

**计算机网络实验报告**

|  |  |
| --- | --- |
| **实验：** | 实验2 |
| **专业**： | 计算机科学与技术 |
| **班级**： | 1班 |
| **姓名**： | 姚怀聿 |
| **学号**： | 22920202204632 |

**2022年10月15日**

**目 录**

[一、 实验目的 3](#_Toc117709638)

[二、 实验内容 3](#_Toc117709639)

[任务1: 捕获和分析有线以太网数据包 3](#_Toc117709640)

[1.1分析MAC帧 3](#_Toc117709641)

[1.2分析IP数据报首部 5](#_Toc117709642)

[1.3观察IP分片 9](#_Toc117709643)

[1.4 ICMP协议分析（以ping指令为例） 13](#_Toc117709644)

[1.5 tracert工作原理分析 17](#_Toc117709645)

[1.6 ARP协议分析 20](#_Toc117709646)

[任务2: 捕获和分析802.11数据 31](#_Toc117709647)

**[2.1](#_Toc117709648)**[搭建实验环境 31](#_Toc117709648)

**[2.2](#_Toc117709649)**[构建无线环境，捕获无线数据包、分析802.11数据 32](#_Toc117709649)

[任务3: 探索Wireshark更多功能和其它抓包工具(选做) 35](#_Toc117709650)

[探索Wireshark更多功能 35](#_Toc117709651)

[三、 实验小结 41](#_Toc117709652)

# 实验目的

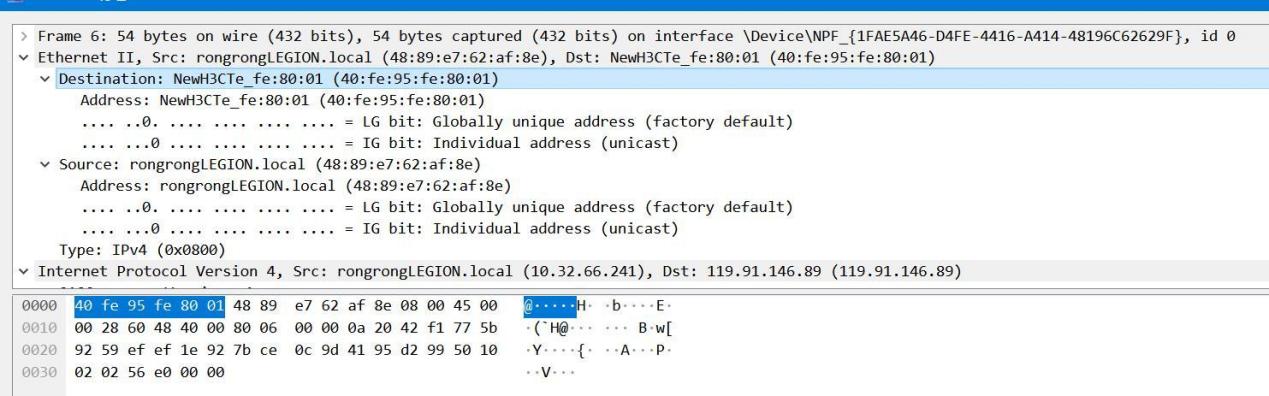
* 学习捕获和分析网络数据包
* 掌握以太网MAC帧、802.11数据帧和IPv4数据包的构成，了解各字段的含义
* 掌握ICMP协议，ping和tracert指令的工作原理
* 掌握ARP协议的请求/响应机理

# 实验内容

## 任务1: 捕获和分析有线以太网数据包

准备步骤:学习WireShark基本操作

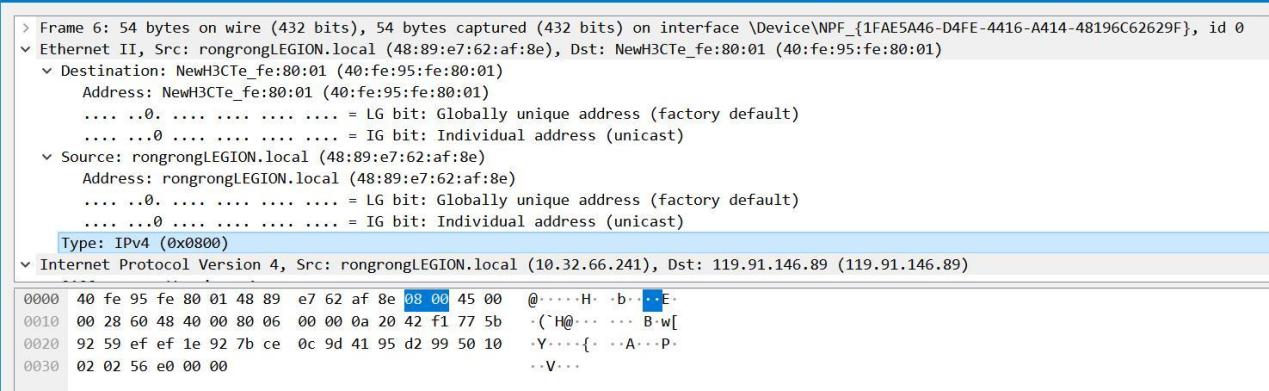
### 1.1分析MAC帧



开头六个字节是以太网MAC帧的目的地址，以该MAC帧为例，其目的地址为NewH3CTe\_fe :80:01(40:fe:95:fe:80:01)。



接下来六个字节是以太网MAC帧的源地址，以该MAC帧为例，其源地址为rongrongLEGION.local(48:89:e7:62:af:8e)，也即该电脑的本地地址。



再接下来的两个字节标志了上一层使用的协议类型，如该类型字段的值为0x0800，表示上层使用的是IP数据报。

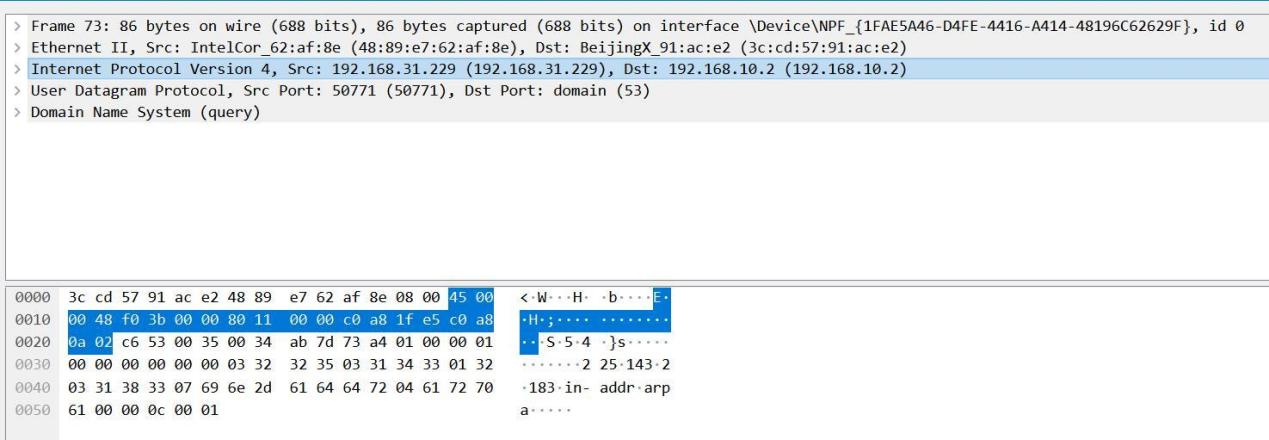
剩余的所有字节均是IP数据报。

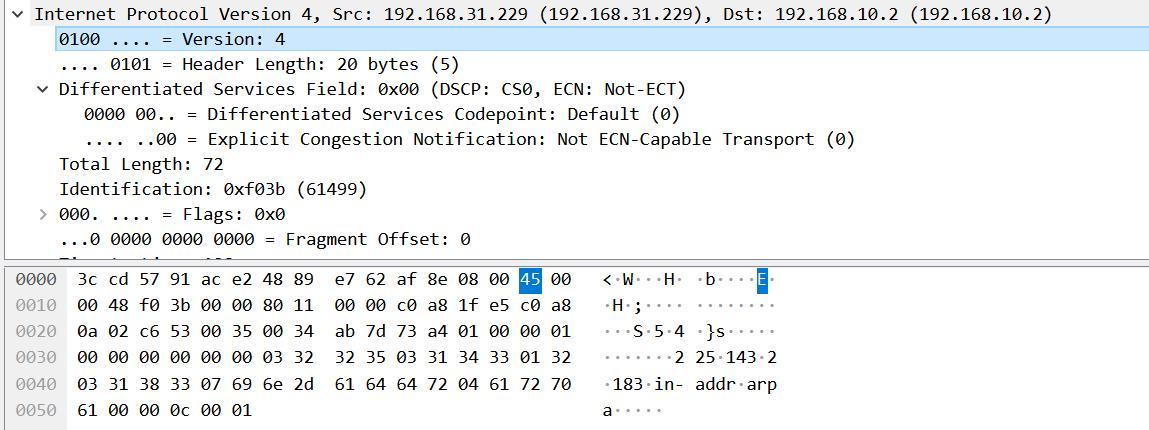
值得一提的是，Wireshark展现给我们的帧中并没有帧检验序列FCS，这是因为

(a) Wireshark抓到的帧，是FCS校验通过的帧，而帧尾的FCS会被硬件去掉，所以没有FCS；

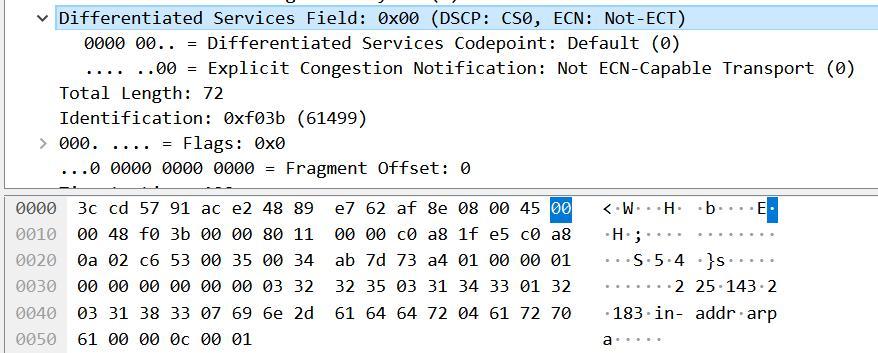
(b) Wireshark不会抓取到FCS校验失败的帧。

### 1.2分析IP数据报首部



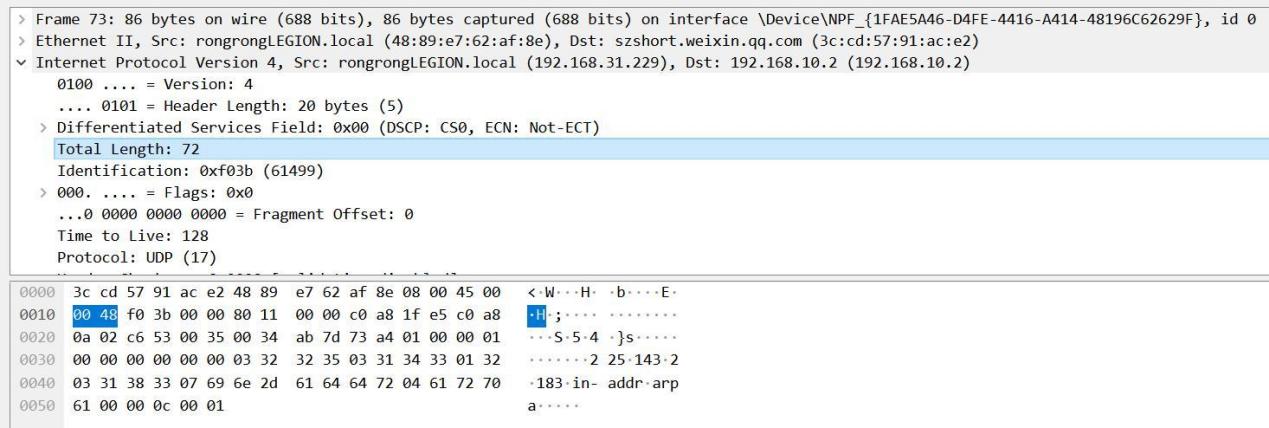


IP数据报的首部是4bit的版本号，以该IP数据报为例，其版本号为4。

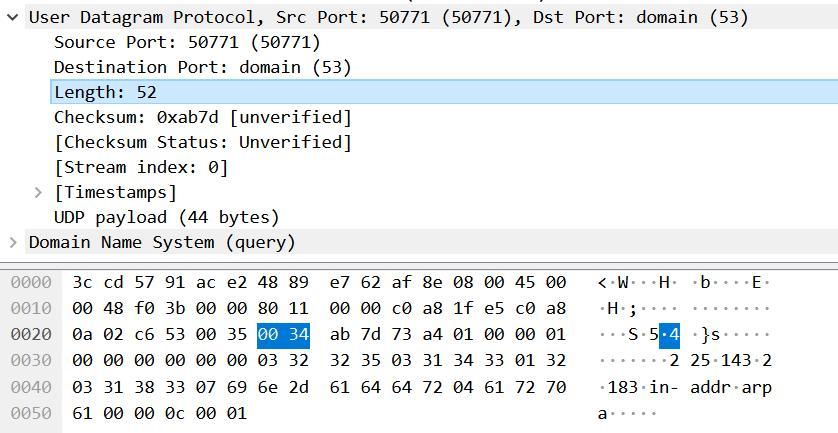


接下来4bit是首部长度，以该IP数据报为例，其首部长度是20字节。

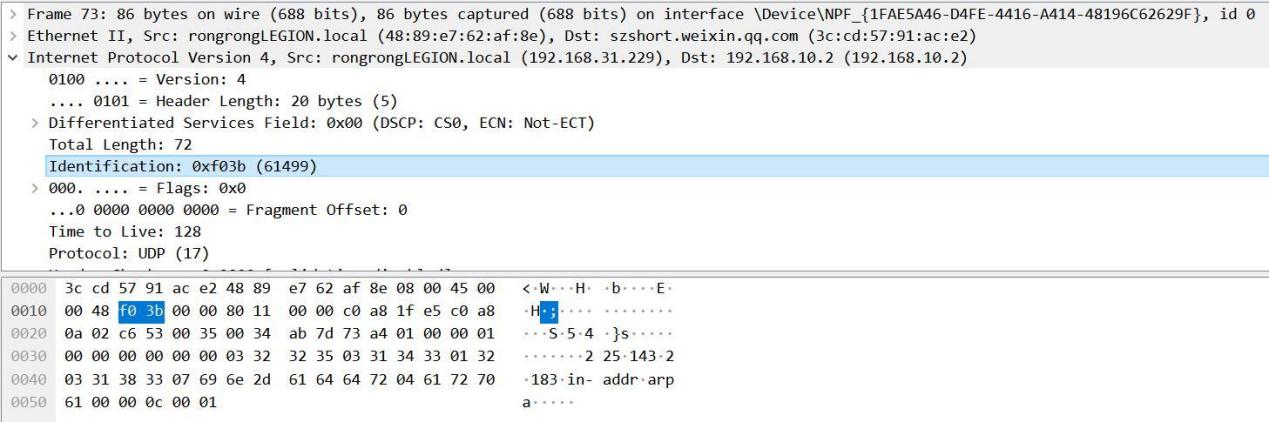
然后是1字节的服务类型，以该IP数据报为例，其服务类型为0。



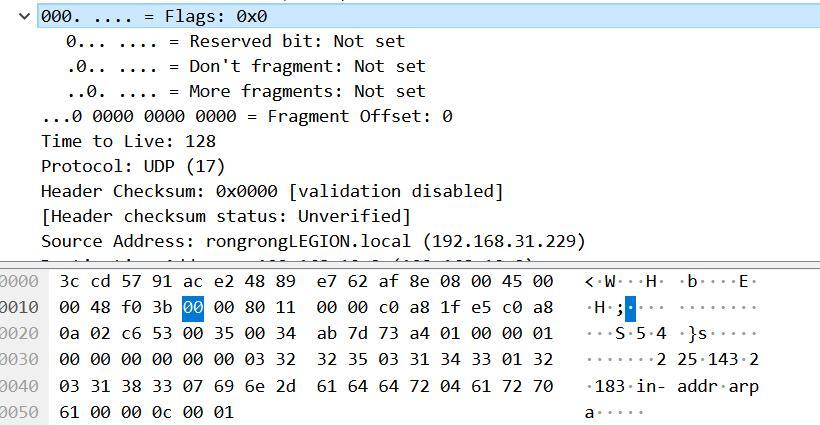
然后是2字节长的总长度，以该IP数据报为例，其总长度为72字节，除去首部固定部分长度20字节，表明可变部分长度有52字节。



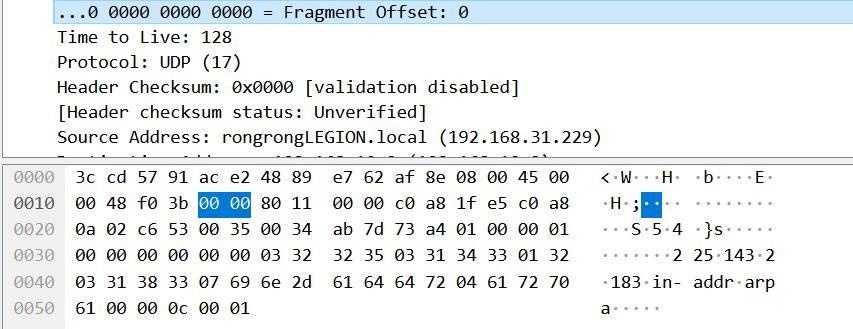
上图显示了可变部分长度为52字节。



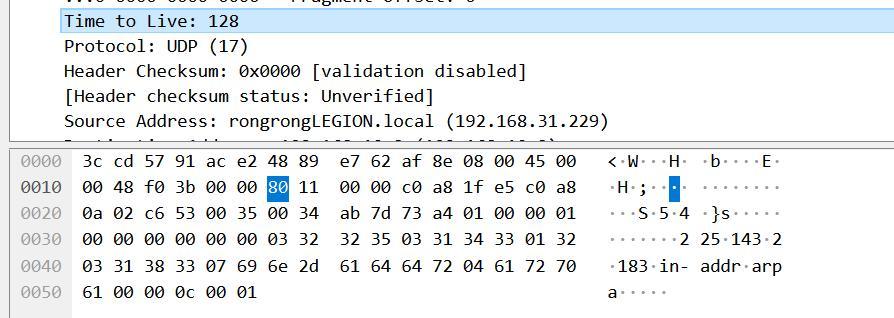
接下来2个字节是标识，以该IP数据报为例，其标识为0xf03b。



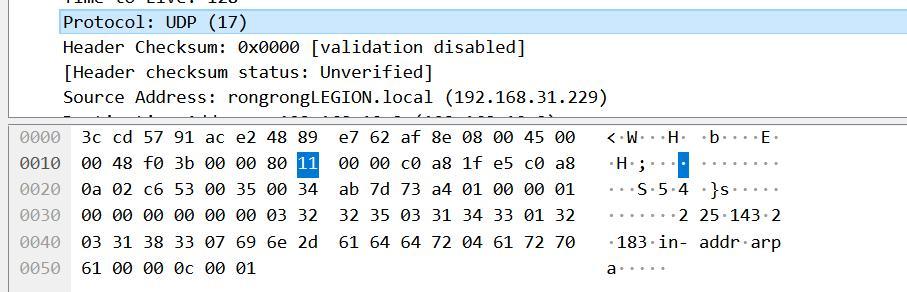
之后3bit是标志位，以该IP数据报为例，其标志位为000。



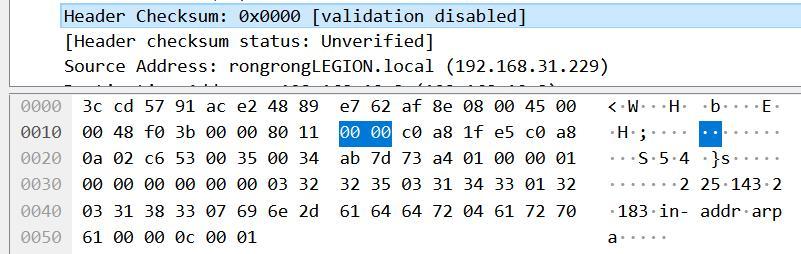
接下来13bit为片偏移，以该IP数据报为例，其片偏移为0。



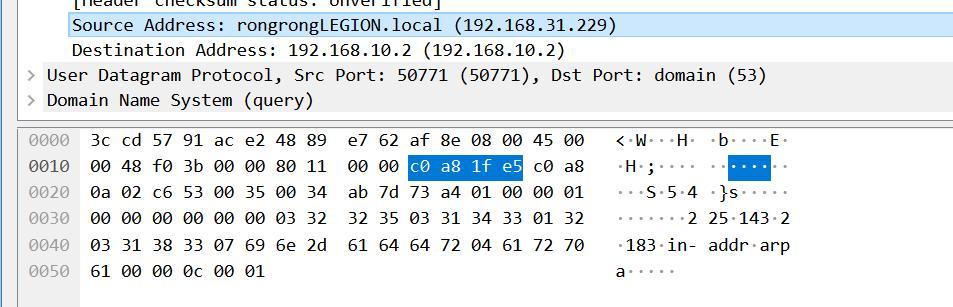
接下来1字节为生存时间(TTL)，以该IP数据报为例，其生存时间(TTL)为128s。



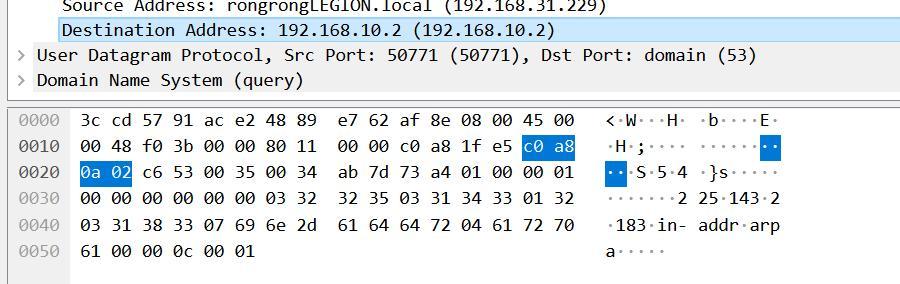
接下来1字节为协议，以该IP数据报为例，其协议为UDP。



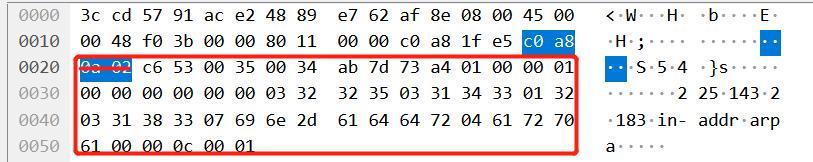
之后的2字节为首部检验和，以该IP数据报为例，其首部检验和为0，保留这个数据报。



接下来4字节为源IP地址，以该IP数据报为例，其源IP地址为192.168.31.229。



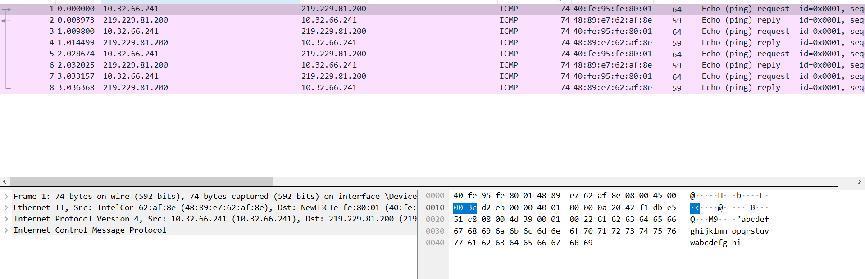
首部固定部分的最后4个字节为目的IP地址，以该IP数据报为例，其目的IP地址为192.168.10.2。

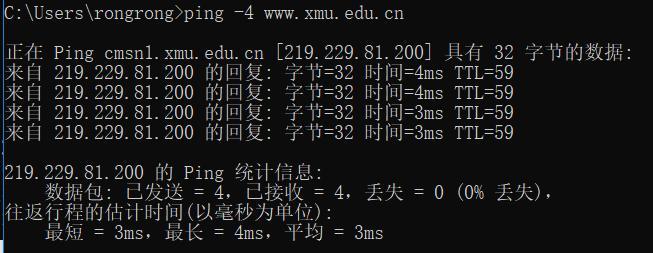


剩余字节均为可变部分。

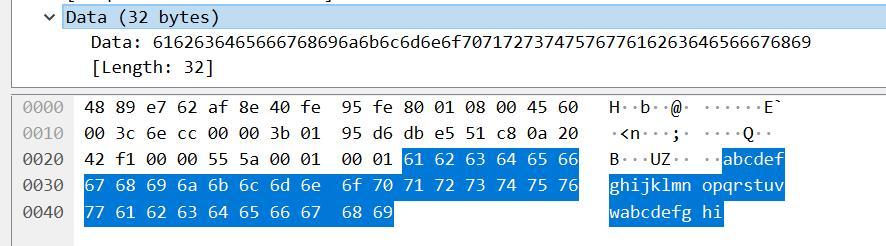
### 1.3观察IP分片

1. ping -4 [www.xmu.edu.cn](http://www.xmu.edu.cn)



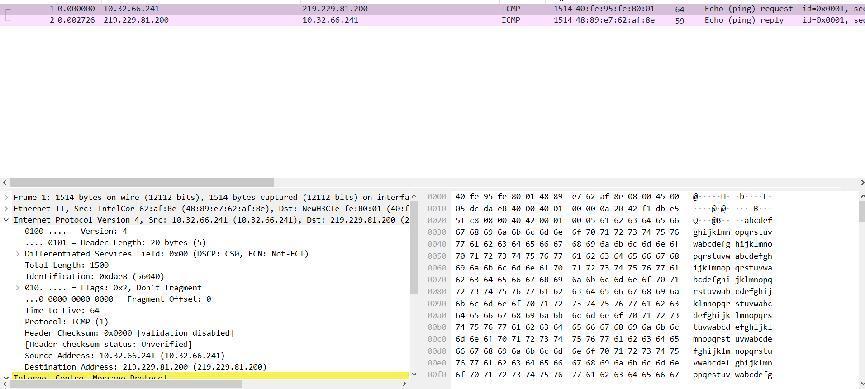


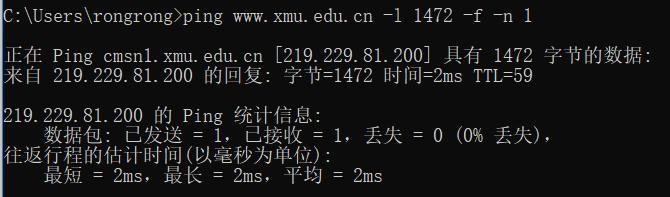
如上图所示，Wireshark抓包数据显示发送了4个数据包、接受了4个数据包；终端窗口显示已发送的数据包个数为4,、已接受的数据包个数为4。



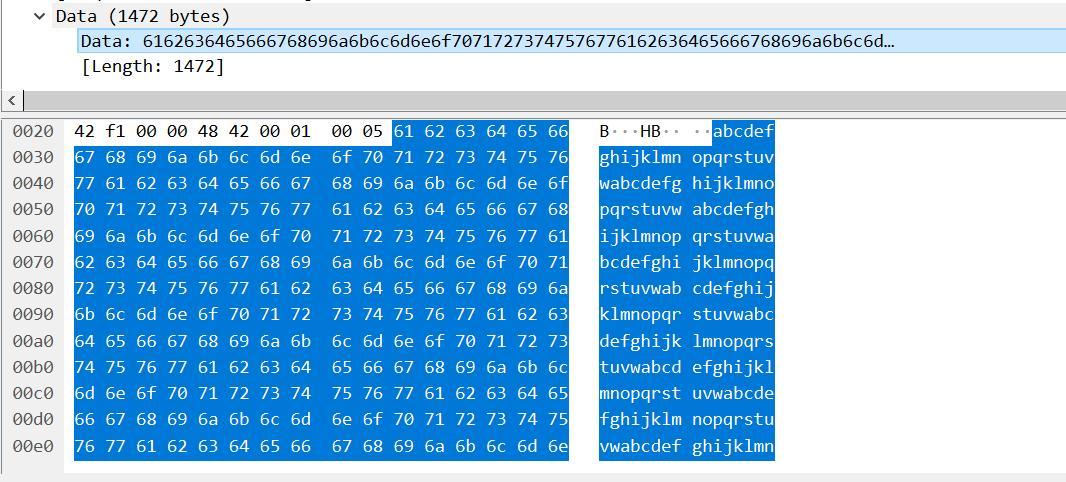
终端窗口显示接收到的字节数为32，ICMP报文数据部分长度为32字节。

1. ping www.xmu.edu.cn -l 1472 -f -n 1



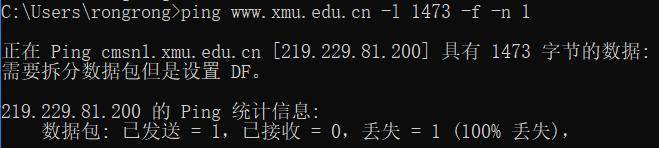


如上图所示，Wireshark抓包数据显示发送了1个数据包、接收了1个数据包；终端窗口显示已发送的数据包个数为1、已接受的数据包个数为1。



终端窗口显示接收到的字节数为1472，ICMP报文数据部分长度为1472字节。

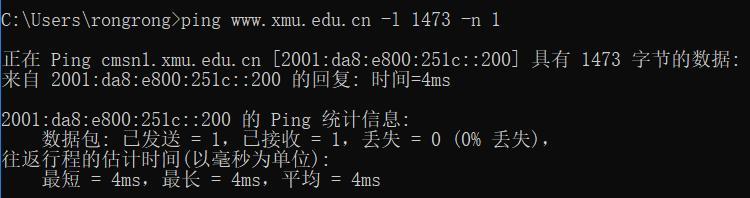
1. ping www.xmu.edu.cn -l 1473 -f -n 1



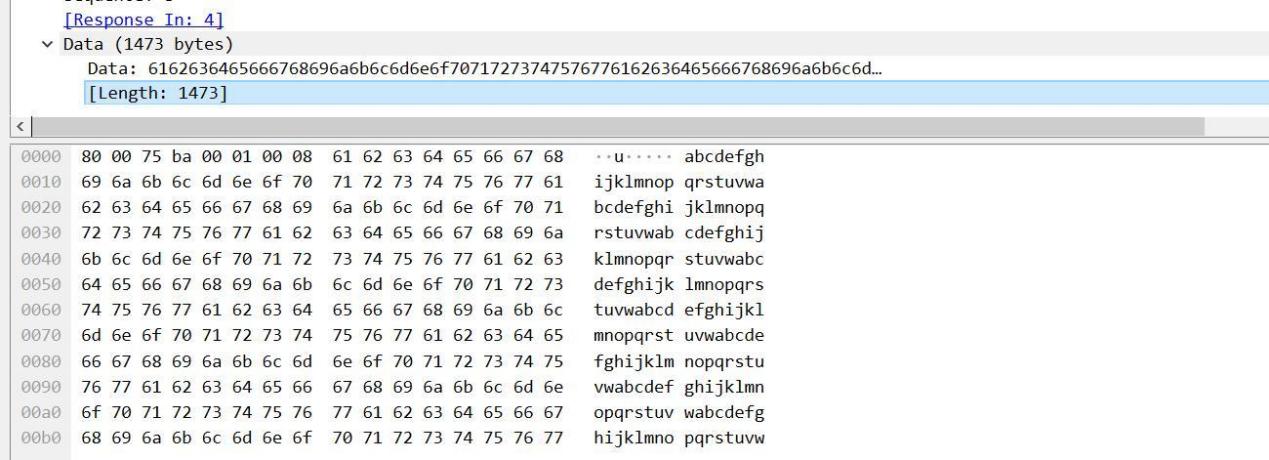
终端显示“需要拆分数据包但是设置DF”，指的是数据包大小超过了网络限定MTU大小，即不分片对超过MTU的长报文无法传输。

1. ping www.xmu.edu.cn -l 1473 -n 1





终端输入“ping [www.xmu.edu.cn](http://www.xmu.edu.cn) -l 1473 -n 1”从www.xmu.edu.cn 请求字节长度为1473字节的报文，并分片传输。Wireshark抓包数据接收到两组报文，报文数据长度为1473字节。

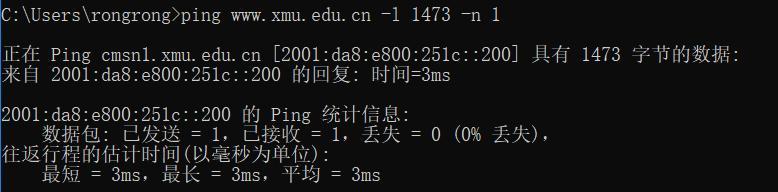


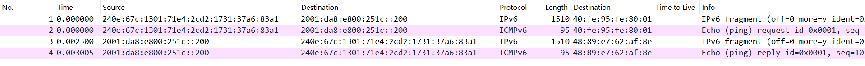
综上，对比以上四条命令：

1. ping -4 [www.xmu.edu.cn](http://www.xmu.edu.cn) 表示从目的IP地址请求默认32字节的报文数据，共4次。
2. ping www.xmu.edu.cn -l 1472 -f -n 1 表示从目的IP地址请求1472字节的报文数据，且设置为不分片，请求1次。
3. ping www.xmu.edu.cn -l 1473 -f -n 1 表示从目的IP地址请求1473字节的报文数据，且设置为不分片，请求1次。
4. ping www.xmu.edu.cn -l 1473 -n 1 表示从目的IP地址请求1473字节的报文数据，且设置为分片，请求1次。

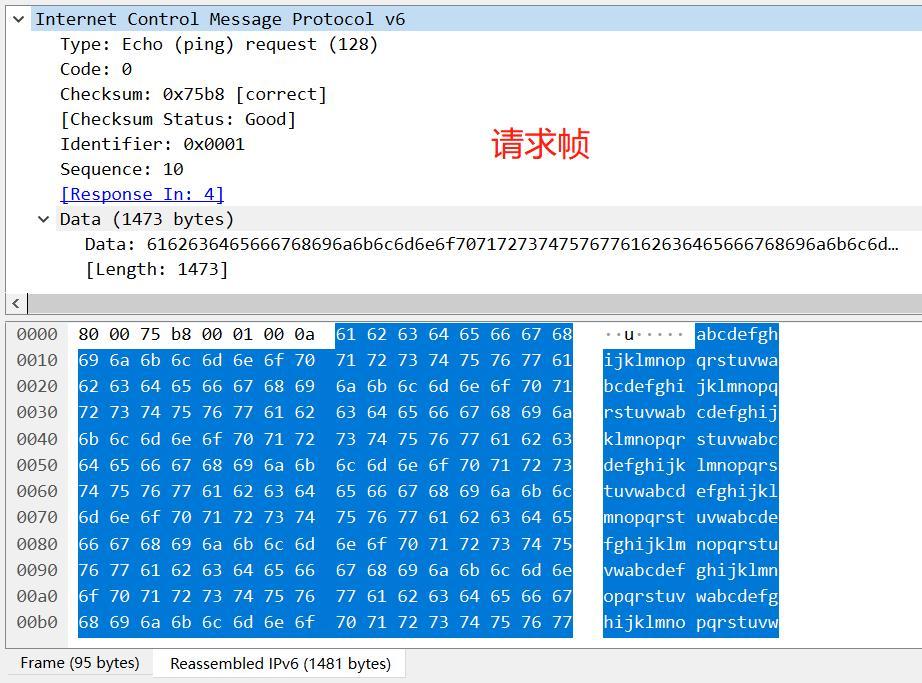
### 1.4 ICMP协议分析（以ping指令为例）

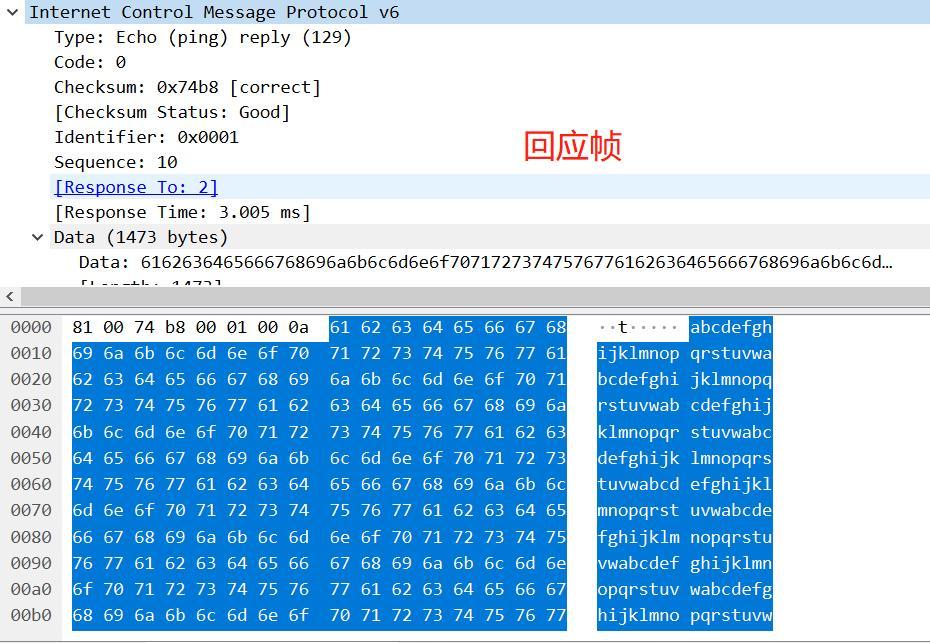
执行一次ping命令会得到一组ICMP请求帧和回应帧。

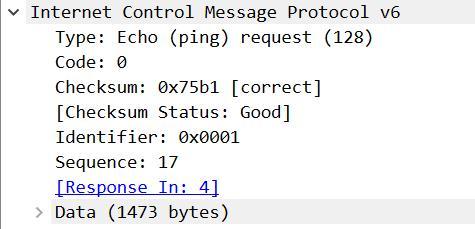


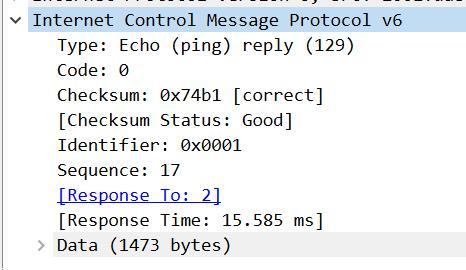


例如，执行上述ping命令，得到一组ICMP请求帧和回应帧。

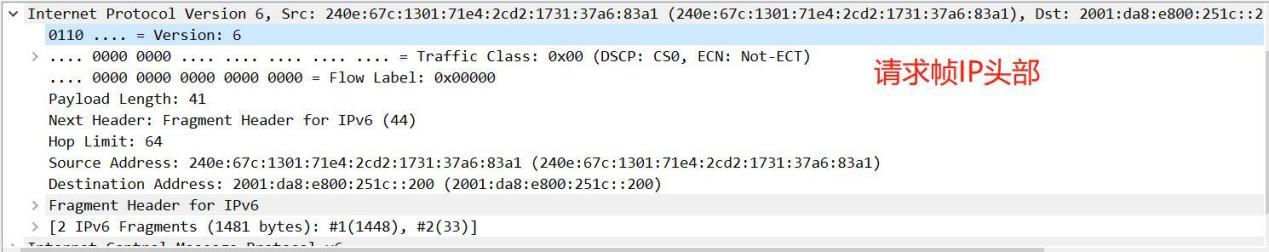


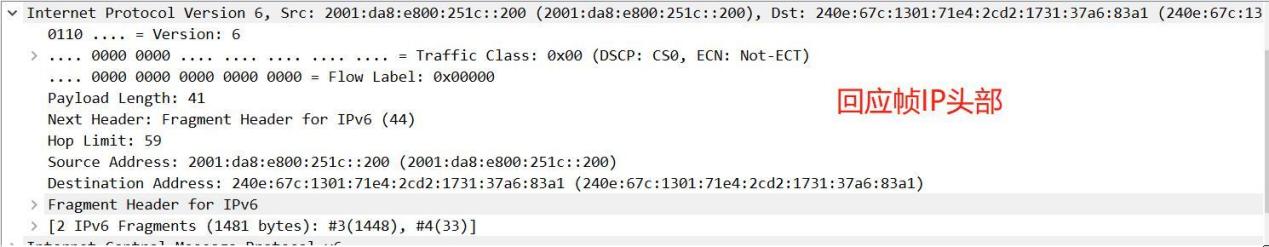
请求帧和回应帧的差别：



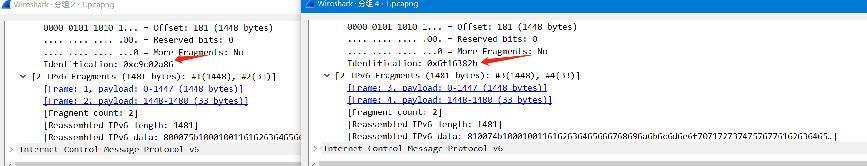


1. 请求帧的Type为：Echo (ping) request (128)；回应帧的Type为：Echo (ping) reply (129)
2. Checksum检验和不同
3. 请求帧[Response In: 4]表示回应帧在分组4；回应帧[Response To: 2]表示请求帧在分组2
4. 回应帧有回应时间
5. 对应IP头部的差别：

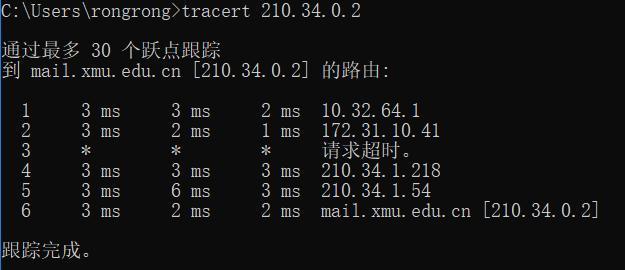


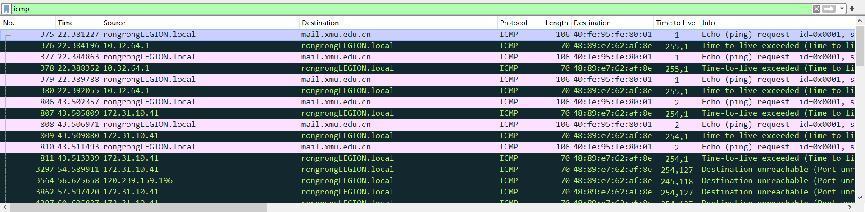


1. 二者的Hop Limit不同，请求帧IP头部的Hop Limit为59；回应帧IP头部的Hop Limit为64
2. 二者的源地址和目的地址对调
3. 二者Fragment Header for IPv6中的Identifacation不同

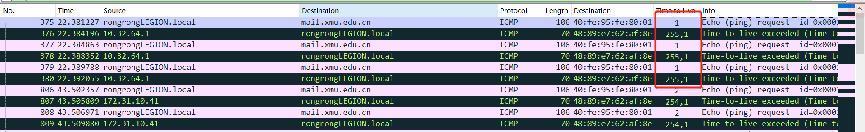


### 1.5 tracert工作原理分析

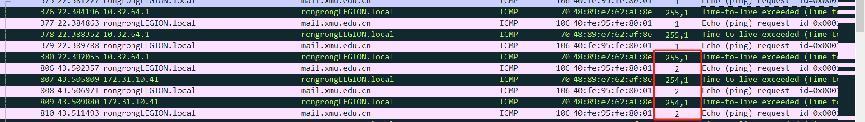




Tracert 命令用 IP 生存时间 (TTL) 字段和 ICMP 错误消息来确定从一个主机到网络上其他主机的路由。

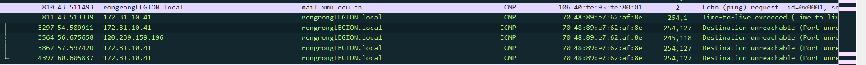


首先，tracert发送出一个TTL是1的IP数据包到目的地，当路径上的第一个路由器收到这个数据包时，它将TTL减1。此时，TTL变为0，所以该路由器会将此数据包丢掉，并送回一个[ICMP time exceeded]消息（包括发IP包的源地址，IP包的所有内容及路由器的IP地址）,如上图中的前3个TTL是1的IP数据包。

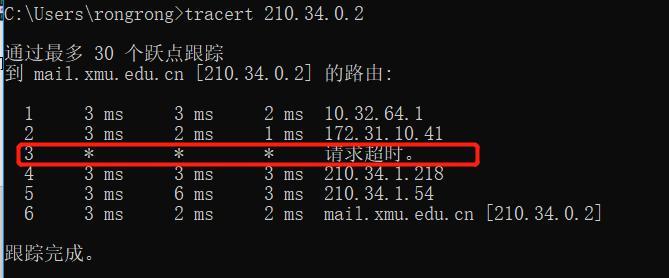


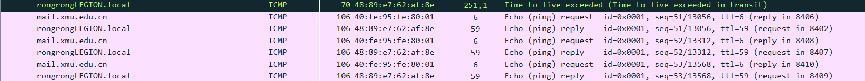
tracert收到这个消息后，便知道这个路由器存在于这个路径上，接着tracert再送出另一个TTL是2 的数据包，发现第2个路由器，如上图中红色框出的3个TTL是2的IP数据包。

tracert每次将送出的数据包的TTL加1来发现另一个路由器，这个重复的动作一直持续到某个数据包抵达目的地。当数据包到达目的地后，该主机则不会送回ICMP time exceeded消息，一旦到达目的地，由于tracert通过UDP数据包向不常见端口(30000以上)发送数据包，因此会收到[ICMP port unreachable]消息，故可判断到达目的地。如下图所示：

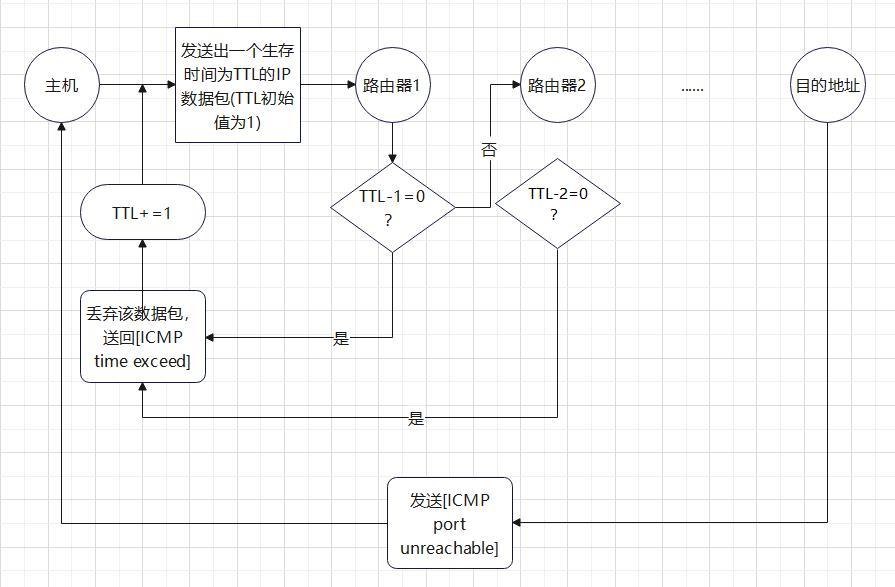


tracert有一个固定的时间等待响应(ICMP TTL到期消息)。如果这个时间过了，它将打印出一系列的\*号表明：在这个路径上，这个设备不能在给定的时间内发出ICMP TTL到期消息的响应。然后，Tracert给TTL记数器加1，继续进行。如下图所示：



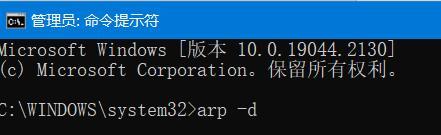


Tracert工作原理示意图如下：

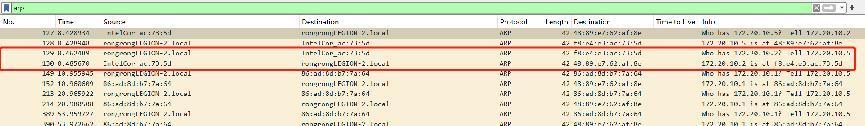


### 1.6 ARP协议分析

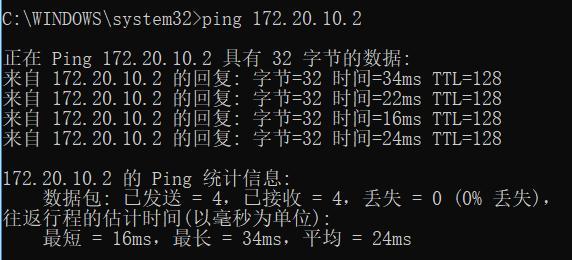
1. 以管理员身份运行终端窗口，运行`arp -d`命令，清空本机已有的ARP缓存



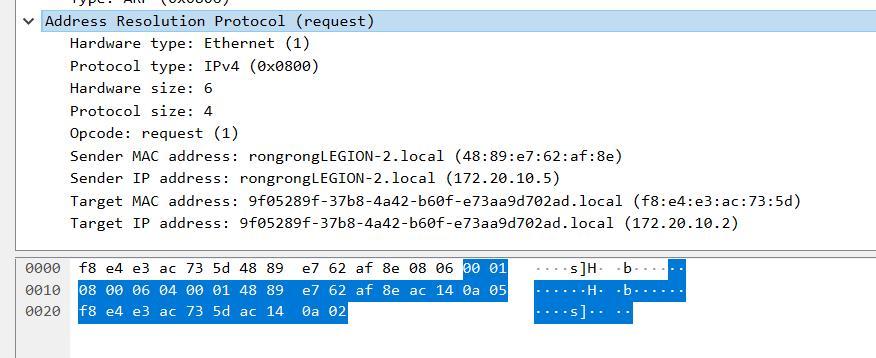
1. 抓包，ping旁边同学的ip



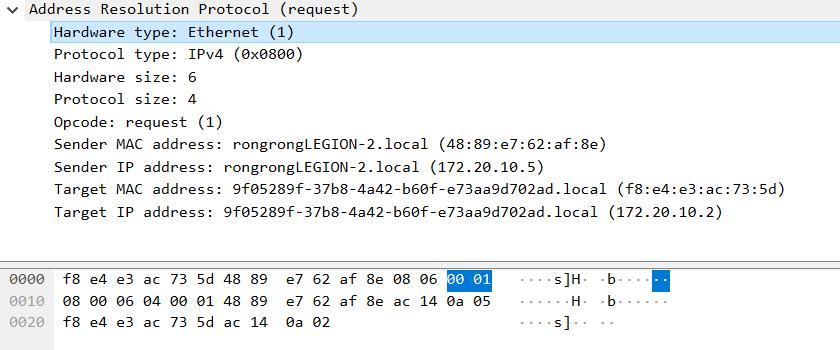
同学的ip为172.20.10.2



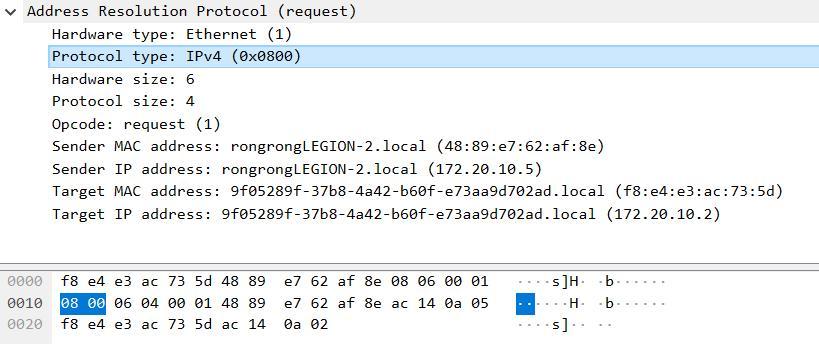
解释ARP报文(请求)字段的含义：



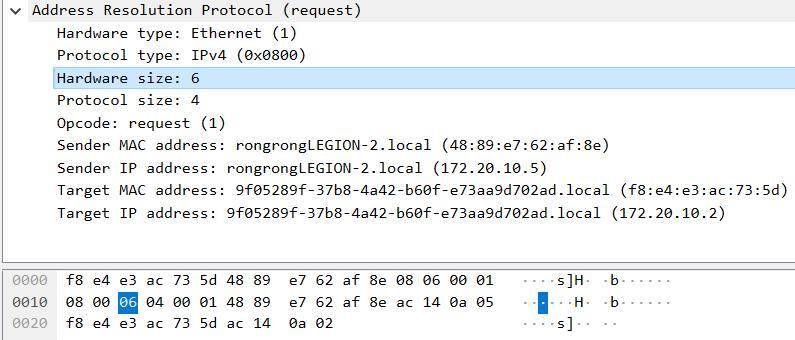
开头2个字节为Hardware type，该ARP报文的Hardware type为Ethernet(1)



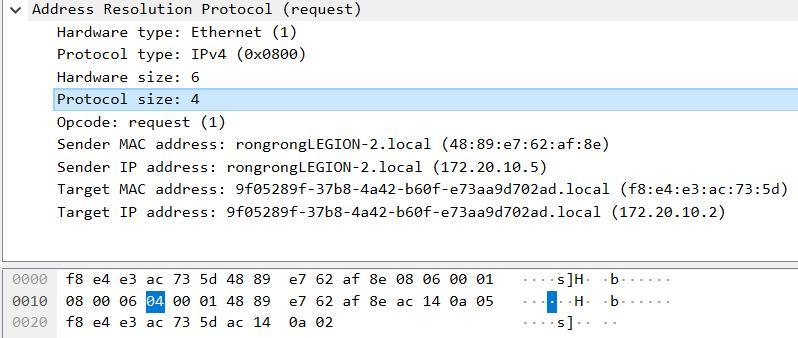
接下来2个字节为Protocol type，该ARP报文为例的Protocol type为IPv4



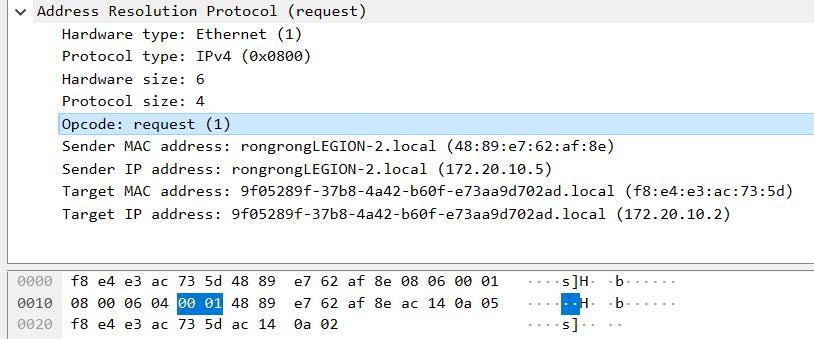
接下来1个字节为Hardware size，该ARP报文的Hardware size为6



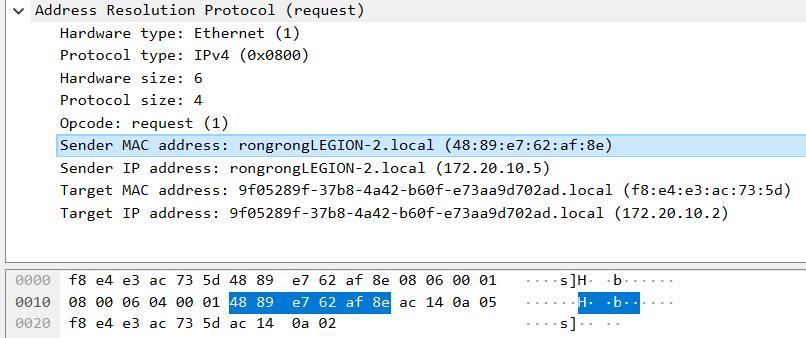
接下来1个字节为Protocol size，该ARP报文的Protocol size为4



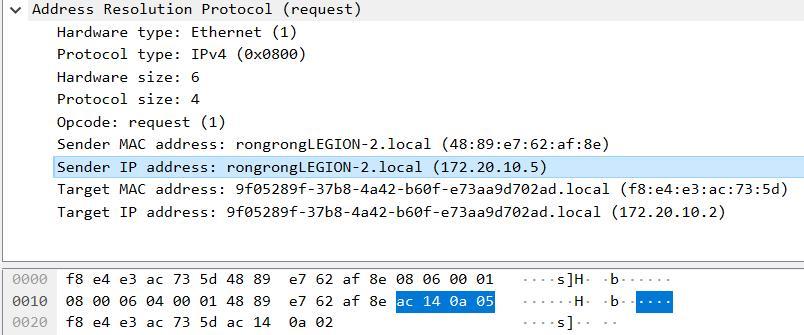
接下来2个字节为Opcode，该ARP报文的Opcode为request(1)



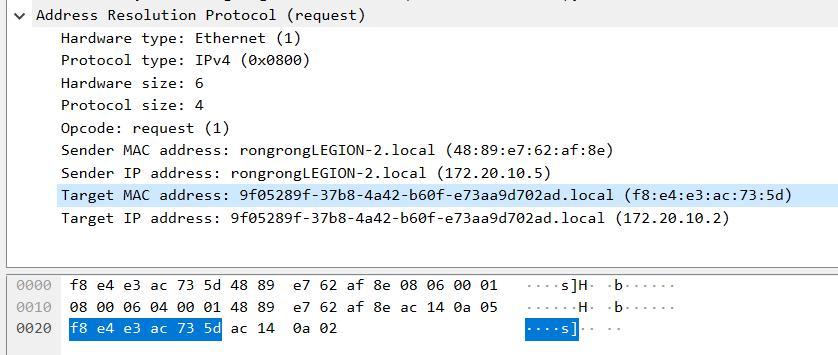
接下来6个字节为Sender MAC address，该ARP报文的Sender MAC address为48:89:e7:62:af:8e



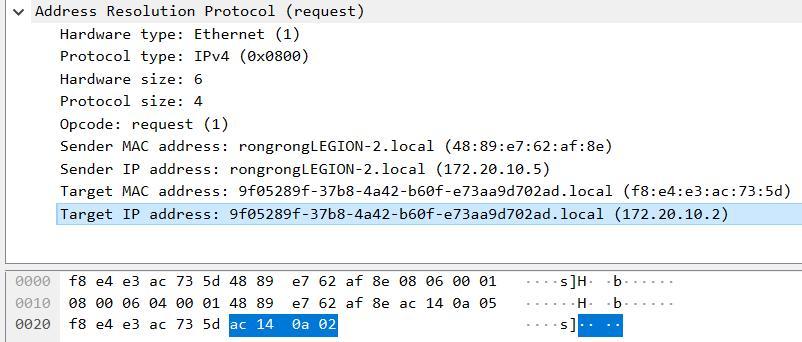
接下来4个字节为Sender IP address，该ARP报文的Sender IP address为172.20.10.5



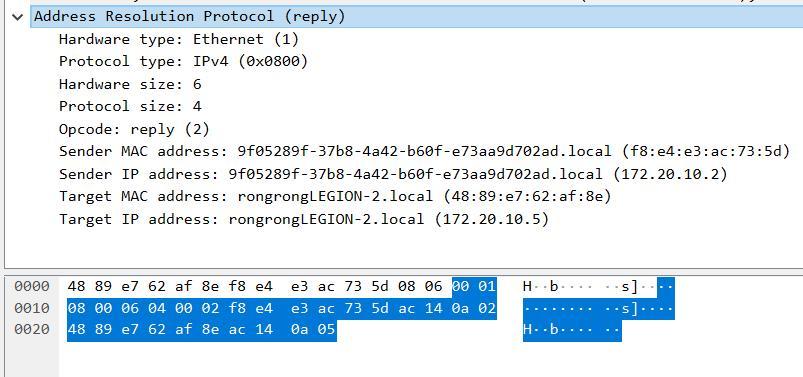
接下来6个字节为Target MAC address，该ARP报文的Sender MAC address为f8:e4:e3:ac:73:5d



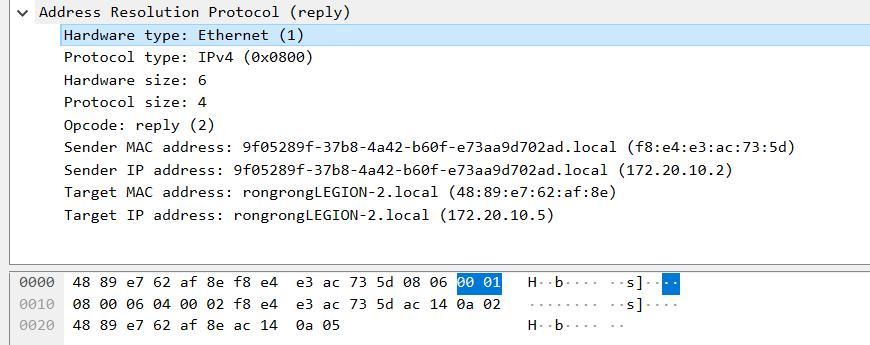
接下来4个字节为Target IP address，该ARP报文的Target IP address为172.20.10.2



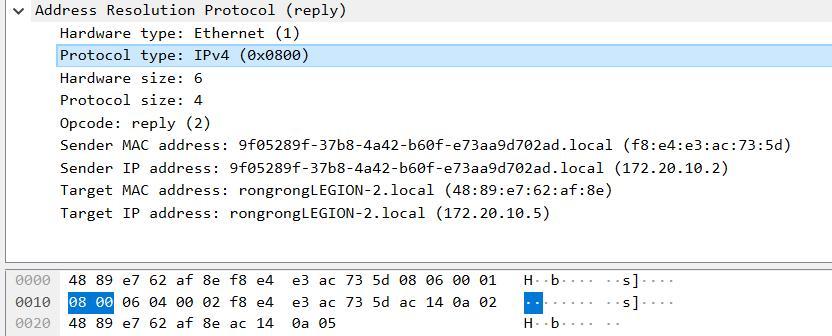
解释ARP报文(响应)字段的含义：



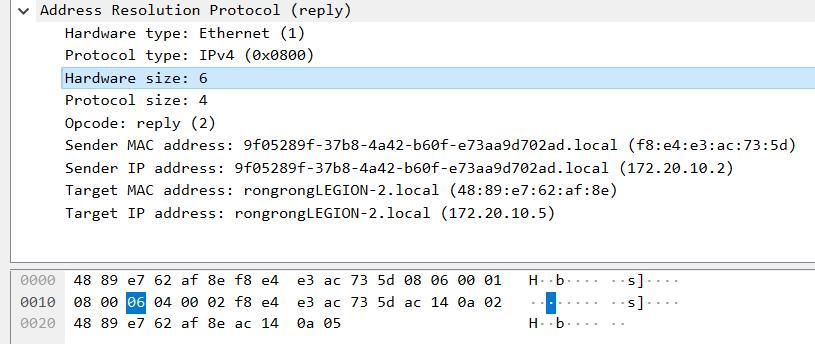
开头2个字节为Hardware type，该ARP报文的Hardware type为Ethernet(1)



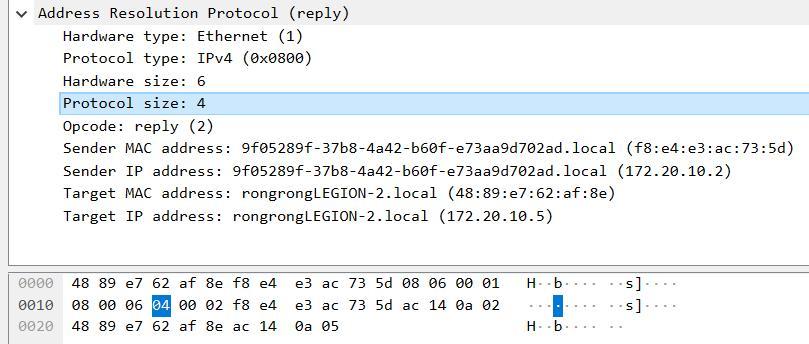
接下来2个字节为Protocol type，该ARP报文Protocol type为IPv4



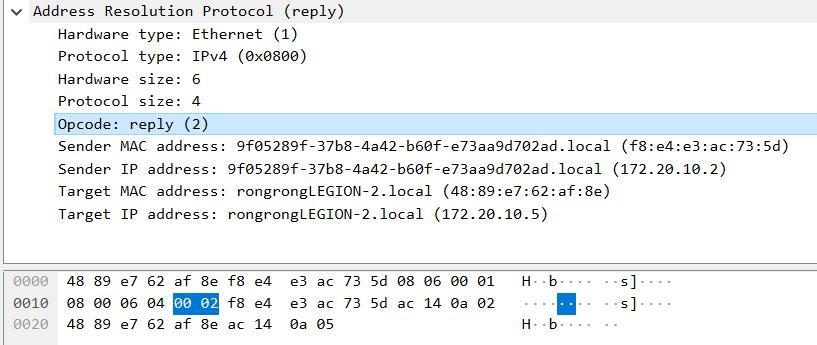
接下来1个字节为Hardware size，该ARP报文Hardware size为6



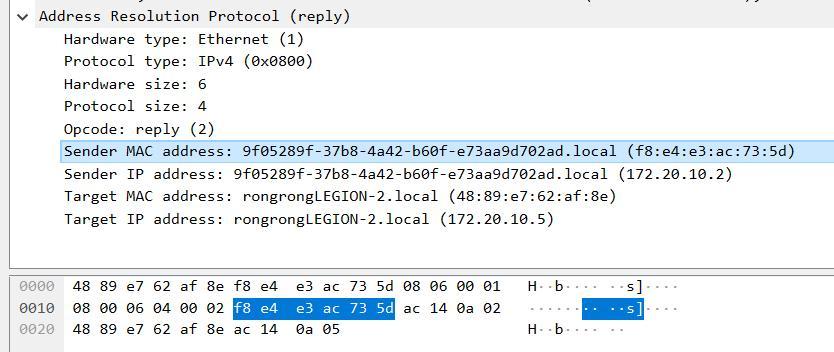
接下来1个字节为Protocol size，该ARP报文Protocol size为4



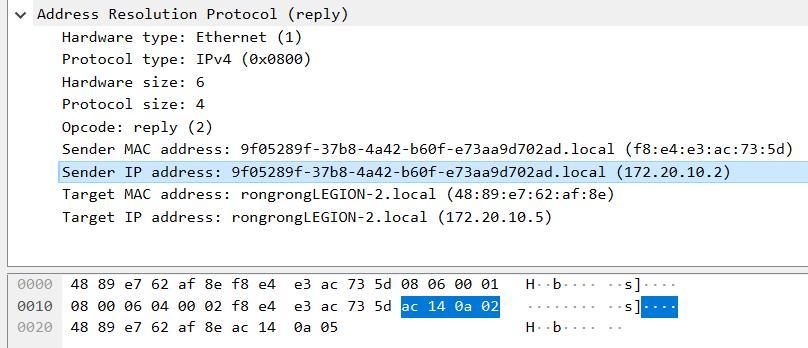
接下来2个字节为Opcode，该ARP报文的Opcode为request(1)



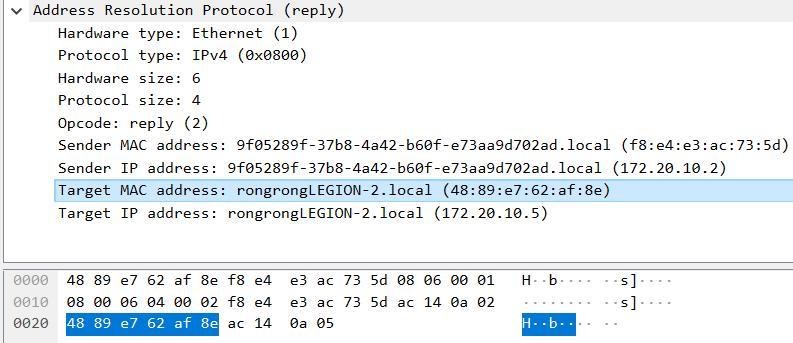
接下来6个字节为Sender MAC address，该ARP报文的Sender MAC address为f8:e4:e3:ac:73:5d



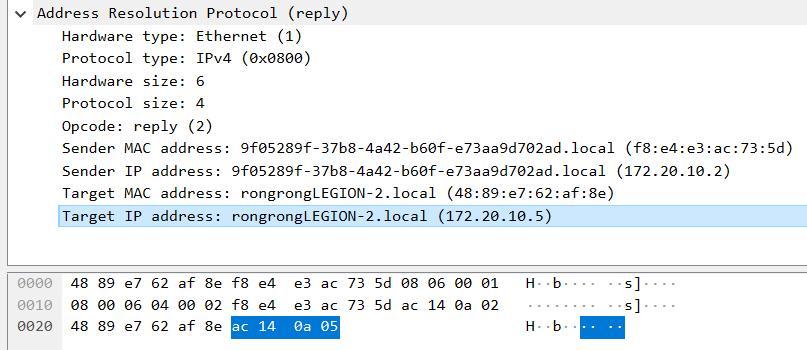
接下来4个字节为Sender IP address，该ARP报文的Sender IP address为172.20.10.2



接下来6个字节为Target MAC address，该ARP报文的Sender MAC address为48:89:e7:62:af:8e



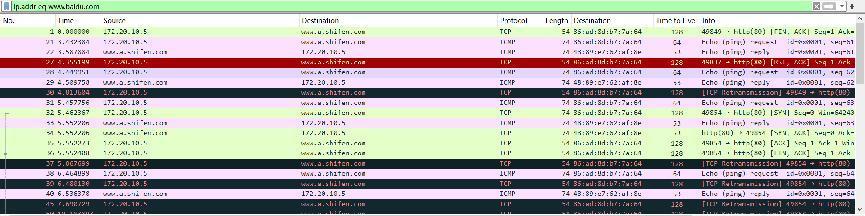
接下来4个字节为Target IP address，该ARP报文的Target IP address为172.20.10.5



1. ping [www.baidu.com](http://www.baidu.com)

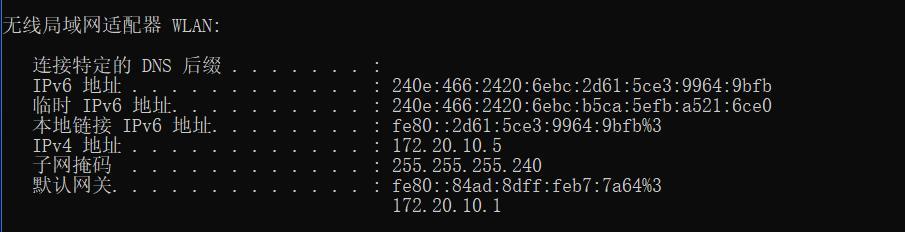


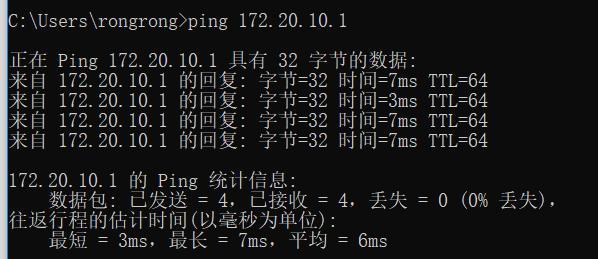
此处，我们发现这里并没有arp报文，但是此处会显示出ICMP类型的IP数据包以及TCP类型的IP数据包，如下图所示：



[www.baidu.com](http://www.baidu.com)并不处于与本地主机的同一个局域网内，我们需要ping网关

默认网关如下图：







但是也没有任何ARP报文。

我们可以结合课本上的工作原理对ping同一局域网下的计算机和局域网外的计算机产生的不同影响进行说明：

对于同一局域网下的计算机：

1. 我们首先清空自己的arp缓存，不知道其他任何一个主机的MAC地址；
2. 我们ping同一局域网下的一台主机，会在本局域网上请求发送一个ARP请求分组；
3. 得到目的主机的MAC地址后，本主机的ARP高速缓存得以更新，将目的主机的IP地址和MAC地址对应上加入ARP高速缓存。

而对于局域网外的计算机：

ARP用于解决同一个局域网上的主机或路由器的IP地址和MAC地址的映射问题，如果要找的主机和源主机不在同一个局域网上，源主机就无法解析出另一个局域网上的主机的MAC地址。

## 任务2: 捕获和分析802.11数据

### **2.1**搭建实验环境

实验环境：Ubuntu 64位虚拟机环境 + USB无线网卡

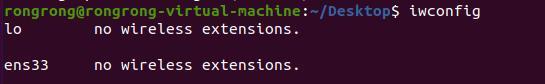
1. 打开终端，安装Wireshark软件：

`sudo apt install wireshark`

`sudo apt install wireshark-gtk`

1. 查询网卡状态：

`iwconfig`

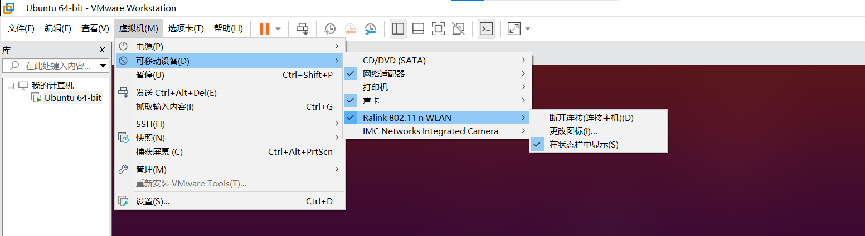


捕获802.11帧需要设置网卡为监控模式（即monitor mode，非混杂模式），但是构建的虚拟机里显示的如上两个网卡都是虚拟机内设的，都不可以设置为监控模式（但是可以设置为混杂模式），所以需要将本地win10中的网卡连接到虚拟机中。

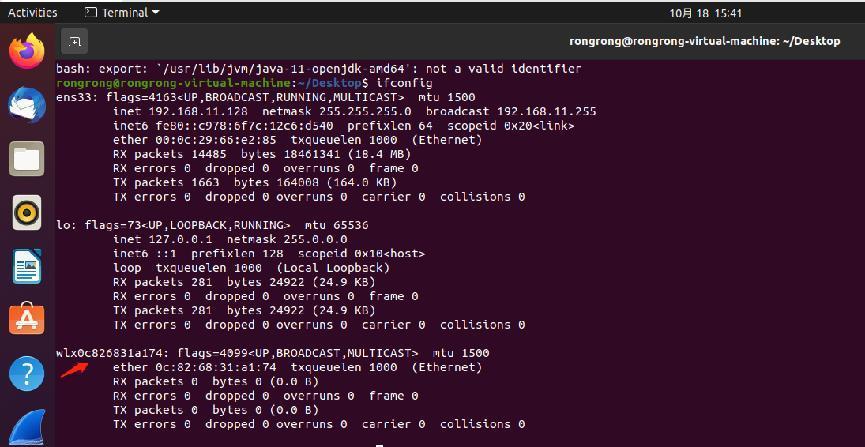
### **2.2**构建无线环境，捕获无线数据包、分析802.11数据

构建无线环境：

1. 将USB无线网卡插到电脑机箱上
2. 打开虚拟机里的Ubuntu 64-bit系统
3. 更改设置，使网卡连接到虚拟机上

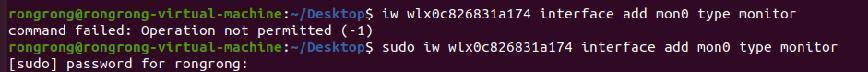


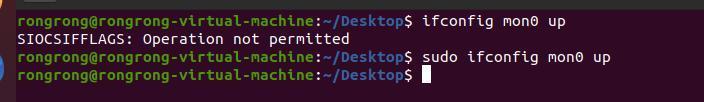
打开终端，输入ifconfig，看到多了一个wlx0c826831a174



无线网卡配置：

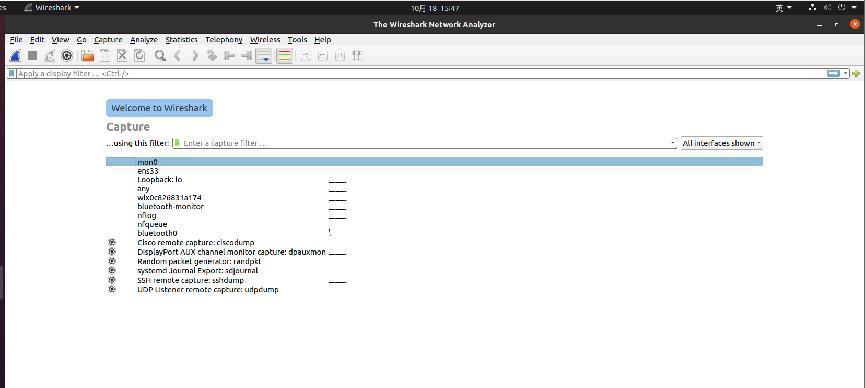
1. 新建一个虚拟网卡，并将其修改为监听模式



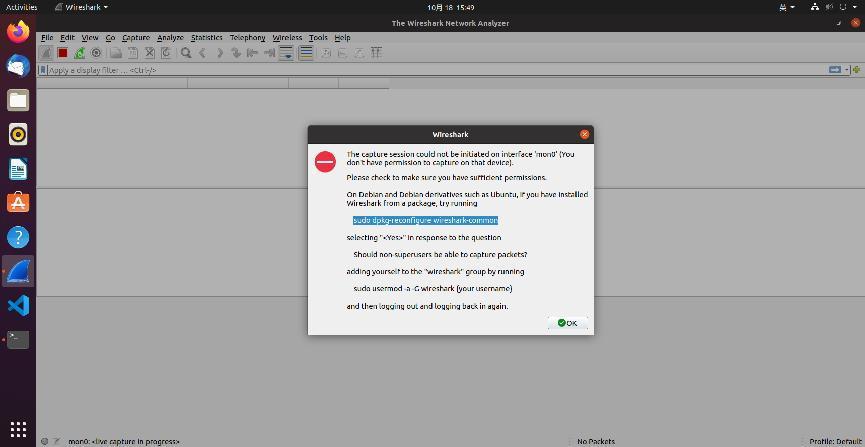


1. 终端输入“wireshark”，打开Wireshark软件

可以看到“mon0”，如下图所示：



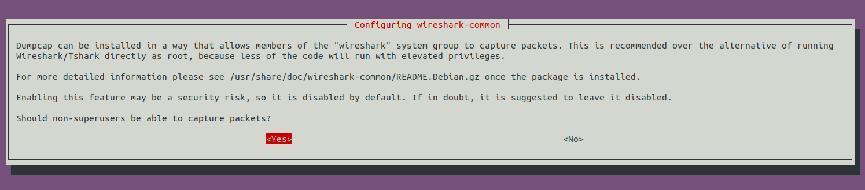
但是这里我们可以并没有看到有上下起伏的折线，应该是哪里有问题，双击尝试一下能否捕捉，果然不行，显示如下：



在终端输入一下指令：

` sudo dpkg-reconfigure wireshark-common `

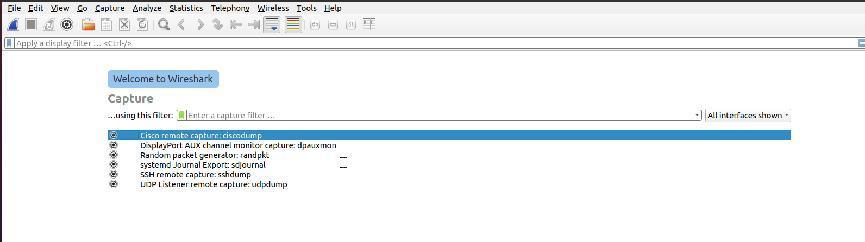




选择yes

再次打开wireshark

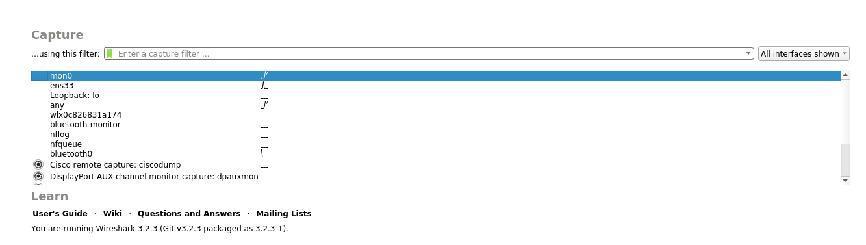
发现什么端口都没有了，只剩下蓝牙端口，于是重新安装Wireshark。



使用sudo命令打开Wireshark

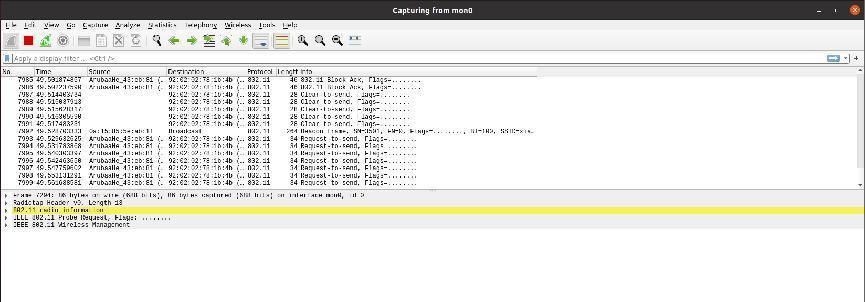


可以看到mon0有上下起伏的波形，如下图所示：



捕获和分析802.11数据：

双击mon0开始捕获，即可捕获到802.11数据



捕获到的802.11数据如上图所示。

## 任务3: 探索Wireshark更多功能和其它抓包工具(选做)

### 探索Wireshark更多功能

1. 数据流追踪

先登录一个免费的图床，我们这里以“postimage”为例，其域名为“postimages.org”

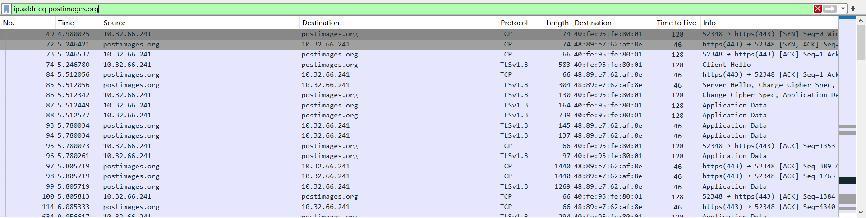
将Wireshark的显示过滤器设置为“ip.addr eq postimages.org”，如下图所示：



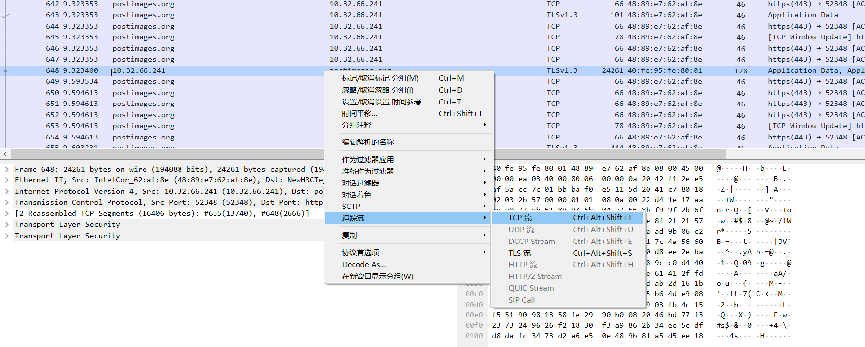
我们选择一张图片上传，如下图：



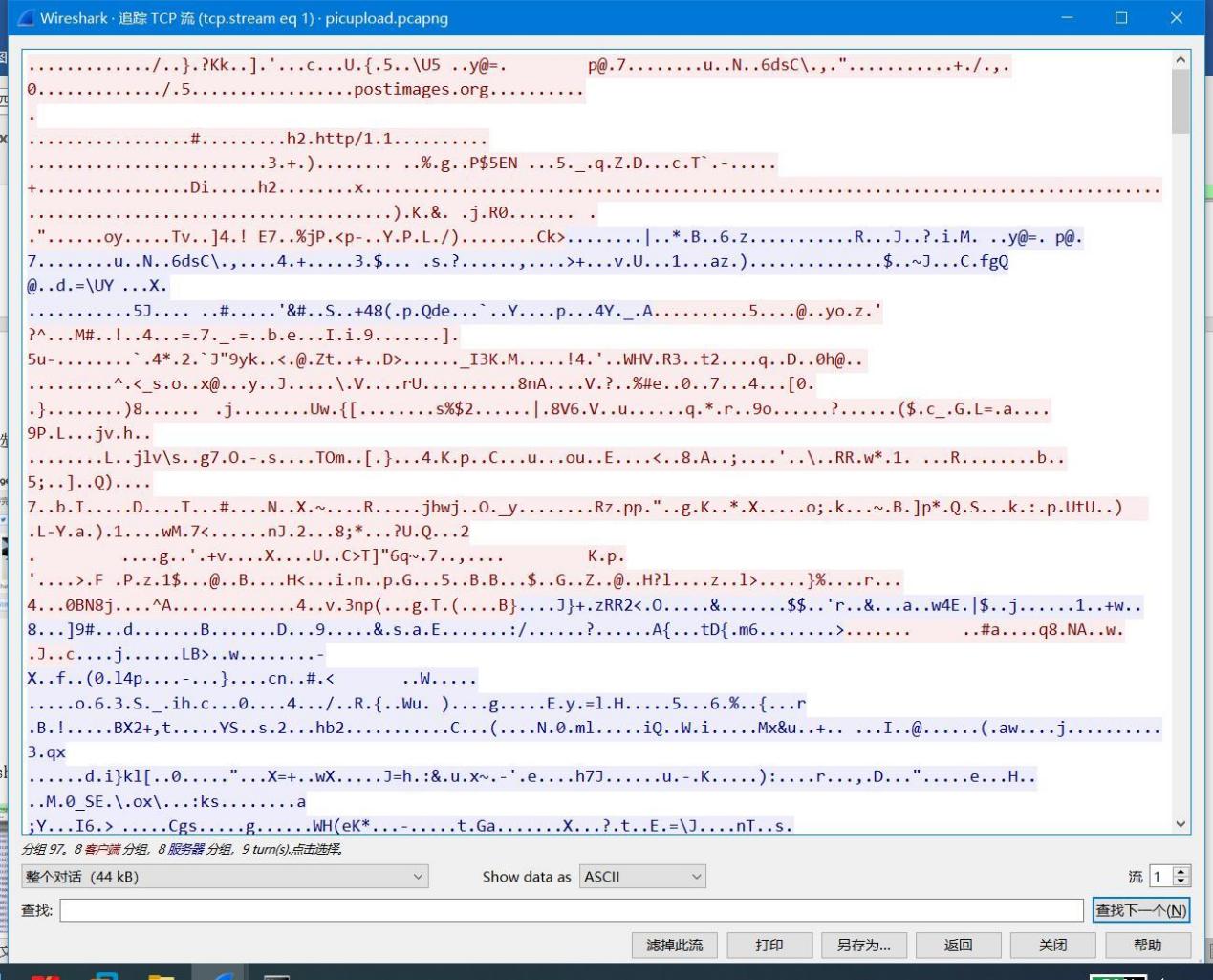
Wireshark抓包结果如下图所示：



这里我们尝试追踪TCP流，如下图所示：

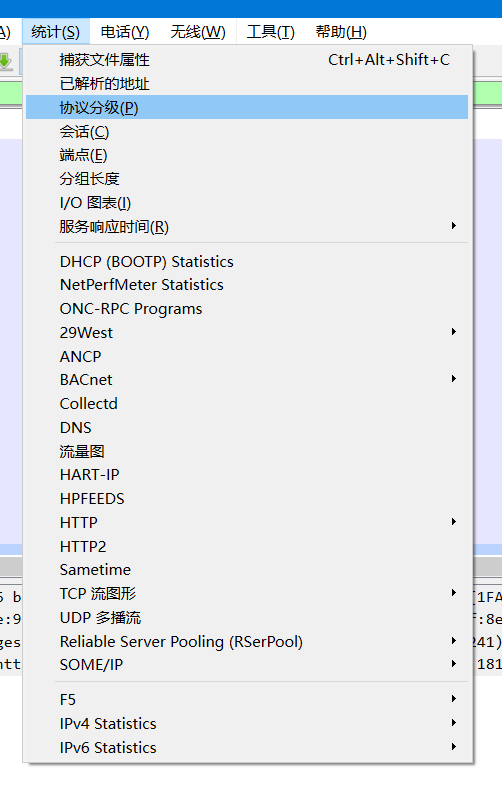


如下图，红色流为源到目的地址，蓝色为目的地址到源：

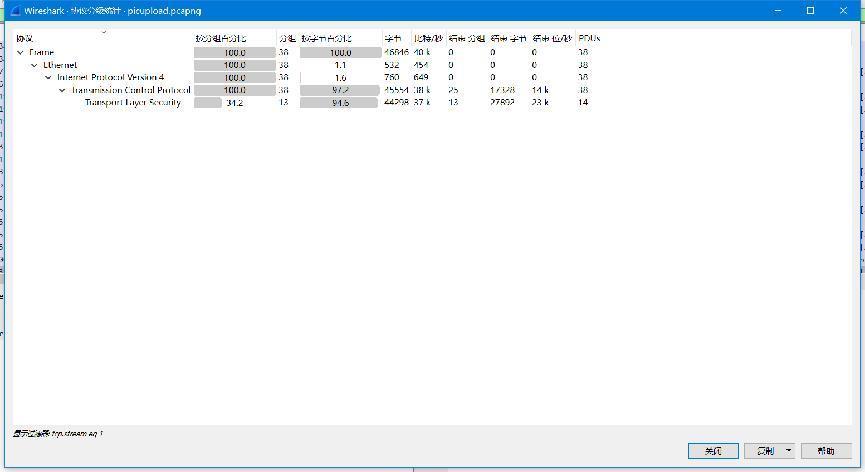


但是我的这里显示乱码，仍未解决……

1. 协议分层统计

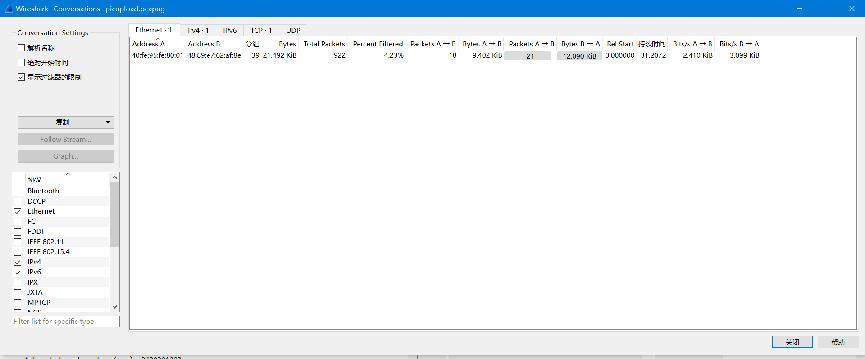


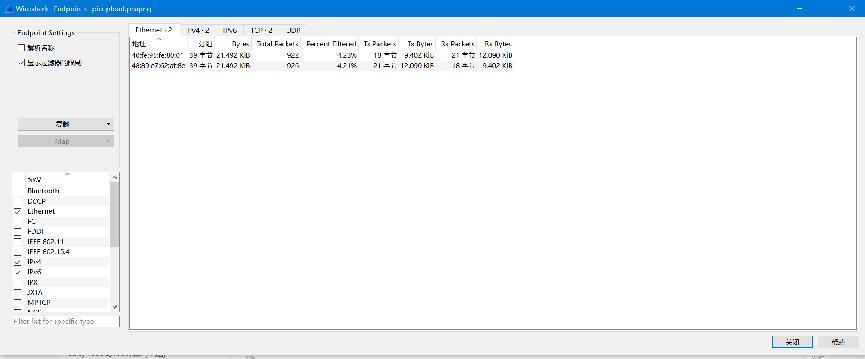
协议分级统计如下图所示：



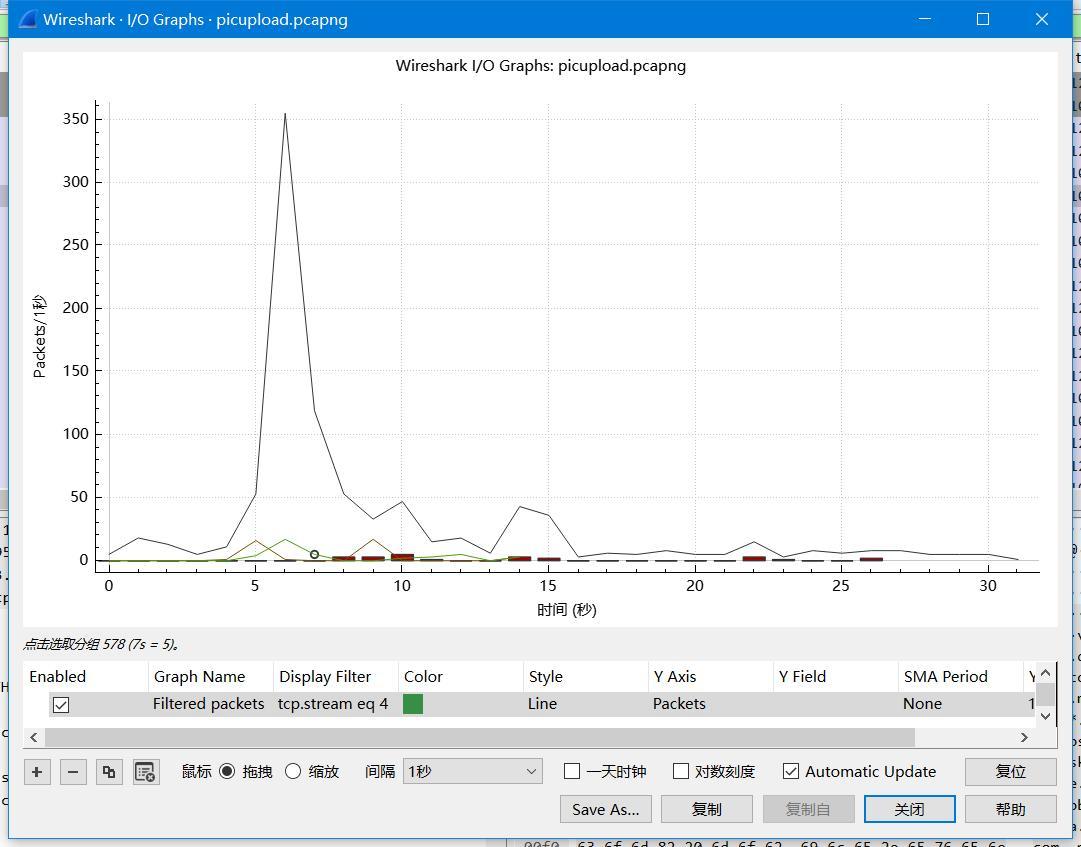
可以看到按分组百分比有34.2%的数据包是TSL协议。

1. 网络节点和会话统计功能

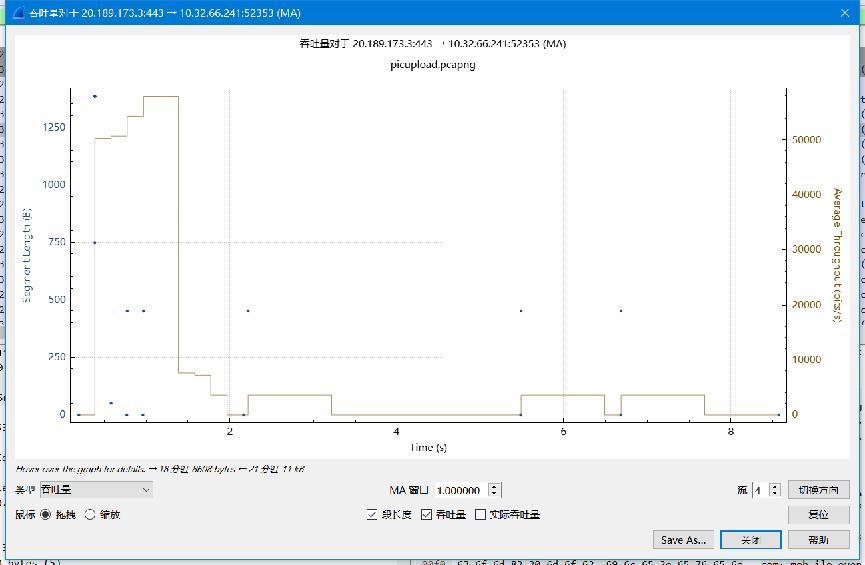




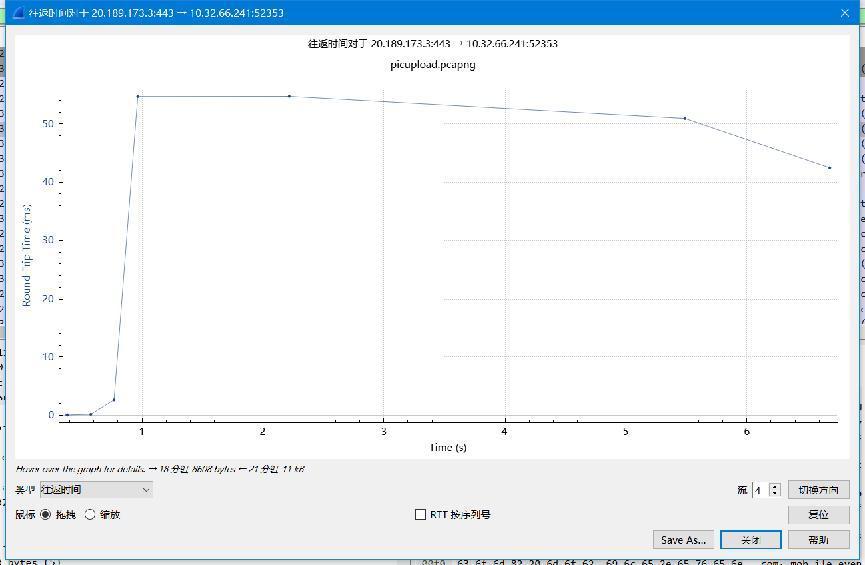
1. IO图表



1. 吞吐量



1. 往返时间



# 实验小结

通过本次实验，我学会了一些使用wireshark抓包的基本操作，并自行探索了wireshark更强大的功能。对一些报文如IPv4、IPv6、ARP等有了更深层、更直观的理解。