一、数据结构

（1）设备表，用于存放每个设备的序号、ip 地址、mac 地址。其中 ip 地址、mac 地址使用大端方式存放

```
struct device_item{
        int index;
        char ip_addr[4];
        char mac_addr[ETH_ALEN];
};
struct device_item device[MAX_DEVICE];
```

（2）路由表。存放目的 ip 对应的网关、掩码、发送设备序号，由于没实现 arp，故还需存放网关对应的 mac 地址。

```
struct route_item{
        char destination[4];
        char gateway[4];
        char netmask[4];
        char mac_addr[6];
        char interface;
};
struct route_item route_info[MAX_ROUTE_INFO];
```

（3）Header 数据结构，用于发送、解析 icmp 包

```
#include <netinet/ip.h>
struct ip *ip_header;
#include <netinet/ip_icmp.h>
struct icmphdr *icmp_header;
```

二、程序运行流程

（1）icmp 发送方主机：

step1：初始化设备表、路由表

step2：发送 icmp 数据包

step3：等待 icmp 回应

（2）路由器：

step1：初始化设备表、路由表

step2：接收报文，并判断是否转发。重复

```
while(1){
        n_read = recvfrom(sock_fd,buffer,2048,0,(struct sockaddr *)&addr,&addr_len);
        if(n_read < 42){
                printf("error when recv msg\n");
                return -1;
        }

        eth_head = buffer;
        type = ((eth_head+12)[0]<<8)+(eth_head+13)[0];
        p= eth_head;

        if (type ==0x0800){
                ip_head = eth_head+14;
                char *ip = ip_head+12;
                proto = (ip_head +9)[0];
                p = ip_head + 12;
                if(proto==IPPROTO_ICMP && addr.sll_pkttype==PACKET_HOST && strncmp(ip,ip+4,4)!=0)
                {
                        printf("%02hhx.%02hhx.%02hhx.%02hhx ==> %02hhx.%02hhx.%02hhx.%02hhx\n",ip[0],ip[1],ip[2],ip[3],ip[4],ip[5],ip[6],ip[7]);
                        printf("ICMP\n",proto);Routing(buffer+14,ip,ip+4);
                }

        }
}
```

死循环不断接收数据包，对于类型为 PACKET_HOST 的 icmp 数据包，调用 Routing 函数进行转发。

Routing 函数 ：

```
void Routing(char *buffer, char *ip_src, char *ip_dest){

        int i=1;
        int flag = 0;
        for(;i<MAX_ROUTE_INFO;i++){
                if(((ip_dest[0] & route_info[i].netmask[0]) == route_info[i].destination[0]) &&
                   (ip_dest[1] & route_info[i].netmask[1]) == route_info[i].destination[1] &&
                   (ip_dest[2] & route_info[i].netmask[2]) == route_info[i].destination[2] &&
                   (ip_dest[3] & route_info[i].netmask[3]) == route_info[i].destination[3] ){
                        flag = 1;
                        break;
                }
        }
        if(flag==0) i = 0;

        int sockfd= socket(AF_PACKET, SOCK_DGRAM, htons(ETH_P_IP));
        if(sockfd ==-1)
        {
                perror("socket error");
                exit(1);
        }
        struct sockaddr_ll dest_addr;
        dest_addr.sll_family = AF_PACKET;
        dest_addr.sll_protocol = htons(ETH_P_IP);
        dest_addr.sll_halen = ETH_ALEN;
        dest_addr.sll_ifindex = route_info[i].interface;
        memcpy(&dest_addr.sll_addr,route_info[i].mac_addr,6);

        if(sendto(sockfd,buffer,20+64,0,(struct sockaddr *)&dest_addr,sizeof(dest_addr)) < 0)
        {
                perror("sendto error");
                return ;
        }

}
```

查询路由表，将数据包中的目的 ip 与掩码按位与后，和路由表中的目的 ip 比较，得到网关 mac 地址和发送设备序号

route_info[0]中存放默认网关，查询路由表从 route_info[1]开始，如果未找到匹配项则发往默认网关

（3）接收方主机：

step1：初始化设备表、路由表

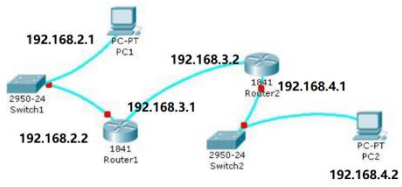
step2：接收 icmp 请求

step3：发送 icmp 应答

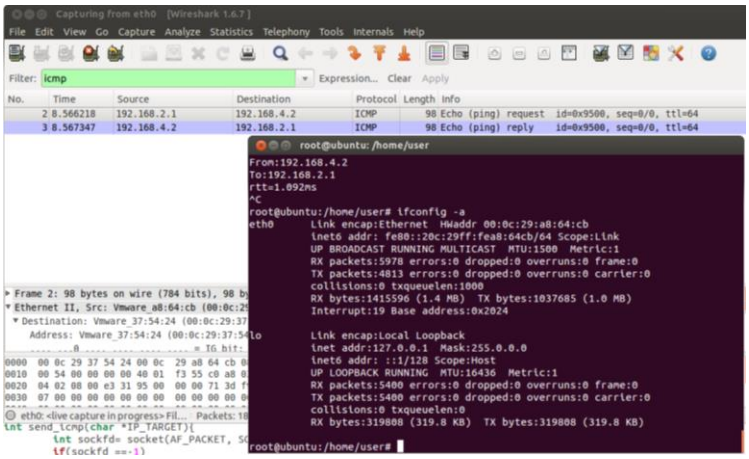
相应功能在实验二中已经实现。

三、运行结果

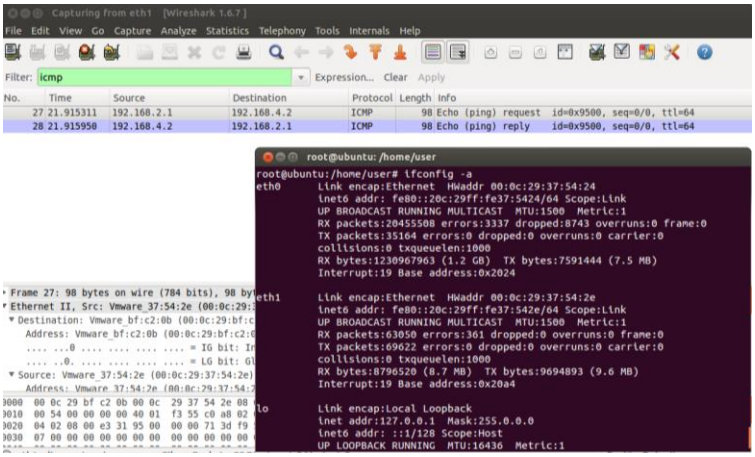
网络拓扑:



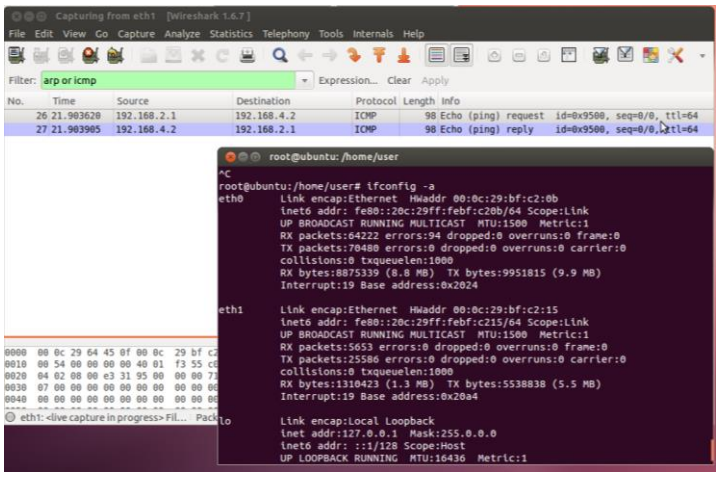
发送方主机:



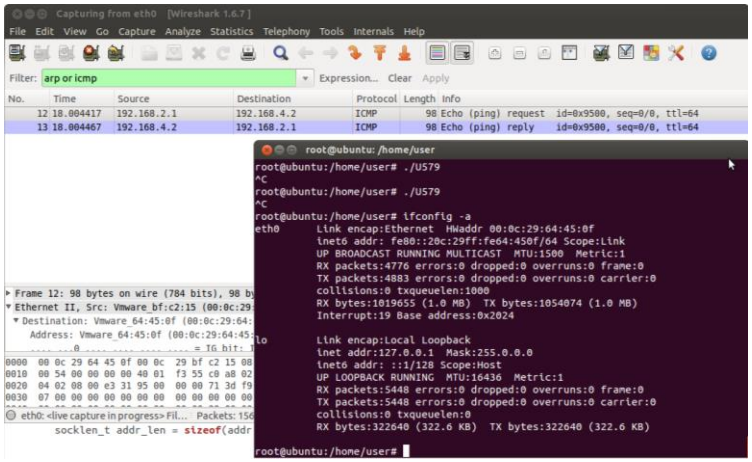
路由器 1:



路由器 2:


Capturing from eth1 [Wireshark 1.6.7]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: arp or icmp
Expression... Clear Apply
No. Time Source Destination Protocol Length Info
26 21.903620 192.168.2.1 192.168.4.2 ICMP 98 Echo (ping) request id=0x9500, seq=0/0, ttl=64
27 21.903905 192.168.4.2 192.168.2.1 ICMP 98 Echo (ping) reply id=0x9500, seq=0/0, ttl=64
root@ubuntu: /home/user
^C
root@ubuntu:/home/user# ifconfig -a
eth0 Link encap:Ethernet HWaddr 00:0c:29:bf:c2:0b
inet6 addr: fe80::20c:29ff:febf:c20b/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:64222 errors:94 dropped:0 overruns:0 frame:0
TX packets:70480 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:8875339 (8.8 MB) TX bytes:9951815 (9.9 MB)
Interrupt:19 Base address:0x2024
eth1 Link encap:Ethernet HWaddr 00:0c:29:bf:c2:15
inet6 addr: fe80::20c:29ff:febf:c215/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:5653 errors:0 dropped:0 overruns:0 frame:0
TX packets:25586 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:1310423 (1.3 MB) TX bytes:5538838 (5.5 MB)
Interrupt:19 Base address:0x20a4
lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:16436 Metric:1

接收方主机:


Capturing from eth0 [Wireshark 1.6.7]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: arp or icmp
Expression... Clear Apply
No. Time Source Destination Protocol Length Info
12 18.004417 192.168.2.1 192.168.4.2 ICMP 98 Echo (ping) request id=0x9500, seq=0/0, ttl=64
13 18.004467 192.168.4.2 192.168.2.1 ICMP 98 Echo (ping) reply id=0x9500, seq=0/0, ttl=64
root@ubuntu: /home/user
root@ubuntu:/home/user# ./U579
^C
root@ubuntu:/home/user# ./U579
^C
root@ubuntu:/home/user# ifconfig -a
eth0 Link encap:Ethernet HWaddr 00:0c:29:64:45:0f
inet6 addr: fe80::20c:29ff:fe64:450f/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:4776 errors:0 dropped:0 overruns:0 frame:0
TX packets:4883 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:1019655 (1.0 MB) TX bytes:1054074 (1.0 MB)
Interrupt:19 Base address:0x2024
lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:5448 errors:0 dropped:0 overruns:0 frame:0
TX packets:5448 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:322640 (322.6 KB) TX bytes:322640 (322.6 KB)
root@ubuntu:/home/user#