计网第二次实验报告

学号：2011743　　姓名：高祎珂

实验概述

（1）搭建Web服务器，并制作简单的Web页面，包含简单文本信息（至少包含专业、学号、姓名）和自己的LOGO。

（2）通过浏览器获取自己编写的Web页面，使用Wireshark捕获浏览器与Web服务器的交互过程，并进行简单的分析说明。

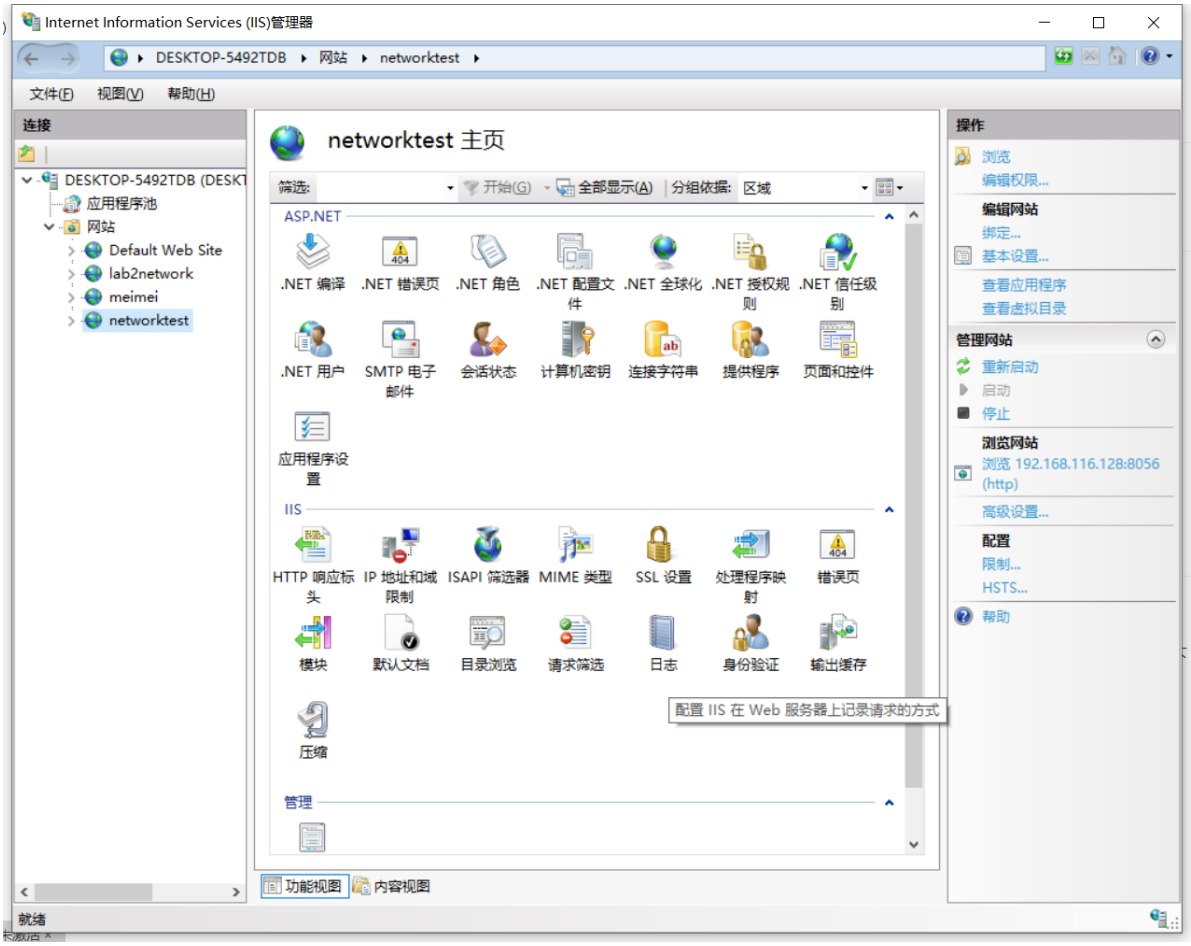
实验环境

- Windows
- IIS
- Wireshark

实验准备

- **web服务器搭建**

使用Windows下自带的IIS，配置一些文件，在IIS上新建服务器端，IP地址设置为192.168.116.128，端口号设置为8056，具体如下：



- **html编写**

我的html文件内容如下：

```
<!DOCTYPE html>
<html>
    <head>
        <title>我的网站</title>
        <link rel="shortcut icon" href="#"/>
    </head>
    <body>
        <h1>信息介绍</h1>
        我的logo：
        <img src="2.jpg" alt="My brother">
        <p>专业：计算机科学与技术</p>
        <p>学号：2011743</p>
        <p>姓名：高祎珂</p>

    </body>
</html>
```

信息介绍

我的logo：

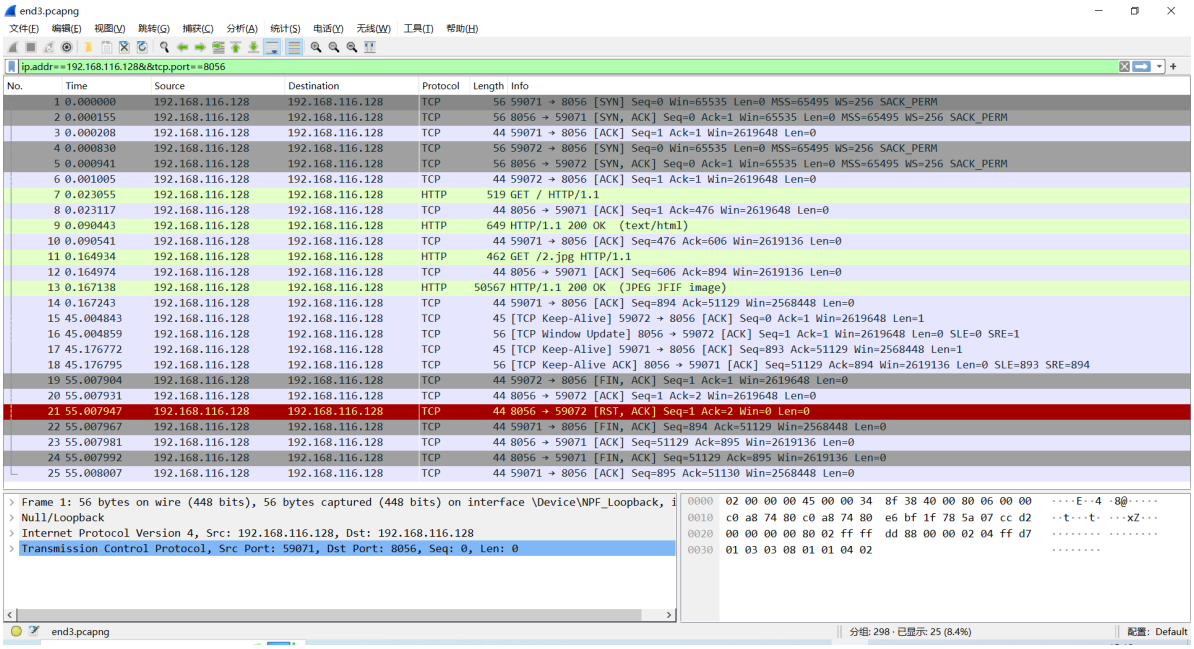
专业：计算机科学与技术

学号：2011743

姓名：高祎珂

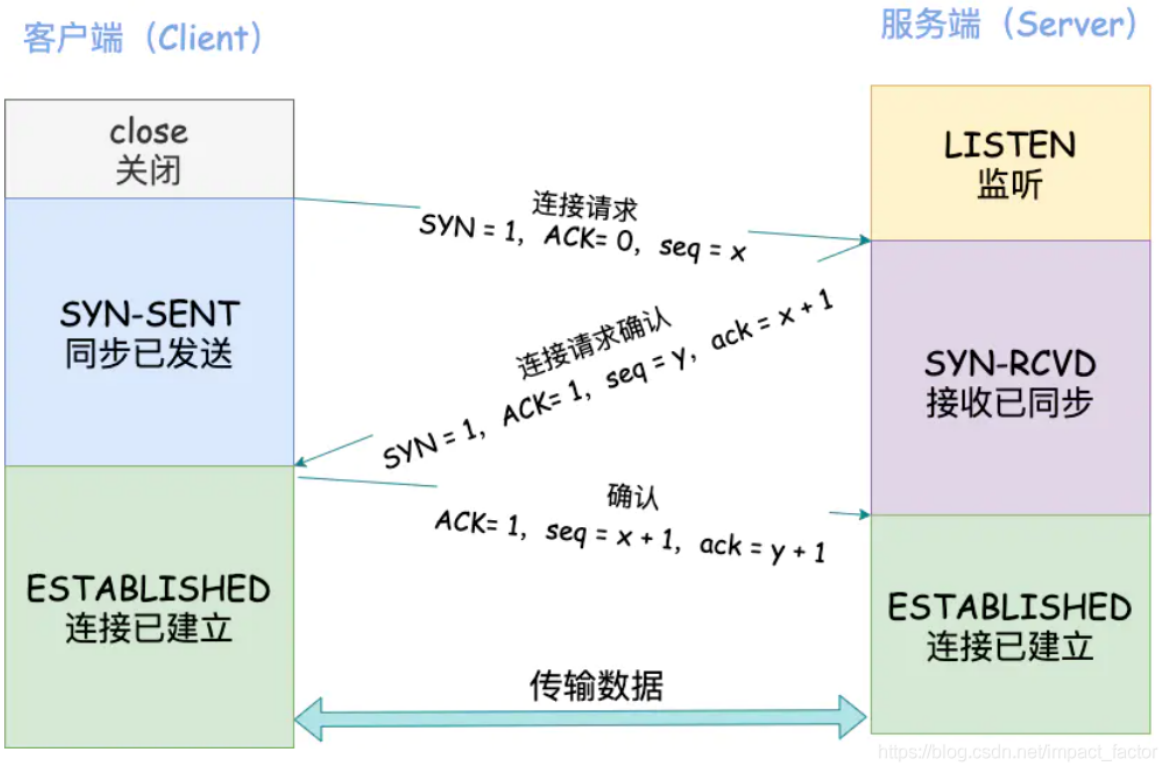
实验抓包

使用wireshark进行分析，进行ip地址和端口号过滤，可以看到结果如下：



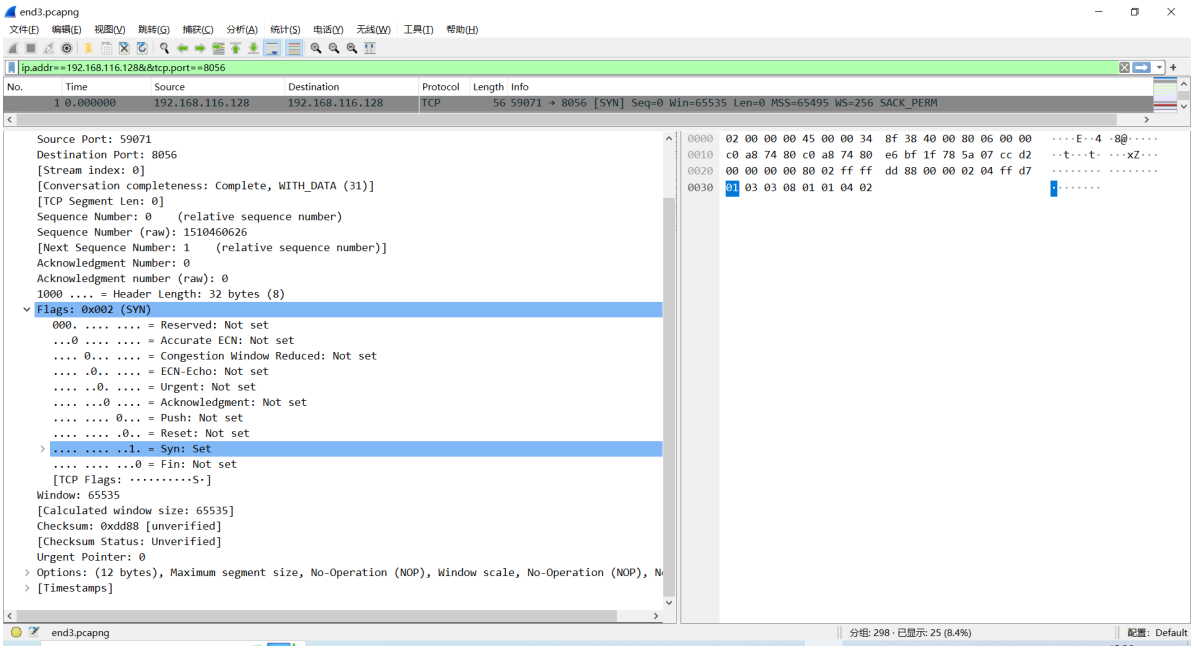
三次握手

三次握手的过程可以看到如下图所示：



根据此图去分析我们的抓包过程，在捕获过程中，客户端有两个端口进行交互，此次只分析进行数据传输的端口59071

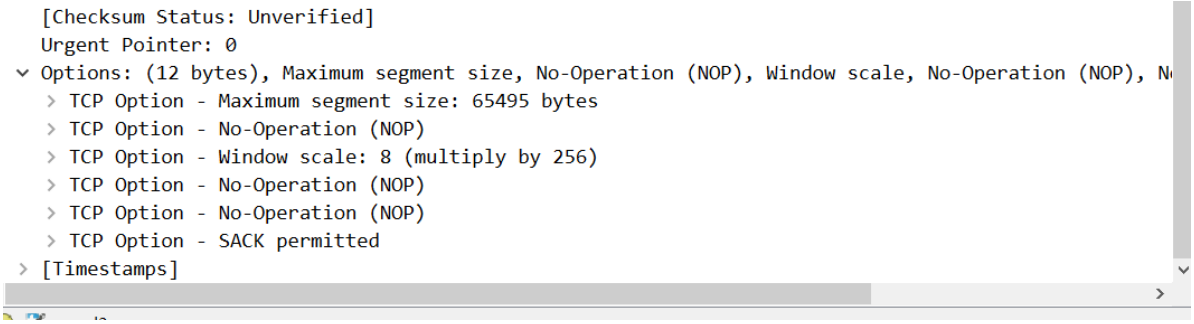
第一次握手



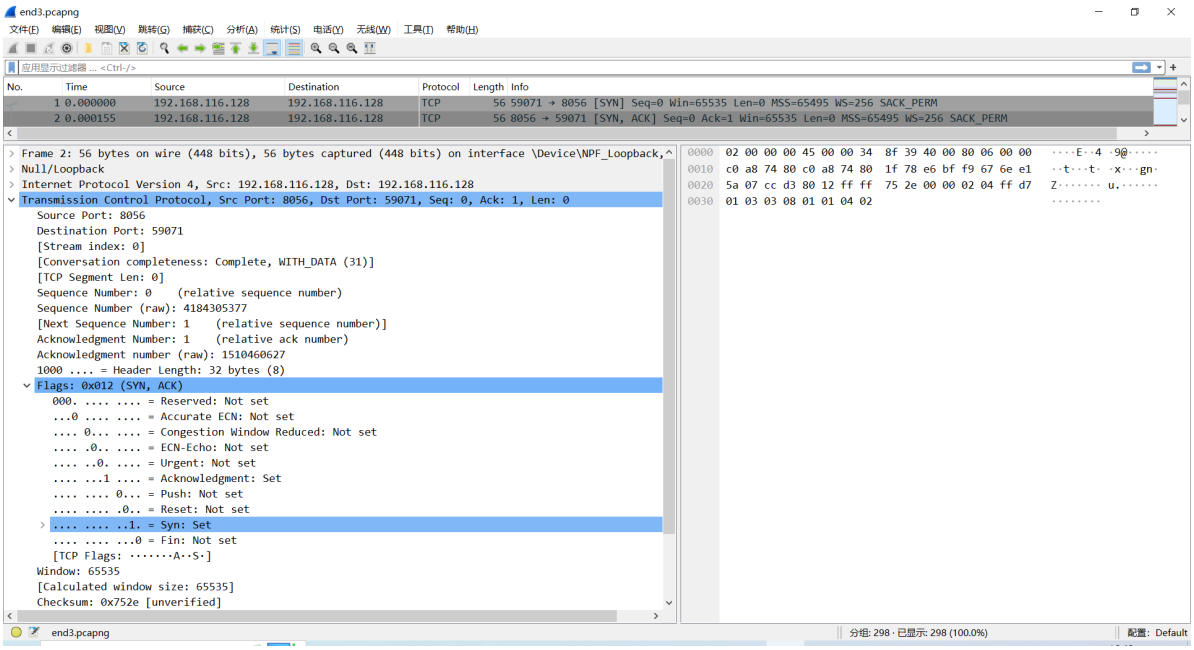
从数据包中我们可以看到一些信息，例如源端口59071和目的端口8056，可以看到SYN置了1，这说明它是TCP握手中的第一个数据包（第一次握手）——客户端程序向服务器发送建连请求。ACK为0，而其他标志位都没有设置，TCP规定除了在最初建立连接时候的SYN包之外该位必须设置为1，该位为1时，确认应答的字段为有效。

此时客户端随机产生一个序列号x=1510460626，相对于整个流中客户端所发送的数据包是第0个数据包。

数据包中还有一个WIN参数，这个是代表了TCP数据窗口大小，它是TCP接收缓冲区，用于尚未由应用程序处理的传入数据。使用TCP头的窗口大小值字段将TCP接收窗口的大小传达给连接伙伴。该字段告诉链路伙伴在接收到确认之前可以在线路上发送多少数据。如果接收器无法尽快处理数据，则接收缓冲区将逐渐填充，并且确认数据包中的TCP窗口将减少。这将警告发送方它需要减少发送的数据量或让接收方有时间清除缓冲区。这里WIN=65535，是指该数据包Window大小为65535（16bits能表示的最大值），在options里还有一个Window Scale，这个是指一个缩放系数，这里是8，要左移八位，也就是说接收窗口为 65335*256=1677960bytes；缩放系数如果以后不调整就固定了。发送方的缩放系数和接收方的乘积因子可以不同，由各自决定，观察后续数据包发现这里两方恰巧是一样的。Window大小是动态调整的。



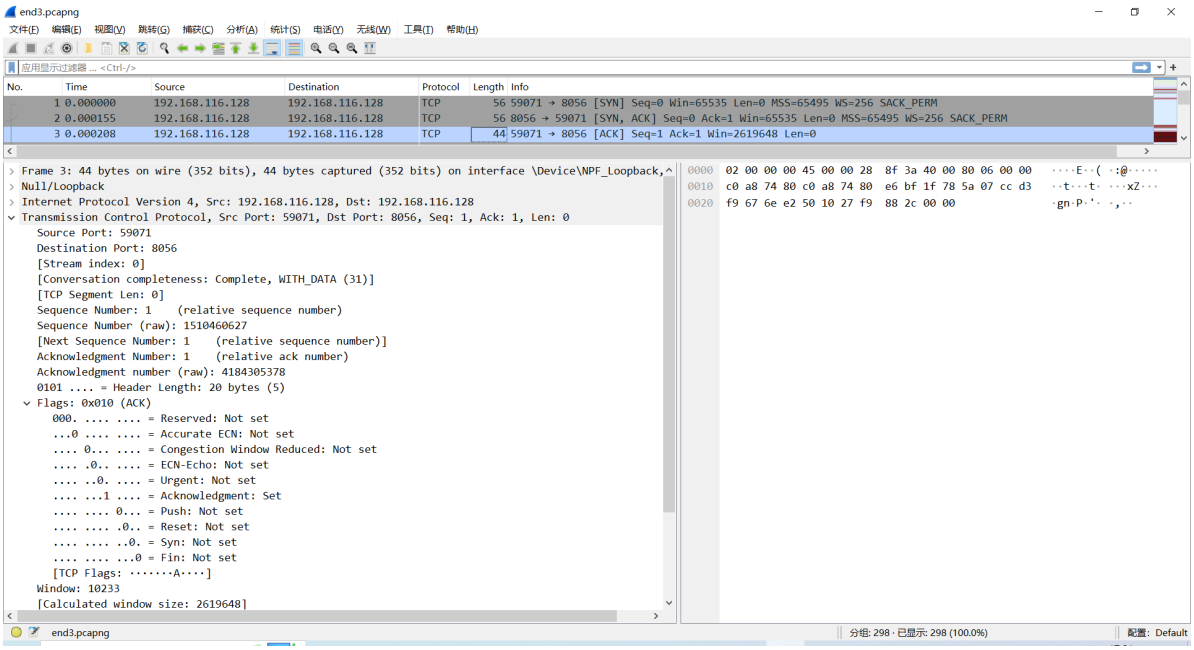
第二次握手



服务端收到请求后，返回 客户端的SYN = 1,加上自己的确认号ACK=1,发送的具体数据第一个字节编号记为y，赋值seq，希望客户端下一次返回编号x + 1个字节为止的数据，记为ack = x + 1，可以看到这里的确认号就是前面的上次握手的序列号1510460626+1=x=1510460627.此时服务器生成了一个新的随机的序列号y，4184305377。要注意的，在整个TCP流中，浏览器和服务器各自作为发送端的时候，使用不同的起始序列号，分别为x和y。

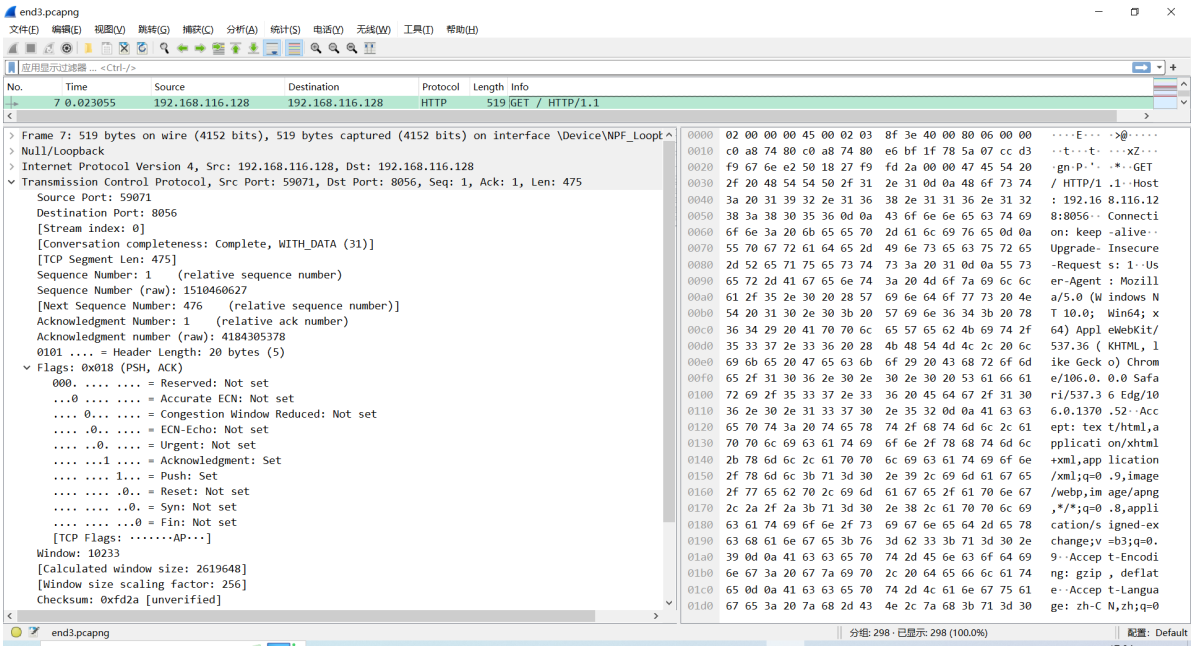
此时Flags标志位的SYN和ACK都被置1，这说明是第二次握手的数据包。

第三次握手

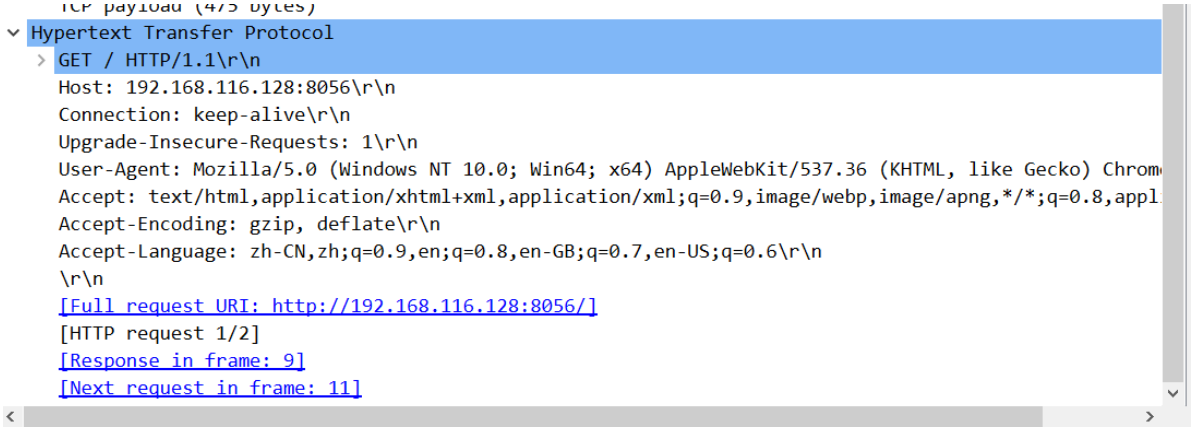


前面的两次握手客户端得出客户端发送接收能力正常，服务端发送接收能力也都正常，但是此时服务器并不能确认客户端的接收能力是否正常。客户端收到服务端返回的请求确认后，再次发送数据，原封不动返回ACK = 1,这里就不需要再发送 SYN=1了，为什么呢？因为此时并不是跟服务端进行连接请求，而是连接确认，所以只需要返回ACK = 1代表确认，同样的，发送的具体数据第一个字节编号记为seq = x + 1，希望服务端下次传输的数据第一个字节编号记为ack = y + 1。

HTTP请求和响应



可以看到，客户端向服务器端发出了一个GET请求，此时的seq和ack依旧和上一个是保持一致的。Flags字段给出了ACK和PSH，PSH表示接收方应该尽快将这个报文交给应用层。TCP层将HTTP报文放在了TCP数据部分，长度为475bytes。



上图是HTTP请求报文

请求报文由三部分组成：请求行(request line)、首部行(header line)、实体体(entity body)。

GET那行是请求行，由方法、URL、HTTP版本。该报文使用GET方法拉取HTML网页，HTTP版本为HTTP/1.1，“/”代表请求当前目录下文件

后面都是首部行，实体部分缺省。

Host：初始URL中的主机和端口。

Connection：表示是否需要持久连接，这里是HTTP1.1默认就是持久连接。

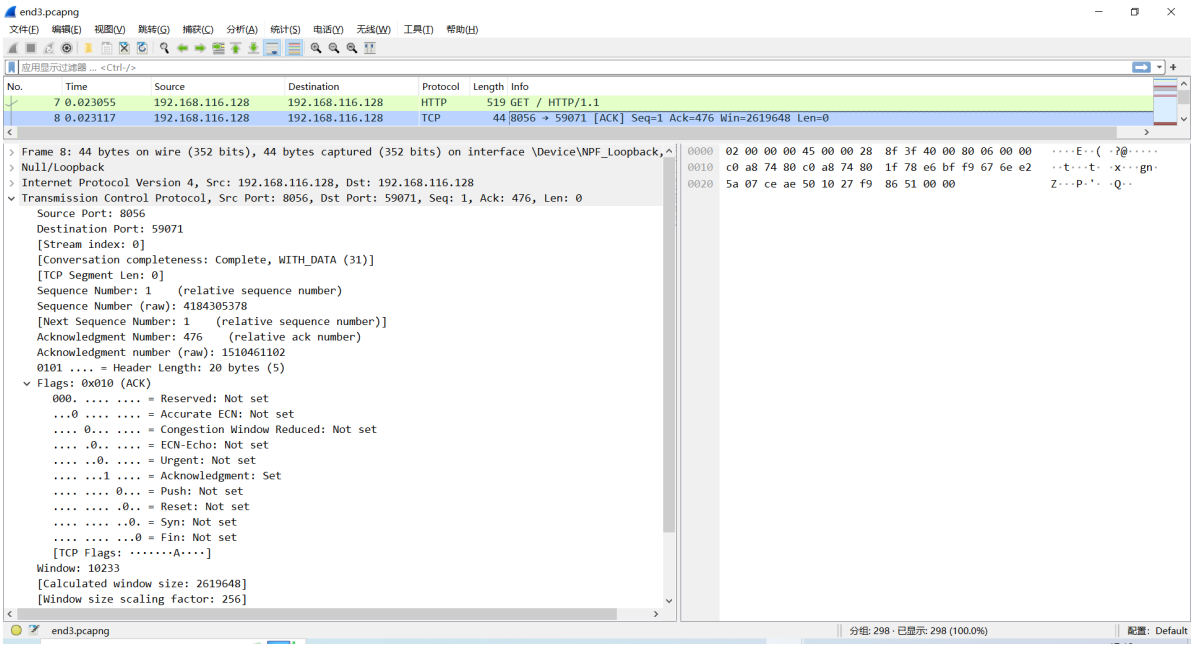
User-Agent：浏览器类型。

如果是第二次打开，会加上如下两个

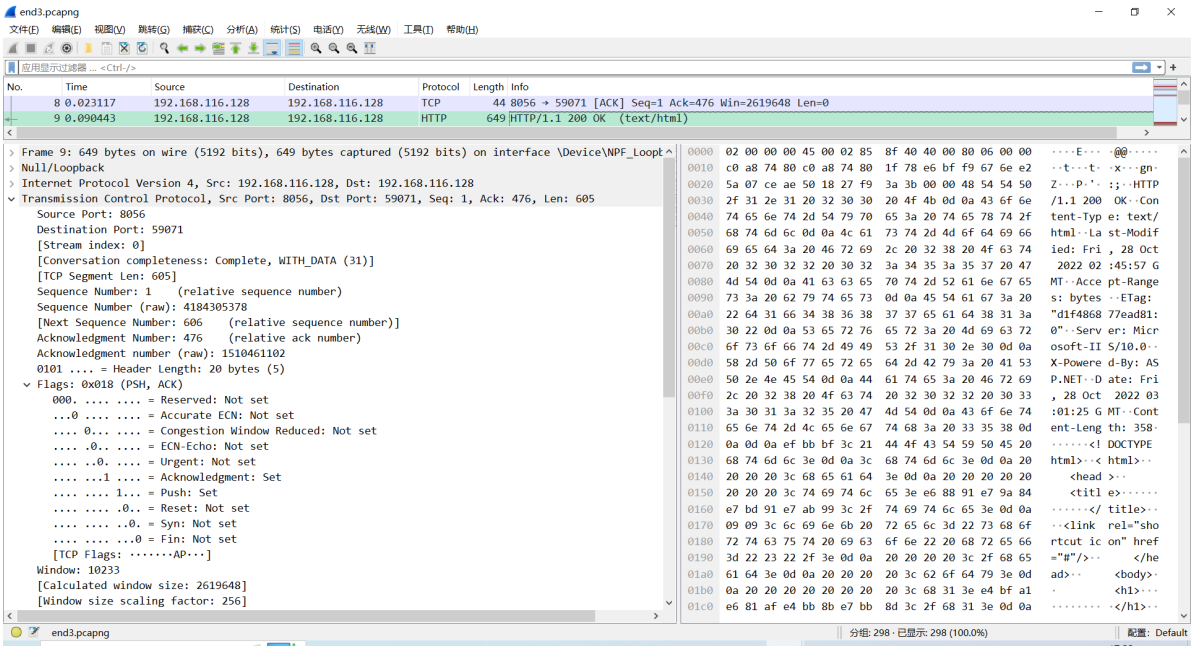
```
Hypertext Transfer Protocol
  GET / HTTP/1.1\r\n
  Host: 192.168.116.128:8056\r\n
  Connection: keep-alive\r\n
  Upgrade-Insecure-Requests: 1\r\n
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,applic
  Accept-Encoding: gzip, deflate\r\n
  Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6\r\n
  If-None-Match: "d1f486877ead81:0"\r\n
  If-Modified-Since: Fri, 28 Oct 2022 02:45:57 GMT\r\n
  \r\n
  [Full request URI: http://192.168.116.128:8056/]
  [HTTP request 1/2]
  [Response in frame: 21]
  [Next request in frame: 23]
```

If-Modified-Since: Fir, 28 Oct 2022 02:45:57 GMT\r\n 这是浏览器缓存中保存的该文件的最后修改时间。浏览器发送HTTP请求时，把If-Modified-Since一起发到服务器去，服务器会把这个时间与服务器上实际文件的最后修改时间进行比较。如果时间一致，那么返回HTTP状态码304（Not Modified），客户端接到之后，直接把本地缓存文件显示到浏览器中。如果时间不一致，就返回HTTP状态码200和新的文件内容，客户端接到之后，会丢弃旧文件，把新文件缓存起来，并显示到浏览器中。

If-None-Match：这里记录的是首次发起这个这个请求时服务器返回的Etag，ETag实体标签一般为资源实体的哈希值是服务器生成的一个标记，用来标识返回值是否有变化，Etag的优先级高于Last-Modified。在下一次发起与之前相同的请求时，客户端会同时发送一个If-NoneMatch，而它的值就是Etag的值。然后，服务器会比对这个客服端发送过来的Etag是否与服务器的相同，如果相同，就将If-None-Match的值设为false，返回状态为304，客户端继续使用本地缓存，不解析服务器返回的数据；如果不相同，就将If-None-Match的值设为true，返回状态为200，客户端重新解析服务器返回的数据。

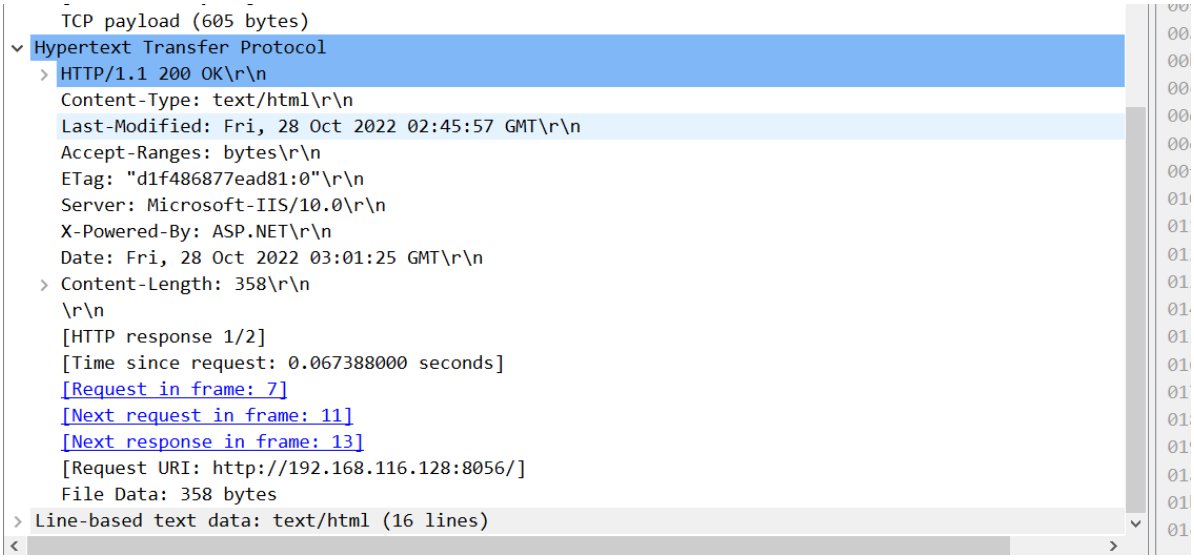


下一个数据包是服务器对该数据包的响应，其Flags中仅ACK被置1，如上图所示。注意到其序列号为y+1，其确认号为1510461102，用这个数字减去x+1的话，正好是要发送的HTTP请求报文的长度，这充分说明序列号衡量的是要发送的信息的长度（而跟TCP首部基本无关）。



上图的数据包时服务器对客户端GET的响应，此时序列号为y+1，Flags字段增加了PSH。

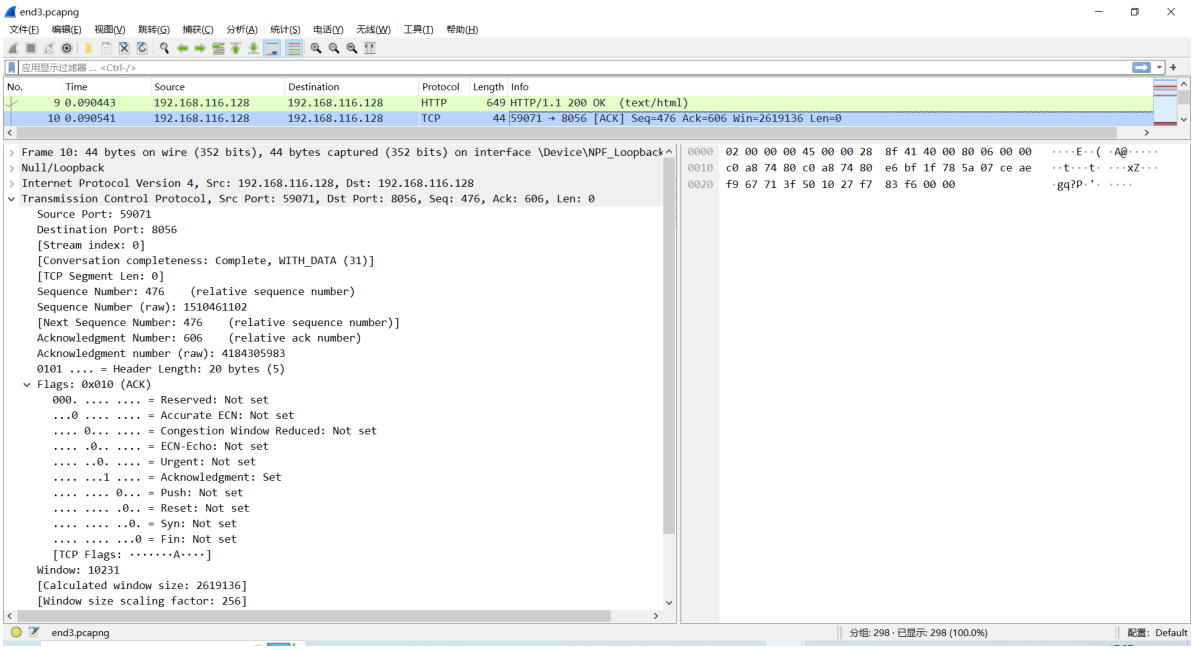
响应报文同样由三部分组成：状态行(status line)、首部行(header line)、实体体(entity body)。



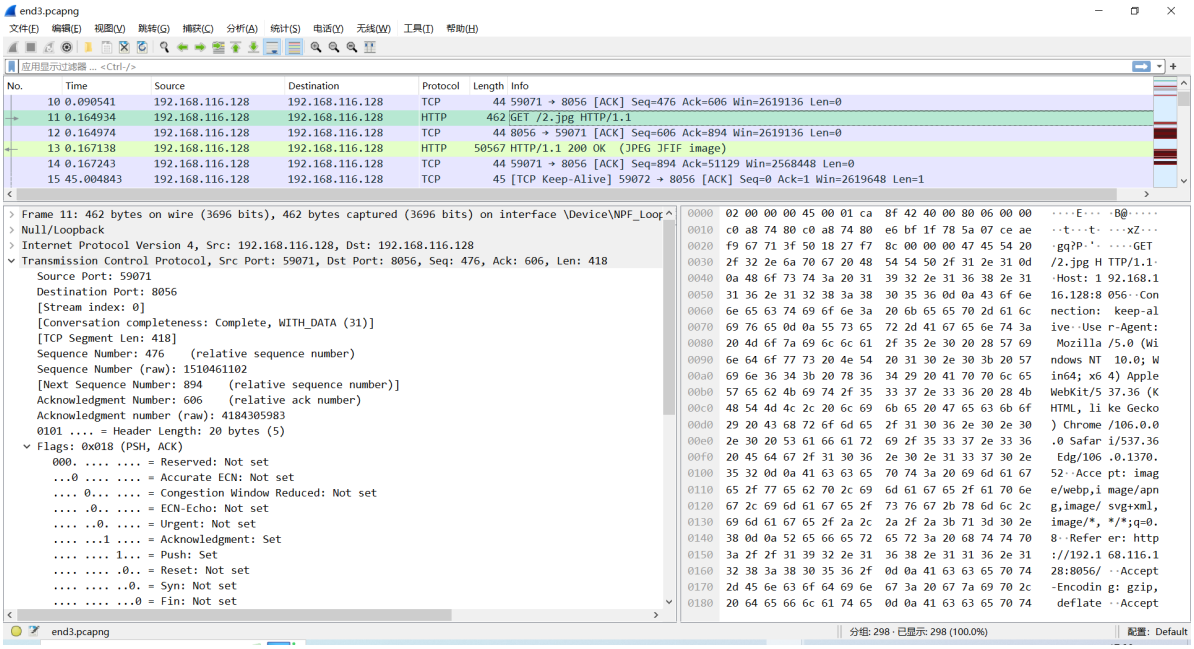
状态行的参数：

- 200 OK：请求成功，信息在返回的响应报文中。
- 304 Not modified 客户端在第一次对服务器业务发出GET请求后，客户端浏览器缓存了该页面，当客户端第二次对服务器发出同样的GET请求时，若客户端缓存中的If-Modified-Since过期，客户端将向服务器发出GET请求，验证If-Modified-Since和If-None-Match是否与WEB-server中信息一致，如果GET页面未做任何修改，服务器对客户端返回HTTP/1.1 304 Not Modified，客户端则直接从本地缓存中将页面调取。
- 404 Not Found：被请求的文档不在服务器上。

之前由于未把图片放在该目录下，返回了404.

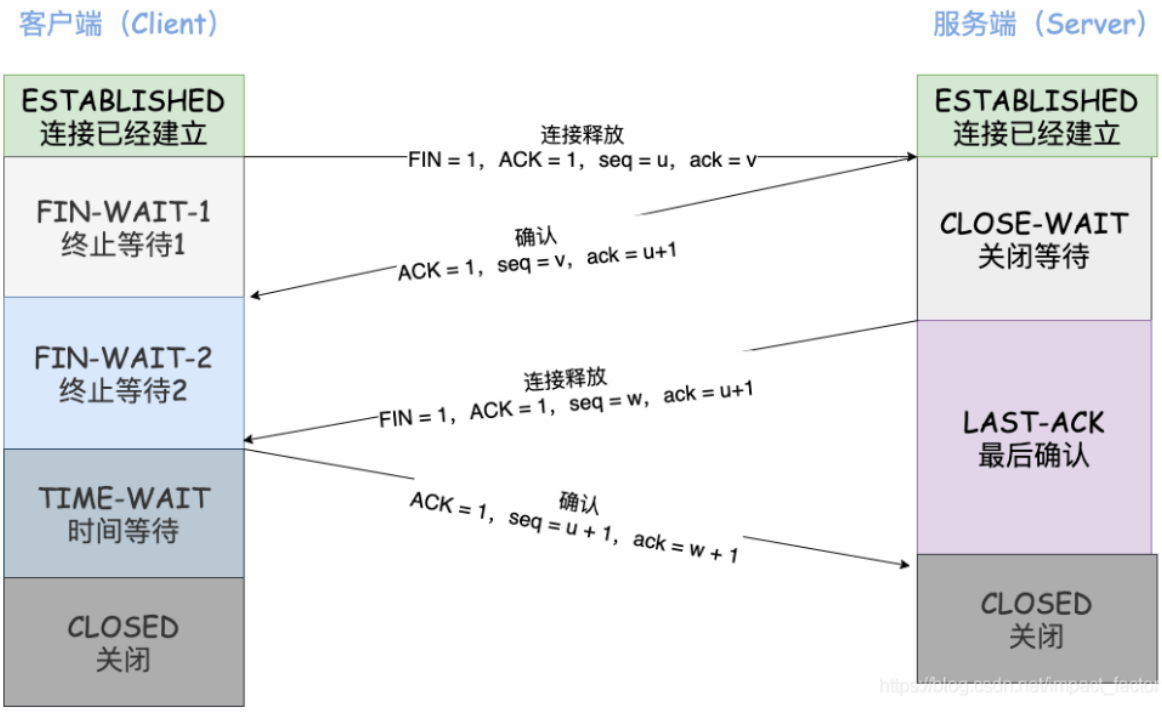


客户端收到信息后，向服务器端传递一个ACK，可以看到，其确认号为4184305983，减去x+1正好是上一个报文中HTTP响应报文的长度。



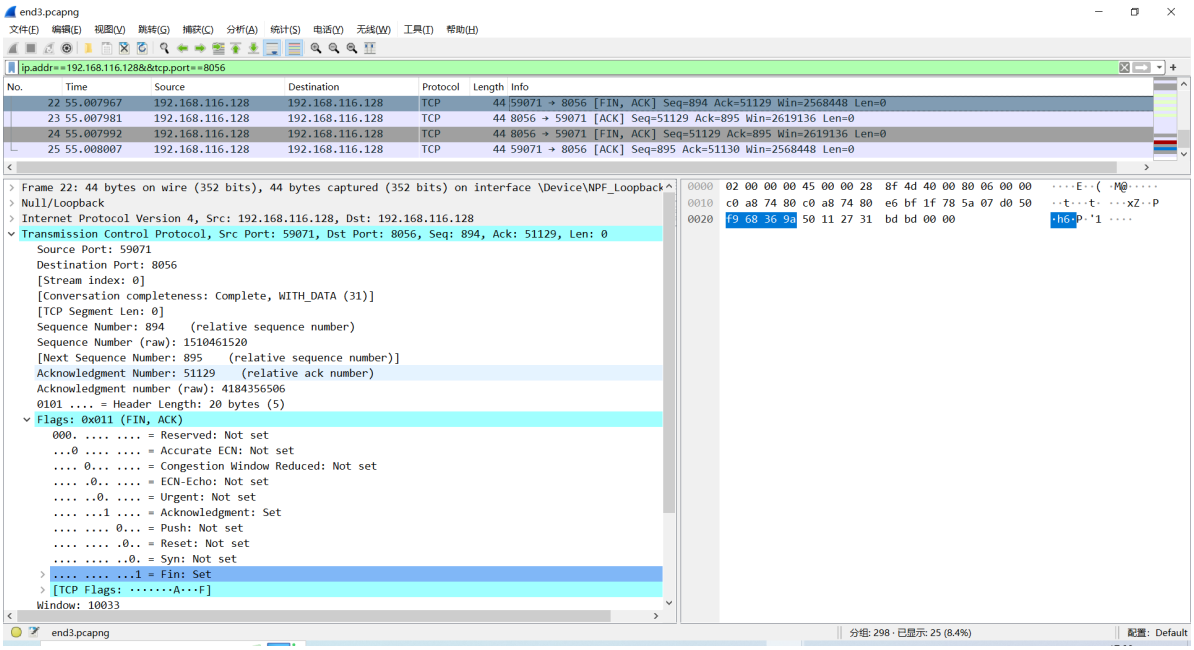
11-14号数据包是客户端发出GET命令请求图片文件，整个过程和上面的相似。

四次挥手



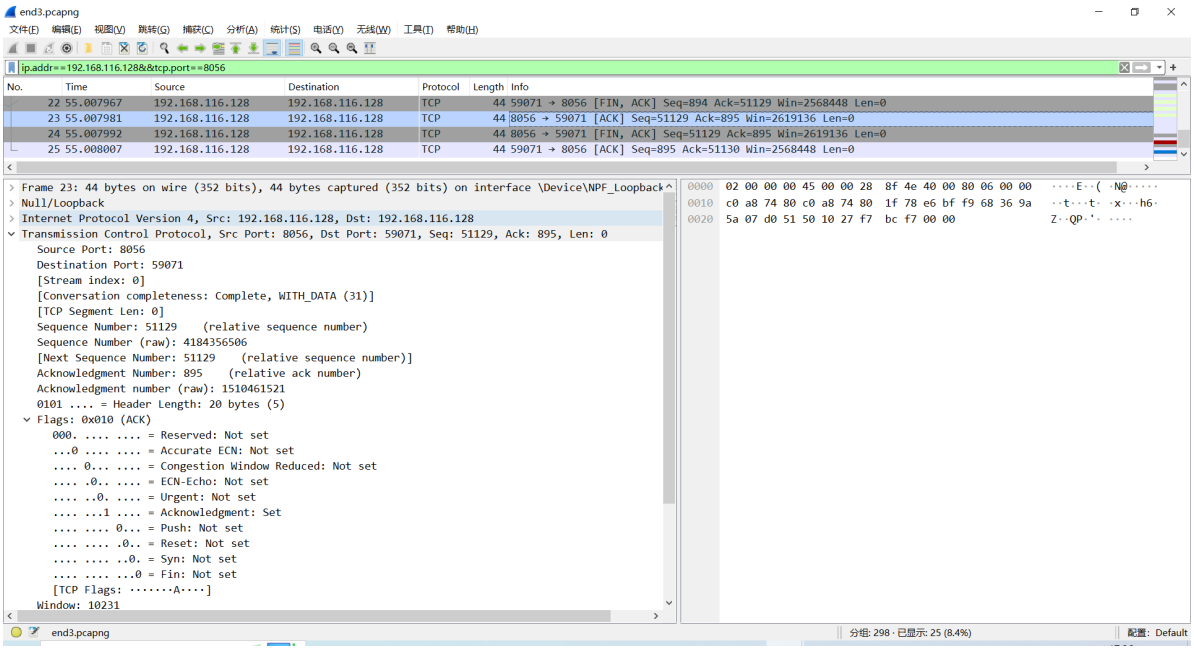
第一次挥手

22号数据包是客户端先发出断连请求



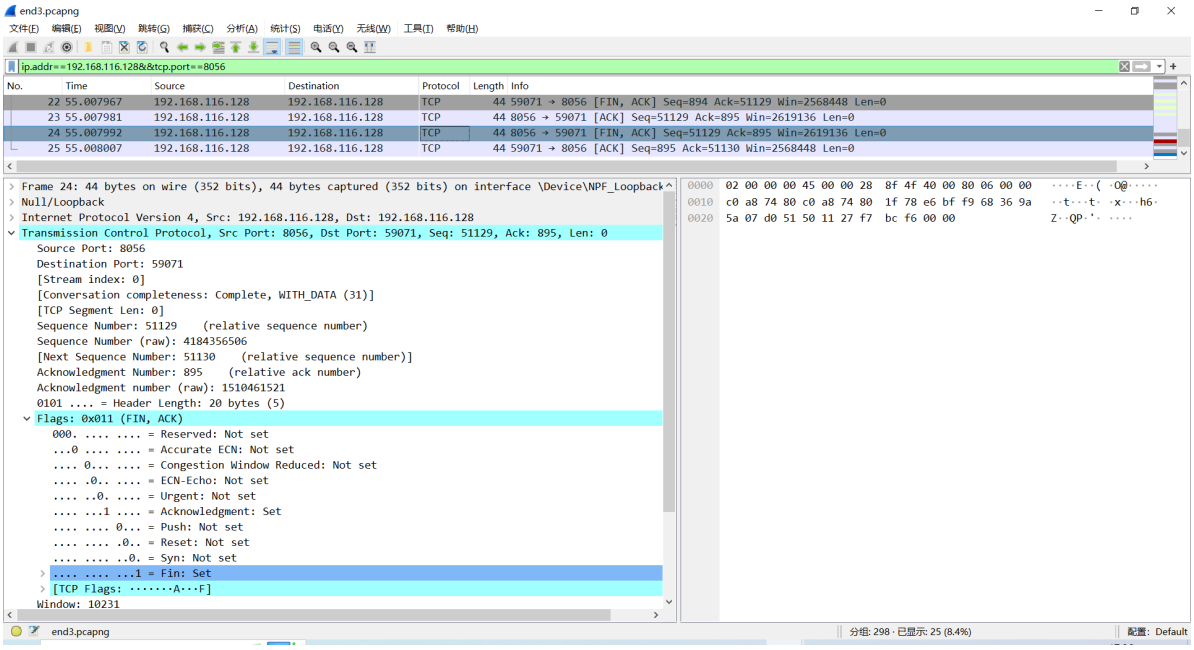
Client发送一个FIN，用来关闭Client到Server的数据传送，Client进入FIN_WAIT_1状态。

第二次挥手



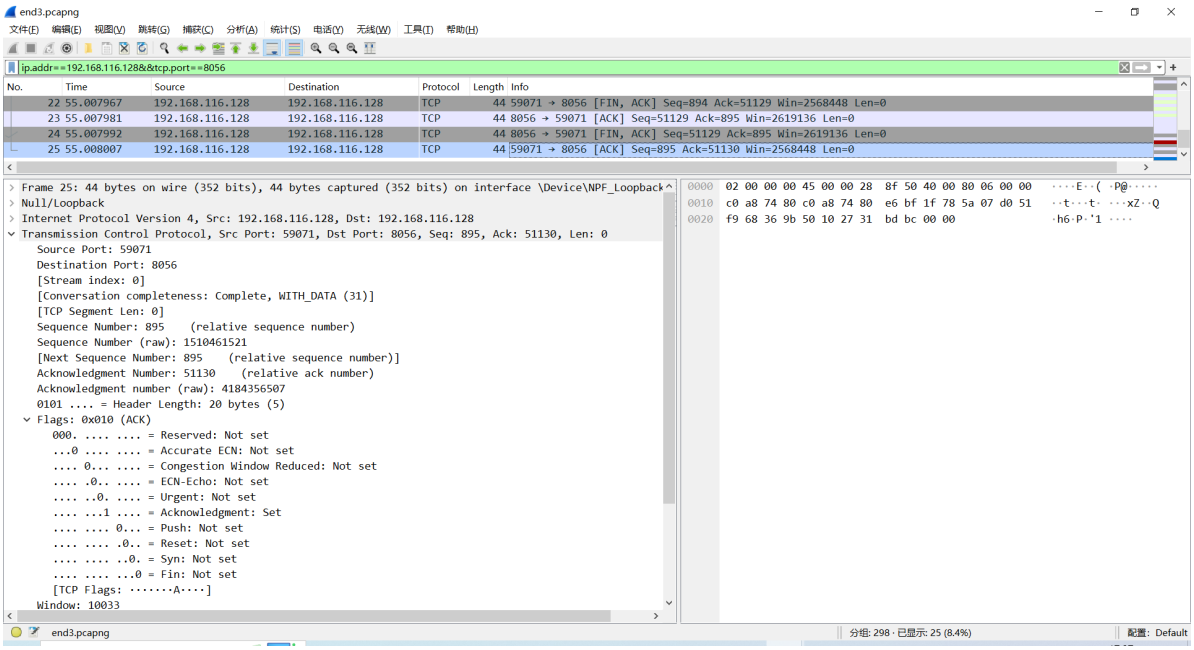
Server收到FIN后，发送一个ACK给Client，确认序号为收到序号+1（与SYN相同，一个FIN占用一个序号），Server进入CLOSE_WAIT状态。

第三次挥手



Server发送一个FIN，用来关闭Server到Client的数据传送，Server进入LAST_ACK状态。

第四次挥手



Client收到FIN后，Client进入TIME_WAIT状态，接着发送一个ACK给Server，确认序号为收到序号+1，Server进入CLOSED状态，完成四次挥手

可以看到每次数据包的序列号和确认号，也满足最开始原理图的运算。

为什么要进行三次握手和四次挥手呢

建立连接的时候， 服务器在LISTEN状态下，收到建立连接请求的SYN报文后，把ACK和SYN放在一个报文里发送给客户端。三次握手主要目的是:防止server端一直等待，浪费资源，而关闭连接时，服务器收到对方的FIN报文时，仅仅表示对方不再发送数据了但是还能接收数据，而自己也未必全部数据都发送给对方了，所以己方可以立即关闭，也可以发送一些数据给对方后，再发送FIN报文给对方来表示同意现在关闭连接，因此，己方ACK和FIN一般都会分开发送，从而导致多了一次。为了确保正确关闭连接，所以需要四次。