计算机网络实验报告

**实验四**

**静态路由编程实现**

学号：1412200065

姓名：刘博

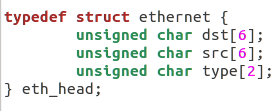
时间：2016.4.18

1. **实验目的：**

本实验主要目的设计和实现一个简单的静态路由机制，用以取代Linux资深通过ip forwarding 实现的静态路由方式，进而加深对二三层协议衔接即静态路由的理解。

1. **数据结构说明：**

以太网帧头部：

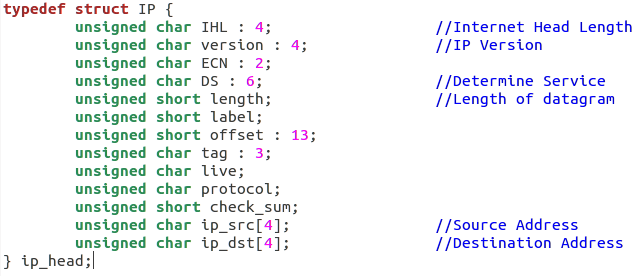


Dst：目的mac地址；

Src：源mac地址；

Type：上层协议类型；

Ip头部：



IHL：网络头部长度；

Version：ip版本号；

Live：ttl数据报生存期；

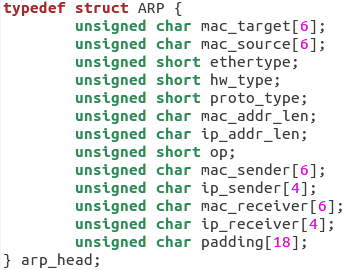
Protocol：上层协议类型；

Check\_sum：校验和；

Ip\_src：源ip地址；

Ip\_dst：目的ip地址；

ARP头部：



Mac\_target：目标mac地址；

Mac\_source：源mac地址；

Hw\_type：硬件类型（网卡编号）；

Proto\_type：协议类型；

Mac\_addr\_len：mac地址长度；

Ip\_addr\_len：ip地址长度；

Op：操作类型；

Mac\_sender：mac发送者地址；

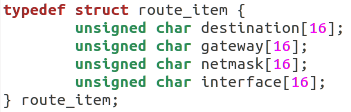
Ip\_sender：ip发送者地址；

Mac\_receiver：mac接受者地址；

Ip\_receiver：ip接受者地址；

Padding：附加段；

路由表项：



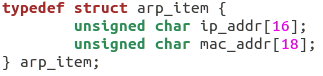
Destination：目的ip地址；

Gateway：网关地址；

Netmask：网络掩码；

Interface：网络接口（网卡名）；

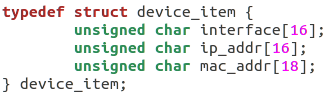
Arp表项：



Ip\_addr：ip地址；

Mac\_addr：mac地址；

Device表项：



Interface：网卡接口（网卡名）；

Ip\_addr：网卡ip地址；

Mac\_addr：网卡mac地址；

1. **配置文件说明：**

ip配置：

router1：

ifconfig eth0 192.168.0.1 netmask 255.255.255.0

ifconfig eth1 192.168.1.1 netmask 255.255.255.0

router2：

ifconfig eth0 192.168.1.2 netmask 255.255.255.0

ifconfig eth1 192.168.2.1 netmask 255.255.255.0

pc1：

ifconfig eth0 192.168.0.2 netmask 255.255.255.0

pc2：

ifconfig eth0 192.168.2.2 netmask 255.255.255.0

route配置：

router1：

ip route add 192.168.2.0/24 via 192.168.1.2

echo 0 > /proc/sys/net/ipv4/ip\_forward

router2：

ip route add 192.168.0.0/24 via 192.168.1.1

echo 1 > /proc/sys/net/ipv4/ip\_forward

pc1：

route add default gw 192.168.0.1

pc2：

route add default gw 192.168.2.1

configuration配置：

route\_table.txt ：

192.168.1.0 \* 255.255.255.0 eth1

192.168.0.0 \* 255.255.255.0 eth0

192.168.2.0 192.168.1.2 255.255.255.0 eth1

device\_table.txt：

eth0 192.168.0.1 00:0c:29:77:9e:74

eth1 192.168.1.1 00:0c:29:77:9e:7e

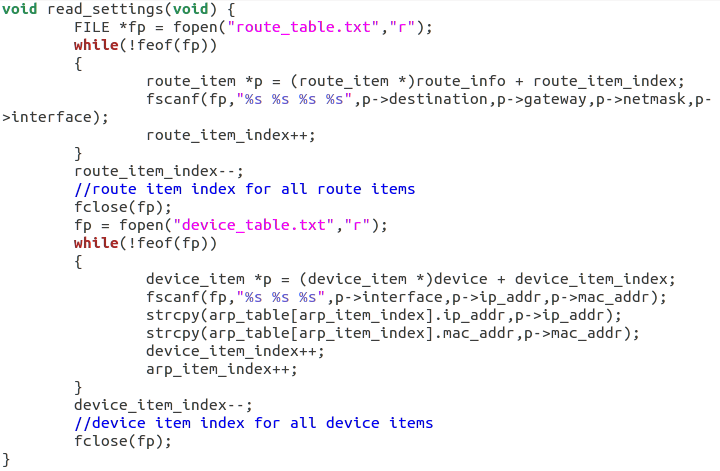
1. **程序设计思路及运行流程：**

设计思路：

设计思路基于静态路由流程进行设计，首先需要理解静态路由的转发流程，然后就可以进行设计；

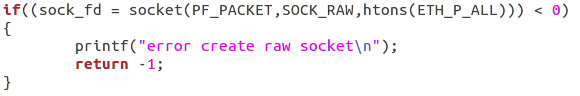
首先路由器进行初始化，直接将路由表和设备表进行读取和赋值：





Read\_settings函数进行初始化，包括路由表，设备表，arp表的初始化；

初始化完成后，创建套接字，监听路由器的所有接口：

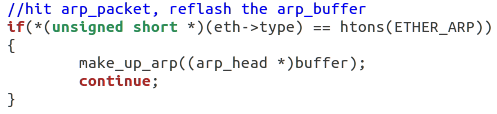


每当收到一个数据包是，路由器需要根据其以太网的地址来确定该包是否属于arp包，如果是arp包，则进行arp表的更新，如果不是，则进行ip判断，如果是ip包，则进行ip转发处理，否则丢弃该包；

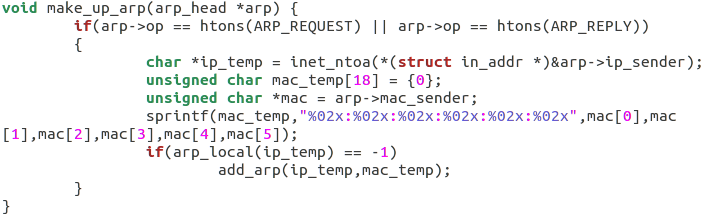
下面分别介绍arp和ip处理：

1. arp处理：

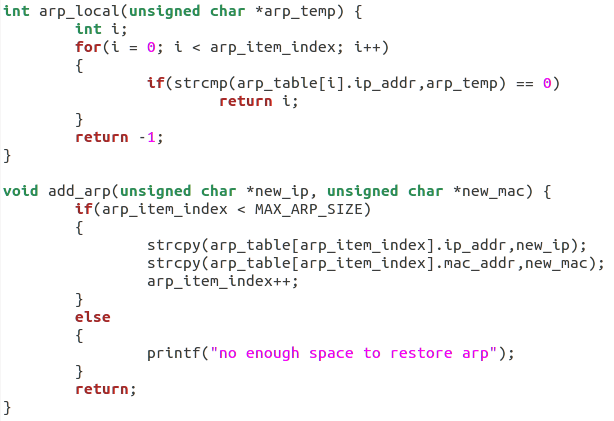
若接收到的是arp包，路由器首先检查该arp中的对应规则是否在自己的arp表中存在，如果是，则不作进一步处理，如果不是，则将新的规则加入自己当前的arp表项中；



（抓到了arp数据包）

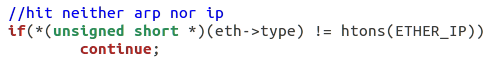


（读取arp包并进行判断）



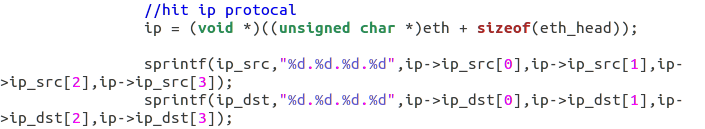
（判断arp是否存在，如果不是，则加入当前的arp表中）

1. 不是arp包和ip包：丢弃！

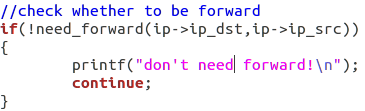


1. Ip包处理：

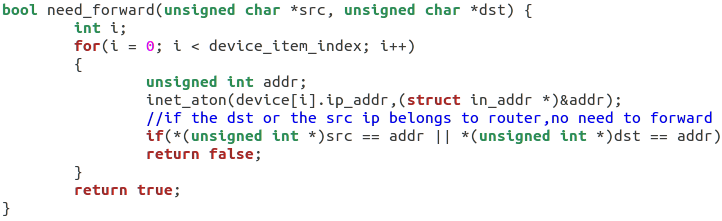
当路由器抓获一个ip包时，路由器进行ip地址读取：



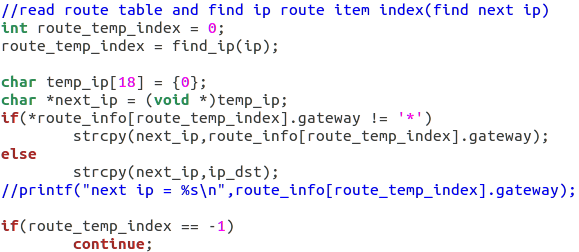
根据ip地址，判断该包是否是发往本路由器的，如果是发往自身的，则不进行转发，直接回复：



（判断是否属于本路由器，如果是，则不再进行转发）

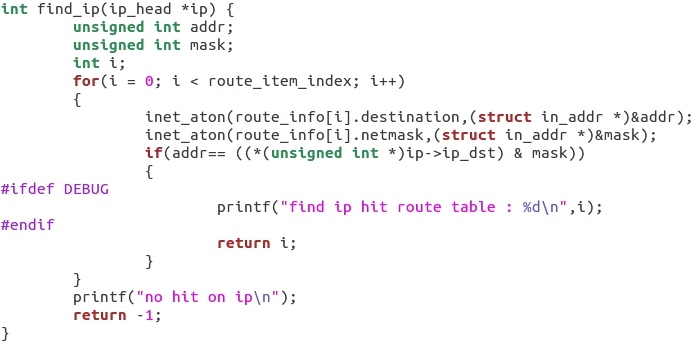


如果该ip包不是发往本路由器的，那么就需要进行转发，首先根据route表找到当前目标ip地址的下一跳的ip地址：



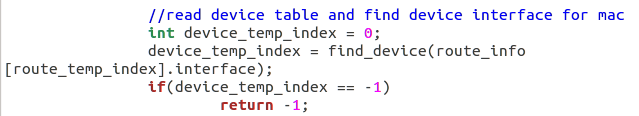
（如果是gateway是\*表示网关为同一个子网，可以直接发送，无需地址转换）

如果route表找不到该目的ip地址，则显示无法连接（unreachable）

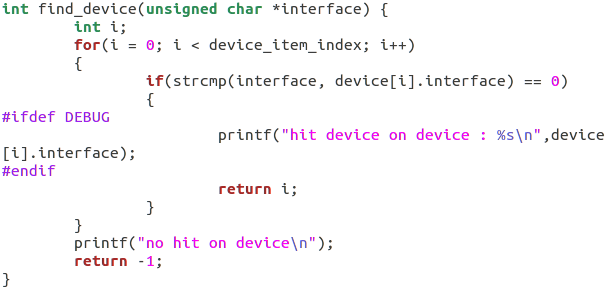


（注意此处需要用到掩码）

找到下一跳的ip地址后，根据arp表找到下一跳的网卡接口地址：



（找到对应的device接口）

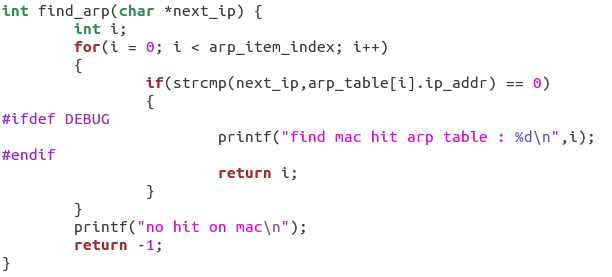


找到网卡接口后，新的以太网帧（转发包）的源地址就确定了，接下来需要确定目的地址：

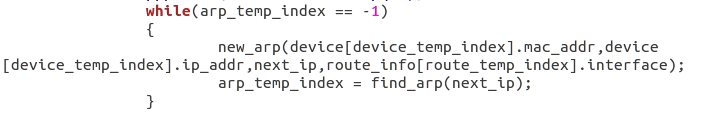
根据目的ip地址进行arp转换寻找mac地址：



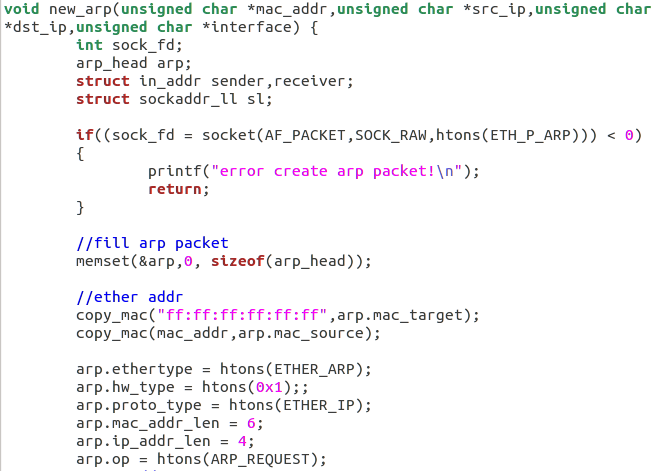
（寻找当前arp表中是否存在该ip地址）

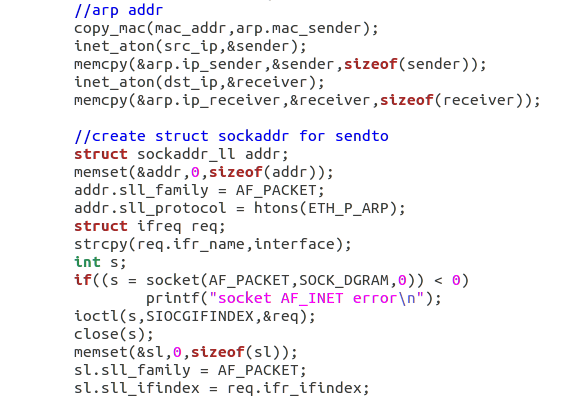


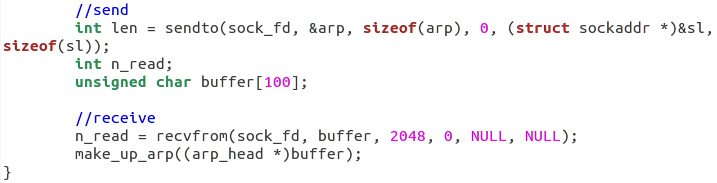
如果当前arp表中没有对应的ip地址存在，则进行广播发送arp包询问：



其中new\_arp（）函数就是用来发送arp询问请求的：





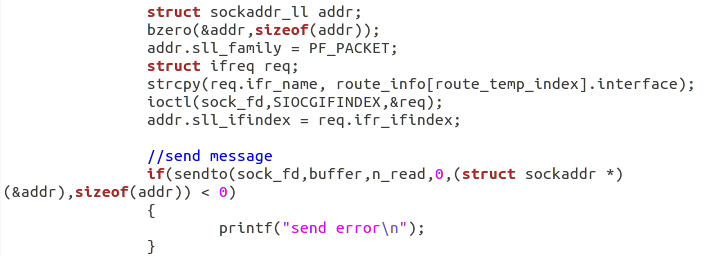


首先发送arp请求的地址是广播地址（ff：ff：ff：ff：ff：ff），然后将对应的以太网头部协议，地址，网卡设备号依次填好，用sendto（）函数进行发送；收到的包进行arp分析；并将其加入到arp表中；

确定了下一跳的mac地址，就可以将以太网帧mac地址进行修改，成为转发包：



最后进行发送整个以太网帧即可：



\*归纳来讲整个运行的流程如下：

1. 路由程序启动

2. 读取配置文件并初始化数据结构

3. 监听所有的网口

4. 捕获来自网口的数据包

5. 判断数据包类型

ARP报文，将发送方的信息登记入ARP table，转到步骤3

IP报文，转到步骤6

其他，丢弃该数据包，转到步骤3

6. 判断是否需要转发，如果不需要，转到步骤3

7. 在路由表中查询下一跳的IP地址，如果无此表项，则网络不可达，丢弃该数据

包，转到步骤3

8. 根据下一跳的发送接口查询设备表，获得发送网卡的MAC地址，无此表项则程序出错

9. 根据下一跳IP地址查询ARP表

无此表项则发送ARP请求，并将ARP信息登记入ARP表，转到步骤9

有此表项则获取下一跳MAC地址，转到步骤10

10. 将数据包中以太层的源MAC地址和目的MAC地址改为路由器发送设备的MAC地址和下一跳的MAC地址

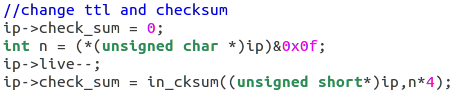
11. 将IP报头的ttl减一

12. 重新计算IP报头的checksum

13. 发送该数据包

14. 转发完成，转到步骤3

其中计算checksum以及ttl改变如下：



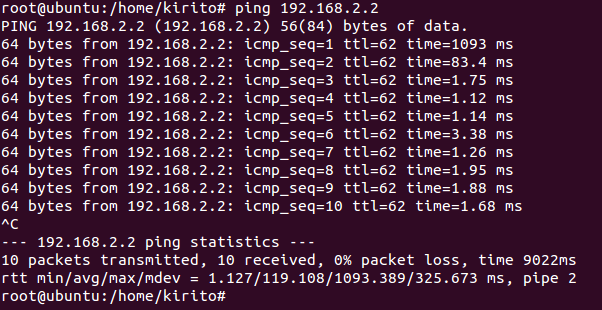
至此路由转发功能完成；

**5. 运行结果截图：**

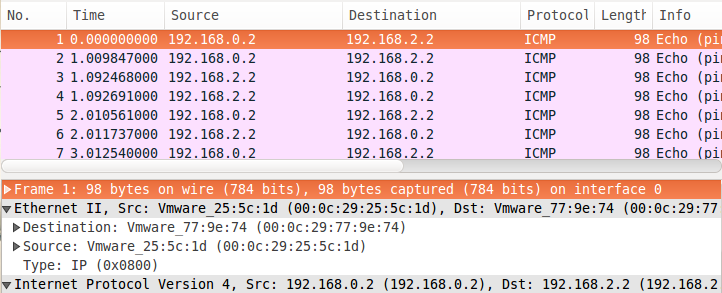
打开router1 的 ip\_forward时：



（pc1 ping pc2）

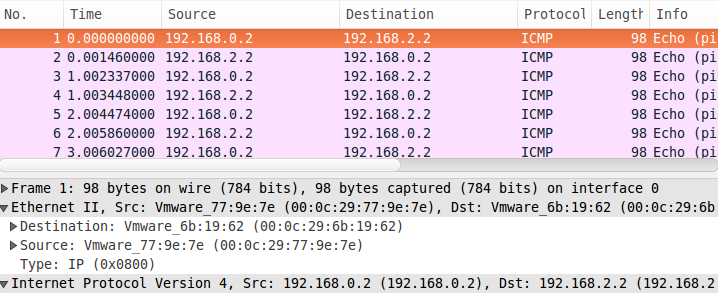


Router1上对eth0进行wireshark抓包：



（可以看到当打开路由转发功能时，对eth0而言，其以太网源地址是pc1的mac地址，目的地址是eth0的mac地址）

Router1上对eth1进行wireshark抓包：



（可以看到当打开路由转发功能时，对eth1而言，其以太网源地址是router1的eth1地址，目的地址是pc2的mac地址）

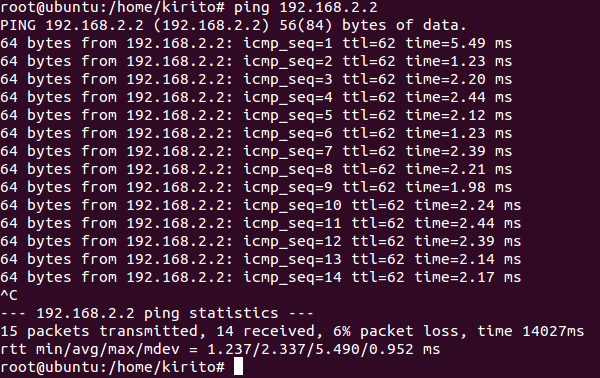
此时pc1和pc2可以互相ping通；

下面关闭router1的ip\_forward：

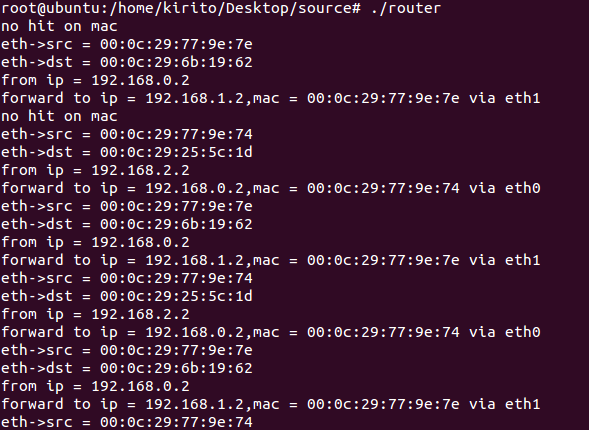


打开编写好的程序（router）：

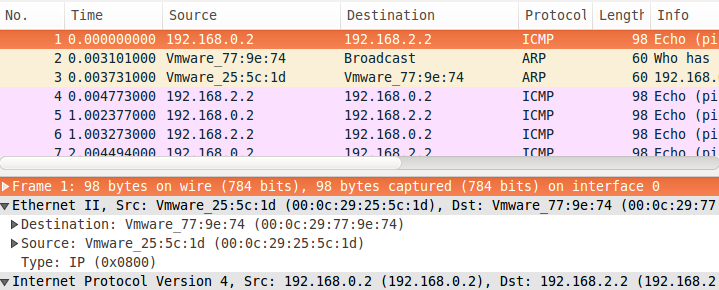
（pc1 ping pc2）



程序显示界面：

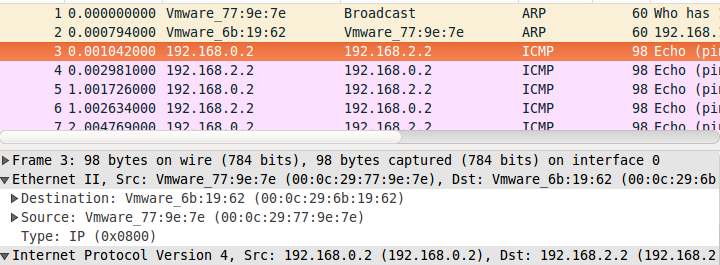


Router1上对eth0进行wireshark抓包：



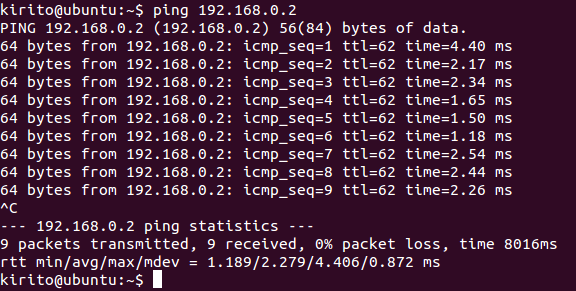
（可以看到当关闭路由转发功能，打开程序时，对eth0而言，其以太网源地址是pc1的mac地址，目的地址是eth0的mac地址，并且有arp包进行询问，可见路由转发正确）

Router1上对eth1进行wireshark抓包：

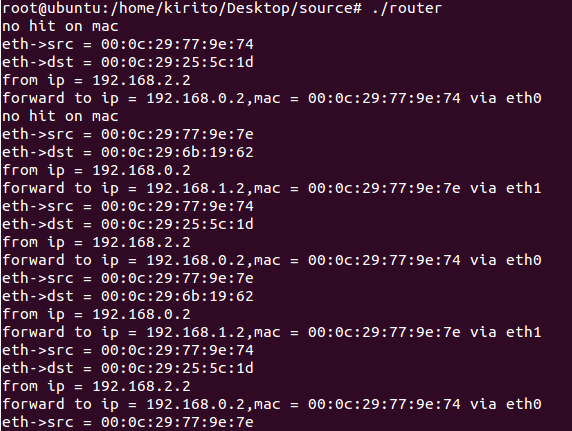


（可以看到当关闭路由转发功能，打开程序时，对eth1而言，其以太网源地址是router1的eth1地址，目的地址是pc2的mac地址，并且首先由arp包进行询问mac地址，可见路由转发正确）

反向（pc2 ping pc1）



程序显示如下：



可见反向任然能够ping通，可见路由转发没有问题；

**6. 相关参考资料：**

（百度，google等）；

**7. 对比样例程序：**

本次实验无任何样例程序；

**8. 代码个人创新及思考：**

所有代码均为原创，纯手打，首先该代码很好地模仿了路由功能，其次还可以在不同时候识别不同的网关形式（\*）；

思考：

1. 开始时使用ping后发现没有显示，用wireshark抓包后发现对于request和reply包的网卡mac地址不同，由于没有填写正确的mac地址导致ping收不到回复包，所以ping一直没有显示，后来改成正确的路由地址后正确；
2. 开始时初始化设备时，没有将设备的ip和mac直接填写到arp表中，导致arp表项不全，同时无法询问（wireshark上显示自己问自己，死循环），所以导致转发无法实现。
3. Ping其实是一个两端互发的过程，所以路由表对于两侧的路由规则都必须了解，否则会只有request没有reply（亲身体会）