# 网络抓包工具wireshark

实验名称：网络抓包工具wireshark

学时安排：2课时 指导老师：李赞

实验类别：验证型、设计型 实验要求：1人1组

学号： 姓名： 班级：

## 一、实验目的和任务

1. 本实验要求掌握wireshark的安装与网络包抓取分析http，udp，icmp协议包组成。

2. 使用原始套接字获取ICMP等网络数据包。

## 二、实验设备介绍

B701实验室基本配置环境如下：

1.软件环境： win10操作系统，VMware workstation，ubuntu18。

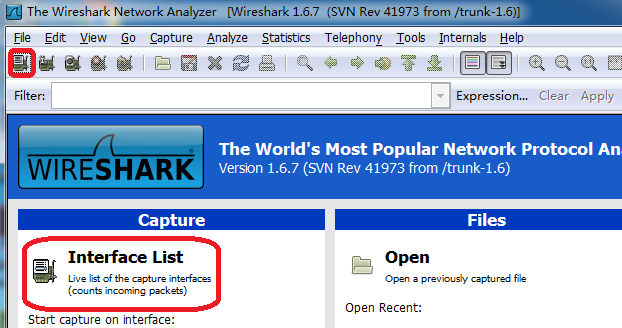
## 三、实验内容和步骤

### 3.1 抓包软件WireShark的使用

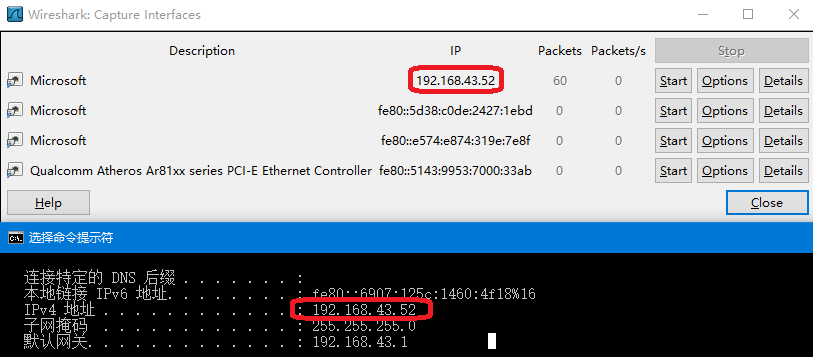
wireshark是linux平台一个非常方便的抓取网络数据包的软件，利用它可以让我们分析已有的网络数据包的格式和构成。下面分别列举了windows平台debian平台和Ubuntu平台wireshark软件的使用过程，通过抓包可以直接学习成熟协议的组成，例如：http，icmp，udp等协议包组成。

#### 3.1.1 windows平台WireShark的使用

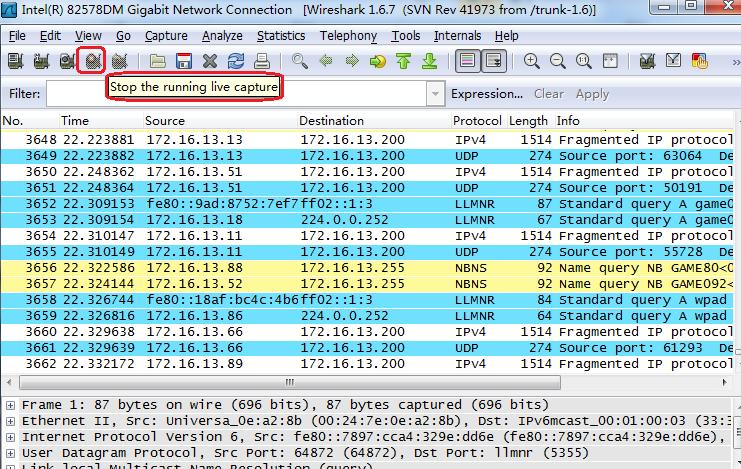
Windows平台WireShark软件的启动界面如下，抓包工作前首先要选择网卡列表。



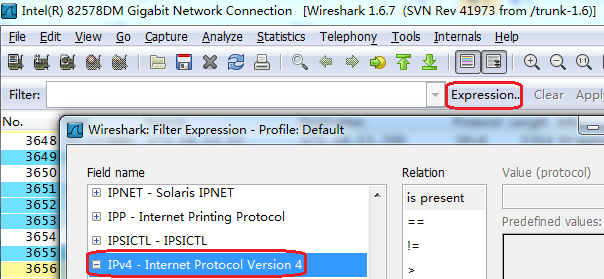
在windows平台查看本机的IP地址为192.168.43.52，在wireshark的网卡列表上点击IP列，找到IP对应的网卡，start按钮开始进行抓包。



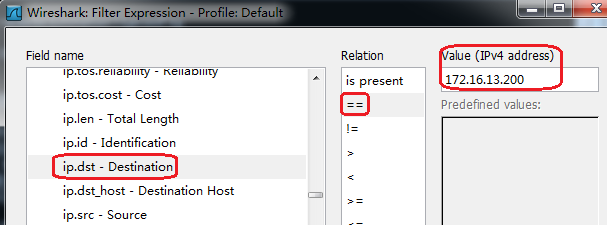
网络抓取后要尽快停止抓包，因为过多的网络包会造成查看困难。

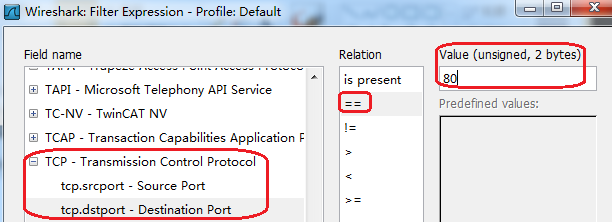


对抓取到的网络进行过滤，在Expressiong按钮下选择筛选条件。



可以设置筛选条件为ip.dst=172.16.13.200，表示只保留指定目录IP地址的网络包。





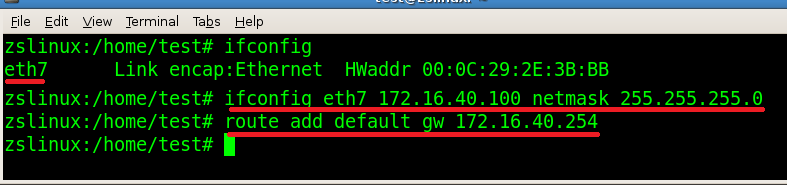
对于HTTP协议包还可以查看或者复制数据包的内容，以进行后续的网络程序开发。

#### 3.1.2 Debian平台WireShark的使用

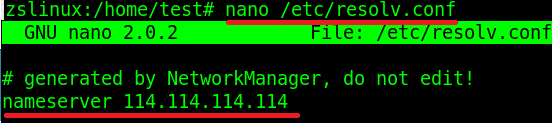
启动虚拟机[Debian]，设置虚拟网络编辑器，选择桥接的网卡，这样虚拟机可以连接外网。请将下面文档中网络地址修改为:172.16.56.XX。



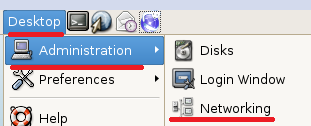
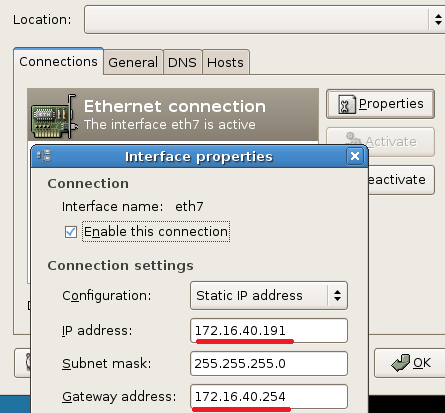
打开命令终端使用命令，切换到root账号，配置Debian网络参数，Debian IP为172.16.40.XX，网络掩码:255.255.255.0，网关:172.16.40.254。



使用nano命令配置DNS：

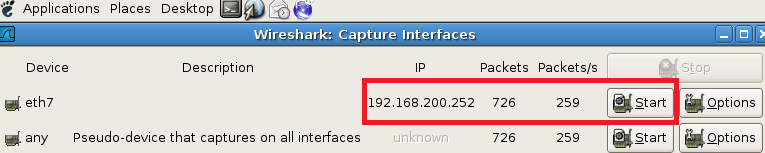


查看网卡信息在Debian中是否正常。

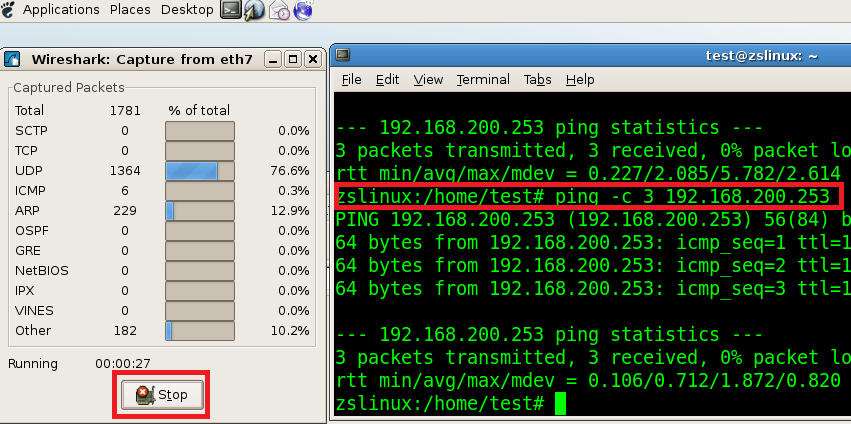
 

打开IceWeasel浏览器，看是否可以打开[www.baidu.com](http://www.baidu.com)的页面。

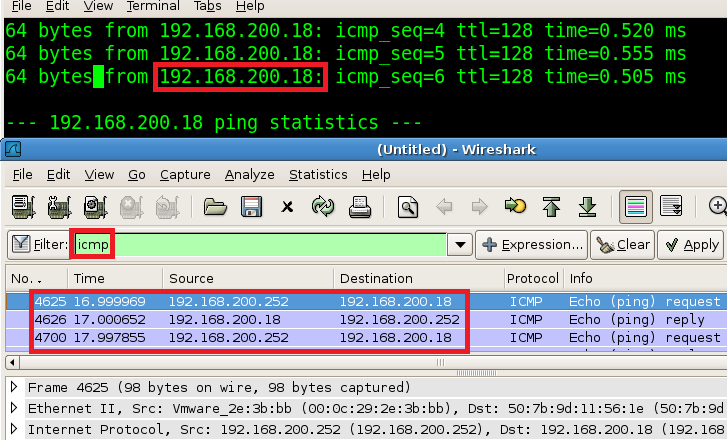
启动Debian终端，使用su命令升级为root账号，密码为333333，输入命令：wireshark，启动抓包软件。选择网卡列表， th7的start按钮，开始抓包。请将下图的IP地址修正为本机的正确IP地址。



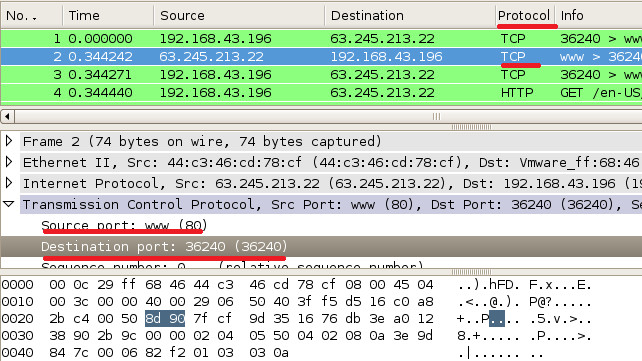
在终端执行ping命令: ping -c 3 192.168.200.253 然后点击stop命令，此时网卡的包信息已经被wireshark捕获。



测试其它的主机，在过滤框中输入:ICMP，可以捕获到ping命令使用的icmp网络包。



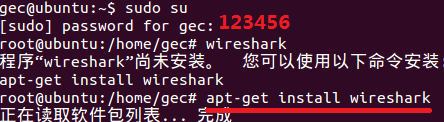
在该网络包中，Source字段指示该网络包源IP地址，Destination字段指示目标IP地址，98bytes captured 指示该数据帧一共长98字节(含以太网+IP+ICMP)。



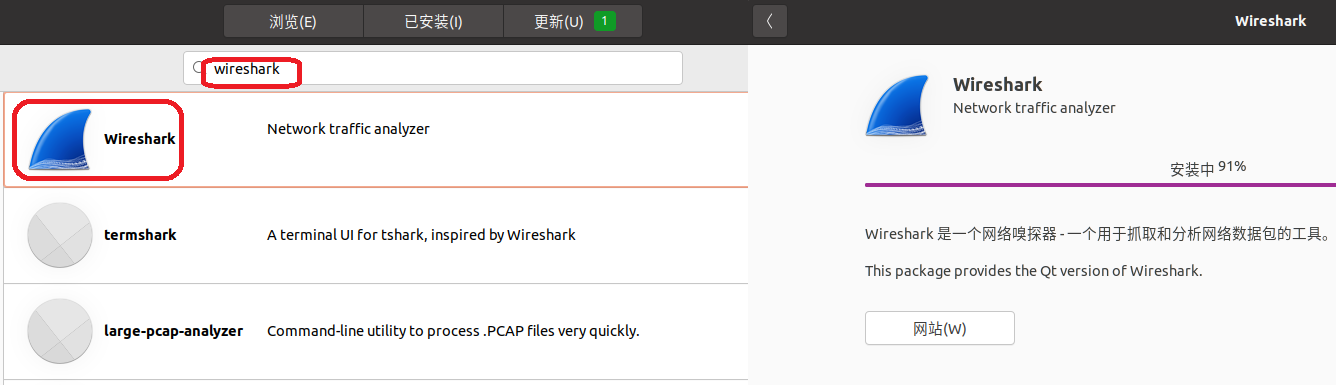
在上面的抓包截图中，Protocol指示该层协议的名称为TCP，Source Port指示源端口值为80，Destination Port指示目标端口值为36240。

#### 3.1.3 ubuntu20平台WireShark的使用

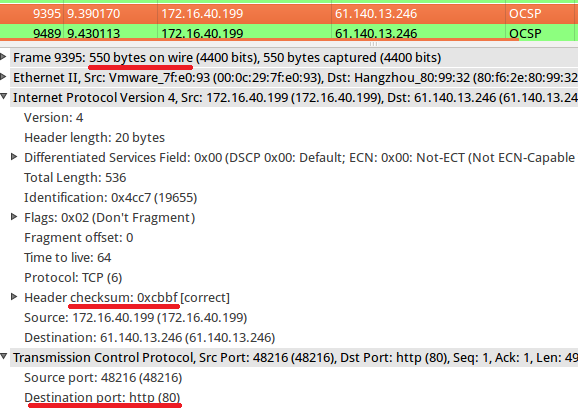
如果Ubuntu没有安装wireshark软件，可选择命令在线安装wireshark。配置ubuntu的外网IP:172.16.56.XX，设置好网卡桥接，通过联网方式安装wireshark软件。



也可以在软件中心查找wireshark，进行在线安装。



下面是在ubuntu中抓包的演示。



### 3.2 原始套接字抓取ICMP网络包

下面的程序使用原始套接字来获取网络上的IP数据包，并实现了包的解析。sudo

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <errno.h>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <linux/in.h>

#include <linux/if\_ether.h>

#include <net/if.h>

#include <sys/ioctl.h>

typedef struct \_ethdr

{

char src\_mac[6];

char dst\_mac[6];

short type;

}ETH\_HEADER;

typedef struct \_iphdr //定义IP首部

{

unsigned char h\_verlen; //4位首部长度+4位IP版本号

unsigned char tos; //8位服务类型TOS

unsigned short total\_len; //16位总长度（字节）

unsigned short ident; //16位标识

unsigned short frag\_and\_flags; //3位标志位

unsigned char ttl; //8位生存时间 TTL

unsigned char proto; //8位协议 (TCP, UDP 或其他)

unsigned short checksum; //16位IP首部校验和

unsigned int sourceIP; //32位源IP地址

unsigned int destIP; //32位目的IP地址

}IP\_HEADER;

int n;

pthread\_t id1;

int sock;

char buffer[1518];

unsigned char \*iphead, \*ethhead;

/\*线程函数声明\*/

void pthread1(void \*arg)

{

printf("it is a new thread.....\n");

while (1)

{

printf("----------\n");

n = recvfrom(sock,buffer,sizeof(buffer),0,NULL,NULL);

printf("%d bytes read\n",n);

int num = 1;

for (num=1; num<=n; num++)

{

printf("%d ", buffer[num]);

if (0 == (num % 16) )

{

printf(";\n");

}

}

printf("\n");

/\* Check to see if the packet contains at least

\* complete Ethernet (14), IP (20) and TCP/UDP

\* (8) headers.

\*/

if (n<42)

{

perror("recvfrom():");

printf("Incomplete packet (errno is %d)\n",

errno);

close(sock);

exit(0);

}

ethhead = buffer;

char protocol = \*(ethhead+14+9); //

if(0x01 == protocol) //

{

printf("Source MAC address: "

"%02x:%02x:%02x:%02x:%02x:%02x\n",

ethhead[0],ethhead[1],ethhead[2],

ethhead[3],ethhead[4],ethhead[5]);

printf("Destination MAC address: "

"%02x:%02x:%02x:%02x:%02x:%02x\n",

ethhead[6],ethhead[7],ethhead[8],

ethhead[9],ethhead[10],ethhead[11]);

}

IP\_HEADER \*ipHdr = (IP\_HEADER \*)(buffer+sizeof(ETH\_HEADER));

unsigned char\* p = (unsigned char\*)&ipHdr->sourceIP;

printf("Source IP\t: %u.%u.%u.%u\n",p[0],p[1],p[2],p[3]);

p = (unsigned char\*)&ipHdr->destIP;

printf("Destination IP\t: %u.%u.%u.%u\n",p[0],p[1],p[2],p[3]);

#if 0

iphead = buffer+14; /\* Skip Ethernet header \*/

if (\*iphead==0x45) { /\* Double check for IPv4

\* and no options present \*/

printf("Source host %d.%d.%d.%d\n",

iphead[12],iphead[13],

iphead[14],iphead[15]);

printf("Dest host %d.%d.%d.%d\n",

iphead[16],iphead[17],

iphead[18],iphead[19]);

printf("Source,Dest ports %d,%d\n",

(iphead[20]<<8)+iphead[21],

(iphead[22]<<8)+iphead[23]);

printf("Layer-4 protocol %d\n",iphead[9]);

}

#endif

}

}

int main(int argc, char \*\*argv)

{

int ret;

printf("entering main thread\n");

if ( (sock=socket(PF\_PACKET, SOCK\_RAW,htons(ETH\_P\_IP)))<0) //

{

perror("socket");

exit(1);

}

/\*创建线程pthread1\*/

ret=pthread\_create(&id1,NULL,(void \*)pthread1, NULL);

if(ret!=0) {

perror("pthread create1");

}

while(1)

{

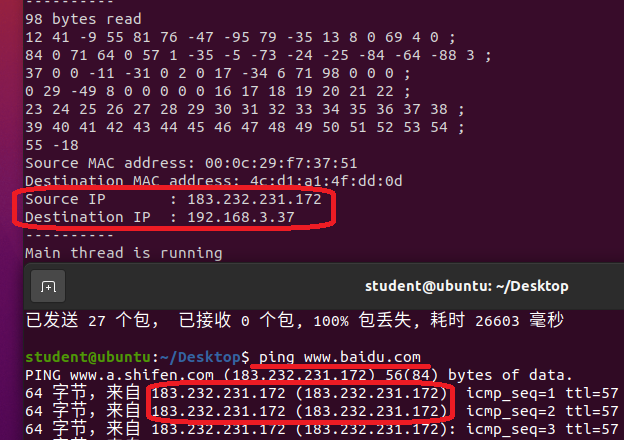
printf("Main thread is running\n");

sleep(1);

}

}

参考的程序运行效果如下图：



## 四、程序调试中遇到的问题和解决过程及运行结果

4.1配置虚拟机IP地址，使用wireshark抓取网络数据包，截屏后上传到学习通平台。

4.2 运行原始套接字程序，分析程序代码，将运行结果截屏后上传到学习通平台。