实验2：配置Web服务器，编写简单页面，分析交互过程

2010239-李思凡

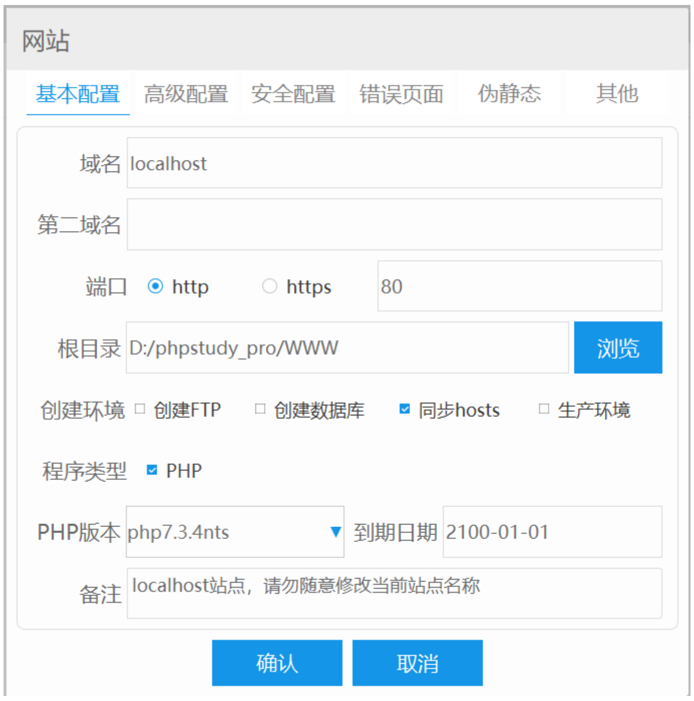
实验要求

（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。

Web服务器搭建

采用phpStudy搭建Web服务器。下载phpStudy，打开后在“网站”处可以看到服务器的域名为localhost，端口号为80，执行文件的根目录在WWW下。将Apache启动，即启动服务器。站点配置如下：



域名	localhost
第二域名	
端口	[✓ http] https 80
根目录	D:/phpstudy_pro/WWW
创建环境	☐ 创建FTP ☐ 创建数据库 ☑ 同步hosts ☐ 生产环境
程序类型	☑ PHP
PHP版本	php7.3.4nts
到期日期	2100-01-01
备注	localhost站点，请勿随意修改当前站点名称

Web页面

使用html语言编写简单的Web页面，其中包括文字信息：标题、专业、学号、姓名，以及图片信息logo。代码如下所示：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>WEB页面</title>
</head>
<body>
    <h1>Christine的WEB页面</h1>
```

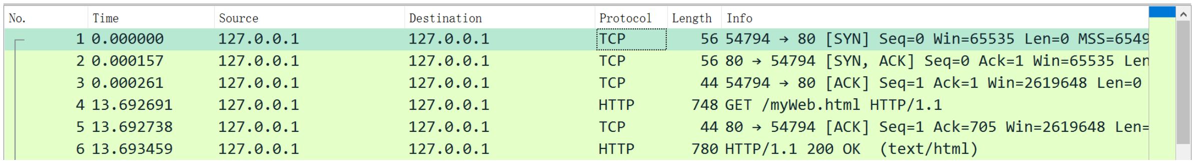
```
        <p>专业：计算机科学与技术</p>
        <p>学号：2010239</p>
        <p>姓名：李思凡</p>
        <p>Logo: </p>
        <img src=/logo.jpg>
    </body>
    </html>
```

分析Wireshark捕获文件

传输的整体流程

1. 客户端与服务器端通过三次握手建立连接
2. 请求页面，服务器返回HTML内容
3. 请求文字、图片等具体内容，服务器返回
4. 四次挥手断开连接

三次握手



```
No.  Time       Source     Destination  Protocol  Length  Info
  1  0.000000   127.0.0.1  127.0.0.1    TCP           56  54794 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=6549
  2  0.000157   127.0.0.1  127.0.0.1    TCP           56  80 → 54794 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len
  3  0.000261   127.0.0.1  127.0.0.1    TCP           44  54794 → 80 [ACK] Seq=1 Ack=1 Win=2619648 Len=0
  4  13.692691  127.0.0.1  127.0.0.1    HTTP         748  GET /myWeb.html HTTP/1.1
  5  13.692738  127.0.0.1  127.0.0.1    TCP           44  80 → 54794 [ACK] Seq=1 Ack=705 Win=2619648 Len=
  6  13.693459  127.0.0.1  127.0.0.1    HTTP         780  HTTP/1.1 200 OK  (text/html)
```

由第4条消息可以看出，采用的协议为HTTP1.1，方式默认为持久连接，在相同的TCP连接上，服务器接收请求、给出响应，响应后保持连接。

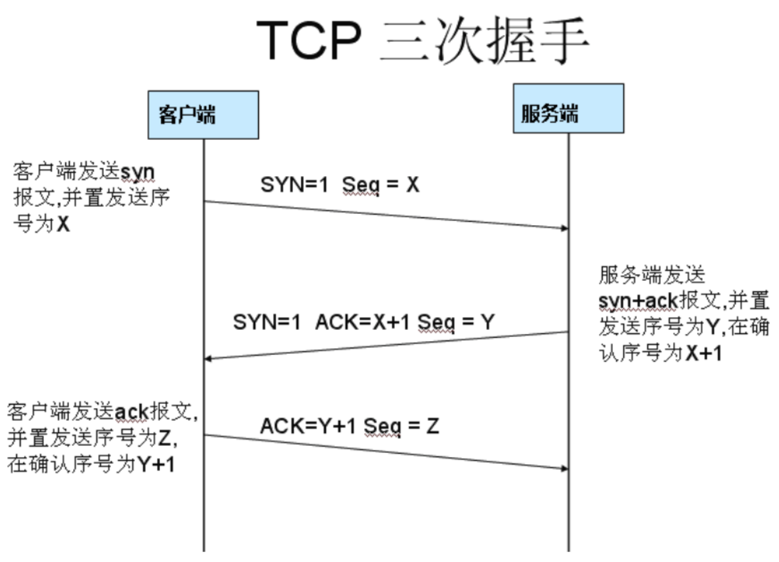
TCP报文中重要的几个字段有：

（1）序列号Seq，用于确定是否成功传输及顺序；

（2）确认序号ACK，ACK标志位为1时，确认序号字段有效，为Seq+1；

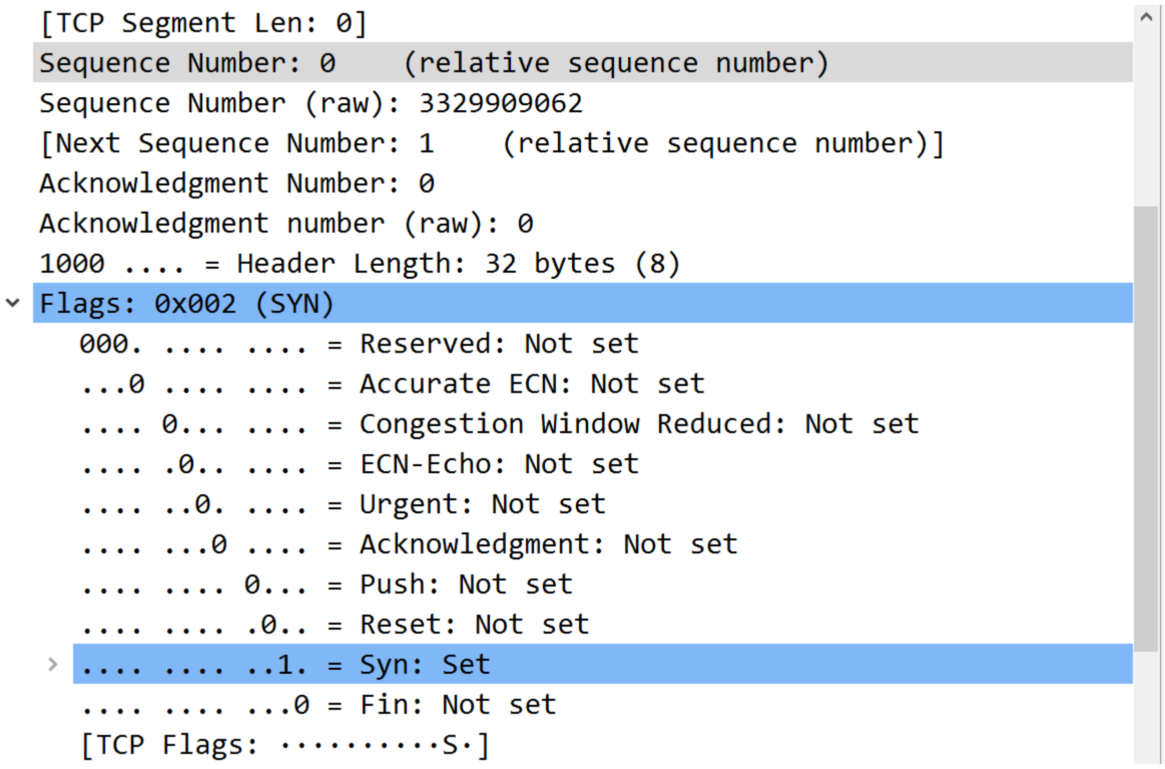
（3）标志位：ACK决定确认序号是否有效，SYN有效表示发起一个连接，FIN有效表示释放一个连接。

三次握手的流程如下图所示：


TCP 三次握手
客户端
服务端
客户端发送syn报文,并置发送序号为X
SYN=1 Seq = X
服务端发送syn+ack报文,并置发送序号为Y,在确认序号为X+1
SYN=1 ACK=X+1 Seq = Y
客户端发送ack报文,并置发送序号为Z,在确认序号为Y+1
ACK=Y+1 Seq = Z

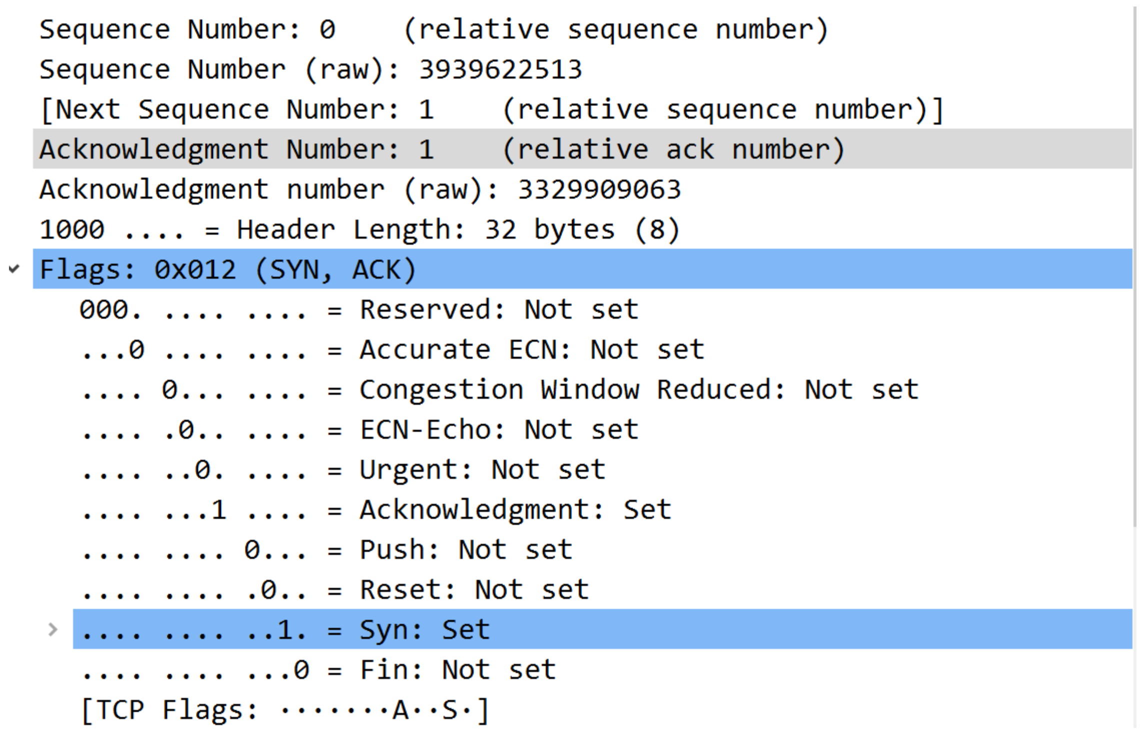
1. **首先客户端向服务器端发送一个TCP报文。** SYN=1，表示“请求建立连接”；Seq为随机产生的X，相对值为0；标志位的ACK=0，表示未确认。之后客户端进入SYN-SENT状态，表示在请求连接的阶

段。



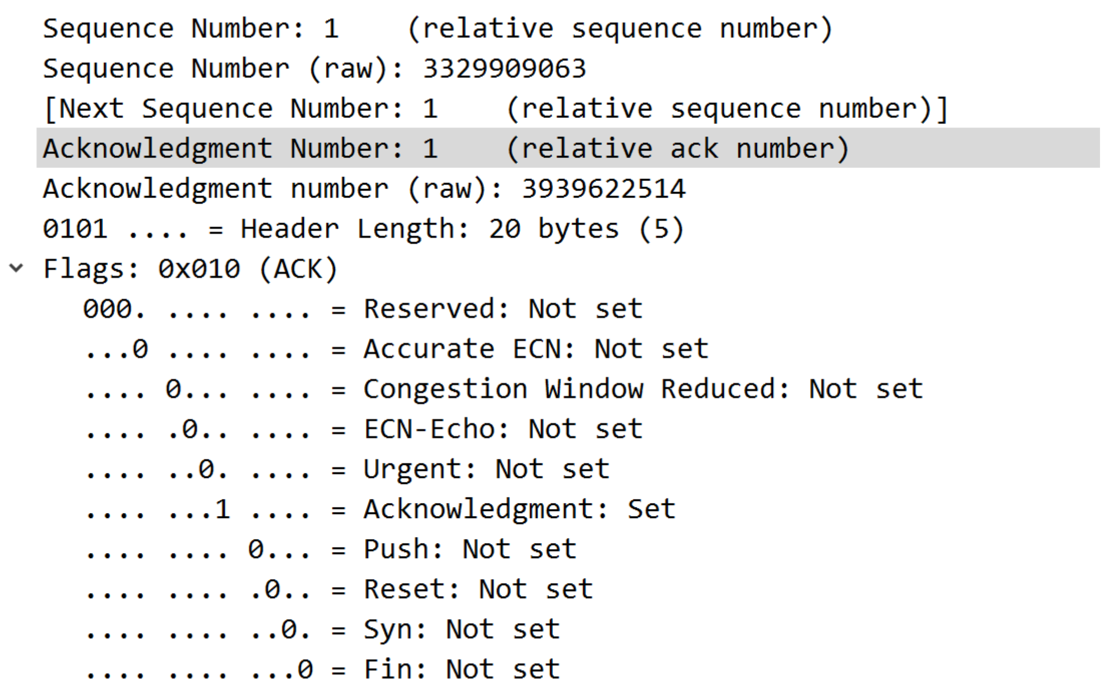
```
[TCP Segment Len: 0]
Sequence Number: 0    (relative sequence number)
Sequence Number (raw): 3329909062
[Next Sequence Number: 1    (relative sequence number)]
Acknowledgment Number: 0
Acknowledgment number (raw): 0
1000 .... = Header Length: 32 bytes (8)
Flags: 0x002 (SYN)
  000. .... .... = Reserved: Not set
  ...0 .... .... = Accurate ECN: Not set
  .... 0... .... = Congestion Window Reduced: Not set
  .... .0.. .... = ECN-Echo: Not set
  .... ..0. .... = Urgent: Not set
  .... ...0 .... = Acknowledgment: Not set
  .... .... 0... = Push: Not set
  .... .... .0.. = Reset: Not set
  .... .... ..1. = Syn: Set
  .... .... ...0 = Fin: Not set
  [TCP Flags: ··········S·]
```

2. **服务器端收到后，返回确认报文。**标志位SYN=1，ACK=1，表示服务器收到了客户端的连接请求；序列号Seq为随机的Y，相对值为0；确认号ACK=X+1，相对值为1，表示收到了客户端的Seq，并+1作为确认，使得两边可以匹配成功。之后服务器端进入SYN-REVD状态，表示已确认客户端的连接请求。



```
Sequence Number: 0    (relative sequence number)
Sequence Number (raw): 3939622513
[Next Sequence Number: 1    (relative sequence number)]
Acknowledgment Number: 1    (relative ack number)
Acknowledgment number (raw): 3329909063
1000 .... = Header Length: 32 bytes (8)
Flags: 0x012 (SYN, ACK)
  000. .... .... = Reserved: Not set
  ...0 .... .... = Accurate ECN: Not set
  .... 0... .... = Congestion Window Reduced: Not set
  .... .0.. .... = ECN-Echo: Not set
  .... ..0. .... = Urgent: Not set
  .... ...1 .... = Acknowledgment: Set
  .... .... 0... = Push: Not set
  .... .... .0.. = Reset: Not set
  .... .... ..1. = Syn: Set
  .... .... ...0 = Fin: Not set
  [TCP Flags: ·······A··S·]
```

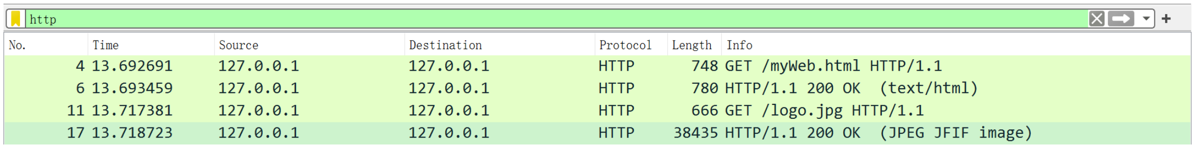
3. **客户端收到后，确认客户端和服务器间数据传输正常，返回确认报文。**首先检查收到的ACK是否正确，若收到的ACK等于第一次发送的序列号加一，则正确。标志位ACK=1，表示确认收到服务器同意连接的信号；序列号Seq=X+1，相对值为1，表示收到服务器的ACK并将其作为自己的序列号；确认号ACK=Y+1，相对值为1，表示收到了服务器端的Seq，并+1作为确认，使得两边可以匹配成功。之后进入连接状态。



```
Sequence Number: 1    (relative sequence number)
Sequence Number (raw): 3329909063
[Next Sequence Number: 1    (relative sequence number)]
Acknowledgment Number: 1    (relative ack number)
Acknowledgment number (raw): 3939622514
0101 .... = Header Length: 20 bytes (5)
Flags: 0x010 (ACK)
    000. .... .... = Reserved: Not set
    ...0 .... .... = Accurate ECN: Not set
    .... 0... .... = Congestion Window Reduced: Not set
    .... .0.. .... = ECN-Echo: Not set
    .... ..0. .... = Urgent: Not set
    .... ...1 .... = Acknowledgment: Set
    .... .... 0... = Push: Not set
    .... .... .0.. = Reset: Not set
    .... .... ..0. = Syn: Not set
    .... .... ...0 = Fin: Not set
```

第三次握手的原因是："第三次握手"其实是客户端告知服务器端是否收到服务器端"第二次握手"传来的数据，若收到了则正常建立连接，否则服务器关闭连接。

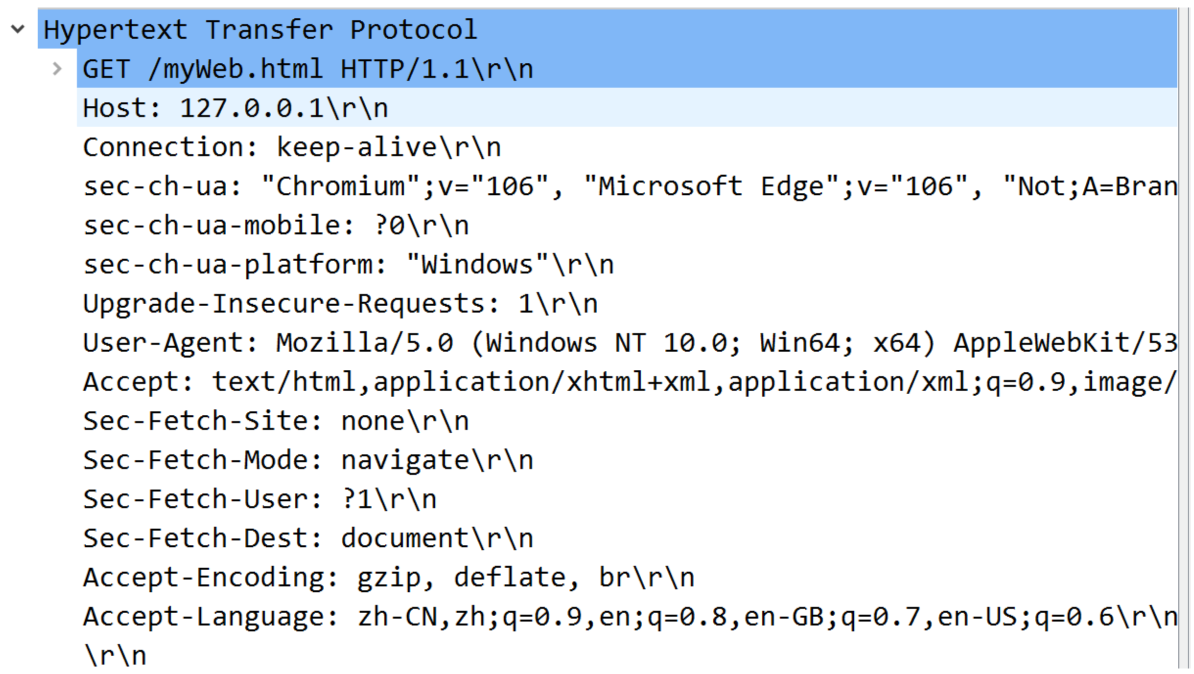
HTTP请求



```
http
No.   Time       Source      Destination  Protocol  Length  Info
   4 13.692691   127.0.0.1   127.0.0.1    HTTP        748 GET /myWeb.html HTTP/1.1
   6 13.693459   127.0.0.1   127.0.0.1    HTTP        780 HTTP/1.1 200 OK  (text/html)
  11 13.717381   127.0.0.1   127.0.0.1    HTTP        666 GET /logo.jpg HTTP/1.1
  17 13.718723   127.0.0.1   127.0.0.1    HTTP      38435 HTTP/1.1 200 OK  (JPEG JFIF image)
```

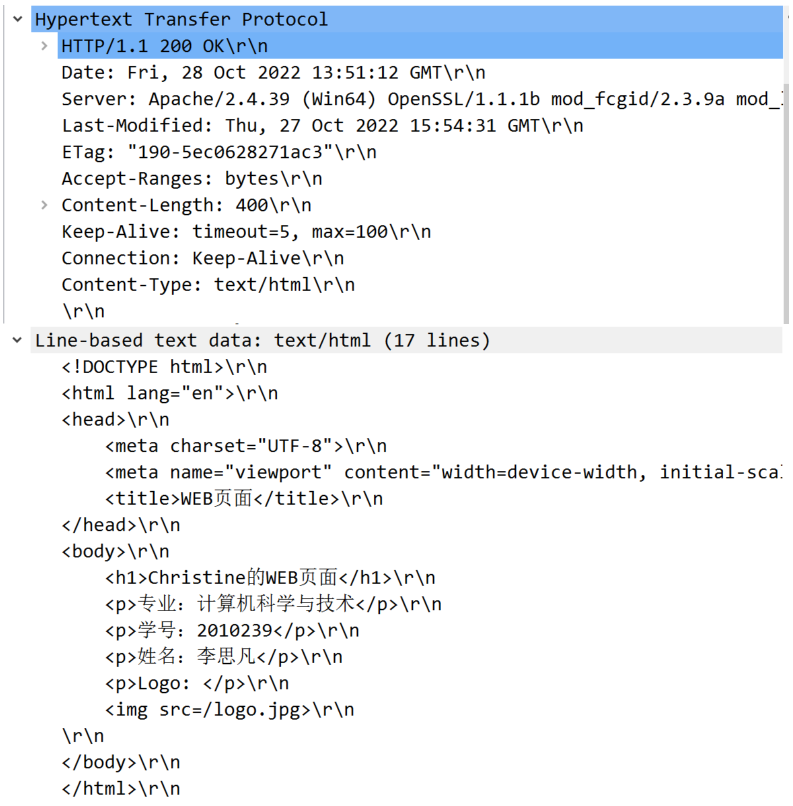
HTTP消息格式为在原有TCP格式的基础上，增加超文本传输协议部分。建立连接后，浏览器向服务器发送请求HTTP命令，服务器接收请求并返回相应的HTTP响应。

1. **首先客户端向服务器发送HTTP请求报文，获取网页文档。**采用请求方法GET，URL为/myWeb.html，HTTP版本为1.1。



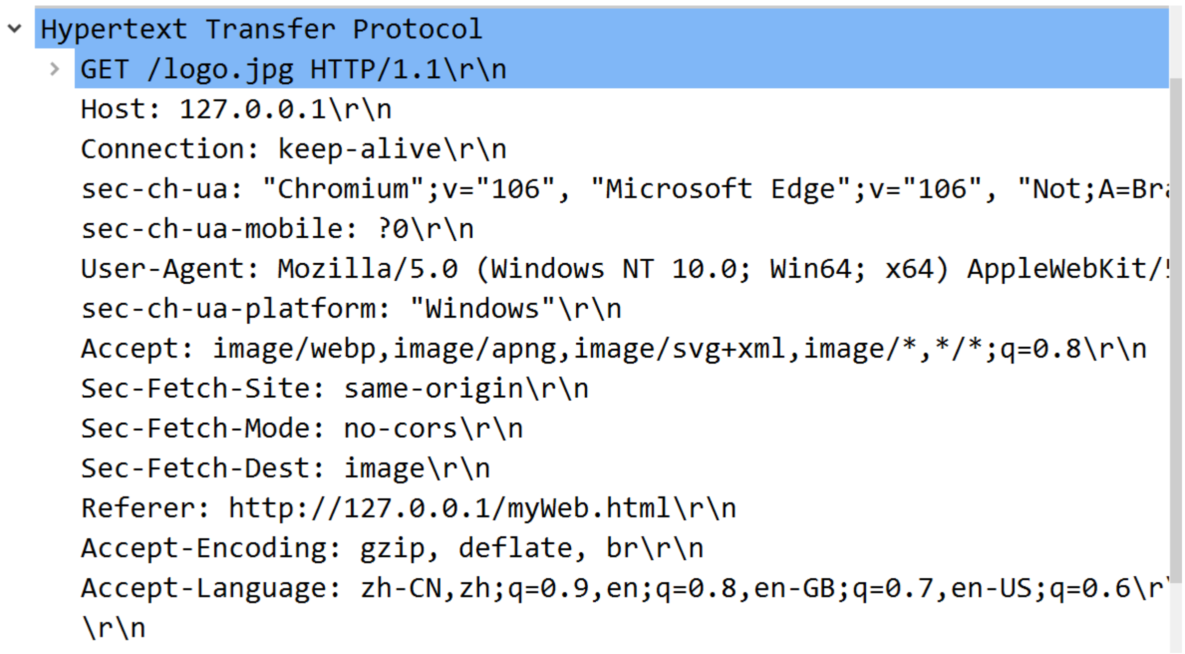
```
Hypertext Transfer Protocol
  GET /myWeb.html HTTP/1.1\r\n
  Host: 127.0.0.1\r\n
  Connection: keep-alive\r\n
  sec-ch-ua: "Chromium";v="106", "Microsoft Edge";v="106", "Not;A=Bran
  sec-ch-ua-mobile: ?0\r\n
  sec-ch-ua-platform: "Windows"\r\n
  Upgrade-Insecure-Requests: 1\r\n
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/53
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/
  Sec-Fetch-Site: none\r\n
  Sec-Fetch-Mode: navigate\r\n
  Sec-Fetch-User: ?1\r\n
  Sec-Fetch-Dest: document\r\n
  Accept-Encoding: gzip, deflate, br\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
  \r\n
```

2. **服务端返回给客户端HTTP响应报文，以及客户端请求的网页文档。**响应的状态码和解释为"200 OK"，表示请求成功，请求的文档内容包含在数据部分。



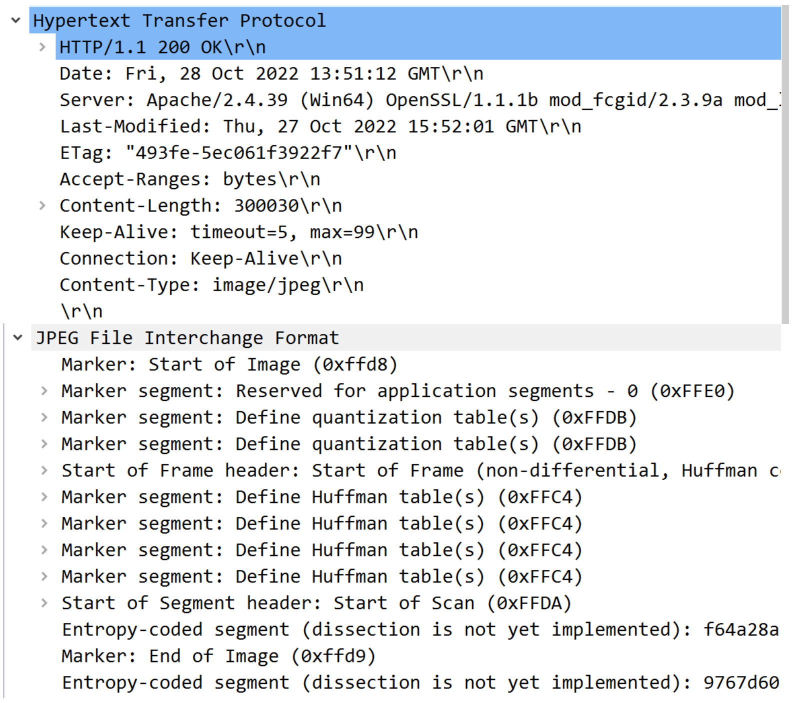
```
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
  Date: Fri, 28 Oct 2022 13:51:12 GMT\r\n
  Server: Apache/2.4.39 (Win64) OpenSSL/1.1.1b mod_fcgid/2.3.9a mod_
  Last-Modified: Thu, 27 Oct 2022 15:54:31 GMT\r\n
  ETag: "190-5ec0628271ac3"\r\n
  Accept-Ranges: bytes\r\n
  Content-Length: 400\r\n
  Keep-Alive: timeout=5, max=100\r\n
  Connection: Keep-Alive\r\n
  Content-Type: text/html\r\n
  \r\n
Line-based text data: text/html (17 lines)
  <!DOCTYPE html>\r\n
  <html lang="en">\r\n
  <head>\r\n
      <meta charset="UTF-8">\r\n
      <meta name="viewport" content="width=device-width, initial-sca
      <title>WEB页面</title>\r\n
  </head>\r\n
  <body>\r\n
      <h1>Christine的WEB页面</h1>\r\n
      <p>专业：计算机科学与技术</p>\r\n
      <p>学号：2010239</p>\r\n
      <p>姓名：李思凡</p>\r\n
      <p>Logo: </p>\r\n
      <img src=/logo.jpg>\r\n
  \r\n
  </body>\r\n
  </html>\r\n
```

3. **编写的Web页面包含一幅图像，客户端向服务器端发送HTTP请求报文，获取图片。**采用请求方法GET，URL为/logo.jpg，HTTP版本为1.1。



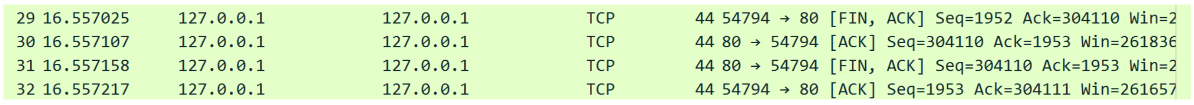
```
Hypertext Transfer Protocol
  GET /logo.jpg HTTP/1.1\r\n
  Host: 127.0.0.1\r\n
  Connection: keep-alive\r\n
  sec-ch-ua: "Chromium";v="106", "Microsoft Edge";v="106", "Not;A=Br
  sec-ch-ua-mobile: ?0\r\n
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
  sec-ch-ua-platform: "Windows"\r\n
  Accept: image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8\r\n
  Sec-Fetch-Site: same-origin\r\n
  Sec-Fetch-Mode: no-cors\r\n
  Sec-Fetch-Dest: image\r\n
  Referer: http://127.0.0.1/myWeb.html\r\n
  Accept-Encoding: gzip, deflate, br\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
  \r\n
```

4. **由于图片较大，服务器分多次传送图片，传送成功后服务端返回给客户端HTTP响应报文，以及客户端请求的图片。**响应的状态码和解释为“200 OK”，表示请求成功，请求的图片包含在数据部分。



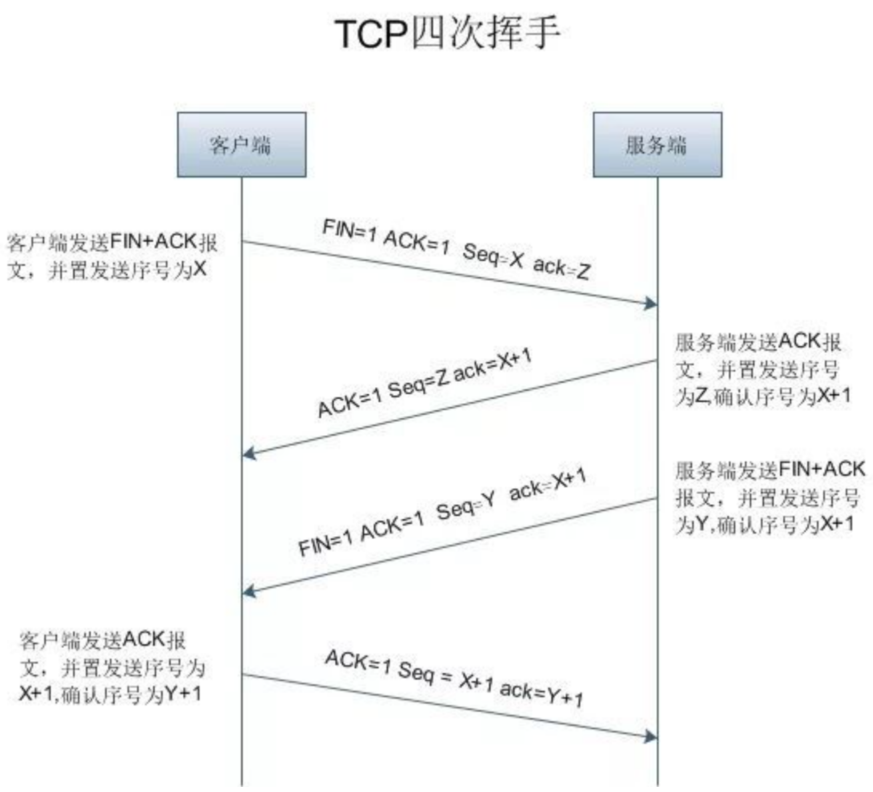
```
⌄ Hypertext Transfer Protocol
  › HTTP/1.1 200 OK\r\n
    Date: Fri, 28 Oct 2022 13:51:12 GMT\r\n
    Server: Apache/2.4.39 (Win64) OpenSSL/1.1.1b mod_fcgid/2.3.9a mod_
    Last-Modified: Thu, 27 Oct 2022 15:52:01 GMT\r\n
    ETag: "493fe-5ec061f3922f7"\r\n
    Accept-Ranges: bytes\r\n
  › Content-Length: 300030\r\n
    Keep-Alive: timeout=5, max=99\r\n
    Connection: Keep-Alive\r\n
    Content-Type: image/jpeg\r\n
    \r\n
⌄ JPEG File Interchange Format
    Marker: Start of Image (0xffd8)
  › Marker segment: Reserved for application segments - 0 (0xFFE0)
  › Marker segment: Define quantization table(s) (0xFFDB)
  › Marker segment: Define quantization table(s) (0xFFDB)
  › Start of Frame header: Start of Frame (non-differential, Huffman c
  › 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): f64a28a
    Marker: End of Image (0xffd9)
    Entropy-coded segment (dissection is not yet implemented): 9767d60
```

四次挥手



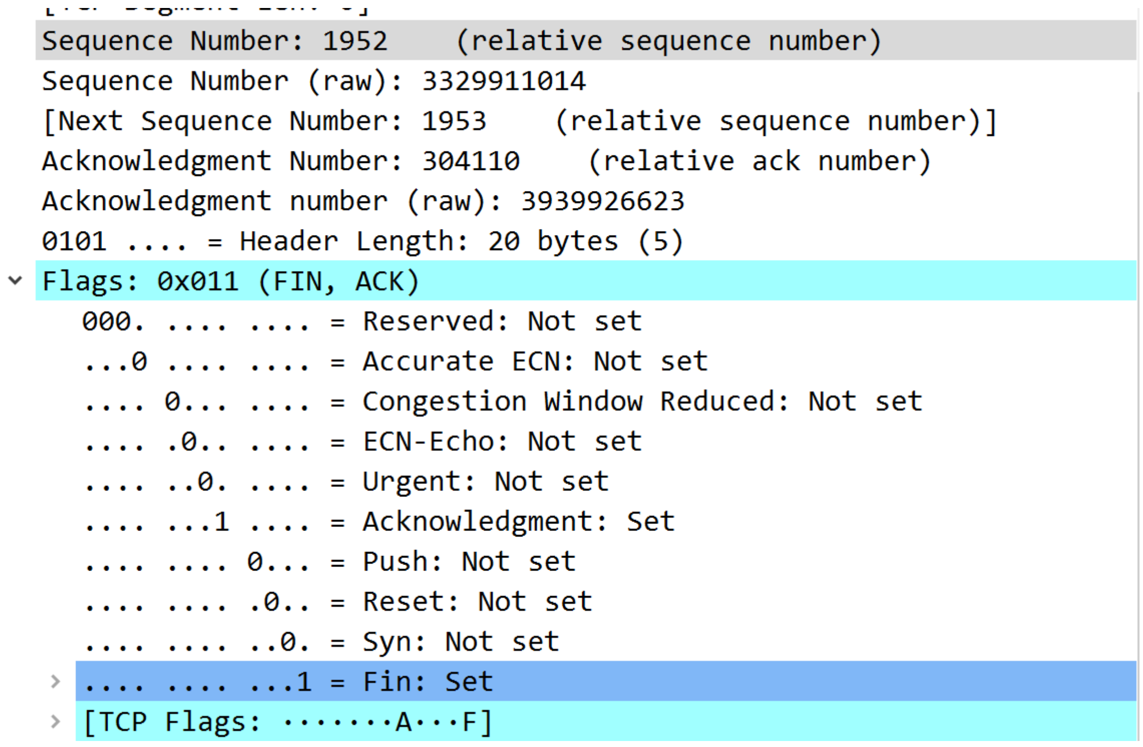
```
29 16.557025   127.0.0.1   127.0.0.1   TCP   44 54794 → 80 [FIN, ACK] Seq=1952 Ack=304110 Win=2
30 16.557107   127.0.0.1   127.0.0.1   TCP   44 80 → 54794 [ACK] Seq=304110 Ack=1953 Win=261836
31 16.557158   127.0.0.1   127.0.0.1   TCP   44 80 → 54794 [FIN, ACK] Seq=304110 Ack=1953 Win=2
32 16.557217   127.0.0.1   127.0.0.1   TCP   44 54794 → 80 [ACK] Seq=1953 Ack=304111 Win=261657
```

四次挥手的流程如下图所示：



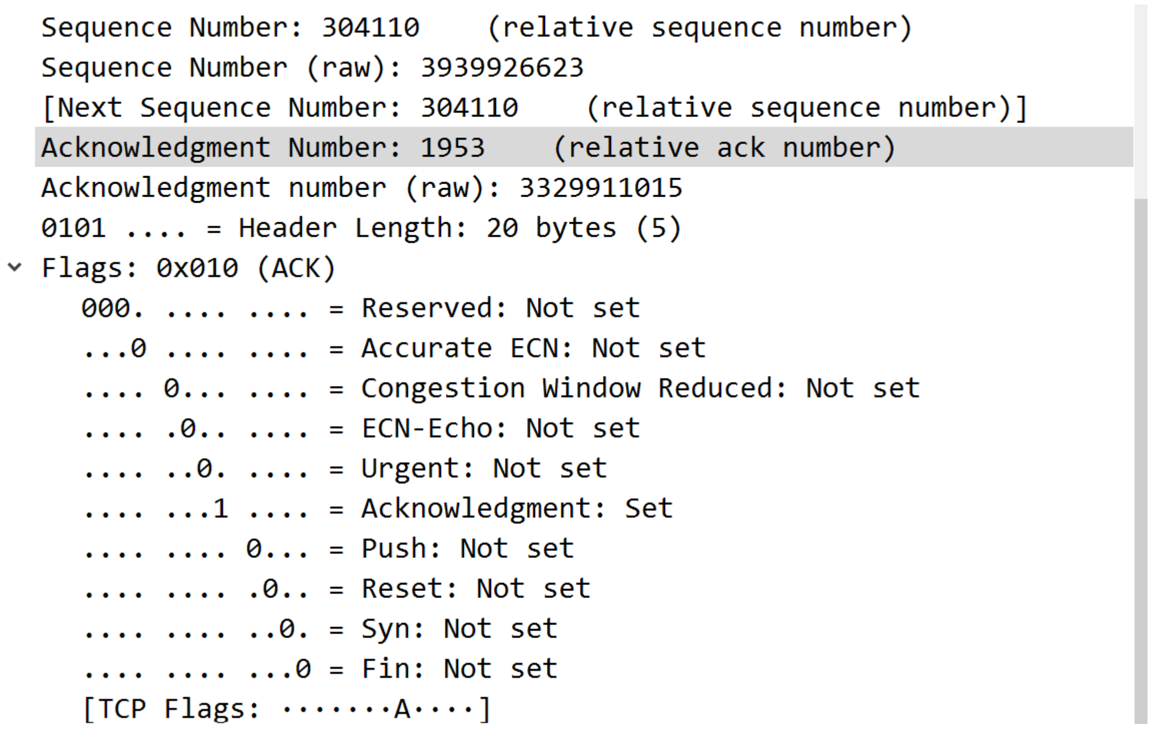
TCP四次挥手
客户端
服务端
客户端发送FIN+ACK报文，并置发送序号为X
FIN=1 ACK=1 Seq=X ack=Z
服务端发送ACK报文，并置发送序号为Z,确认序号为X+1
ACK=1 Seq=Z ack=X+1
服务端发送FIN+ACK报文，并置发送序号为Y,确认序号为X+1
FIN=1 ACK=1 Seq=Y ack=X+1
客户端发送ACK报文，并置发送序号为X+1,确认序号为Y+1
ACK=1 Seq = X+1 ack=Y+1

1. **首先客户端向服务器端发送请求断开连接的TCP报文。**FIN=1表示“请求关闭连接”；Seq为随机的U，相对值为1952。之后客户端进入FIN_WAIT_1状态。



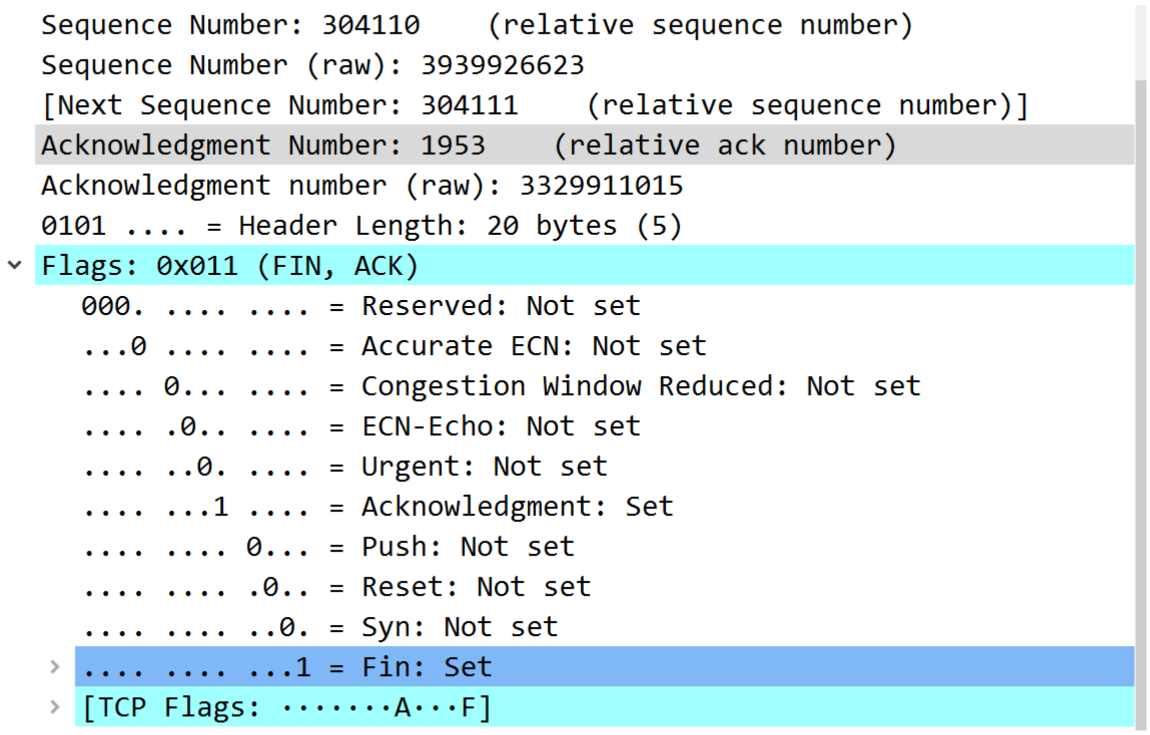
```
Sequence Number: 1952    (relative sequence number)
Sequence Number (raw): 3329911014
[Next Sequence Number: 1953    (relative sequence number)]
Acknowledgment Number: 304110    (relative ack number)
Acknowledgment number (raw): 3939926623
0101 .... = Header Length: 20 bytes (5)
Flags: 0x011 (FIN, ACK)
    000. .... .... = Reserved: Not set
    ...0 .... .... = Accurate ECN: Not set
    .... 0... .... = Congestion Window Reduced: Not set
    .... .0.. .... = ECN-Echo: Not set
    .... ..0. .... = Urgent: Not set
    .... ...1 .... = Acknowledgment: Set
    .... .... 0... = Push: Not set
    .... .... .0.. = Reset: Not set
    .... .... ..0. = Syn: Not set
    .... .... ...1 = Fin: Set
    [TCP Flags: ·······A···F]
```

2. **服务器端收到后，返回确认报文。**标志位ACK=1，表示服务器收到了客户端的释放连接请求；序列号Seq为随机的V，相对值为304110；确认号ACK=U+1，相对值为1953，表示收到了客户端的Seq，并+1作为确认，使得两边可以匹配成功。之后服务器端进入CLOSE_WAIT状态，表示已确认客户端的释放连接请求。客户端收到来自服务器的ACK应答报文段后，进入FIN_WAIT_2状态。此时TCP连接处于半关闭状态，客户端已不再向服务器发送内容，服务器还可以向客户端发送内容。



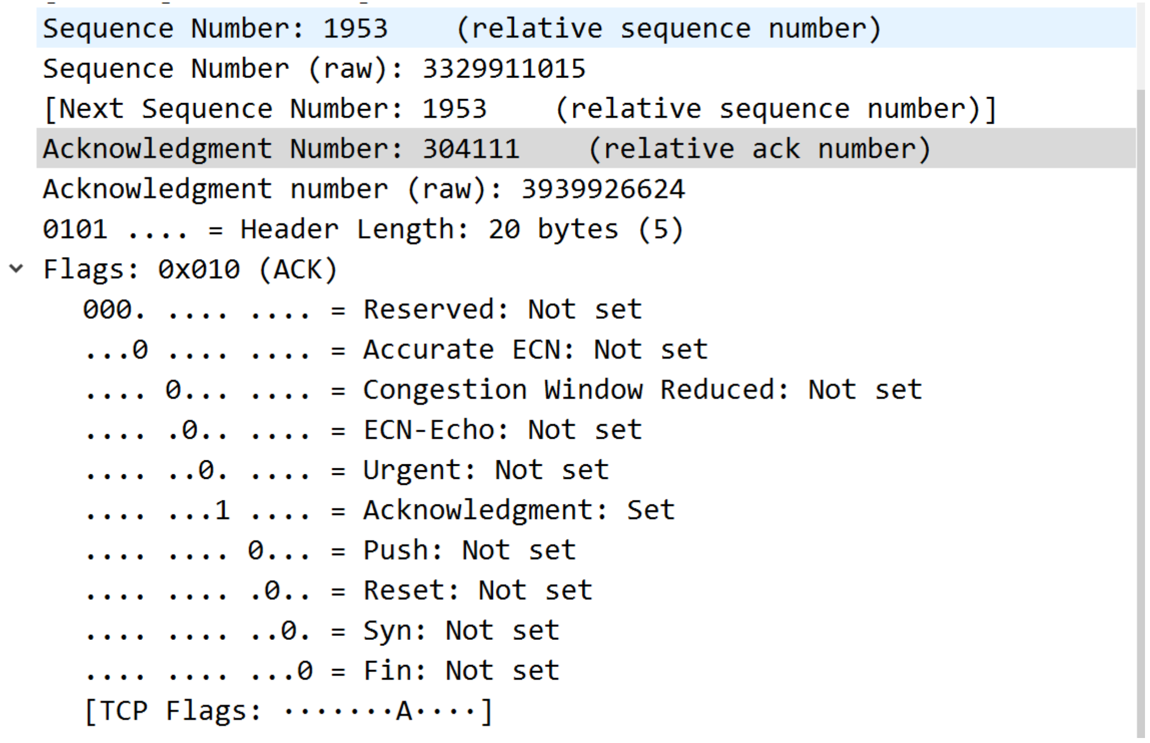
```
Sequence Number: 304110    (relative sequence number)
Sequence Number (raw): 3939926623
[Next Sequence Number: 304110    (relative sequence number)]
Acknowledgment Number: 1953    (relative ack number)
Acknowledgment number (raw): 3329911015
0101 .... = Header Length: 20 bytes (5)
Flags: 0x010 (ACK)
    000. .... .... = Reserved: Not set
    ...0 .... .... = Accurate ECN: Not set
    .... 0... .... = Congestion Window Reduced: Not set
    .... .0.. .... = ECN-Echo: Not set
    .... ..0. .... = Urgent: Not set
    .... ...1 .... = Acknowledgment: Set
    .... .... 0... = Push: Not set
    .... .... .0.. = Reset: Not set
    .... .... ..0. = Syn: Not set
    .... .... ...0 = Fin: Not set
    [TCP Flags: ·······A····]
```

3. **服务器向客户端发送连接释放报文。**标志位FIN=1表示服务器要释放连接；Seq为304110；ACK=1953。之后服务器进入LASK_ACK状态，等待客户端的确认。



```
Sequence Number: 304110    (relative sequence number)
Sequence Number (raw): 3939926623
[Next Sequence Number: 304111    (relative sequence number)]
Acknowledgment Number: 1953    (relative ack number)
Acknowledgment number (raw): 3329911015
0101 .... = Header Length: 20 bytes (5)
Flags: 0x011 (FIN, ACK)
   000. .... .... = Reserved: Not set
   ...0 .... .... = Accurate ECN: Not set
   .... 0... .... = Congestion Window Reduced: Not set
   .... .0.. .... = ECN-Echo: Not set
   .... ..0. .... = Urgent: Not set
   .... ...1 .... = Acknowledgment: Set
   .... .... 0... = Push: Not set
   .... .... .0.. = Reset: Not set
   .... .... ..0. = Syn: Not set
   .... .... ...1 = Fin: Set
   [TCP Flags: ·······A···F]
```

4. **客户端收到服务器的连接释放报文后，返回应答报文。**标志位ACK=1；Seq为服务器端FIN报文的ACK，相对值为1953;ACK为服务器端报文Seq+1，相对值为304111。之后客户端进入TIME_WAIT状态，服务器收到ACK应答报文段后，服务器就进入CLOSE状态，服务器的连接已经关闭。



```
Sequence Number: 1953    (relative sequence number)
Sequence Number (raw): 3329911015
[Next Sequence Number: 1953    (relative sequence number)]
Acknowledgment Number: 304111    (relative ack number)
Acknowledgment number (raw): 3939926624
0101 .... = Header Length: 20 bytes (5)
Flags: 0x010 (ACK)
   000. .... .... = Reserved: Not set
   ...0 .... .... = Accurate ECN: Not set
   .... 0... .... = Congestion Window Reduced: Not set
   .... .0.. .... = ECN-Echo: Not set
   .... ..0. .... = Urgent: Not set
   .... ...1 .... = Acknowledgment: Set
   .... .... 0... = Push: Not set
   .... .... .0.. = Reset: Not set
   .... .... ..0. = Syn: Not set
   .... .... ...0 = Fin: Not set
   [TCP Flags: ·······A····]
```

四次挥手的原因是：TCP允许半关闭状态。