# 利用wireshark分析HTTP协议

1. 实验目的

分析HTTP协议

1. 实验环境

与因特网连接的计算机，操作系统为Windows，安装有Wireshark、IE等软件。

1. 实验步骤
2. 利用Wireshark俘获HTTP分组
3. 在进行跟踪之前，我们首先清空Web 浏览器的高速缓存来确保Web网页是从网络中获取的，而不是从高速缓冲中取得的。之后，还要在客户端清空DNS高速缓存，来确保Web服务器域名到IP地址的映射是从网络中请求。在WindowsXP机器上，可在命令提示行输入ipconfig/flushdns完成操作。
4. 启动Wireshrk 分组俘获器。
5. 在Web 浏览器中输入：<http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html>
6. 停止分组捕获。

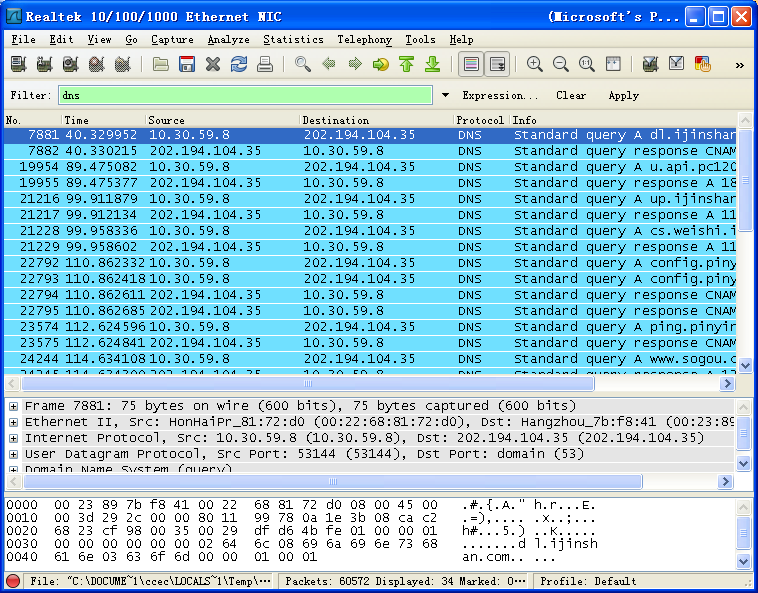


图1：利用Wireshark俘获的HTTP分组

浏览Web页面经过如下三个过程：

1. DNS解析

在URL http://gaia.cs.umass.edu中，gaia.cs.umass.edu 是一个具体的web 服务器的域名。最前面有两个DNS分组。第一个分组是将域名gaia.cs.umass.edu转换成为对应的IP 地址的请求，第二个分组包含了转换的结果。这个转换是必要的，因为网络层协议——IP协议，是通过点分十进制来表示因特网主机的，而不是通过gaia.cs.umass.edu这样的域名。当输入URL http://gaia.cs.umass.edu时，将要求Web服务器从主机gaia.cs.umass.edu上请求数据，但首先Web浏览器必须确定这个主机的IP地址。

1. TCP连接建立

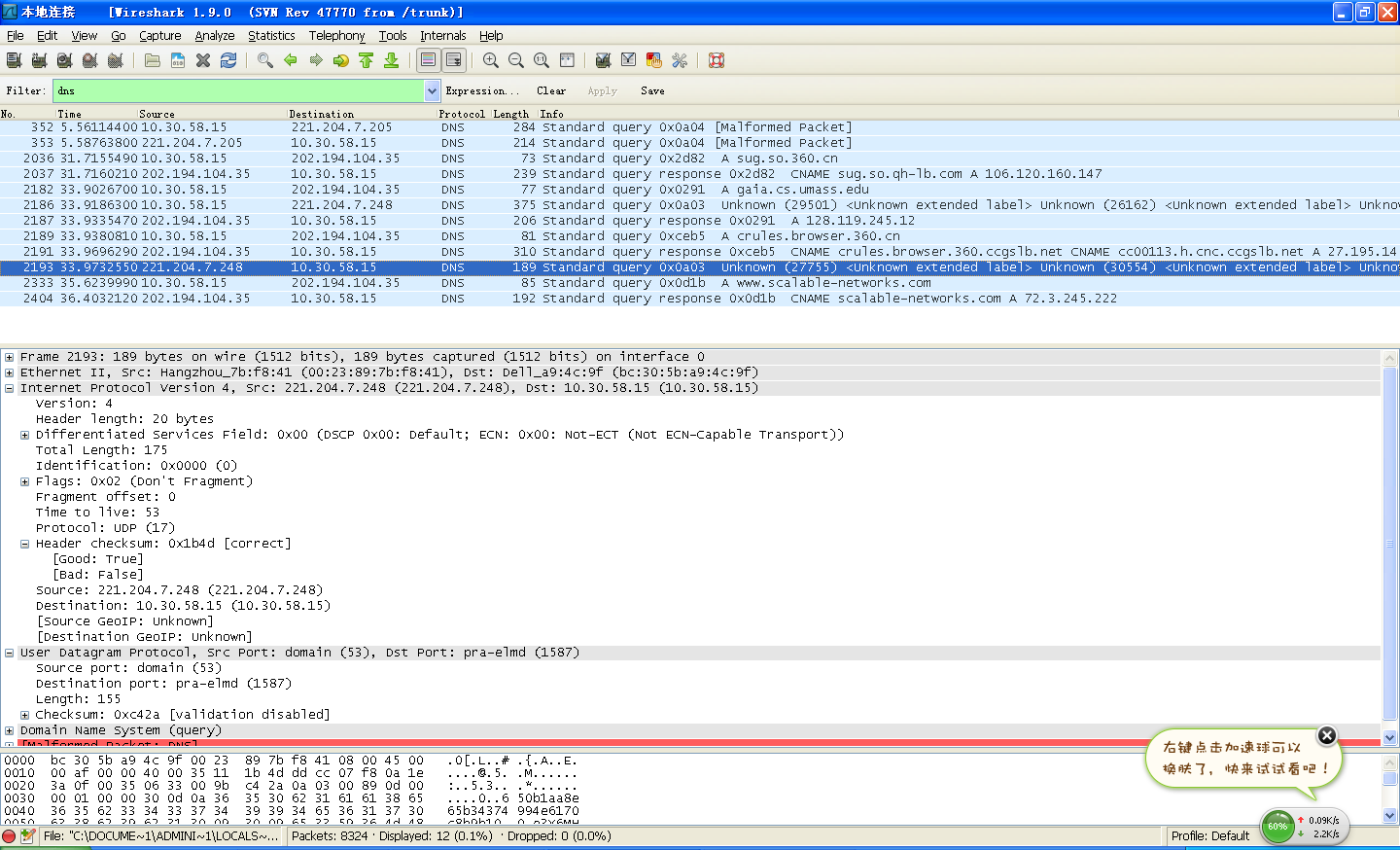
随着转换的完成，Web浏览器与Web服务器建立一个TCP连接。

1. HTTP交互

Web 浏览器使用已建立好的TCP连接来发送请求“GET/HTTP/1.1”。这个分组描述了要求的行为（“GET”）及文件（只写“/”是因为我们没有指定额外的文件名），还有所用到的协议的版本（“HTTP/1.1”）。

分析截获的结果，回答下列问题：

**粘贴**前面的地址解析的两个DNS分组

****

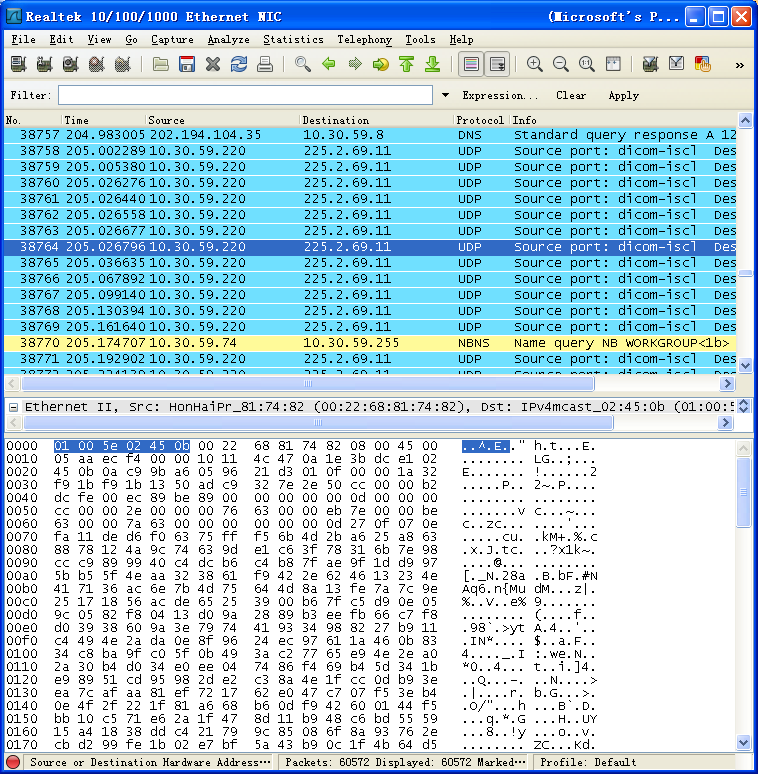
**1)DNS协议主要使用UDP封装还是TCP封装？**

**\_\_\_TCP封装\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_**

**2) DNS查询消息的IP地址是\_\_\_\_221.204.7.205\_\_\_\_\_\_\_\_\_\_\_\_\_\_？你默认的本地DNS服务器的IP地址是\_\_\_10.30.58.15\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_？**

**3)DNS查询消息的目的端口是\_\_\_\_domain(53)\_\_\_\_\_\_\_\_\_\_\_\_\_\_？DNS*响应消息*的源端口是\_\_\_\_\_\_20430\_\_\_\_\_\_\_\_\_\_\_\_？**

粘贴Web浏览器与 Web服务器建立一个TCP连接的三次握手报文



2、HTTP GET/response交互

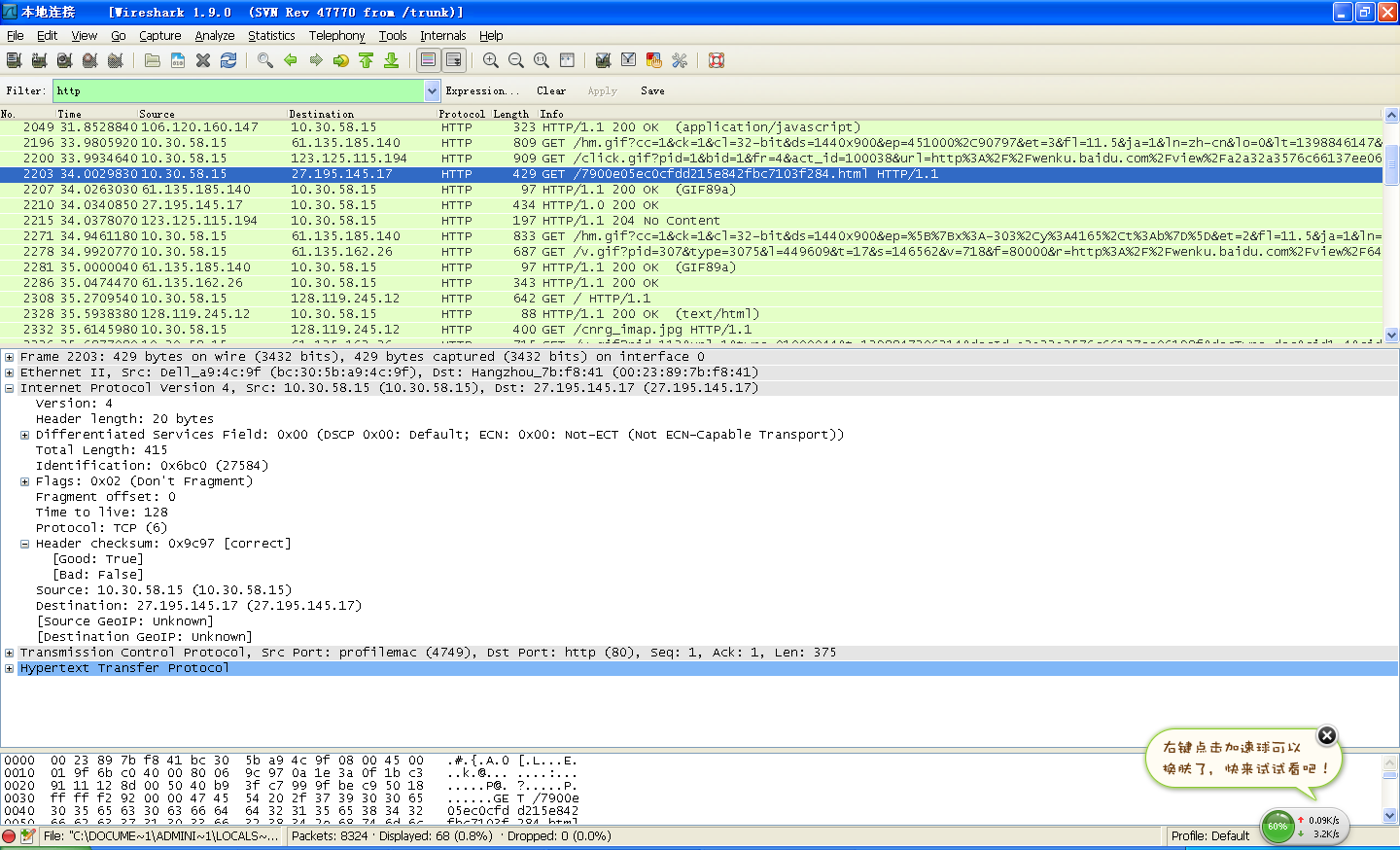


图 HTTP GET请求信息

（1）在协议框中，选择“GET/HTTP/1.1” 所在的分组会看到这个基本请求行后跟随着一系列额外的请求首部。在首部后的“\r\n”表示一个回车和换行，以此将该首部与下一个首部隔开。

“Host”首部在HTTP1.1版本中是必须的，它描述了URL中机器的域名，本例中是[http://gaia.cs.umass.edu](http://gaia.cs.umass.edu/)。这就允许了一个Web服务器在同一时间支持许多不同的域名。有了这个数不，Web服务器就可以区别客户试图连接哪一个Web服务器，并对每个客户响应不同的内容，这就是HTTP1.0到1.1版本的主要变化。

User-Agent首部描述了提出请求的Web浏览器及客户机器。

接下来是一系列的Accpet首部，包括Accept（接受）、Accept-Language（接受语言）、Accept-Encoding（接受编码）、Accept-Charset（接受字符集）。它们告诉Web服务器客户Web浏览器准备处理的数据类型。Web服务器可以将数据转变为不同的语言和格式。这些首部表明了客户的能力和偏好。

Keep-Alive及Connection首部描述了有关TCP连接的信息，通过此连接发送HTTP请求和响应。它表明在发送请求之后连接是否保持活动状态及保持多久。大多数HTTP1.1连接是持久的（persistent）,意思是在每次请求后不关闭TCP连接，而是保持该连接以接受从同一台服务器发来的多个请求。

（2）我们已经察看了由Web浏览器发送的请求，现在我们来观察Web服务器的回答。响应首先发送“HTTP/1.1 200 ok”，指明它开始使用HTTP1.1版本来发送网页。同样，在响应分组中，它后面也跟随着一些首部。最后，被请求的实际数据被发送。

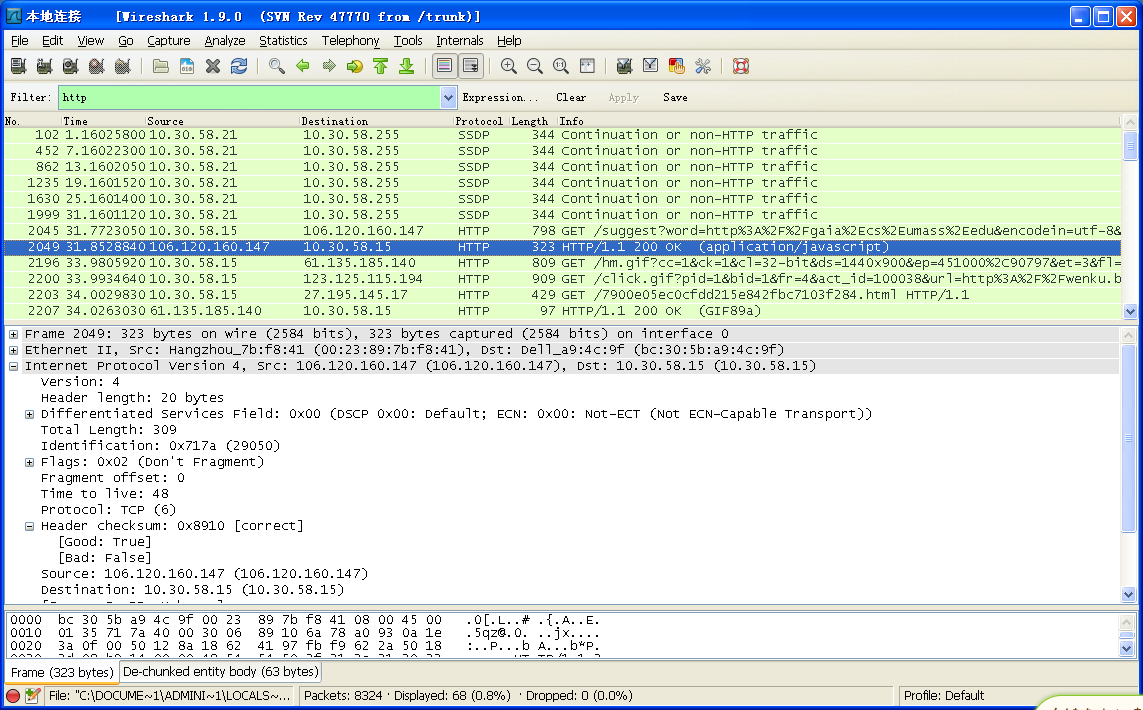


图 HTTP 响应

第一个Cache-control首部，用于描述是否将数据的副本存储或高速缓存起来，以便将来引用。一般个人的Web浏览器会高速缓存一些本机最近访问过的网页，随后对同一页面再次进行访问时，如果该网页仍存储于高速缓存中，则不再向服务器请求数据。类似地，在同一个网络中的计算机可以共享一些存在高速缓存中的页面，防止多个用户通过到其他网路的低速网路连接从网上获取相同的数据。这样的高速缓存被称为代理高速缓存（proxy cache）。在我们所俘获的分组中我们看到“Cache-control”首部值是“private”的。这表明服务器已经对这个用户产生了一个个性化的响应，而且可以被存储在本地的高速缓存中，但不是共享的高速缓存代理。

在HTTP请求中，Web服务器列出内容类型及可接受的内容编码。此例中Web服务器选择发送内容的类型是text/html且内容编码是gzip。这表明数据部分是压缩了的HTML。

服务器描述了一些关于自身的信息。此例中，Web服务器软件是Google自己的Web服务器软件。响应分组还用Content-Length首部描述了数据的长度。最后，服务器还在Date首部中列出了数据发送的日期和时间。

根据俘获窗口内容，回答1-6题。

（1）你的浏览器运行的是HTTP1.0，还是HTTP1.1？你所访问的服务器所运行的HTTP版本号是多少？

\_\_\_\_\_\_\_\_\_ HTTP1.1\_\_\_\_\_\_\_\_\_version1.1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

（2）你的浏览器向服务器指出它能接收何种语言版本的对象？

\_\_\_\_\_\_\_ Accept-Language:zh-cn\r\n\_\_\_

（3）你的计算机的IP地址是多少？

\_\_\_\_10.30.58.15\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

（4）从服务器向你的浏览器返回的状态代码是多少？

\_\_\_\_\_\_\_\_\_\_\_200ok\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

（5）你从服务器上所获取的HTML文件的最后修改时间是多少？\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: Wed, 30 Apr 2014 08:16:13 GMT\r\n\_\_\_\_

（6）返回到你的浏览器的内容一共多少字节？

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_52031\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3、HTTP条件GET/response交互

（1）启动浏览器，清空浏览器的缓存。(**在ie:工具----Internet选项-----删除----Internet临时文件 历史记录 删除缓存中的内容**)

（2）启动Wireshark分组俘获器，开始Wireshark分组俘获。

（3）在浏览器地址栏中如下网址：

http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html

你的浏览器中将显示一个具有五行的非常简单的HTML文件。

（4）在你的浏览器中重新输入相同的URL或单击浏览器中的“刷新”按钮。

（5）停止Wireshark分组俘获，在显示过滤筛选说明处输入“http”,分组列表子窗口中将只显示所俘获到的HTTP报文。

根据操作回答7-10题。

（7）分析你的浏览器向服务器发出的第一个HTTP GET请求的内容，在该请求报文中，是否有一行是：IF-MODIFIED-SINCE？

没有

（8）分析服务器响应报文的内容，服务器是否明确返回了文件的内容？如何获知？\_\_\_\_ Congratulations again! Now you've downloaded the file lab2-2.html.   
This file's last modification date will not change.

Thus if you download this multiple times on your browser, a complete copy   
will only be sent once by the server due to the inclusion of the IN-MODIFIED-SINCE  
field in your browser's HTTP GET request to the server.

（9）分析你的浏览器向服务器发出的第二个“HTTP GET”请求，在该请求报文中是否有一行是：IF-MODIFIED-SINCE？如果有，在该首部行后面跟着的信息是什么？ \_\_没有

（10）服务器对第二个HTTP GET请求的响应中的HTTP状态代码是多少？服务器是否明确返回了文件的内容？请解释

\_\_\_\_状态码和相应状态信息的值为304 NOT Modified,他表示缓存器可以使用该对象。 第二次没有返回文件的内容，因为他只是作为对该条件GET的响应，WEB服务器只发送一个响应报文，不包含请求的对象

4、获取长文件

（1）启动浏览器，将浏览器的缓存清空。

（2）启动Wireshark 分组俘获器，开始Wireshark分组俘获。

（3）在浏览器地址栏中输入如下网址:

<http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file3.html>

浏览器将显示一个相当大的美国权力法案

（4）停止Wireshark分组俘获，在显示过滤筛选说明处输入“http”,分组列表子窗口中将只显示所俘获到的HTTP报文。

根据操作回答11-14题。

（11）你的浏览器一共发出了多少个HTTP GET请求？\_\_\_\_\_\_\_\_1个

（12）**查看GET的响应消息，该单个HTTP响应消息，需要多少个TCP报文段**？\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_4个

（13）与这个HTTP GET请求相对应的响应报文的状态代码和状态短语是什么

\_HTTP/1.1 200OK\r\\_\_\_

5、嵌有对象的HTML文档

（1）启动浏览器，将浏览器的缓存清空。

（2）启动Wireshark分组俘获器。开始Wireshark分组俘获。

（3）在浏览器地址栏中输入如下网址：

<http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html>

浏览器将显示一个具有两个图片的短HTTP文件。

（4）停止Wireshark分组俘获，在显示过滤筛选说明处输入“http”,分组列表子窗口中将只显示所俘获到的HTTP报文。

根据操作回答15-16题。

（15）你的浏览器一共发出了多少个HTTP GET请求？这些请求被发送到的目的地的IP地址是多少？\_3个\_\_222.198.2.63\_

（16）浏览器在下载这两个图片时，是串行下载还是并行下载？请解释。

\_\_当前一个GET还没有得到回复就发送了下一个GET

6、HTTP认证

（1）启动浏览器，将浏览器的缓存清空。

（2）启动Wireshark分组俘获器。开始Wireshark分组俘获。

（3）在浏览器地址栏中输入如下网址：

<http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wireshark-file5.html>

浏览器将显示一个HTTP文件，输入所需要的用户名和密码(用户名：wireshark-students,密码:network)。

（4）停止Wireshark分组俘获，在显示过滤筛选说明处输入“http”,分组列表子窗口中将只显示所俘获到的HTTP报文。

根据操作回答17-18题。

（17）对于浏览器发出的最初的HTTP GET请求，服务器的响应是什么(状态代码和状态短语)?

２００ＯＫ

（18）当浏览器发出第二个HTTP GET请求时，在HTTP GET响应报文中包含了哪些新的字段？

Authorization :basic

# 利用Wireshark分析ICMP协议

1. 实验目的

分析ICMP协议。

**要求**

1、结果分析与保存的数据一致，否则没有实验成绩

2、数据保存名称：

Icmp数据：w0901-icmp.pcap(网络09班0号icmp协议)

实验结果分析报告名称：利用Wireshark分析ICMP协议\_w0901.doc

1. 实验环境

与因特网连接的计算机，操作系统为Windows，安装有Wireshark、IE等软件。

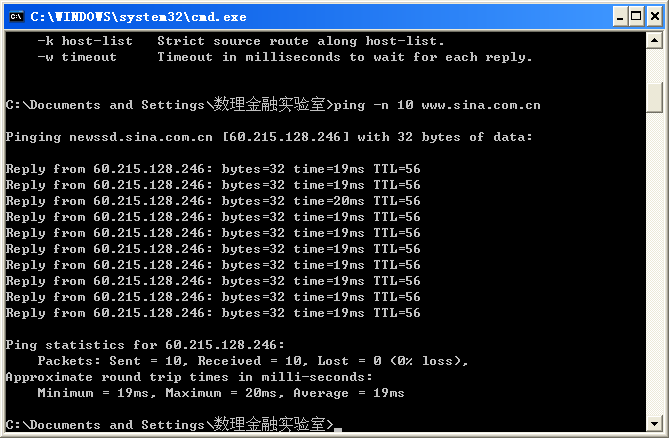
1. 实验内容
2. ICMP协议分析

步骤1：在 PC1 运行 Wireshark，开始截获报文，为了只截获和实验内容有关的报文，将 Wireshark 的 Captrue Filter 设置为“No Broadcast and no Multicast”；

步骤2：在 PC1 以 www.sina.com.cn为目标主机，在命令行窗口执行 Ping 命令，要求ping通10次；

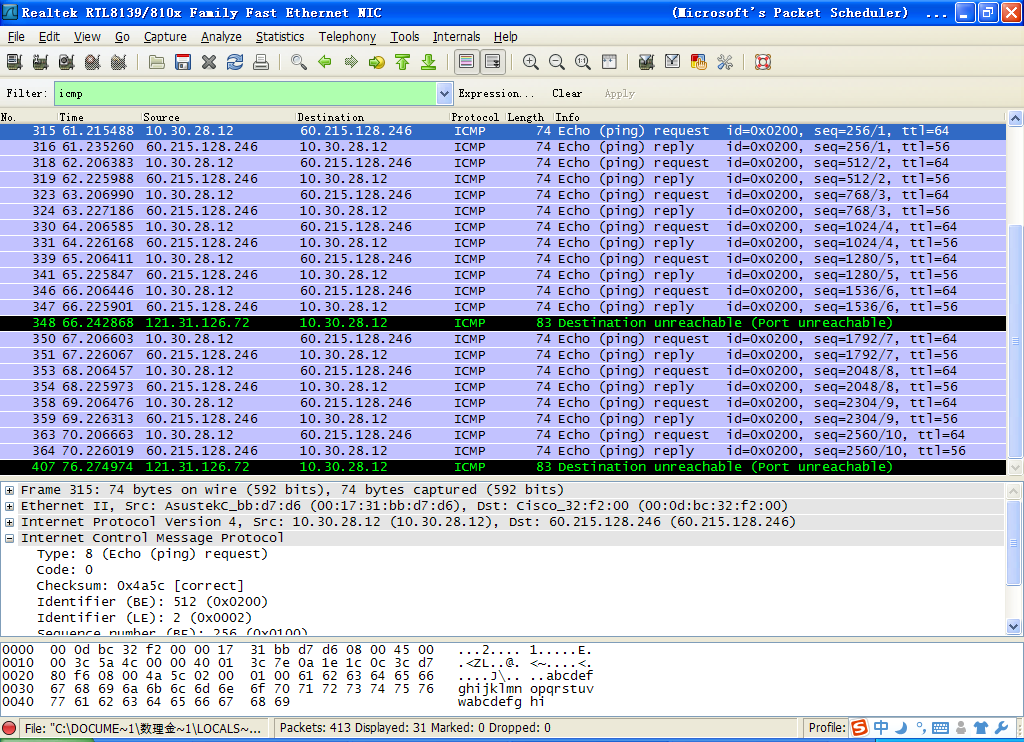
Ping命令为： ping – n 10 [www.sina.com.cn](http://www.sina.com.cn)

将命令行窗口进行截图：ping -n 10 www.sina.com.cn



步骤3：停止截获报文，分析截获的结果，回答下列问题：

1. 将抓包结果进行截图（要求只显示ping的数据包）：



1. 截获的ICMP 报文有几种类型？分别是：两种类型 0和8

3）分析截获的 ICMP 报文，查看表 5.1 中要求的字段值，填入表中。只需要填写6个报文信息。

表 5.1 ICMP报文分析

报文号

源IP

目的IP

报文格式

类型

代码

标识

序列号

315

10.30.28.12

60.215.128.246

8

0

0x5a4c

256/1

316

60.215.128.246

10.30.28.12

0

0

0x2437

256/1

318

10.30.28.12

60.215.128.246

8

0

0x5a4d

512/2

319

60.215.128.246

10.30.28.12

0

0

0x2438

512/2

323

10.30.28.12

60.215.128.246

8

0

0x5a4e

768/3

324

60.215.128.246

10.30.28.12

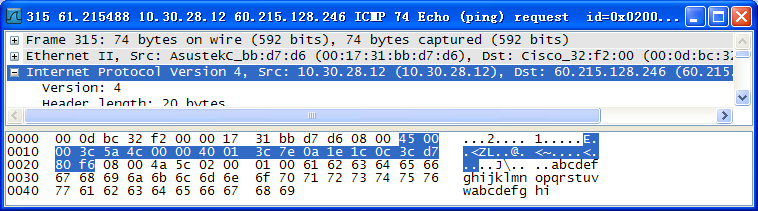
0

0

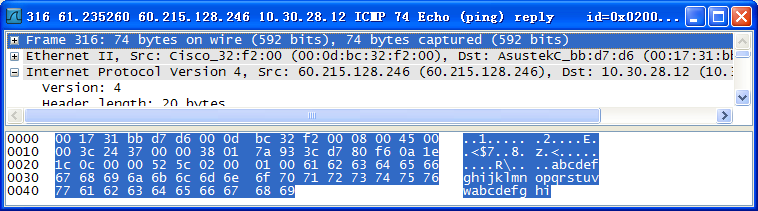
0x2439

768/3

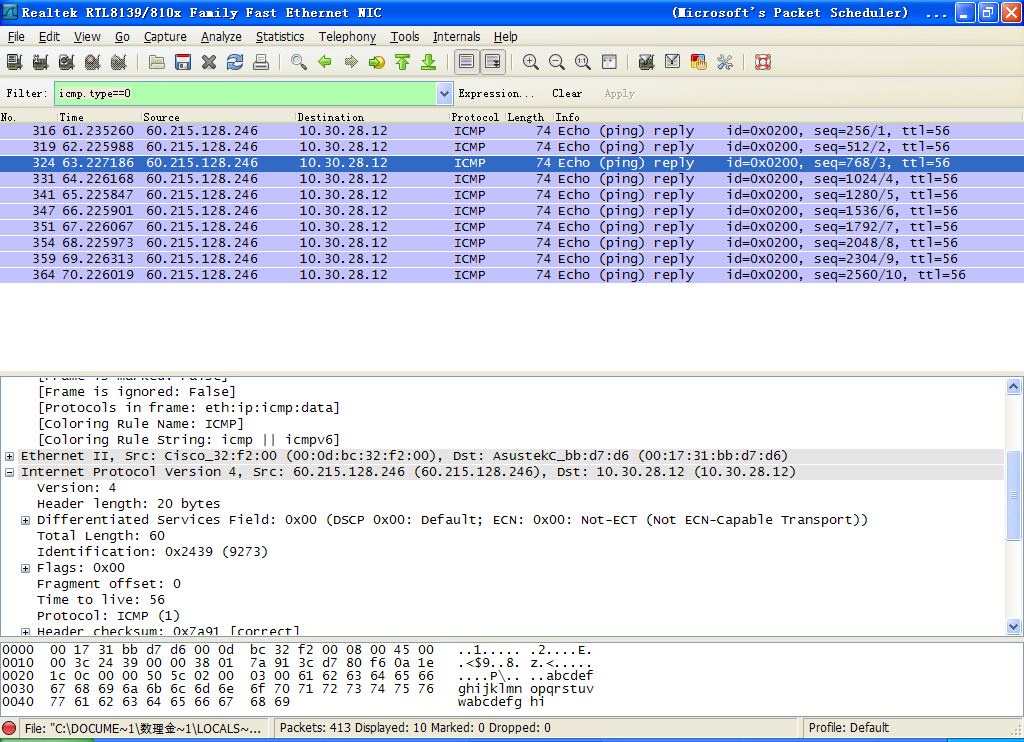
4）查看ping请求分组，ICMP的type是 8 和code是 0 并截图替换下图



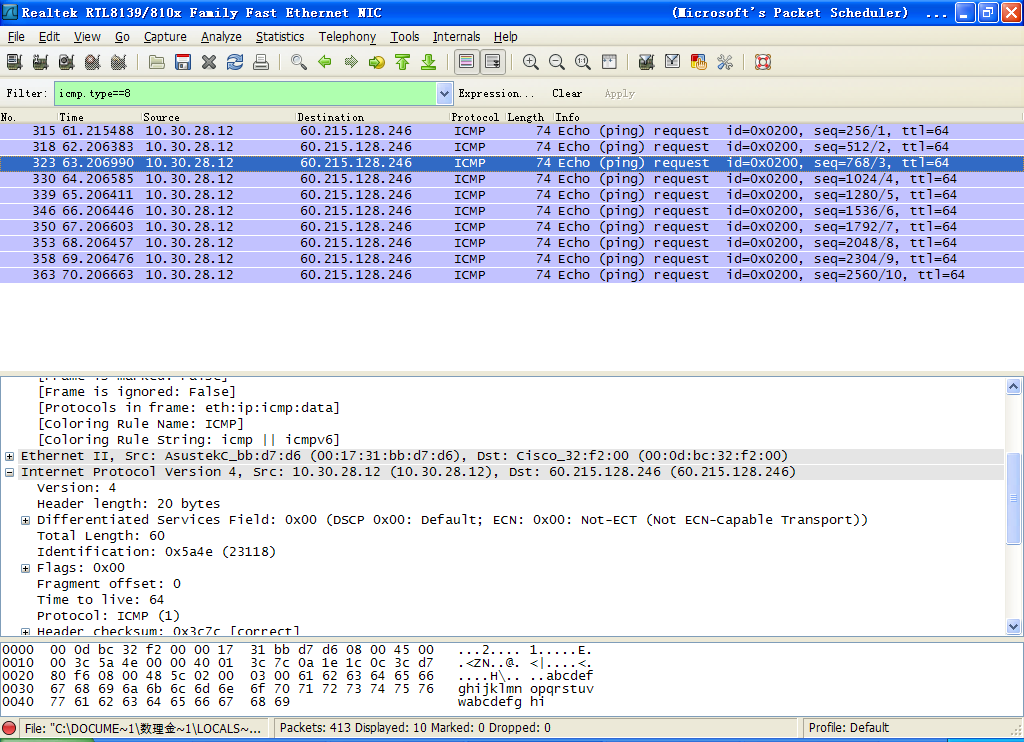
5）查看相应得ICMP响应信息，ICMP的type是 0 和code是 0 并截图替换下图



6）若要只显示ICMP的echo响应数据包，显示过滤器的规则为 icmp.type==0 并根据过滤规则进行抓包截图



7）若要只显示ICMP的echo请求数据包，显示过滤器的规则为 icmp.type==8 并根据过滤规则进行抓包截图



1. ICMP和Traceroute

在Wireshark 下，用Traceroute程序俘获ICMP分组。Traceroute能够映射出通往特定的因特网主机途径的所有中间主机。

源端发送一串ICMP分组到目的端。发送的第一个分组时，TTL=1；发送第二个分组时，TTL=2，依次类推。路由器把经过它的每一个分组TTL字段值减1。当一个分组到达了路由器时的TTL字段为1时，路由器会发送一个ICMP错误分组（ICMP error packet）给源端。

步骤4：在 PC1 上运行 Wireshark 开始截获报文；

步骤5：在PC1上执行Tracert命令，如：Tracert [www.sina.com.cn](http://www.sina.com.cn)；将命令窗口进行截图。

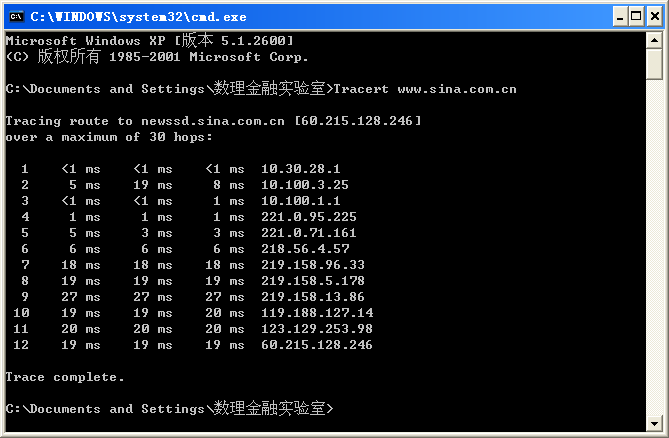


图4：命令提示窗口显示Traceroute程序结果（替换截图）

设置显示过滤器为icmp，图5显示的是一个路由器返回的ICMP超时报告分组（ICMP error packet）。注意到ICMP超时报告分组中包括的信息比Ping ICMP中超时报告分组包含的信息多。

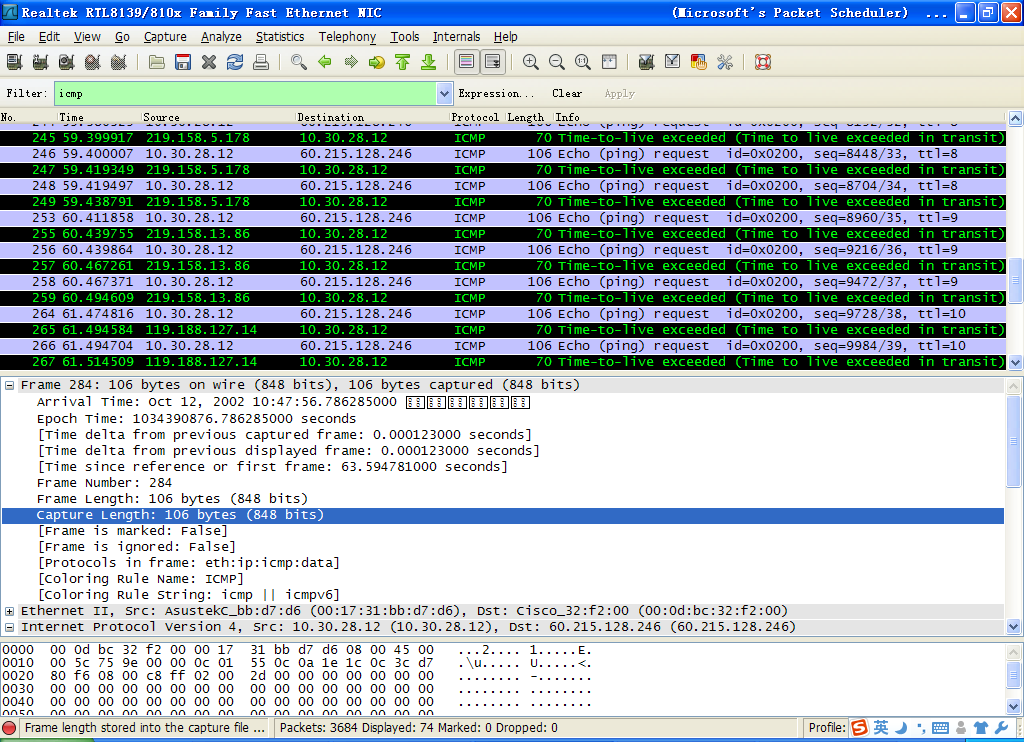


图5：一个扩展ICMP超时报告分组信息的Wireshark 窗口（替换截图）

步骤6：停止截获报文，分析截获的报文，回答下列问题：

1）截获了报文中哪几种 ICMP 报文？其类型码和代码各为多少？

ICMP报文类型

类型码（type）

代码（code）

reply

0

0

request

8

0

Time-to-live exceeded

11

0

2）在截获的报文中，超时报告报文的源地址分别是多少？

10.30.28.1 10.100.3.25 10.100.1.1 221.0.95.225 221.0.71.161 218.56.4.57 219.158.96.33 219.158.5.178 219.158.13.86 119.188.127.14 123.129.253.98

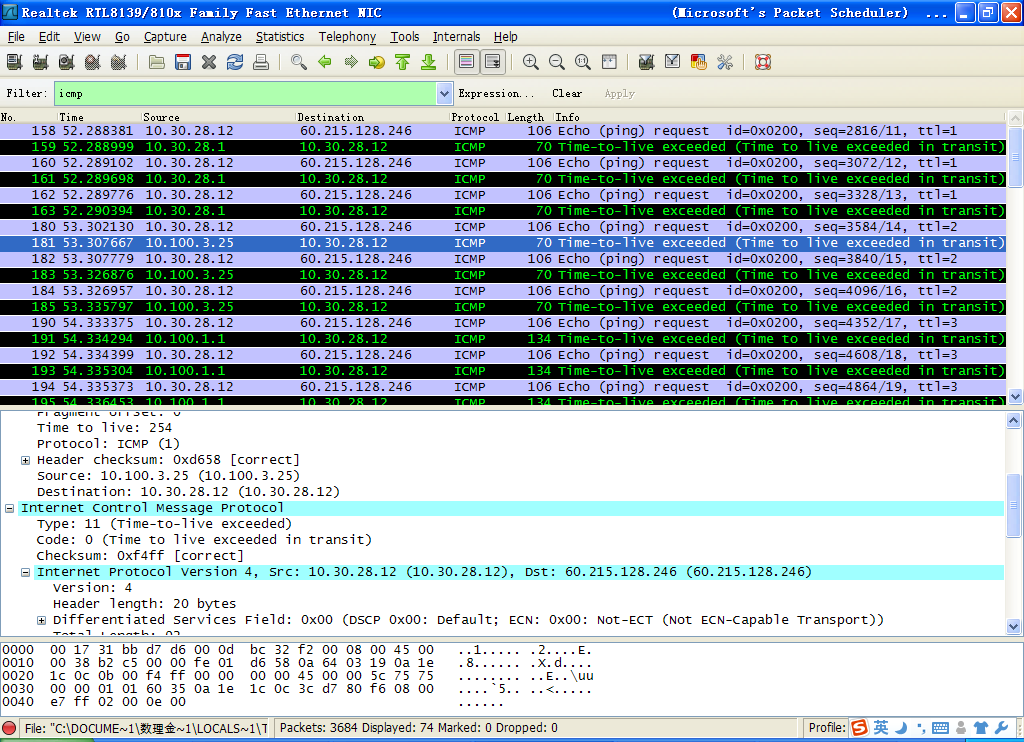
3)查看ICMP echo 分组 ，是否这个分组和前面使用 ping命令的ICMP echo 一样？对于TTL值有什么变化规律。

这个icmp发的echo是增长的而之前的是不变的

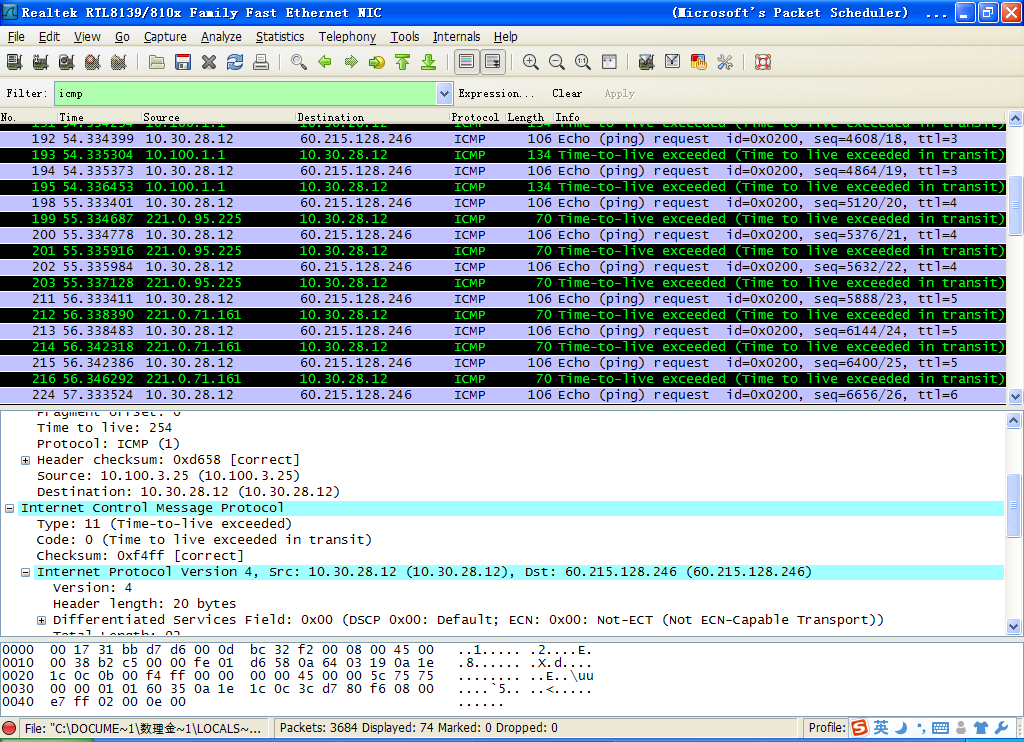
4)查看ICMP超时报告分组，它比ICMP echo 分组包括的信息多。对照ICMP协议，分析一下ICMP超时报告分组比ICMP echo 分组多包含的信息有哪些？

IP数据报的首部以及TCP或UDP协议的端口号

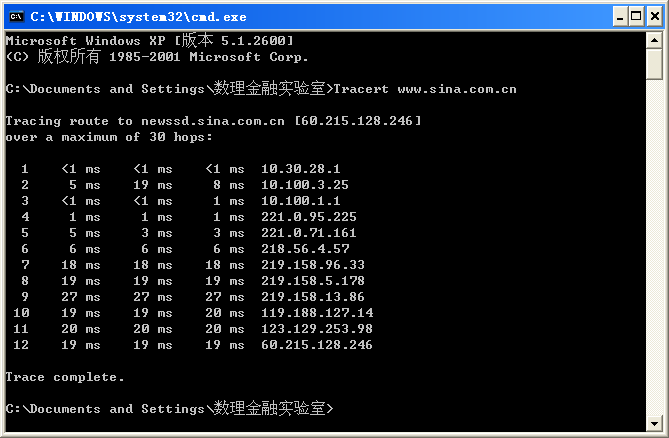
5）对于ICMP超时报告分组，找出与命令提示窗口截图中的第二跳路由器的接口IP地址为 10.100.3.25 ，在Wireshark抓包图中截图与第二跳路由器的接口IP地址对应的部分并截图。



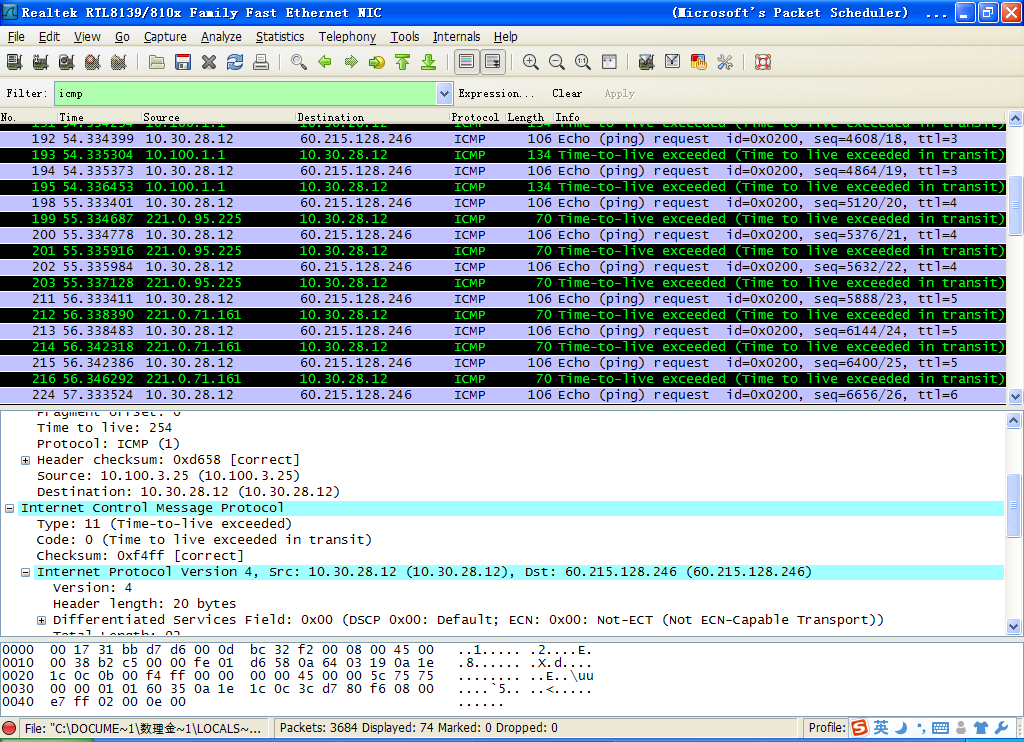
6）同理，找出第四跳路由器的接口IP地址为 221.0.95.225 ，在Wireshark抓包图中截图与第二跳路由器的接口IP地址对应的部分并截图。



举例如下，命令提示窗口截图：



其中，第二跳路由器的接口IP地址为 10.100.3.25 ，在Wireshark抓包图中截图与第二跳路由器的接口IP地址对应的部分并截图如下。



# 利用Wireshark分析tcp协议

1. 实验原理和要求

TCP协议工作原理参考TCP协议

Tcp显示过滤规则:

tcp.flags 显示包含TCP标志的封包。

tcp.flags.syn == 1 显示包含TCP SYN标志的封包。

tcp.flags.syn == 1 and tcp.flags.ack == 0 显示包含TCP SYN并且不包含ACK标志的封包。

tcp.flags.fin == 1 and tcp.flags.ack == 1 显示包含TCP FIN和ACK标志的封包。

tcp.window\_size == 0 && tcp.flags.reset != 1

**要求：**

1. 掌握TCP连接建立的三次握手过程
2. 理解TCP连接释放的四次握手过程
3. 实验环境

PC , WIRESHARK

1. 实验内容

1） 启动WireShark抓包

2） 访问学校主页服务器 ，通过Wireshark捕获通信内容

3） 分析TCP连接建立的三次握手和连接释放的四次握手过程

浏览网页，抓取三次握手的包，根据TCP包头格式将各字段取值填下来。

源IP： 10.30.28.13

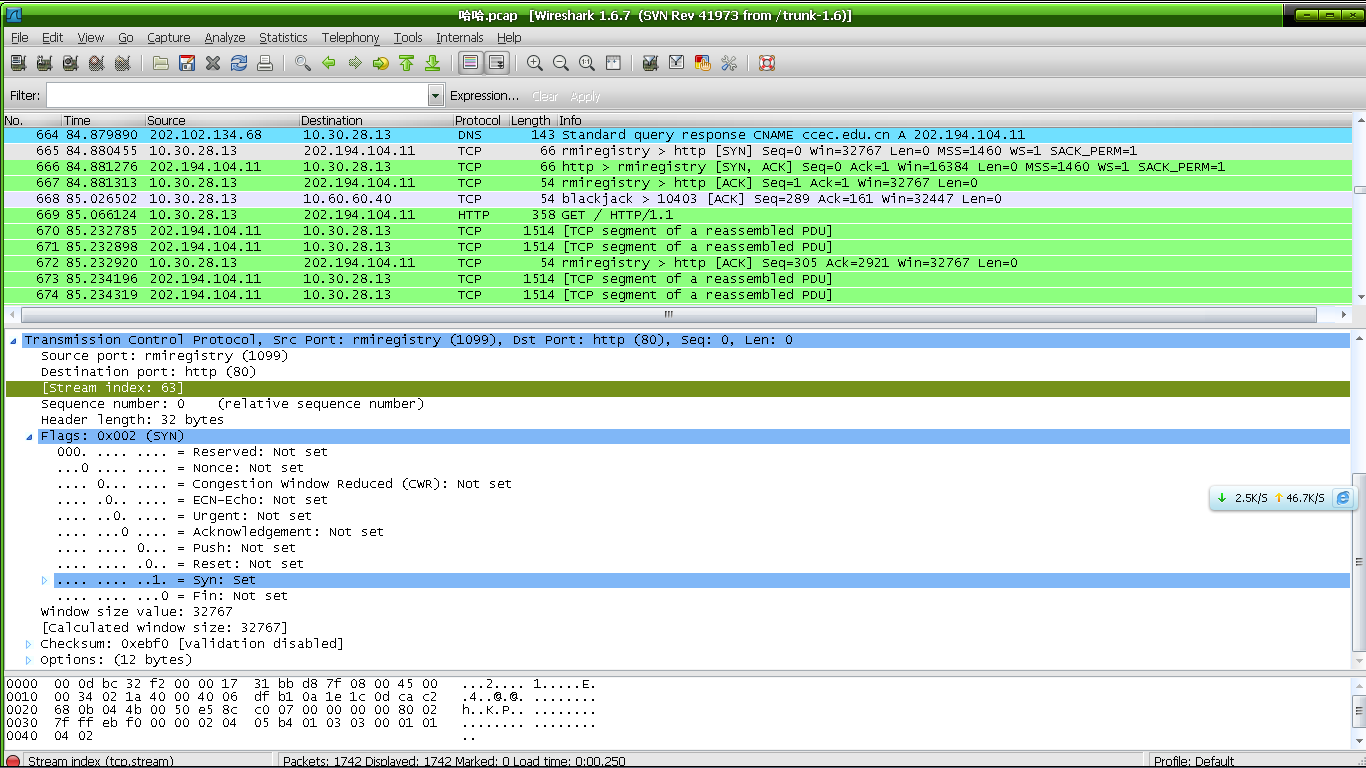
目的IP: 202.194.104.11

源端口： 1060

目的端口： 80

第一次握手：

找出第一次握手的数据包并截取对该数据包的展开图，根据截图填写横线内容。



序号： 0

确认号： 0

数据偏移： 32

URG: 0

ACK: 0

PSH: 0

RST: 0

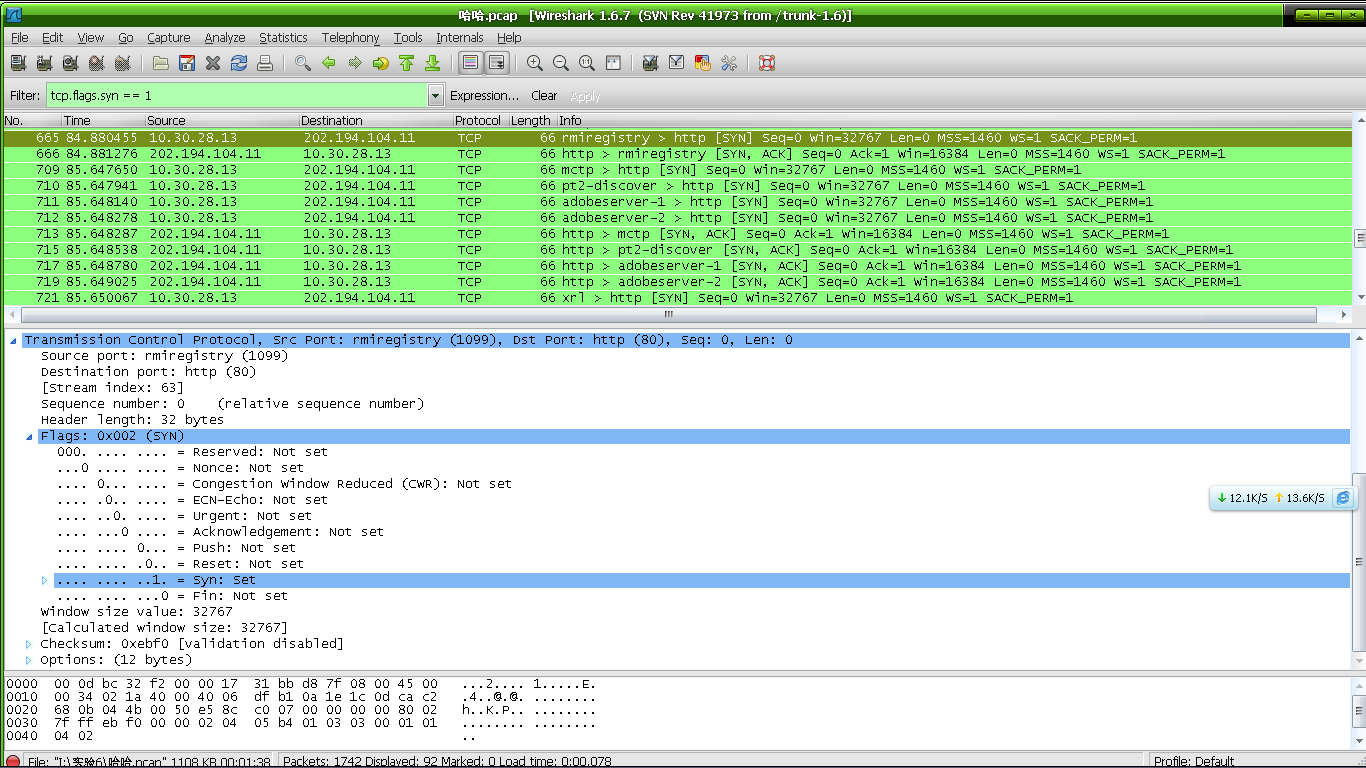
SYN: 1

FIN: 0

窗口： 32767

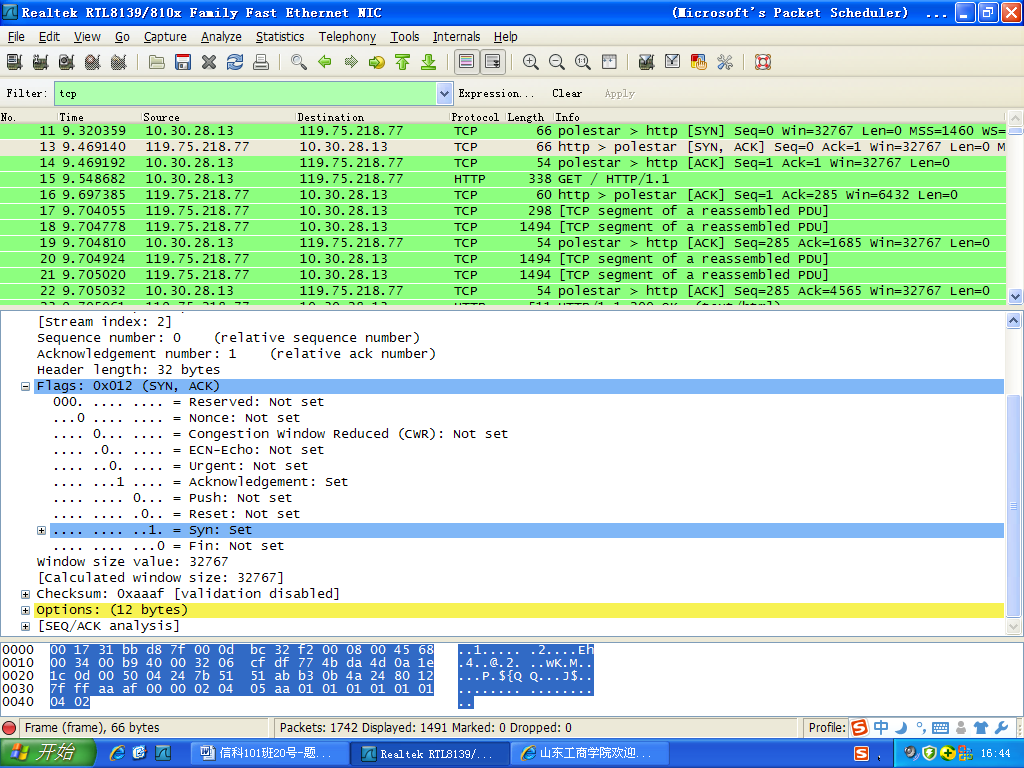
若只抓取第一次握手的数据包，则显示过滤器的规则为 tcp.flags.syn == 1

并截图替换下图：



**第二次握手：**

找出第二次握手的数据包并截取对该数据包的展开图，根据截图填写横线内容。

****

序号： 0

确认号： 1

数据偏移： 32

URG: 0

ACK: 1

PSH: 0

RST: 0

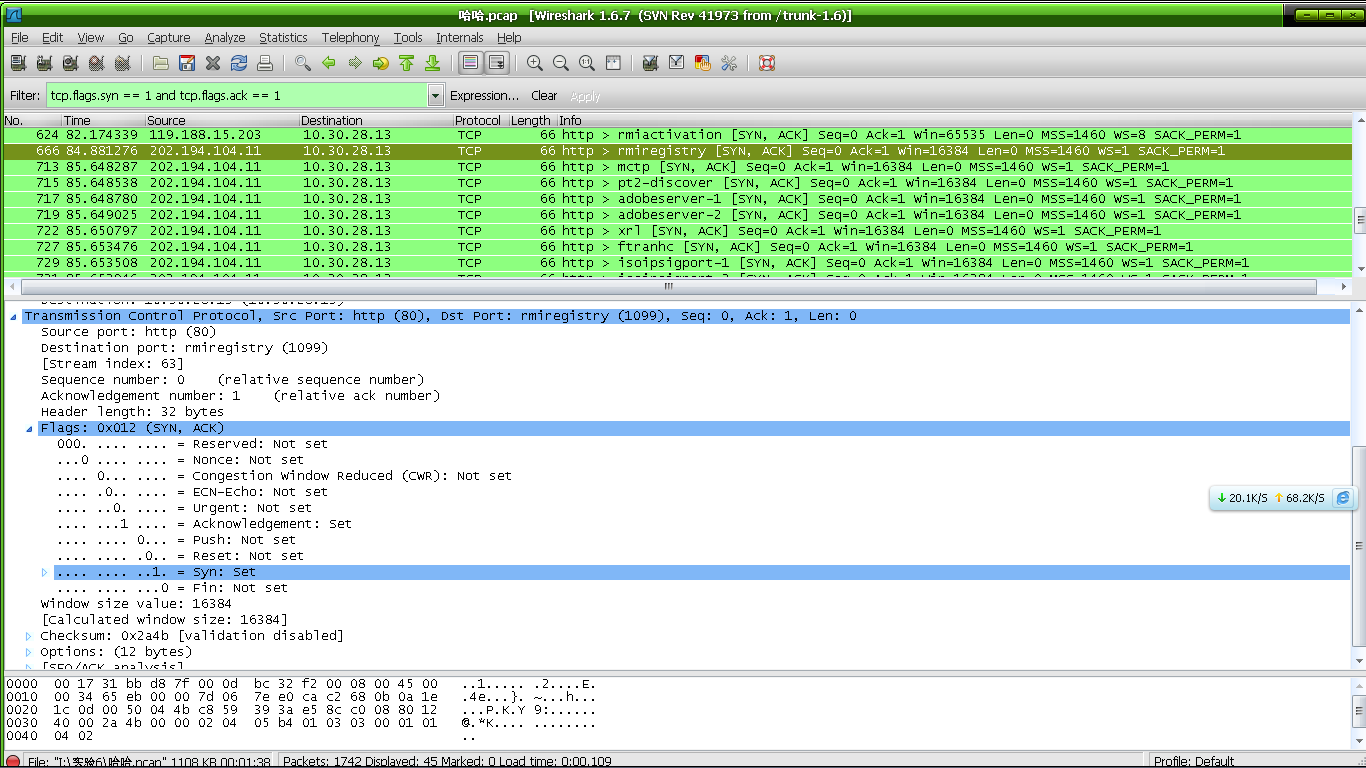
SYN: 1

FIN: 0

窗口： 32767

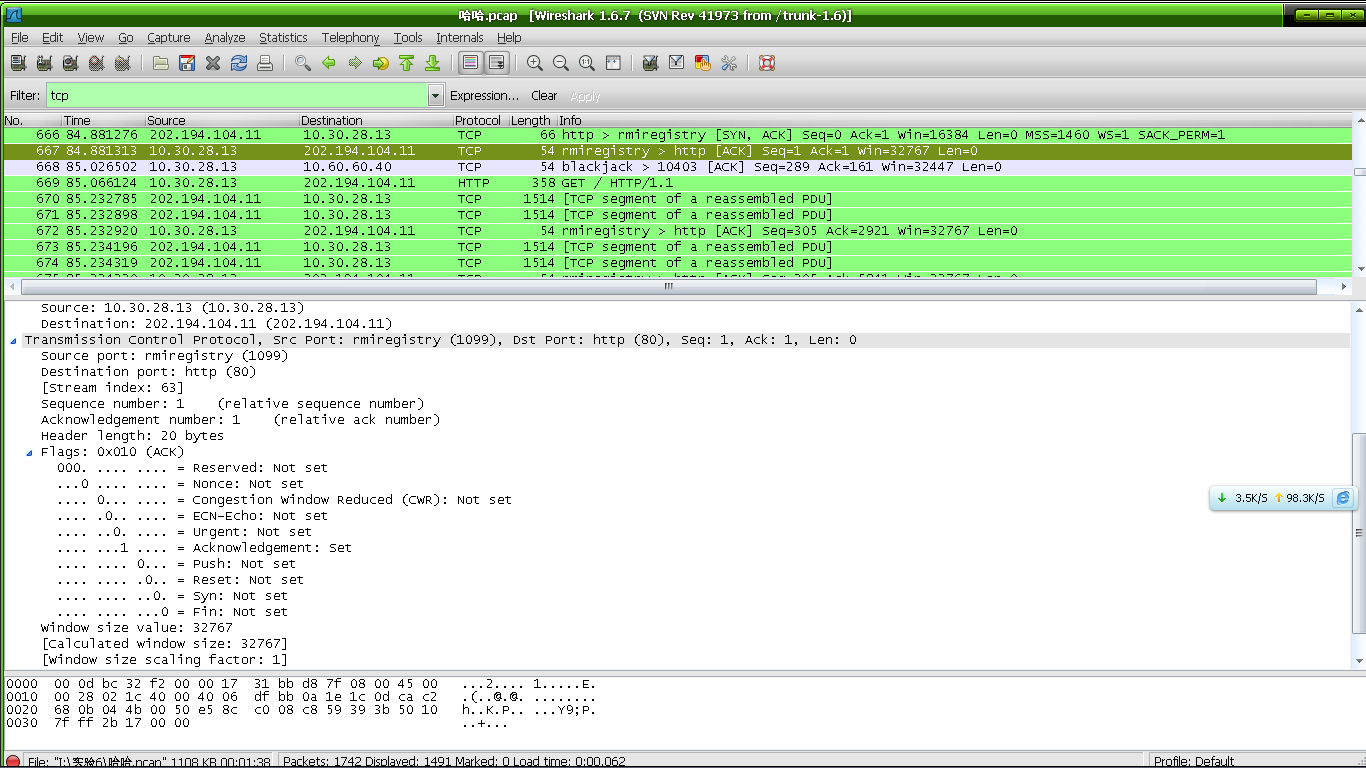
若只抓取第二次握手的数据包，则显示过滤器的规则为

tcp.flags.syn == 1 and tcp.flags.ack == 1并截图替换下图：（提示需要syn字段为1而ack字段为1）



**第三次握手：**

找出第三次握手的数据包并截取对该数据包的展开图，根据截图填写横线内容。

****

序号： 1

确认号： 1

数据偏移： 20

URG: 0

ACK: 1

PSH: 0

RST: 0

SYN: 0

FIN: 0

窗口： 32767

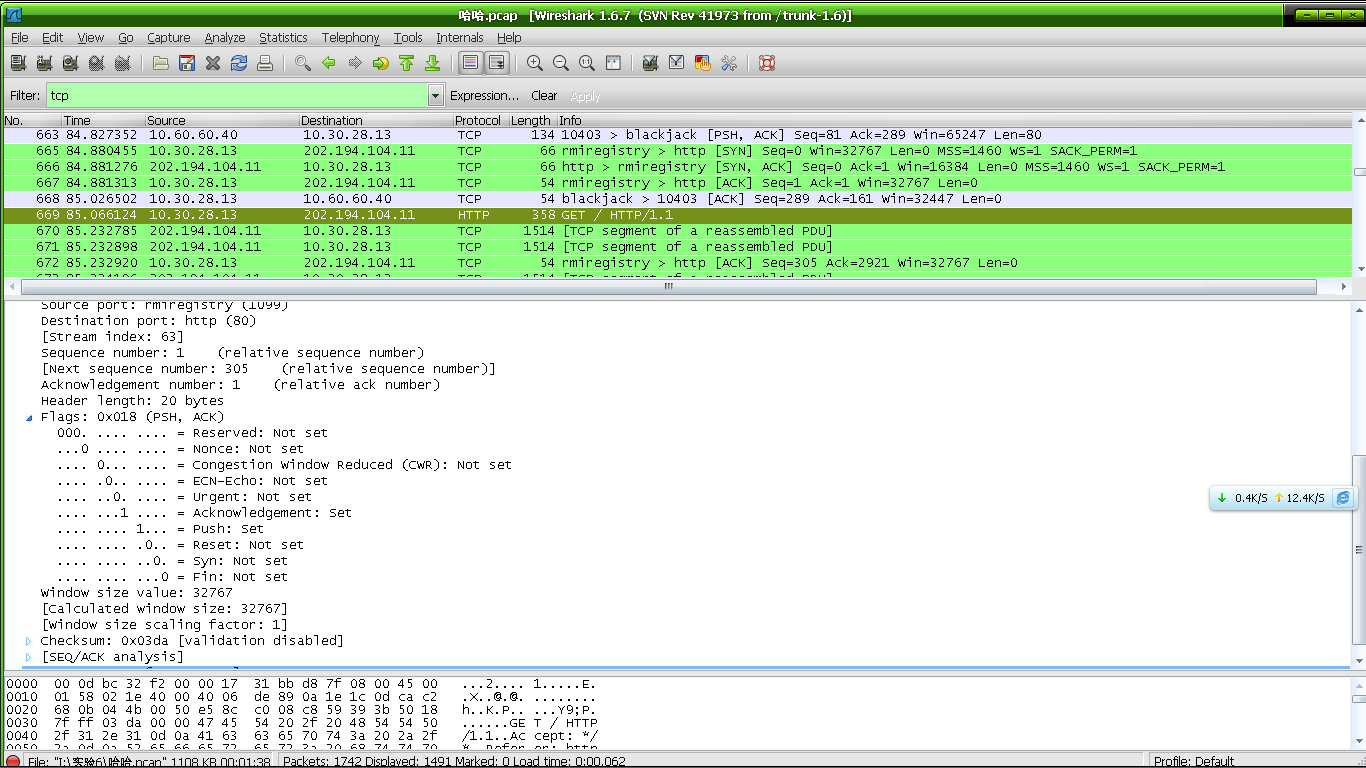
是否可以只抓取第三次握手的数据包，为什么?若不能，如何确定是第三次握手的数据包。

不可以只抓取第三次握手的数据包，因为只要是确认包， ACK都是1 ，所以要再根据序号判断

三次握手之后，客户端要请求服务器传送数据

**第一次请求数据：**

找出第一次请求的数据包并截取对该数据包的展开图，根据截图填写横线内容。

****

序号： 1

确认号： 1

数据偏移： 20

URG: 0

ACK: 1

PSH: 1

RST: 0

SYN: 0

FIN: 0

窗口： 32767

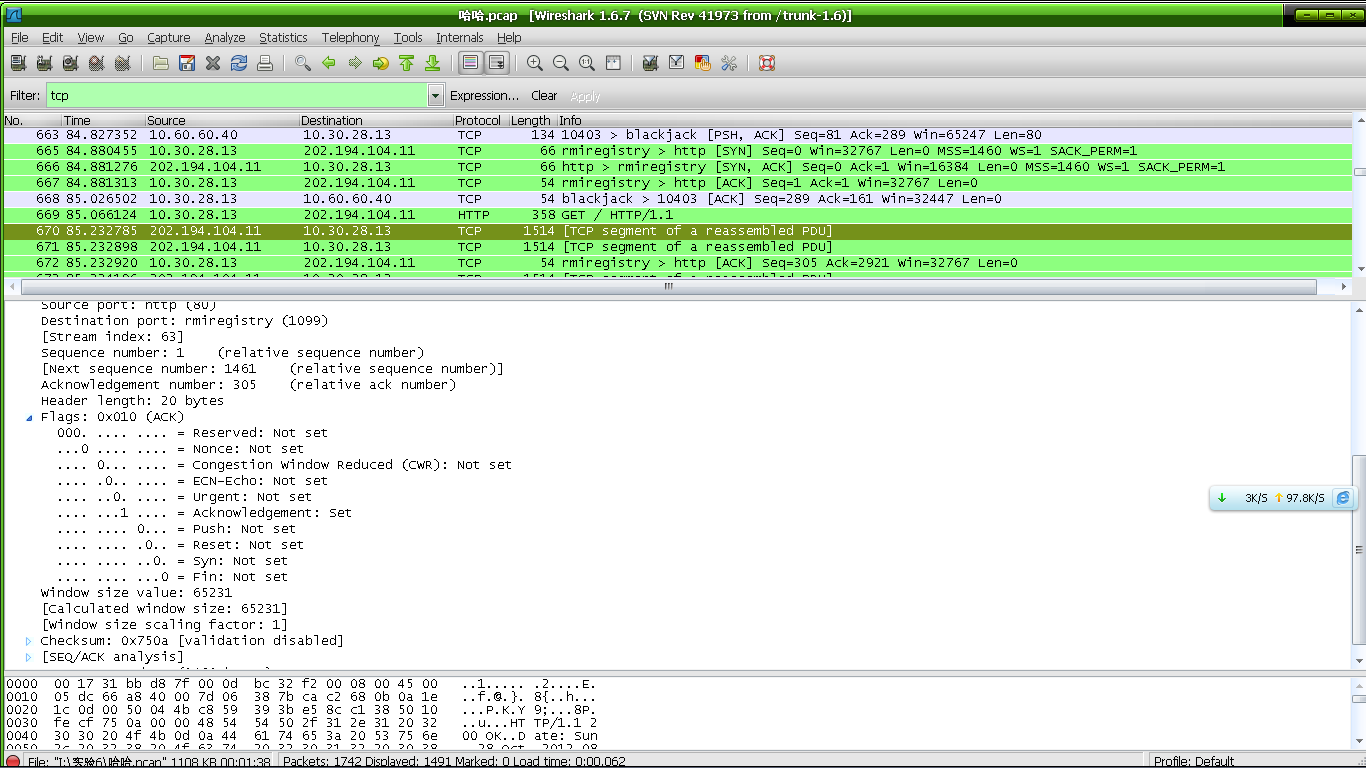
你是如何确定是第一次的请求的数据包： http协议 并且 push=1

若通过过滤规则如何找出第一次的请求数据包：tcp.flags.syn == 1and tcp.flags.push == 1

服务器要传送数据给客户端

**第一次回复数据：**

找出第一次回复的数据包并截取对该数据包的展开图，根据截图填写横线内容。

****

序号： 1

确认号： 305

数据偏移： 20

URG: 0

ACK: 1

PSH: 0

RST: 0

SYN: 0

FIN: 0

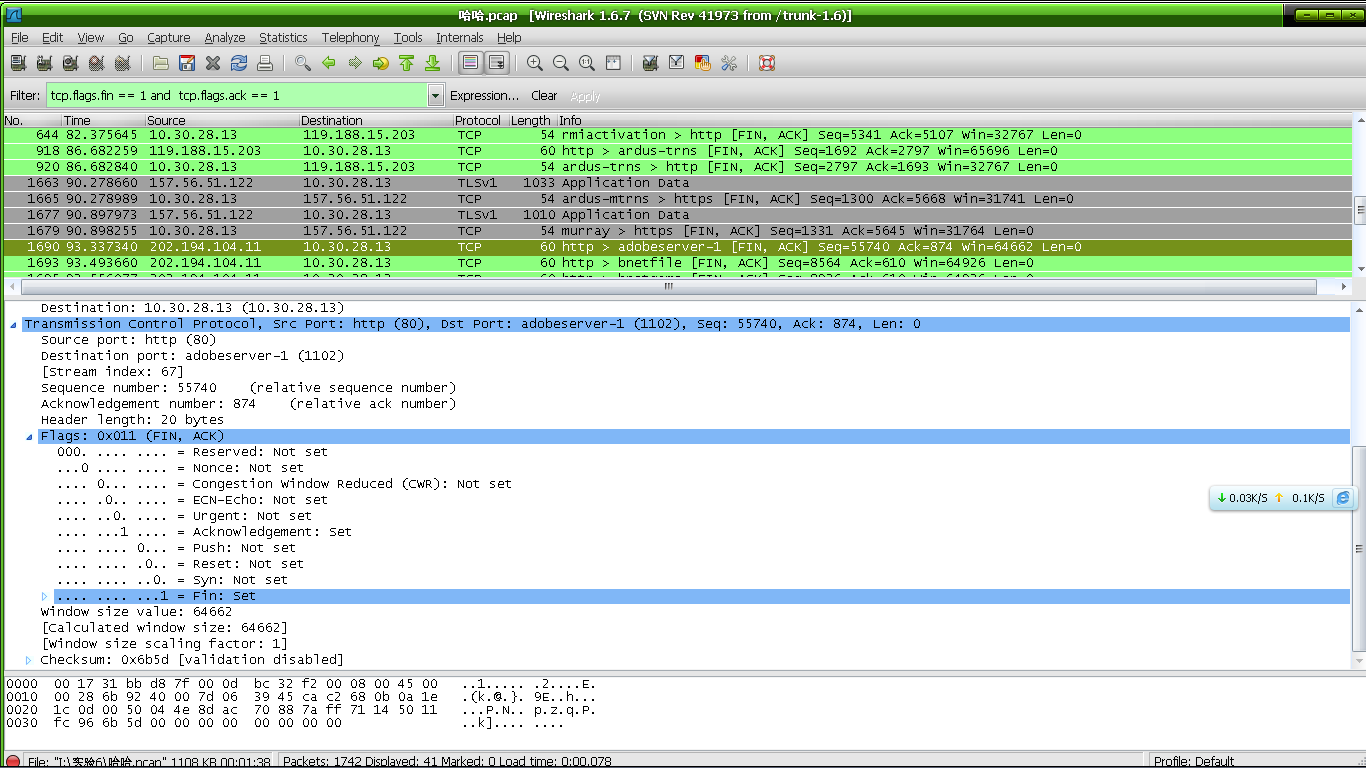
窗口： 65231

你是如何确定是第一次的回复的数据包：

ACK=1,PUSH=0且报文显示为[TCP segment of a reassembled PDU]

捕捉断开连接的数据包，显示过滤器的规则为 tcp.flags.fin == 1 and  tcp.flags.ack == 1

并截图替换下图：



根据截图内容填写：

第一次断开：

URG: 0

ACK: 1

PSH: 0

RST: 0

SYN: 0

FIN: 1

第二次断开：

URG: 0

ACK: 1

PSH: 0

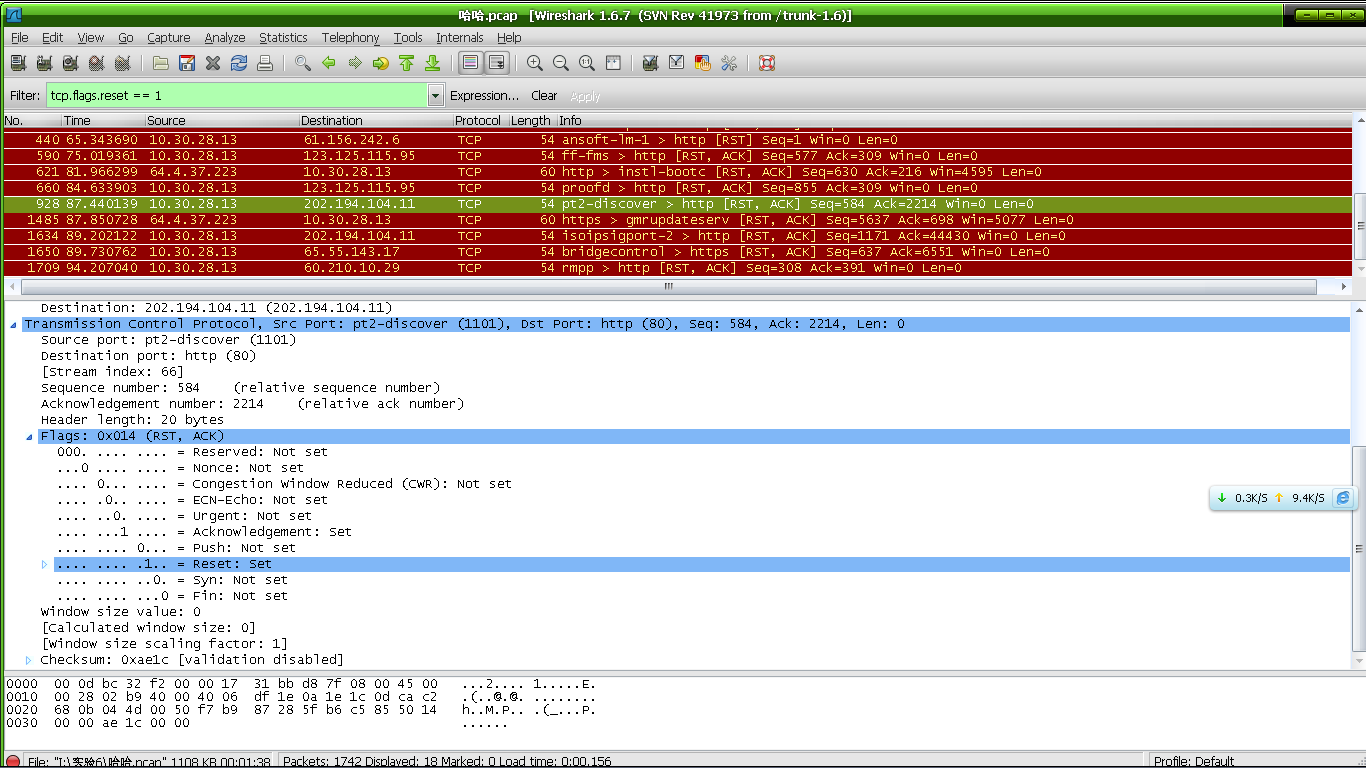
RST: 0

SYN: 0

FIN: 1

3、捕捉RST置位的包，显示过滤器的规则为 tcp.flags.reset == 1

截图（RST置位）并填写横线

****

序号： 584

确认号： 2214

数据偏移： 20

URG: 0

ACK: 1

PSH: 0

RST: 1

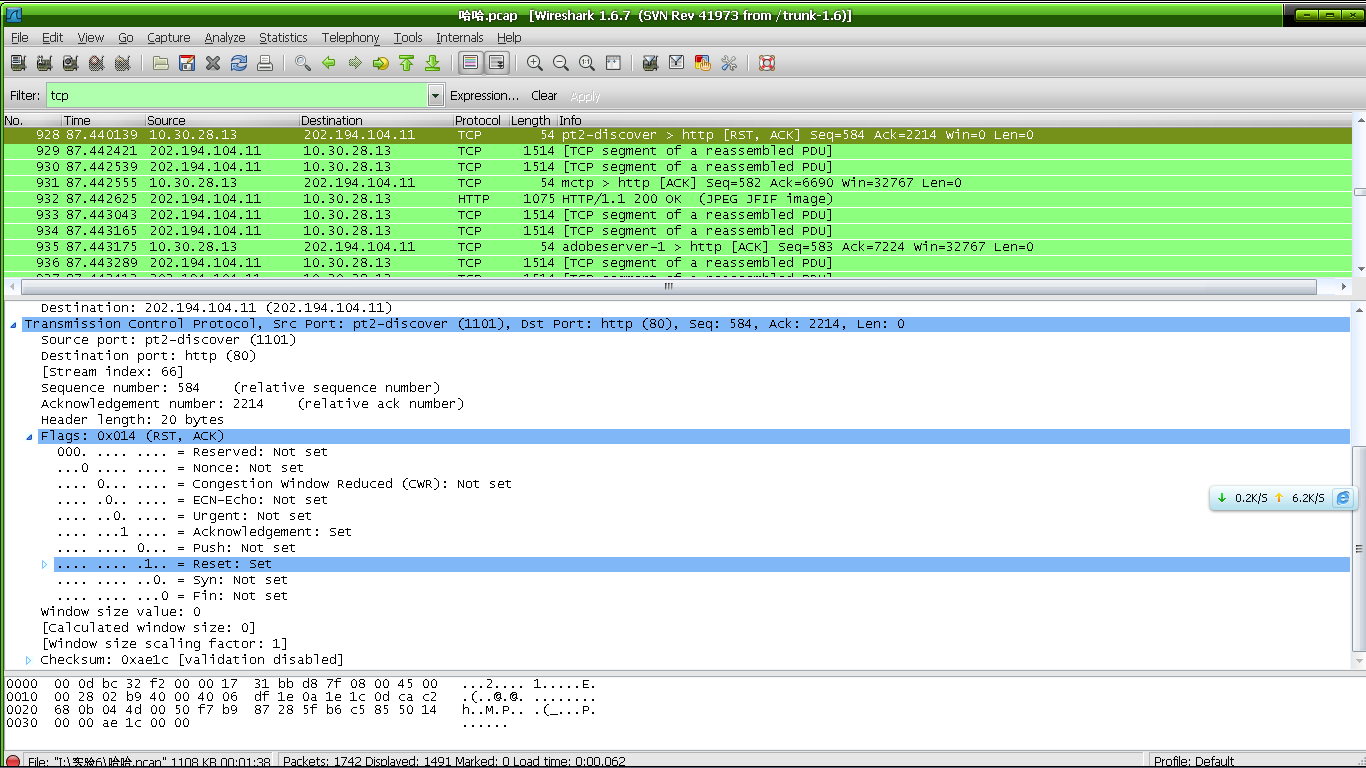
SYN: 0

FIN: 0

窗口： 0

传输层TCP数据报结构详解

选取一条TCP数据包并截图，



填写各项的信息及作用

TCP数据报中依次包括以下信息：

1、Source Port： 1101 ，表示 目标通过1101端口传送数据 。该部分占 16个BIT。

2、Destination Port： 80 ，表示 本机通过80接口接收数据 。该部分占 16 个BIT。

3、Initial Sequence Number：584，表示 每一个字节的编号 ，即SEQ值。该部分占32个BIT，值从1到2的32次方减1。   
4、Next Expected SEQ Numbe：2214，表示 下一个要接到数据的编号，即对方返回的ACK值。该部分占 32 个BIT，值从1到2的32次方减1。

5、Data Offset： 4 Bytes，表示 报头大小 。该部分占 32 个BIT。   
6、Reserved Bites：保留位，此处不用。该部分占 6 个BIT。   
7、Flags： 0x0010 。该值用两个十六进制数来表示。该部分长度为 3 个BIT，6个标志位的含义分别是：   
URG: 0 ，为1表示 紧急指针有效 。   
ACK: 1 ，为1表示 确认报文 。  
PSH: 0 ，为1表示 发送报文 。   
RST: 0 。为1表示 TCP出现严重错误必须释放

SYN: 0 。为1表示 一个连接请求 。   
FIN: 0 。为1表示 随访发送数据已完毕 。

8、Window： 0 ，表示 接收方目前允许对方发送的数据量 。该部分占 16 个BIT。

9、CheckSum： 0xae1c，表示 校验和 。该部分占 16 个BIT，用十六进制表示。   
10、Urgent Pointer： 00 ，表示 紧急数据的字节数 。该部分占 2 个BIT。   
11、Maximum Segment Size： 1514 ，表示 数据字段的最大长度 。

**【思考问题】**

1.试用具体例子说明为什么传输连接建立时要使用三次握手。如不这样做可能会出现什么情况。

**答：我们知道，3次握手完成两个重要的功能，既要双方做好发送数据的准备工作（双方都知道彼此已准备好），也要允许双方就初始序列号进行协商，这个序列号在握手过程中被发送和确认。**

**现在把三次握手改成仅需要两次握手，死锁是可能发生的。作为例子，考虑计算机A和B之间的通信，假定B给A发送一个连接请求分组，A收到了这个分组，并发送了确认应答分组。按照两次握手的协定，A认为连接已经成功地建立了，可以开始发送数据分组。可是，B在A的应答分组在传输中被丢失的情况下，将不知道A是否已准备好，不知道A建议什么样的序列号，B甚至怀疑A是否收到自己的连接请求分组。在这种情况下，B认为连接还未建立成功，将忽略A发来的任何数据分组，只等待连接确认应答分组。而A在发出的分组超时后，重复发送同样的分组。这样就形成了死锁。**

2.使用TCP对实时话音数据的传输有什么问题？使用UDP在传送数据文件时会有什么问题？

**答：1.如果语音数据不是实时播放（边接受边播放）就可以使用TCP，因为TCP传输可靠。接收端用TCP讲话音数据接受完毕后，可以在以后的任何时间进行播放。但假定是实时传输，则必须使用UDP。**

**3.UDP不保证可靠交付，但UCP比TCP的开销要小很多。因此只要应用程序接受这样的服务质量就可以使用UDP。**

4.TCP在进行流量控制时是以分组的丢失作为产生拥塞的标志。有没有不是因拥塞而引起的分组丢失的情况？如有，请举出三种情况。

**答：当IP数据报在传输过程中需要分片，但其中的一个数据报未能及时到达终点，而终点组装IP数据报已超时，因而只能丢失该数据报；IP数据报已经到达终点，但终点的缓存没有足够的空间存放此数据报；数据报在转发过程中经过一个局域网的网桥，但网桥在转发该数据报的帧没有足够的差错空间而只好丢弃。**

# 使用wireshark分析arp协议

1. 实验目的：

1．学习ARP协议的工作原理以及ARP分组格式；

2．学习使用WireShark对ARP协议进行分析。

1. 实验环境

PC,wireshark

1. 实验内容

启动WireShark

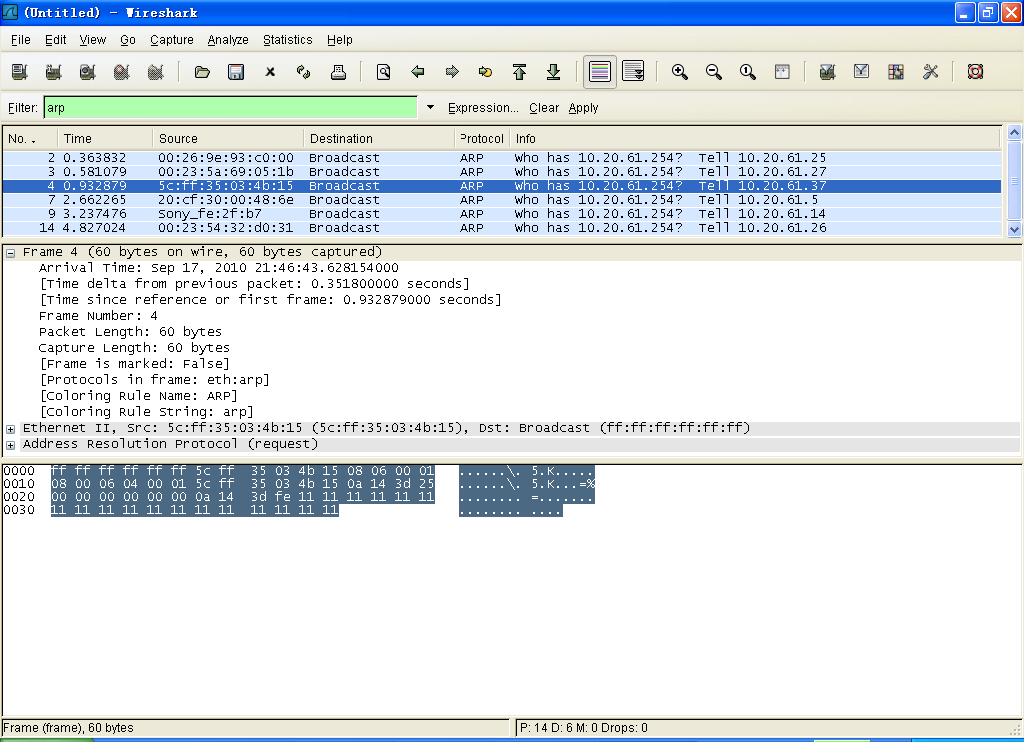
（2）捕获数据

（3） 停止抓包并分析ARP请求报文

将Filter过滤条件设为arp，回车或者点击“Apply”按钮，

（4）ARP请求报文分析

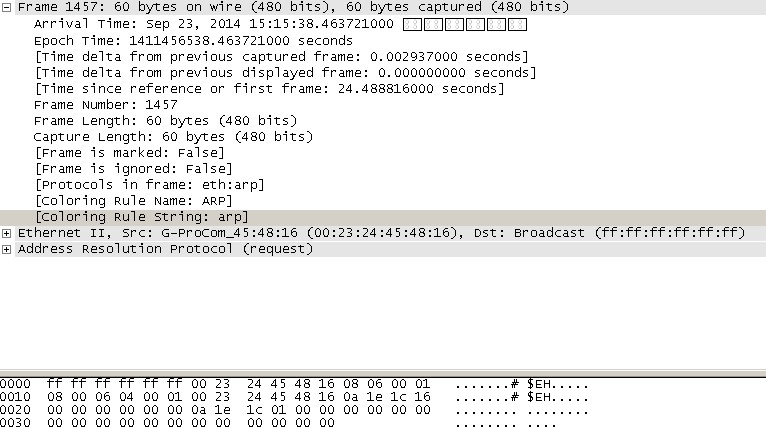
1）粘贴你捕获的ARP请求报文



2）分析你捕获的ARP请求报文

第一行 帧基本信息分析

（粘贴你的Frame信息）



Frame Number（ 帧的编号）：¬¬¬¬\_\_\_\_\_\_1457\_\_\_（捕获时的编号）

Frame Length(帧的大小)：\_\_\_\_60\_\_\_\_字节。（以太网的帧最小64个字节，而这里只有６０个字节，应该是没有把四个字节的CRC计算在里面，加上它就刚好。）

Arrival Time(帧被捕获的日期和时间): \_\_sep 23,\_2014 15 :15 :38 .463721000\_\_\_\_\_\_

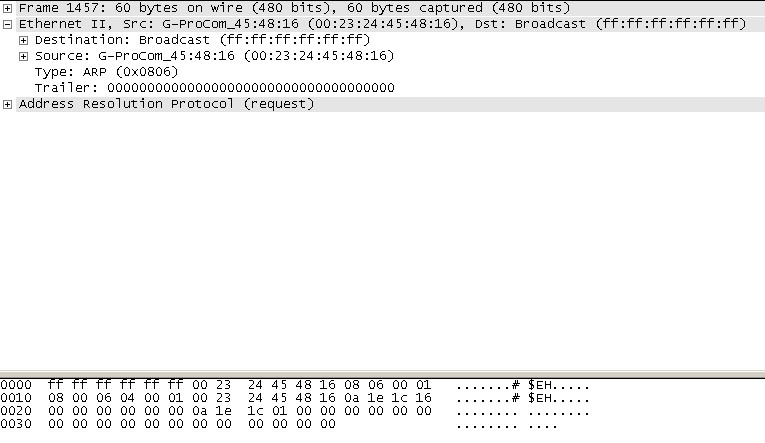
Time delta from previous captured frame(帧距离前一个帧的捕获时间差):\_\_\_\_0.002937000 seconds\_\_\_\_

Time since refernce or first frame(帧距离第一个帧的捕获时间差)：\_\_\_24.488816000 seconds\_\_\_\_\_\_\_\_\_\_\_\_

Protocols in frame(帧装载的协议)：\_\_eth:arp\_\_\_\_\_\_\_\_\_\_

第二行 数据链路层：

（粘贴你的数据链路层信息）



Destination（目的地址）：\_Broadcast(ff:ff:ff:ff:ff:ff)\_\_\_\_\_\_\_\_\_（这是个MAC地址，这个MAC地址是一个广播地址，就是局域网中的所有计算机都会接收这个数据帧）

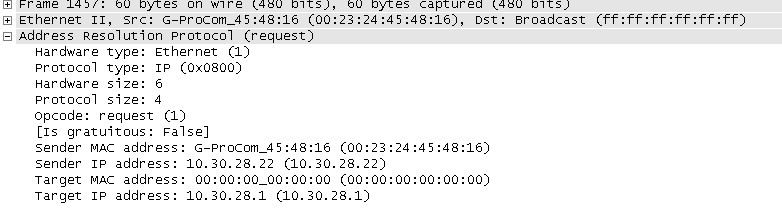
Source（源地址）：G-ProCom\_45:48:16(00:23:24:45:48:16)

帧中封装的协议类型：ARP(0x0806)（这个是ARP协议的类型编号。）

Trailer：是协议中填充的数据，为了保证帧最少有64字节。

第三层 ARP协议：

（粘贴你的ARP请求报文）



在上图中，我们可以看到如下信息：

　　　Ｈardware type（硬件类型）：\_\_\_Ethernet(1)\_\_\_\_\_\_\_\_\_

　　　Ｐrotocol type（协议类型）： IP(0x0800)\_\_\_\_\_\_\_\_\_\_\_\_

　　　Ｈardware size(硬件信息在帧中占的字节数)：\_\_6\_\_\_\_\_\_\_\_\_\_\_\_\_

　　　Ｐrotocol size(协议信息在帧中占的字节数)：\_\_\_\_\_4\_\_\_\_\_\_\_\_\_

　　　操作码（opcode）：requset(0X0001)

　　　发送方的ＭＡＣ地址（Sender MAC address）：G-ProCom\_45:48:16(00:23:24:45:48:16)\_\_\_\_

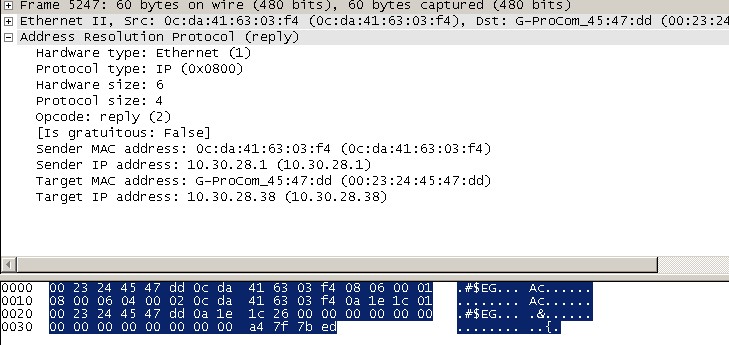
　　　发送方的IP地址（Sender IP address）：\_\_10.30.28.22(10.30.28.22)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

　　　目标的ＭＡＣ地址（Target MAC address:）：\_\_\_\_\_\_\_00:00:00\_00:00:00(00:00:00:00:00:00)\_\_\_\_\_\_\_\_\_\_\_\_

　　　目标的IP地址（Target IP address:）：\_\_\_\_10.30.28.1(10.30.28.1)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

（3）分析ARP应答报文

（粘贴你的ARP应答报文）



应答报文中的

　 操作码（opcode）：reply(0X0002)

　　　发送方的ＭＡＣ地址（Sender MAC address）： \_0c:da:41:63:03:f4(0c:da:41:63:03:f4)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

　　　发送方的IP地址（Sender IP address）：\_10.30.28.1(10.30.28.1)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

　　　目标的ＭＡＣ地址（Target MAC address:）：\_\_\_ G-ProCom\_45:47:dd(00:23:24:45:47:dd)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

　　　目标的IP地址（Target IP address:）：\_\_\_\_\_\_\_10.30.28.38(10.30.28.38)\_\_\_\_\_\_\_\_\_\_\_\_\_

练习1：对于上述2、3中的arp请求报文和应答报文，将ARP请求报文和ARP应答报文中的字段信息填入表3-1。

表3-1 RPP请求报文和ARP应答报文的字段信息

字段项

ARP请求数据报文

ARP应答数据报文

链路层Destination项

Broadcast(ff:ff:ff:ff:ff:ff)

G-ProCom\_45:47:dd(00:23:24:45:47:dd)

链路层Source项

G-ProCom\_45:48:16(00:23:24:45:48:16)

0c:da:41:63:03:f4(0c:da:41:63:03:f4)

网络层Sender MAC Address

G-ProCom\_45:48:16(00:23:24:45:48:16)

0c:da:41:63:03:f4(0c:da:41:63:03:f4)

网络层Sender IP Address

10.30.28.22(10.30.28.22)

10.30.28.1(10.30.28.1)\_

网络层Target MAC Address

00:00:00\_00:00:00(00:00:00:00:00:00)

G-ProCom\_45:47:dd(00:23:24:45:47:dd)

网络层Target IP Address

10.30.28.1(10.30.28.1)

10.30.28.38(10.30.28.38)

字段项 ARP请求数据报文 ARP应答数据报文

链路层Destination项 Broadcast(ff:ff:ff:ff:ff:ff) G-ProCom\_45:47:dd(00:23:24:45:47:dd)

链路层Source项 G-ProCom\_45:48:16(00:23:24:45:48:16) 0c:da:41:63:03:f4(0c:da:41:63:03:f4)

网络层Sender MAC Address G-ProCom\_45:48:16(00:23:24:45:48:16) 0c:da:41:63:03:f4(0c:da:41:63:03:f4)

网络层Sender IP Address 10.30.28.22(10.30.28.22) 10.30.28.1(10.30.28.1)\_

网络层Target MAC Address 00:00:00\_00:00:00(00:00:00:00:00:00) G-ProCom\_45:47:dd(00:23:24:45:47:dd)

网络层Target IP Address 10.30.28.1(10.30.28.1) 10.30.28.38(10.30.28.38)

练习2：ARP报文是直接封装在以太帧中的，为此以太帧所规定的类型字段值为\_\_\_0806h\_\_\_\_\_\_\_\_\_\_\_\_

练习3：对地址转换协议（ARP）描述正确的是（ D ）

A ARP封装在IP数据报的数据部分

B 发送ARP包需要知道对方的MAC地址

C ARP是用于IP地址到域名的转换

D ARP是采用广播方式发送的

练习4：ARP欺骗的原理是什么？如何进行防范？

ARP欺骗分为二种，一种是对路由器ARP表的欺骗；另一种是对内网PC的网关欺骗。第一种ARP欺骗的原理是——截获网关数据。它通知路由器一系列错误的内网MAC地址，并按照一定的频率不断进行，使真实的地址信息无法通过更新保存在路由器中，结果路由器的所有数据只能发送给错误的MAC地址，造成正常PC无法收到信息。第二种ARP欺骗的原理是——伪造网关。它的原理是建立假网关，让被它欺骗的PC向假网关发数据，而不是通过正常的路由器途径上网。

防范措施：建立DHCP服务器；建立MAC数据库；网关机器关闭机器ARP动态刷新的过程，使用静态路由；网关监听网络安全。

练习5：有人认为：“ARP协议向网络层提供了转换地址的服务，因此ARP应当属于数据链路层。”这种说法为什么是错误的？

ARP是进行地址间的转换，而LLC和MAC均没有包含这样的功能。

练习6：ARP 协议的作用是 （A ）

A．由IP地址查找对应的MAC地址 B．由MAC 地址查找对应的IP 地址

C．由IP地址查找对应的端口号 D．由MAC 地址查找对应的端口号

ARP 报文封装在 （A ） 中传送。

* 1. 以太帧 B．IP 数据报 C．UDP 报文 D．TCP 报文