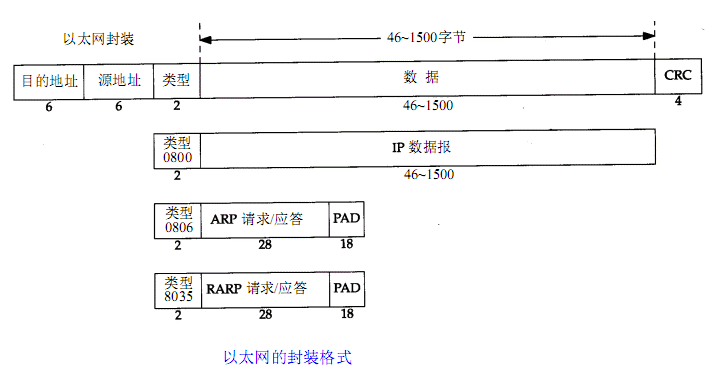
**以太网报文格式：**



**IP 报文格式：**



**UDP 报文格式：**



校验和函数：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

功能：

校验和函数

参数：

buf: 需要校验数据的首地址

nword: 需要校验数据长度的一半

返回值：

校验和

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

unsigned short checksum(unsigned short \*buf, int nword)

{

unsigned long sum;

for(sum = 0; nword > 0; nword--)

{

sum += htons(\*buf);

buf++;

}

sum = (sum>>16) + (sum&0xffff);

sum += (sum>>16);

return ~sum;

}

这里是在 ubuntu 下通过原始套接字组一个 udp 数据包，给 PC 机的网络调试助手发送信息：

#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

unsigned short checksum(unsigned short \*buf, int nword);//校验和函数

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------

        0x74, 0x27, 0xea, 0xb5, 0xef, 0xd8, //dst\_mac: 74-27-EA-B5-FF-D8

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

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

        //--------------组IP---------20------

        0x45, 0x00, 0x00, 0x00,             //版本号：4, 首部长度：20字节, TOS:0, --总长度--：

        0x00, 0x00, 0x00, 0x00,             //16位标识、3位标志、13位片偏移都设置0

        0x80, 17,   0x00, 0x00,             //TTL：128、协议：UDP（17）、16位首部校验和

        10,  221,   20,  11,                //src\_ip: 10.221.20.11

        10,  221,   20,  10,                //dst\_ip: 10.221.20.10

        //--------------组UDP--------8+78=86------

        0x1f, 0x90, 0x1f, 0x90,             //src\_port:0x1f90(8080), dst\_port:0x1f90(8080)

        0x00, 0x00, 0x00, 0x00,               //#--16位UDP长度--30个字节、#16位校验和

    };

    int len = sprintf(send\_msg+42, "%s", "this is for the udp test");

    if(len % 2 == 1)//判断len是否为奇数

    {

        len++;//如果是奇数，len就应该加1(因为UDP的数据部分如果不为偶数需要用0填补)

    }

    \*((unsigned short \*)&send\_msg[16]) = htons(20+8+len);//IP总长度 = 20 + 8 + len

    \*((unsigned short \*)&send\_msg[14+20+4]) = htons(8+len);//udp总长度 = 8 + len

    //3.UDP伪头部

    unsigned char pseudo\_head[1024] = {

        //------------UDP伪头部--------12--

        10,  221,   20,  11,                //src\_ip: 10.221.20.11

        10,  221,   20,  10,                //dst\_ip: 10.221.20.10

        0x00, 17,   0x00, 0x00,                 //0,17,#--16位UDP长度--20个字节

    };

    \*((unsigned short \*)&pseudo\_head[10]) = htons(8 + len);//为头部中的udp长度（和真实udp长度是同一个值）

    //4.构建udp校验和需要的数据报 = udp伪头部 + udp数据报

    memcpy(pseudo\_head+12, send\_msg+34, 8+len);//--计算udp校验和时需要加上伪头部--

    //5.对IP首部进行校验

    \*((unsigned short \*)&send\_msg[24]) = htons(checksum((unsigned short \*)(send\_msg+14),20/2));

    //6.--对UDP数据进行校验--

    \*((unsigned short \*)&send\_msg[40]) = htons(checksum((unsigned short \*)pseudo\_head,(12+8+len)/2));

    //6.发送数据

    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

    if(-1 == ioctl(sock\_raw\_fd, SIOCGIFINDEX, &req))    //获取网络接口

    {

        perror("ioctl");

        close(sock\_raw\_fd);

        exit(-1);

    }

    /\*将网络接口赋值给原始套接字地址结构\*/

    bzero(&sll, sizeof(sll));

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

    len = sendto(sock\_raw\_fd, send\_msg, 14+20+8+len, 0 , (struct sockaddr \*)&sll, sizeof(sll));

    if(len == -1)

    {

        perror("sendto");

    }

    return 0;

}

unsigned short checksum(unsigned short \*buf, int nword)

{

    unsigned long sum;

    for(sum = 0; nword > 0; nword--)

    {

        sum += htons(\*buf);

        buf++;

    }

    sum = (sum>>16) + (sum&0xffff);

    sum += (sum>>16);

    return ~sum;

}