# **实验3报告模板**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 姓名： |  | 学号： |  | 班级： |  |
|  |  |  |  |  |  |

# **3 实验三 应用层协议分析**

## 3.1 实验目的

1. 分析应用层协议（如HTTP、HTTPS）的工作过程，理解应用层与传输层及下层协议的关系，分析缓存工作原理及好处。

2. 能够依据HTTP协议编写简单的HTTP客户端程序，从几个网站去读取主页，或者取文件。

## 3.2 实验内容

（1）利用现有校园网络及外网服务器搭建内网、外网环境；

（2）用Wireshark截获HTTP/HTTPS报文，分析报文结构及浏览器和服务器的交互过程；分析HTTP协议的缓存机制，分析HTTPS的加密传输机制。分析应用层协议跟TCP/DNS等协议的交互关系；

（3）依据HTTP协议，编写简单的HTTP客户端程序，从几个网站去读取主页，或者取文件。

## 3.3 实验步骤

### 3.3.1 实验环境

依托校园网和外网服务器搭建内网、外网环境，每名同学单独完成实验并撰写实验报告。

### 3.3.2 HTTP协议分析

清空缓存后的DNS和HTTP协议分析：

步骤1：在计算机终端上运行Wireshark开始截获报文。

步骤2：清空DNS和HTTP浏览器的缓存：

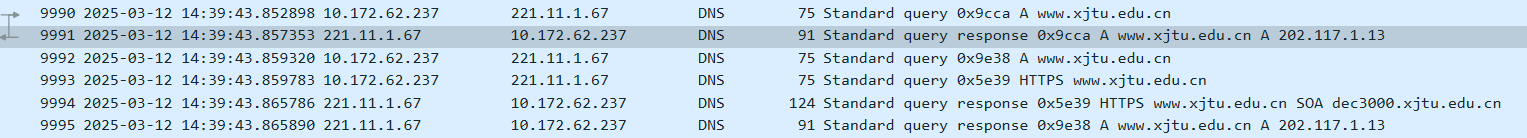
浏览器缓存的清除以Chrome浏览器为例，地址栏中输入chrome://settings/，找到高级选项中的“隐私设置和安全性”，清除浏览数据。

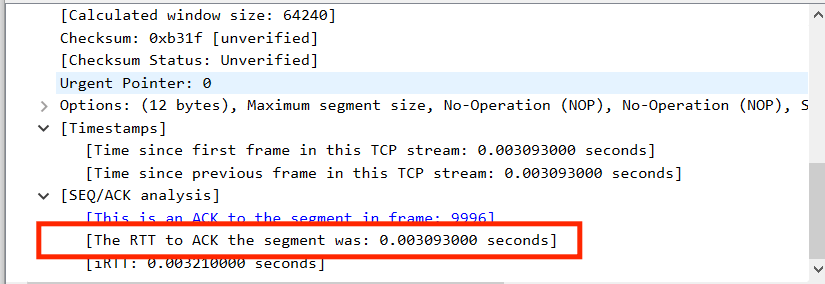
在cmd命令行执行“ipconfig /flushdns”清除本地DNS缓存（或者在PowerShell用Clear-DnsClientCache/Get-DnsClientCache命令清除/查看DNS缓存）。

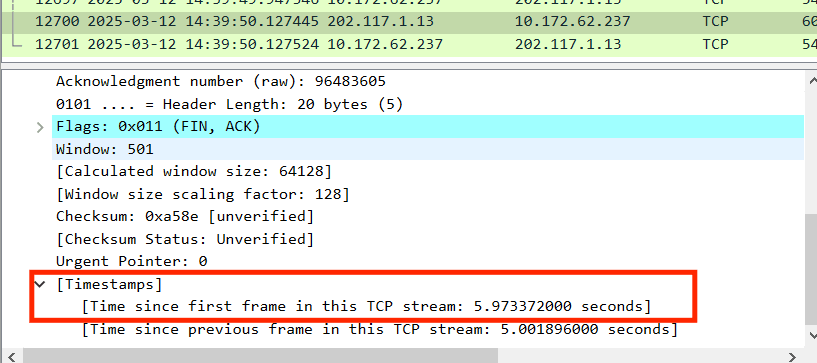
步骤3：在浏览器中访问3个网址，分别是国内、国外和南半球网址。

国内网址：如http://www.xjtu.edu.cn（或http://www.people.com.cn/）。

**注意：必须指定http，目前很多网站默认是https，无法抓到http数据包；**

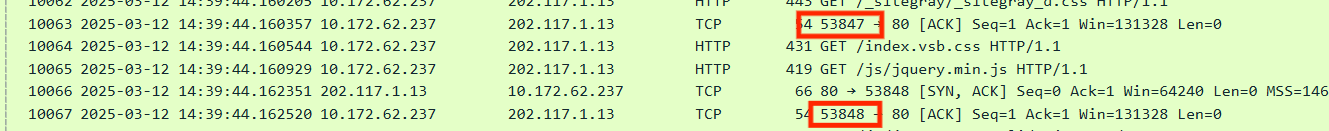
****

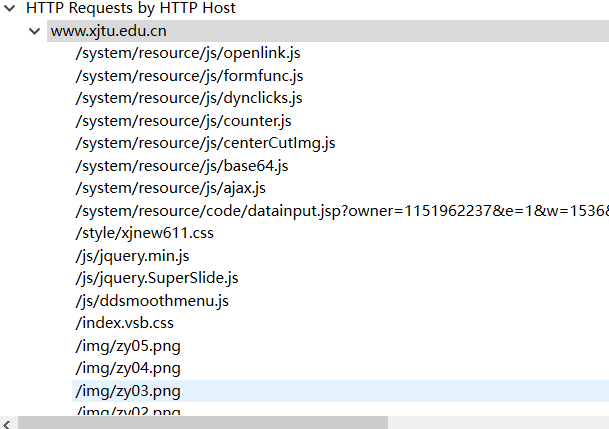
****



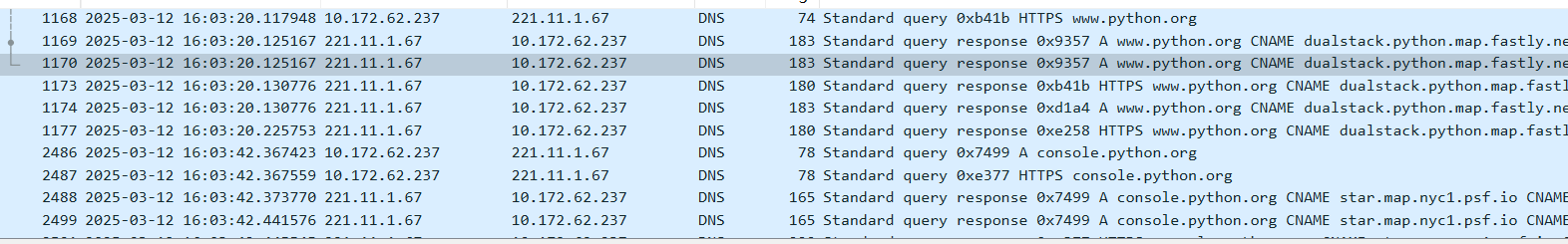
****



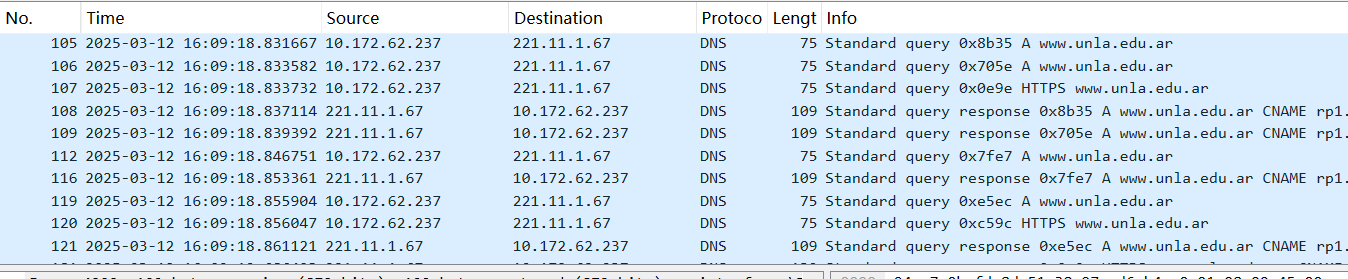


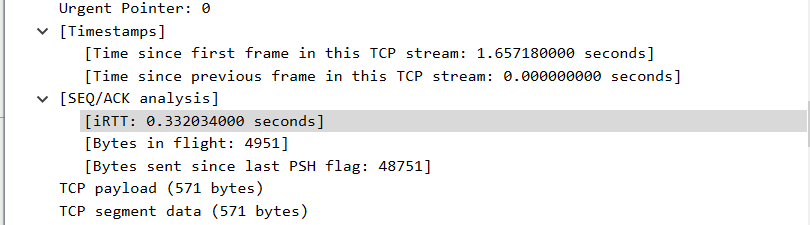
****

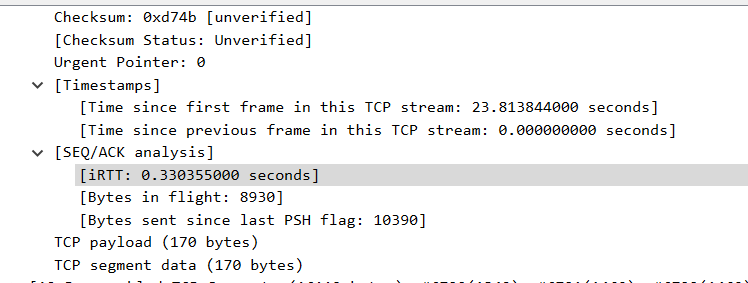
国外网址：如www.github.com, [www.python.org](http://www.python.org)；



南半球网址：如阿根廷的www.unla.edu.ar（或www.uns.edu.ar，en.unne.edu.ar，www.unne.edu.ar，www.unsj.edu.ar等）；或巴西的www.unb.br；或南非的www.unisa.ac.za（或[www.up.ac.za](http://www.up.ac.za)）。







步骤4：网站加载完成之后，停止报文截获并保存数据，分析截获的报文，填写表5-1中“无缓存”列信息。

表5-1 不同网站访问信息统计表

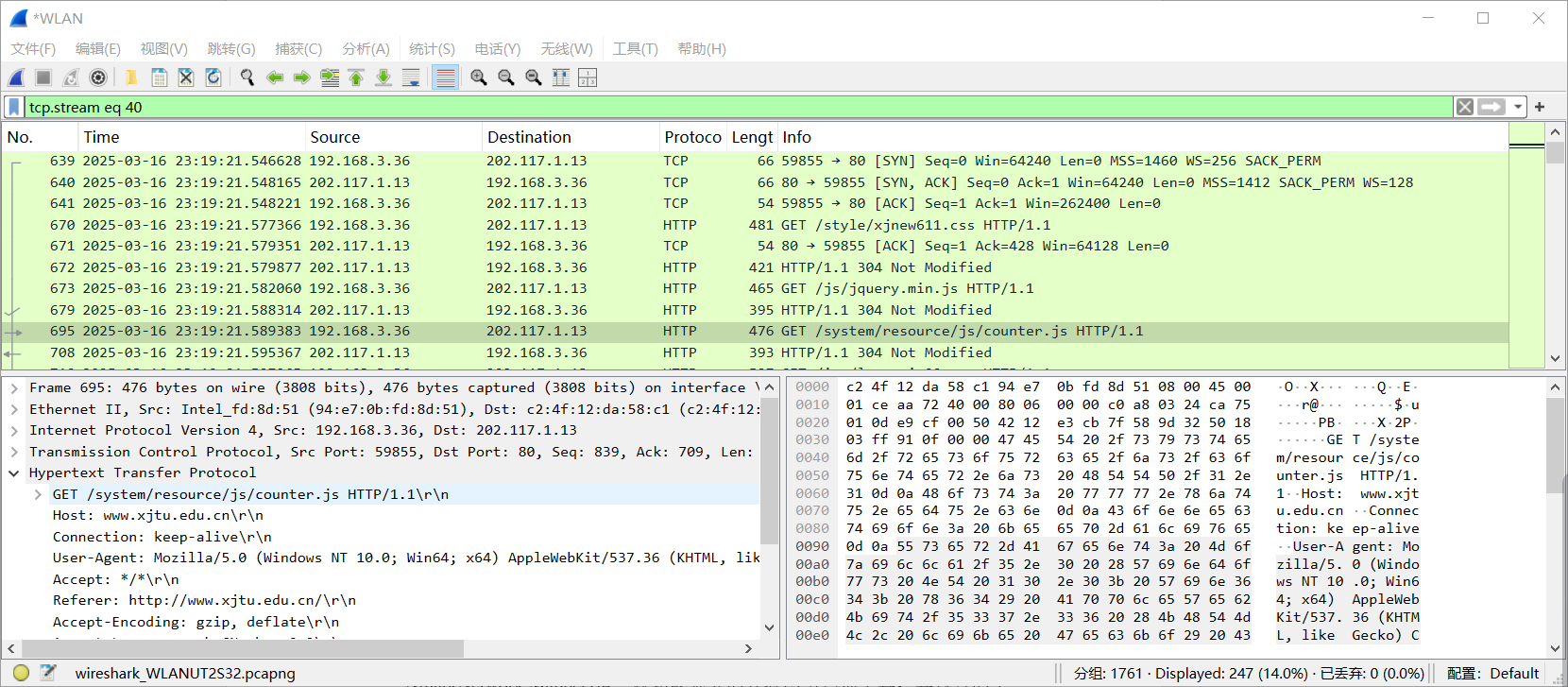
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 访问网站 | 国内 | | 国外 | | 南半球 | |
| 域名： | *例：http://www.xjtu.edu.cn* | | www.originlab.com/ | | www.unla.edu.ar | |
| IP地址： | *202.117.1.13* | | 208.117.247.127 | | 170.78.184.20 | |
| 是否带缓存 | 无缓存 | 带缓存 | 无缓存 | 带缓存 | 无缓存 | 带缓存 |
| DNS次数 | 8 | 0 | 12 | 8 | 18 | 8 |
| 应答时间(s) | 0.0022 | 0.0020 | 0.2535 | 0.2444 | 0.3863 | 0.3741 |
| 加载时间(s) | 0.53 | 0.07 | 17.32 | 5.04 | 234.79 | 136.72 |
| 建立连接数 | 7 | 2 | 3 | 3 | 11 | 3 |
| 请求对象数 | 76 | 1 | 3 | 1 | 133 | 0 |

带缓存的DNS和HTTP协议分析：

按上一节中的步骤1-4再次执行一遍，但不执行步骤2，填写表5-1中“带缓存”列信息。

**HTTP分析：**

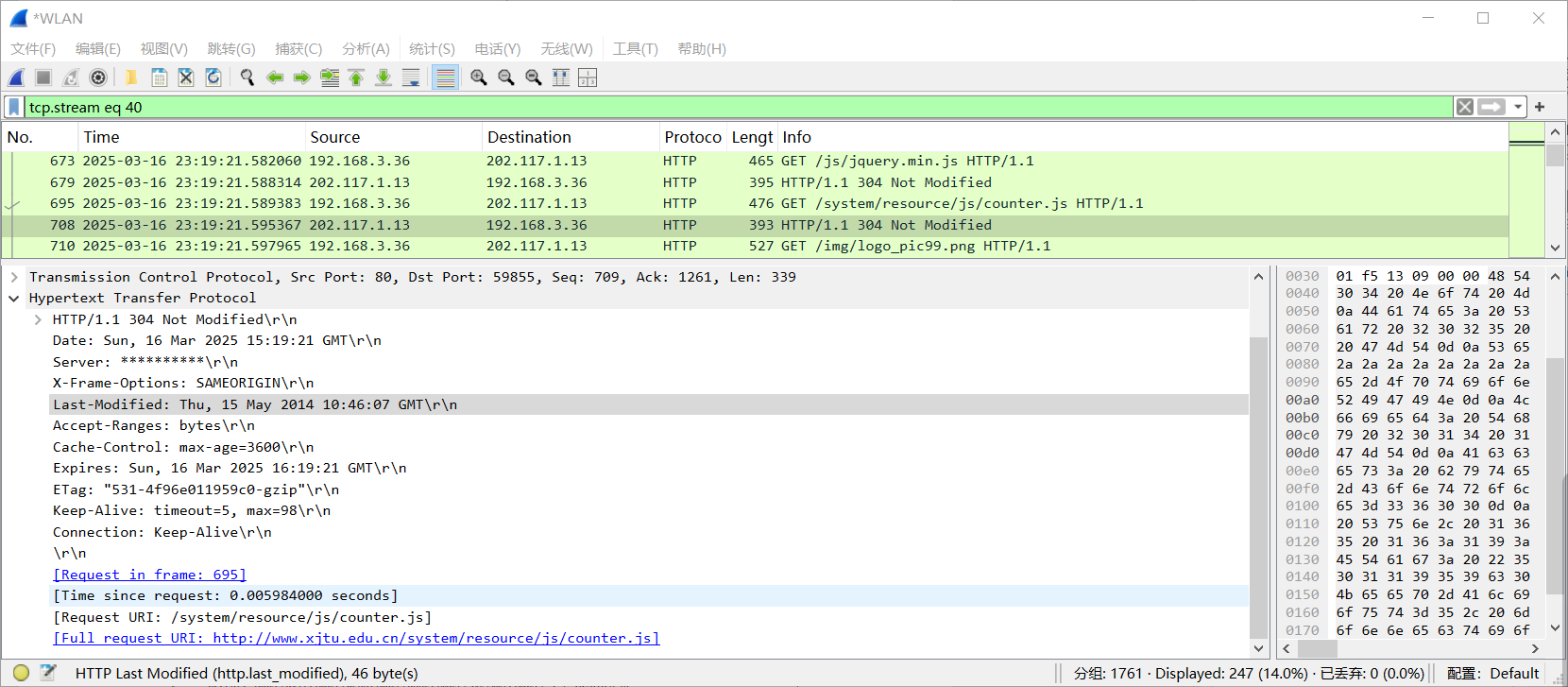
选中一条HTTP报文，点击右键，选择“追踪流->HTTP Stream”。关闭统计窗口，在主界面观察数据流，结合数据包分析HTTP的请求和应答，以及与TCP协议的配合使用。



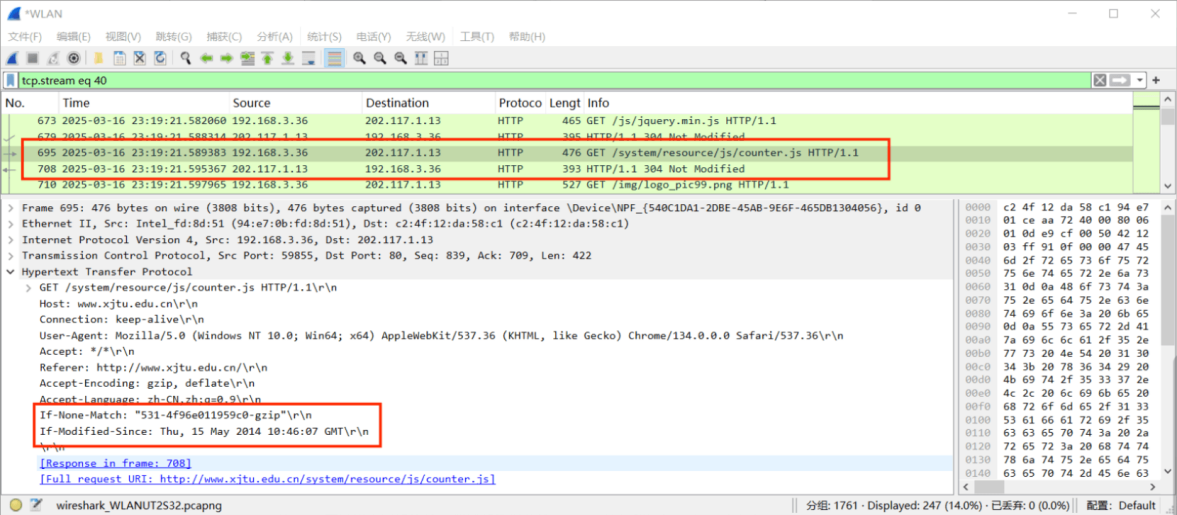
追踪数据流以后，可以看到过滤出的报文是HTTP请求应答，以及相关的TCP请求和应答。HTTP客户端发起一个请求，创建一个到服务器指定端口（默认80）的TCP连接。HTTP服务器则在该端口监听客户端的请求。一旦收到请求，服务器会向客户端返回一个状态，比如"HTTP/1.1 200 OK"或是图示的"HTTP/1.1 304 Not Modified"及返回的内容，如请求的文件、错误消息、或者其它信息。HTTP依赖CP来确保数据可靠地从客户端传输到服务器端或者反向传输。当进行HTTP交互时，HTTP请求或者应答的数据被分割成TCP报文段，然后通过TCP协议建立的连接进行传输。TCP负责处理数据的分段、排序、重传等工作，以确保数据能够完整、按序地到达目的地，HTTP无需关心底层传输的这些细节，只专注于应用层数据的处理，比如请求和应答的结构和语义等。

找到HTTP应答包中状态码为304的数据包和对应的请求包，分析缓存工作原理和带来的好处。

状态码304的http应答报文：



与之对应的请求报文：

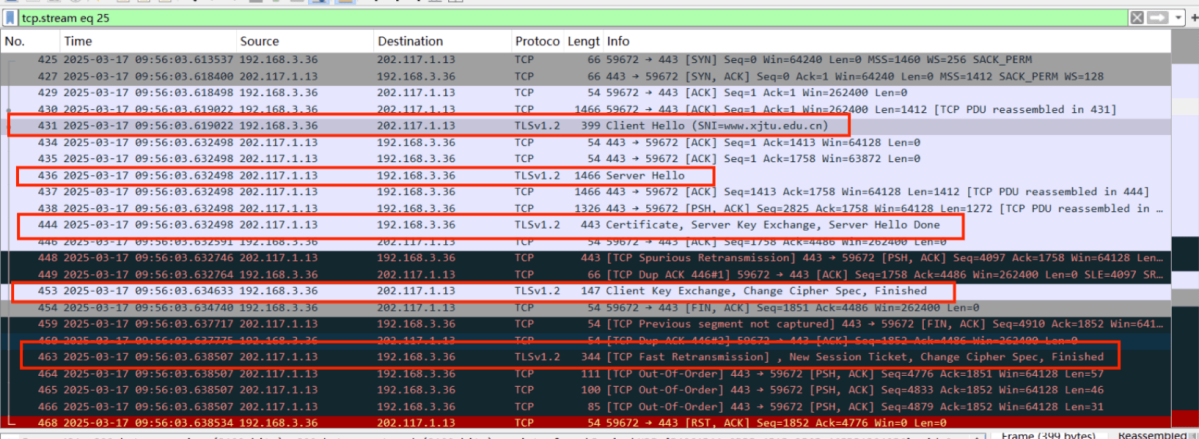


当浏览器首次请求一个资源时（例如一个网页、图片等），服务器会返回该资源以及包含有关资源的元信息（如最后修改时间、ETag等）的HTTP应答头。浏览器会缓存这个资源和对应的元信息。当浏览器再次请求同一资源时，它会在请求头中包含上次获取到的元信息（如图中截获的请求报文中If - Modified – Since字段和If - None -Match字段，分别对应服务器返回的最后修改时间和ETag）。服务器收到这个请求后，会根据这些信息来检查资源是否有修改。如果资源没有修改，服务器就返回304状态码，表示浏览器可以继续使用缓存中的资源，而无需再次下载整个资源。

HTTP缓存机制的好处是，可以减少带宽消耗、降低服务器负担，并且对客户端而言加载网页的速度更快，用户体验更好。

**HTTPS分析：**

访问HTTPS网站时，因数据是加密传输无法观察内容，可观察分析TLS加密传输的建立过程、传输端口等（选中一条TLS数据报文，点击右键，选择“追踪流->TLS Stream”）。



从截获的报文中大致可以看出TLS加密传输的建立过程大致如下：

客户端向服务器发送ClientHello消息，其中包含客户端支持的TLS协议版本、随机数（用于生成会话密钥）、支持的加密套件以及可选的扩展信息。

服务器收到ClientHello后，发送ServerHello消息，选择一个双方都支持的TLS版本、加密套件，并提供自己的随机数。如果需要服务器身份验证，服务器会发送其数字证书Certificate（包含公钥），证书由受信任的证书颁发机构（CA）签名。对于某些加密套件，还可能发送服务器密钥交换信息。最后发送ServerHelloDone消息，表示服务器的初始协商部分完成。

客户端密钥交换Server Key Exchange，客户端根据服务器的响应生成预主密钥（Pre - Master Secret）。如果服务器发送了证书，客户端会验证证书的有效性，包括检查CA签名、证书有效期、是否被吊销等。然后客户端使用服务器公钥（从服务器证书获取）对预主密钥进行加密（如果需要身份验证）并发送给服务器。

客户端发送Change Cipher Spec消息，表示之后的通信将使用协商好的加密套件和密钥进行加密。接着发送Finished消息，其中包含基于之前协商的密钥、随机数等计算出的校验和，用于验证密钥交换和协商过程是否正确。

服务器收到客户端的Change Cipher Spec和Finished消息后，发送自己的Change Cipher Spec消息，表示开始使用协商好的加密方式，然后发送Finished消息给客户端，同样包含校验和。双方通过验证对方的Finished消息来确认密钥交换和协商成功。

以上连接建立过程的关键消息和字段在数据包请求和应答中都有所体现。对于HTTPS加密而言，根据使用的TCP服务的数据流可以看出，服务器的常用端口为443.

### 3.3.3 Ncat访问HTTP

Nmap(Network Mapper)是一款功能强大的开源网络扫描工具，其包含的工具Ncat可以用于网络连接、调试和数据传输。Nmap软件解压后，在cmd命令行进入目录，就可以执行命令，也可以将软件目录添加至系统环境变量。

前面学习了HTTP协议，也用Wireshark分析了典型的HTTP请求和应答。现在使用ncat来**模拟**浏览器来跟服务器进行HTTP交互。在命令窗口执行“ncat -C 域名/ip地址 端口（80）”连接HTTP服务器（如www.xjtu.edu.cn），输入HTTP请求（可参考之前捕获的数据包中的请求），请求头至少有Host和User-Agent，其中**User-Agent后跟上自己的英文名字**。查看服务器的应答。

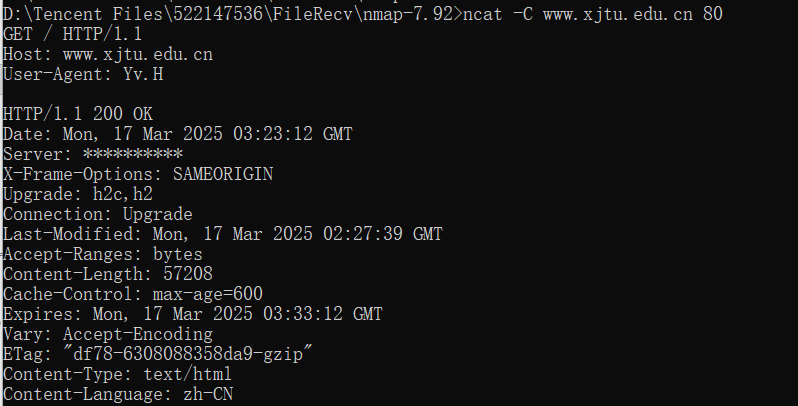
例：

ncat -C **xxx.xxx.xxx.xxx** 80 **（xxx.xxx.xxx.xxx是具体的域名或IP地址）**

GET / HTTP/1.1

Host: **xxx.xxx.xxx.xxx（xxx.xxx.xxx.xxx是具体的域名。）**

User-Agent: Zhang San （**跟上自己的英文名字。**然后敲两次回车，应答状态码200表示请求成功）



### 3.3.4 编写HTTP客户端程序

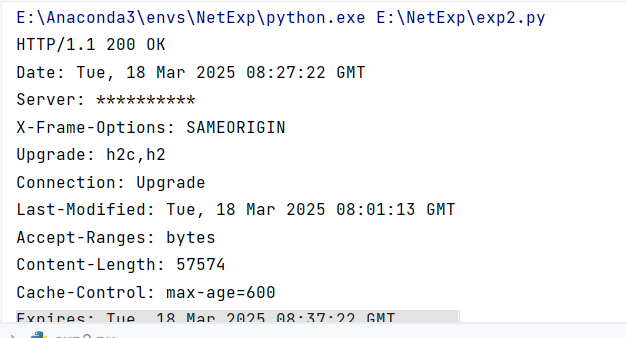
（1）编写一个简单的HTTP客户端程序。建议使用Python的socket模块，但不能使用Python的高级HTTP模块（如requests、urllib.request、http.client等）。

（2）运行TCP客户端程序进行连接测试（如连接到www.xjtu.edu.cn或www.hit.edu.cn服务器80端口，获取并打印主页）。

输入HTTP请求时，除了请求行以外，请求头至少有Host和User-Agent，其中**User-Agent后跟上自己的英文名字**。



在终端打印获得的网页信息，同时将网页源代码写入本地的page.html文件当中。返回的状态码为200 OK，连接正常。



网页源代码同时包含了Javascript脚本：



## 3.4 互动讨论主题

（1）HTTP协议的缓存，DNS的缓存；

HTTP协议的缓存有浏览器缓存和代理服务器缓存两种。浏览器会根据服务器返回的缓存指令来存储资源。如果请求资源未过期且缓存可用，浏览器直接使用缓存而无需再次向服务器请求。代理服务器位于客户端和源服务器之间。当代理服务器接收到请求时，如果它已经缓存了相应资源并且资源仍然有效，就直接向客户端返回缓存的资源。

DNS的缓存有本地缓存和DNS服务器缓存两种。客户端操作系统或浏览器会对DNS查询结果进行缓存。在缓存有效期内，再次查询相同域名时就直接从本地缓存获取结果，而无需向DNS服务器发送请求。各级DNS服务器也会进行缓存。递归DNS服务器在解析域名成功后会缓存结果，以便后续对相同域名的查询能够快速响应。

（2）缓存对网络访问速度的影响。

显然HTTP缓存和DNS缓存都加快了网络访问的速度。当HTTP缓存的资源仍然与服务器保持一致时，就减少了重复请求相同资源到服务器的次数，节省了网络带宽并加快了资源的加载速度，也减少了服务器的负担；DNS缓存同样使得域名到IP地址的解析速度大大加快，特别是对于经常访问的域名，同时减少了DNS解析服务器的负载。

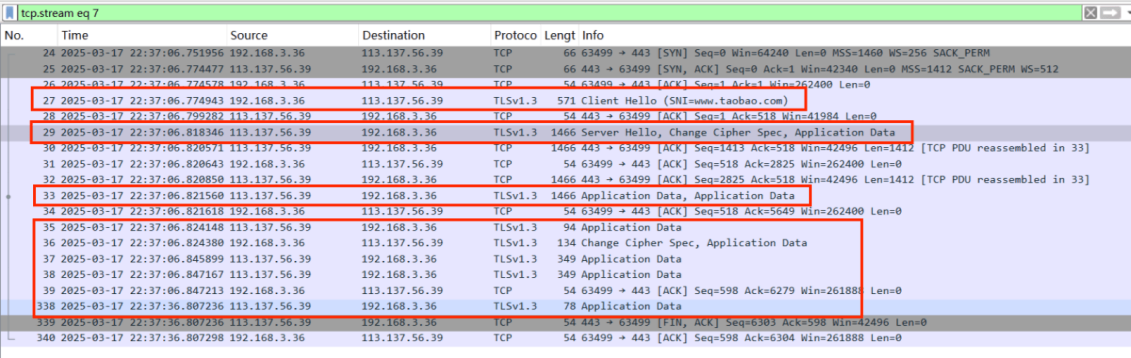
## 3.5 进阶自设计

用nmap的ncat来模拟https客户端，访问1-2个网站，抓包分析访问过程。

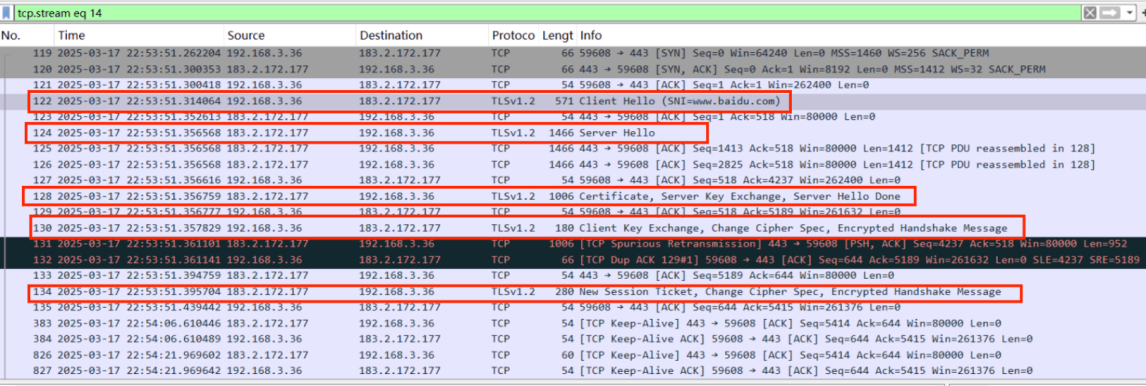
访问淘宝：



抓包：



访问百度：



截获的访问百度的TLS流中，可以清楚的看到TLS的Client Hello、Server Hello以及密钥交换和确认等握手信息。在截获的访问淘宝的TLS流中也能看到Client Hello和Server Hello信息等握手信息。访问的连接过程与上面HTTPS协议的过程相同。

除此以外，还注意到淘宝已经使用了更新的TLSv1.3协议，而百度仍然使用TLSv1.2协议。相比于TLSv1.2，TLSv1.3非常明显地减少了握手次数和开销，表现在我们截获的报文中，就是淘宝建立TLS传输的报文数量明显减少。并且，TLSv1.3引入了0-RTT模式，可以看到TLSv1.3在Server Hello以及密钥交换的报文中就已经附带了应用信息Application Data。总的来说，TLSv1.3协议更加安全，效率也更快。