计算机网络实验实验报告

学院（系）名称：信息学院

姓名	高星杰	**学号**	2021307220712	**专业**	计算机科学与技术
班级	计科 2102	**实验项目**	使用 WireShark 分析网络协议		
课程名称		计算机网络实验		**课程代码**	3103009336
实验时间		10 月 18 日		**实验地点**	逸夫楼 C207

考核内容	目的和原理 40	内容及过程分析 30	实验方案设计 10	实验结果（结论正确性以及分析合理性）20	**成绩**	
各项得分						
考核标准	○原理明确 ○原理较明确 ○原理不明确	○分析清晰 ○分析较清晰 ○分析不清晰	○设计可行 ○设计基本可行 ○设计不可行	○结论正确，分析合理 ○结论正确，分析不充分 ○结论不正确，分析不合理	**教师签字：**	

1. 实验目的：

（1）熟练掌握 Wireshark 的使用方法。

（2）掌握 TCP、IP、ARP、DNS、HTTP 和 Ethernet 等协议的基本原理。

（3）详细分析 HTTP 协议的通信过程。

2. 实验原理：

Wireshark 抓包原理：

Wireshark 使用 WinPCAP 作为接口，直接与网卡进行数据报文交换。WinPcap 是 Windows 平台下的一种网络数据包捕获库，它可以使用户在本地网络上捕获和发送网络数据包 WinPcap 基本上工作在操作系统内核中，它可以通过网络适配器驱动程序捕获包括数据链路层、网络层和传输层等各个层次的数据包，也可以通过同样的驱动程序发送自定义的网络数据包。单机情况下，Wireshark 直接抓取本机网卡的网络流量。

简单来说 wireshark 就是调用的系统 WinPcap 的接口实现的抓包的功能。

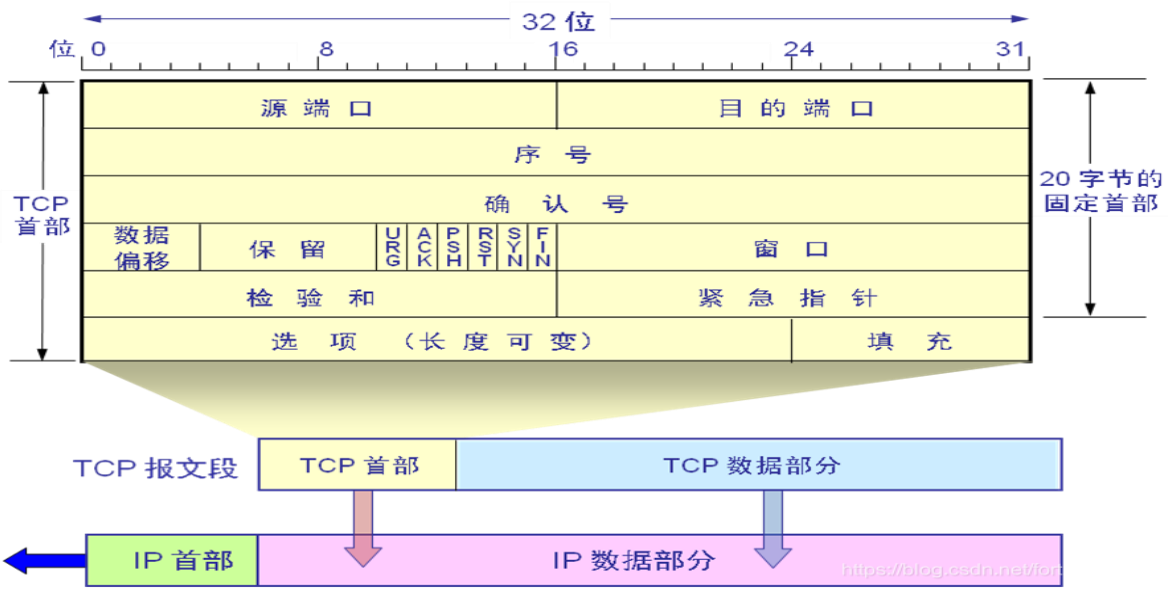
Wireshark 使用的环境大致分为两种，一种是电脑直连网络的单机环境，另外一种就是应用比较多的网络环境，即连接交换机的情况，在这次实验是单机情况，也就是直接抓取的是本机网络卡的网络流量。

总的来说 wireshark 抓取的包是数据链路层的帧，基于此我们就可以对它抓的包的进行分析以达到实验目的。

除了要对 wireshark 的原理有基本的了解还要对链路层往上的协议也要有一定的了解。

TCP 报文段的内容：

Tcp 报文的结构示意图：



源端口(Source Port，2 字节)：源端口，标识哪个应用程序发送。

目的端口(Destination Port，2 字节):目的端口，标识哪个应用程序接收。

序号(Sequence Number，4 字节)：序号字段。TCP 链接中传输的数据流中每个字节都编上一个序号。序号字段的值指的是本报文段所发送的数据的第一个字节的序号。

确认号（Acknowledgment Number，4 字节）：确认号，是期望收到对方的下一个报文段的数据的第 1 个字节的序号，即上次已成功接收到的数据字节序号加 1。只有 ACK 标识为 1，此字段有效。

数据偏移（Data Offset）：数据偏移，即首部长度，指出 TCP 报文段的数据起始处距离 TCP 报文段的起始处有多远，以 32 比特（4 字节）为计算单位。最多有 60 字节的首部，若无选项字段，正常为 20 字节。

保留(Reserved):保留，必须填 0。

URG:紧急指针有效标识。它告诉系统此报文段中有紧急数据，应尽快传送（相当于高优先级的数据）。

ACK:确认序号有效标识。只有当 ACK=1 时确认号字段才有效。当 ACK=0 时，确认号无效。

PSH:标识接收方应该尽快将这个报文段交给应用层。接收到 PSH = 1 的 TCP 报文段，应尽快的交付接收应用进程，而不再等待整个缓存都填满了后再向上交付。

RST:重建连接标识。当 RST=1 时，表明 TCP 连接中出现严重错误（如由于主机崩溃或其他原因），必须释放连接，然后再重新建立连接。

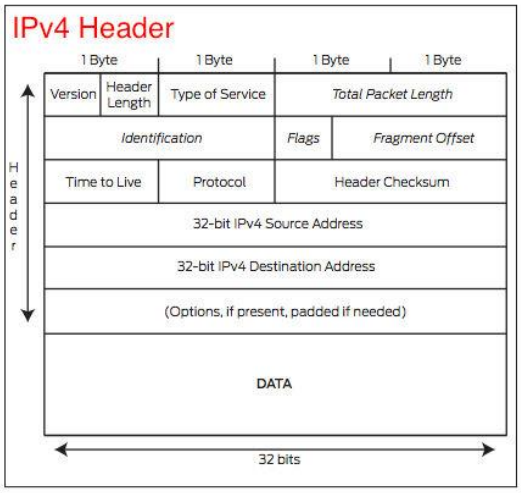
RST:重建连接标识。当 RST=1 时，表明 TCP 连接中出现严重错误（如由于主机崩溃或其他原因），必须释放连接，然后再重新建立连接。

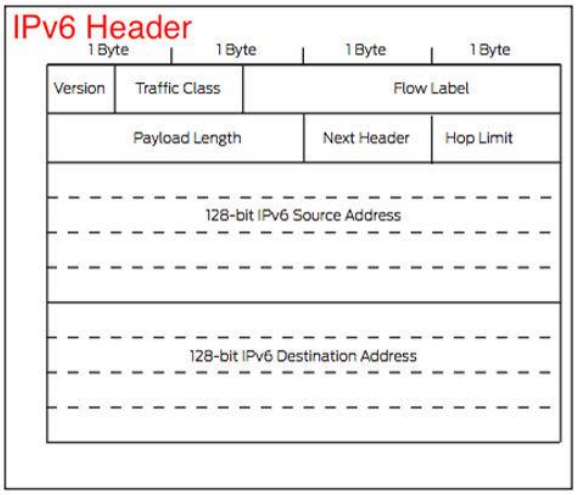
SYN：同步序号，用于连接建立过程，在请求连接的时候，SYN=1 和 ACK=0 表示该数据段没有捎带确认域，而连接应答捎带一个确认，表示为 SYN=1 和 ACK=1

FIN：断开连接标志，用于释放连接，为 1 表示发送方已经没有数据发送，即关闭数据流

IP 数据报的内容：

IP 数据报的结构示意图：





此时 IP 协议有两个版本 IPv4 和 IPv6,并且有不同的版本。

IPv4 数据报格式:

Version:版本，大小为 4bit IP 协议版本号，固定为 4bit

Internet Header Length(IHL): 首部长度，4bit 以 4 字节 为单位,最小值 5(20Byte),最大值 15(60Byte)

Type of Service TOS: 服务类型,8 字节几乎不用

Total Length: 总长度,16bit 整个数据报的长度，2^{16} -1 = 65535 字节，不过由于链路层的 MTU 限制,超过 1480 字节后就会被分片（以太帧 MTU 最大为 1500 - 固定首部 20）

Identification:标识大小为 16bit，报文的唯一标识

Flag:标志,大小为 3bit，是否分片的标志。DF: Don't Fragment; MF: More Fragment ;DF=1:不能分片;DF=0: 允许分片;MF=1: 后面还有分片，MF=0: 最后一个

Fragment Offset:片偏移大小为 13bit 分片在原分组中的相对位置，以 8 个字节 为偏移单位

Time To Live，TTL:生存时间,大小为 8bit 数据报可以经过的最多路由器数，每经一个，值减 1，为 0 时丢弃该报文

Protocol:协议大小为 8bit 封装的协议类型 ICMP(1)、IGMP(2)、TCP(6)、UDP(17)

Header CheckSum :头部校验和,16bit 仅校验数据报的首部，使用二进制反码求和

Source Address:源地址,32bit,源 IP 地址

Destination Address:目的地址,32bit 目标 IP 地址

Options :可选项,可变主要用于测试

Padding:填充，填充 0，确保首部长度为 4 字节的整数倍

Data:数据，报文数据部分

IPv6 数据报格式:

Version:(版本,4)IP 协议版本号，固定为 6

Traffic Class:（通信类型,8）类似于 IPv4 中的 服务类型(TOS)

Flow Label: (流标签,20)识别某些需要特别处理的分组

Payload Length: (载荷长度,16)类似于 IPv4 中的 总长度(Total Length)，区别在于不含基本首部

Next Header: (下一头部,8) 类似于 IPv4 中的 协议(Protocol)

Hop Limit: (跳数限制,8) 类似于 IPv4 中的 生存时间(TTL)

Source Address: (源地址,128)源 IPv6 地址

Destination Address: （目的地址，128）目的 IPv6 地址

Extension Header: (扩展首部，可变)可选择继续使用 IPv4 中首部部分，详见下表

Data: 数据，报文数据部分

ARP 协议原理：

ARP 协议为链路层的通信协议，通过 ARP 协议，可以将网络层的 IP 地址转换为 MAC 地址。MAC 地址一般烧录在硬件网卡设备上，不可修改，所以比 IP 更加具有安全性。

1.Host1 发送数据前设备会先查找自己的 ARP 缓存表，如果有直接封装到帧里进行发送,如 ARP 缓存表没有对应 IP 地址的 MAC 信息，则会通过 ARP 进行获取

2.Host1 会发送 ARP Request 报文来请求获取 Host2 的 MAC 地址（因为帧内没有目的 MAC 地址是不可以进行传输的，所以 ARP 报文内的目的 MAC 地址为全 F）

3.因为 ARP Request 目的 MAC 地址为 FF-FF-FF-FF-FF-FF（广播数据帧），所以交换机收到后会直接对该帧进行泛洪（广播）操作，并且学习该 IP 的 MAC 地址以及端口号到交换机自己的 MAC 缓存表

4.所有主机都接受到该 ARP Request 报文后，都会检查该帧的目的 IP 地址与自身的 IP 地址是否匹配，不匹配就直接丢弃,Host 发现与自己 IP 地址匹配,就会先把发送端的 IP 与 MAC 地址信息记录到自己的 ARP 缓存表之中，然后 Host2 就会发送 ARP Reply 报文（因为刚才进行了学习所以知道 Host1 的 MAC 地址，所以 ARP Reply 是单播数据帧）来进行响应

5.交换机收到单播数据帧以后，会对该帧进行转发操作，并且学习 Host2 的 MAC 地址和端口号到自己的 MAC 缓存表

6.Host1 收到 Host2 的 ARP Reply 报文后会检查目的 IP 与自己 IP 地址字段是否相同，如果匹配就将回应报文的源 IP 地址与 MAC 地址学习到自己的 ARP 缓存表之中，然后就可以传输信息进行通信

DNS 协议原理：

当实验中所有使用域名访问的服务器，都要通过 DNS 协议对域名进行解析才能得到正确的 IP 地址。

DNS 是应用层协议，事实上他是为其他应用层协议工作的，包括不限于 HTTP 和 SMTP 以及 FTP，用于将用户提供的主机名解析为 IP 地址。

具体过程如下：

①用户主机上运行着 DNS 的客户端，就是我们的 PC 机或者手机客户端运行着 DNS 客户端了。

②浏览器将接收到的 url 中抽取出域名字段，就是访问的主机名，比如 http://www.baidu.com/，并将这个主机名传送给 DNS 应用的客户端。

③DNS 客户机端向 DNS 服务器端发送一份查询报文，报文中包含着要访问的主机名字段（中间包括一些列缓存查询以及分布式 DNS 集群的工作）。

④该 DNS 客户机最终会收到一份回答报文，其中包含有该主机名对应的 IP 地址。

⑤一旦该浏览器收到来自 DNS 的 IP 地址，就可以向该 IP 地址定位的 HTTP 服务器发起 TCP 连接。

HTTP 协议工作过程：

一次 HTTP 操作称为一个事务，其工作整个过程如下：

1. 地址解析

如用客户端浏览器请求这个页面：http://localhost.com:8080/index.htm

从中分解出协议名、主机名、端口、对象路径等部分，对于我们的这个地址，解析得到的结果如下：

协议名：http

主机名：localhost.com

端口：8080

对象路径：/index.htm

在这一步，需要域名系统 DNS 解析域名 localhost.com,得主机的 IP 地址。

2. 封装 HTTP 请求数据包

把以上部分结合本机自己的信息，封装成一个 HTTP 请求数据包

3. 封装成 TCP 包，建立 TCP 连接（TCP 的三次握手）

在 HTTP 工作开始之前，客户机（Web 浏览器）首先要通过网络与服务器建立连接，该连接是通过 TCP 来完成的，该协议与 IP 协议共同构建 Internet，即著名的 TCP/IP 协议族，因此 Internet 又被称作是 TCP/IP 网络。HTTP 是比 TCP 更高层次的应用层协议，根据规则，只有低层协议建立之后才能，才能进行更

层协议的连接，因此，首先要建立 TCP 连接，一般 TCP 连接的端口号是 80。这里是 8080 端口

4. 客户机发送请求命令

建立连接后，客户机发送一个请求给服务器，请求方式的格式为：统一资源标识符（URL）、协议版本号，后边是 MIME 信息包括请求修饰符、客户机信息和可内容。

5. 服务器响应

服务器接到请求后，给予相应的响应信息，其格式为一个状态行，包括信息的协议版本号、一个成功或错误的代码，后边是 MIME 信息包括服务器信息、实体信息和可能的内容。

实体消息是服务器向浏览器发送头信息后，它会发送一个空白行来表示头信息的发送到此为结束，接着，它就以 Content-Type 应答头信息所描述的格式发送用户所请求的实际数据

6. 服务器关闭 TCP 连接

一般情况下，一旦 Web 服务器向浏览器发送了请求数据，它就要关闭 TCP 连接，然后如果浏览器或者服务器在其头信息加入了这行代码

Connection:keep-alive

TCP 连接在发送后将仍然保持打开状态，于是，浏览器可以继续通过相同的连接发送请求。保持连接节省了为每个请求建立新连接所需的时间，还节约了网络带宽。

Ethernet 协议原理

以太网，实现链路层的数据传输和地址封装（MAC）

字段：

Destination：目标地址

Source：源地址

Type：标志上层协议，ARP（0x0806），IPv4（0x0800）等

前导码	帧起始界定符	目的 MAC	源 MAC	类型/长度	数据	校验

3. 实验分析：

在清楚了一些实验原理之后我们可以对实验进行分析。

对于 WireShark 抓包，我们要着重观察各个协议的流，然后通过抓网络层的包来验证每个协议实际的工作是如何实现的，这里要主要观察两个问题：一个是工作流程是怎么样的，第二个是工作流程的是怎么实现的。

对于打开一个网页的流程

如果你在浏览器地址栏里键入百度的地址：www.baidu.com/ 那么接下来，浏览器会完成哪些动作呢？下面我们就一步一步详细“追踪”下。

1. 构建请求

首先，浏览器构建请求行信息（如下所示），构建好后，浏览器准备发起网络请求。

bash 复制代码 GET /index.html HTTP1.1

2. 查找缓存

在真正发起网络请求之前，浏览器会先在浏览器缓存中查询是否有要请求的文件。其中，浏览器缓存是一种在本地保存资源副本，以供下次请求时直接使用的技术。

当浏览器发现请求的资源已经在浏览器缓存中存有副本，它会拦截请求，返回该资源的副本，并直接结束请求，而不会再去源服务器重新下载。这样做的好处有：

缓解服务器端压力，提升性能（获取资源的耗时更短了）；

对于网站来说，缓存是实现快速资源加载的重要组成部分。

当然，如果缓存查找失败，就会进入网络请求过程了。

3. 准备 IP 地址和端口

不过，先不急，在了解网络请求之前，我们需要先看看 HTTP 和 TCP 的关系。因为浏览器使用 HTTP 协

议作为应用层协议，用来封装请求的文本信息；并使用 TCP/IP 作传输层协议将它发到网络上，所以在 HTTP 工作开始之前，浏览器需要通过 TCP 与服务器建立连接。也就是说 HTTP 的内容是通过 TCP 的传输数据阶段来实现的。

那接下来你可以想到：

HTTP 网络请求的第一步是做什么呢？是和服务器建立 TCP 连接。

那建立连接的信息都有了吗？而建立 TCP 连接的第一步就是需要准备 IP 地址和端口号。

那怎么获取 IP 地址和端口号呢？这得看看我们现在有什么，我们有一个 URL 地址，那么是否可以利用 URL 地址来获取 IP 和端口信息呢？

我们知道数据包都是通过 IP 地址传输给接收方的。由于 IP 地址是数字标识，难以记忆，但使用域名就好记多了，所以基于这个需求又出现了一个服务，负责把域名和 IP 地址做一一映射关系。这套域名映射为 IP 的系统就叫做“域名系统”，简称 DNS（Domain Name System）。

所以，这样一路推导下来，你会发现在第一步浏览器会请求 DNS 返回域名对应的 IP。当然浏览器还提供了 DNS 数据缓存服务，如果某个域名已经解析过了，那么浏览器会缓存解析的结果，以供下次查询时直接使用，这样也会减少一次网络请求。

拿到 IP 之后，接下来就需要获取端口号了。通常情况下，如果 URL 没有特别指明端口号，那么 HTTP 协议默认是 80 端口。

4. 等待 TCP 队列

现在已经把端口和 IP 地址都准备好了，那么下一步是不是可以建立 TCP 连接了呢？

答案依然是“不行”。Chrome 有个机制，同一个域名同时最多只能建立 6 个 TCP 连接，如果在同一个域名下同时有 10 个请求发生，那么其中 4 个请求会进入排队等待状态，直至进行中的请求完成。

当然，如果当前请求数量少于 6，会直接进入下一步，建立 TCP 连接。

5. 建立 TCP 连接

排队等待结束之后，终于可以快乐地和服务器握手了，在 HTTP 工作开始之前，浏览器通过 TCP 与服务器建立连接。而 TCP 的工作方式，我之前发过文章详细介绍过，你可以自行回顾，这里我就不再重复讲述了。

6. 发送 HTTP 请求

一旦建立了 TCP 连接，浏览器就可以和服务器进行通信了。而 HTTP 中的数据正是在这个通信过程中传输的。

不妨再来理解一下浏览器是如何发送请求信息给服务器的。

首先浏览器会向服务器发送请求行，它包括了请求方法、请求 URI（Uniform Resource Identifier）和 HTTP 版本协议。

发送请求行，就是告诉服务器浏览器需要什么资源，最常用的请求方法是 Get。比如，直接在浏览器地址栏键入域名，这就是告诉服务器要 Get 它的首页资源。

另外一个常用的请求方法是 POST，它用于发送一些数据给服务器，比如登录一个网站，就需要通过 POST 方法把用户信息发送给服务器。如果使用 POST 方法，那么浏览器还要准备数据给服务器，这里准备的数据是通过请求体来发送。

在浏览器发送请求行命令之后，还要以请求头形式发送其他一些信息，把浏览器的一些基础信息告诉服务器。比如包含了浏览器所使用的操作系统、浏览器内核等信息，以及当前请求的域名信息、浏览器端的 Cookie 信息，等等。

对于打开一个网页服务器的反应是什么：

历经千辛万苦，HTTP 的请求信息终于被送达了服务器。接下来，服务器会根据浏览器的请求信息来准备相应的内容。

1. 返回请求

一旦服务器处理结束，便可以返回数据给浏览器了。可以通过工具软件 curl 来查看返回请求数据，具体使用方法是在命令行中输入以下命令：

curl -i https://XXX

注意这里加上了-i 是为了返回响应行、响应头和响应体的数据。

首先服务器会返回响应行，包括协议版本和状态码。

但并不是所有的请求都可以被服务器处理的，那么一些无法处理或者处理出错的信息，怎么办呢？服务器会通过请求行的状态码来告诉浏览器它的处理结果，比如：

最常用的状态码是 200，表示处理成功；

如果没有找到页面，则会返回 404。

状态码类型很多，这里我就不过多介绍了，网上有很多资料，可以自行查询和学习。

随后，正如浏览器会随同请求发送请求头一样，服务器也会随同响应向浏览器发送响应头。响应头包含了服务器自身的一些信息，比如服务器生成返回数据的时间、返回的数据类型（JSON、HTML、流媒体等类型），以及服务器要在客户端保存的 Cookie 等信息。

发送完响应头后，服务器就可以继续发送响应体的数据，通常，响应体就包含了 HTML 的实际内容。

以上这些就是服务器响应浏览器的具体过程。

2. 断开连接

通常情况下，一旦服务器向客户端返回了请求数据，它就要关闭 TCP 连接。不过如果浏览器或者服务器在其头信息中加入了：

Connection:Keep-Alive

那么 TCP 连接在发送后将仍然保持打开状态，这样浏览器就可以继续通过同一个 TCP 连接发送请求。保持 TCP 连接可以省去下次请求时需要建立连接的时间，提升资源加载速度。比如，一个 Web 页面中内嵌的图片就都来自同一个 Web 站点，如果初始化了一个持久连接，你就可以复用该连接，以请求其他资源，而不需要重新再建立新的 TCP 连接。

3. 重定向

到这里似乎请求流程快结束了，不过还有一种情况你需要了解下，比如当你在浏览器中打开某个网址，你会发现最终打开的页面地址是不一样的。

这两个 URL 之所以不一样，是因为涉及到了一个重定向操作。

响应行返回的状态码一般是 301，状态 301 就是告诉浏览器，我需要重定向到另外一个网址，而需要重定向的网址正是包含在响应头的 Location 字段中，接下来，浏览器获取 Location 字段中的地址，并使用该地址重新导航，这就是一个完整重定向的执行流程。

不过也不要认为这种跳转是必然的。如果你打开 12306.cn，你会发现这个站点是打不开的。这是因为 12306 的服务器并没有处理跳转，所以必须要手动输入完整的 www.12306.com 才能打开页面。

我们分析了打开一个网页会引发一列的操作，然后我们可以基于此对实验进行设计，以达到实验目的。

4. 实验设计：

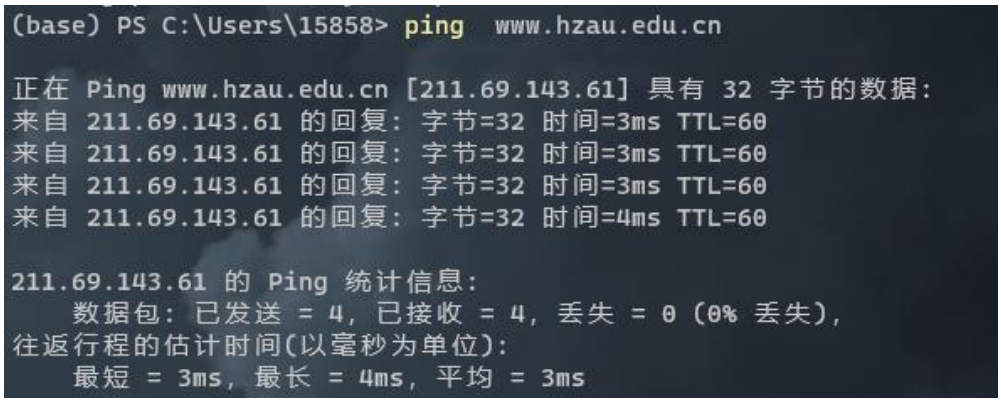
1. 熟悉 Wireshark 软件的界面和操作
2. 使用 Wireshark 抓取浏览器的数据包
3. 分析验证是否和之前实验分析的打开一个网页的流程是否相同
4. 打开数据包，根据实验原理分析其中的数据结构是否相同
5. 分析应用层的数据包的结构
6. 分析传输层的数据包的结构
7. 分析链路层的数据包的结构
8. 尝试抓取浏览器下载文件的包
9. 分析该包，并尝试从包中恢复出来数据。
10. 回答实验问题

5. 实验过程：

1. 安装并启动 Wireshark。选择网卡，根据需要设置过滤条件，开始抓包。

2. 在浏览器中输入网址：http://www.hzau.edu.cn/，保存首页图片，输入网址 www.people.com.cn。对网站进行操作。

   首先我们根据这个网址找到对应的 ip，这里使用 ping 指令来获取



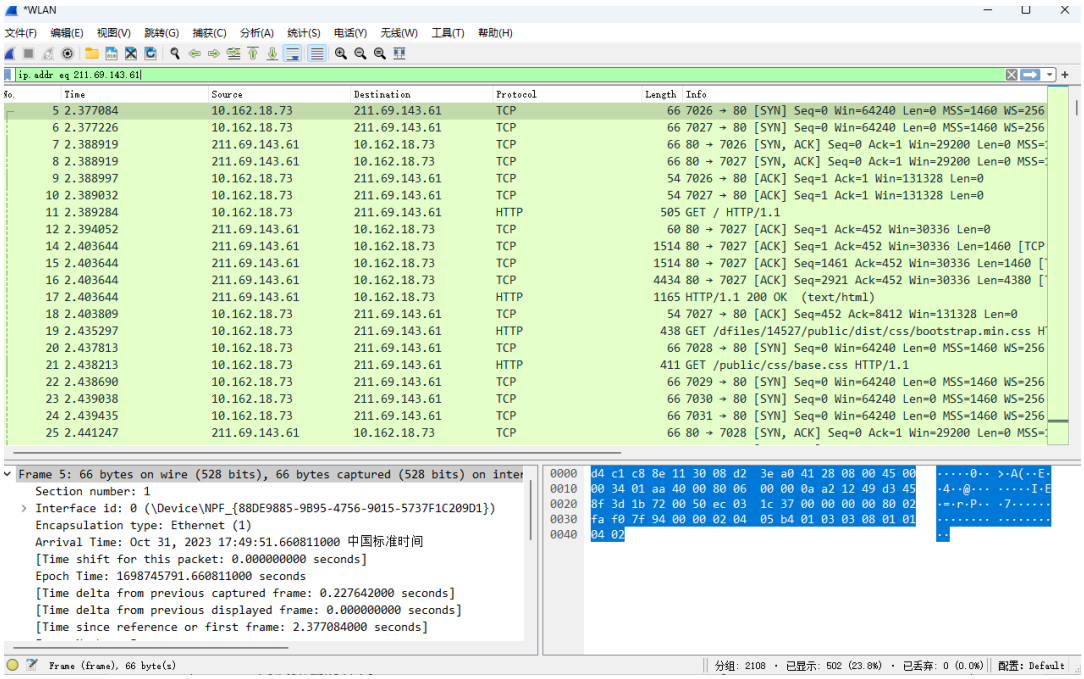
   可以得知 ip 为 211.69.143.61

3. 停止抓包，保存抓包数据。

   然后我们根据这个 IP 去 wireshark 中找到这个有关这个 ip 的数据包

   使用这个筛选 ： ip.addr eq 211.69.143.61

   筛选结果：

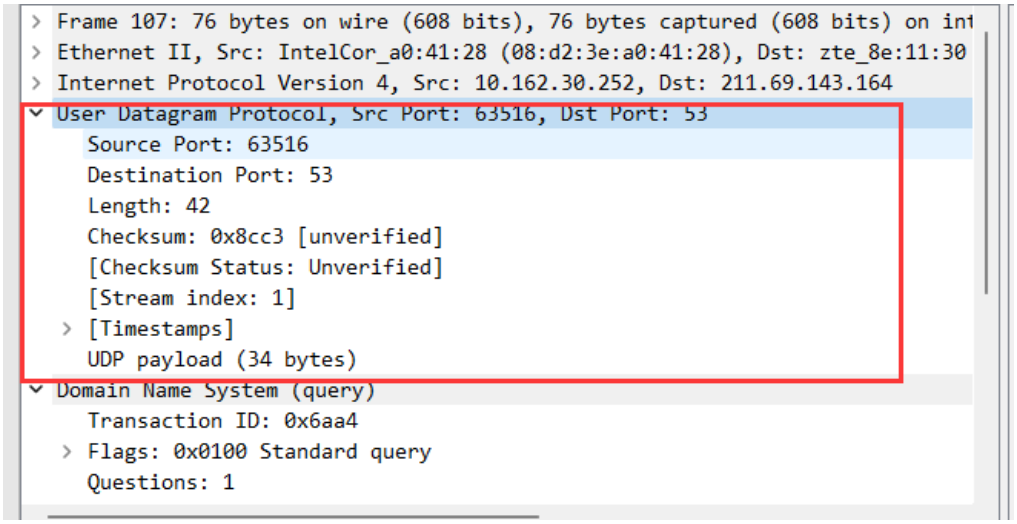


可以发现得到了很多数据报，下面我们就来分析它们。

4. 测试 UDP 协议

5. 停止抓包，保存抓包数据为

通过点开网页，然后就会有 DNS 来查询域名的 ip，其中就有 udp 协议的报文头



其中 User Datagram Protocol 就是 udp 的数据头

可以看到其中有 Source Port 和 Destination Port

6. 两次抓包的种类：TCP、HTTP、DNS、ARP、UDP、TLSv1.2、QUIC、STP。

7. 列表写出客户端、网关及 WEB 服务器的 IP 地址和 MAC 地址。HTTP 客户端和服务端的端口号。

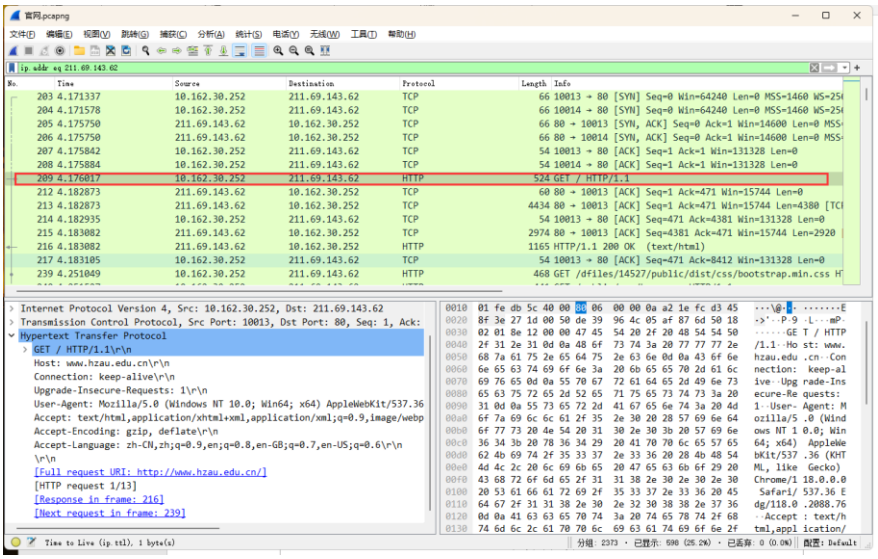
	Clint	Gateway	Server
IP	10.162.30.252	10.162.255.253	211.69.143.62
MAC	08:d2:3e:a0:41:28	d4-c1-c8-8e-11-30	d4-c1-c8-8e-11-30
Port#	10013	-------	80

其中 Gateway 的 ip 是通过 ipconfig 查看的，mac 地址是通过查看 arp 表查询到的

然后可以发现的我通过查询的 webserver 发来的数据包，找到了源 mac 地址，但是与网关的 mac 地址相同，但是 Mac 地址是全球唯一的一个地址，**不可能重复的**，所以猜测该 MAC 地址不是 webserver 的 mac 地址，真正的 WebServer 地址只有离 **WebServer 最近的路由器才知道。**

8. 将 TCP、IP、HTTP 和 Ethernet 的首部字段的名字和值按照协议的格式分别记录下来。

访问指定服务器的第一个 HTTP 请求报文格式：



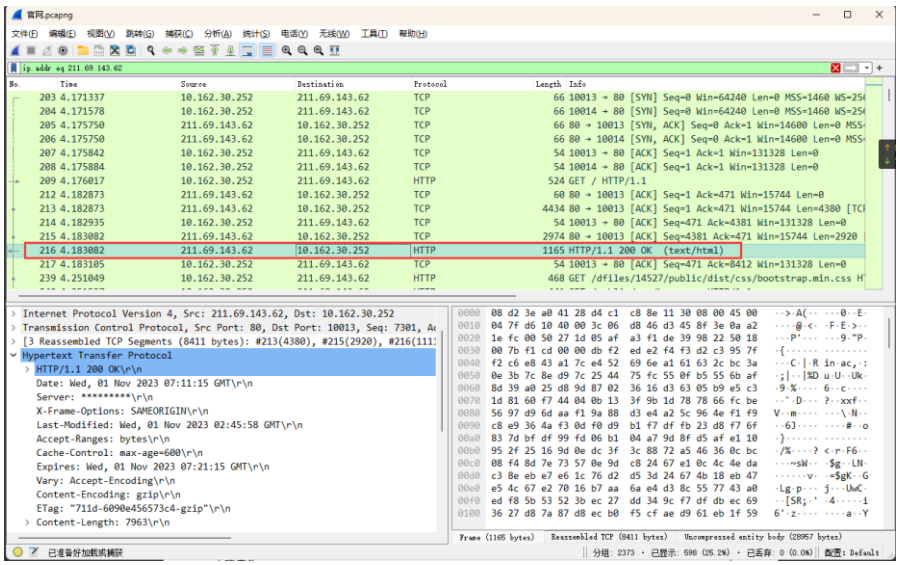
请求行：

GET	/	HTTP/1.1

请求头：

首部行	值
Host	www.hazu.edu.cn
Connection	Keep-alive
Upgrade-Insecure-Requests	1
User-Agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36 Edg/118.0.2088.76
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding	gzip, deflate
Accept-Language	zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6

服务器返回的第二个 HTTP 响应报文格式：



状态行：

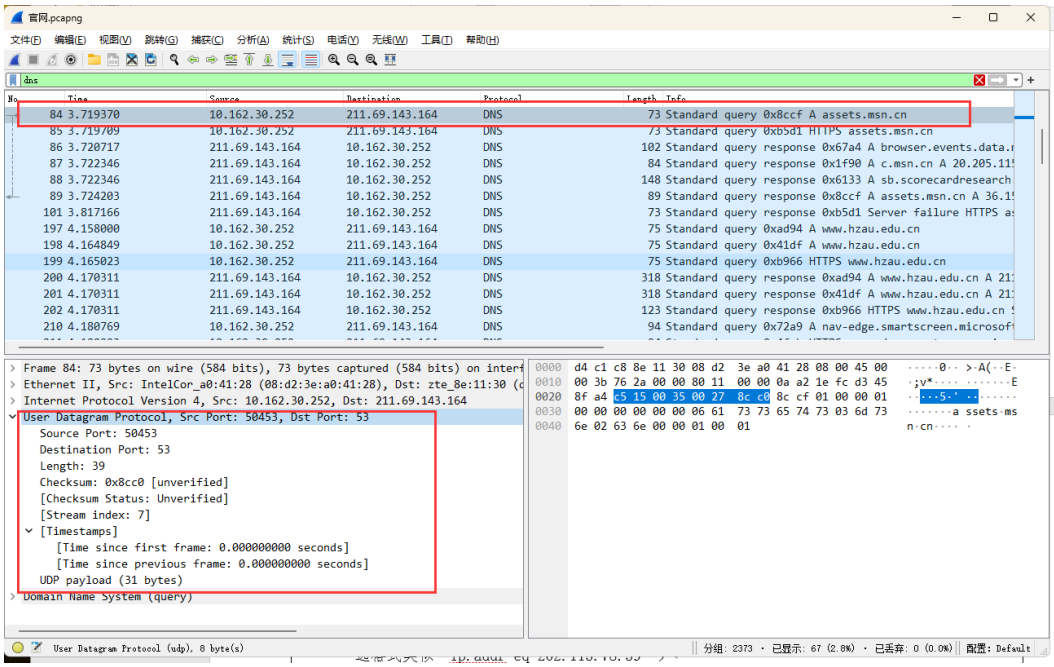
HTTP/1.1	200	OK

响应头：

首部行	值
Data	Wed, 01 Nov 2023 07:11:15 GMT
Server	*********
X-Frame-Options	SAMEORIGIN
Last-Modified	Wed, 01 Nov 2023 02:45:58 GMT

Accept-Ranges	bytes
Cache-Control	max-age=600
Expires	Expires: Wed, 01 Nov 2023 07:21:15 GMT
Vary	Accept-Encoding
Content-Encoding	gzip
ETag	711d-6090e456573c4-gzip
Content-Length	7963
Keep-Alive	timeout=5, max=100
Connection	Keep-Alive
Content-Type	text/html
Content-Language	zh-CN

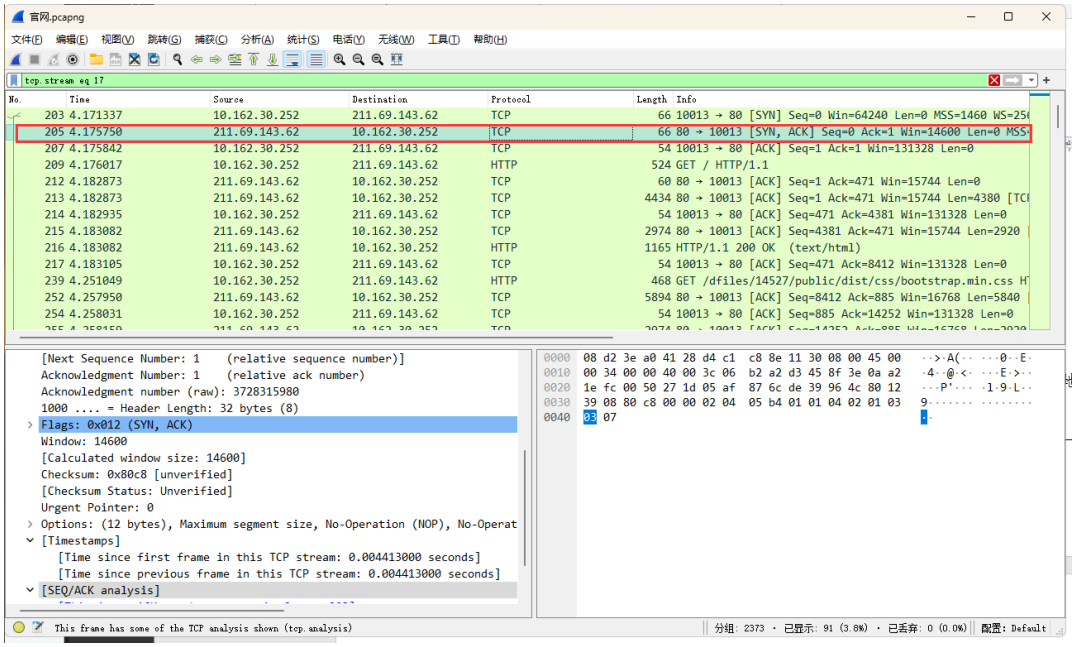
任意一个 UDP 报文首部:



报文首部:

50453	53
39	0x8cc0

与服务器建立连接的第二次握手的 TCP 报文格式:

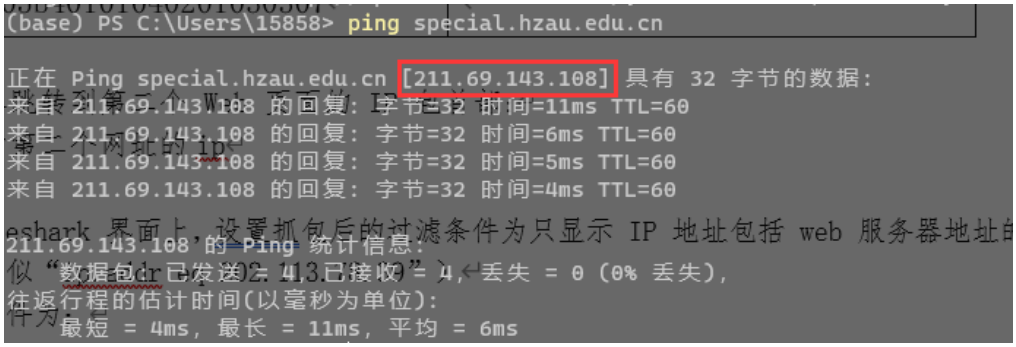


这是第二次握手

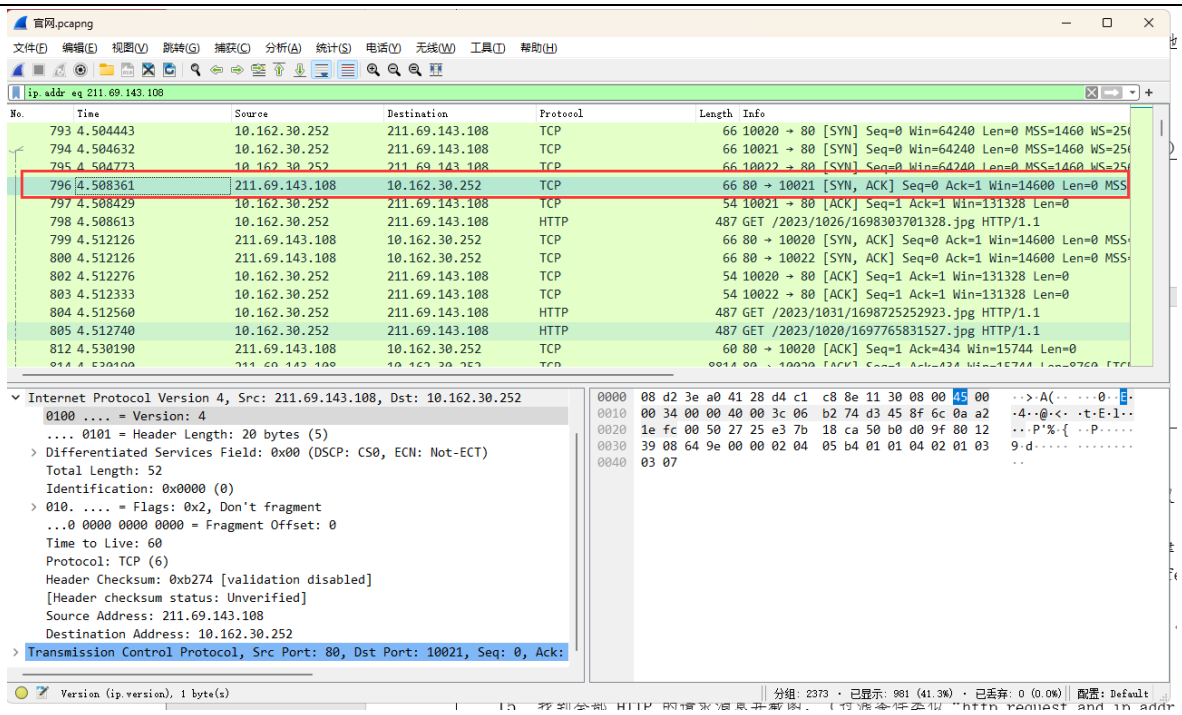
80			10013
95389548			
3728315980			
32	000	00001001	14600
0x80c8			0
020405b401010402010303 07			

浏览器跳转到第二个 Web 页面的 IP 包首部:

跳转后第二个网址的 ip



然后筛选对应的 ip



4	20	0x00	52
0x0000		010	0
60	TCP	0xb274	
211. 69. 143. 108			
10. 162. 30. 252			

任意 802.3 帧头：

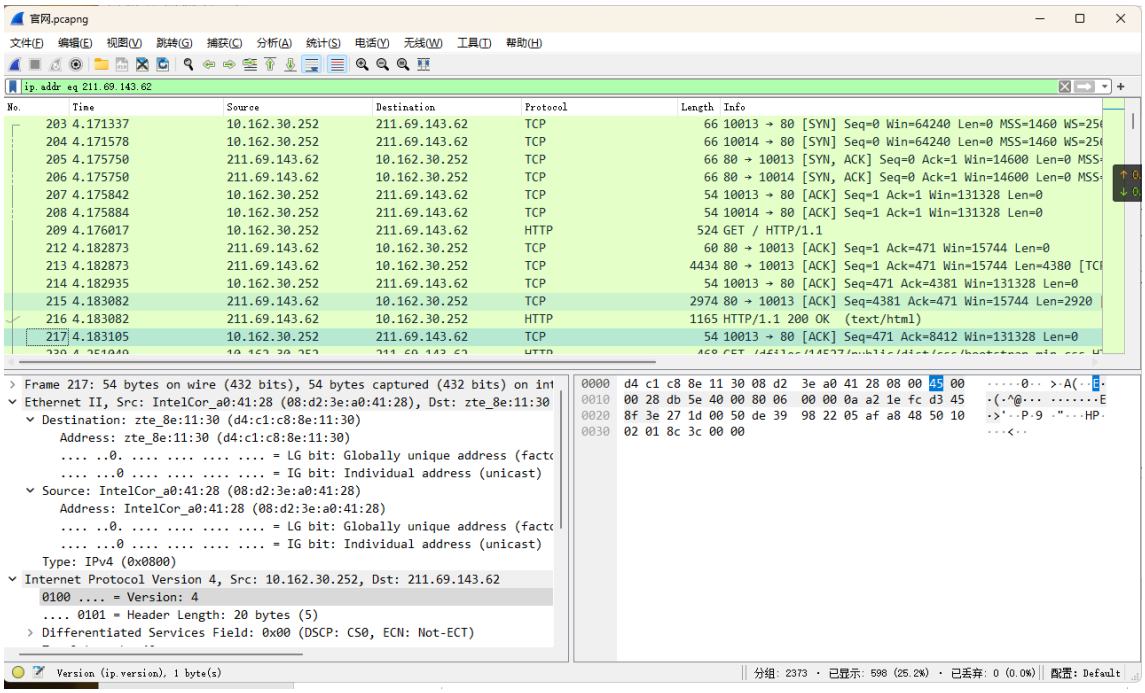
```
Ethernet II, Src: zte_8e:11:30 (d4:c1:c8:8e:11:30), Dst: IntelCor_a0:41:28 (08:d2:3e:a0:41:28)
  Destination: IntelCor_a0:41:28 (08:d2:3e:a0:41:28)
      Address: IntelCor_a0:41:28 (08:d2:3e:a0:41:28)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  Source: zte_8e:11:30 (d4:c1:c8:8e:11:30)
      Address: zte_8e:11:30 (d4:c1:c8:8e:11:30)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  Type: IPv4 (0x0800)
```

08:d2:3e:a0:41:28	d4:c1:c8:8e:11:30	0x0800

9. 在 Wireshark 界面上，设置抓包后的过滤条件为只显示 IP 地址包括 web 服务器地址的包

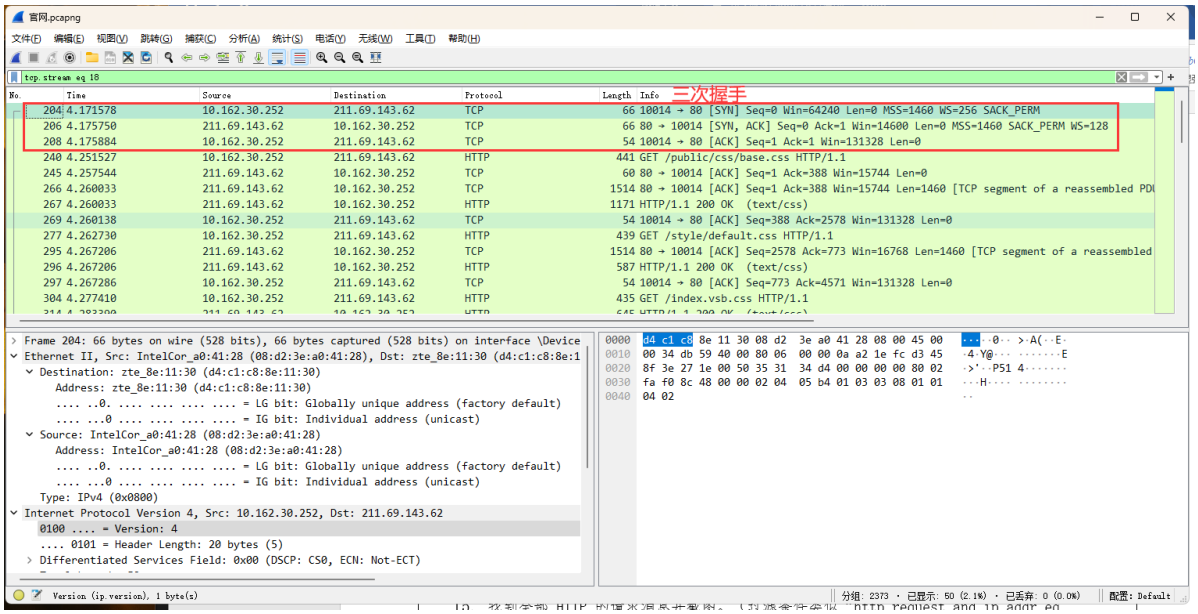
筛选条件为：ip.addr eq 211.69.143.62

筛选结果为：

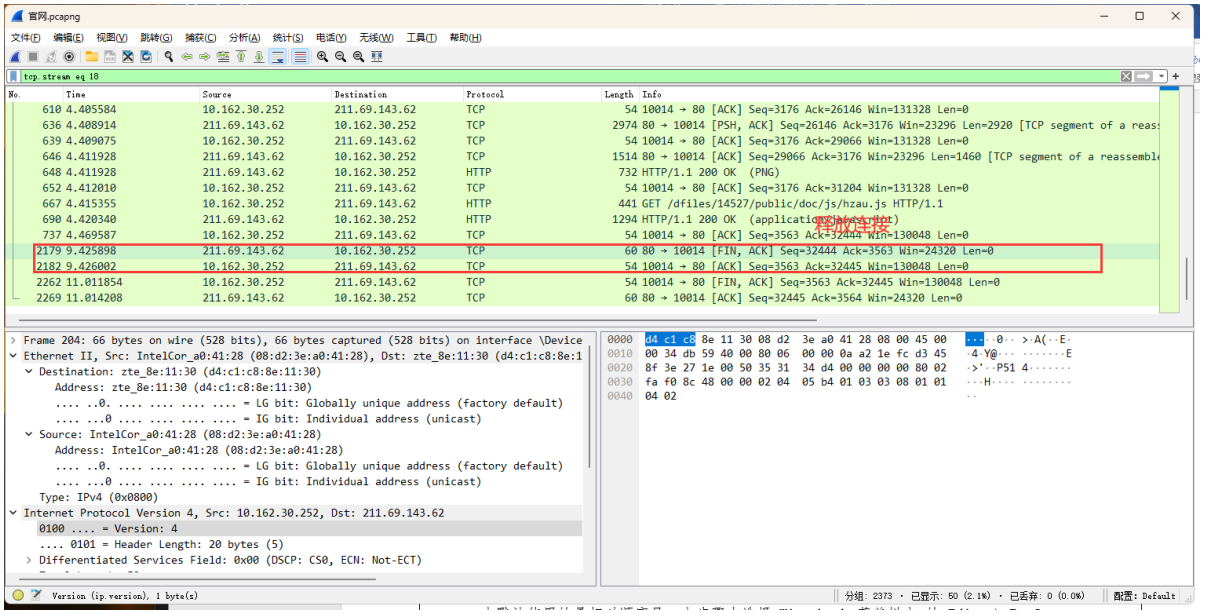


10. 在 Wireshark 界面上分别圈出 TCP 建立连接和释放连接的数据包。（抓图 展示）

建立连接过程为：



释放连接过程为：

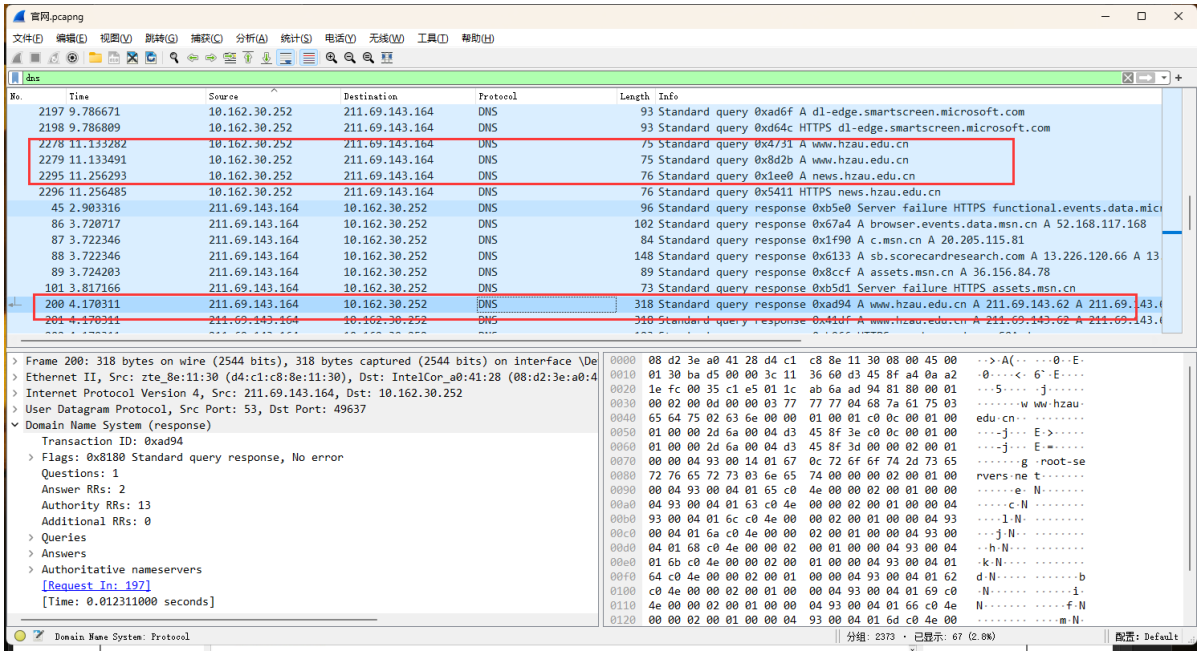


11. 在 Wireshark 界面上圈出你的主机如何找到 Web 服务器的 MAC 地址（ARP 协议）或者 IP 地址（DNS 协议）。

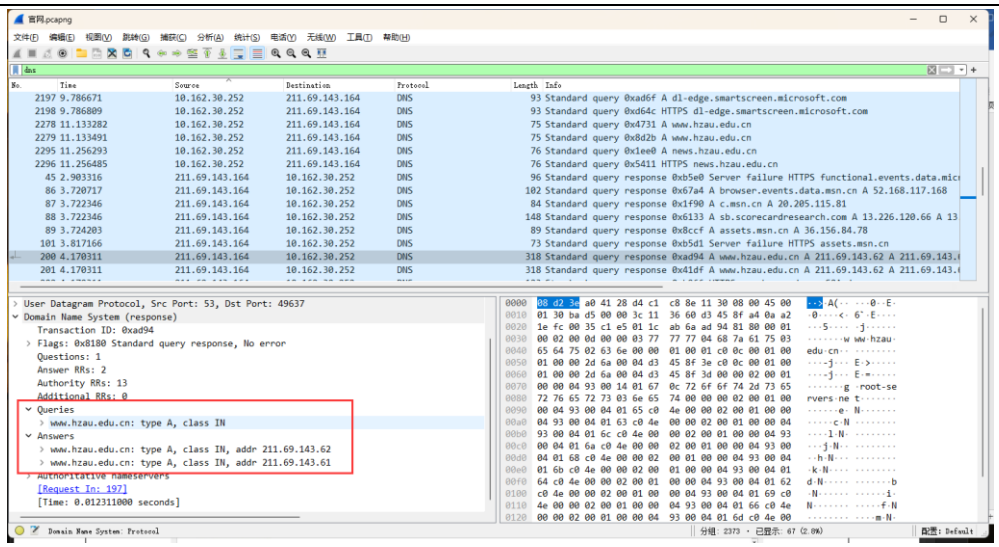
因为和 web 服务器不在同一个局域网下，所以就不能获取到 web 服务器的 Mac 地址。802 数据包中的地址是网关的 mac 地址而不是 web 服务器的。

只能通过 DNS 找到他的 ip 地址

如图，首先通过 DNS 的请求来是实现了对域名的查询

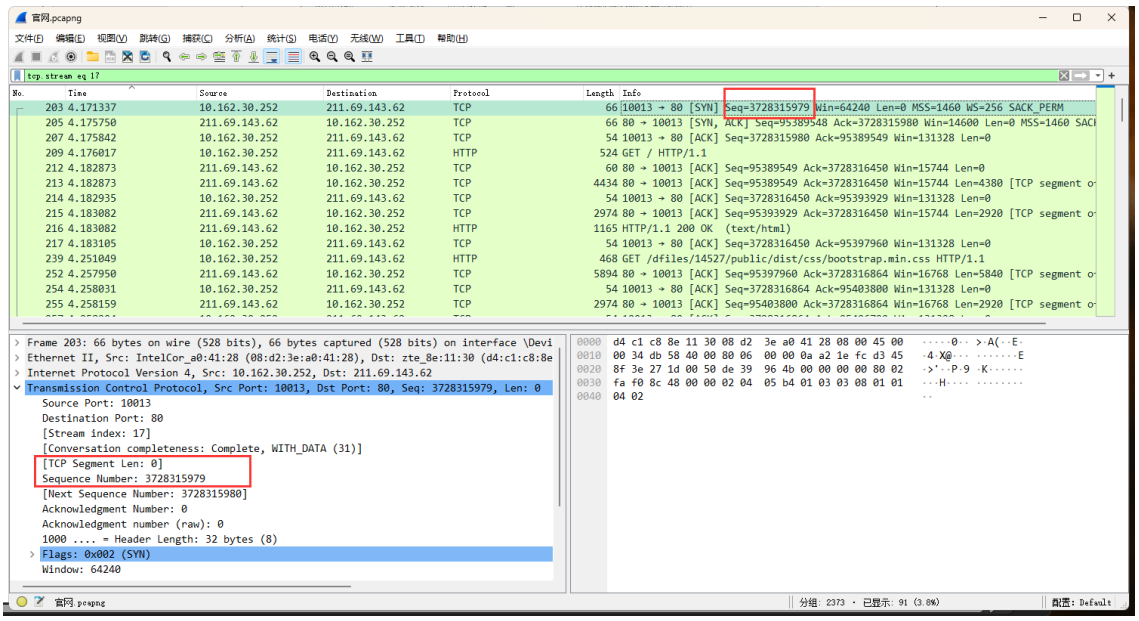


然后又有 DNS 返回了该请求，对应该域名的 ip 地址就该返回中。

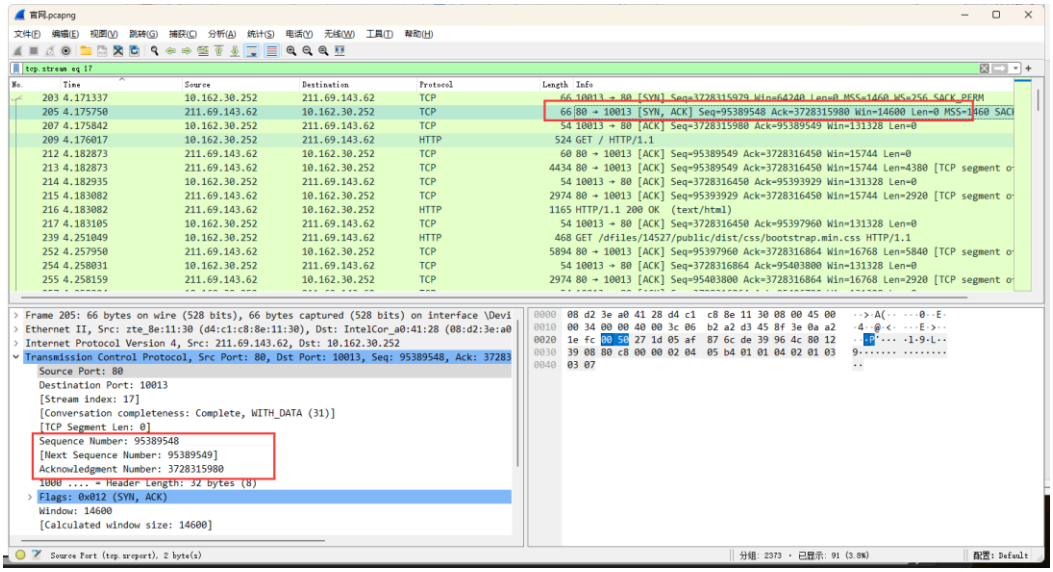


12. 依据实际抓到的数据包，截图并圈出 TCP 顺序号和确认号的使用方法及 变化规律。

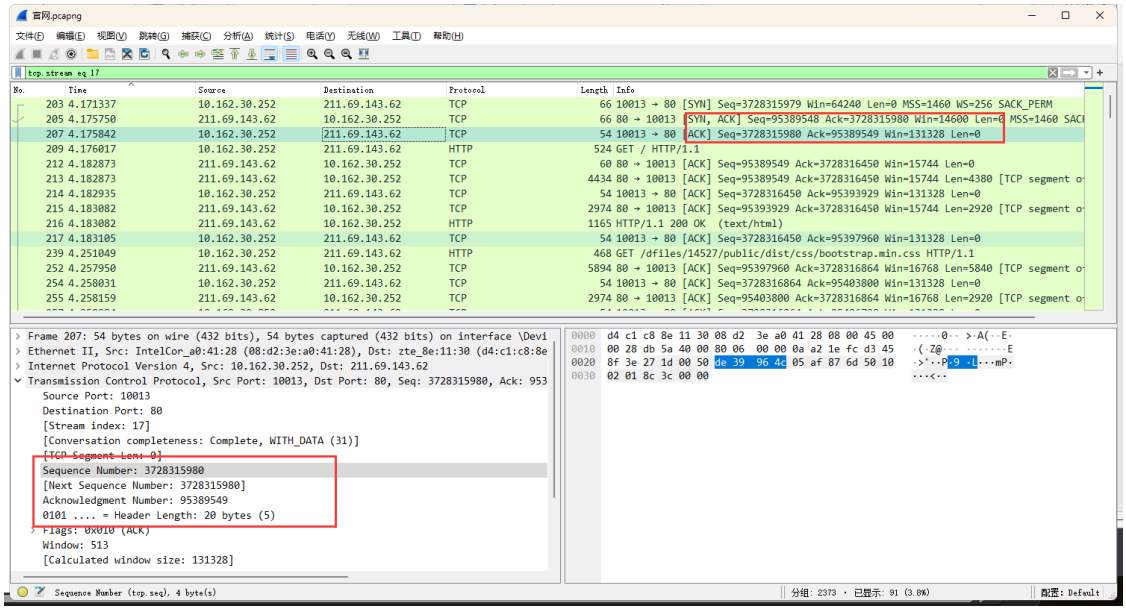
TCP 连接时，第一次握手时随机生成一个序列号 Sequence Number 3728315980,确认号 Acknowledgement Number 为 0;



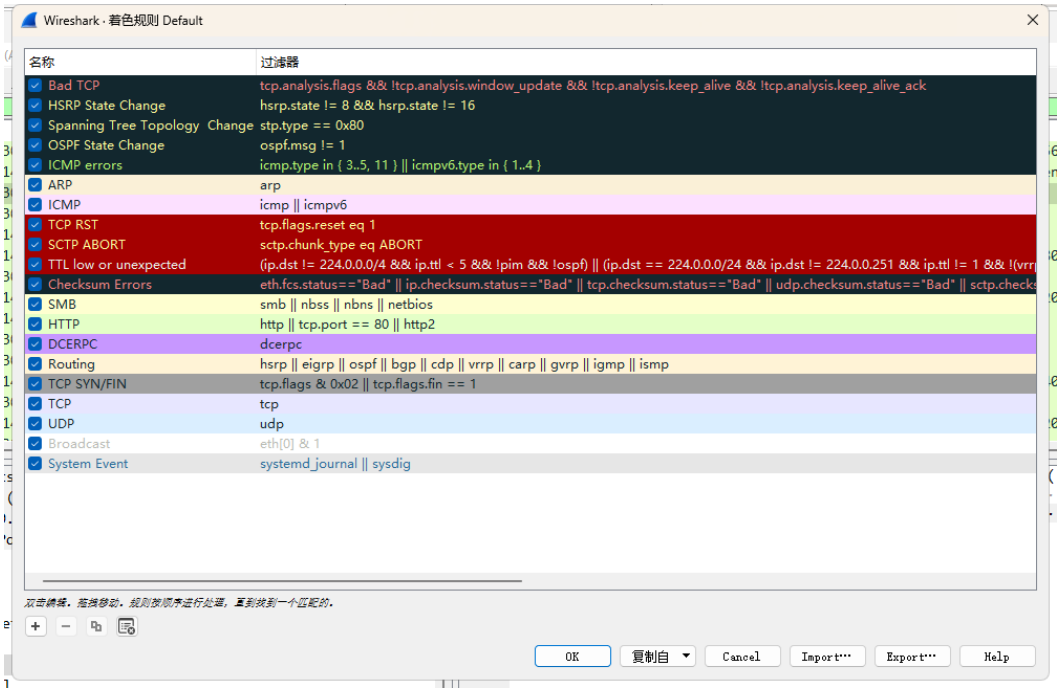
第二次握手时服务端随机生成一个序列号 Sequence Number 95389548,确认号 Acknowledgement Number 为第一次握手时发送的序列号 3728315980



第三次握手时客户端序列号为第二次握手时的确认号,确认号为第二次握手时的序列号加一

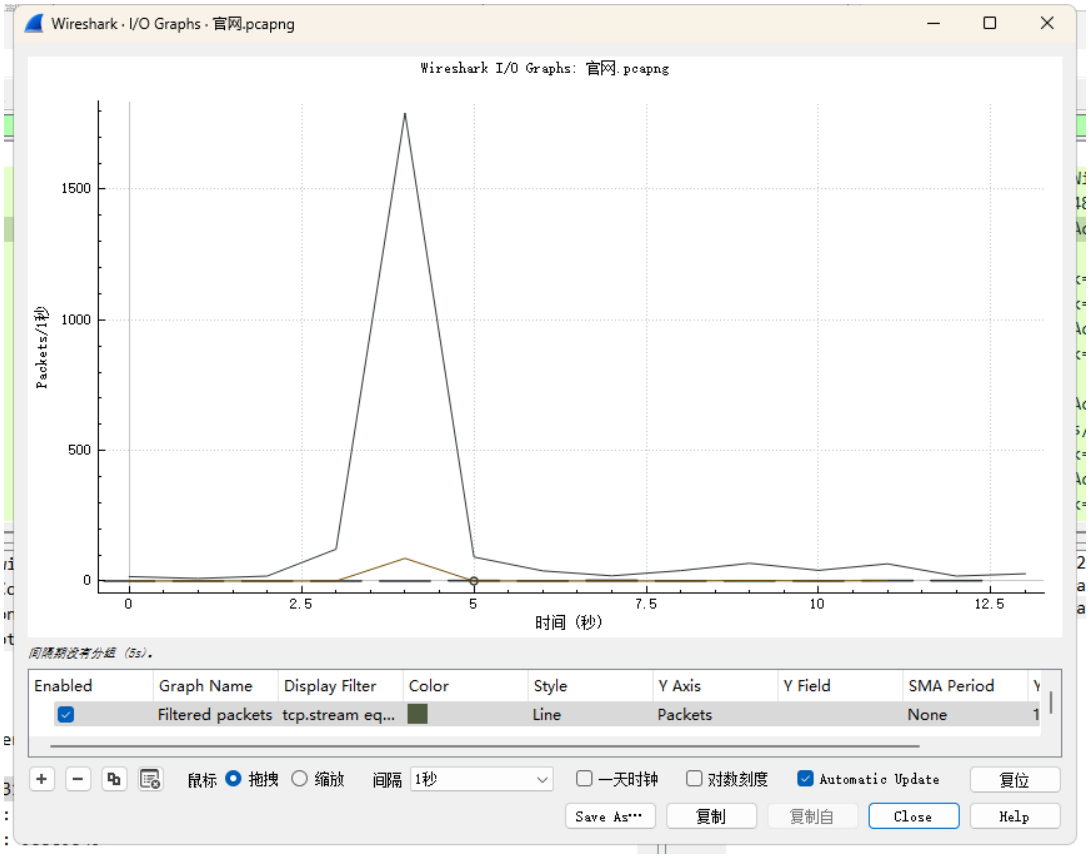


13. 在你所抓到的各种类型数据包中，在 Wireshark 的主界面上是以何种底 纹标注？



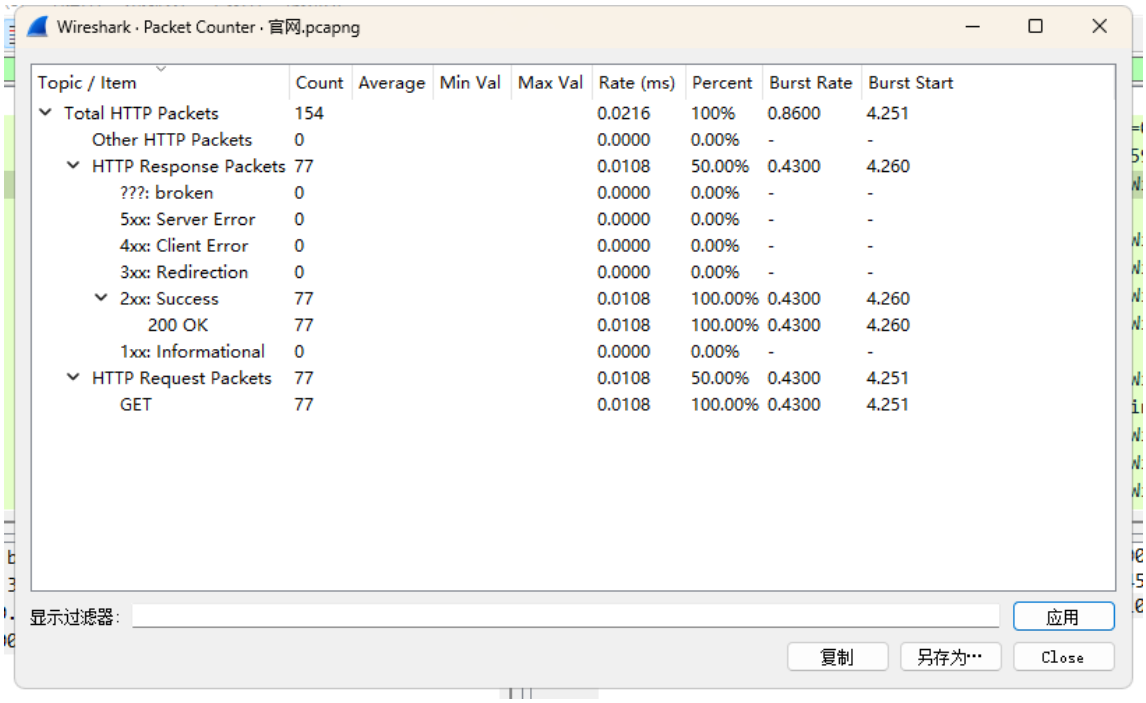
14. 尝试使用 Statistics 菜单中“IO graph”、“HTTP”、“Protocol Hierarchy” 等功能，并记录结果。

IO gragh:显示当前接入网的 IO 数据流的流通情况。

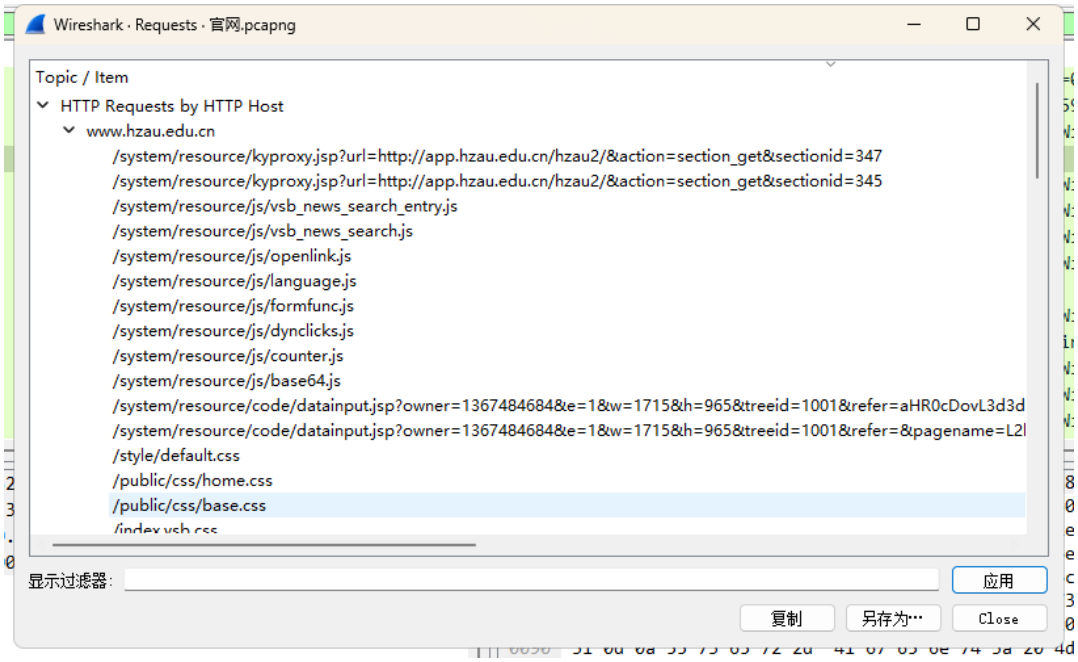


HTTP:

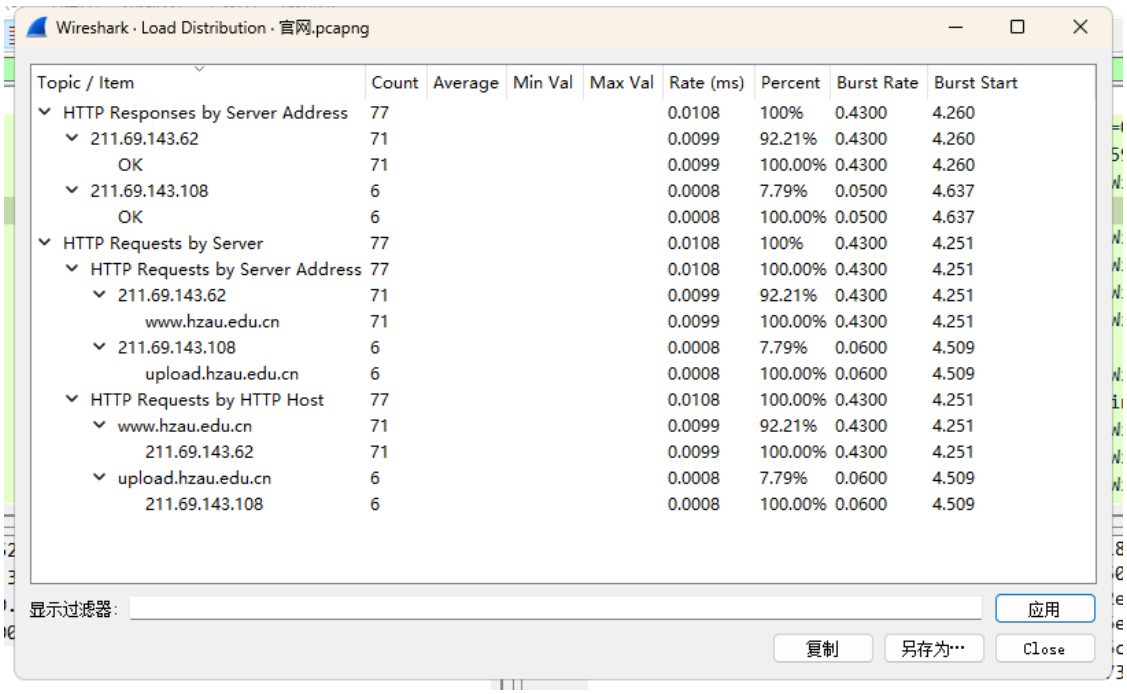
(1)分组计数器：记录了 HTTP 协议中，每种状态码的请求数量统计、每种 GET、PUT 等请求类型的 请求、还有请求与响应统计等等。



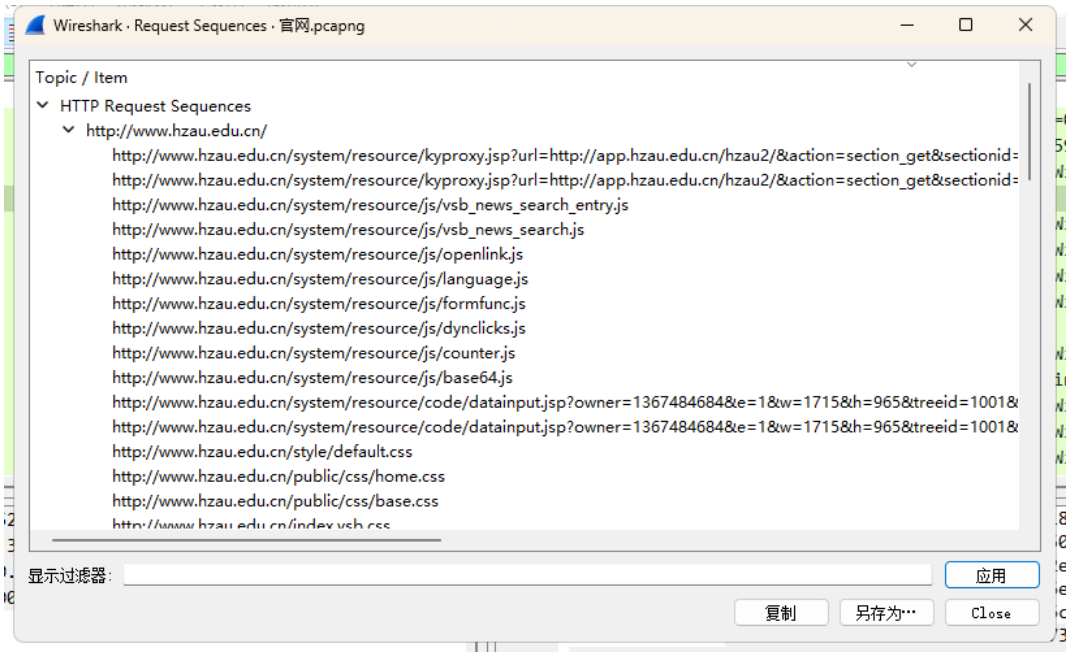
（2）请求：对每一种请求进行了处理，显示其 URL 以及请求的资源目录。



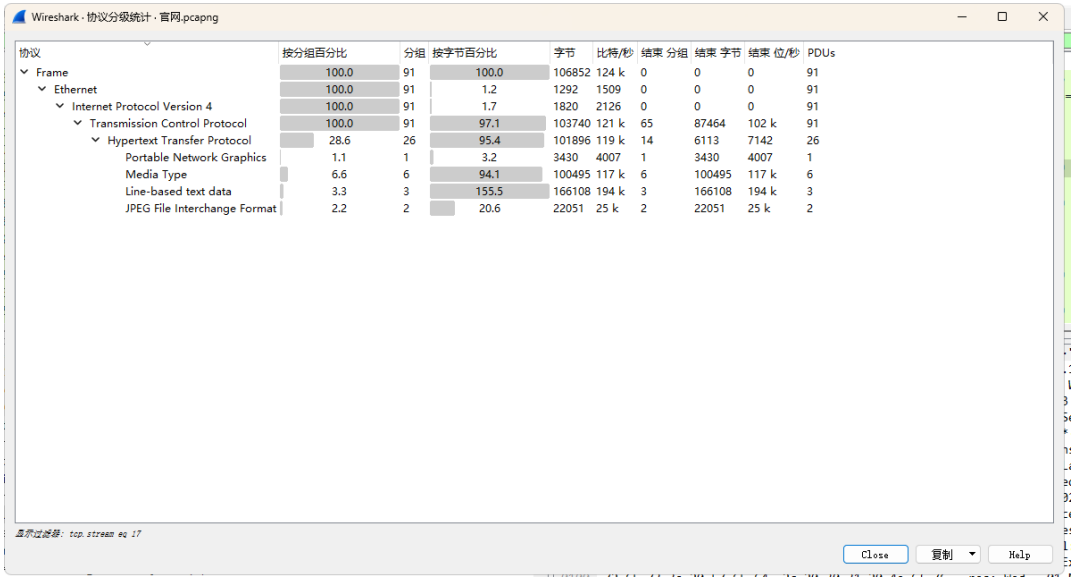
（3）负载分配：显示各服务器地址占用的资源负载



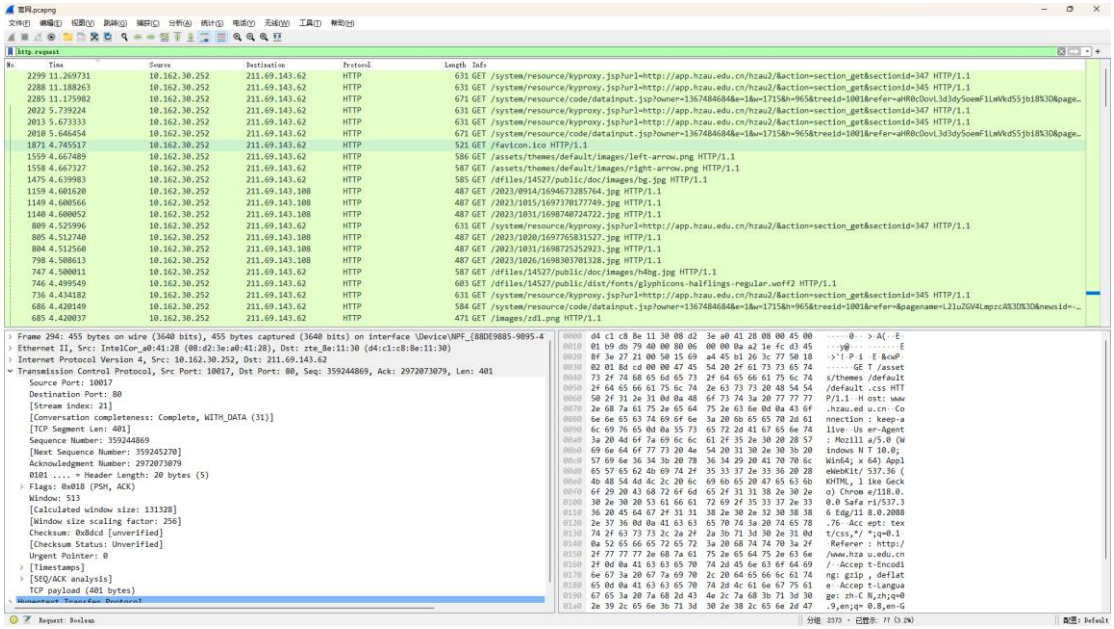
（4）请求序列：显示 HTTP 请求的 URL 序列。



Protocol Hierarchy:显示各个协议的占比。

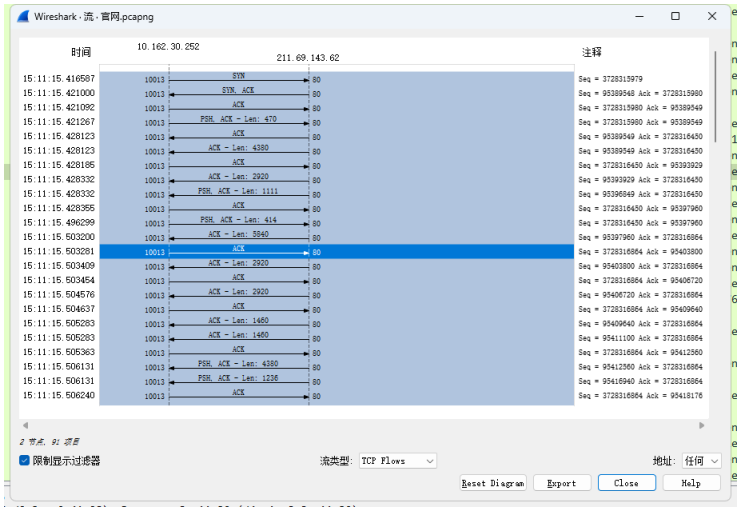


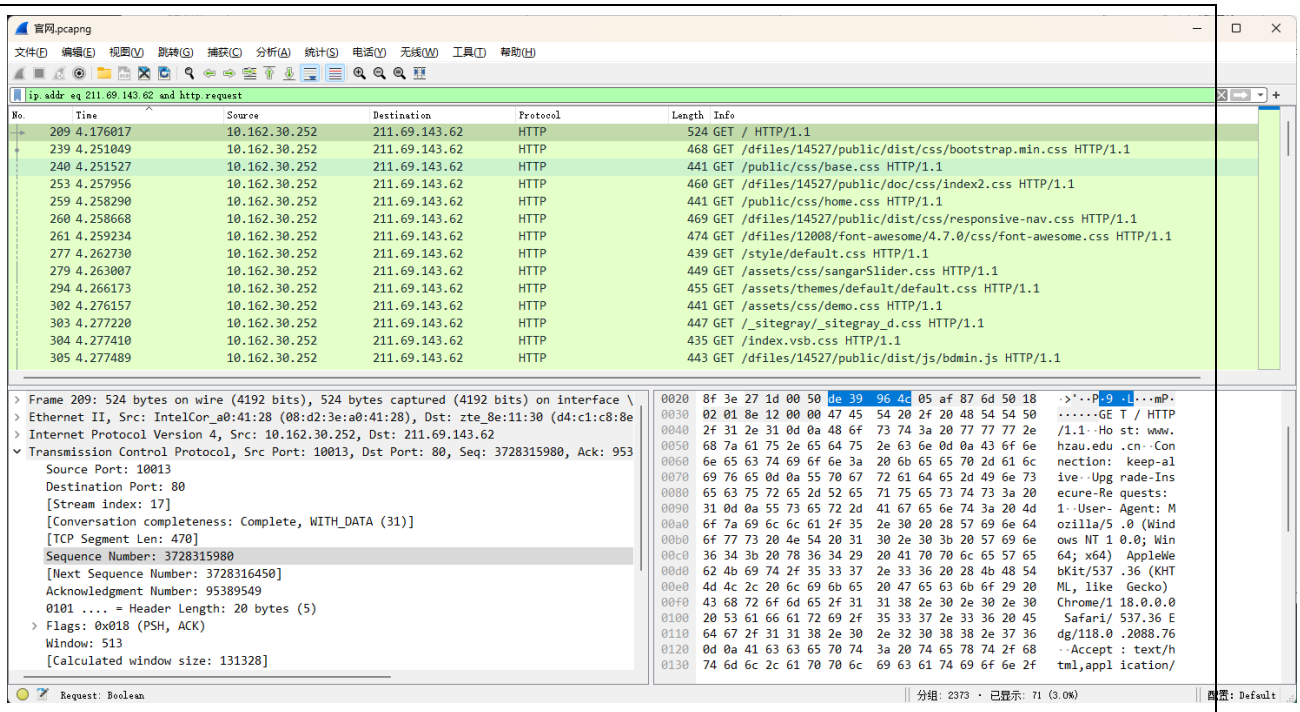
15. 找到全部 HTTP 的请求消息并截图。



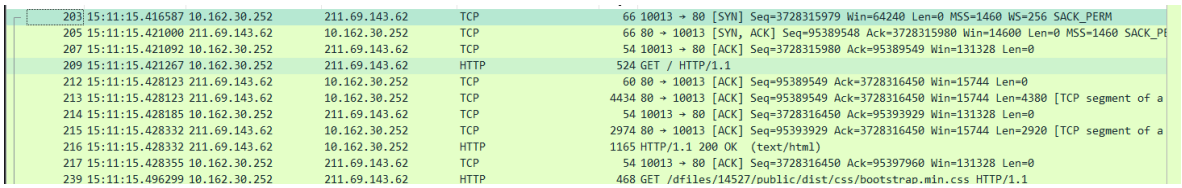
16. 找到全部源 IP 地址为指定 web 服务器地址的 HTTP 响应消息并截图

筛选条件：ip.addr eq 211.69.143.62 and http.request





17. 查看你访问指定 Web 服务器 HTTP 会话的工作过程。将结果截图，并 对前 10 个包进行详细分析。（参见附录 2） 本步骤可以使用 Statistics ->Flow Graph...->TCP flow ->OK 命令



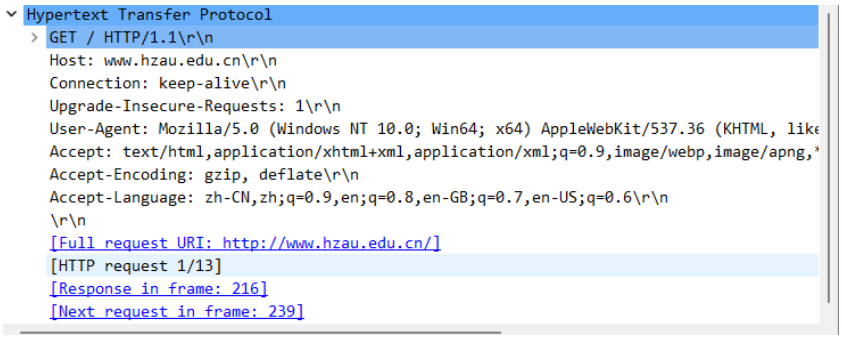
（1）首先本机向 webserver 的 80 端口发送 TCP 握手请求并且用 SYN=0 来表示该标志仅在三次握手建立 TCP 连接时有效。它提示 TCP 连接的服务端检查序列编号，该序列编号为 TCP 连接初始端(一般是客户端)的初始序列编号。这是一次握手

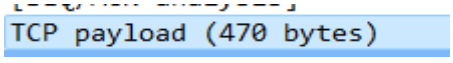
（2）然后 webserver 接收到请求后，返回响应并且标志 SYN=1，ACK=1，提醒客户端检查序列编号这是第二次握手

（3）然后客户端接收到 webserver 的第二次握手后的，返回 ACK=1 的数据报，并准备向客户端发送请求数据的数据报

前三次的 tcp 数据报中都是不包含数据的

（4）然后客户端发送 HTTP 请求



包含请求头和请求行一共 470bytes 

（5）之后的 webServer 返回接收到请求的确认 ACK=1(注意本次的 tcp 的数据报仍不包含数据)

（6）之后 webServer 开始返回数据，这里它将原本的数据分了两次 tcp 数据报来发送，第六个包是

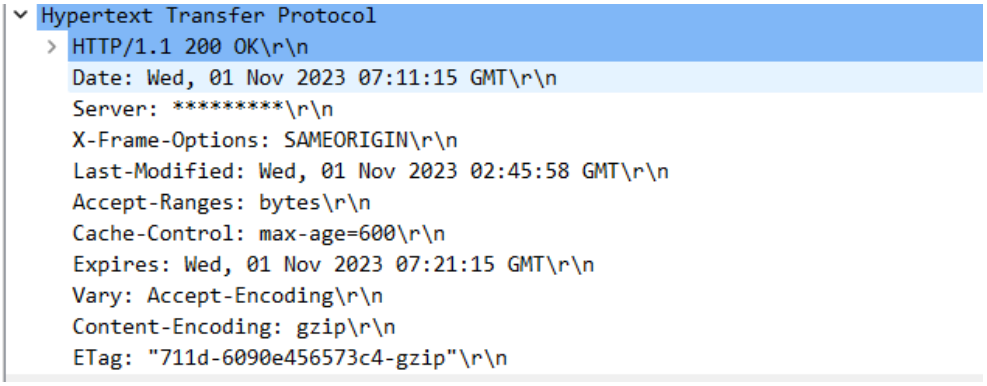
第一个包

（7）然后承载数据的第一个包到底本地，本地发送接收确认的数据包 ACK=1，

（8）对应服务器收到的上一个确认的包，发送了下一个承载数据的数据报（注意这里是流水线实现的也就是说并不是等收到 ACK 服务器才发送这个数据报）

（9）之后本地收到对应的数据报并且返回 ACK=1 的确认的数据报

（10） 最后加上这次收到的文件本地收到了这个文件的完整数据，返回为 HTTP 的响应报文



包含请求头、请求行和数据（这里的数据是一个 html 文件）

（11） 然后客户端对于上次收到的数据返回了 ACK 的数据报，告诉服务器已经正确收到了上一个发送的包了

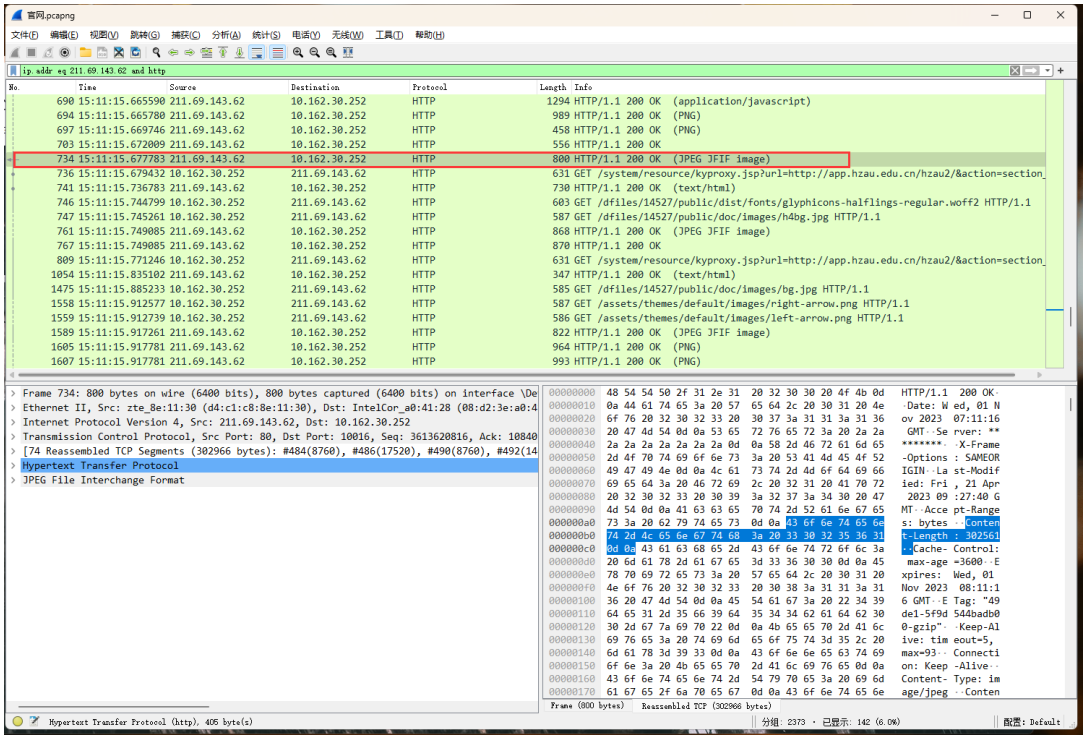
至此一个文件就完整的准确无误的传输过来了。

18. 使用 Follow TCP Stream 功能，将你看到的图片从你收到的 HTTP 响应 消息数据包中恢复出来

（1） 打开 WireShark，开始抓包

（2） 打开 http://www.hzau.edu.cn/网站，此时浏览器会通过 HTTP 协议中的 GET 方法获取到当前网页部署到服 务器上的所有图片。

（2）Wireshark 中选择一个 HTTP 接收文件的报文

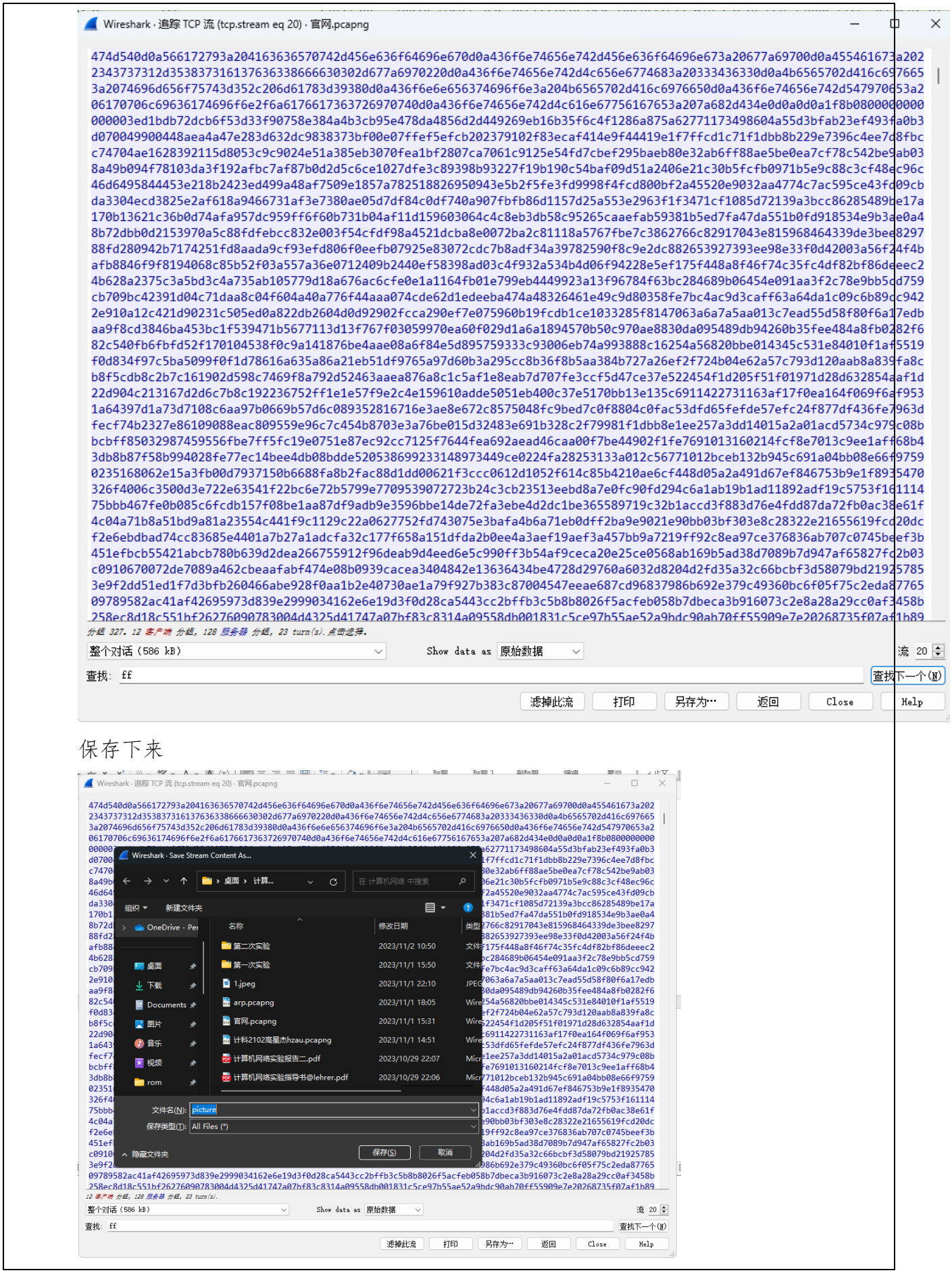


然后我们可以获得这个文件的数据

```
JPEG File Interchange Format
    Marker: Start of Image (0xffd8)
  > Marker segment: Reserved for application segments - 0 (0xFFE0)
  > Marker segment: Reserved for application segments - 1 (0xFFE1)
  > Marker segment: Define quantization table(s) (0xFFDB)
  > Marker segment: Define quantization table(s) (0xFFDB)
  > Start of Frame header: Start of Frame (non-differential, Huffman coding) - Baseline DCT
  > Marker segment: Define Huffman table(s) (0xFFC4)
  > Marker segment: Define Huffman table(s) (0xFFC4)
  > Marker segment: Define Huffman table(s) (0xFFC4)
  > Marker segment: Define Huffman table(s) (0xFFC4)
  > Start of Segment header: Start of Scan (0xFFDA)
    Entropy-coded segment (dissection is not yet implemented): f9e22b0be8ef7d99a3ccf31bafd4
    Marker: End of Image (0xffd9)
```

然后使用 Follow stream

找到该报文的数据

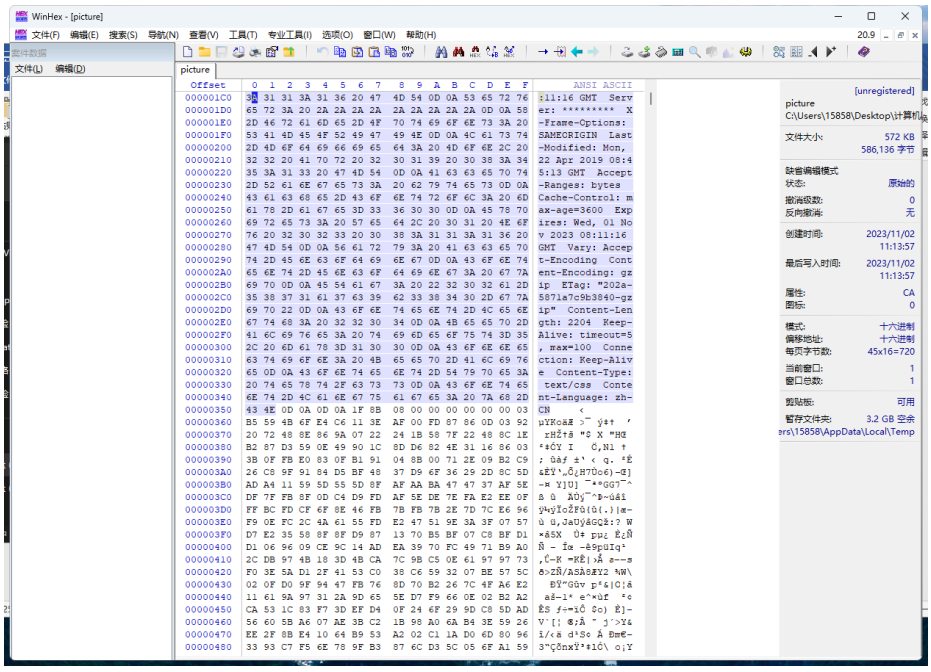

Wireshark · 追踪 TCP 流 (tcp.stream eq 20) · 官网.pcapng
整个对话（586 kB）
Show data as 原始数据
流 20
查找: ff
查找下一个(N)
滤掉此流
打印
另存为…
返回
Close
Help

保存下来

Wireshark · Save Stream Content As...
组织
新建文件夹
OneDrive - Pe
桌面
下载
Documents
图片
音乐
视频
rom
名称
修改日期
第二次实验 2023/11/2 10:50
第一次实验 2023/11/1 15:50
1.jpeg 2023/11/1 22:10
arp.pcapng 2023/11/1 18:05
官网.pcapng 2023/11/1 15:31
计科2102高星杰hzau.pcapng 2023/11/1 14:51
计算机网络实验报告二.pdf 2023/10/29 22:07
计算机网络实验指导书@lehrer.pdf 2023/10/29 22:06
文件名(N): picture
保存类型(T): All Files (*)
隐藏文件夹
保存(S)
取消

然后使用 winHex 编辑十六进制

把数据之前的响应头和响应行都删除然后保存



然后修改后缀就会发现变成了正常的图片了

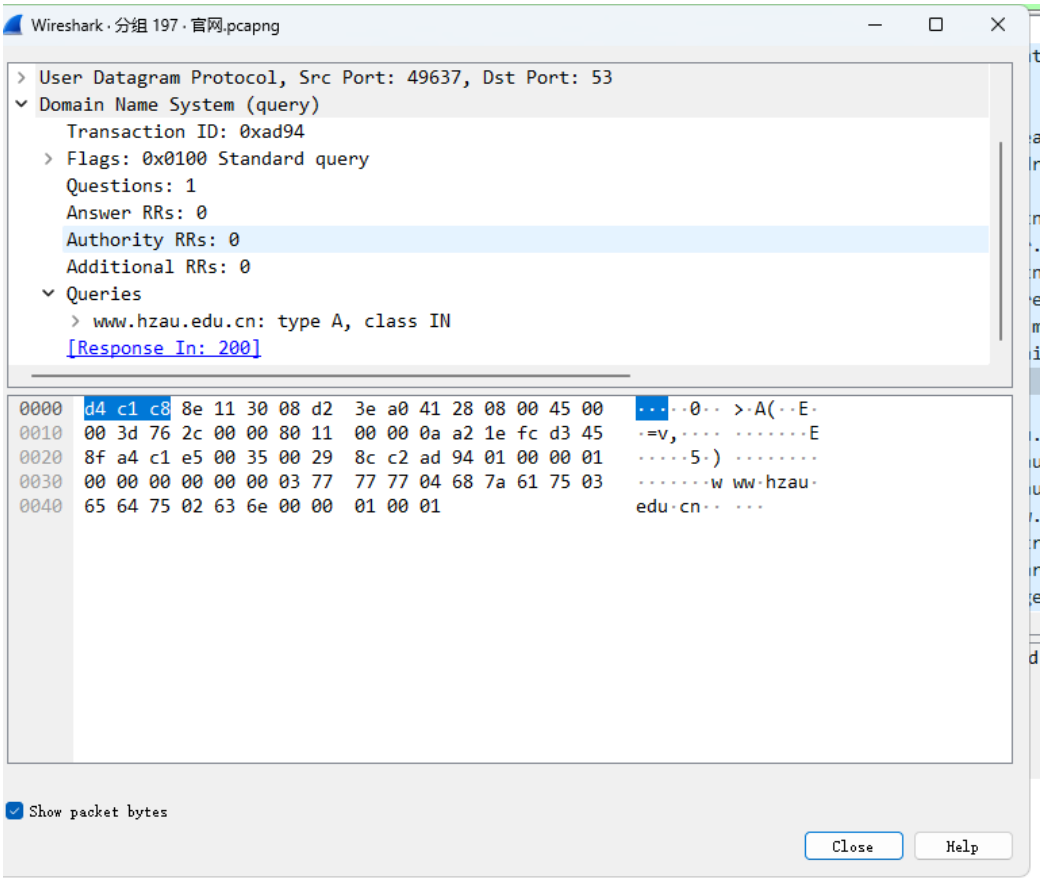
19. 回答下列问题。

（1）简述访问 web 页面的过程。

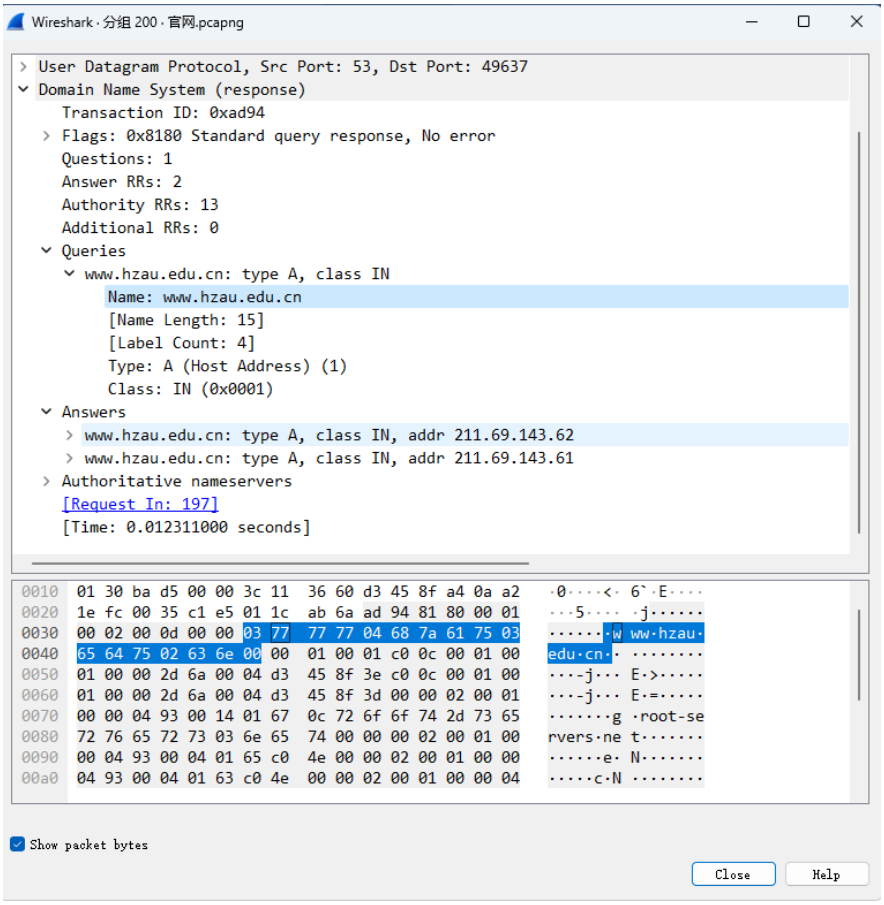
答：用户在浏览器地址栏键入 URL，浏览器作为客户端向 DNS 服务器发送域名解析请求，DNS 服务器返回 IP 地址；浏览器通过该 IP 地址，与 WEB 服务器建立 TCP 连接（三次握手）；建立连接后，浏览器发送 HTTP 请求，服务器根据请求种类等数据，向浏览器回送响应数据包；最后浏览器根据接收到的响应数据，将数据渲染在 WEB 页面上，显示请求的所有资源。

（2）找出 DNS 解析请求、应答相关分组，传输层使用了何种协议，端口 号是多少？所请求域名的 IP 地址是什么？

DNS 的请求分组：

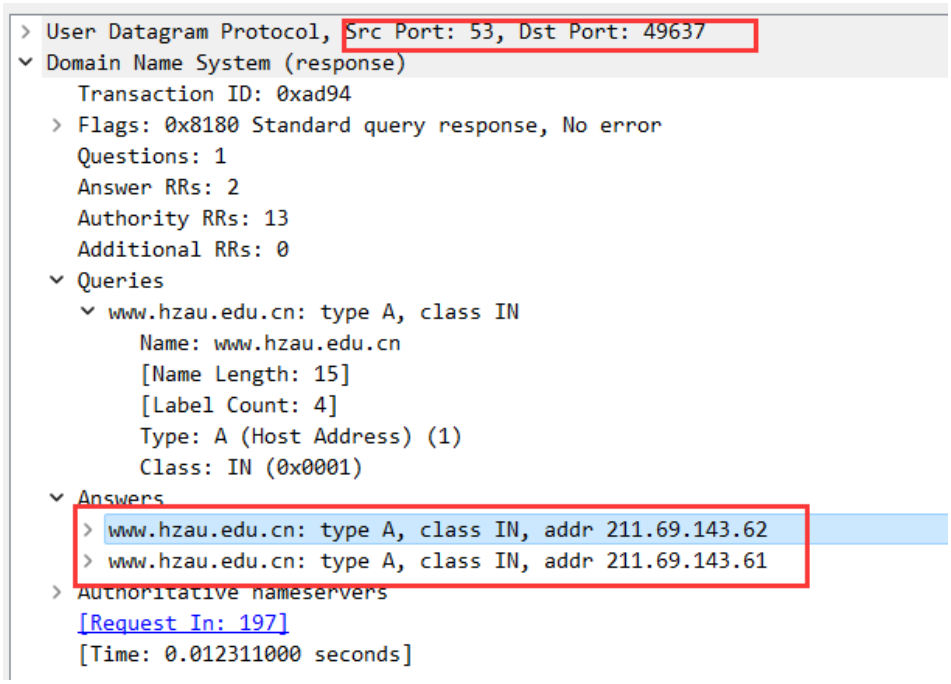

Wireshark · 分组 197 · 官网.pcapng
User Datagram Protocol, Src Port: 49637, Dst Port: 53
Domain Name System (query)
Transaction ID: 0xad94
Flags: 0x0100 Standard query
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
Queries
www.hzau.edu.cn: type A, class IN
[Response In: 200]
0000 d4 c1 c8 8e 11 30 08 d2 3e a0 41 28 08 00 45 00
0010 00 3d 76 2c 00 00 80 11 00 00 0a a2 1e fc d3 45
0020 8f a4 c1 e5 00 35 00 29 8c c2 ad 94 01 00 00 01
0030 00 00 00 00 00 00 03 77 77 77 04 68 7a 61 75 03
0040 65 64 75 02 63 6e 00 00 01 00 01
Show packet bytes
Close
Help

DNS 应答分组


Wireshark · 分组 200 · 官网.pcapng
User Datagram Protocol, Src Port: 53, Dst Port: 49637
Domain Name System (response)
Transaction ID: 0xad94
Flags: 0x8180 Standard query response, No error
Questions: 1
Answer RRs: 2
Authority RRs: 13
Additional RRs: 0
Queries
www.hzau.edu.cn: type A, class IN
Name: www.hzau.edu.cn
[Name Length: 15]
[Label Count: 4]
Type: A (Host Address) (1)
Class: IN (0x0001)
Answers
www.hzau.edu.cn: type A, class IN, addr 211.69.143.62
www.hzau.edu.cn: type A, class IN, addr 211.69.143.61
Authoritative nameservers
[Request In: 197]
[Time: 0.012311000 seconds]
0010 01 30 ba d5 00 00 3c 11 36 60 d3 45 8f a4 0a a2
0020 1e fc 00 35 c1 e5 01 1c ab 6a ad 94 81 80 00 01
0030 00 02 00 0d 00 00 03 77 77 77 04 68 7a 61 75 03
0040 65 64 75 02 63 6e 00 00 01 00 01 c0 0c 00 01 00
0050 01 00 00 2d 6a 00 04 d3 45 8f 3e c0 0c 00 01 00
0060 01 00 00 2d 6a 00 04 d3 45 8f 3d 00 00 02 00 01
0070 00 00 04 93 00 14 01 67 0c 72 6f 6f 74 2d 73 65
0080 72 76 65 72 73 03 6e 65 74 00 00 00 02 00 01 00
0090 00 04 93 00 04 01 65 c0 4e 00 00 02 00 01 00 00
00a0 04 93 00 04 01 63 c0 4e 00 00 02 00 01 00 00 04
Show packet bytes
Close
Help

传输层使用了 udp 协议目标端口是 53 源端口是 49637

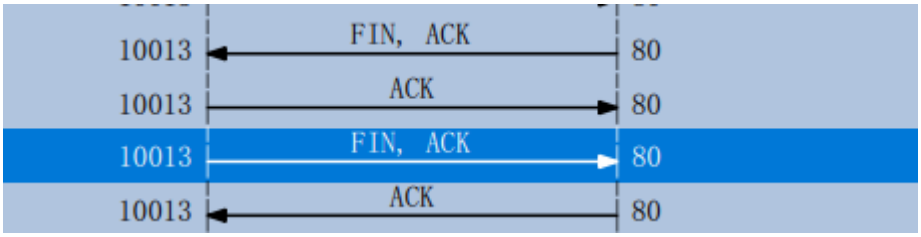
所请求域名的 ip 为 211.69.143.62



（3）针对 TCP 连接，该 TCP 连接的四元组是什么？双方协商的起始序号 是什么？TCP 连接建立的过程中，第三次握手是否带有数据？是否消耗了一个序 号？

答：四元组为源 IP 地址、源端口号、目标 IP 地址、目标端口号；起始序号为 0；不带数据；消耗一个序号。

（4）找到 TCP 连接的释放过程，绘出 TCP 连接释放的完整过程，注明每严禁复制 15 个 TCP 报文段的序号、确认号、以及 FIN\ACK 的设置。



这四个包的序列号和确认号依次是

Seq = 95485834 Ack=3728321591

Seq = 3728321591 Ack=95485835

Seq = 3728321591 Ack=95485835

Seq = 95485835 Ack=372832159

（5）针对 TCP 连接释放，请问释放请求由服务器还是客户发起？FIN 报文段是否携带数据，是否消耗一个序号？FIN 报文段的序号是什么？为什么是这个值？

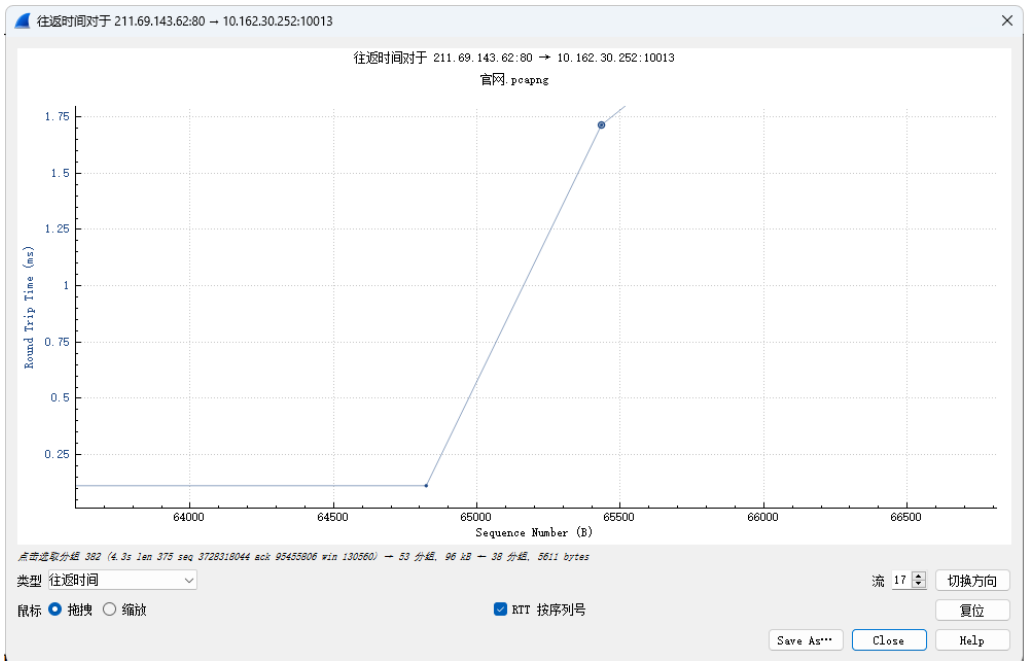
答：收发任意一方；FIN 报文段不携带数据并消耗一个序列号；序号为上一个请求的序列号加上其数据报长度，这样设置可以通过序列号与数据长度直接的关系快速解析分组数据。存储时可以直接利用 Seq 与 Len 得到下一个包的序列号

（6）在该 TCP 连接的数据传输过程中，找出每一个 ACK 报文段与相应数 据报文段的对应关系，计算这些数据报文段的往返时延 RTT（即 RTT 样本值）。

答：ACK 报文段的 Seq 值为相应数据报文段的 Ack 值，Ack 的值为对应数据报文 Seq+1

RTT 可以通过计算当前时间与 TCP 报文中 TimeStamp 的差值计算.

```
[Time since first frame in this TCP stream: 0.004505000 seconds]
[Time since previous frame in this TCP stream: 0.000092000 seconds]
```



（7）请描述 HTTP 协议的持续连接的两种工作方式。访问这些页面（同一网站的不同页面）的过程中，采用了哪种方式？

答：流水线方式：与流水线一样，必须得到上一个请求的响应之后，才能发出下一个请求。每一次访问一个对象都需要一个 RTT 时间。

非流水线方式：无需等待，收到响应之前都可以发出新的请求。这样一定程度上可以减少 RTT 时间的浪费。

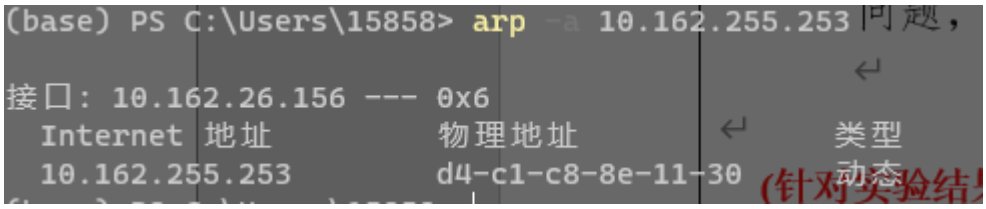
访问同一网站的过程中采用了流水线方式。浏览器会在与服务器建立的 TCP 连接上发送多个 HTTP 请求，服务器会在同一连接上返回多个 HTTP 响应，从而减少了连接建立和关闭的开销，提高了页面加载的效率。

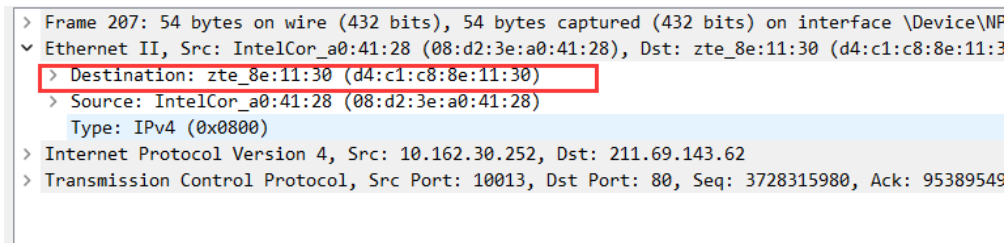
6. 结论与分析：

遇到的一些问题：

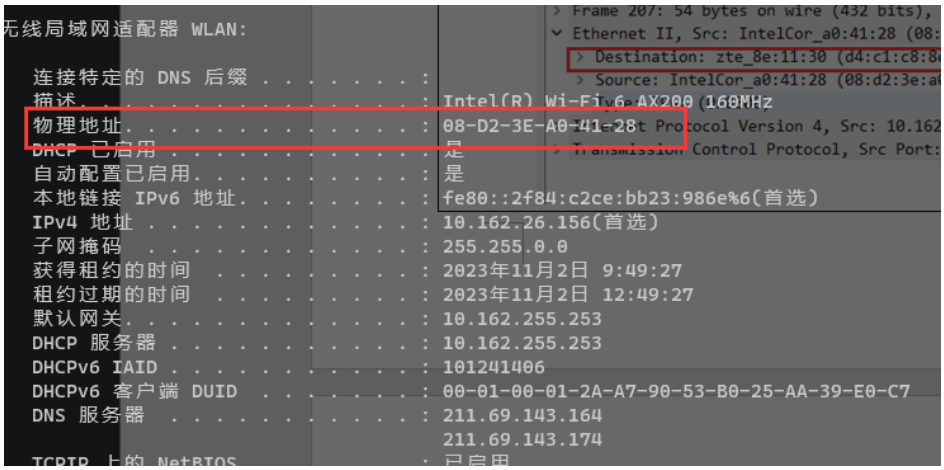
(1) WebServer 的 Mac 地址问题：

当我尝试从数据链路层的数据包来寻找 WebServer 的 mac 地址时，发现了一个奇怪的问题，就是数据链路层的目标 MAC 地址是网关的 MAC 地址





源地址是本机的网卡地址这个没有问题



但是目的地址非常奇怪的与网关地址相同

于是上网搜查资料

原来，链路层的目的地址不是最终的目的地址，**而是下一跳网络设备（一般是交换机或路由器）的 MAC 地址**，这里就是网关的 mac 地址，也就说明我们是无法直接获取到 webserver 的 Mac 地址，只有离服务器最近的路由器才能看到。

(2) ARP 协议问题：

问题：一直抓不到 arp 协议的包

解决办法：原来是因为本地已经有了缓存所有先使用 arp -ad 删除本地缓存

然后再使用抓包工具抓就可以抓到了

(3) 图片恢复的问题

问题：最一开始一直尝试恢复图片，图片一直恢复不出来，

解决办法：更改恢复图片的方式，原理之前一直用文本文档复制图片的十六进制然后保存改后缀名 jpeg 一直行不通，后来转念一想，这里的保存方式不是把 16 进制转换成了字符串了吗？而不是真正保存了 16 进制，所以改为了用 WinHex 这个工具来编辑 16 进制，这样保存后，然后就可以打开了。

（4）心得与体会

通过本次实验将课上的理论应用到了实际，书本上的知识不在是死知识，而是能够通过实践证明的的知识。看着这一个个协议看似紊乱实则高度有序的工作，我不禁感受到无数前人的伟大，完整的大厦不是一蹴而就的，也不是一人之作，是千千万万的学者和工程师的汗水，才让我们今天能够直接尝到果实。但是我们作为新一代计算机科学的学者，在享受这些成果的同时也要的发展它，而发展它就要求我们首先要继承之前学者的智慧，站在巨人的肩膀上，才能远眺和发展。

【过程记录（源程序、测试用例、测试结果等）】