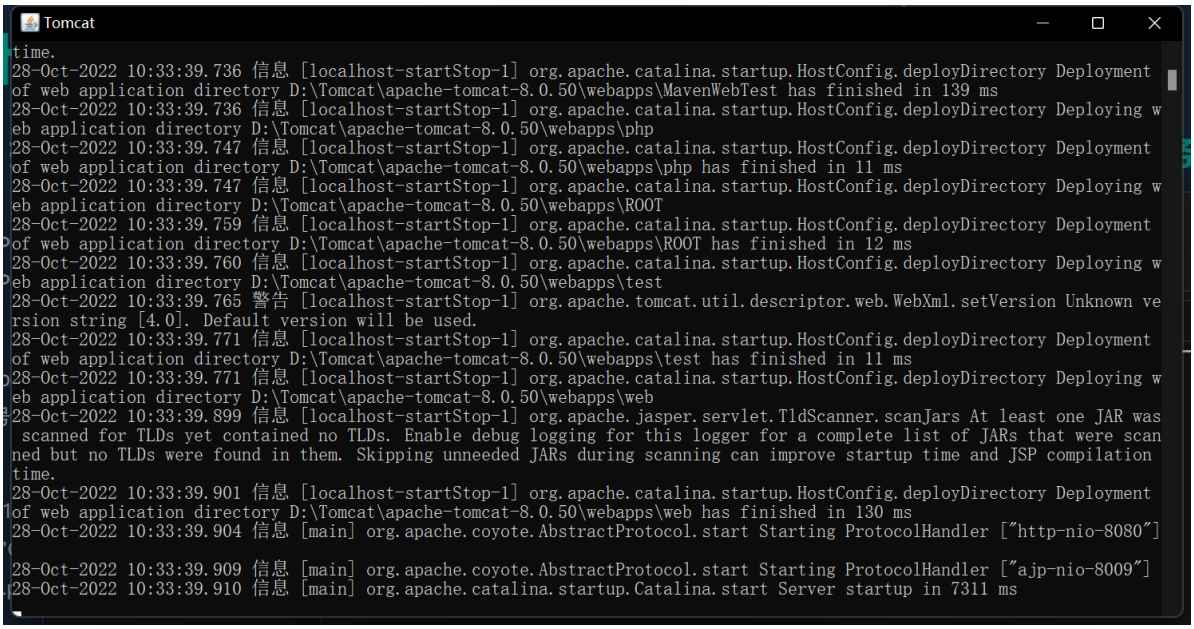
Lab2

搭建web服务器

本次实验使用的是之前已经配置好的Apache-Tomcat-8.0.50服务器，将编写好得HTML文件和所需要得资源文件放到对应的webapps目录下，然后启动Tomcat服务器，即可完成服务器的搭建工作，具体效果如图所示。



服务器的默认端口号为8080，启动后，访问localhost:8080/test/index.html即可看到我们的页面

HTML页面

```
1 <!DOCTYPE html>
```

```html
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta Cache-Control="no-store">
    <title>Title</title>
    <link rel="stylesheet" href="static/css/test.css">
</head>
<body>
<h1>Hello World!</h1>
<!--插入图片 static/image/logo.png 并设置居中 大小100*100-->
<img class="logo" src="static/image/logo.png" alt="">
<div>
<p>姓名 : 刘宇轩</p>
<p>学号 : 2012677</p>
<p>年级 : 2020级</p>
<p>专业 : 计算机科学与技术</p>
</div>
</body>
</html>
```

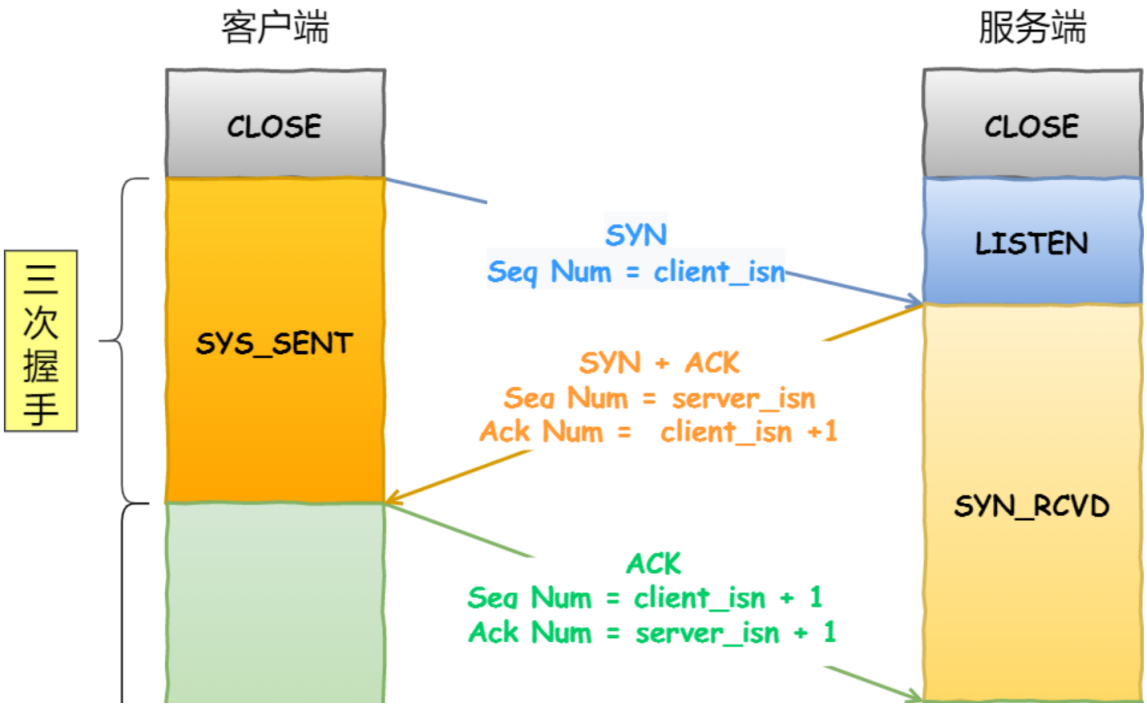
```css
/*<h1>标签居中展示*/
h1{
    text-align: center;
}

img.logo{
    display: block;
    margin: 0 auto;
    width: 100px;
    height: 100px;
}

div {
    align-content: center;
    text-align: center;
}
```

Wireshark捕获交互过程

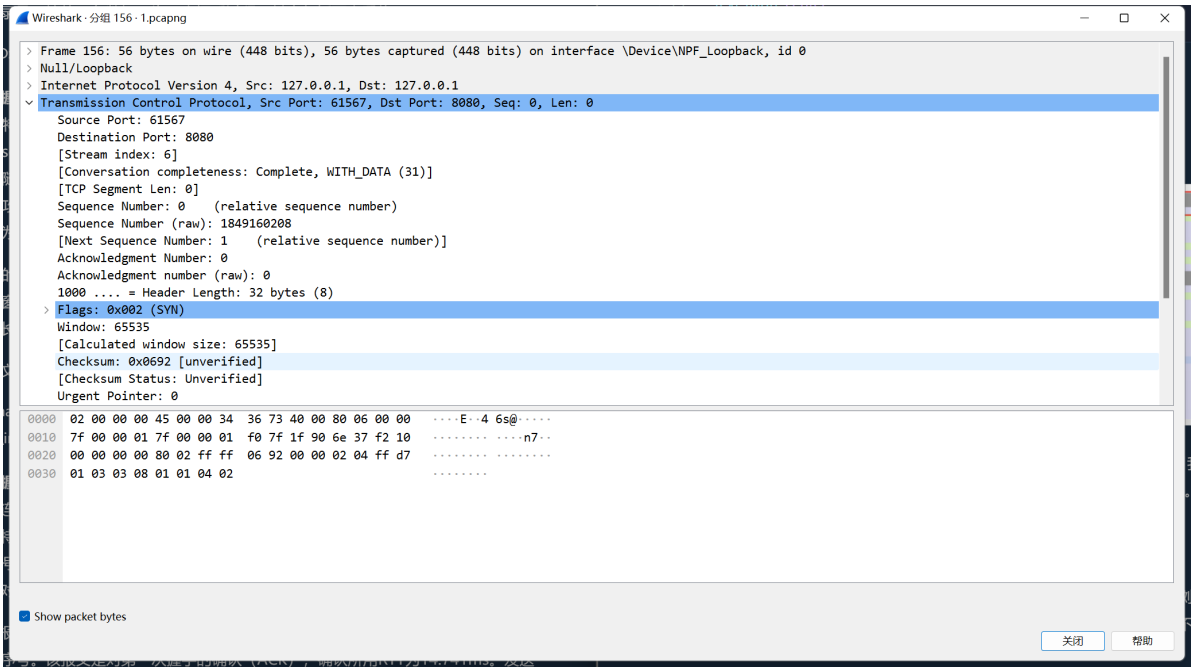
三次握手建连

No.	Time	Source	Destination	Protocol	Length	Info
156	2022-10-26 12:25:29.630653	127.0.0.1	127.0.0.1	TCP	56	61567 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
157	2022-10-26 12:25:29.630689	127.0.0.1	127.0.0.1	TCP	56	8080 → 61567 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
158	2022-10-26 12:25:29.630708	127.0.0.1	127.0.0.1	TCP	44	61567 → 8080 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
159	2022-10-26 12:25:29.630861	127.0.0.1	127.0.0.1	HTTP	777	GET /test/index.html HTTP/1.1
160	2022-10-26 12:25:29.630875	127.0.0.1	127.0.0.1	TCP	44	8080 → 61567 [ACK] Seq=1 Ack=734 Win=2160384 Len=0
163	2022-10-26 12:25:29.669962	127.0.0.1	127.0.0.1	TCP	270	8080 → 61567 [PSH, ACK] Seq=1 Ack=734 Win=2160384 Len=226 [TCP segment of a reassembled PDU]
164	2022-10-26 12:25:29.669994	127.0.0.1	127.0.0.1	TCP	44	61567 → 8080 [ACK] Seq=734 Ack=227 Win=2160896 Len=0
167	2022-10-26 12:25:29.670413	127.0.0.1	127.0.0.1	HTTP	552	HTTP/1.1 200 OK (text/html)
168	2022-10-26 12:25:29.670426	127.0.0.1	127.0.0.1	TCP	44	61567 → 8080 [ACK] Seq=734 Ack=735 Win=2160384 Len=0
173	2022-10-26 12:25:29.681181	127.0.0.1	127.0.0.1	HTTP	711	GET /test/static/css/test.css HTTP/1.1
174	2022-10-26 12:25:29.681212	127.0.0.1	127.0.0.1	TCP	44	8080 → 61567 [ACK] Seq=735 Ack=1401 Win=2159872 Len=0
175	2022-10-26 12:25:29.681424	127.0.0.1	127.0.0.1	TCP	56	61568 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
176	2022-10-26 12:25:29.681454	127.0.0.1	127.0.0.1	TCP	56	8080 → 61568 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
177	2022-10-26 12:25:29.681473	127.0.0.1	127.0.0.1	TCP	44	61568 → 8080 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
178	2022-10-26 12:25:29.681560	127.0.0.1	127.0.0.1	HTTP	713	GET /test/static/image/logo.png HTTP/1.1
179	2022-10-26 12:25:29.681571	127.0.0.1	127.0.0.1	TCP	44	8080 → 61568 [ACK] Seq=1 Ack=670 Win=2160640 Len=0
182	2022-10-26 12:25:29.683261	127.0.0.1	127.0.0.1	TCP	269	8080 → 61567 [PSH, ACK] Seq=735 Ack=1401 Win=2159872 Len=225 [TCP segment of a reassembled PDU]
183	2022-10-26 12:25:29.683278	127.0.0.1	127.0.0.1	TCP	44	61567 → 8080 [ACK] Seq=1401 Ack=960 Win=2160128 Len=0
186	2022-10-26 12:25:29.683338	127.0.0.1	127.0.0.1	HTTP	265	HTTP/1.1 200 OK (text/css)
187	2022-10-26 12:25:29.683345	127.0.0.1	127.0.0.1	TCP	44	61567 → 8080 [ACK] Seq=1401 Ack=1181 Win=2160128 Len=0
192	2022-10-26 12:25:29.684079	127.0.0.1	127.0.0.1	TCP	276	8080 → 61568 [PSH, ACK] Seq=1 Ack=670 Win=2160640 Len=232 [TCP segment of a reassembled PDU]
193	2022-10-26 12:25:29.684096	127.0.0.1	127.0.0.1	TCP	44	61568 → 8080 [ACK] Seq=670 Ack=233 Win=2160896 Len=0
196	2022-10-26 12:25:29.685268	127.0.0.1	127.0.0.1	TCP	65539	8080 → 61568 [ACK] Seq=233 Ack=670 Win=2160640 Len=65495 [TCP segment of a reassembled PDU]
197	2022-10-26 12:25:29.685294	127.0.0.1	127.0.0.1	TCP	65539	8080 → 61568 [ACK] Seq=65728 Ack=670 Win=2160640 Len=65495 [TCP segment of a reassembled PDU]
198	2022-10-26 12:25:29.685314	127.0.0.1	127.0.0.1	TCP	125	8080 → 61568 [PSH, ACK] Seq=131223 Ack=670 Win=2160640 Len=81 [TCP segment of a reassembled PDU]
199	2022-10-26 12:25:29.685356	127.0.0.1	127.0.0.1	TCP	44	61568 → 8080 [ACK] Seq=670 Ack=131304 Win=2161152 Len=0
200	2022-10-26 12:25:29.685425	127.0.0.1	127.0.0.1	TCP	65539	8080 → 61568 [ACK] Seq=131304 Ack=670 Win=2160640 Len=65495 [TCP segment of a reassembled PDU]
201	2022-10-26 12:25:29.685444	127.0.0.1	127.0.0.1	TCP	65539	8080 → 61568 [ACK] Seq=196799 Ack=670 Win=2160640 Len=65495 [TCP segment of a reassembled PDU]
202	2022-10-26 12:25:29.685463	127.0.0.1	127.0.0.1	TCP	125	8080 → 61568 [PSH, ACK] Seq=262294 Ack=670 Win=2160640 Len=81 [TCP segment of a reassembled PDU]
203	2022-10-26 12:25:29.685510	127.0.0.1	127.0.0.1	TCP	44	61568 → 8080 [ACK] Seq=670 Ack=262375 Win=2095616 Len=0
204	2022-10-26 12:25:29.685594	127.0.0.1	127.0.0.1	TCP	65539	8080 → 61568 [ACK] Seq=262375 Ack=670 Win=2160640 Len=65495 [TCP segment of a reassembled PDU]
205	2022-10-26 12:25:29.685644	127.0.0.1	127.0.0.1	TCP	44	[TCP Window Update] 61568 → 8080 [ACK] Seq=670 Ack=262375 Win=2161152 Len=0

在浏览器中输入 127.0.0.1:8080/test/index.html 敲回车后，会跳转到我们编写的HTML页面中，此时抓包能够看到浏览器和服务器刚开始建连时三次握手的过程。



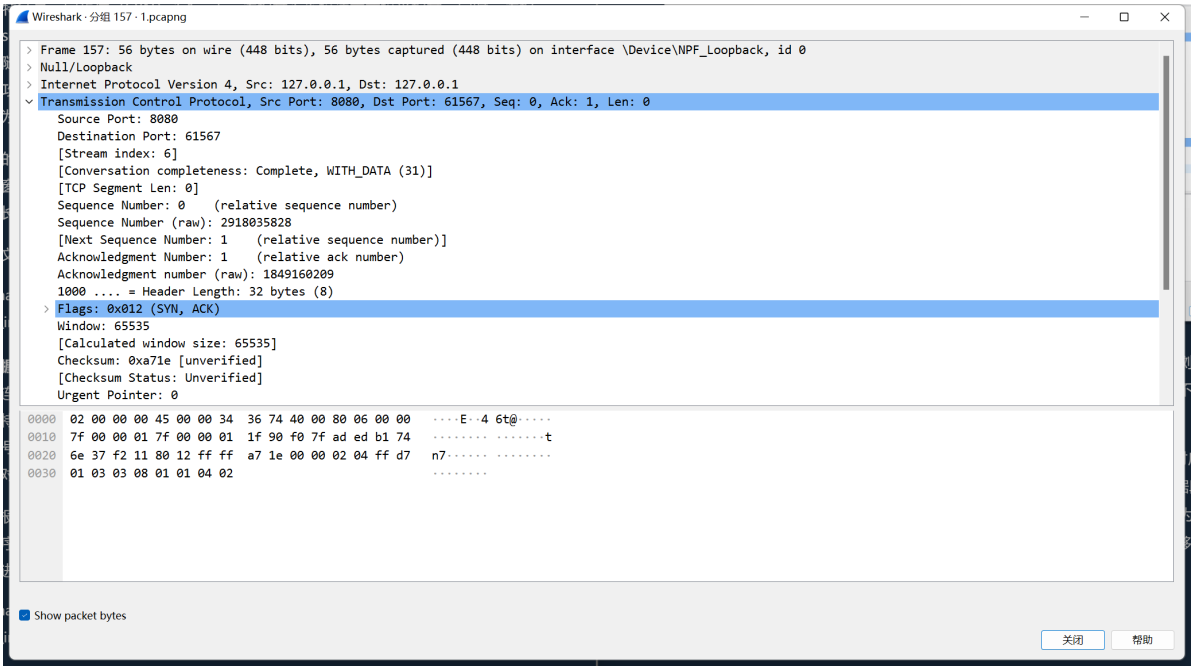
第一次握手



在输入了网址之后，浏览器要主动和服务器建立连接，因此第一次握手是浏览器向服务器发送了TCP请求，由于是回环测试，所以IP地址均为127.0.0.1，但是端口号不同，浏览器端口号为61567，服务器端口号为8080。

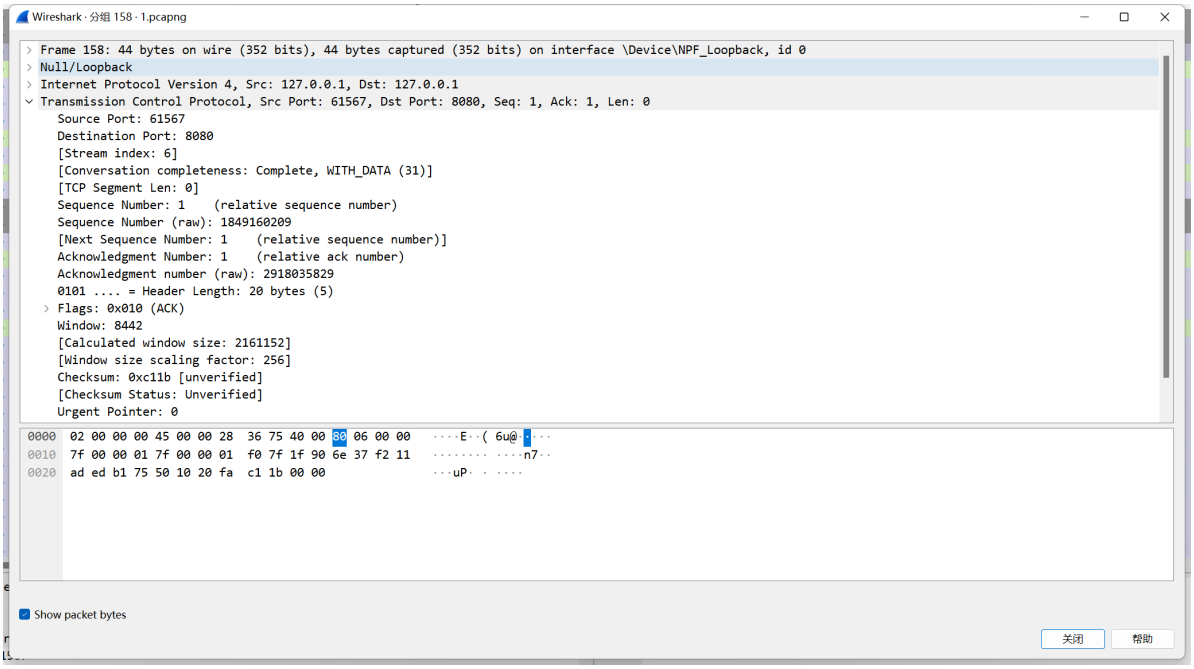
此时同步位`SYN`为1，表示这条报文为建立连接或接收请求的报文。相对序号`Seq`为0，表示之前并没有数据发送。由于只是在建立连接的过程中，因此报文的数据段长度`Len`为0。报文段的最大长度`MSS`为65459，表示所能够接收到的报文段数据最大为65459个字节。`WIN`字段表示接收端还能够接收的字节数，下一次发送的数据大小不能够超过这个数值。`WS`表示窗口的大小为8*256。`SACK_PERM`表示允许选择确认重传机制。

第二次握手



第二次握手是服务器接收到了浏览器发起的连接请求后，给浏览器的应答，并在之后为该TCP分配缓存和变量。可以看到这条TCP报文是从8080端口发向61567端口的。此时的标志位为`SYN`和`ACK`，自身的相对序号`Seq`仍为0，确认号为接收到的`Seq+1`，所以此时的`ACK`为1。其余的数据段和第一次握手分析一致，不再重复。

第三次握手

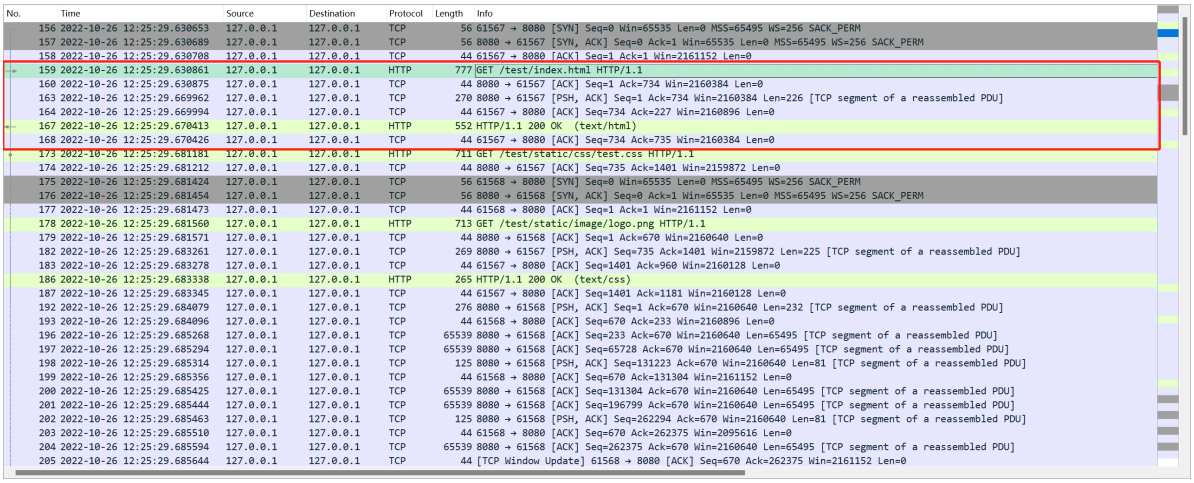


第三次握手是浏览器收到服务器回应的确认报文后，给服务器的应答，并为TCP连接分配缓存和变量。这里的flag中只有`ACK`一位有效，表示这只是一条确认报文。由于之前已经消耗了一个序号，因此这条报文的序号`Seq`为1，确认号`ACK`为第二次握手的`Seq+1`，值为1。注意到此时的窗口大小已经发生了调整，变成了8442*256=2161152。

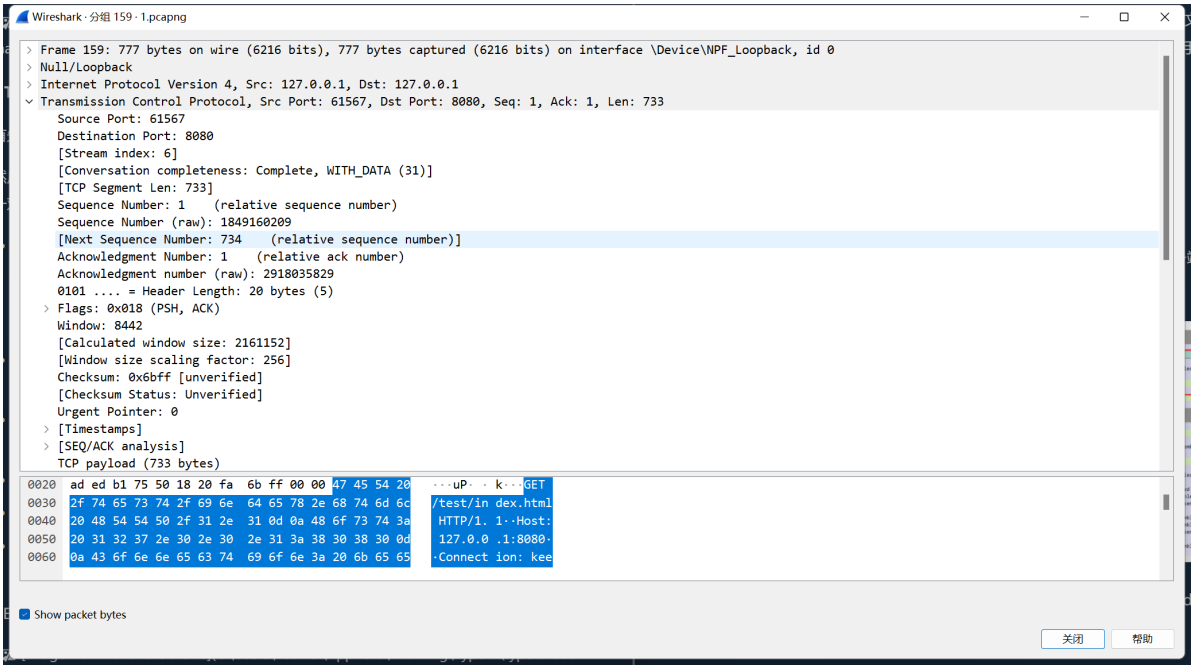
经过三次握手之后，客户端和服务端都进入了ESTABLISHED状态，双方确认连接完成，可以进行数据的传输交互。

获取HTML文件

为了或许到HTML网页资源，客户端首先通过HTTP的GET请求向服务端请求HTML文件资源，我们截获了请求HTML文件的TC报文进行分析。

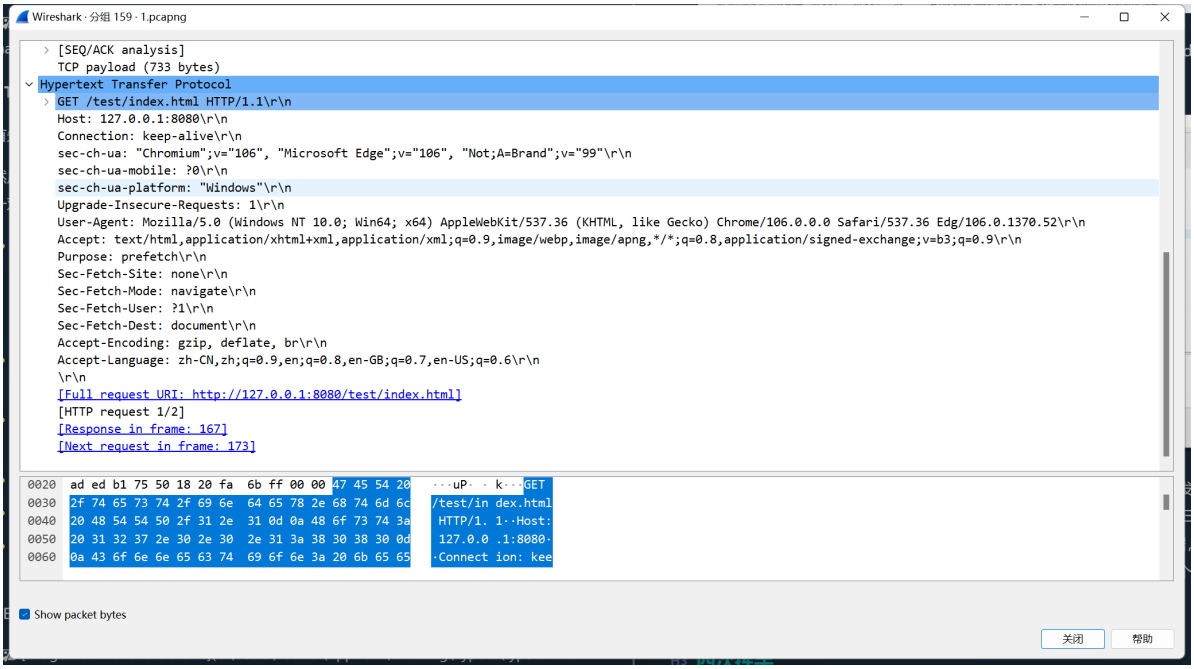


可以看到，浏览器先是发起了一次HTTP的GET请求，想服务器请求index.html文件。此时对于这条GET请求的报文进行分析。



从源端口号61567到目的端口号8080可以确认，请求是从浏览器发送给服务器的。FLAG中`PSH`表示接收端应该尽快将这条报文交付给上层应用。由于在第三次握手时，消耗了一个序号，因此本次的`Seq`为1，由于TCP报文段封装了HTTP报文数据，因此TCP载荷也就是数据段的值不再为0，可以看到数据段长度为733字节，因此下一个`Seq`的值就到了734。

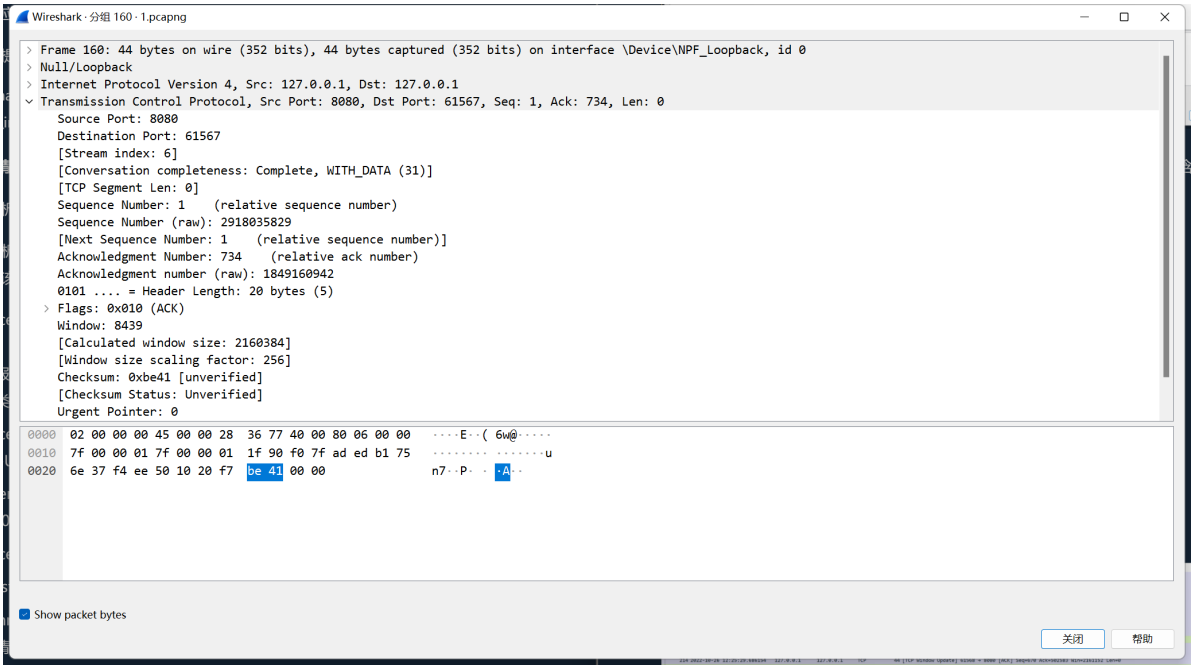
对于本条HTTP的GET请求，分析一下HTTP报文内容。



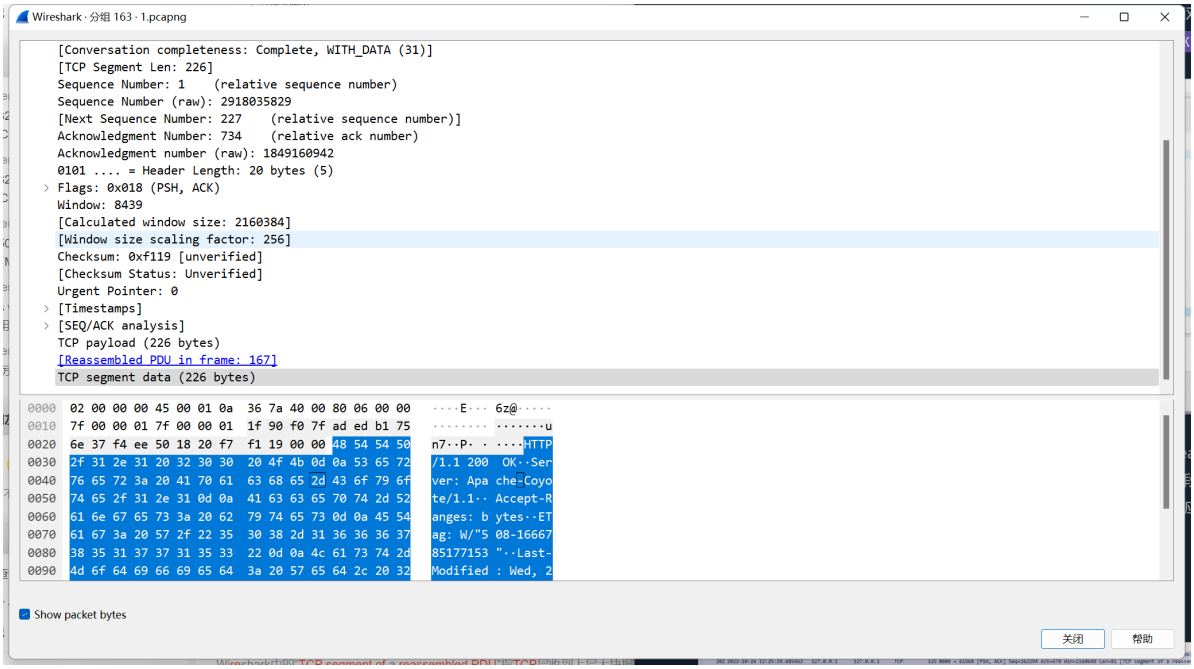
由于GET请求只有请求头，没有请求体，因此数据段长度为0。请求头中包含了

1. `GET`协议方法
2. `Host`请求的主机名，也就是服务器的地址
3. `Connection`连接方式，`keep-alive`表示长连接的方式。
4. `User-Agent`表示的客户端代理，也就是浏览器的类型
5. `Accept`表示客户端可以识别的响应内容。
6. `Accept-Encoding`表示客户端接受的编码格式
7. `Accept-Language`表示客户端接受的语言集

服务器在接收到了客户端发送的GET请求之后，会通过TCP回应一条确认报文。

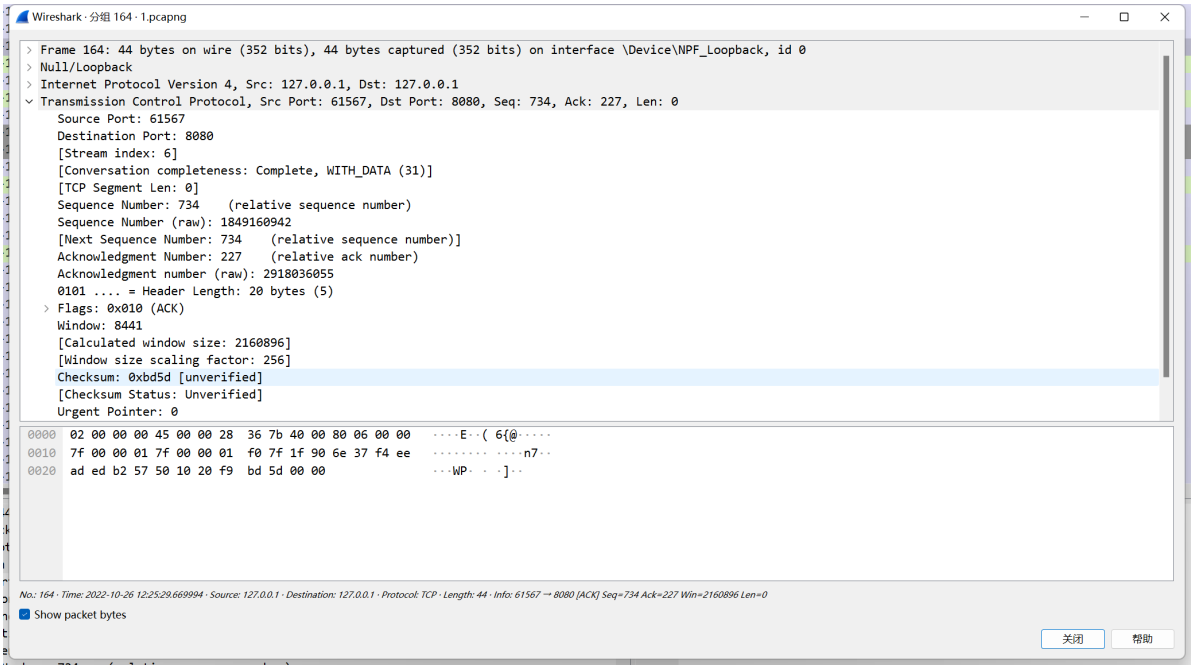


由于之前在第二次握手的时候服务端消耗了一个序号，因此本次的相对序号`Seq`为1，确认号`Ack`为请求报文的`Seq+Len+1`，此时的值为734。Flag中只包含`ACK`。此时注意到缓冲区的可用数据段减少了256*3，这是因为窗口大小为256，而接受的报文长度需要用3个窗口才能装下。



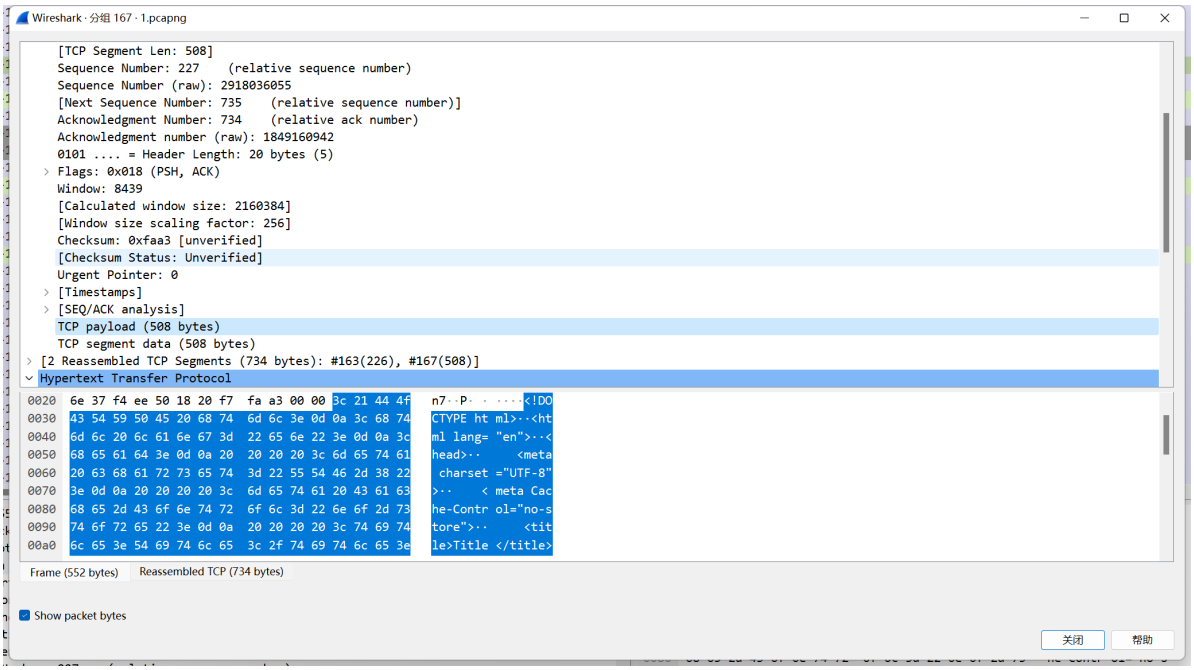
此时注意到服务端还想客户端发送了一条带有[TCP segment of a reassembled PDU]标签的报文，这个标签的意思是，一个完整的TCP报文被拆分成了不同的段发送，因此可以看到两条报文的`ACK`和`SEQ`是一样的，在第二条报文中包含了HTTP的响应信息。因此数据段长度不再为0，长度为226。这个响应信息中包含了HTTP的响应头中的数据。

客户端在收到服务器的响应之后，同样会给客户端发送确认报文。



可以看到，确认报文中的`Seq`值根据客户端上一次发送时的`Seq`值顺序递增，`ACK`值则等于服务端响应的`Seq+Len+1`。由于接收到了消息，所以接收端缓冲区剩余容量按照TCP报文长度对向上256取整递减。

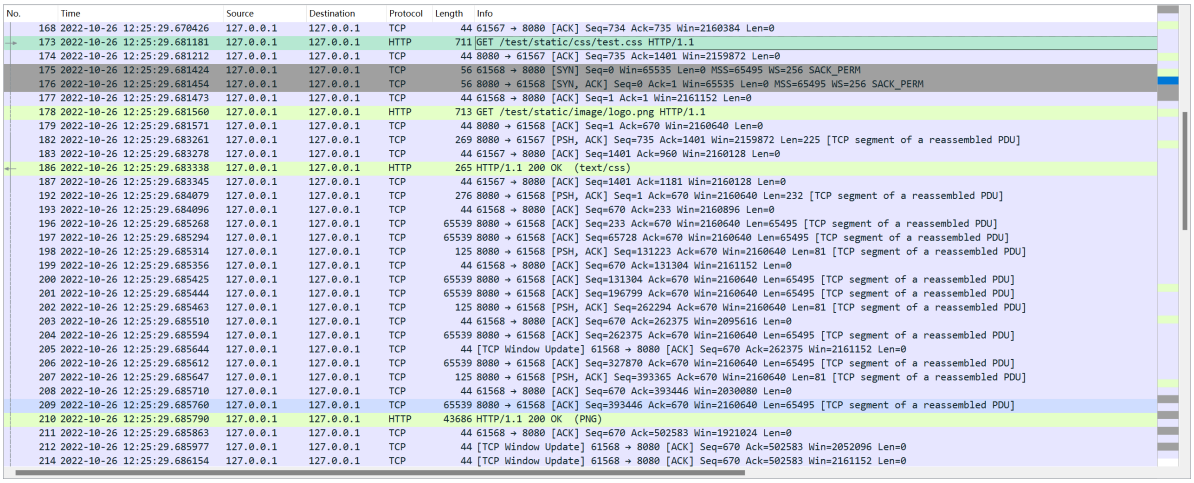
服务器在收到了客户端发送来的确认报文之后，会将HTML文件数据返回给客户端。



可以看到，`Seq`和`ACK`的计算规则并没有发生变化，TCP数据段承载了HTTP报文，在HTTP报文的数据段中包含了HTML页面的具体内容。

客户端在接收到服务端响应的数据后，会再发送一次确认报文，其中`Seq`和`ACK`以及`WIN`的计算方法和上面一致，不再赘述。

请求解析资源

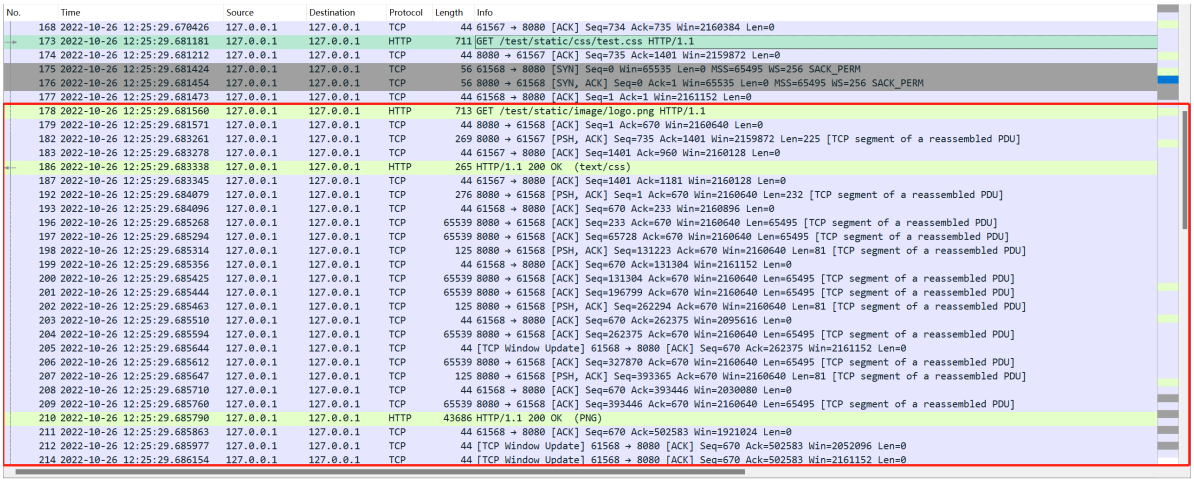


No.	Time	Source	Destination	Protocol	Length	Info
168	2022-10-26 12:25:29.670426	127.0.0.1	127.0.0.1	TCP	44	61567 → 8080 [ACK] Seq=734 Ack=735 Win=2160384 Len=0
173	2022-10-26 12:25:29.681181	127.0.0.1	127.0.0.1	HTTP	711	GET /test/static/css/test.css HTTP/1.1
174	2022-10-26 12:25:29.681212	127.0.0.1	127.0.0.1	TCP	44	8080 → 61567 [ACK] Seq=735 Ack=1401 Win=2159872 Len=0
175	2022-10-26 12:25:29.681424	127.0.0.1	127.0.0.1	TCP	56	61568 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
176	2022-10-26 12:25:29.681454	127.0.0.1	127.0.0.1	TCP	56	8080 → 61568 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
177	2022-10-26 12:25:29.681473	127.0.0.1	127.0.0.1	TCP	44	61568 → 8080 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
178	2022-10-26 12:25:29.681560	127.0.0.1	127.0.0.1	HTTP	713	GET /test/static/image/logo.png HTTP/1.1
179	2022-10-26 12:25:29.681571	127.0.0.1	127.0.0.1	TCP	44	8080 → 61568 [ACK] Seq=1 Ack=670 Win=2160640 Len=0
182	2022-10-26 12:25:29.683261	127.0.0.1	127.0.0.1	TCP	269	8080 → 61567 [PSH, ACK] Seq=735 Ack=1401 Win=2159872 Len=225 [TCP segment of a reassembled PDU]
183	2022-10-26 12:25:29.683278	127.0.0.1	127.0.0.1	TCP	44	61567 → 8080 [ACK] Seq=1401 Ack=960 Win=2160128 Len=0
186	2022-10-26 12:25:29.683338	127.0.0.1	127.0.0.1	HTTP	265	HTTP/1.1 200 OK (text/css)
187	2022-10-26 12:25:29.683345	127.0.0.1	127.0.0.1	TCP	44	61567 → 8080 [ACK] Seq=1401 Ack=1181 Win=2160128 Len=0
192	2022-10-26 12:25:29.684079	127.0.0.1	127.0.0.1	TCP	276	8080 → 61568 [PSH, ACK] Seq=1 Ack=670 Win=2160640 Len=232 [TCP segment of a reassembled PDU]
193	2022-10-26 12:25:29.684096	127.0.0.1	127.0.0.1	TCP	44	61568 → 8080 [ACK] Seq=670 Ack=233 Win=2160896 Len=0
196	2022-10-26 12:25:29.685268	127.0.0.1	127.0.0.1	TCP	65539	8080 → 61568 [ACK] Seq=233 Ack=670 Win=2160640 Len=65495 [TCP segment of a reassembled PDU]
197	2022-10-26 12:25:29.685294	127.0.0.1	127.0.0.1	TCP	65539	8080 → 61568 [ACK] Seq=65728 Ack=670 Win=2160640 Len=65495 [TCP segment of a reassembled PDU]
198	2022-10-26 12:25:29.685314	127.0.0.1	127.0.0.1	TCP	125	8080 → 61568 [PSH, ACK] Seq=131223 Ack=670 Win=2160640 Len=81 [TCP segment of a reassembled PDU]
199	2022-10-26 12:25:29.685356	127.0.0.1	127.0.0.1	TCP	44	61568 → 8080 [ACK] Seq=670 Ack=131304 Win=2161152 Len=0
200	2022-10-26 12:25:29.685425	127.0.0.1	127.0.0.1	TCP	65539	8080 → 61568 [ACK] Seq=131304 Ack=670 Win=2160640 Len=65495 [TCP segment of a reassembled PDU]
201	2022-10-26 12:25:29.685444	127.0.0.1	127.0.0.1	TCP	65539	8080 → 61568 [ACK] Seq=196799 Ack=670 Win=2160640 Len=65495 [TCP segment of a reassembled PDU]
202	2022-10-26 12:25:29.685463	127.0.0.1	127.0.0.1	TCP	125	8080 → 61568 [PSH, ACK] Seq=262294 Ack=670 Win=2160640 Len=81 [TCP segment of a reassembled PDU]
203	2022-10-26 12:25:29.685510	127.0.0.1	127.0.0.1	TCP	44	61568 → 8080 [ACK] Seq=670 Ack=262375 Win=2095616 Len=0
204	2022-10-26 12:25:29.685594	127.0.0.1	127.0.0.1	TCP	65539	8080 → 61568 [ACK] Seq=262375 Ack=670 Win=2160640 Len=65495 [TCP segment of a reassembled PDU]
205	2022-10-26 12:25:29.685644	127.0.0.1	127.0.0.1	TCP	44	[TCP Window Update] 61568 → 8080 [ACK] Seq=670 Ack=262375 Win=2161152 Len=0
206	2022-10-26 12:25:29.685612	127.0.0.1	127.0.0.1	TCP	65539	8080 → 61568 [ACK] Seq=327870 Ack=670 Win=2160640 Len=65495 [TCP segment of a reassembled PDU]
207	2022-10-26 12:25:29.685647	127.0.0.1	127.0.0.1	TCP	125	8080 → 61568 [PSH, ACK] Seq=393365 Ack=670 Win=2160640 Len=81 [TCP segment of a reassembled PDU]
208	2022-10-26 12:25:29.685710	127.0.0.1	127.0.0.1	TCP	44	61568 → 8080 [ACK] Seq=670 Ack=393446 Win=2030080 Len=0
209	2022-10-26 12:25:29.685760	127.0.0.1	127.0.0.1	TCP	65539	8080 → 61568 [ACK] Seq=393446 Ack=670 Win=2160640 Len=65495 [TCP segment of a reassembled PDU]
210	2022-10-26 12:25:29.685790	127.0.0.1	127.0.0.1	HTTP	43686	HTTP/1.1 200 OK (PNG)
211	2022-10-26 12:25:29.685863	127.0.0.1	127.0.0.1	TCP	44	61568 → 8080 [ACK] Seq=670 Ack=502583 Win=1921024 Len=0
212	2022-10-26 12:25:29.685977	127.0.0.1	127.0.0.1	TCP	44	[TCP Window Update] 61568 → 8080 [ACK] Seq=670 Ack=502583 Win=2052096 Len=0
214	2022-10-26 12:25:29.686154	127.0.0.1	127.0.0.1	TCP	44	[TCP Window Update] 61568 → 8080 [ACK] Seq=670 Ack=502583 Win=2161152 Len=0

由于在index.html页面中，引用了/static/css/test.css和/static/image/logo.png两个静态资源，因此客户端会再次通过HTTP协议中的GET方法和服务器请求相关的资源。

从报文中可以看到，在请求`test.css`样式文件的时候，整体的流程以及其中各数据段的变化情况和请求`index.html`时是完全一致的。但注意到，在请求`logo.png`之前，客户端又和服务器建立了一个TCP连接，完成了三次握手的过程。分析其原因是因为此时请求`test.css`的过程还没有完成，由于发生了拥塞，所以客户端又打开了一条TCP连接来减轻这种拥塞的影响，这个TCP连接用来负责请求`logo.png`。

可以注意到在请求`logo.png`的时候，报文的内容和之前有所区别，因此单独进行分析。新建立的用来请求`logo.png`的TCP在61568端口。



No.	Time	Source	Destination	Protocol	Length	Info
168	2022-10-26 12:25:29.670426	127.0.0.1	127.0.0.1	TCP	44	61567 → 8080 [ACK] Seq=734 Ack=735 Win=2160384 Len=0
173	2022-10-26 12:25:29.681181	127.0.0.1	127.0.0.1	HTTP	711	GET /test/static/css/test.css HTTP/1.1
174	2022-10-26 12:25:29.681212	127.0.0.1	127.0.0.1	TCP	44	8080 → 61567 [ACK] Seq=735 Ack=1401 Win=2159872 Len=0
175	2022-10-26 12:25:29.681424	127.0.0.1	127.0.0.1	TCP	56	61568 → 8080 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
176	2022-10-26 12:25:29.681454	127.0.0.1	127.0.0.1	TCP	56	8080 → 61568 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
177	2022-10-26 12:25:29.681473	127.0.0.1	127.0.0.1	TCP	44	61568 → 8080 [ACK] Seq=1 Ack=1 Win=2161152 Len=0
178	2022-10-26 12:25:29.681560	127.0.0.1	127.0.0.1	HTTP	713	GET /test/static/image/logo.png HTTP/1.1
179	2022-10-26 12:25:29.681571	127.0.0.1	127.0.0.1	TCP	44	8080 → 61568 [ACK] Seq=1 Ack=670 Win=2160640 Len=0
182	2022-10-26 12:25:29.683261	127.0.0.1	127.0.0.1	TCP	269	8080 → 61567 [PSH, ACK] Seq=735 Ack=1401 Win=2159872 Len=225 [TCP segment of a reassembled PDU]
183	2022-10-26 12:25:29.683278	127.0.0.1	127.0.0.1	TCP	44	61567 → 8080 [ACK] Seq=1401 Ack=960 Win=2160128 Len=0
186	2022-10-26 12:25:29.683338	127.0.0.1	127.0.0.1	HTTP	265	HTTP/1.1 200 OK (text/css)
187	2022-10-26 12:25:29.683345	127.0.0.1	127.0.0.1	TCP	44	61567 → 8080 [ACK] Seq=1401 Ack=1181 Win=2160128 Len=0
192	2022-10-26 12:25:29.684079	127.0.0.1	127.0.0.1	TCP	276	8080 → 61568 [PSH, ACK] Seq=1 Ack=670 Win=2160640 Len=232 [TCP segment of a reassembled PDU]
193	2022-10-26 12:25:29.684096	127.0.0.1	127.0.0.1	TCP	44	61568 → 8080 [ACK] Seq=670 Ack=233 Win=2160896 Len=0
196	2022-10-26 12:25:29.685268	127.0.0.1	127.0.0.1	TCP	65539	8080 → 61568 [ACK] Seq=233 Ack=670 Win=2160640 Len=65495 [TCP segment of a reassembled PDU]
197	2022-10-26 12:25:29.685294	127.0.0.1	127.0.0.1	TCP	65539	8080 → 61568 [ACK] Seq=65728 Ack=670 Win=2160640 Len=65495 [TCP segment of a reassembled PDU]
198	2022-10-26 12:25:29.685314	127.0.0.1	127.0.0.1	TCP	125	8080 → 61568 [PSH, ACK] Seq=131223 Ack=670 Win=2160640 Len=81 [TCP segment of a reassembled PDU]
199	2022-10-26 12:25:29.685356	127.0.0.1	127.0.0.1	TCP	44	61568 → 8080 [ACK] Seq=670 Ack=131304 Win=2161152 Len=0
200	2022-10-26 12:25:29.685425	127.0.0.1	127.0.0.1	TCP	65539	8080 → 61568 [ACK] Seq=131304 Ack=670 Win=2160640 Len=65495 [TCP segment of a reassembled PDU]
201	2022-10-26 12:25:29.685444	127.0.0.1	127.0.0.1	TCP	65539	8080 → 61568 [ACK] Seq=196799 Ack=670 Win=2160640 Len=65495 [TCP segment of a reassembled PDU]
202	2022-10-26 12:25:29.685463	127.0.0.1	127.0.0.1	TCP	125	8080 → 61568 [PSH, ACK] Seq=262294 Ack=670 Win=2160640 Len=81 [TCP segment of a reassembled PDU]
203	2022-10-26 12:25:29.685510	127.0.0.1	127.0.0.1	TCP	44	61568 → 8080 [ACK] Seq=670 Ack=262375 Win=2095616 Len=0
204	2022-10-26 12:25:29.685594	127.0.0.1	127.0.0.1	TCP	65539	8080 → 61568 [ACK] Seq=262375 Ack=670 Win=2160640 Len=65495 [TCP segment of a reassembled PDU]
205	2022-10-26 12:25:29.685644	127.0.0.1	127.0.0.1	TCP	44	[TCP Window Update] 61568 → 8080 [ACK] Seq=670 Ack=262375 Win=2161152 Len=0
206	2022-10-26 12:25:29.685612	127.0.0.1	127.0.0.1	TCP	65539	8080 → 61568 [ACK] Seq=327870 Ack=670 Win=2160640 Len=65495 [TCP segment of a reassembled PDU]
207	2022-10-26 12:25:29.685647	127.0.0.1	127.0.0.1	TCP	125	8080 → 61568 [PSH, ACK] Seq=393365 Ack=670 Win=2160640 Len=81 [TCP segment of a reassembled PDU]
208	2022-10-26 12:25:29.685710	127.0.0.1	127.0.0.1	TCP	44	61568 → 8080 [ACK] Seq=670 Ack=393446 Win=2030080 Len=0
209	2022-10-26 12:25:29.685760	127.0.0.1	127.0.0.1	TCP	65539	8080 → 61568 [ACK] Seq=393446 Ack=670 Win=2160640 Len=65495 [TCP segment of a reassembled PDU]
210	2022-10-26 12:25:29.685790	127.0.0.1	127.0.0.1	HTTP	43686	HTTP/1.1 200 OK (PNG)
211	2022-10-26 12:25:29.685863	127.0.0.1	127.0.0.1	TCP	44	61568 → 8080 [ACK] Seq=670 Ack=502583 Win=1921024 Len=0
212	2022-10-26 12:25:29.685977	127.0.0.1	127.0.0.1	TCP	44	[TCP Window Update] 61568 → 8080 [ACK] Seq=670 Ack=502583 Win=2052096 Len=0
214	2022-10-26 12:25:29.686154	127.0.0.1	127.0.0.1	TCP	44	[TCP Window Update] 61568 → 8080 [ACK] Seq=670 Ack=502583 Win=2161152 Len=0

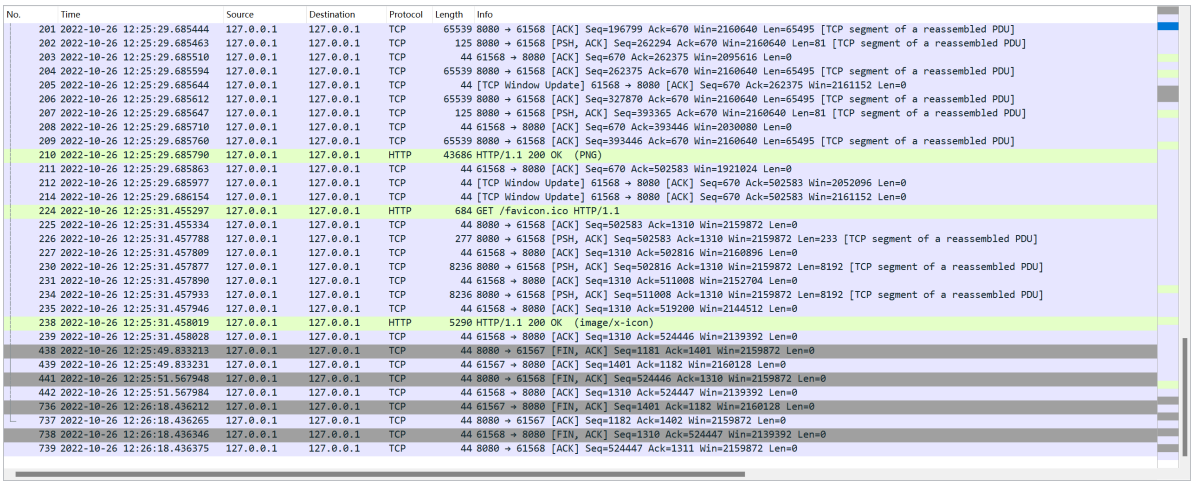
在正常和服务器完成了GET方法的确认之后，服务端开始向客户端传输数据。通过对报文的分析，可以推测，服务端是通过分别发送RGB三个通道的方式来传输图片的。又由于每一个通道中的数据量比较大，超过了`MSS`，所以一个完整的TCP报文被拆分成了三段分别进行传输。在客户端这边采用了延迟相应的方式，接收到一个完整的TCP报文之后才会给服务端发送`ACK`确认报文。由于单独发送确认报文并不会消耗序号，因此可以看到在接收图像的过程中，客户端这边的`Seq`值一直为670。

但是通过接收数据可以发现，剩余缓冲区的接受容量在减少，由于第一段的数据大小为65495，对256取整之后的大小为65536，因此前后两条报文所剩的缓冲区大小相差65536，并且Window的差值也为256。

```
Window: 8186
[Calculated window size: 2095616]
[Window size scaling factor: 256]

Window: 7930
[Calculated window size: 2030080]
[Window size scaling factor: 256]
```

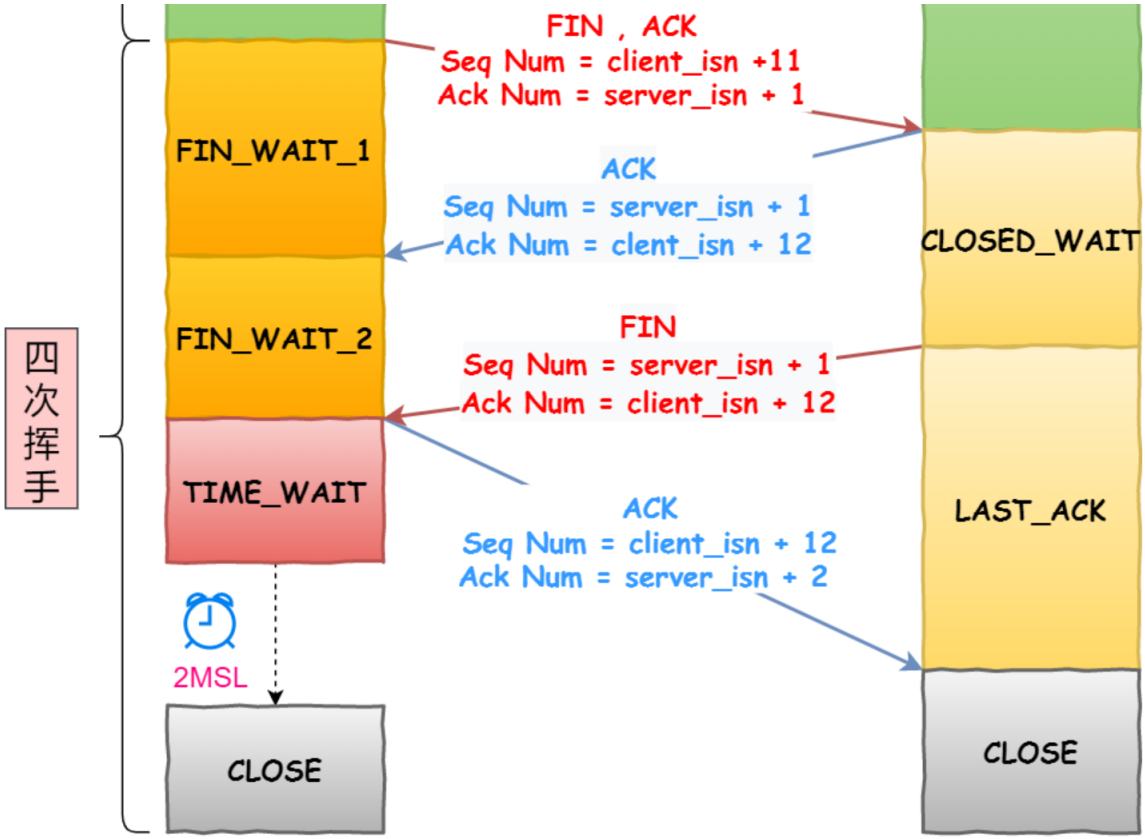
同样注意到在传输图片的过程中出现了几次[TCP Window Update]，这里是重新更新了窗口缓冲区，并以256的整数倍增加window。

四次挥手



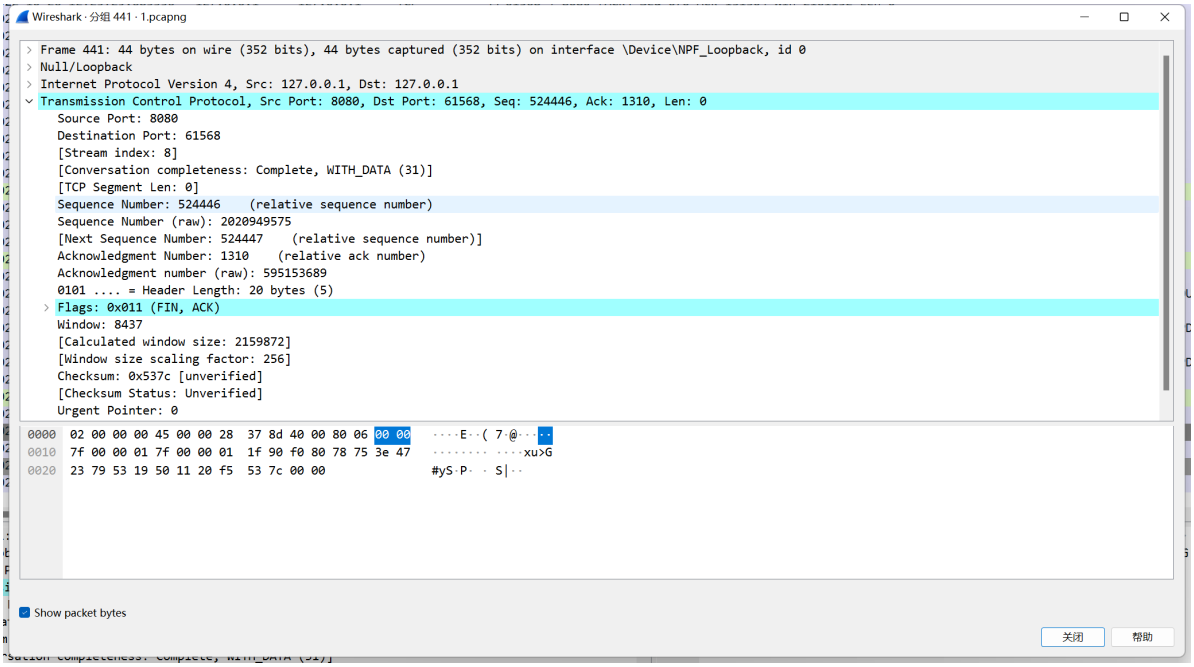
No.	Time	Source	Destination	Protocol	Length	Info
201	2022-10-26 12:25:29.685444	127.0.0.1	127.0.0.1	TCP	65539	8080 → 61568 [ACK] Seq=196799 Ack=670 Win=2160640 Len=65495 [TCP segment of a reassembled PDU]
202	2022-10-26 12:25:29.685463	127.0.0.1	127.0.0.1	TCP	125	8080 → 61568 [PSH, ACK] Seq=262294 Ack=670 Win=2160640 Len=81 [TCP segment of a reassembled PDU]
203	2022-10-26 12:25:29.685510	127.0.0.1	127.0.0.1	TCP	44	61568 → 8080 [ACK] Seq=670 Ack=262375 Win=2095616 Len=0
204	2022-10-26 12:25:29.685594	127.0.0.1	127.0.0.1	TCP	65539	8080 → 61568 [ACK] Seq=262375 Ack=670 Win=2160640 Len=65495 [TCP segment of a reassembled PDU]
205	2022-10-26 12:25:29.685644	127.0.0.1	127.0.0.1	TCP	44	[TCP Window Update] 61568 → 8080 [ACK] Seq=670 Ack=262375 Win=2161152 Len=0
206	2022-10-26 12:25:29.685612	127.0.0.1	127.0.0.1	TCP	65539	8080 → 61568 [ACK] Seq=327870 Ack=670 Win=2160640 Len=65495 [TCP segment of a reassembled PDU]
207	2022-10-26 12:25:29.685647	127.0.0.1	127.0.0.1	TCP	125	8080 → 61568 [PSH, ACK] Seq=393365 Ack=670 Win=2160640 Len=81 [TCP segment of a reassembled PDU]
208	2022-10-26 12:25:29.685710	127.0.0.1	127.0.0.1	TCP	44	61568 → 8080 [ACK] Seq=670 Ack=393446 Win=2030080 Len=0
209	2022-10-26 12:25:29.685760	127.0.0.1	127.0.0.1	TCP	65539	8080 → 61568 [ACK] Seq=393446 Ack=670 Win=2160640 Len=65495 [TCP segment of a reassembled PDU]
210	2022-10-26 12:25:29.685790	127.0.0.1	127.0.0.1	HTTP	43686	HTTP/1.1 200 OK (PNG)
211	2022-10-26 12:25:29.685863	127.0.0.1	127.0.0.1	TCP	44	61568 → 8080 [ACK] Seq=670 Ack=502583 Win=1921024 Len=0
212	2022-10-26 12:25:29.685977	127.0.0.1	127.0.0.1	TCP	44	[TCP Window Update] 61568 → 8080 [ACK] Seq=670 Ack=502583 Win=2052096 Len=0
214	2022-10-26 12:25:29.686154	127.0.0.1	127.0.0.1	TCP	44	[TCP Window Update] 61568 → 8080 [ACK] Seq=670 Ack=502583 Win=2161152 Len=0
224	2022-10-26 12:25:31.455297	127.0.0.1	127.0.0.1	HTTP	684	GET /favicon.ico HTTP/1.1
225	2022-10-26 12:25:31.455334	127.0.0.1	127.0.0.1	TCP	44	8080 → 61568 [ACK] Seq=502583 Ack=1310 Win=2159872 Len=0
226	2022-10-26 12:25:31.457788	127.0.0.1	127.0.0.1	TCP	277	8080 → 61568 [PSH, ACK] Seq=502583 Ack=1310 Win=2159872 Len=233 [TCP segment of a reassembled PDU]
227	2022-10-26 12:25:31.457809	127.0.0.1	127.0.0.1	TCP	44	61568 → 8080 [ACK] Seq=1310 Ack=502816 Win=2160896 Len=0
230	2022-10-26 12:25:31.457877	127.0.0.1	127.0.0.1	TCP	8236	8080 → 61568 [PSH, ACK] Seq=502816 Ack=1310 Win=2159872 Len=8192 [TCP segment of a reassembled PDU]
231	2022-10-26 12:25:31.457890	127.0.0.1	127.0.0.1	TCP	44	61568 → 8080 [ACK] Seq=1310 Ack=511008 Win=2152704 Len=0
234	2022-10-26 12:25:31.457933	127.0.0.1	127.0.0.1	TCP	8236	8080 → 61568 [PSH, ACK] Seq=511008 Ack=1310 Win=2159872 Len=8192 [TCP segment of a reassembled PDU]
235	2022-10-26 12:25:31.457946	127.0.0.1	127.0.0.1	TCP	44	61568 → 8080 [ACK] Seq=1310 Ack=519200 Win=2144512 Len=0
238	2022-10-26 12:25:31.458019	127.0.0.1	127.0.0.1	HTTP	5290	HTTP/1.1 200 OK (image/x-icon)
239	2022-10-26 12:25:31.458028	127.0.0.1	127.0.0.1	TCP	44	61568 → 8080 [ACK] Seq=1310 Ack=524446 Win=2139392 Len=0
438	2022-10-26 12:25:49.833213	127.0.0.1	127.0.0.1	TCP	44	8080 → 61567 [FIN, ACK] Seq=1181 Ack=1401 Win=2159872 Len=0
439	2022-10-26 12:25:49.833231	127.0.0.1	127.0.0.1	TCP	44	61567 → 8080 [ACK] Seq=1401 Ack=1182 Win=2160128 Len=0
441	2022-10-26 12:25:51.567948	127.0.0.1	127.0.0.1	TCP	44	8080 → 61568 [FIN, ACK] Seq=524446 Ack=1310 Win=2159872 Len=0
442	2022-10-26 12:25:51.567984	127.0.0.1	127.0.0.1	TCP	44	61568 → 8080 [ACK] Seq=1310 Ack=524447 Win=2139392 Len=0
736	2022-10-26 12:26:18.436212	127.0.0.1	127.0.0.1	TCP	44	61567 → 8080 [FIN, ACK] Seq=1401 Ack=1182 Win=2160128 Len=0
737	2022-10-26 12:26:18.436265	127.0.0.1	127.0.0.1	TCP	44	8080 → 61567 [ACK] Seq=1182 Ack=1402 Win=2159872 Len=0
738	2022-10-26 12:26:18.436346	127.0.0.1	127.0.0.1	TCP	44	61568 → 8080 [FIN, ACK] Seq=1310 Ack=524447 Win=2139392 Len=0
739	2022-10-26 12:26:18.436375	127.0.0.1	127.0.0.1	TCP	44	8080 → 61568 [ACK] Seq=524447 Ack=1311 Win=2159872 Len=0

可以看到，当整个页面加载完成一段时间之后，客户端就会主动和服务端断开TCP连接，释放占用的资源，避免长时间占用连接导致服务端资源紧张。



第一次挥手

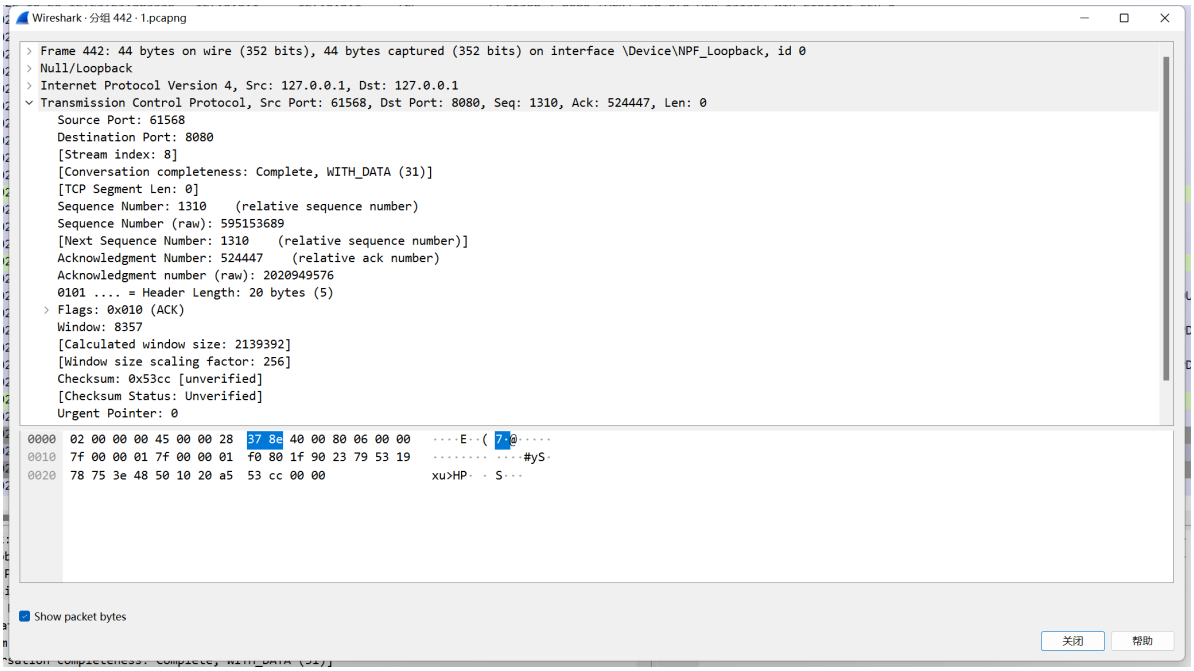
在客户端获取到全部所需的资源后，服务器向客户端发起了断开连接的请求。



该报文的FLAG字段包含`FIN`和`ACK`，随后服务器进入了FIN_WATI_1阶段，等待客户端接收到报文后的响应报文。由于上次一51568和8080端口通信后的`Seq`为1310，`Ack`为524446，因此第一次挥手的时候，服务端8080给客户端51568发送的TCP报文中，`Seq`为上一个报文的`Ack`，值为52446；`Ack`的值为上一个报文的`Seq`，值为1310。

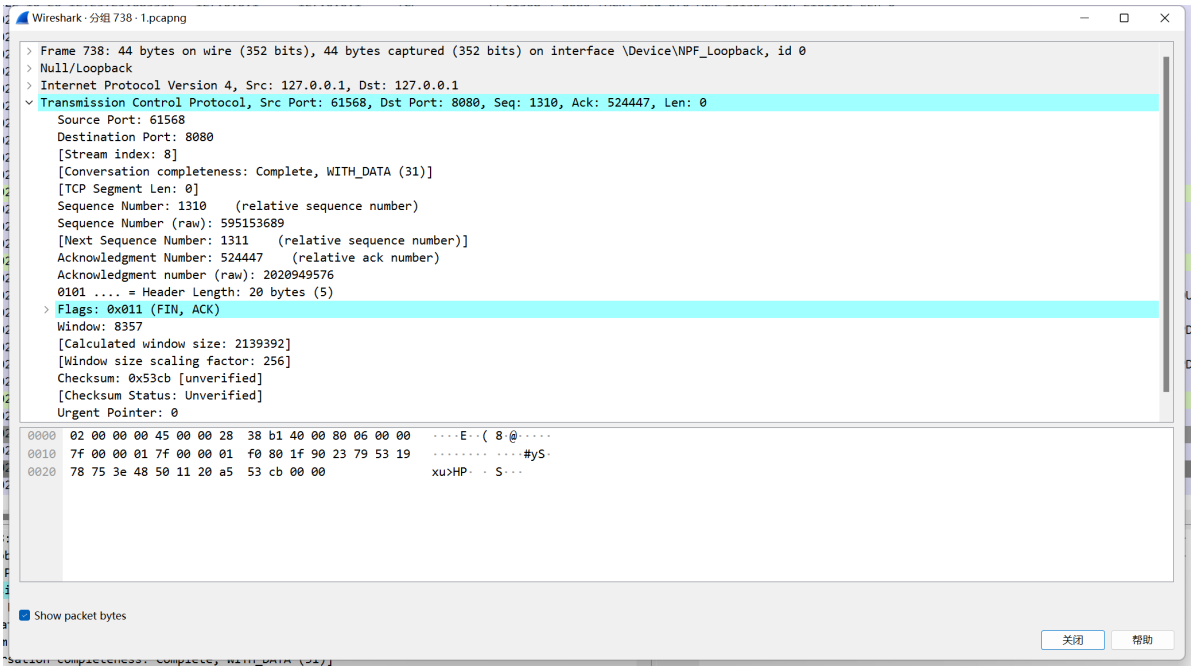
第二次挥手

客户端在接收到服务器断开连接的请求之后，会回复一条确认报文，其中FLAG字段标志位只有`ACK`有效，`Seq`的值为第一次握手中的`Ack`值，`Ack`的值为第一次握手中的`Seq+1`。



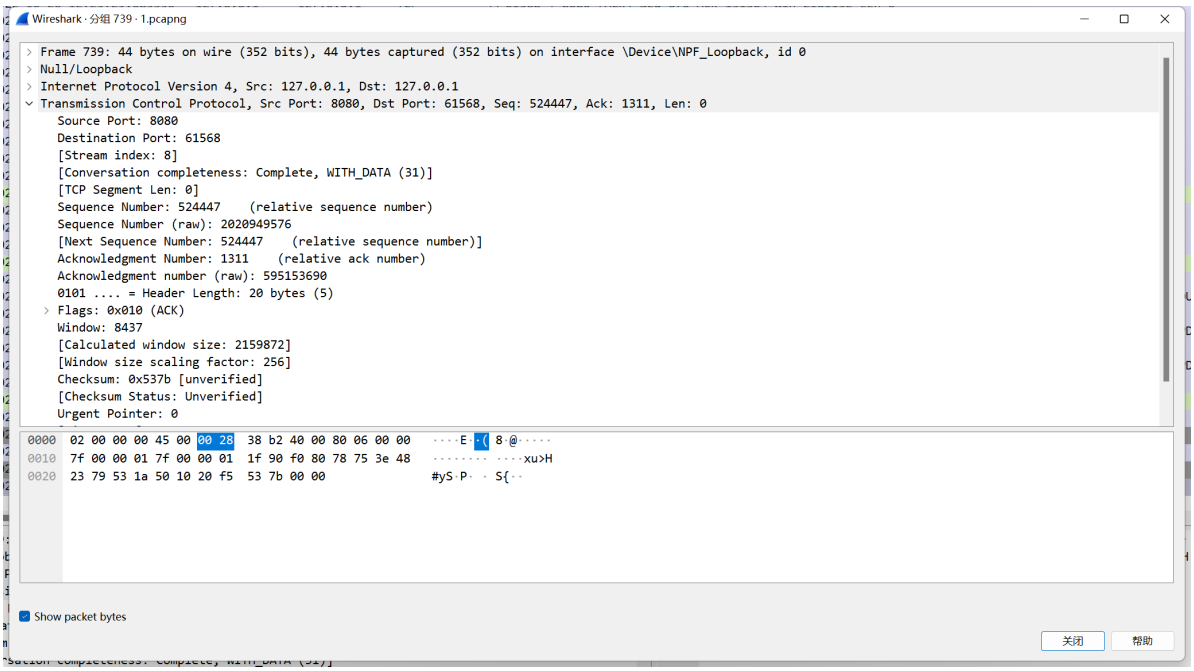
第三次握手

当客户端关闭浏览器的时候，浏览器会向服务器发送一次断开连接的报文。其中FLAG字段中`FIN`和`ACK`标志位有效。报文中的`Seq`字段和第二次握手中的`Ack`字段的值保持一致，`Ack`字段和第二次握手中的`Seq`字段的值保持一致。



第四次握手

服务端在接收到客户端断开连接的请求之后，会回复一条确认报文，其中FLAG字段标志位只有`ACK`有效，`Seq`的值为第三次握手中的`Ack`值，`Ack`的值为第三次握手中的`Seq+1`。


Wireshark · 分组 739 · 1.pcapng
Frame 739: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface \Device\NPF_Loopback, id 0
Null/Loopback
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 8080, Dst Port: 61568, Seq: 524447, Ack: 1311, Len: 0
Source Port: 8080
Destination Port: 61568
[Stream index: 8]
[Conversation completeness: Complete, WITH_DATA (31)]
[TCP Segment Len: 0]
Sequence Number: 524447 (relative sequence number)
Sequence Number (raw): 2020949576
[Next Sequence Number: 524447 (relative sequence number)]
Acknowledgment Number: 1311 (relative ack number)
Acknowledgment number (raw): 595153690
0101 .... = Header Length: 20 bytes (5)
Flags: 0x010 (ACK)
Window: 8437
[Calculated window size: 2159872]
[Window size scaling factor: 256]
Checksum: 0x537b [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
0000 02 00 00 00 45 00 00 28 38 b2 40 00 80 06 00 00
0010 7f 00 00 01 7f 00 00 01 1f 90 f0 80 78 75 3e 48
0020 23 79 53 1a 50 10 20 f5 53 7b 00 00
Show packet bytes
关闭
帮助