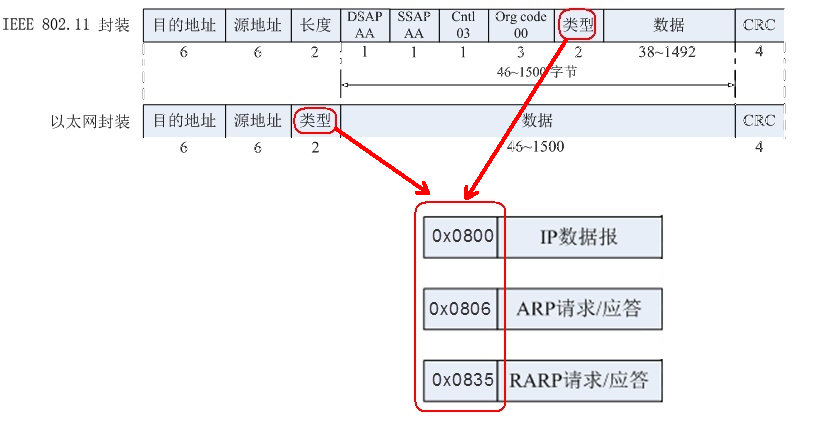
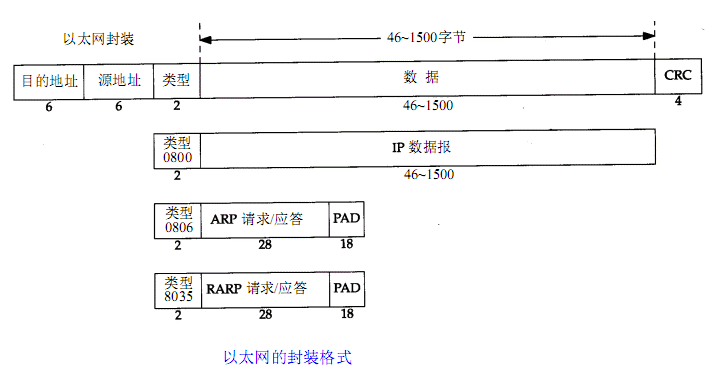
**一、前导**

我们可以通过原始套接字以及 recvfrom( ) 可以获取链路层的数据包，那我们接收的链路层数据包到底长什么样的呢？

**二、链路层封包格式**



**三、MAC 头部（有线局域网）**



注意：CRC、PAD 在组包时可以忽略

**四、链路层数据包的其中一种情况：**

unsigned char msg[1024] = {

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

    0xb8, 0x88, 0xe3, 0xe1, 0x10, 0xe6, // dst\_mac: b8:88:e3:e1:10:e6

    0xc8, 0x9c, 0xdc, 0xb7, 0x0f, 0x19, // src\_mac: c8:9c:dc:b7:0f:19

    0x08, 0x00,                         // 类型：0x0800 IP协议

    // …… ……

    // …… ……

};

**五、接收的链路层数据包，并对其进行简单分析：**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <netinet/ether.h>

int main(int argc, char \*argv[])

{

    int i = 0;

    unsigned char buf[1024] = "";

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

    char src\_mac[18] = "";

    char dst\_mac[18] = "";

    //获取链路层的数据帧

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

    //从buf里提取目的mac、源mac

    sprintf(dst\_mac, "%02x:%02x:%02x:%02x:%02x:%02x", buf[0], buf[1], buf[2], buf[3], buf[4], buf[5]);

    sprintf(src\_mac, "%02x:%02x:%02x:%02x:%02x:%02x", buf[6], buf[7], buf[8], buf[9], buf[10], buf[11]);

    //判断是否为IP数据包

    if (buf[12] == 0x08 && buf[13] == 0x00)

    {

        printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_IP数据报\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

        printf("MAC:%s >> %s\n", src\_mac, dst\_mac);

    } //判断是否为ARP数据包

    else if (buf[12] == 0x08 && buf[13] == 0x06)

    {

        printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_ARP数据报\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

        printf("MAC:%s >> %s\n", src\_mac, dst\_mac);

    } //判断是否为RARP数据包

    else if (buf[12] == 0x80 && buf[13] == 0x35)

    {

        printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_RARP数据报\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

        printf("MAC:%s>>%s\n", src\_mac, dst\_mac);

}

    return 0;

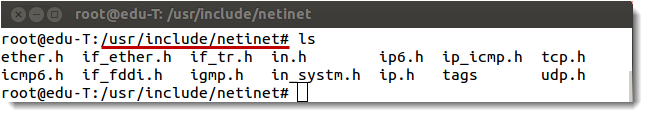
}

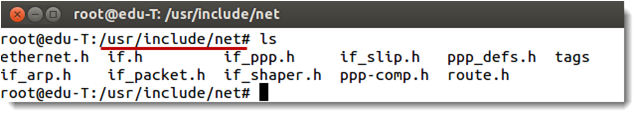
记得以管理者权限运行程序：



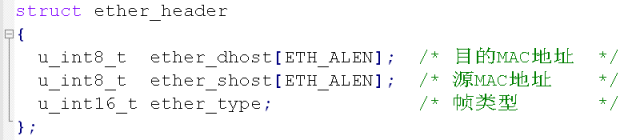
每个报文头部都有一个相应的结构体，通过这些结构体对报文进行相应的组包或拆包会方便很多。

ubuntu 12.04 中描述网络协议结构的文件如下：





**六、以太网头部（所需要头文件：#include <net/ethernet.h>）：**



上面的例子，改为用结构体实现，如下：

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <netinet/ether.h>

#include <net/ethernet.h>         // 以太网头部 头文件

#include <netinet/ip.h>           // ip头部 头文件

// #include <net/if\_arp.h>            // arp头部 头文件

int main(int argc, char \*argv[])

{

    int i = 0;

    unsigned char buf[1024] = "";

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

    while(1)

    {

        char src\_mac[18] = "";

        char dst\_mac[18] = "";

        //获取链路层的数据帧

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

        //从数据中提取mac首部信息(14个字节)

        struct ether\_header \*ethdr = NULL;

        ethdr = (struct ether\_header \*)buf;

        //从buf里提取目的mac、源mac

        sprintf(dst\_mac,"%02x:%02x:%02x:%02x:%02x:%02x", ethdr->ether\_dhost[0], ethdr->ether\_dhost[1],ethdr->ether\_dhost[2],ethdr->ether\_dhost[3],ethdr->ether\_dhost[4],ethdr->ether\_dhost[5]);

        sprintf(src\_mac,"%02x:%02x:%02x:%02x:%02x:%02x", ethdr->ether\_shost[0], ethdr->ether\_shost[1],ethdr->ether\_shost[2],ethdr->ether\_shost[3],ethdr->ether\_shost[4],ethdr->ether\_shost[5]);

        //判断是否为IP数据包

        if( 0x0800 == ntohs(ethdr->ether\_type) )

        {

            printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_IP数据报\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

            printf("MAC:%s >> %s\n",src\_mac,dst\_mac);

        }//0x0806为ARP数据包, 0x8035为RARP数据包

        else if( 0x0806 == ntohs(ethdr->ether\_type) || 0x8035 == ntohs(ethdr->ether\_type) )

        {

            printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_ARP数据报\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

            printf("MAC:%s >> %s\n",src\_mac,dst\_mac);

        }

    }

    return 0;

}