**Lab2实验报告**

姓名：吴静

专业：信息安全

学号：2113285

# 一、实验要求

1.搭建Web服务器(自由选择系统)，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)、自己的LOGO、自我介绍的音频信息。页面不要太复杂，包含要求的基本信息即可。

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.

3.使用HTTP，不要使用HTTPS。

4.提交实验报告

# 二、实验步骤

1.搭建web服务器

使用平台：xampp

编写简单的html代码start.html，代码如下：

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>计网第二次作业</title>

</head>

<body>

    <h1>计网第二次作业</h1>

<p>姓名：吴静</p>

<p>学号：2113285</p>

<p>专业：信息安全</p>

<img src="./wanderer.jpg" alt="小圆帽" width="304" height="304">

<p>introduction</p>

<audio controls>

    <source src="./introduction.mp3" type="audio/mp3">

</audio>

</body>

</html>

打开xampp，然后在web服务器里面输入“localhost:8081/start.html”即可出现页面，截图如下：

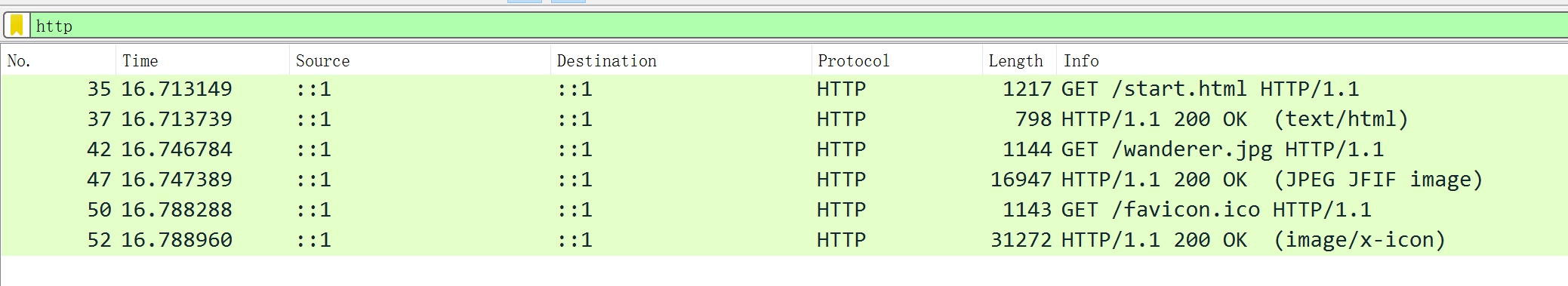


2.捕获交互过程

首先打开wireshark软件，由于客户端和服务器都是主机，所以选择loopback进行数据包的捕获。

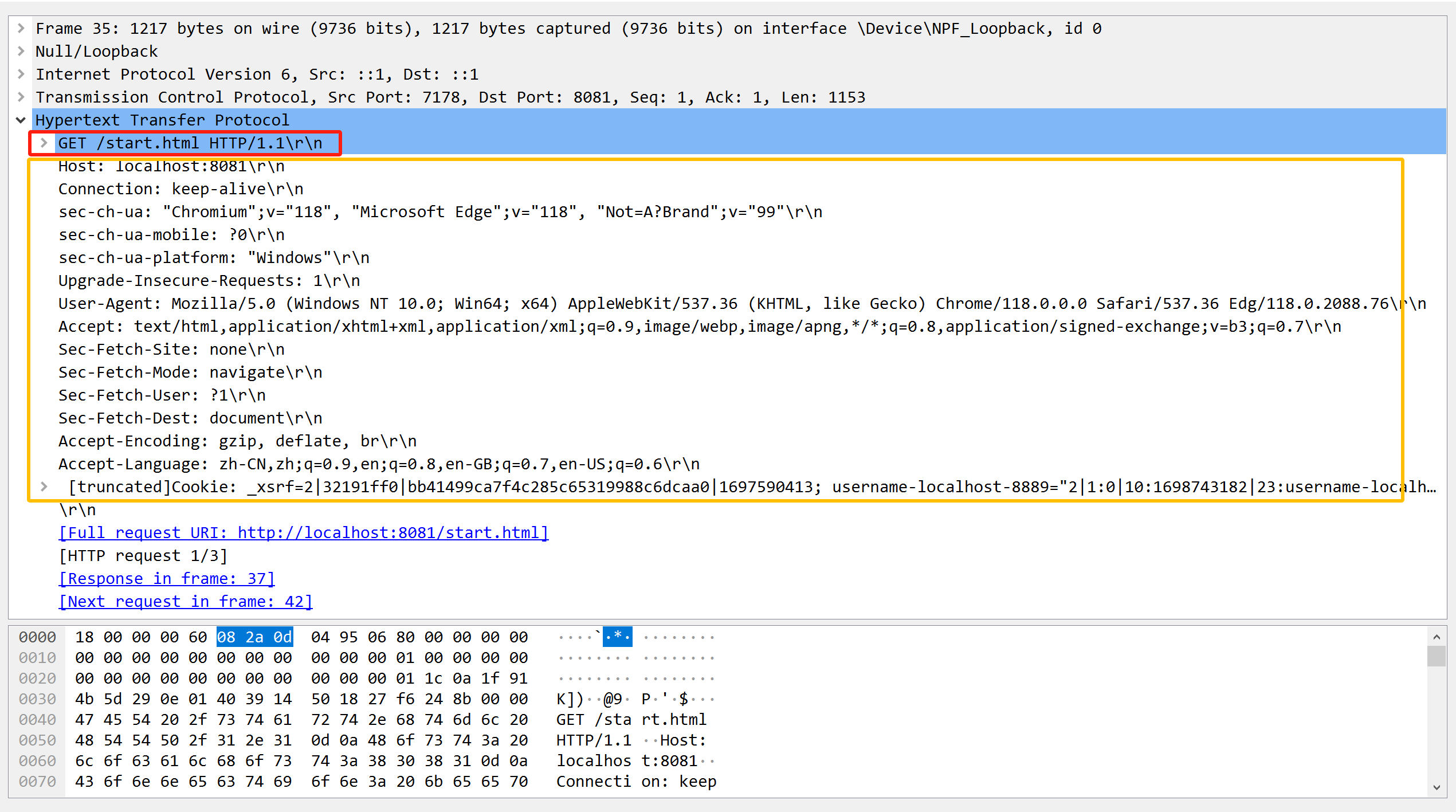
然后在网页中打开对应的网址，在wireshark中设置过滤规则为“http”，表示查看捕获到的http数据包，接下来进行wireshark捕获到的数据包的分析。

3.数据包分析——http相关数据包



这是一些HTTP请求和响应的示例，以第一二句为例，首先点进“GET /start.html HTTP/1.1”查看，查询有关该请求报文的内容：

（1）HTTP请求报文



http请求报文分成三个部分：请求行（request line），请求头部（head），空行和请求数据四个部分。

**请求行**即图中的红色部分，由请求方法字段，URL字段和HTTP协议版本字段3个字段组成，由空格分隔。客户端发起了一个GET请求，请求的URL是/start.html，而且使用的是HTTP/1.1协议版本。

接下来的部分是**请求头部**，即图中的黄色部分。请求头部由关键字/值对组成，每行一对（由”\r\n”结尾）关键字和值用英文冒号”:”分隔，他通知了服务器有关客户端的请求的信息。举例如下：

host表示请求的主机名；

connection表示连接方式，keep-alive指示连接应保持活动；

Sec-Ch-UA字段通常包括浏览器的名称和版本以及其他特定的客户端信息；

Accept字段用于指定客户端可以接受的响应内容类型，包括多个内容类型，按优先级排序；

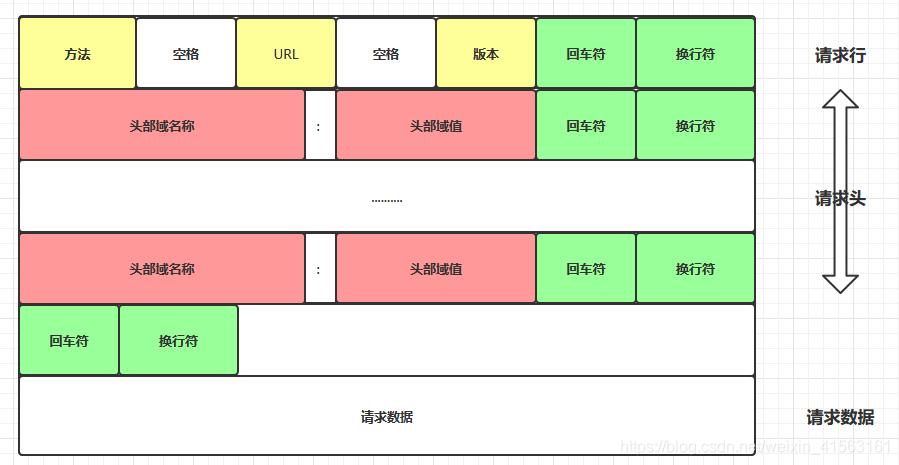
User-Agent：产生请求的浏览器类型;

Cookie：存储于客户端扩展字段，向同一域名的服务端发送属于该域的cookie;

可以看到，请求头部后面跟着一个小**空行**表示以下不再有请求头部发内容。

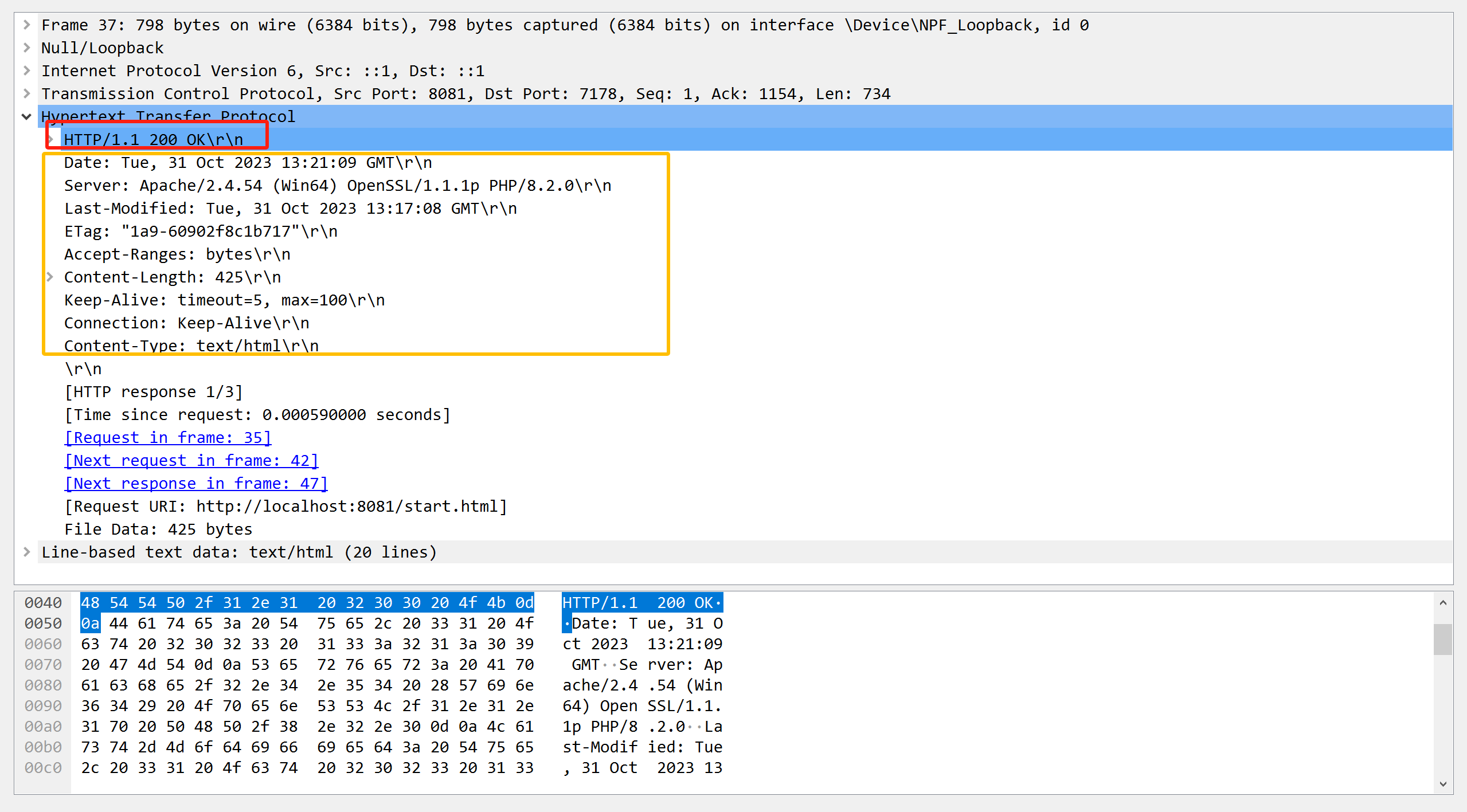
后面没有**请求体**的部分，因为此时的请求方法是GET，没有请求体部分。

结构如下：



（2）HTTP响应报文

例如图中的“HTTP/1.1 200 OK (text/html) ”字段，同样点进去查看。



响应报文同样由四个部分组成：响应行，响应头，空行和响应数据。

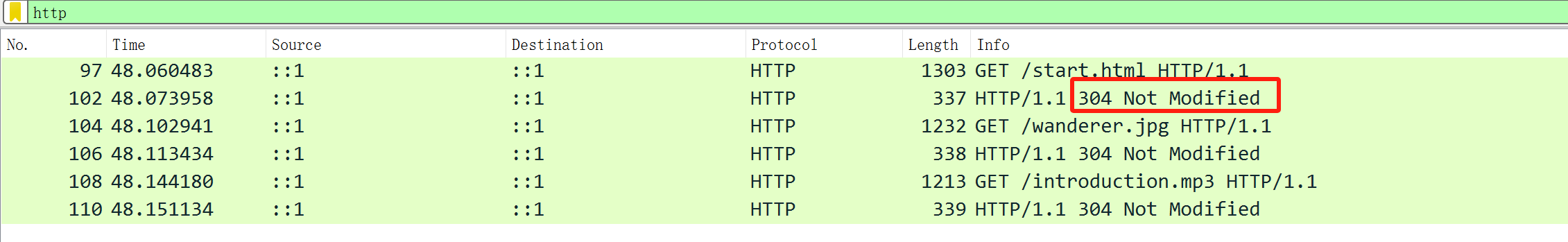
**响应行**，即图中的红色部分由三个部分组成：服务器HTTP协议版本，响应状态码，状态码的文本描述。经过查阅得知，响应状态码有五种取值：“1xx”表示请求已接受，“2xx”表示请求被成功接收；“3xx”表示发生了重定向，“4xx”表示客户端错误，“5xx”表示服务器端错误；常见的有：

响应状态码：“200”，文本描述：“OK”，表示客户端请求成功。

响应状态码：“304”，文本描述：“Not Modified”，表示资源未被修改，客户端可以使用缓存的版本。

响应状态码：“404”，文本描述：“Not Found”，表示资源不存在。

下图中即为未清除缓存时出现304响应的情况。



图中的黄色部分为**响应头**。也是由多个属性构成，举例如下：

Date表示响应时间；

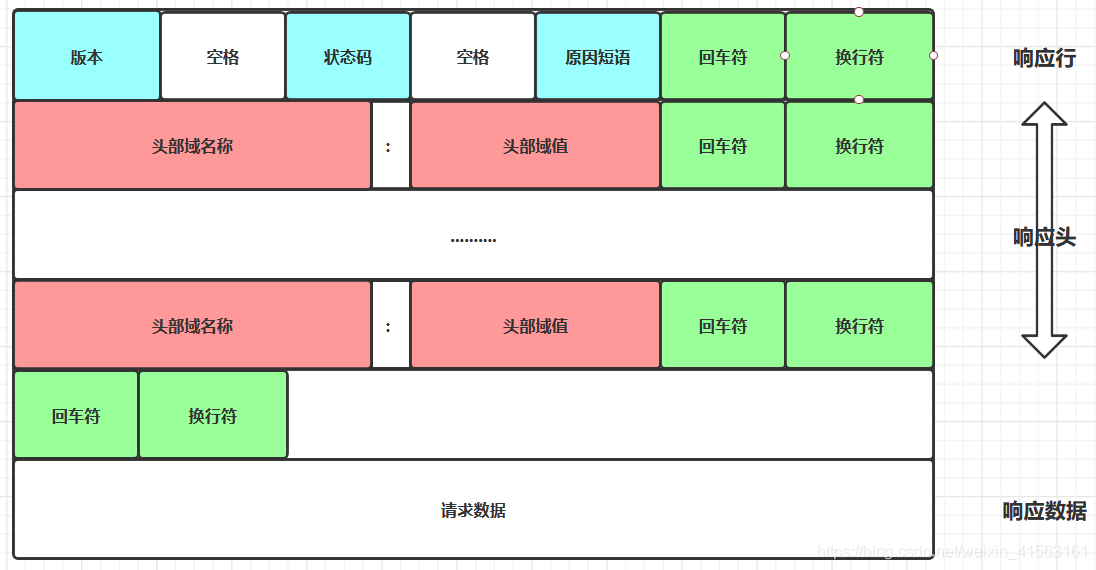
Server表示服务器应用程序软件的名称和版本；

Accept-Ranges表示对此资源来说，服务器可接受的范围类型；

Content-Length表示实体的长度，指示了客户端应该期望接收多少字节的响应数据；

Content-Tyep表示实体的媒体类型；

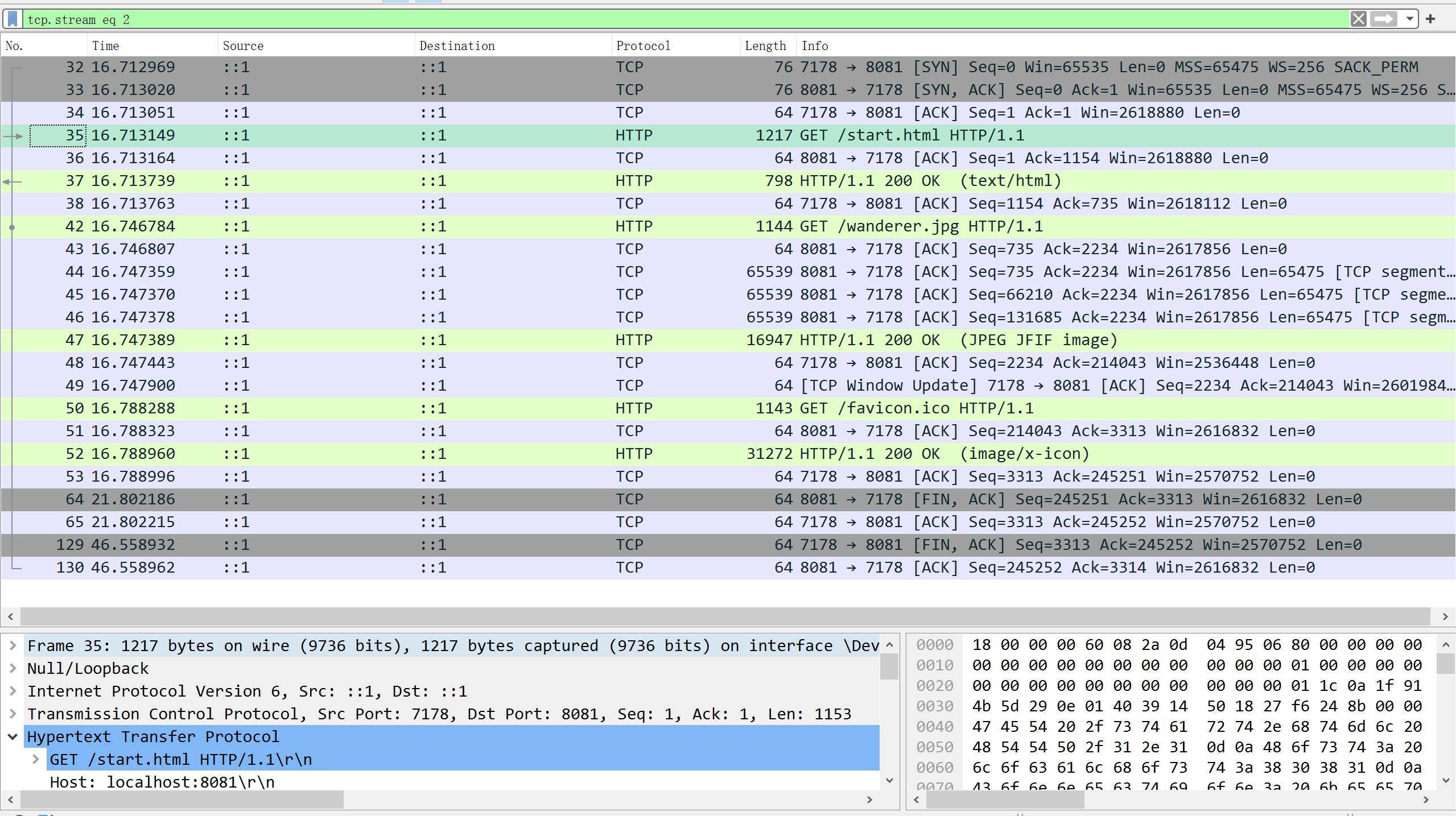
格式如下：



这里注意”\r\n”后的部分不再是响应报文中的部分，二是wireshark列出来帮助解释和分析捕获的数据包，比如[HTTP response 1/3]表示总共有3个HTTP响应，这是第一个响应；[Request in frame: 35]：这表明HTTP请求在捕获的数据包的第35帧中。

4.传输过程分析——传输控制协议TCP

首先随机选择一个HTTP响应/请求报文，右键选中“追踪流”，选中追踪TCP流，得到如下界面：

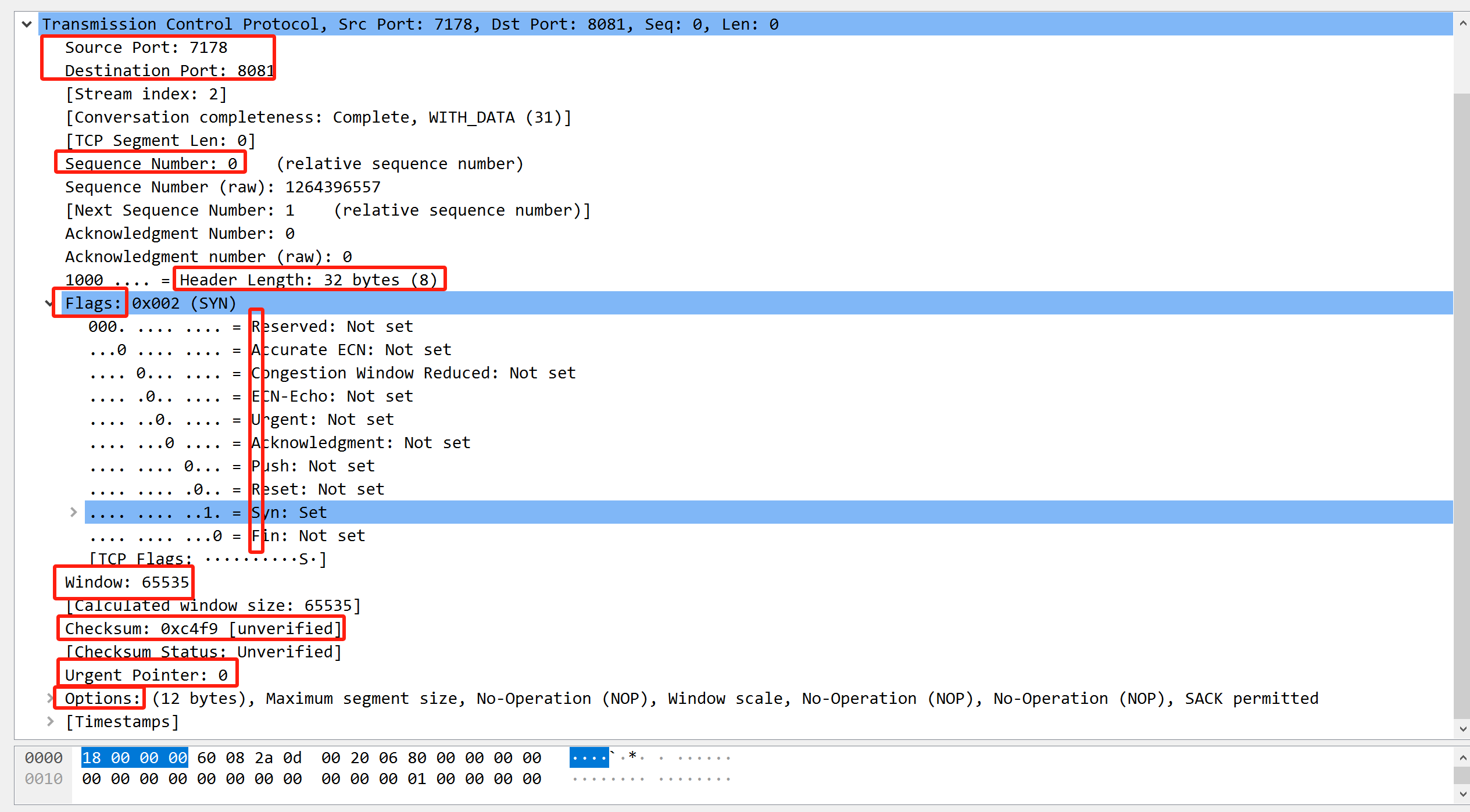


我们知道，TCP是面向连接的服务，在传送数据之前必须先建立连接，数据传送完后要释放连接，而在建立连接的时候要经过三次握手的过程，释放连接的时候要经过四次挥手，这一个部分，我讲重点分析这三次握手和四次挥手的过程。

（1）TCP报文格式

首先先来看TCP报文格式：

同样随机选择一个TCP报文，点进去查看：



Source Port：源端口

Destination Port：目的端口

Sequence Number：发送序号，TCP字节流中的每个字节都按顺序进行编号，这里从0开始是因为这是使用的相对的编号。

Header Length：头长度

Flags：其中有6个bit的保留位，以及“U”“A”“P”“R”“S”“F”这六个保留位。

①Urgent表示紧急指针字段是否有效，如果置位，则表示他告诉系统此报文中有紧急数据，需要尽快传送；

②ACK为确认位，仅当ACK置位时确认号字段ack才有效，注意在连接建立后，TCP规定会把所有的报文段的ACK都置1，由于采用累计确认的方式，ack通常等于上一个数据包的序列号的值加1；

③RST表示复位，表示TCP连接中出现严重差错需要释放连接在重新建立运输连接；

④SYM为同步标志；

⑤最后的FIN表示终止，用来释放一个连接。

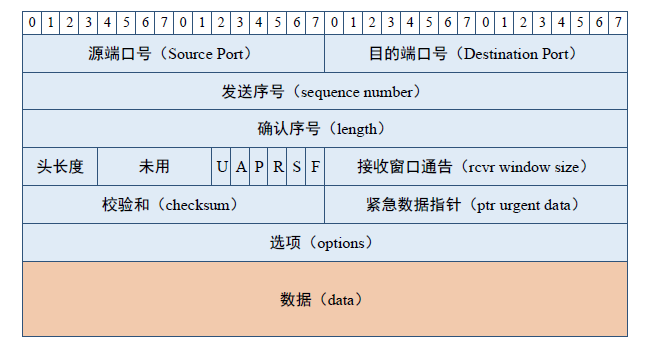
Window：接收窗口通告，即告诉接收方，发送本报文需要多大的空间来接受。

Checksum：校验和，用于检验首部和数据两个部分。

Urgent Pointer：紧急数据指针，指出本报文段中的紧急数据的字节数。

Options：定义一些其他可选的参数（长度可变，此时为12字节）

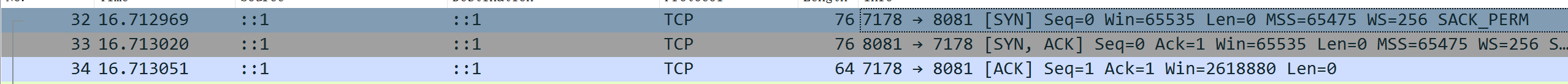
总结来说，结构如下：



接下来，我们将根据这个结构，联系三次握手和四次挥手的具体过程进行分析。

（2）三次握手

三次握手如图所示。



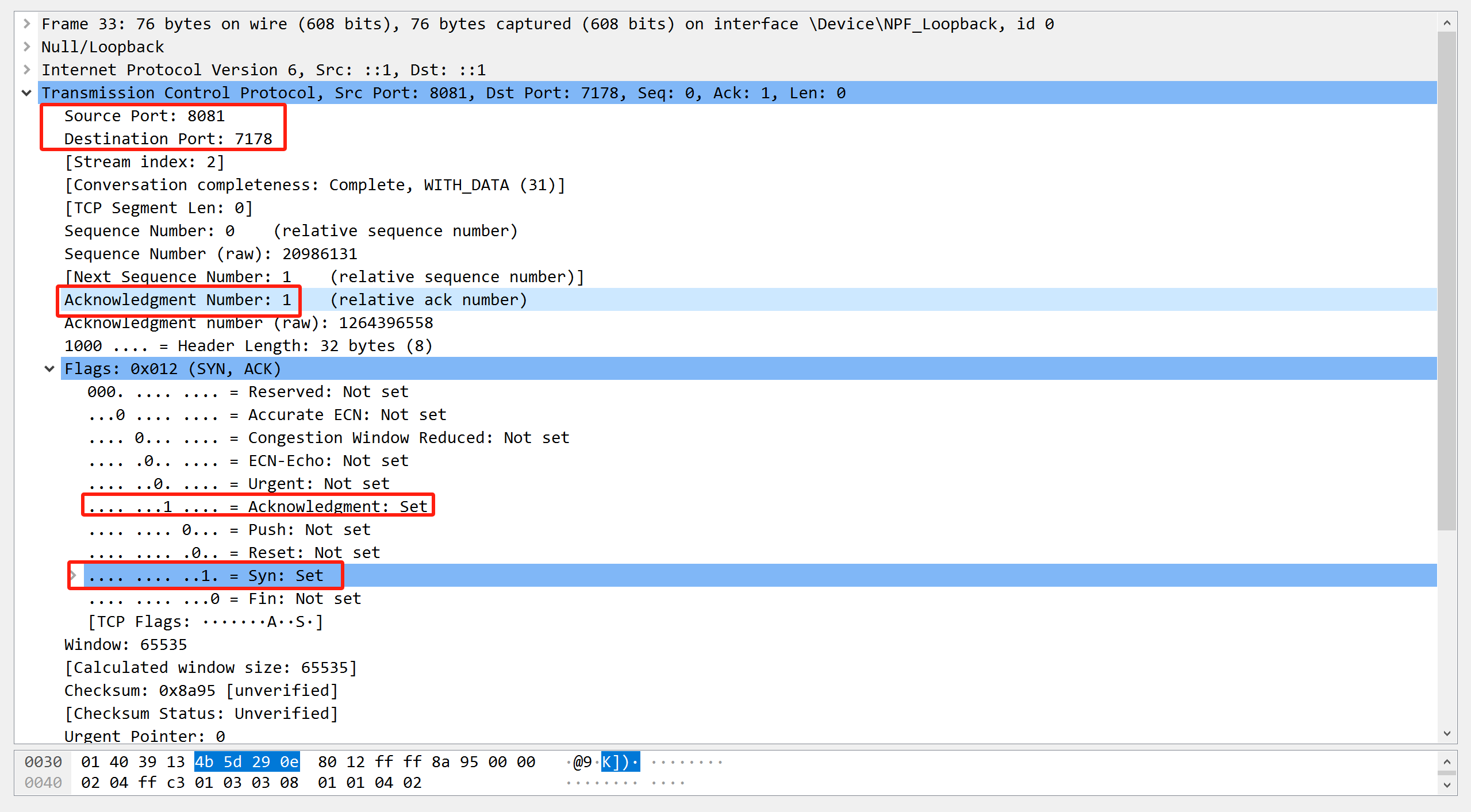
①服务器端进入LISTEN监听状态

②客户进程向服务器发出连接请求报文，即该报文的SYN位置1，同时选择一个初始序列号seq=x。发送完SYN后，该客户进程进入SYN\_SENT状态（同步已发送状态），等待服务器端确认（该报文即为上图所示报文）

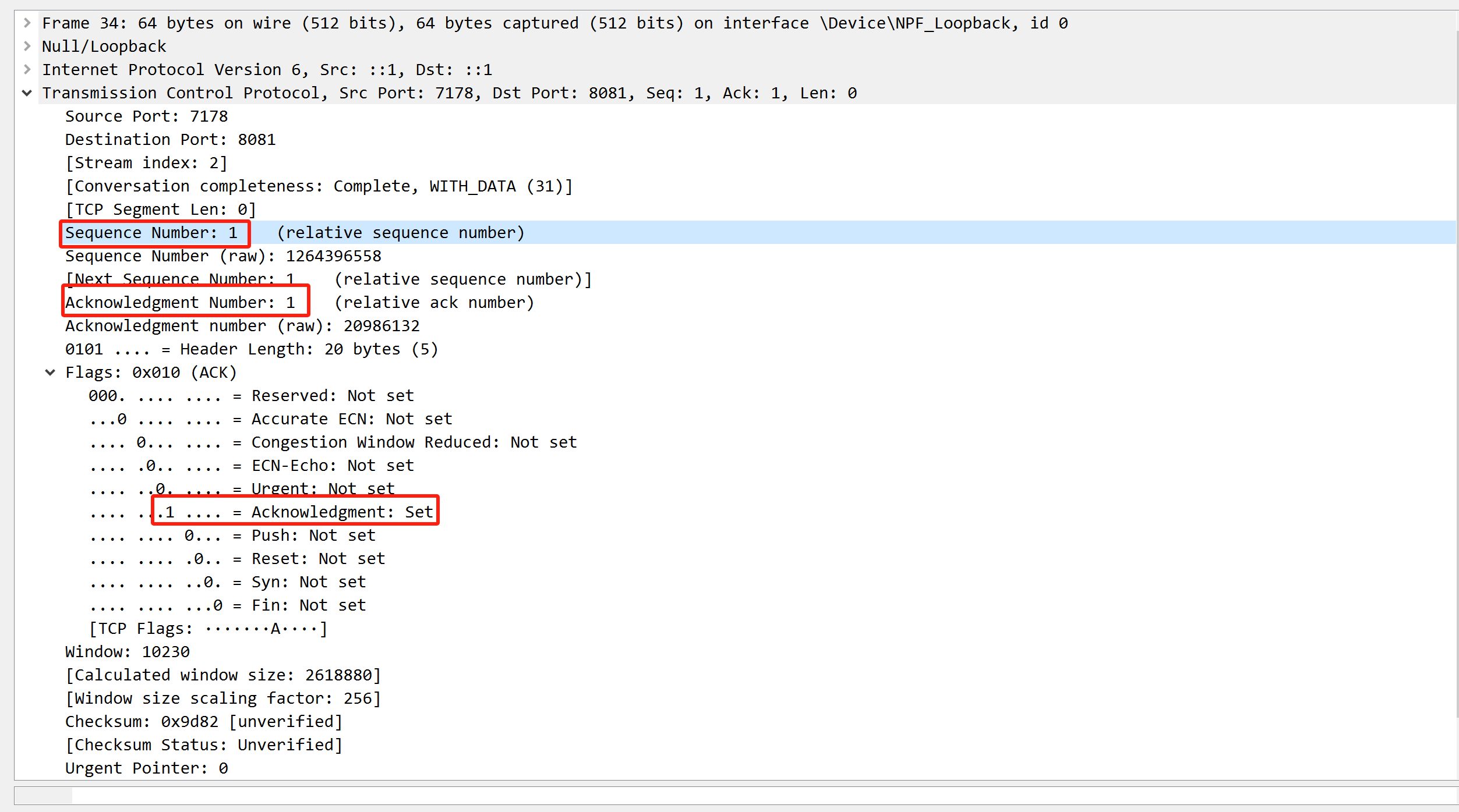
③服务器收到请求报文（SYN位置1的报文）后，如果同意连接，则发出确认报文，确认报文中应将ACK置1，将SYN置1，同时确认号Acknowledgement Number应该为x+1（可以看到图中为1），即收到的连接请求报文的seq的值加上1；

同时，为自己初始化一个序列号seq=y（此时图中显示为1）；

服务器进入SYN-RCVD状态（同步收到状态）；



④客户进程收到确认后，还要向服务器给出确认。发送确认报文，ACK位置1，Acknowledgement Number为y+1（此时图中为1），自己的序列号为x+1（此时图中为1）。建立TCP连接，客户端进入ESTABLISHED（已建立连接状态）。

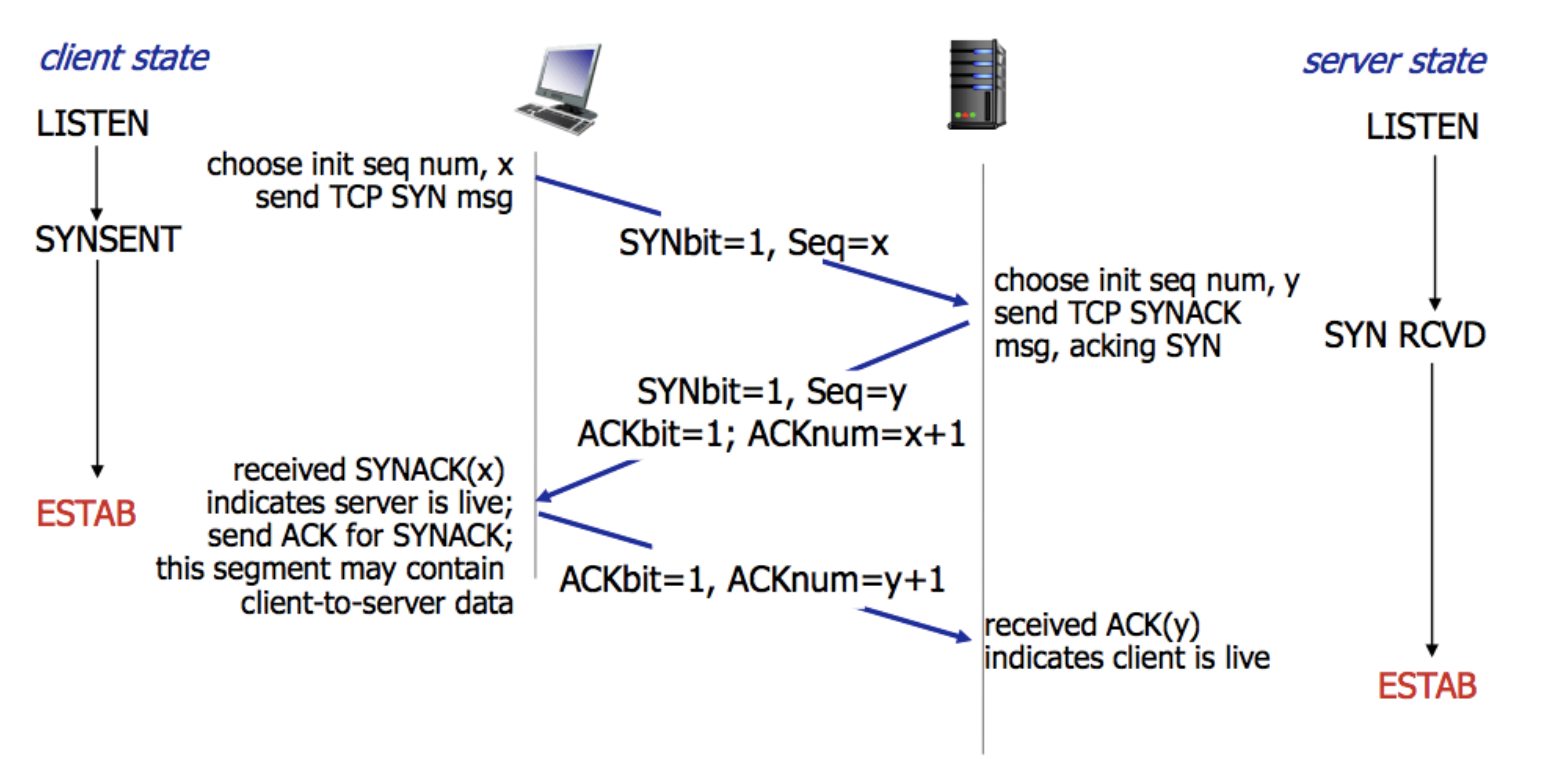


⑤服务器端收到客户端的确认后也进入ESTABLISHED状态，连接建立。

总结：

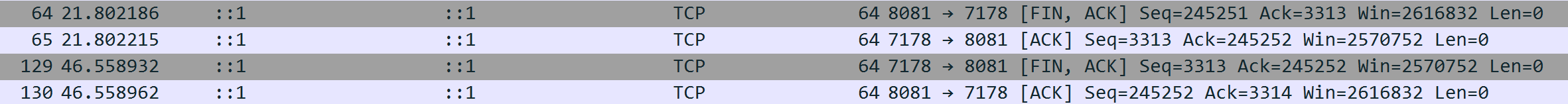
在建立过程中总共出现了三个类型的报文：客户进程发送的请求报文，确认报文和服务器发送的确认报文。客户进程先发送了一个请求报文，其中SYN=1，且选择了一个初始序列号seq=x；然后服务器端发送确认报文，其中ACK=1，SYN=1，（因为ACK=1了所以要关注ack即确认序号Acknowledgement Number），ack=x+1，seq=y；然后客户端再次发送一个确认报文，其中ACK=1，ack=y+1，seq=x+1。

图解如下：

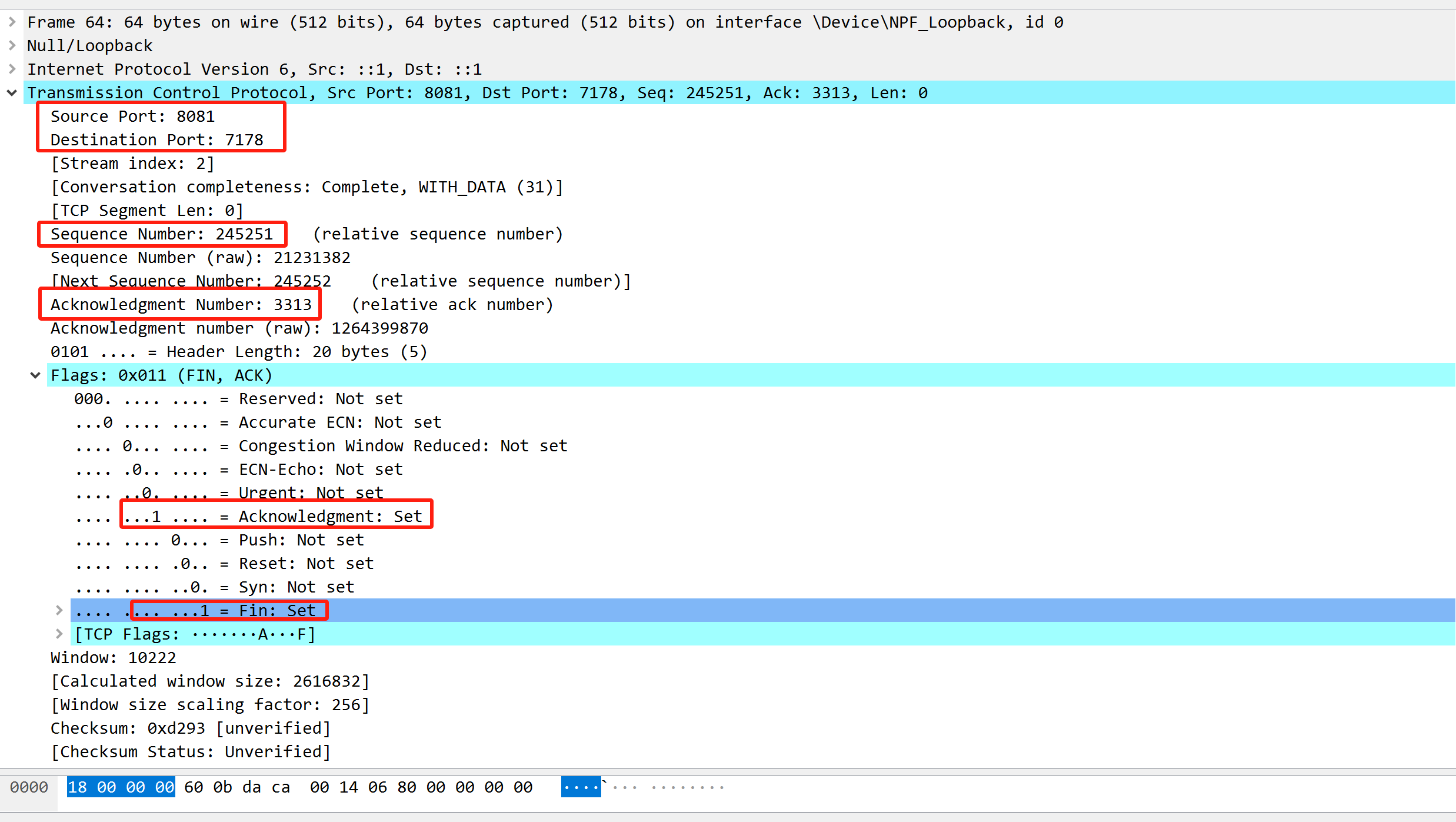


（3）四次挥手

四次挥手如图所示。（注意常见的是客户端先发送连接释放报文，但是本例中的服务器先发送连接释放报文，源端口和目的端口不一样，但是大致的过程是一样的，我的分析以常见情况为准，而不是以服务器发起连接释放报文为准）



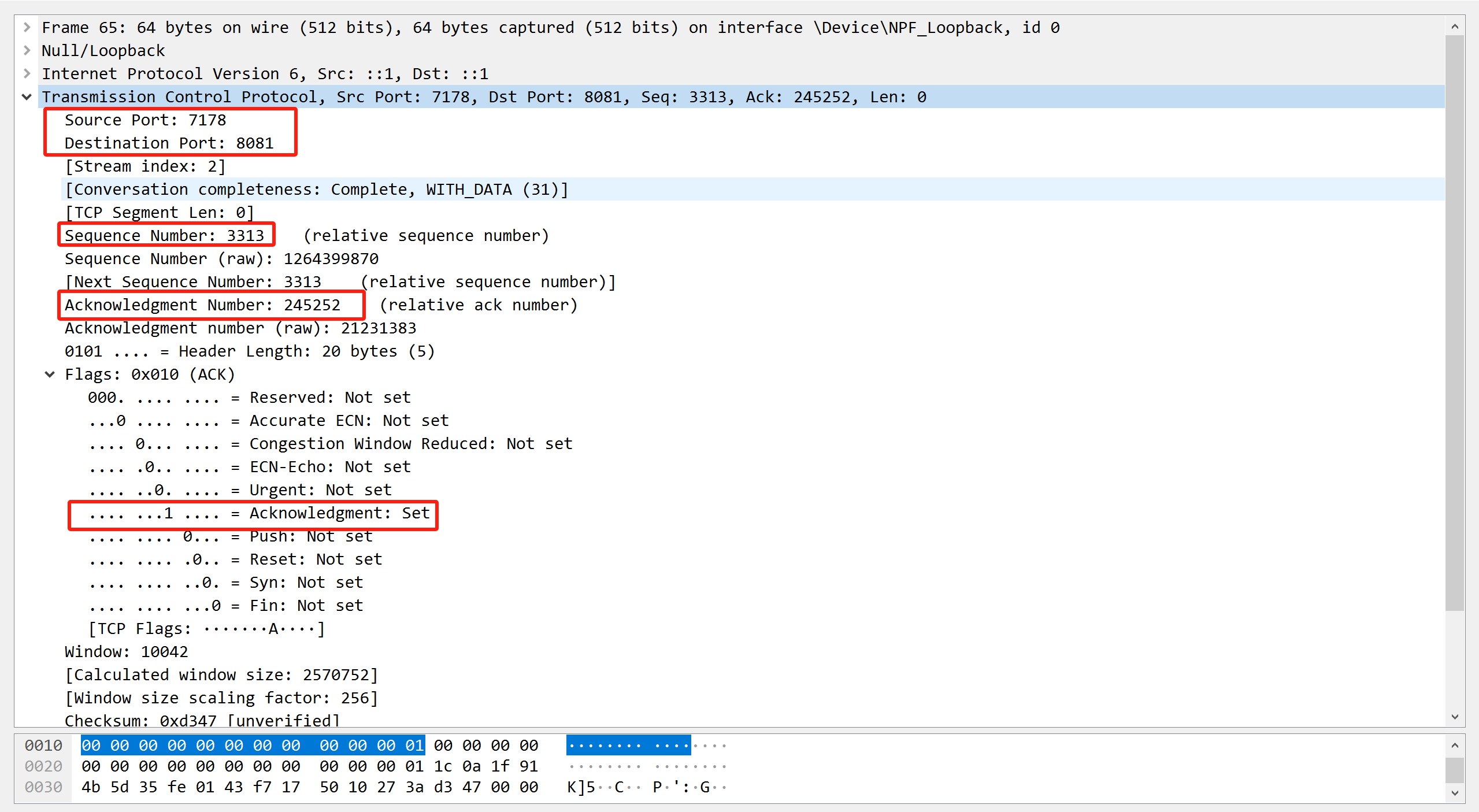
1. 客户端进程发出连接释放报文，并且停止发送数据。该连接释放报文的FIN位置1，（此时ACK位也可能置1，是因为这是对上一个包的确认，与四次挥手并没有关系）序列号seq=u（前面已经传送过来的数据的最后一个字节的序号加1，此时为245251），同时，客户端进入FIN-WAIT-1（终止等待1）状态，对应的报文如下：



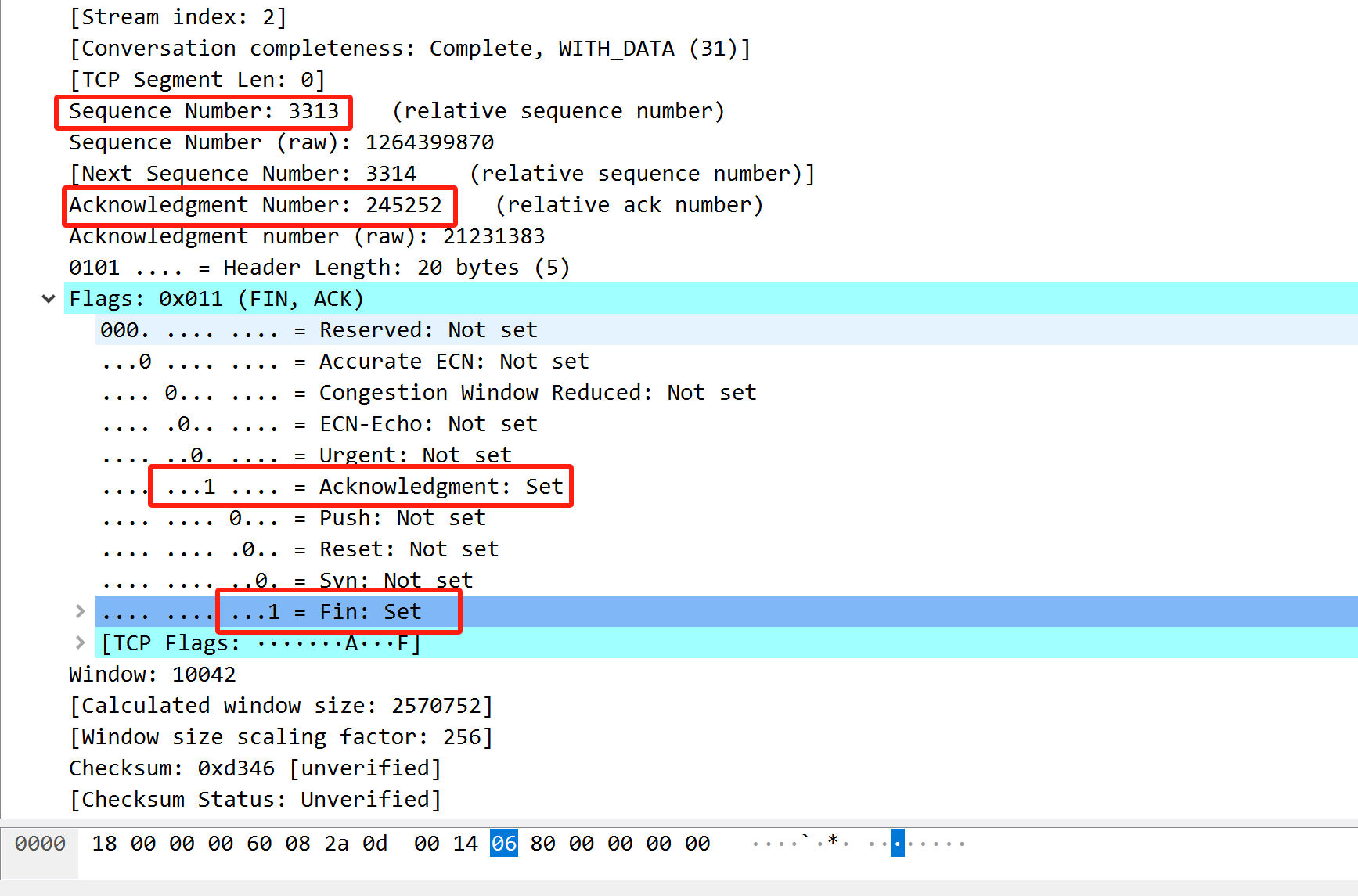
1. 服务器端收到连接释放报文，发出确认报文（ACK=1,ack=u+1，此时为245252），序列号seq=v（此时为3313）,服务器端进入CLOSE-WAIT状态（关闭等待状态），表明自己接收到了客户端关闭连接的请求，但还没有准备好关闭连接。

此时处于半关闭状态，即客户端已经没有数据要发送了，但是服务器若发送数据，客户端还要继续接受。

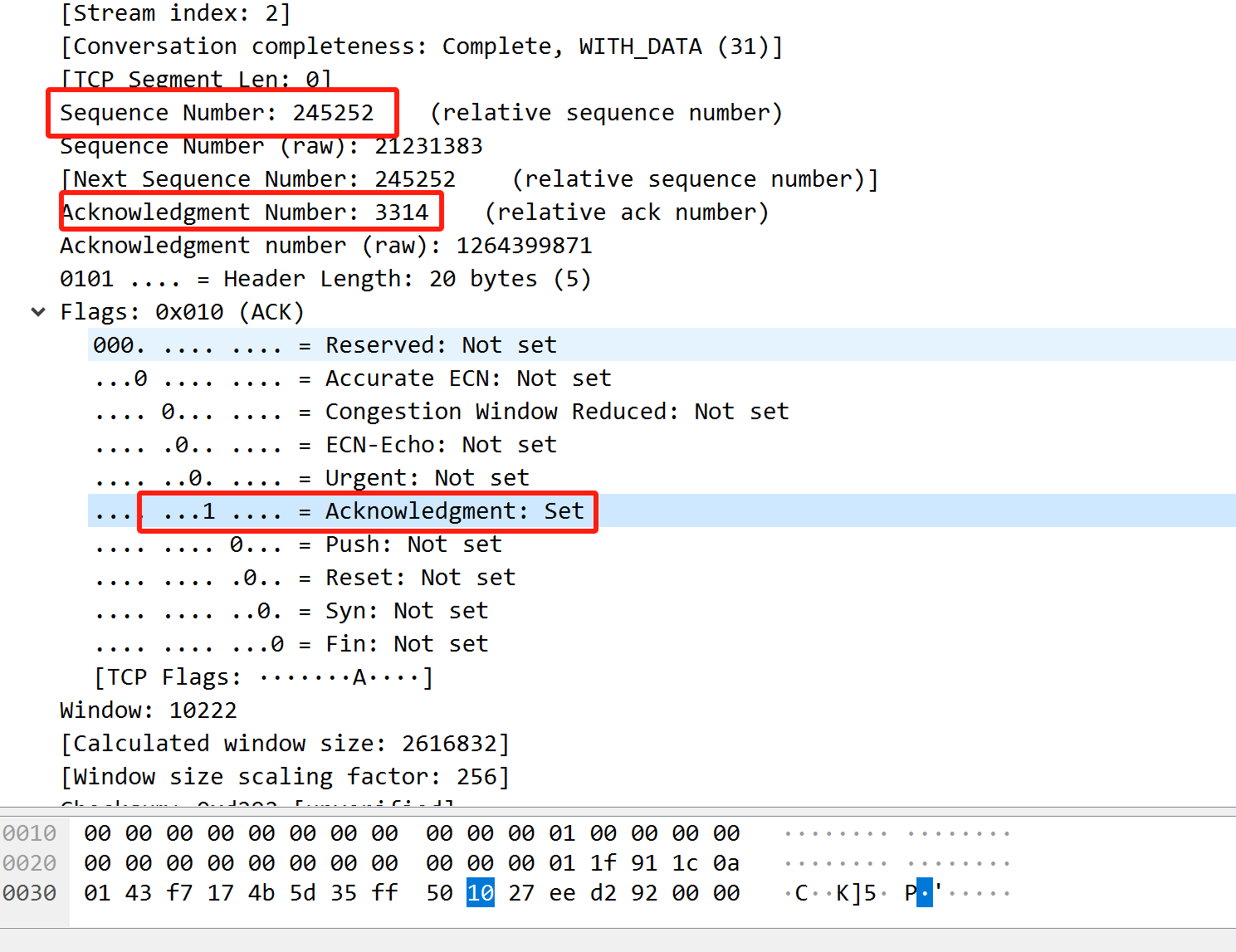
对应报文如下：



1. 客户端收到确认包后，进入FIN-WAIT-2状态（终止等待2状态），等待服务器关闭连接。
2. 服务器端准备好关闭连接后（数据全都发送完毕），向客户端发送结束连接请求，发送一个连接释放报文，该报文FIN=1,ACK=1，ack=u+1（此时为245252），此时的序列号seq=w（此时为3313）；接下来，服务器进入了LAST-ACK状态（最后确认状态），等待客户端的确认。对应报文如下：



1. 客户端收到服务器的连接释放报文后，发出确认报文，该报文ACK=1，ack=w+1（此时为3314），自己的序列号seq=u+1（此时为3314），客户端进图TIME-WAIT状态（时间等待状态），但是这个状态不等于TCP连接的释放，还需要等待2\*MSL（最长报文寿命）的时间后，才进入CLOSED状态。对应报文如下：

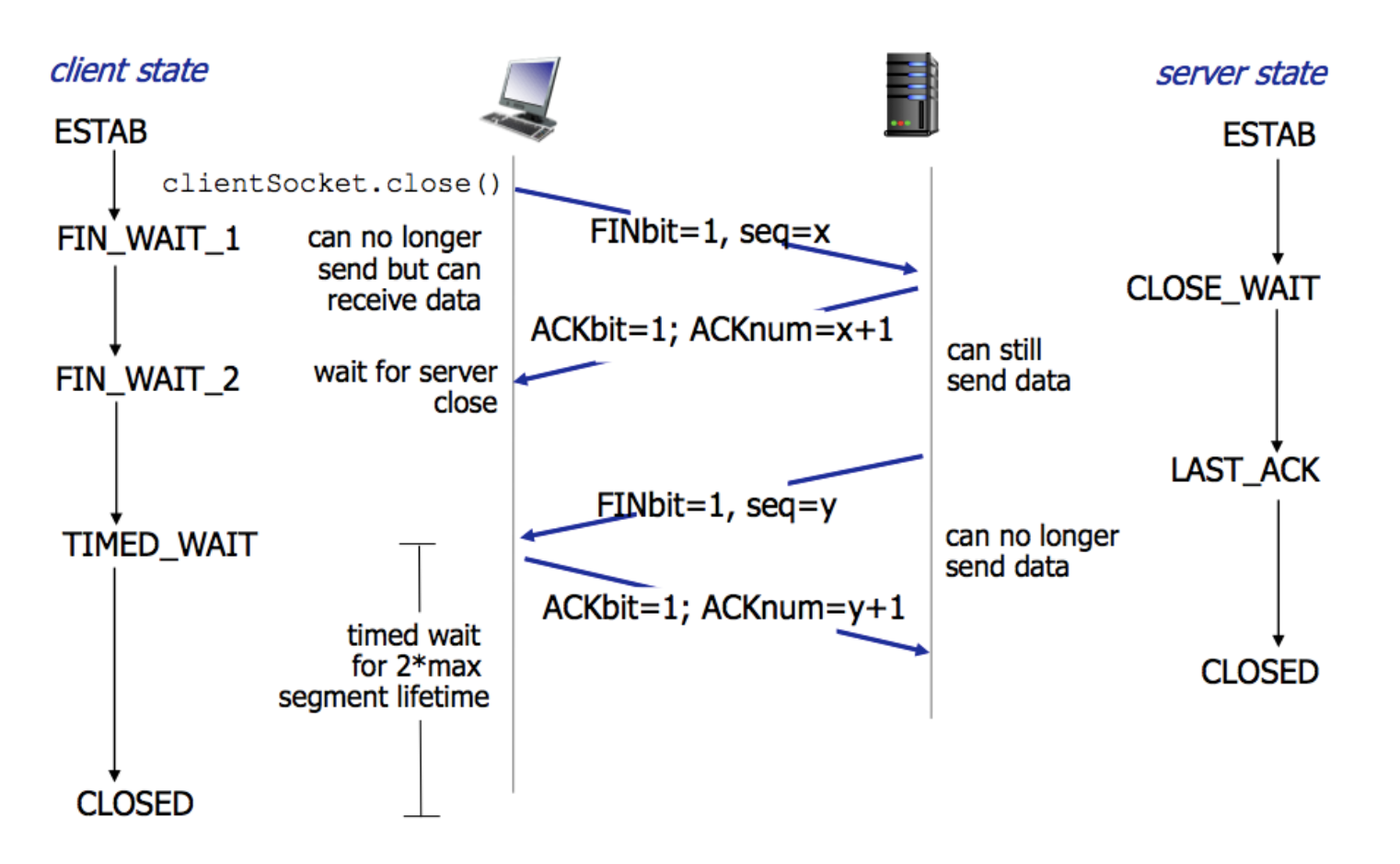


⑥服务器收到客户端发出的确认后，立即进入CLOSED状态。

总结：

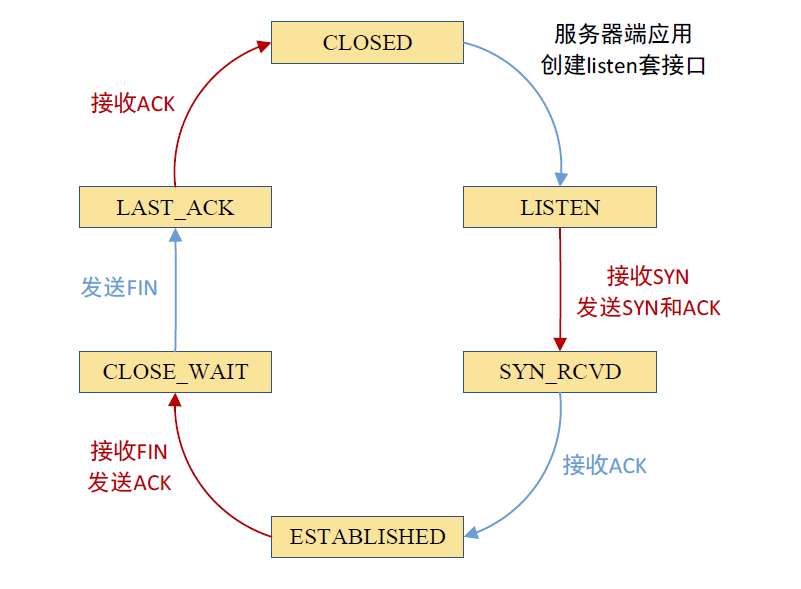
四次挥手可以由客户端发起也可以由服务器端发起，假设A端先挥手。A端首先发送连接释放报文，其中FIN=1，seq=u，表示这一段没有什么数据要继续发送了；B端接收到之后，发送确认报文表示成功收到，ACK=1，ack=u+1；当B端的数据也发送完成后，发送结束连接请求，其中FIN=1，ACK=1，ack=u+1；然后A端接收到之后，发出确认报文，ACK=1，ack=w+1，seq=u+1，同时进入TIME-WAIT状态，等待2\*MSL时间后进入CLOSED状态，而B端收到确认报文后立刻进入CLOSED状态。

图解如下：

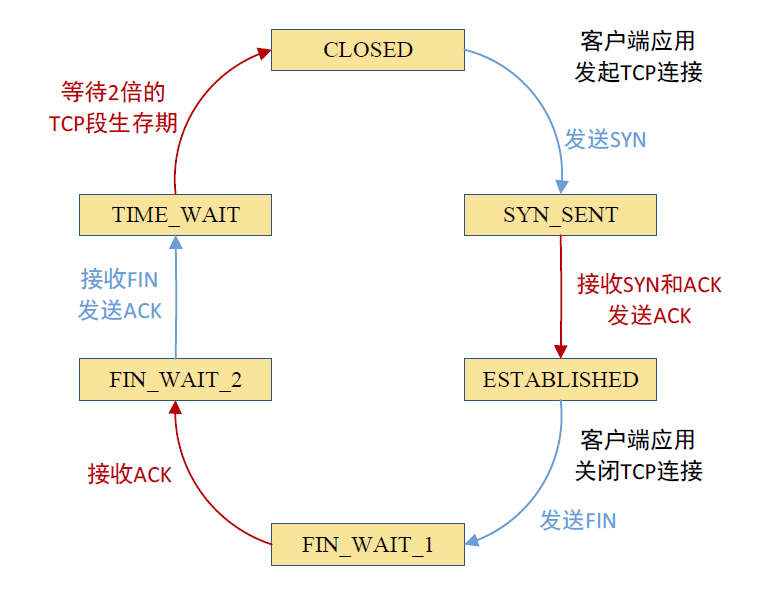


（4）服务器和客户端建立连接和释放连接的过程中的流程图。

服务器端：



客户端：



# 三、实验思考

1.建立连接时为什么TCP客户端最后还要发送一次确认

防止已经失效的连接请求报文突然又传送到了服务器，从而产生错误。

2. 如果已经建立了连接，但是客户端突然出现故障了会发生什么

服务器端每隔一段时间会发送一个探测报文，该探测报文包含的数据非常少，如果连续几个探测报文都没有得到响应，则认为当前的 TCP 连接已经死亡，系统内核将错误信息通知给上层应用程序。

3.为什么建立连接时三次握手，关闭连接确是四次挥手

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。

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。

4.有时候只出现了三次挥手是为什么

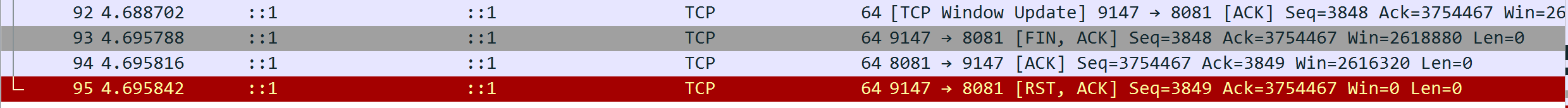
实际上是将第二次挥手和第三次挥手合并了，即服务器端已经没有数据可以发送了，当客户端发送关闭连接请求的时候，服务器端将ACK（确认包）和FIN（关闭连接）一起发送，于是表象上看起来只有三次挥手。

5.为什么客户端最后还要等待2MSL

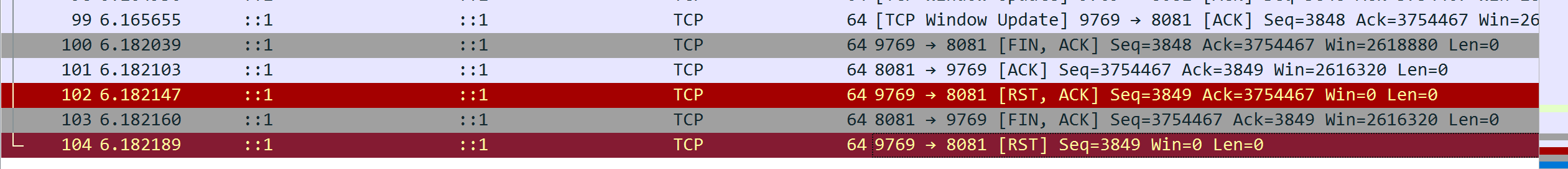
首先，保证客户端发送的最后一个ACK报文能够到达服务器。当客户端已经发送数据包但是没有接收到确认时，服务器端就会重新发送一次，此时就有可能在2MSL时间内到达客户端，接着给出回应报文，并目重启2MSL计时器。

其次，防止类似与“三次握手”中提到了的"已经失效的连接请求报文段”出现在本连接中。

6.客户端发起FIN关闭连接请求后，发送了一个RST数据包如下图所示。



或者下图所示。



首先，产生RST的三个情况是：

（1）目的地为某端口的SYN到达，然而在该端口上并没有正在监听的服务器；

（2）TCP想取消一个已有连接；

（3）在一个已关闭的TCP连接上收到数据

图中所示为第三种情况，客户端先向服务器发送了一个FIN包，然后服务器确认接受，发送了ACK，此时只进行了两次挥手，服务器并没有发送FIN；后服务器继续向客户端发送数据，但是客户端已经关闭，由于此时连接状态异常，所以客户端不会向服务器发送ACK字段，而是发送一个RST报文请求将处于异常状态的连接复位。

7. http1.0版本和1.1版本有什么区别

http1.0版本是短链接，每个HTTP请求/响应都需要建立一个新的TCP连接；

http1.1版本是长连接，允许多个请求和响应共享同一连接。

8. 一张图片为什么会被分段传输（TCP Segment）

因为TCP是基于字节流传输的