

实验报告

实验要求

实现路由程序，实现IP数据报和ARP数据报的捕获和转发，并能与其他路由器协同工作。

设计思路

本次实验主要包括以下模块：

- 路由表的建立和查找。
- ARP缓存的建立和查找，过期处理。
- ARP报文的处理。
- IP数据报的处理，对于直接投递和转发这两种情况的区别。
- 本机网卡IP地址和MAC地址的获取。

实现方法

路由表

路由表中的每一条包含以下字段：

- 网段。
- 子网掩码。
- 下一跳。
- 是否为直接投递。

```
1  typedef struct RouterTable_t {
2      DWORD IP;
3      DWORD Mask;
4      DWORD Next;
5      WORD flag;
6  } RouterTable_t;
7  vector<RouterTable_t> routerTable;
```

使用 `vector` 数据结构管理所有的路由表。

ARP缓存

ARP缓存中的每一条包含以下字段：

- 建立时间。
- 保存时长。
- IP地址。
- MAC地址。

```

1 typedef struct ARPTable_t {
2     clock_t time;
3     int keep;
4     DWORD IP;
5     BYTE Mac[6];
6 } ARPTable_t;
7 vector<ARPTable_t> ARPTable;

```

使用 `vector` 数据结构管理所有的ARP缓存。

路由表的添加

添加路由表需要输入网段、子网掩码和下一跳。如果添加的路由表项为直接投递，则下一跳的位置输入 `d`，以此决定路由表项的 `flag`：

```

1 string IP, mask, next;
2 cin >> IP >> mask >> next;
3 RouterTable_t t;
4 inet_pton(AF_INET, IP.c_str(), &t.IP);
5 inet_pton(AF_INET, mask.c_str(), &t.Mask);
6 if (next == "D" || next == "d")
7 {
8     t.flag = RO_DIRECT;
9     t.Next = 0;
10 }
11 else
12 {
13     inet_pton(AF_INET, next.c_str(), &t.Next);
14     t.flag = RO_STATIC;
15 }
16 routerTable.push_back(t);

```

路由表的查找

查询路由表的步骤如下：

- 遍历 `vector`，查找网络和掩码的与是否和网络号一致。
- 查询到一致的数据后，记录掩码长度，判断和上一次的掩码的大小。如果掩码更长，则保留当前查询项。
- 遍历结束，返回匹配项。

```

1 RouterTable_t getRouteEntry(DWORD ip)
2 {
3     auto it = routerTable.begin();
4     RouterTable_t choose;
5     choose.Mask = 0;
6     while (it != routerTable.end())
7     {
8         if ((ip & it->Mask) == (it->IP & it->Mask) && it->Mask >= choose.Mask)
9             choose = *it;
10        it++;
11    }
12    return choose;
13 }

```

路由表的删除

删除路由表需要输入删除的网段，查询路由表将匹配项删除即可：

```

1 string delIP;
2 cin >> delIP;
3 auto it = routerTable.begin();
4 for (; it != routerTable.end(); it++)
5 {
6     DWORD ip;
7     inet_pton(AF_INET, delIP.c_str(), &ip);
8     if (it->IP == ip)
9         break;
10 }
11 if(it->flag)
12 routerTable.erase(it);

```

ARP缓存查找

查询ARP缓存的步骤如下：

- 遍历ARP缓存，查找IP地址。
- 如果存在，则判断时间是否过期，，如果没过期则返回true，保存MAC地址。
- 其余情况返回false。

```

1 bool getMAC(DWORD ip, BYTE MAC[])
2 {
3     auto it = ARPTable.begin();
4     for (; it != ARPTable.end(); it++)
5     {
6         if (it->IP == ip)
7         {
8             // 判断表项是否有效
9             time_t local = clock();
10            if (it->time + it->keep > local)

```

```

11         {
12             memcpy(MAC, it->Mac, 6);
13             return true;
14         }
15         else
16             break;
17     }
18 }
19 if(it != ARPTable.end())
20     ARPTable.erase(it);
21 sendARP(adhandle, getSendIP(ip), deviceMAC, ip, nullptr, false);
22 return false;
23 }

```

本机MAC地址的获取

构造虚拟主机，发送ARP请求，获取本机的ARP响应。与实验3相同。

ARP报文的处理

收到ARP报文，需要判断是否为ARP响应，如果是，则读取IP和MAC地址，设置过期时间，写入ARP缓存中：

```

1  ARPFrame_t* ARPFrame = (ARPFrame_t*)packet;
2  int op = ARPFrame->Operation;
3  if (op == ntohs(2))
4  {
5      ARPTable_t t;
6      t.IP = ARPFrame->SendIP;
7      memcpy(t.Mac, ARPFrame->SendHa, 6);
8      t.time = clock();
9      t.keep = 300000;
10     ARPTable.push_back(t);
11     char output[INET_ADDRSTRLEN];
12     inet_ntop(AF_INET, &t.IP, output, INET_ADDRSTRLEN);
13     string MACstr = format("{:02X}::{:02X}::{:02X}::{:02X}::{:02X}::{:02X}",
ARPFrame->SendHa[0], ARPFrame->SendHa[1], ARPFrame->SendHa[2], ARPFrame->SendHa[3], ARPFrame->SendHa[4], ARPFrame->SendHa[5]);
14     cout << "收到ARP报文响应，添加ARP表项: " << output << " " << MACstr << endl;
15     continue;
16 }

```

IP数据报的处理

对于IP数据报，需要经过以下步骤：

- 校验并检查以太帧头部信息中的目的MAC地址是否为本机MAC地址，如果不是，则丢弃。
- 查询路由表项，如果是直接投递，则应将MAC地址设置为目的MAC地址；如果不是，则应将MAC地址设置为下一跳的MAC地址。

- 查询MAC地址，如果存在，则填入报文，将TTL减一，重新计算头部检验和，转发数据报；如果不存在，发送ARP请求。

```
1 // IP数据包需要判断捕获长度，并转发
2 Data_t* IPPacket = (Data_t*)packet;
3 recvIP = IPPacket->IPHeader.DstIP;
4 int len = ntohs(IPPacket->IPHeader.TotalLen);
5 RouterTable_t choose;
6 choose = getRouteEntry(recvIP);
7 //
8 if (calcChecksum((uint16_t*)&IPPacket->IPHeader, sizeof(IPHeader_t)) != 0)
9 {
10     cout << "校验错误" << endl;
11     continue;
12 }
13 IPPacket->IPHeader.Checksum = 0;
14 IPPacket->IPHeader.TTL -= 1;
15 IPPacket->IPHeader.Checksum = calcChecksum((uint16_t*)&IPPacket->IPHeader,
16     sizeof(IPHeader_t));
17 if (choose.flag == RO_DIRECT)
18 {
19     bool ret = getMAC(IPPacket->IPHeader.DstIP, IPPacket->FrameHeader.DesMAC);
20     if(ret)
21         pcap_sendpacket(adhandle, (u_char*)packet, sizeof(FrameHeader_t) + len);
22     if (ret) cout << "获取MAC地址成功 ";
23     cout << "直接投递 ";
24 }
25 else if(compMAC(IPPacket->FrameHeader.DesMAC, deviceMAC))
26 {
27     bool ret = getMAC(choose.Next, IPPacket->FrameHeader.DesMAC);
28     if(ret)
29         pcap_sendpacket(adhandle, (u_char*)packet, sizeof(FrameHeader_t) + len);
30     if (ret) cout << "获取MAC地址成功 ";
31     cout << "路由转发 ";
32 }
```

其他

为了加快程序执行，我们采取双线程，一个线程负责抓包，将获取的报文加入到消息队列中；另一个线程负责处理从消息队列中取出并处理数据报文。

```
1 thread cap(capData); // 建立子线程
2 // 主线程处理IP数据包和ARP报文
3 while (true)
4 {
5     Packet_t* packet;
6     {
7         lock_guard<mutex> lock(queueMutex);
8         if (msgQuene.size() == 0)
9             continue;
10        packet = msgQuene.front();
```

```
11         msgQuene.pop();
12     }
13     // ...
14 }
```

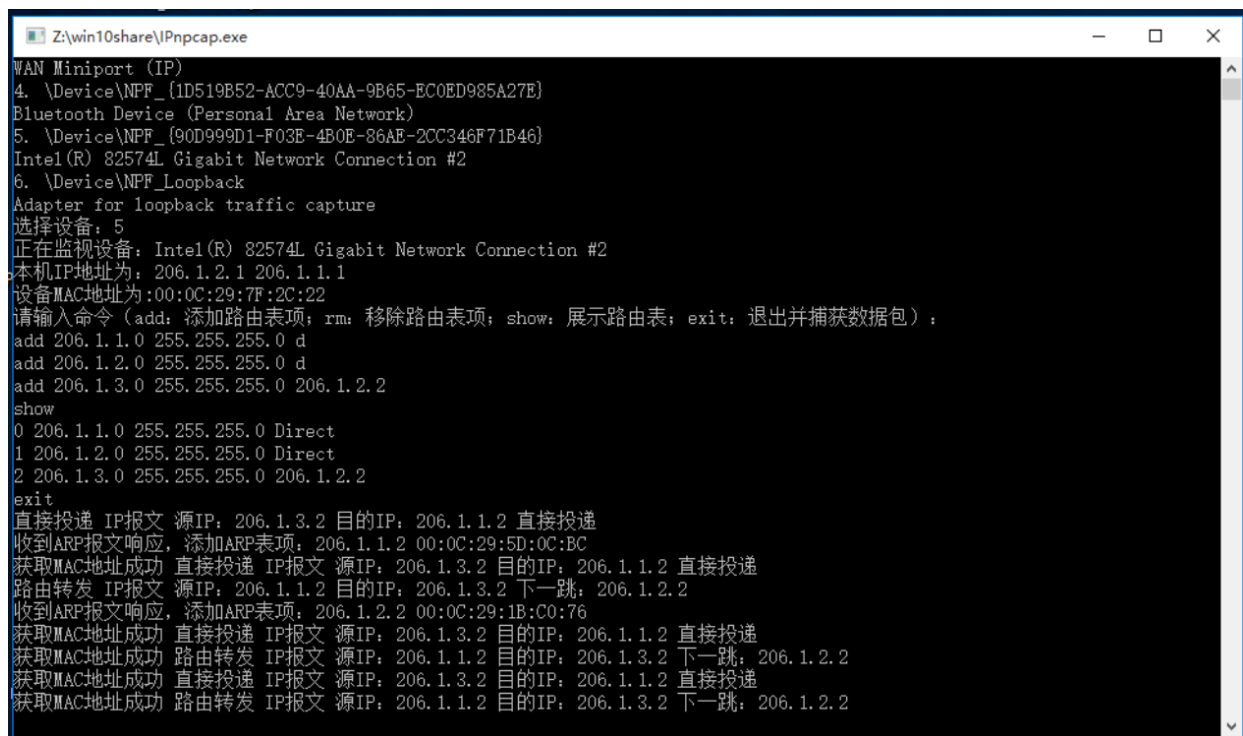
测试方法

添加直接投递和转发的路由表项，在虚拟机上运行，在需要ping的主机上运行wireshark。由于在虚拟机中，四台虚拟机采取桥接网络，在vmware中使用的是相同的网卡，所以可以直接看到所有的转发过程。

查看wireshark是否能识别转发过程中发送的数据报，并且各主机是否能正确响应。

实验结果

路由程序运行截图，包括配置路由表项和工作日志：



```
Z:\win10share\IPnpcap.exe
WAN Miniport (IP)
4. \Device\NPF_{1D519B52-ACC9-40AA-9B65-EC0ED985A27E}
Bluetooth Device (Personal Area Network)
5. \Device\NPF_{90D999D1-F03E-4B0E-86AE-2CC346F71B46}
Intel(R) 82574L Gigabit Network Connection #2
6. \Device\NPF_{Loopback}
Adapter for loopback traffic capture
选择设备: 5
正在监视设备: Intel(R) 82574L Gigabit Network Connection #2
本机IP地址为: 206.1.1.1 206.1.1.1
设备MAC地址为: 00:0C:29:7F:2C:22
请输入命令 (add: 添加路由表项; rm: 移除路由表项; show: 展示路由表; exit: 退出并捕获数据包):
add 206.1.1.0 255.255.255.0 d
add 206.1.2.0 255.255.255.0 d
add 206.1.3.0 255.255.255.0 206.1.2.2
show
0 206.1.1.0 255.255.255.0 Direct
1 206.1.2.0 255.255.255.0 Direct
2 206.1.3.0 255.255.255.0 206.1.2.2
exit
直接投递 IP报文 源IP: 206.1.3.2 目的IP: 206.1.1.2 直接投递
收到ARP报文响应, 添加ARP表项: 206.1.1.2 00:0C:29:5D:0C:BC
获取MAC地址成功 直接投递 IP报文 源IP: 206.1.3.2 目的IP: 206.1.1.2 直接投递
路由转发 IP报文 源IP: 206.1.1.2 目的IP: 206.1.3.2 下一跳: 206.1.2.2
收到ARP报文响应, 添加ARP表项: 206.1.2.2 00:0C:29:1B:C0:76
获取MAC地址成功 直接投递 IP报文 源IP: 206.1.3.2 目的IP: 206.1.1.2 直接投递
获取MAC地址成功 路由转发 IP报文 源IP: 206.1.1.2 目的IP: 206.1.3.2 下一跳: 206.1.2.2
获取MAC地址成功 直接投递 IP报文 源IP: 206.1.3.2 目的IP: 206.1.1.2 直接投递
获取MAC地址成功 路由转发 IP报文 源IP: 206.1.1.2 目的IP: 206.1.3.2 下一跳: 206.1.2.2
```

主机互相ping，均成功：

```
C:\ 命令提示符

C:\Documents and Settings\Administrator>ping 206.1.1.2

Pinging 206.1.1.2 with 32 bytes of data:

Request timed out.
Request timed out.
Reply from 206.1.1.2: bytes=32 time=1366ms TTL=126
Reply from 206.1.1.2: bytes=32 time=1996ms TTL=126

Ping statistics for 206.1.1.2:
    Packets: Sent = 4, Received = 2, Lost = 2 (50% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1366ms, Maximum = 1996ms, Average = 1681ms

C:\Documents and Settings\Administrator>ipconfig

Windows IP Configuration

Ethernet adapter 本地连接:

    Connection-specific DNS Suffix  . : 
    IP Address. . . . . : 206.1.3.2
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 206.1.3.1

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

```
C:\ 命令提示符

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=1453ms TTL=126
Reply from 206.1.3.2: bytes=32 time=1994ms TTL=126
Reply from 206.1.3.2: bytes=32 time=2043ms TTL=126
Reply from 206.1.3.2: bytes=32 time=2036ms 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 = 1453ms, Maximum = 2043ms, Average = 1881ms

C:\Documents and Settings\Administrator>ipconfig

Windows IP Configuration

Ethernet adapter 本地连接:

    Connection-specific DNS Suffix  . : 
    IP Address. . . . . : 206.1.1.2
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 206.1.1.1

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

在开始ping的时候有两次 Request timed out，对此的解释如下：

- 第一次发送，路由器A收到主机A的消息，查找路由表，得到下一跳。查询ARP缓存，不存在该项，则发送ARP广播，请求路由器B的MAC地址。第一次发送失败。
- 第二次发送，路由器A收到主机A的消息，转发给路由器B。主机B回应后，查找路由表，发现是直接发送给主机A，查询ARP缓存，不存在该项，则发送ARP广播，请求主机A的MAC地址。第二次发送失败。
- 第三四次发送，正常转发。
- 主机B后续ping主机A，由于时间间隔很短，ARP缓存有效，所以可以查找MAC地址并直接转发。

测试删除路由表项：

```
请输入命令（add：添加路由表项；del：移除路由表项；show：展示路由表；exit：退出并捕获数据包）：
add 206.1.1.0 255.255.255.0 d
add 206.1.2.0 255.255.255.0 d
show
0 206.1.1.0 255.255.255.0 Direct
1 206.1.2.0 255.255.255.0 Direct
del 206.1.1.0
show
0 206.1.2.0 255.255.255.0 Direct
|
```