**《计算机网络实验》实验报告**

实验名称：**静态路由编程实现**

姓名：胡育玮

学号：171860574

邮箱：[yeevee@qq.com](mailto:yeevee@qq.com)

班级：17级计算机科学与技术系 2 班

一、实验目的

1、了解Ethernet、IP、ARP等格式的数据包的结构

2、了解ARP协议的实现细节

3、了解路由器的转发原理

二、数据结构说明

PC1：

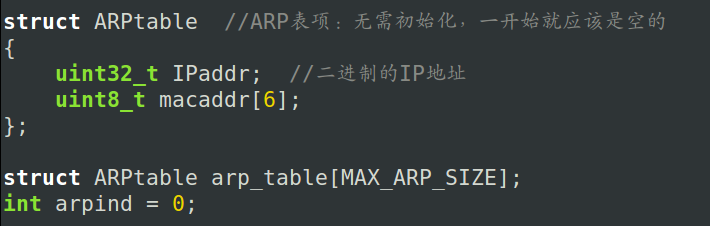
PC要实现IP（ICMP）数据包的构建和发送以及ARP包的构建和发送。使用的数据结构：

（1）ethhdr, ip, icmp : 分别代表以太网包头，IP包，ICMP包的结构体。这3个结构体使用Linux系统自带的相应的结构体类型，其各个字段不再赘述。

（2）ARP包：使用自己定义的结构体类型：



（3）ARP表结构体类型以及ARP表：

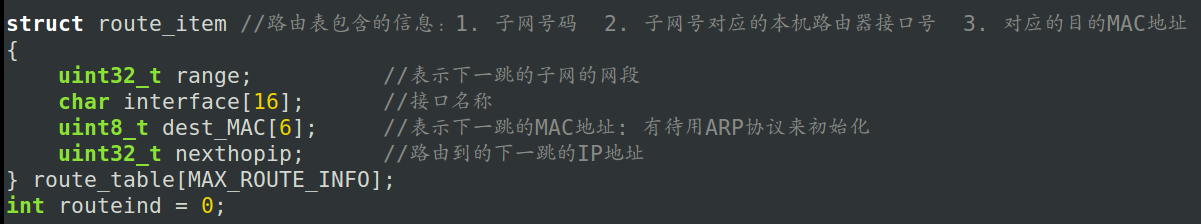


PC2：与PC1完全一致

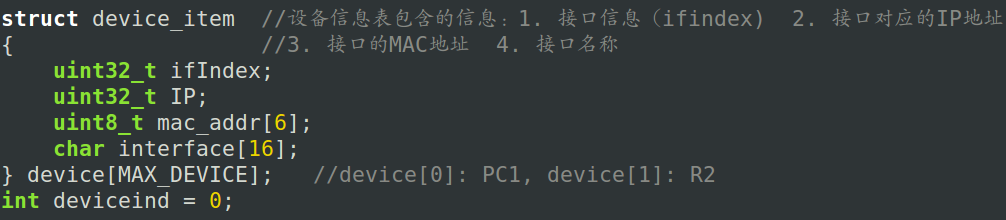
Router1：

（1）ethhdr, ip, icmp : 同PC1

（2）自定义的ARP结构类型、ARP表结构体类型、ARP表：同PC1

（3）路由表结构体类型 和 路由表：

（4）设备信息表结构体类型和设备信息表：

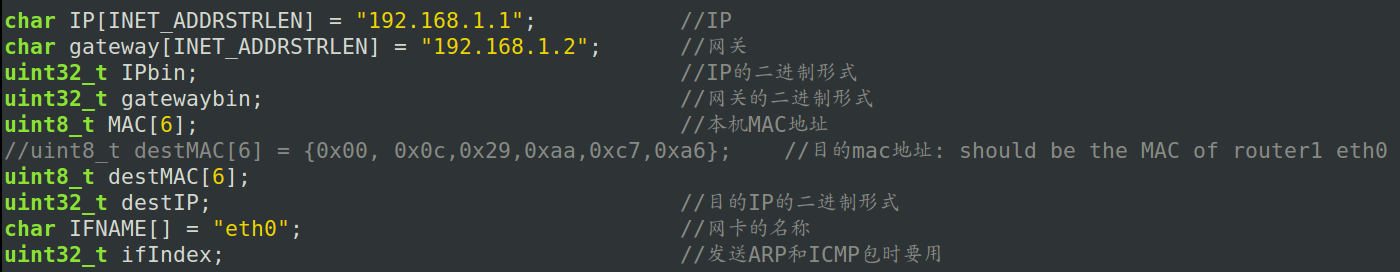


Router2：与Router1完全一致

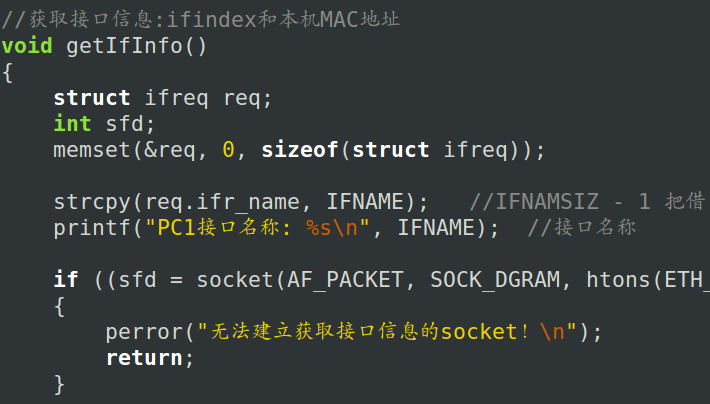
三、配置文件说明

在我的实现中，不需要配置文件。但必要的配置信息是必须要有的，比如本机的IP地址需要人为地指定；由于本实验仅要求做到静态路由故路由表中各个表项的一些字段应初始化；等等。

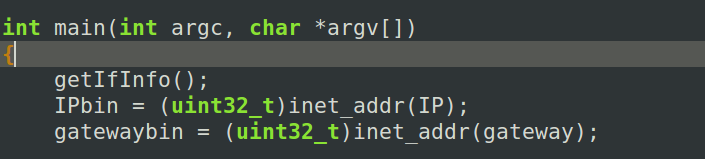
在我的实现中，采用在各个机器（共4台机器，两台PC两台路由器）的程序中设立初始化好的全局变量和初始化函数的方式来实现机器属性的配置：

PC1：

以上是为PC1指定相应参数的全局变量。



上图为初始化函数：getIfInfo()。该函数获取PC1的MAC地址和ifindex，并将其存储到前面定义好的MAC[]全局数组和ifIndex全局变量中。

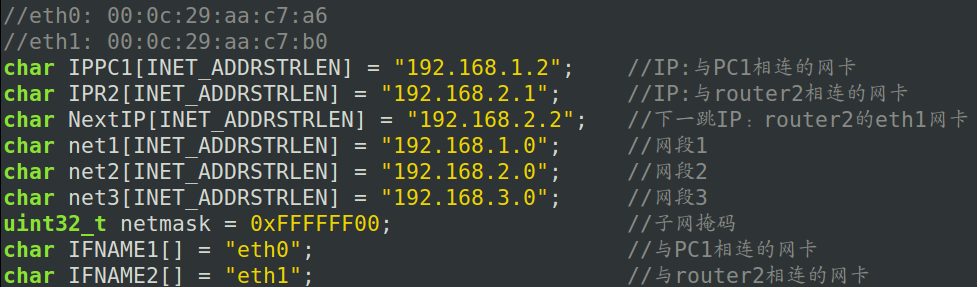


在main函数内一开始，先调用getIfInfo()来获取本机MAC地址和ifindex，在调用inet\_addr()来得到本机IP地址和本机网关地址的二进制形式，存到IPbin和gatewaybin中。

通过以上过程，PC1实现了初始化配置。

PC2的初始化配置也是完全一样的。IP地址和网关会不一样。

对router1：



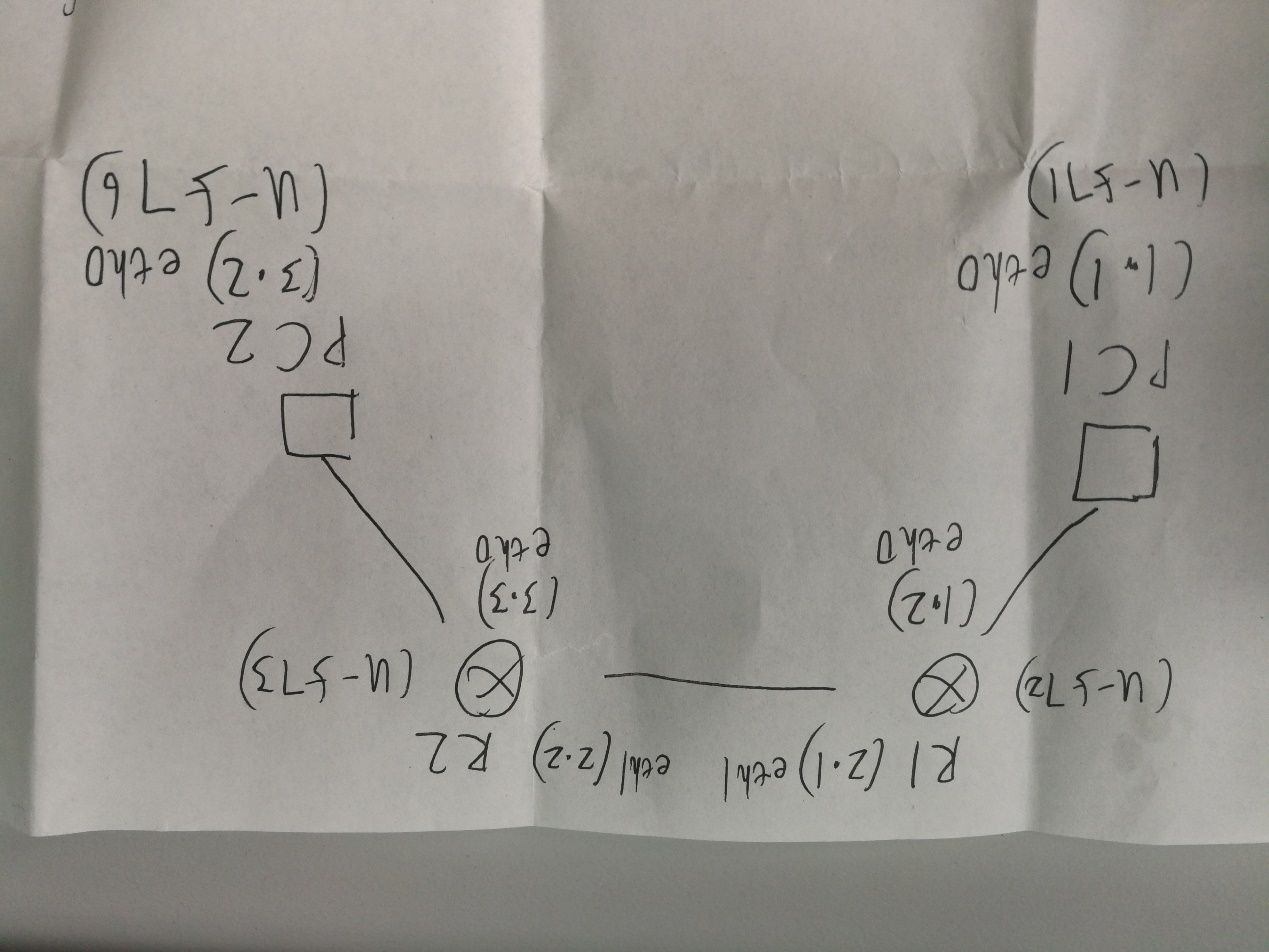
然后在init\_router()中，利用上图中的信息来初始化device[]数组（设备信息表），填写两张网卡的IP地址，MAC地址等信息；初始化route\_table[]路由表，填写网段、到达这一网段对应的接口的名称、下一跳的IP地址。路由表还有一个字段是下一跳的MAC地址，这个不能初始化，应在程序运行时由ARP协议来获取。

对router2也是如此。

最终各个设备的配置如下：

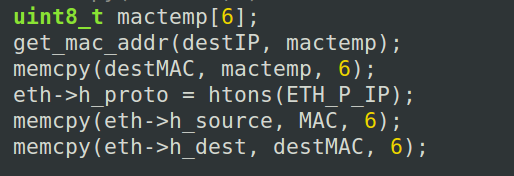
|  |  |  |
| --- | --- | --- |
| 机器名 | IP地址 | 网关 |
| Router0 | eth0: 192.168.1.2 |  |
| eth1: 192.168.2.1 |  |
| Router1 | eth0: 192.168.3.3 |  |
| eth1: 192.168.2.2 |  |
| PC1 | 192.168.1.1 | 192.168.1.2 |
| PC2 | 192.168.3.2 | 192.168.3.3 |

拓扑图：

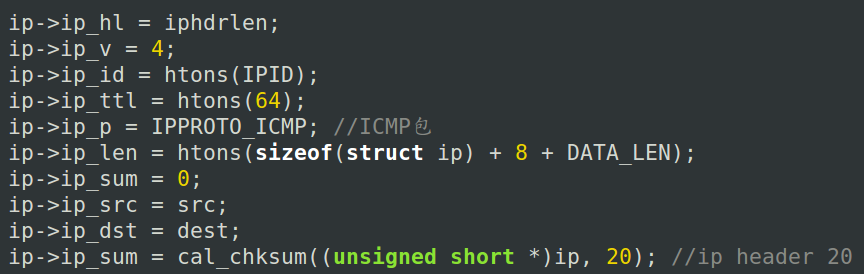


四、程序设计思路及运行流程

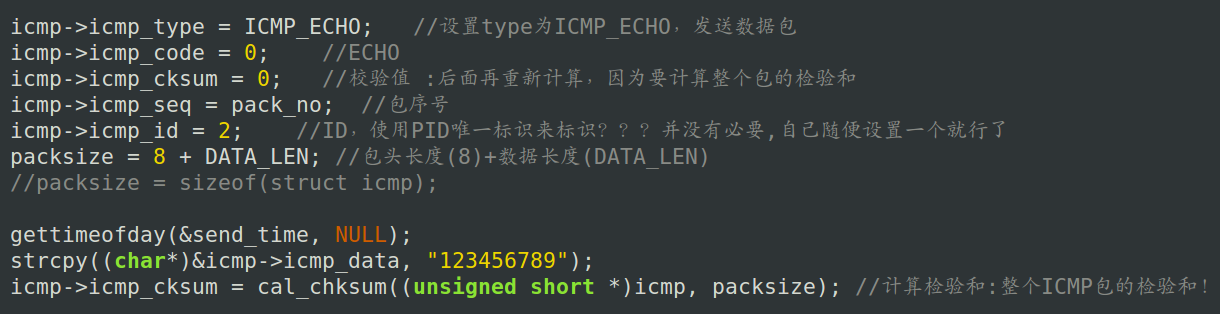
该程序要求实现PC1 ping通PC2。于是首先从PC1开始。首先在实验2实现的Ping程序的基础上进行修改。实验2的ping程序中，程序仅需填充ICMP包的字段即可，其余的IP包头字段和以太网头字段由系统自动填充。而本实验中，程序需要自行填充这两个包头：

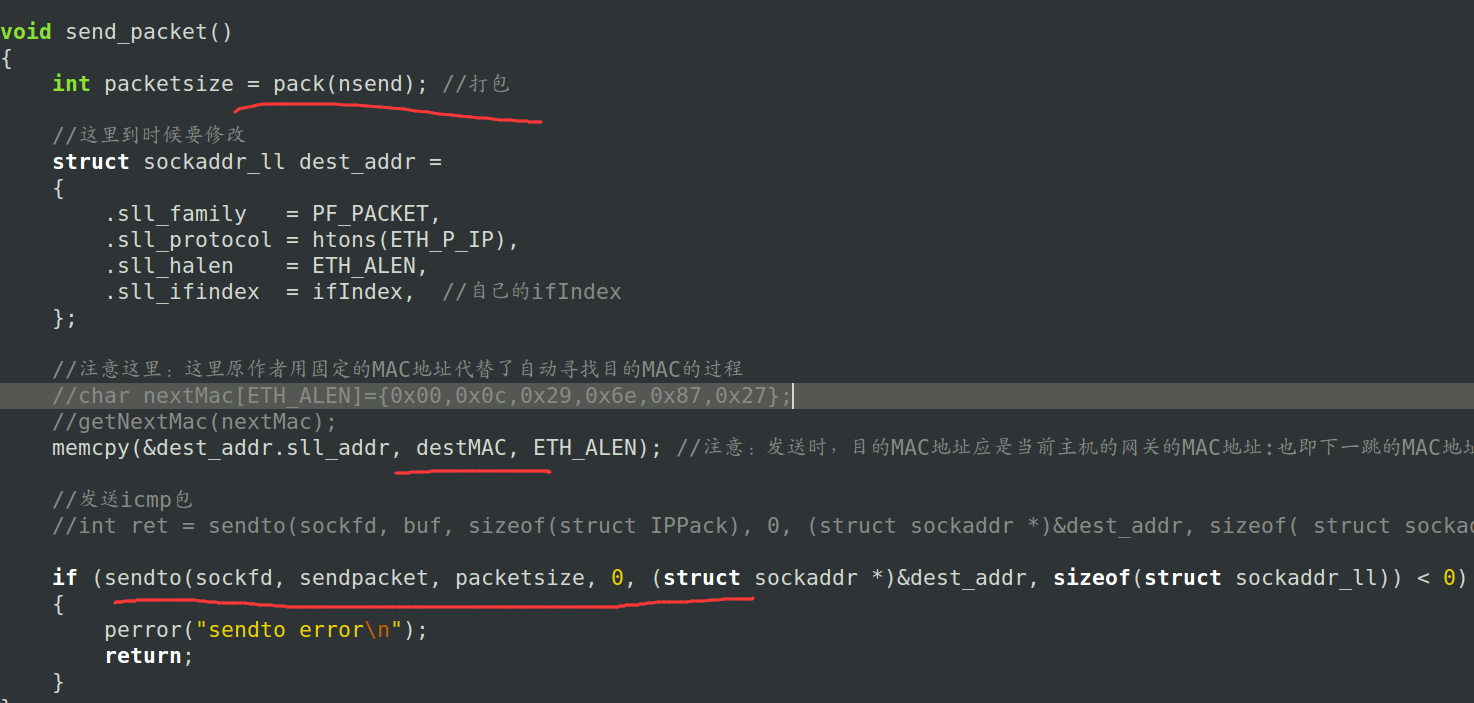


以太网包头的填充。其中get\_mac\_addr()用来获取以太网包的目的地址，就这个ICMP包而言该地址为PC1的网关的Mac地址



IP包头的填充。IP包头的长度为20；整个IP包的长度为IP包头的长度加上ICMP包的长度。图中ip->ip\_len字段的值为IP包头的长度 + 8（ICMP包头长度） + ICMP数据区长度。源IP地址为PC1的IP地址，目的IP地址为程序执行时在命令行输入的IP地址，应为PC2的IP地址

ICMP字段的填充。其中数据区的内容为“123456789”。



各个字段填好后可以开始发送包。图中destMAC为get\_mac\_addr()获取的目的MAC地址。

至此ICMP request包发送完毕。PC1上的ICMP reply包的解析函数的功能也很简单，满足以下条件即可判断该包是想要的ICMP reply包：

1. 以太网包头的协议类型字段为IP协议

2. 目的MAC地址为PC1的MAC地址，源IP地址为PC2的IP地址

3. 按ICMP包来解析后面的字段时，ICMP包长度大于8

4. ICMP包的类型是 ICMP\_ECHOREPLY且ICMP包ID为2（就是发送时填上的ID）。

满足以上条件则表明收到ping的回复包，则打印ttl等信息。

对PC2：

PC2判断收到的以太网包是否满足如下条件：

1. 协议类型为IP协议

2. 目的MAC地址为PC2的MAC地址

3. IP包的载荷字段为ICMP协议类型，且ICMP包大小>8 4. 目的IP为PC2的IP地址

5. ICMP包的类型为ICMP\_ECHO。

若满足上述全部条件，则该包为发给PC2的一个ping请求包，此时PC2的程序调换该包中的源IP地址和目的IP地址；然后重新计算IP包头的检验和；然后把ICMP包的类型字段改为ICMP\_ECHOREPLY；然后重新计算ICMP包的检验和。最后把处理好的包再次发送出去，这个发送的包的源MAC地址为PC2的MAC地址，目的MAC地址为PC2的网关的MAC地址。

对router1 和 router2：

Router程序的工作流程如下：

1. 收到一个以太网包

2. 若其类型字段为IP协议：继续下一步

3. 得到该包的目的IP地址所属的网段，在路由表中查找，看看这个网段是否存在于路由表中。若不存在则丢弃该包，否则执行下一步。

4. 设该网段在路由表route\_table[]中的route\_table[k].range中被找到。则令tmp = route\_table[k].interface, 也即要将数据包转发到这个网段需要使用的路由器接口的名称。同时拷贝出route\_table[k]. dest\_MAC到数组destmac中，此时destmac为

往该网段转发时的目的Mac地址。

5. 在设备信息表device[]中根据网卡名称tmp来查找对应的网卡的MAC地址（作为转发的包的源Mac地址）和ifindex（填写sockaddr\_ll结构类型结构体时需要用）。

6. 使用以上获得的信息来重新填写该包的以太网包头字段的源MAC地址和目的MAC地址，然后发送该包。此时转发工作完成。

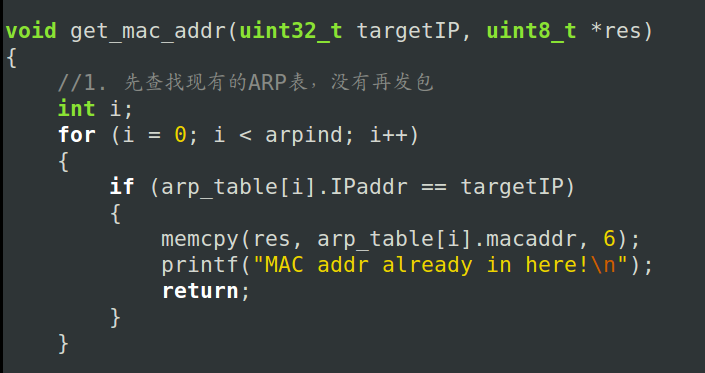
ARP部分：

对PC1：

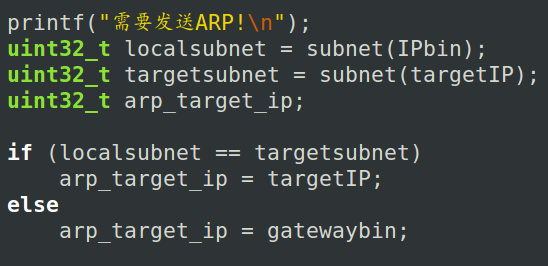
PC1发送的ICMP请求包的目的Mac地址一定为PC1的网关的MAC地址，也即router1的与PC1共处同一子网的网卡的MAC地址。在第一次发包时，PC1的程序里并不知道这一MAC地址的值，于是get\_mac\_addr()函数需要发送ARP包：

get\_mac\_addr()的实现细节：

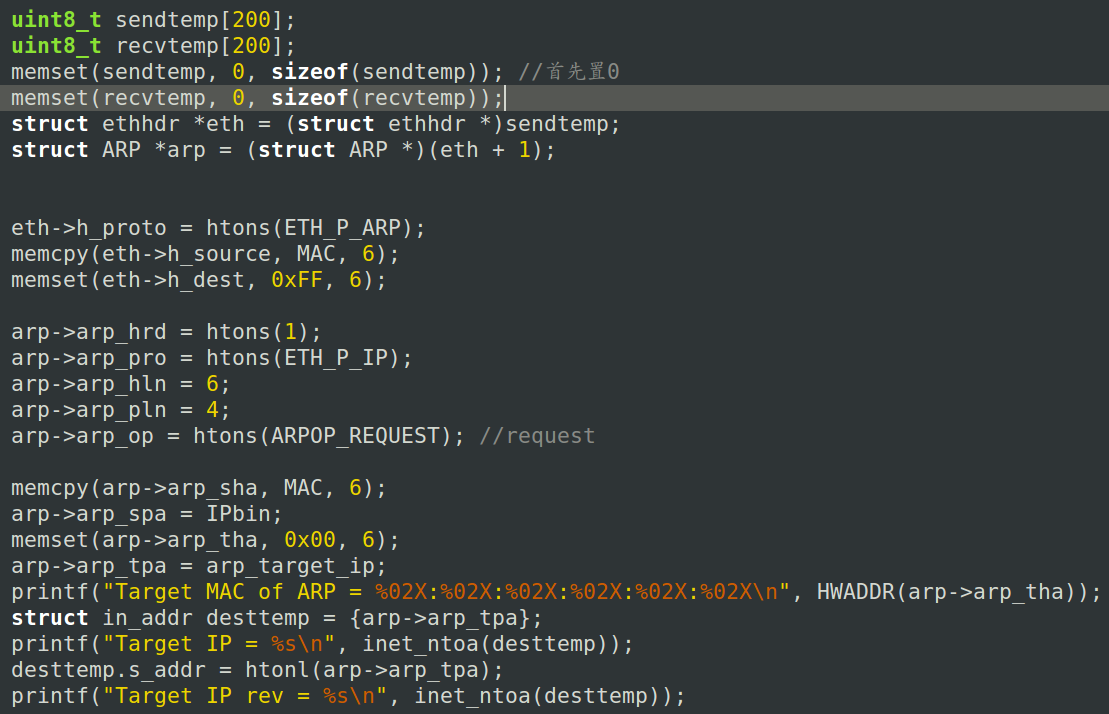
1. 在ARP表中查找targetIP是否有对应的MAC地址，有则直接取出，存入res数组中，函数结束：



2. 否则需要发送ARP包来获取targetIP对应的MAC地址：



首先判断targetIP，也即目的IP是否与本机IP处于同一子网中，若是则arp包的目的IP为targetIP，否则为PC1的网关的IP。

3.

建立好发送缓存和接受缓存数组，填充以太网包头和ARP包的字段。其中，以太网包头的目的地址为全F，为广播地址。ARP包的目的MAC字段为全0. 填完之后发送该包。

然后循环等待来接收想要的ARP回应包。每收到一个包，判断其是否满足以下条件：

1. 以太网包头的协议类型字段值为ARP包

2. ARP包的opcode为ARPOP\_REPLY

3. ARP包的目的IP为PC1的IP地址

不满足则放弃该包，继续循环等待；满足则取出该包的源MAC地址，与targetIP一起，构成PC1的ARP表的一个新表项。然后将这个包的源MAC地址copy到res数组里，函数结束。res数组内部最后就是从PC1发送的ICMP包的目的Mac地址。

对PC2：

PC2收到包后，若判断是IP包，则执行上面提到的步骤。若是ARP包；且opcode为ARPOP\_REQUEST；且其目的IP为PC2的IP，则该包为发给PC2的ARP请求包。此时取出该包的源Mac和源IP地址，在本机ARP表中看有没有一样的表项，没有则构成新的ARP表项。同时这个源Mac地址一定是PC2的网关的Mac地址，因为本实验中只有它会发ARP包给PC2，于是记这个MAC地址为destMAC.

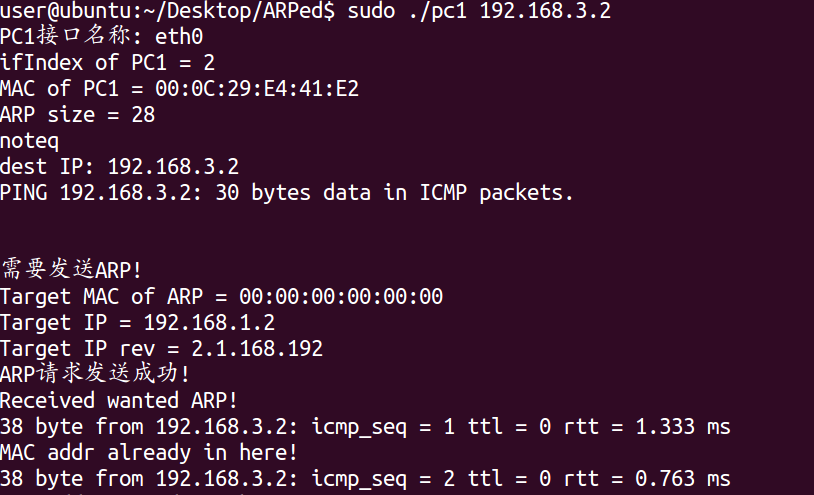
然后填写ARP回复包，重新发送。

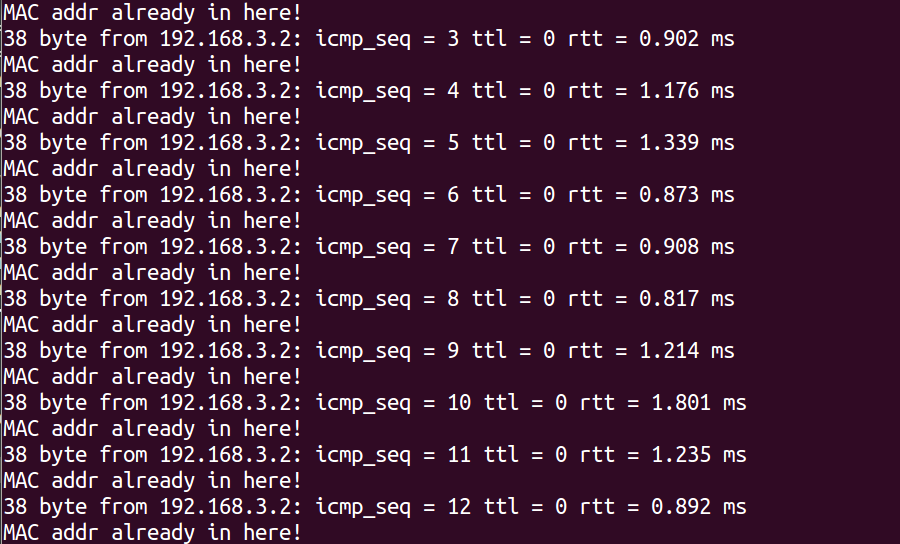
对路由器：

在前面提到的路由器工作步骤的第4步结束后，destmac中存储了转发包的目的Mac。此时检查这个地址是否每个字节的值都为0x00，若是则说明不知道真正的destmac为多少（因为路由表在初始化时没有初始化这一字段，如全为0x00）。此时填写ARP包的各个字段，ARP目的地址为路由表相应表项中的nexthopip字段。然后循环收包，判断是想要的ARP回应包，则取出其源Mac地址来更新路由表相应表项的dest\_MAC字段。此时得到了转发包的目的Mac，然后开始转发。（是的，我的实现中路由器没有ARP表，只有路由表，路由表中的dest\_MAC字段和range字段起到了ARP表的作用。实际应用中这两个表不能合二为一，但模拟实验可以这样）

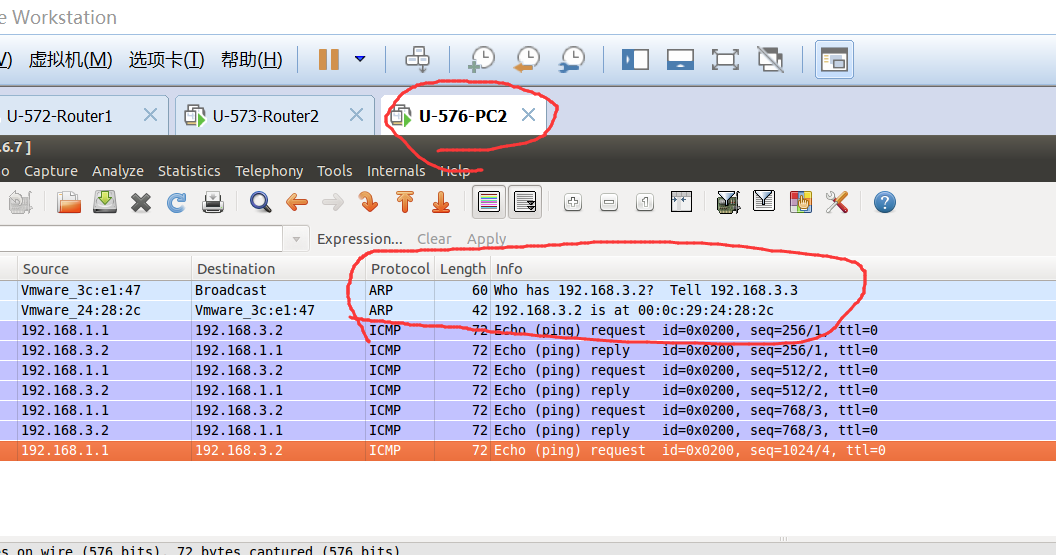
路由器也需要单独接收ARP包（而不是像上面一样，发送跟接收连在一起）：路由器在工作循环时，判断收到的是IP包则进行上面提到的处理，若是ARP请求包，且该包的目的IP是路由器的网卡的IP，则填写ARP回应包，将相应网卡的Mac地址作为源Mac，发送ARP回应包。

五、运行结果截图





可见一开始PC1需要发送ARP包；后面不需要再发送ARP包，因为已经知道了网关的MAC地址，故打印“MAC addr already in here!”



PC2收到的router2发来的ARP包。

六、参考资料

无

七、代码思考

本次实验中自定义的ARP结构体一开始会导致ARP包不正确，原因是编译器在内部实现结构体时为了满足对齐需求，在结构体内部插入缝隙。后来上网查找资料后，在程序中加入了#pragma pack(1)编译选项，使结构体紧致化，解决了问题。