【原创评选】2014年7月-8月原创博文评选

wjlkoorey的博客

wjlkoorey.blog.chinaunix.net

wanderlust in the sea...

推荐博客

wjlkoorey258

博客访问：509964
博文数量：97
博客积分：671
博客等级：上尉
技术积分：10228
用户组：普通用户
注册时间：2010-12-18 16:08

加关注 短消息

原创 Linux网络编程：原始套接字的魔力【续】

2012-07-30 21:38:21

分类：LINUX

如何从链路层直接发送数据帧

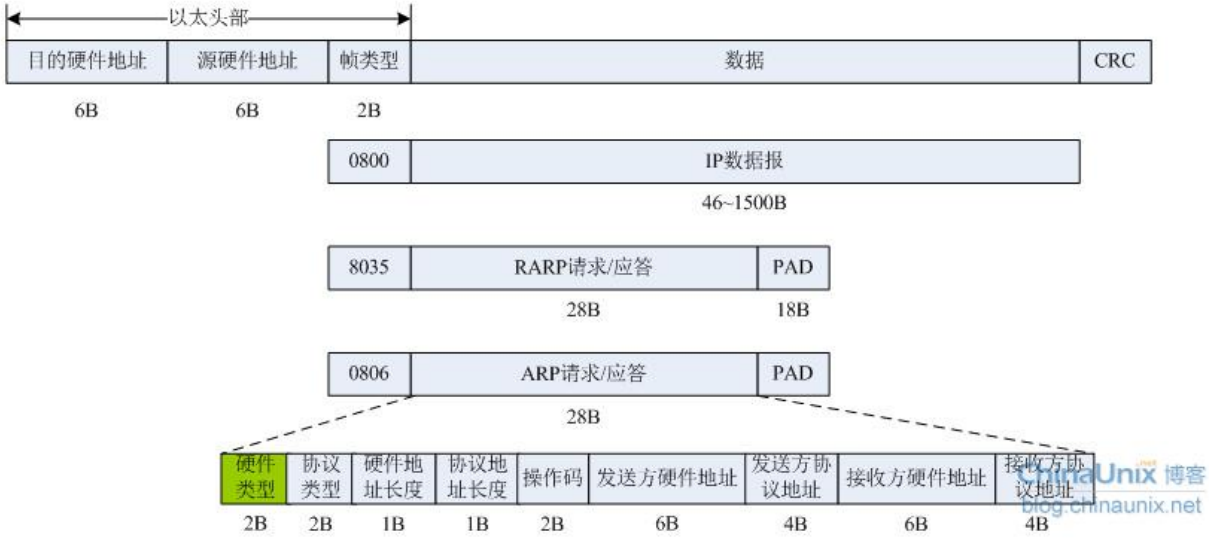
本来以为这部分都弄完了，结果有朋友反映说看了半天还是没看到如何从链路层直接发送数据。因为上一篇里面提到的是从链路层“收发”数据，结果只“收”完，忘了“发”，实在抱歉，所以就有这篇续出来了。

上一节我们主要研究了如何从链路层直接接收数据帧，可以通过bind函数来将原始套接字绑定到本地一个接口上，然后该套接字就只接收从该接口收上来的对应的数据包。今天我们用原始套接字来手工实现链路层ARP报文的发送和接收，以便大家对原始套接字有更深刻的掌握和理解。

ARP全称为地址解析协议，是链路层广泛使用的一种寻址协议，完成32比特IP地址到48比特MAC地址的映射转换。在以太网中，当一台主机需要向另外一台主机发送消息时，它会首先在自己本地的ARP缓存表中根据目的主机的IP地址查找其对应的MAC地址，如果找到了则直接向其发送消息。如果未找到，它首先会在全网发送一个ARP广播查询，这个查询

的消息会被以太网中所有主机接收到，然后每个主机就根据ARP查询报文中所指定的IP地址来检查该报文是不是发给自己的，如果不是则直接丢弃；只有被查询的目的主机才会对这个消息进行响应，然后将自己的MAC地址通告给发送者。

也就是说，链路层中是根据MAC地址来确定唯一一台主机。以太帧格式如下：



以太帧首部中2字节的帧类型字段指定了其上层所承载的具体协议，常见的有0x0800表示是IP报文、0x0806表示RARP协议、0x0806即为我们将要讨论的ARP协议。

硬件类型：1表示以太网。

协议类型：0x0800表示IP地址。和以太头部中帧类型字段相同。

硬件地址长度和协议地址长度：对于以太网中的ARP协议而言，分别为6和4；

操作码：1表示ARP请求；2表示ARP应答；3表示RARP请求；4表示RARP应答。

我们这里只讨论硬件地址为以太网地址、协议地址为IP地址的情形，所以剩下四个字段就分别表示发送方的MAC和IP地址、接收方的MAC和IP地址了。

注意：对于一个ARP请求报文来说，除了接收方硬件地址外，其他字段都要填充。当系统收到一个ARP请求时，会查询该请求报文中接收方的协议地址是否和自己的IP地址相等，如果相等，它就把自己的硬件地址和协议地址填充进去，将发送和接收方的地址互换，然后将操

作码改为2，发送回去。

下面看一个使用原始套接字发送ARP请求的例子：

点击(此处)折叠或打开

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include <string.h>
4.  #include <unistd.h>
5.  #include <errno.h>
6.  #include <sys/socket.h>
7.  #include <sys/ioctl.h>
8.  #include <sys/types.h>
9.  #include <netinet/in.h>
10. #include <netinet/ip.h>
11. #include <netinet/if_ether.h>
12. #include <net/if_arp.h>
13. #include <netpacket/packet.h>
14. #include <net/if.h>
15. #include <net/ethernet.h>
16.
17. #define BUFLEN 42
18.
19. int main(int argc,char** argv){
20.     int skfd,n;
21.     char buf[BUFLEN]={0};
22.     struct ether_header *eth;
23.     struct ether_arp *arp;
24.     struct sockaddr_ll toaddr;
25.     struct in_addr targetIP,srcIP;
26.     struct ifreq ifr;
27.
28.     unsigned char src_mac[ETH_ALEN]={0};
29.     unsigned char dst_mac[ETH_ALEN]={0xff,0xff,0xff,0xff,0xff,0xff}; //全网广播ARP请求
30.     if(3 != argc){
31.             printf("Usage: %s netdevName dstIP\n",argv[0]);
32.             exit(1);
33.     }
34.
35.     if(0>(skfd=socket(PF_PACKET,SOCK_RAW,htons(ETH_P_ALL)))){
36.             perror("Create Error");
37.             exit(1);
38.     }
39.
```

我的朋友

2005227 lbird_11 tyj19891

lawrence mzh2100 huangba

liucaipi Zackary1 WbBullFr

最近访客

acorpe 雷锋不谢 y841618

cym0417 猪也有春 米娜拉夜

hmily36 VEGETA phoenixc

订阅

推荐博文

·模仿之中也少不了创新——Leo...
·学习Swift之(二)：swift开发...
·学习Swift之(一)：关于swift...
·实现dup2函数,要求不使用fcnt...
·LR模型的Spark实现
·对Oracle高水位线的研究实践...
·为学习Hadoop使用VMware准备3...
·【故障处理】opmn启动失败及...
·oracle 11g ASM 磁盘组在线扩...
·数据迁移中的数据库检查和建...

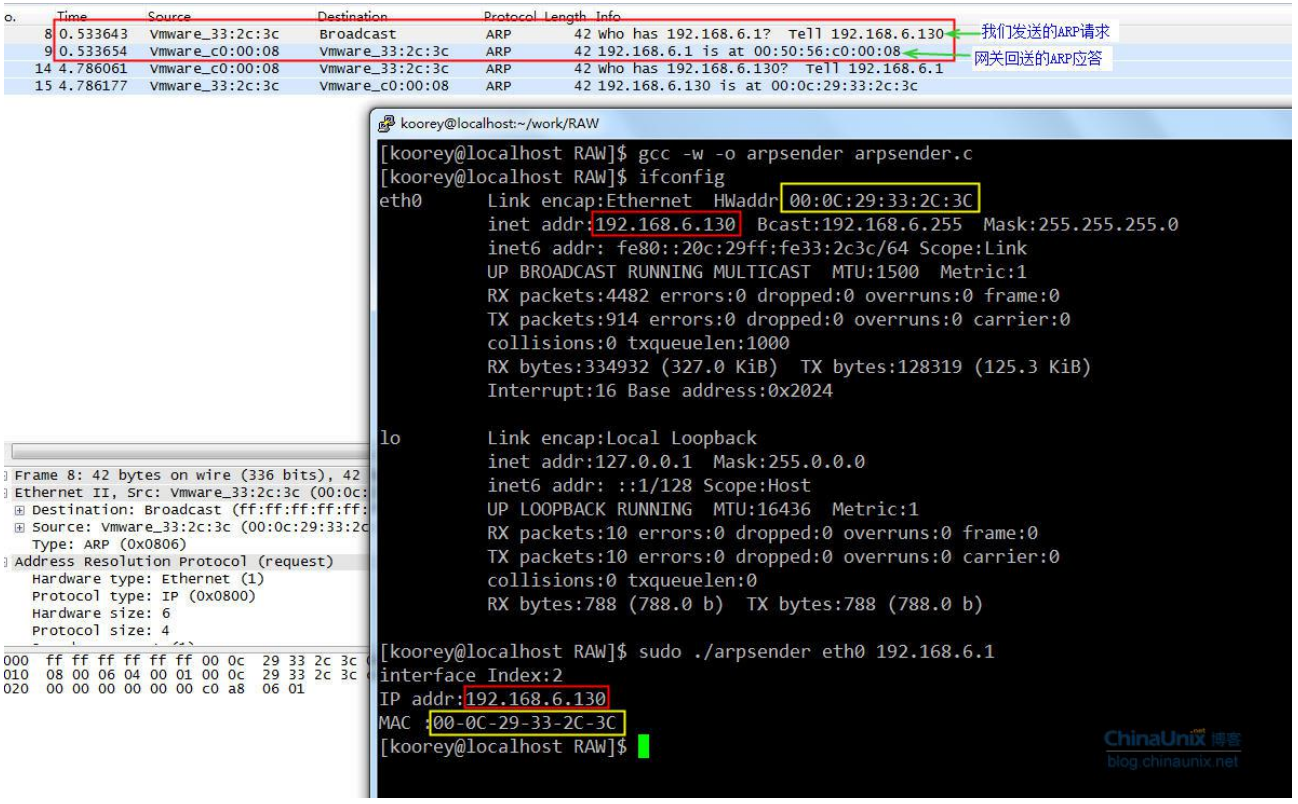
热词专题

·李体育老师荣获“杰出传承人...
·string.h
·安装oracle
·VMware VDP
·Android + 系统属性

```
40.       bzero(&toaddr,sizeof(toaddr));
41.       bzero(&ifr,sizeof(ifr));
42.       strcpy(ifr.ifr_name,argv[1]);
43.
44.       //获取接口索引
45.       if(-1 == ioctl(skfd,SIOCGIFINDEX,&ifr)){
46.             perror("get dev index error:");
47.             exit(1);
48.       }
49.       toaddr.sll_ifindex = ifr.ifr_ifindex;
50.       printf("interface Index:%d\n",ifr.ifr_ifindex);
51.       //获取接口IP地址
52.       if(-1 == ioctl(skfd,SIOCGIFADDR,&ifr)){
53.             perror("get IP addr error:");
54.             exit(1);
55.       }
56.       srcIP.s_addr = ((struct sockaddr_in*)&(ifr.ifr_addr))->sin_addr.s_addr;
57.       printf("IP addr:%s\n",inet_ntoa(((struct sockaddr_in*)&(ifr.ifr_addr))->sin_addr));
58.
59.       //获取接口的MAC地址
60.       if(-1 == ioctl(skfd,SIOCGIFHWADDR,&ifr)){
61.             perror("get dev MAC addr error:");
62.             exit(1);
63.       }
64.
65.       memcpy(src_mac,ifr.ifr_hwaddr.sa_data,ETH_ALEN);
66.       printf("MAC :%02X-%02X-%02X-%02X-%02X-%02X\n",src_mac[0],src_mac[1],src_mac[2],src_mac[3],s
    rc_mac[4],src_mac[5]);
67.
68.
69.       //开始填充，构造以太头部
70.       eth=(struct ether_header*)buf;
71.       memcpy(eth->ether_dhost,dst_mac,ETH_ALEN);
72.       memcpy(eth->ether_shost,src_mac,ETH_ALEN);
73.       eth->ether_type = htons(ETHERTYPE_ARP);
74.
75.       //手动开始填充用ARP报文首部
76.       arp=(struct arphdr*)(buf+sizeof(struct ether_header));
77.       arp->arp_hrd = htons(ARPHRD_ETHER); //硬件类型为以太
78.       arp->arp_pro = htons(ETHERTYPE_IP); //协议类型为IP
79.
80.       //硬件地址长度和IPV4地址长度分别是6字节和4字节
81.       arp->arp_hln = ETH_ALEN;
82.       arp->arp_pln = 4;
83.
84.       //操作码，这里我们发送ARP请求
```

```
85.       arp->arp_op = htons(ARPOP_REQUEST);
86.
87.       //填充发送端的MAC和IP地址
88.       memcpy(arp->arp_sha,src_mac,ETH_ALEN);
89.       memcpy(arp->arp_spa,&srcIP,4);
90.
91.       //填充目的端的IP地址，MAC地址不用管
92.       inet_pton(AF_INET,argv[2],&targetIP);
93.       memcpy(arp->arp_tpa,&targetIP,4);
94.
95.       toaddr.sll_family = PF_PACKET;
96.       n=sendto(skfd,buf,BUFLEN,0,(struct sockaddr*)&toaddr,sizeof(toaddr));
97.
98.       close(skfd);
99.       return 0;
100.  }
```

结果如下：



可以看到，我向网关发送一个ARP查询请求，报文中携带了网关的IP地址以及我本地主机的IP和MAC地址。网关收到该请求后，对我的这个报文进行了回应，将它的MAC地址

在ARP应答报文中发给我了。

在这个示例程序中，我们完全自己手动构造了以太帧头部，并完成了整个ARP请求报文的填充，最后用sendto函数，将我们的数据通过eth0接口发送出去。这个程序的灵活性还在于支持多网卡，使用时只要指定网卡名称(如eth0或eth1)，程序便会自动去获取指定接口相应的IP和MAC地址，然后用它们去填充ARP请求报文中对应的各字段。

在头文件<net/thernet.h>里，主要对以太帧首部进行了封装：

点击(此处)折叠或打开

```
1.  struct ether_header
2.  {
3.     u_int8_t ether_dhost[ETH_ALEN]; /* destination eth addr */
4.     u_int8_t ether_shost[ETH_ALEN]; /* source ether addr */
5.     u_int16_t ether_type; /* packet type ID field */
6.  } __attribute__ ((__packed__));
```

在头文件<net/if_arp.h>中，对ARP首部进行了封装：

点击(此处)折叠或打开

```
1.  struct arphdr
2.  {
3.      unsigned short ar_hrd; /* format of hardware address */
4.      unsigned short ar_pro; /* format of protocol address */
5.      unsigned char ar_hln; /* length of hardware address */
6.      unsigned char ar_pln; /* length of protocol address */
7.      unsigned short ar_op; /* ARP opcode (command) */
8.  }
```

而头文件<netinet/if_ether.h>里，又对ARP整个报文进行了封装：

点击(此处)折叠或打开

```
1.  struct ether_arp {
2.      struct arphdr ea_hdr; /* fixed-size 8 bytes header */
3.      u_int8_t arp_sha[ETH_ALEN]; /* sender hardware address */
4.      u_int8_t arp_spa[4]; /* sender protocol address */
5.      u_int8_t arp_tha[ETH_ALEN]; /* target hardware address */
6.      u_int8_t arp_tpa[4]; /* target protocol address */
7.  };
8.
9.  #define arp_hrd ea_hdr.ar_hrd
```

```
#define arp_pro ea_hdr.ar_pro
#define arp_hln ea_hdr.ar_hln
#define arp_pln ea_hdr.ar_pln
#define arp_op ea_hdr.ar_op
```

最后再看一个简单的接收ARP报文的小程序：

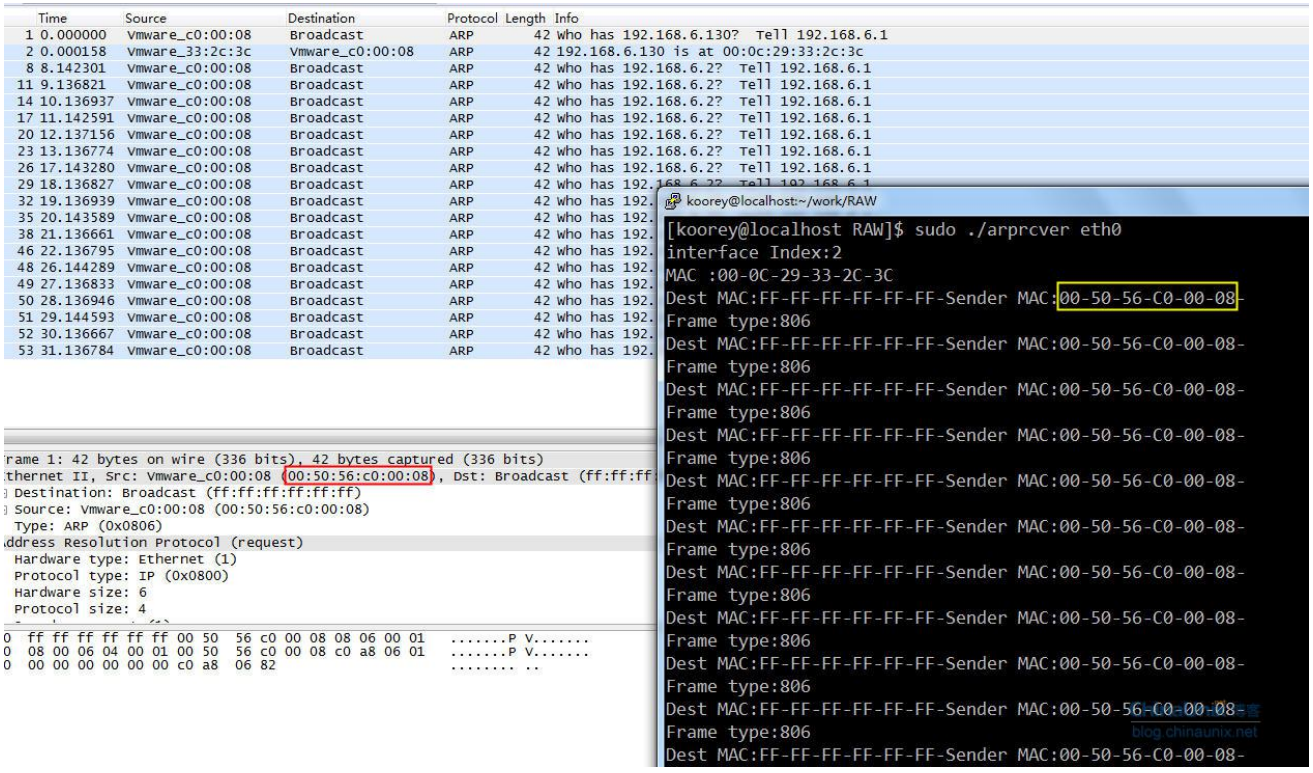
点击(此处)折叠或打开

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <errno.h>
#include <sys/socket.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/if_ether.h>
#include <net/if_arp.h>
#include <netpacket/packet.h>
#include <net/if.h>
#define BUFLEN 60

int main(int argc,char** argv){
    int i,skfd,n;
    char buf[ETH_FRAME_LEN]={0};
    struct ethhdr *eth;
    struct ether_arp *arp;
    struct sockaddr_ll fromaddr;
    struct ifreq ifr;

    unsigned char src_mac[ETH_ALEN]={0};

    if(2 != argc){
        printf("Usage: %s netdevName\n",argv[0]);
        exit(1);
    }

    //只接收发给本机的ARP报文
    if(0>(skfd=socket(PF_PACKET,SOCK_RAW,htons(ETH_P_ARP)))){
        perror("Create Error");
        exit(1);
    }

    bzero(&fromaddr,sizeof(fromaddr));
```

```
39.       bzero(&ifr,sizeof(ifr));
40.       strcpy(ifr.ifr_name,argv[1]);
41.
42.       //获取接口索引
43.       if(-1 == ioctl(skfd,SIOCGIFINDEX,&ifr)){
44.           perror("get dev index error:");
45.           exit(1);
46.       }
47.       fromaddr.sll_ifindex = ifr.ifr_ifindex;
48.       printf("interface Index:%d\n",ifr.ifr_ifindex);
49.
50.       //获取接口的MAC地址
51.       if(-1 == ioctl(skfd,SIOCGIFHWADDR,&ifr)){
52.           perror("get dev MAC addr error:");
53.           exit(1);
54.       }
55.
56.       memcpy(src_mac,ifr.ifr_hwaddr.sa_data,ETH_ALEN);
57.       printf("MAC :%02X-%02X-%02X-%02X-%02X-%02X\n",src_mac[0],src_mac[1],src_mac[2],src_mac[3],s
      rc_mac[4],src_mac[5]);
58.
59.       fromaddr.sll_family = PF_PACKET;
60.       fromaddr.sll_protocol=htons(ETH_P_ARP);
61.       fromaddr.sll_hatype=ARPHRD_ETHER;
62.       fromaddr.sll_pkttype=PACKET_HOST;
63.       fromaddr.sll_halen=ETH_ALEN;
64.       memcpy(fromaddr.sll_addr,src_mac,ETH_ALEN);
65.
66.       bind(skfd,(struct sockaddr*)&fromaddr,sizeof(struct sockaddr));
67.
68.       while(1){
69.           memset(buf,0,ETH_FRAME_LEN);
70.           n=recvfrom(skfd,buf,ETH_FRAME_LEN,0,NULL,NULL);
71.           eth=(struct ethhdr*)buf;
72.           arp=(struct ether_arp*)(buf+14);
73.
74.           printf("Dest MAC:");
75.           for(i=0;i<ETH_ALEN;i++){
76.               printf("%02X-",eth->h_dest[i]);
77.           }
78.           printf("Sender MAC:");
79.           for(i=0;i<ETH_ALEN;i++){
80.               printf("%02X-",eth->h_source[i]);
81.           }
82.
83.           printf("\n");
```

```
84.            printf("Frame type:%0X\n",ntohs(eth->h_proto));
85.
86.            if(ntohs(arp->arp_op)==2){
87.                printf("Get an ARP replay!\n");
88.            }
89.        }
90.        close(skfd);
91.        return 0;
92.    }
```

该示例程序中，调用recvfrom之前我们调用了bind系统调用，目的是仅从指定的接口接收ARP报文(由socket函数的第三个参数“ETH_P_ARP”决定)。可以对比一下，该程序与博文“Linux网络编程：原始套接字的魔力【下】”里介绍的抓包程序的区别。



小结：通过这几个章节的热身，相信大家对网络编程中常见的一系列API函数socket,bind,listen,connect,sendto,recvfrom,close等的认识应该会有一个较高的突破。当然，你也必须赶快对它们熟悉起来，因为后面我们不但要“知其然”，还要知其“所以然”。后面，我们会以这些函数调用为主线，看看它们到底在内核中做些哪些事情，而这又对我们理解协议

栈的实现原理有什么帮助做进一步的分析和讨论。

上一篇：Linux网络编程：原始套接字的魔力【下】
下一篇：揭开网络编程常见API的面纱【上】

相关热门文章

热 Java 网络编程 tcp/udp 文章收...
荐 linux 常见服务端口
热 kernel 报错l701.exel[16922]:...
热 WebSocket
荐 【ROOTFS搭建】busybox的httpd...
热 C语言 如何在一个整型左边补0...
热 网络编程总结
荐 xmanager 2.0 for linux配置
热 python无法爬取阿里巴巴的数据...
热 Linux高性能服务器编程 第五...
荐 什么是shell
热 linux-2.6.28 和linux-2.6.32....
热 网络编程总结
荐 linux socket的bug??
热 linux su - username -c 命...

给主人留下些什么吧！~~

qiaoshui 2013-11-22 14:33:59

看了您的几篇博文，确实受益匪浅啊！
我想问一下，通过原始套接口抓包，源地址不变，然后修改包中的目标地址，是否就可以做负载均衡服务了？
客户端返回的时候，发的包，是否能直接发给目标服务器，而不是发给负载均衡服务器？

回复 | 举报

wjlkoorey258 2013-10-30 18:30:50

> cqf00：是wireshark么？这个工具好不好用？

习惯了就好用了，至少我是这么觉得

回复 | 举报

cqf00 2013-10-29 14:58:23

> cqf00：我想知道，你图片中的那个带界面的抓包工具是啥

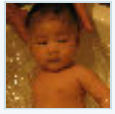
是wireshark么？这个工具好不好用？

回复 | 举报

cqf00 2013-10-29 13:37:49

我想知道，你图片中的那个带界面的抓包工具是啥

回复 | 举报



wenqilee 2012-08-16 23:21:00

好文！

回复 | 举报

评论热议

请登录后评论。

登录 注册