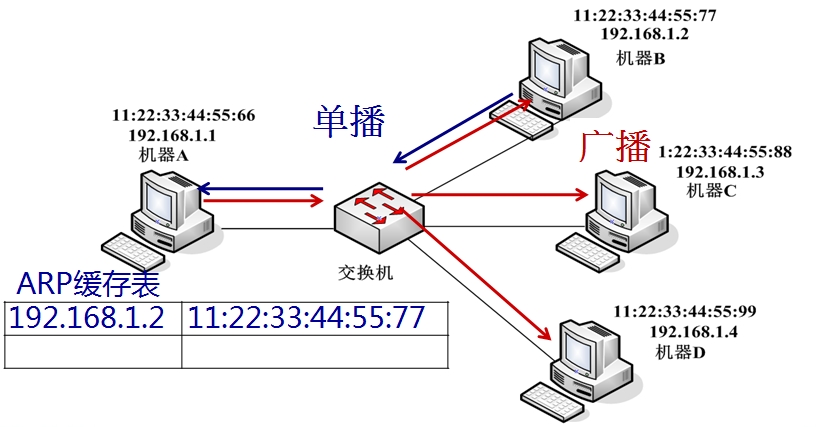
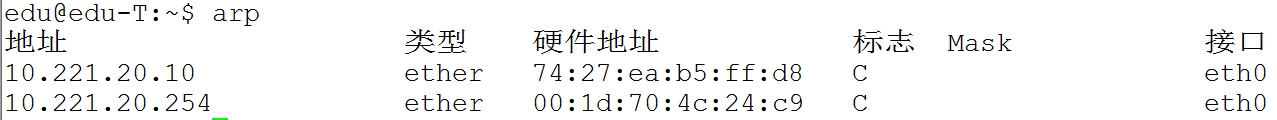
如果 A （192.168.1.1 ）向 B （192.168.1.2 ）发送一个数据包，那么需要的条件有 ip、port、使用的协议（TCP/UDP）之外还需要 MAC 地址，因为在以太网数据包中 MAC 地址是必须要有的。那么怎样才能知道对方的 MAC 地址？答案是：它通过 ARP 协议来获取对方的 MAC 地址。

ARP（Address Resolution Protocol，地址解析协议），是 TCP/IP 协议族中的一个，主要用于查询指定 ip 所对应的的 MAC（通过 ip 找 MAC）。  
  
请求方使用广播来发送请求，应答方使用单播来回送数据。收到返回消息后将该 IP 地址和物理地址存入本机 ARP 缓存中并保留一定时间，下次请求时直接查询 ARP 缓存以节约资源。

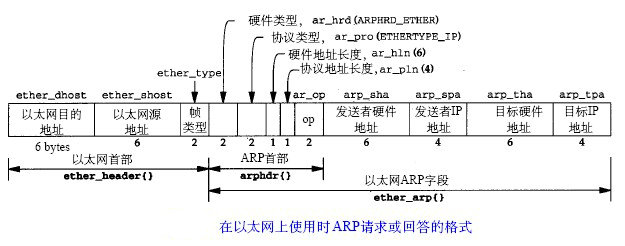
以机器 A 获取机器 B 的 MAC 为例，A 广播发送一个 ARP 请求包，和 A 同在一个局域网的主机都会收到这个请求包，每个机器都会比较自己的 ip 和请求包的目的 ip 是不是一样的，如果不一样，就丢弃这个请求包，结果，只有 B 机器符合条件，B 机器单独给 A 发送 ARP 应答包，应答包带上了 B 的 ip 所对应的 MAC 地址，当 A 收到这个应答包后，就把 B 的 ip 以及其对应的 MAC 地址存入本机 ARP 缓存中。

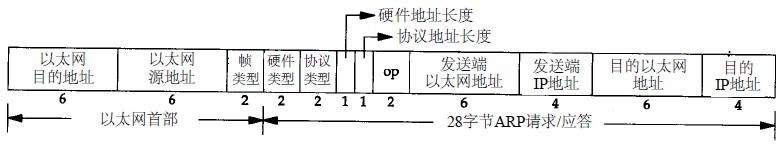


在 Linux 查看 ARP 缓存表：arp



ARP头部

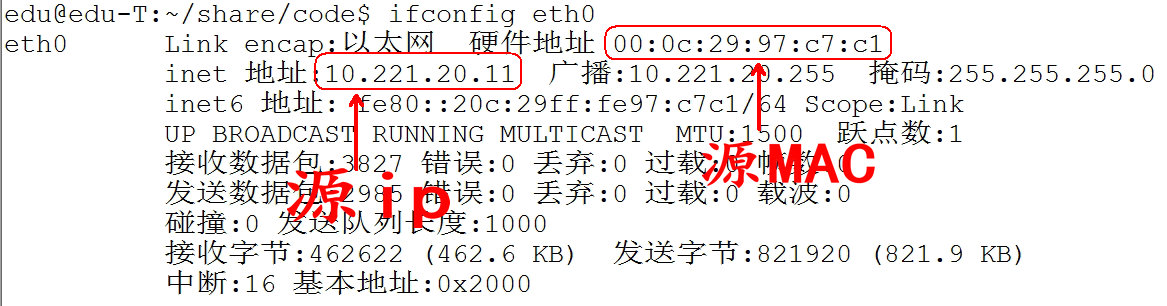




1、Dest MAC：目的 MAC 地址  
2、Src MAC：源 MAC 地址  
3、帧类型：0x0806  
4、硬件类型：1（以太网）  
5、协议类型：0x0800（IP地址）  
6、硬件地址长度：6  
7、协议地址长度：4  
8、OP：1（ARP请求），2（ARP应答），3（RARP请求），4（RARP应答）

接下来这个例子为，虚拟机（ubuntu）获取 PC 机的 MAC 地址：

先查看 ubuntu 的 ip 和 MAC 地址：



完整代码如下：

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <net/if.h>             //struct ifreq

#include <sys/ioctl.h>          //ioctl、SIOCGIFADDR

#include <sys/socket.h>

#include <netinet/ether.h>      //ETH\_P\_ALL

#include <netpacket/packet.h>   //struct sockaddr\_ll

#include <netinet/in.h>

int main(int argc,char \*argv[])

{

    //1.创建通信用的原始套接字

    int sock\_raw\_fd = socket(PF\_PACKET, SOCK\_RAW, htons(ETH\_P\_ALL));

    //2. 根据各种协议首部格式构建发送数据报

    unsigned char send\_msg[1024] = {

        //--------------组MAC--------14------

        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, //dst\_mac: FF:FF:FF:FF:FF:FF

        0x00, 0x0c, 0x29, 0x97, 0xc7,0xc1, //src\_mac: 00:0c:29:97:c7:c1

        0x08, 0x06,                         //类型：0x0806 ARP协议

        //--------------组ARP--------28-----

        0x00, 0x01, 0x08, 0x00,             //硬件类型1(以太网地址),协议类型0x0800(IP)

        0x06, 0x04, 0x00, 0x01,             //硬件、协议地址分别是6、4，op:(1：arp请求，2：arp应答)

        0x00, 0x0c, 0x29, 0x97, 0xc7,0xc1,  //发送端的MAC地址

        10,  221,  0, 11,               //发送端的IP地址

        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, //目的MAC地址（由于要获取对方的MAC,所以目的MAC置零）

        10, 221, 20, 10             //目的IP地址

    };

    //3.数据初始化

    struct sockaddr\_ll sll;                 //原始套接字地址结构

    struct ifreq req;                   //网络接口地址

    strncpy(req.ifr\_name, "eth0", IFNAMSIZ);    //指定网卡名称

    //4.将网络接口赋值给原始套接字地址结构

    ioctl(sock\_raw\_fd, SIOCGIFINDEX, &req);

    bzero(&sll, sizeof(sll));

    sll.sll\_ifindex = req.ifr\_ifindex;

    //5. 发送 ARP 请求包

    int len = sendto(sock\_raw\_fd, send\_msg, 42, 0 , (struct sockaddr \*)&sll, sizeof(sll));

    if(len == -1)

    {

        perror("sendto");

    }

    //6.接收对方的ARP应答

    unsigned char recv\_msg[1024] = {0};

    recvfrom(sock\_raw\_fd, recv\_msg, sizeof(recv\_msg), 0, NULL, NULL);

    if(recv\_msg[21] == 2)           //ARP应答

    {

        char resp\_mac[18] = "";     //arp响应的MAC

        char resp\_ip[16] = "";      //arp响应的IP

        sprintf(resp\_mac, "%02x:%02x:%02x:%02x:%02x:%02x", \

        recv\_msg[22],recv\_msg[23],recv\_msg[24],recv\_msg[25],recv\_msg[26],recv\_msg[27]);

        sprintf(resp\_ip, "%d.%d.%d.%d", recv\_msg[28], recv\_msg[29], recv\_msg[30], recv\_msg[31]);

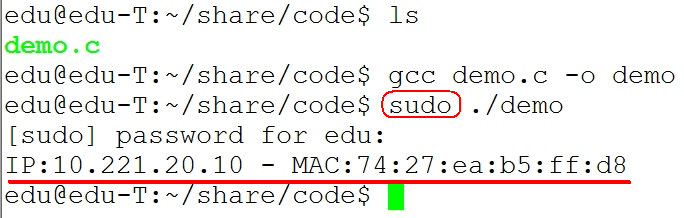
        printf("IP:%s - MAC:%s\n",resp\_ip, resp\_mac);

    }

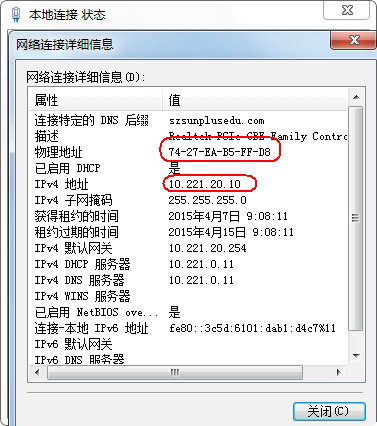
    return 0;

}

程序运行结果如下：



查看 PC 的网卡信息：



下面的例子能够获取指定网段所有机器的 MAC 地址：

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <net/if.h>             //struct ifreq

#include <sys/ioctl.h>          //ioctl、SIOCGIFADDR

#include <sys/socket.h>

#include <netinet/ether.h>      //ETH\_P\_ALL

#include <netpacket/packet.h>   //struct sockaddr\_ll

#include <pthread.h>

#include <netinet/in.h>

void \*send\_arp\_ask(void \*arg);

int main(int argc,char \*argv[])

{

    //1.创建通信用的原始套接字

    int sock\_raw\_fd = socket(PF\_PACKET, SOCK\_RAW, htons(ETH\_P\_ALL));

    //2.创建发送线程

    pthread\_t tid;

    pthread\_create(&tid, NULL, (void \*)send\_arp\_ask, (void \*)sock\_raw\_fd);

    while(1)

    {

        //3.接收对方的ARP应答

        unsigned char recv\_msg[1024] = "";

        recvfrom(sock\_raw\_fd, recv\_msg, sizeof(recv\_msg), 0, NULL, NULL);

        if(recv\_msg[21] == 2)           //ARP应答

        {

            char resp\_mac[18] = "";     //arp响应的MAC

            char resp\_ip[16] = "";      //arp响应的IP

            sprintf(resp\_mac, "%02x:%02x:%02x:%02x:%02x:%02x", \

            recv\_msg[22],recv\_msg[23],recv\_msg[24],recv\_msg[25],recv\_msg[26],recv\_msg[27]);

            sprintf(resp\_ip, "%d.%d.%d.%d", recv\_msg[28], recv\_msg[29], recv\_msg[30], recv\_msg[31]);

            printf("IP:%s - MAC:%s\n",resp\_ip, resp\_mac);

        }

    }

    return 0;

}

void \*send\_arp\_ask(void \*arg)

{

    int i = 0;

    int sock\_raw\_fd = (int)arg;

    //1.根据各种协议首部格式构建发送数据报

    unsigned char send\_msg[1024] = {

        //--------------组MAC--------14------

        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, //dst\_mac: FF:FF:FF:FF:FF:FF

        0x00, 0x0c, 0x29, 0x75, 0xa6, 0x51, //src\_mac: 00:0c:29:75:a6:51

        0x08, 0x06,                         //类型：0x0806 ARP协议

        //--------------组ARP--------28-----

        0x00, 0x01, 0x08, 0x00,             //硬件类型1(以太网地址),协议类型0x0800(IP)

        0x06, 0x04, 0x00, 0x01,             //硬件、协议地址分别是6、4，op:(1：arp请求，2：arp应答)

        0x00, 0x0c, 0x29, 0x75, 0xa6, 0x51, //发送端的MAC地址

        172,  20,   226,  12,               //发送端的IP地址

        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, //目的MAC地址（由于要获取对方的MAC,所以目的MAC置零）

        172,  20,   226,  11                //目的IP地址

    };

    //2.数据初始化

    struct sockaddr\_ll sll;                 //原始套接字地址结构

    struct ifreq req;                   //网络接口地址

    strncpy(req.ifr\_name, "eth0", IFNAMSIZ);    //指定网卡名称

    //3.将网络接口赋值给原始套接字地址结构

    ioctl(sock\_raw\_fd, SIOCGIFINDEX, &req);

    bzero(&sll, sizeof(sll));

    sll.sll\_ifindex = req.ifr\_ifindex;

    //4.本地机的IP

    if(!(ioctl(sock\_raw\_fd, SIOCGIFADDR, &req)))

    {

        int num = ntohl(((struct sockaddr\_in\*) (&req.ifr\_addr))->sin\_addr.s\_addr);

        for(i=0; i<4; i++)

        {

            send\_msg[31-i] = num>>8\*i & 0xff;   //将发送端的IP地址组包

        }

    }

    //5.获取本地机(eth0)的MAC

    if (!(ioctl(sock\_raw\_fd, SIOCGIFHWADDR, (char \*) &req)))

    {

        for(i=0; i<6; i++)

        {

            //将src\_mac、发送端的MAC地址组包

            send\_msg[22+i] = send\_msg[6+i] = (unsigned char) req.ifr\_hwaddr.sa\_data[i];

        }

    }

    while(1)

    {

        int i = 0;

        int num[4] = {0};

        unsigned char input\_buf[1024] = "";

        //6.获取所要扫描的网段（172.20.226.0）

        printf("input\_dst\_Network:172.20.226.0\n");

        fgets(input\_buf, sizeof(input\_buf), stdin);

        sscanf(input\_buf, "%d.%d.%d.", &num[0], &num[1], &num[2]//目的IP地址

        );

        //7.将键盘输入的信息组包

        for(i=0;i<4;i++)

            send\_msg[38+i] = num[i];//将目的IP地址组包

        //8.给1~254的IP发送ARP请求

        for(i=1; i<255; i++)

        {

            send\_msg[41] = i;

            int len = sendto(sock\_raw\_fd, send\_msg, 42, 0 , (struct sockaddr \*)&sll, sizeof(sll));

            if(len == -1)

            {

                perror("sendto");

            }

        }

        sleep(1);

    }

    return;

}

程序运行结果如下：

