
14    memcpy(temp->FrameHeader.DesMAC, desmac, 6); //目的MAC为下一跳MAC
15
16    printf(" 现源MAC: \t");
17    for (int i = 0; i < 6; i++)
18        printf("%02x:", temp->FrameHeader.SrcMAC[i]);
19    printf("\n");
20
21    printf("现目的MAC: \t");
22    for (int i = 0; i < 6; i++)
23        printf("%02x:", temp->FrameHeader.DesMAC[i]);
24    printf("\n");
25
26
27    temp->IPHeader.TTL -= 1;
28    if (temp->IPHeader.TTL < 0)
29    {
30        return;
31    }
32    SetChecksum(temp); //重新设置校验和
33    int rtn = pcap_sendpacket(adhandle, (const u_char*)temp, 74); //发送数
    据报
34    if (rtn == 0)
35    {
36        LT.WritelogIP("转发", temp);
37    }
38 }
```

4. DWORD WINAPI Thread(LPVOID lparam)

- **功能：**线程函数，不断监听网络数据包。监听PCAP捕获到的数据包，如果是自己的MAC地址并且是ARP包或IP包，则根据路由表进行数据包的转发。

```
1 DWORD WINAPI Thread(LPVOID lparam)
2 {
3     RouterTable RT = *(RouterTable*)(LPVOID)lparam;
4     while (1)
```

```

5      {
6          pcap_pkthdr* pkt_header;
7          const u_char* pkt_data;
8          while (1)
9          {
10             int rtn = pcap_next_ex(adhandle, &pkt_header, &pkt_data);
11             if (rtn)//接收到消息
12             {
13                 break;
14             }
15         }
16         FrameHeader_t* header = (FrameHeader_t*)pkt_data;
17         if (Compare(header->DesMAC, selfmac))//目的mac是自己的mac
18         {
19             if (ntohs(header->FrameType) == 0x0806)//收到ARP
20             {
21                 //do nothing
22             }
23             else if (ntohs(header->FrameType) == 0x0800)//收到IP
24             {
25                 Data_t* data = (Data_t*)pkt_data;
26                 LT.WritelogIP("接收", data);
27                 DWORD dstip = data->IPHeader.DstIP;
28                 DWORD IFip = RT.RouterFind(dstip);//查找是否有对应表项
29                 if (IFip == -1)
30                 {
31                     continue;
32                 }
33                 if (Checksum(data))//如果校验和不正确，则直接丢弃不进行处理
34                 {
35                     if (data->IPHeader.DstIP != inet_addr(ip[0]) &&
data->IPHeader.DstIP != inet_addr(ip[1]))
36                     {
37                         int t1 = Compare(data->FrameHeader.DesMAC,
broadcast);
38                         int t2 = Compare(data->FrameHeader.SrcMAC,
broadcast);
39                         if (!t1 && !t2)
40                         {
41                             //ICMP报文包含IP数据包报头和其它内容
42                             ICMP_t* temp_ = (ICMP_t*)pkt_data;
43                             ICMP_t temp = *temp_;
44                             BYTE mac[6];
45                             if (IFip == 0)
46                             {
47                                 //如果ARP表中没有所需内容，则需要获取ARP
48                                 if (!ArpTable::FindArp(dstip, mac))
49                                 {
50                                     ArpTable::InsertArp(dstip, mac);
51                                 }
52                                 resend(temp, mac);
53                             }
54                             else if (IFip != -1)//非直接投递，查找下一条IP的
MAC
56                             {

```



```

57         if (!ArpTable::FindArp(IFip, mac))
58         {
59             ARPTable::InsertArp(IFip, mac);
60         }
61         resend(temp, mac);
62     }
63 }
64 }
65 }
66 }
67 }
68 }
69 }

```

三、实验结果

1. 添加路由表

首先根据要求在主机3上添加路由表项

```

CA 命令提示符

IPv4 Route Table
=====
Interface List
0x1 ..... MS TCP Loopback interface
0x10003 ...00 0c 29 c2 c4 f1 ..... Intel(R) PRO/1000 MT Network Connection
=====
Active Routes:
Network Destination        Netmask          Gateway          Interface        Metric
127.0.0.0                  255.0.0.0        127.0.0.1        127.0.0.1         1
206.1.1.0                  255.255.255.0    206.1.2.1        206.1.2.2         1
206.1.2.0                  255.255.255.0    206.1.2.2        206.1.2.2        10
206.1.2.2                  255.255.255.255  127.0.0.1        127.0.0.1        10
206.1.2.255               255.255.255.255  206.1.2.2        206.1.2.2        10
206.1.3.0                  255.255.255.0    206.1.3.1        206.1.2.2        10
206.1.3.1                  255.255.255.255  127.0.0.1        127.0.0.1        10
206.1.3.255               255.255.255.255  206.1.3.1        206.1.2.2        10
224.0.0.0                  240.0.0.0        206.1.2.2        206.1.2.2        10
255.255.255.255           255.255.255.255  206.1.2.2        206.1.2.2         1
=====
Persistent Routes:
None

C:\Documents and Settings\Administrator>

```

其次在主机2上打开路由程序，选择网卡1，添加对应路由表项

```
C:\Documents and Settings\Administrator\桌面\Router.exe
以下为可用的网络适配器:
1:Network adapter 'Intel(R) PRO/1000 MT Network Connection' on local host
请输入要打开的网络接口号 (1~1):
1
正在监听: Network adapter 'Intel(R) PRO/1000 MT Network Connection' on local host
206.1.2.1      255.255.255.0
206.1.1.1      255.255.255.0
已发送伪造的ARP请求包
捕获到ARP响应
Mac地址: 00:0c:29:ea:ad:8f
请输入相应数字进行对应操作:
1: 打印路由表; 2: 添加路由表项; 3: 删除路由表项; 0: 退出
2
DesIP: 206.1.3.0
Mask: 255.255.255.0
NestPop: 206.1.2.2
请输入相应数字进行对应操作:
1: 打印路由表; 2: 添加路由表项; 3: 删除路由表项; 0: 退出
S_
```

2. 删除路由表（测试）

尝试删除默认表项，结果不可行；

```
C:\Documents and Settings\Administrator\桌面\Router.exe
206.1.1.1      255.255.255.0
已发送伪造的ARP请求包
捕获到ARP响应
Mac地址: 00:0c:29:ea:ad:8f
请输入相应数字进行对应操作:
1: 打印路由表; 2: 添加路由表项; 3: 删除路由表项; 0: 退出
2
DesIP: 206.1.3.0
Mask: 255.255.255.0
NestPop: 206.1.2.2
请输入相应数字进行对应操作:
1: 打印路由表; 2: 添加路由表项; 3: 删除路由表项; 0: 退出
1
0 255.255.255.0 206.1.1.0      0.0.0.0 0
1 255.255.255.0 206.1.2.0      0.0.0.0 0
2 255.255.255.0 206.1.3.0      206.1.2.2      1
请输入相应数字进行对应操作:
1: 打印路由表; 2: 添加路由表项; 3: 删除路由表项; 0: 退出
3
请输入删除表项编号: (可以使用打印操作查看)
1
该项不可删除
```

尝试删除新添加的表项，结果可行；

```
C:\Documents and Settings\Administrator\桌面\Router.exe
t
206.1.2.1      255.255.255.0
206.1.1.1      255.255.255.0
已发送伪造的ARP请求包
捕获到ARP响应
Mac地址: 00:0c:29:ea:ad:8f
请输入相应数字进行对应操作:
1: 打印路由表; 2: 添加路由表项; 3: 删除路由表项; 0: 退出
2
DesIP: 206.1.3.0
Mask: 255.255.255.0
NestPop: 206.1.2.2
请输入相应数字进行对应操作:
1: 打印路由表; 2: 添加路由表项; 3: 删除路由表项; 0: 退出
3
请输入删除表项编号: (可以使用打印操作查看)
2
请输入相应数字进行对应操作:
1: 打印路由表; 2: 添加路由表项; 3: 删除路由表项; 0: 退出
1
0 255.255.255.0 206.1.1.0      0.0.0.0 0
1 255.255.255.0 206.1.2.0      0.0.0.0 0
```

3. 连通性测试

主机1与主机4互相ping通:

```
C:\命令提示符
Microsoft Windows [版本 5.2.3790]
(C) 版权所有 1985-2003 Microsoft Corp.

C:\Documents and Settings\Administrator>ping 206.1.3.2

Pinging 206.1.3.2 with 32 bytes of data:

Reply from 206.1.3.2: bytes=32 time=3139ms TTL=126
Reply from 206.1.3.2: bytes=32 time=2008ms TTL=126
Reply from 206.1.3.2: bytes=32 time=2013ms TTL=126
Reply from 206.1.3.2: bytes=32 time=1998ms TTL=126

Ping statistics for 206.1.3.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1998ms, Maximum = 3139ms, Average = 2289ms

C:\Documents and Settings\Administrator>
```

```
C:\ 命令提示符
Microsoft Windows [版本 5.2.3790]
(C) 版权所有 1985-2003 Microsoft Corp.

C:\Documents and Settings\Administrator>ping 206.1.1.2

Pinging 206.1.1.2 with 32 bytes of data:

Reply from 206.1.1.2: bytes=32 time=3584ms TTL=126
Reply from 206.1.1.2: bytes=32 time=2009ms TTL=126
Reply from 206.1.1.2: bytes=32 time=2009ms TTL=126
Reply from 206.1.1.2: bytes=32 time=2012ms TTL=126

Ping statistics for 206.1.1.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 2009ms, Maximum = 3584ms, Average = 2403ms
```

4. 转发日志

```
C:\Documents and Settings\Administrator\桌面\Router.exe
原源MAC: 00:0c:29:9e:b0:5b:
原目的MAC: 00:0c:29:ea:ad:8f:
现源MAC: 00:0c:29:ea:ad:8f:
现目的MAC: 00:0c:29:c2:c4:f1:
已转发数据包:
原源MAC: 00:0c:29:c2:c4:f1:
原目的MAC: 00:0c:29:ea:ad:8f:
现源MAC: 00:0c:29:ea:ad:8f:
现目的MAC: 00:0c:29:9e:b0:5b:
已转发数据包:
原源MAC: 00:0c:29:9e:b0:5b:
原目的MAC: 00:0c:29:ea:ad:8f:
现源MAC: 00:0c:29:ea:ad:8f:
现目的MAC: 00:0c:29:c2:c4:f1:
已转发数据包:
原源MAC: 00:0c:29:c2:c4:f1:
原目的MAC: 00:0c:29:ea:ad:8f:
现源MAC: 00:0c:29:ea:ad:8f:
现目的MAC: 00:0c:29:9e:b0:5b:
已转发数据包:
原源MAC: 00:0c:29:9e:b0:5b:
原目的MAC: 00:0c:29:ea:ad:8f:
现源MAC: 00:0c:29:ea:ad:8f:
现目的MAC: 00:0c:29:c2:c4:f1:
S
```

四、总结与分析

通过本次实验，我不仅深入了解了路由表的基本工作原理，还掌握了如何利用链表来管理路由表项。我还学习了如何通过面向对象的方法设计和实现路由表的相关操作。实验过程中，我还意识到性能优化和内存管理的重要性，未来在类似的项目中会更加注重这些方面。

这次实验不仅加深了我对网络协议、路由原理的理解，也提高了我在实际编码中对数据结构和内存管理的运用能力，为后续的网络编程课程和实际项目奠定了基础。